Nimbusスケジューラ2仕様書

2010年5月24日

目次

[1. 概要 1](#_Toc453691577)

[2. インタフェース設計 3](#_Toc453691578)

[2.1. ScheduleMaster 3](#_Toc453691579)

[2.2. ScheduleMaker 7](#_Toc453691580)

[2.3. Schedule 8](#_Toc453691581)

[2.4. ScheduleManager 11](#_Toc453691582)

[2.5. Scheduler 11](#_Toc453691583)

[2.6. ScheduleExecuter 12](#_Toc453691584)

[3. 実装設計 13](#_Toc453691585)

[3.1. ScheduleMaker 13](#_Toc453691586)

[3.1.1. DefaultScheduleMakerService 13](#_Toc453691587)

[3.1.2. DatabaseConditionScheduleMakerService 14](#_Toc453691588)

[3.1.3. BeanFlowConditionScheduleMakerService 15](#_Toc453691589)

[3.2. ScheduleManager 16](#_Toc453691590)

[3.2.1. DefaultScheduleManagerService 16](#_Toc453691591)

[3.2.1. DatabaseScheduleManagerService 18](#_Toc453691592)

[3.3. Scheduler 31](#_Toc453691593)

[3.3.1. DefaultSchedulerService 31](#_Toc453691594)

[3.3.2. JMSSchedulerService 34](#_Toc453691595)

[3.4. ScheduleExecutor 37](#_Toc453691596)

[3.4.1. BeanFlowScheduleExecutorService 37](#_Toc453691597)

[3.4.2. CommandScheduleExecutorService 39](#_Toc453691598)

[3.4.1. ConcentrateScheduleExecutorService 41](#_Toc453691599)

[4. 分散構成 44](#_Toc453691600)

[5. クラスタ構成 45](#_Toc453691601)

[6. スケジュール管理画面 47](#_Toc453691602)

[6.1. 簡易画面 48](#_Toc453691603)

[6.2. リッチ管理画面 49](#_Toc453691604)

[6.2.1. スケジュール管理画面 49](#_Toc453691605)

[6.2.2. スケジュールマスタ管理画面 50](#_Toc453691606)

[6.2.3. スケジュール作成確認画面 51](#_Toc453691607)

[6.2.4. スケジュールチャート 52](#_Toc453691608)

# 概要

スケジュールをマスタ管理し、条件に応じた当日スケジュールを作成して、スケジュールの依存関係を解決しつつ実行する。スケジュールの状態は常に記録され、実行エラーはログ出力される。実行中のスケジュールの制御もサポートする。また、スケジュール実行を分散環境で行う構成や、クラスタ構成も可能である。

以下に、スケジューラの基本的なサービス構成の概要図を示す。

ScheduleMaster

Schedule

Schedule

ScheduleMaker

ScheduleManager

Scheduler

ScheduleControlListener

Schedule

Schedule

Schedule

Queue

ScheduleExecuter

ScheduleExecuter

ScheduleMasterは、日付の概念を持たないスケジュールのマスタ情報で、様々なスケジュールの属性を持つ。その属性の１つであるスケジュール種別で、ScheduleMakerとのマッピングを行う。

ScheduleMakerは、ScheduleMasterに一定の条件を加味しながらScheduleの作成を行う。一定の条件とは、例えば、今日が平日ならScheduleを生成するといったものである。また、1つのScheduleMasterから複数のScheduleを生成する場合がある。これは、ScheduleMasterに定期実行の属性がある場合である。

Scheduleは、スケジュールの実体であり、ScheduleMasterに準じた様々な属性を持つ。それに加えて、日付、状態等を併せ持つ。

ScheduleManagerは、ScheduleMasterを読み込み、スケジュール種別毎のScheduleMakerにScheduleの作成を依頼して、Scheduleを登録する。また、登録されたScheduleの状態管理、依存関係制御、監視及び実行状態制御を行う。実行中のScheduleに対して、実行状態制御が行われた場合は、Schedulerが実装しているScheduleControlListenerを経由して実行中のScheduleに命令する。

Schedulerは、定期的にSchedulerManagerに実行可能なScheduleの問い合わせを行い、実行可能なScheduleがあれば、ScheduleExecuterに実行を依頼する。Scheduleの問い合わせスレッドと、Scheduleを実行するScheduleExecuterの呼び出しスレッドの間には、何らかのQueueを持っているため、Shceduleの問い合わせと実行は非同期に行われる。また、Scheduleを実行するScheduleExecuterの呼び出しスレッドは、Schedulerがプールしており、通常複数のスレッドを待機させている。

　ScheduleExecuterは、Scheduleの実行を行い、Scheduleの状態遷移をScheduleManagerに通知する。また、実行中のScheduleの実行状態制御を行う。

　これらのインタフェースそれぞれに対して、実装クラスが提供されており、実装クラスが複数あるものでは、どのクラスを選択するかで機能が異なってくる。また、それぞれの実装クラスには、様々な属性があり、それらの設定によって、さらに機能が分化する。

以降では、各インタフェースの実装クラスと、それぞれの機能について説明していく。

JavaDocは、以下を参照。

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/scheduler2/package-summary.html>

サンプルコードは、以下のzipファイル内のscheduler2を参照。

<https://osdn.jp/projects/nimbus/downloads/64236/nimbus-sample-1.2.3.zip>

# インタフェース設計

## ScheduleMaster

スケジュールマスタのDTOインタフェースである。

実装クラスは、jp.ossc.nimbus.service.scheduler2.DefaultScheduleMasterのみであるが、このインスタンスをどのように生成するかは、ScheduleManagerの実装に依存する。

スケジュールマスタは、以下の属性を持つ。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 説明 | 備考 |
| 1 | マスタID | スケジュールマスタ内で一意なID | 必須 |
| 2 | 所属するマスタグループID | スケジュールマスタをグルーピングするマスタグループID。  グループに所属しない場合や、複数のグループに所属する場合がある。 | 0から複数 |
| 3 | スケジュールマスタに対する依存関係 | このスケジュールの開始条件に、他のスケジュールの終了が必要な場合、依存関係（ScheduleDepends）を持つ。  ここでいう「他のスケジュール」のマスタIDが、依存するマスタID（ScheduleDepends#getMasterId()）となる。  また、「スケジュールの終了」は、正常終了のみを条件とするか、正常または異常終了を条件とするかを、依存関係の異常無視フラグ（ScheduleDepends#isIgnoreError()）で選択することが可能である。異常無視フラグがtrueの場合は、異常終了も終了とみなす。  依存するスケジュールが存在しない場合や、複数のスケジュールに依存する場合がある。 | 0から複数 |
| 4 | グループ内のスケジュールマスタに対する依存関係 | 依存するスケジュールマスタをグループ内だけに限定して指定する。  依存するスケジュールが存在しない場合や、複数のスケジュールに依存する場合がある。 | 0から複数 |
| 5 | グループに対する依存関係 | １つのスケジュールに依存するのではなく、グルーピングされた複数のスケジュールに依存する事を指定する。  依存するグループが存在しない場合や、複数のグループに依存する場合がある。 | 0から複数 |
| 6 | 自身が所属するグループと他グループに対する依存関係 | 自身が所属するグループが他グループに依存する事を指定する。  依存するグループが存在しない場合や、複数のグループに依存する場合がある。 | 0から複数 |
| 7 | スケジュール種別 | ScheduleMakerと紐づけるスケジュール作成の条件名。  値は、任意に決めることが可能で、例えば「平日のみ」、「休日のみ」、「毎日」など。 | 任意 |
| 8 | タスク名 | スケジュールで実行するタスクの名前。  ScheduleExecutorが実行するタスクを識別する名前である。指定する内容は、ScheduleExecutorの実装に依存する。 | 必須 |
| 9 | 入力データ | スケジュールで実行するタスクに渡す引数。  記述フォーマットは、ScheduleExecutorや、ScheduleManagerの実装に依存する。 | 任意 |
| 10 | 開始時刻 | スケジュールの開始時刻を指定する。  記述フォーマットは、ScheduleManagerの実装に依存する。 | 必須 |
| 11 | 繰り返し実行間隔 | 開始時刻から繰り返し終了時刻まで、一定間隔で繰り返しスケジュールする場合の、時間間隔をミリ秒で指定する。 | 繰り返し実行をスケジュールする場合は、必須 |
| 12 | 繰り返し終了時刻 | 開始時刻から繰り返し終了時刻まで、一定間隔で繰り返しスケジュールする場合の、終了時刻を指定する。  記述フォーマットは、ScheduleManagerの実装に依存する。 | 繰り返し実行をスケジュールする場合は、必須 |
| 13 | リトライ間隔 | スケジュールの実行結果が、リトライを要求する限り、開始時刻からリトライ終了時刻まで、スケジュールを再スケジュールする場合の、時間間隔をミリ秒で指定する。 | リトライ実行をスケジュールする場合は、必須 |
| 14 | リトライ終了時刻 | スケジュールの実行結果が、リトライを要求する限り、開始時刻からリトライ終了時刻まで、スケジュールを再スケジュールする場合の、終了時刻を指定する。  終了時刻までに、正常終了できない場合は、異常終了とする。  記述フォーマットは、ScheduleManagerの実装に依存する。 | リトライ実行をスケジュールする場合は、必須 |
| 15 | 最大遅延時間 | スケジュールが予約された時の開始時刻（Schedule#getInitialTime()）から、許容され得る終了までの時間をミリ秒で指定する。 | 任意 |
| 16 | 有効／無効フラグ | スケジュールマスタを一時的に無効にするためのフラグ。  無効になったスケジュールマスタは、日次スケジュール作成（ScheduleManager#makeSchedule(Date)）の対象外となる。 | True/False |
| 17 | 実行種別 | どのScheduleExecutorで、スケジュールを実行するかを指定する。 | ScheduleExecutorが１つしか存在しない場合は、指定する必要はない。  複数存在する場合は、必須。 |
| 18 | 実行キー | 分散環境下で、ScheduleExecutorやSchedulerと紐づけるキー | 分散環境でない場合は、指定する必要はない。 |
| 19 | テンプレートフラグ | 任意のタイミングで必要に応じてスケジュールするための、スケジュールマスタであることを示すフラグ。  テンプレートフラグが有効なスケジュールは、日次スケジュール作成（ScheduleManager#makeSchedule(Date)）の対象外となり、動的スケジュール作成（ScheduleManager#makeSchedule(Date, ScheduleMaster)）のみ対象となる。 | True/False |

ScheduleMasterの時刻系の属性の指定パターンで、３種類のスケジュールマスタを表現できる。

|  |  |  |  |
| --- | --- | --- | --- |
| # | スケジュール分類 | 時刻指定パターン | 説明 |
| 1 | 通常 | 開始時刻 | 指定された時刻に1度だけ実行される。 |
| 2 | 繰り返し | 開始時刻  繰り返し間隔  繰り返し終了時刻 | 指定された開始時刻から繰り返し終了時刻まで、繰り返し間隔ずつの時間間隔をあけて、繰り返しスケジュールが実行される。  １つのスケジュールマスタから、複数のスケジュールが作成される。 |
| 3 | リトライ | 開始時刻  リトライ間隔  リトライ終了時刻 | 指定された開始時刻に実行され、スケジュールの実行結果がリトライを要求するものであれば、リトライ間隔後にスケジュールが移動し、再度実行される。  但し、実行結果がリトライを要求しても、次のリトライ時刻がリトライ終了時刻を過ぎるようであれば、スケジュールの状態を異常終了として、リトライは行わない。 |

ScheduleMasterの依存関係の属性の指定で、4種類の依存関係を制御可能である。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 依存関係 | 依存関係属性 | 説明 |
| 1 | １：１ | スケジュールマスタに対する依存関係 | スケジュールマスタに対する依存関係。  通常、１つのスケジュールマスタから１つのスケジュールが作成されるため、1：１の依存関係となる。  但し、繰り返しスケジュール、前日のスケジュール、動的スケジュールなど、同じスケジュールマスタIDを持つスケジュールが複数存在する場合がある。その場合は、１：多の依存関係となる。 |
| 2 | グループ内１：１ | グループ内のスケジュールマスタに対する依存関係 | 同一グループ内での、スケジュールマスタに対する依存関係。  同一グループ内という限定条件以外は、上述と同じ。 |
| 3 | １：多 | グループに対する依存関係 | グループマスタに対する依存関係。  通常、１つのグループには複数のスケジュールが所属するため、１：多の依存関係となる。  また、１つのグループマスタから複数のグループが作成される場合があるため、複数のグループに依存する場合もある。 |
| 4 | 多：多 | 自身が所属するグループと他グループに対する依存関係 | 自分が所属するグループとグループマスタの依存関係。  通常、１つのグループには複数のスケジュールが所属するため、多：多の依存関係となる。 |

## ScheduleMaker

スケジュールマスタからスケジュールを作成する機能のインタフェースである。

スケジュールマスタは、日付の概念を持たないため、指定された日付では必要のないスケジュールである場合がある。例えば、平日しか実行しないスケジュールである場合などである。そのような一定の条件を加味しながら、スケジュールの作成を行う。また、スケジュールマスタは、繰り返しの概念を持てるため、スケジュールマスタとスケジュールの関係は、1:nになる。

以下に、ScheduleMakerインタフェースの実装クラスの一覧を示す。

但し、実装クラス名のパッケージ名は、全てjp.ossc.nimbus.service.scheduler2なので省略する。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | DefaultScheduleMakerService | 条件機能を持たず、スケジュールマスタの属性通りに、スケジュールを作成するデフォルトの実装クラスである。 |
| 2 | DatabaseConditionScheduleMakerService | 任意のSQLを指定する事で、DB上の条件を使って、条件に該当する場合のみスケジュールを作成する。 |
| 3 | BeanFlowConditionScheduleMakerService | 任意のBeanFlowを呼び出し、その戻り値がtrueの場合のみスケジュールを作成する。最も汎用性が高い。 |

## Schedule

スケジュールのDTOインタフェースである。

実装クラスは、jp.ossc.nimbus.service.scheduler2.DefaultScheduleのみである。

スケジュールは、以下の属性を持つ。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 説明 | 備考 |
| 1 | スケジュールID | スケジュール内で一意なID | 必須 |
| 2 | マスタID | スケジュールマスタと紐づけるID | 必須 |
| 3 | 所属するグループID | グループ毎に一意なID。  マスタグループIDとグループIDのマッピングとして管理する。  グループに所属しない場合や、複数のグループに所属する場合がある。 | 0から複数 |
| 4 | スケジュールマスタに対する依存関係 | 依存するスケジュールマスタを依存関係（ScheduleDepends）で指定する。  依存するスケジュールが存在しない場合や、複数のスケジュールに依存する場合がある。 | 0から複数 |
| 5 | グループ内のスケジュールマスタに対する依存関係 | 依存するスケジュールマスタをグループ内だけに限定して指定する。  グループIDと依存関係のマッピングとして管理する。  依存するスケジュールが存在しない場合や、複数のスケジュールに依存する場合がある。 | 0から複数 |
| 6 | グループに対する依存関係 | １つのスケジュールに依存するのではなく、グルーピングされた複数のスケジュールに依存する事を指定する。  マスタグループIDと依存関係のマッピングとして管理する。  依存するグループが存在しない場合や、複数のグループに依存する場合がある。 | 0から複数 |
| 7 | 自身が所属するグループと他グループとの依存関係 | 自身が所属するグループが他グループに依存する事を指定する。  自身が所属するグループのグループIDと依存関係のマッピングとして管理する。  依存するグループが存在しない場合や、複数のグループに依存する場合がある。 | 0から複数 |
| 8 | タスク名 | スケジュールで実行するタスクの名前。  ScheduleExecutorが実行するタスクを識別する名前である。指定する内容は、ScheduleExecutorの実装に依存する。 | 必須 |
| 9 | 入力データ | スケジュールで実行するタスクに渡す引数。  記述フォーマットは、ScheduleExecutorや、ScheduleManagerの実装に依存する。 | 任意 |
| 10 | 出力データ | スケジュールの実行結果。 | 実行後に、出力 |
| 11 | 初期スケジュール日時 | スケジュールが作成された時点での、開始予定日時。  遅延監視時に、スケジュール開始時刻とみなされ、この時刻+最大遅延時間が示す時刻までに、スケジュールが正常終了していない場合は、遅延監視エラーとなる。 | 必須 |
| 12 | スケジュール日時 | スケジュールの開始予定日時。  リトライや、手動でのスケジュールのリスケジュールにより、変更される場合がある。  ScheduleManagerが、実行すべきスケジュールを検索する時に、使用する属性となる。 | 必須 |
| 13 | 実行開始時刻 | 直近でスケジュールが実行された開始時刻 | 実行開始時に、出力 |
| 14 | 実行終了時刻 | 直近でスケジュールが実行された終了時刻 | 実行後に、出力 |
| 15 | 最大遅延時間 | スケジュールが予約された時の開始時刻（Schedule#getInitialTime()）から、許容され得る終了までの時間をミリ秒で指定する。 | 任意 |
| 16 | リトライ間隔 | スケジュールの実行結果が、リトライを要求する限り、開始時刻からリトライ終了時刻まで、スケジュールを再スケジュールする場合の、時間間隔をミリ秒で指定する。 | リトライ実行をスケジュールする場合は、必須 |
| 17 | リトライ終了時刻 | スケジュールの実行結果が、リトライを要求する限り、開始時刻からリトライ終了時刻まで、スケジュールを再スケジュールする場合の、終了時刻を指定する。  終了時刻までに、正常終了できない場合は、異常終了とする。  記述フォーマットは、ScheduleManagerの実装に依存する。 | リトライ実行をスケジュールする場合は、必須 |
| 18 | 状態 | スケジュールの状態  初期→投入→実行中→正常終了または異常終了またはリトライ  と遷移する。  スケジュール制御機能を利用した場合、一時停止または中断となる場合がある。  また、手動で無効化の状態にする場合がある。 | 必須 |
| 19 | 制御状態 | スケジュールの制御機能を利用する場合に、遷移させたい状態を設定する。  初期→中断または一時停止  一時停止→再開  また、制御に失敗した場合に、制御失敗の状態になる場合がある。 | 任意 |
| 20 | 監視状態 | スケジュールの監視機能を利用する場合に、監視結果が出力される。  初期→遅延時間オーバー | 監視結果が出力される |
| 21 | 実行種別 | どのScheduleExecutorで、スケジュールを実行するかを指定する。 | ScheduleExecutorが１つしか存在しない場合は、指定する必要はない。  複数存在する場合は、必須。 |
| 22 | 実行キー | 分散環境下で、ScheduleExecutorやSchedulerと紐づけるキー | 分散環境でない場合は、指定する必要はない。 |
| 23 | リトライフラグ | 実行されたタスク側から、ScheduleExecutorに対してリトライを要求するためのフラグ | True/false |

## ScheduleManager

スケジュールを管理する機能のインタフェースである。

ScheduleManagerは、主に以下の機能を持つ。

* スケジュールマスタの読み込み
* スケジュールマスタから日次スケジュールの作成（ScheduleMakerへ依頼）
* スケジュールマスタから動的スケジュールの作成（ScheduleMakerへ依頼）
* スケジュールの登録・削除
* スケジュールの検索
* スケジュールの依存関係制御
* スケジュールの状態管理
* スケジュールの実行状態制御
* スケジュールの監視

以下に、ScheduleManagerインタフェースの実装クラスの一覧を示す。

但し、実装クラス名のパッケージ名は、全てjp.ossc.nimbus.service.scheduler2なので省略する。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | DefaultScheduleManagerService | スケジュール管理をメモリ上で行うデフォルトの実装クラスである。  スケジュールの状態管理は、設定によりファイルに永続化する事ができる。 |
| 2 | DatabaseScheduleManagerService | スケジュール管理をDB上で行う実装クラスである。 |

## Scheduler

スケジュールを制御する機能のインタフェースである。

スケジューラは、ScheduleManagerを使って、定期的に実行すべきスケジュール（スケジュール時刻が現在時刻を過ぎており、依存関係も解決されているもの）の検索を行い、実行すべきスケジュールがあれば、非同期にScheduleExecuterに実行を依頼する。

実行可能なスケジュールの監視と、スケジュールの実行を非同期に行うために、スケジュール監視スレッドと、スケジュール実行スレッドを別々に持つ。また、スケジュールの実行は、複数のタスクを並列に行う必要もあるため、複数のスレッドをプールして管理する。

スケジュール監視スレッドと、スケジュール実行スレッドの間には、何らかのキューを持ち、監視と実行を非同期に行う。どのようなキュー実装を使うかは、実装に依存する。

以下に、Schedulerインタフェースの実装クラスの一覧を示す。

但し、実装クラス名のパッケージ名は、全てjp.ossc.nimbus.service.scheduler2なので省略する。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | DefaultSchedulerService | スケジュールの非同期実行に、Nimbus Queueを使用するデフォルトの実装クラスである。 |
| 2 | JMSSchedulerService | スケジュールの非同期実行に、JMS Queueを使用する実装クラスである。 |

## ScheduleExecuter

スケジュール実行機能のインタフェースである。

Scheduleに該当する処理を実行し、都度スケジュールの状態変更をScheduleManagerに通知する。また、ScheduleManagerから実行中のスケジュールに対して、実行状態の制御（中断、再開、中止など）を依頼されると、その制御を行う。

Scheduleをどのように解釈し、実行するかは、実装に依存する。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | BeanFlowScheduleExecuterService | タスク名をBeanFlow名として、BeanFlowを実行する。BeanFlowの実行引数として、Scheduleオブジェクトを渡す。 |
| 2 | CommandScheduleExecutorService | タスク名をコマンド、入力データをコマンドへの引数とみなして、バッチやシェルを別プロセスで実行する。 |
| 3 | ConcentrateScheduleExecutorService | スケジュールの入力データを集配信リクエスト文字列として解析して、集配信リクエストオブジェクトを生成し、タスク名をBeanFlow名として、集配信BeanFlowを実行する。 |

# 実装設計

インタフェース設計で示した各インタフェースのうち、機能性のあるインタフェースの実装クラスの機能を説明する。

## ScheduleMaker

### DefaultScheduleMakerService

ScheduleMakerインタフェースの基本的な機能を実装するデフォルト実装クラスである。

ScheduleMasterの属性のチェックを行い、問題なければScheduleMasterの属性に従って、Scheduleの作成を行う。従って、スケジュールの作成条件は、チェックで問題がないことのみとなる。

ScheduleMaster

Schedule

Schedule

スケジュール作成

**DefaultScheduleMakerService**

ScheduleMasterの属性のチェックでは、以下のチェックを行う。

* スケジュールマスタID（ScheduleMaster #getId()）の必須チェック
* タスク名（ScheduleMaster #getTaskName()）の必須チェック
* 開始時刻（ScheduleMaster #getStartTime()）の必須チェック
* 終了時刻（ScheduleMaster #getEndTime()）が指定されている場合は、開始時刻よりも前の時刻でないことのチェック
* 開始時刻、終了時刻の両方が指定されていて、終了時刻が開始時刻の後である場合は、繰り返し間隔（ScheduleMaster #getRepeatInterval()）に有効な値（>0）が設定さていることのチェック
* リトライ終了時刻（ScheduleMaster #getRetryEndTime ()）が指定されている場合は、開始時刻よりも後の時刻であることのチェック
* リトライ終了時刻指定されている場合は、リトライ間隔（ScheduleMaster # getRetryInterval ()）に有効な値（>0）が設定されていることのチェック
* リトライ終了時刻、リトライ間隔が指定されている場合は、開始時刻からリトライ終了時刻の間に、リトライ可能な時間間隔が存在することのチェック

ScheduleMasterに繰り返し間隔と繰り返し終了時刻が指定されている場合は、１つのScheduleMasterから、複数のScheduleを作成する。

### DatabaseConditionScheduleMakerService

DefaultScheduleMakerServiceのサブクラスであり、親クラスの機能に加えて、スケジュールの作成条件をSQLで指定する事で、データベースの状態で制御する事ができる。

例えば、業務日テーブルに平日、休日の区分があれば、平日のみスケジュールが作成されるようにしたり、休日のみスケジュールが作成されるようにしたりする事ができる。

ScheduleMaster

ConnectionFactory

スケジュール作成

JDBC

Connection

**DatabaseCondition**

**ScheduleMakerService**

Schedule

条件判定

SQL

Schedule

データベース

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ConnectionFactoryServiceName | ServiceName | JDBC接続を取得するjp.ossc.nimbus.service.connection.ConnectionFactoryインタフェースを実装したサービスのサービス名を設定する。 |
| 2 | Query | String | スケジュールの作成有無を判断するSQLを設定する。  必ず、スケジュール作成日付を埋め込みパラメータとして持ち、結果がBoolean型、数値型、文字列型のいずれかになるSQLとする事。  Boolean型の場合は、trueの場合、スケジュールを作成する。  数値型の場合は、0以外の値の場合、スケジュールを作成する。  文字列型の場合は、"0"以外の値の場合、スケジュールを作成する。  例：営業日であればスケジュールを作成する場合  select count(1) from businessday\_calendar where date = ? |
| 3 | DateFormat | String | 日付フォーマットを設定する。Query属性で指定したSQL内に、パラメータとして埋め込むスケジュール作成日付を文字列として渡したい場合に、その日付フォーマットを指定する。 |

### BeanFlowConditionScheduleMakerService

DefaultScheduleMakerServiceのサブクラスであり、親クラスの機能に加えて、スケジュールの作成条件をNimbus BeanFlowで指定する事で、任意の条件判定を実装する事ができる。

BeanFlow

定義XML

BeanFlowInvokerFactory

ScheduleMaster

BeanFlowInvoker

スケジュール作成

Schedule

Schedule

**BeanFlowCondition**

**ScheduleMakerService**

条件判定

BeanFlow

定義XML

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | BeanFlowInvokerFactoryServiceName | ServiceName | 業務フローを取得するjp.ossc.nimbus.service.beancontrol.interfaces.BeanFlowInvokerFactoryインタフェースを実装したサービスのサービス名を設定する。 |
| ２ | FlowName | String | 業務フロー名を設定する。 |

## ScheduleManager

### DefaultScheduleManagerService

ScheduleManagerインタフェースの簡易な実装を提供するデフォルト実装クラスである。

ScheduleMasterをサービス定義で登録し、生成されたScheduleはメモリ中で管理する。

JVMを停止した際に、スケジュールの状態を保存しておくために、ディスク上にファイルとしてスケジュールの状態を保存する機能もある。

また、内部に遅延監視デーモンを持ち、スケジュールの遅延監視を定期的に行い、遅延したスケジュールを見つけると、エラーログを出力する。

実行中のスケジュールに対する制御要求を受け付けると、制御要求をScheduleControlListenerに通知する。

ScheduleControlListener

Schedule

**DefaultScheduleManagerService**

制御要求の通知

ScheduleMaker

状態保存

ScheduleMaster

サービス定義

XML

スケジュール

状態保存ファイル

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleMasterServiceNames | ServiceName[] | スケジュールマスタとして登録するjp.ossc.nimbus.service.scheduler2.ScheduleMasterインタフェースを実装したクラスのサービス名を設定する。 |
| 2 | ScheduleMakerTypeMapping | Properties | スケジュール種別とjp.ossc.nimbus.service.scheduler2.ScheduleMakerインタフェースを実装したサービスのサービス名のマッピングを設定する。 |
| 3 | DefaultScheduleMakerServiceName | ServiceName | ScheduleMakerTypeMapping属性で指定されていないスケジュール種別のScheduleMasterの場合に、使用するjp.ossc.nimbus.service.scheduler2.ScheduleMakerインタフェースを実装したサービスのサービス名 |
| 4 | ScheduleMakerTypeRegexEnabled | boolean | ScheduleMakerTypeMappingで設定したスケジュール種別を正規表現として評価するかどうかを設定する。  デフォルトは、false |
| 5 | SequenceServiceName | ServiceName | スケジュールIDを発番するjp.ossc.nimbus.service.sequence.Sequenceサービスのサービス名を設定する。 |
| 6 | MakeScheduleOnStart | boolean | サービスの開始時に、日次スケジュール作成を行うかどうかを設定する。但し、当日のスケジュールが作成されている場合は、作成しない。  デフォルトは、true |
| 7 | PersistDir | String | スケジュールの状態をファイルとして保存するディレクトリのパスを設定する。 |
| 8 | TimeoverCheckInterval | long | 最大遅延時間の監視を行う間隔[ms]を設定する。  デフォルトは、1000[ms] |

### DatabaseScheduleManagerService

ScheduleManagerインタフェースのデータベースを使った実装クラスである。

ScheduleMasterをデータベースのテーブルに登録し、作成したScheduleもデータベースのテーブルで管理する。

使用するテーブルは、あらかじめ作成しておく必要がある。列の型は、列毎に仕様として決まっているが、テーブル名や列名などは、デフォルト値は決まっているものの、サービス定義で変更可能なため、自由である。

また、内部に遅延監視デーモンと制御状態監視デーモンを持つ。遅延監視デーモンは、データベース上のスケジュールの遅延監視を定期的に行い、遅延したスケジュールを見つけると、エラーログを出力する。制御状態監視デーモンは、データベース上の制御状態の値を監視し、実行中のスケジュールに対する制御要求があれば、制御要求をScheduleControlListenerに通知する。

ScheduleControlListener

制御要求の通知

**DatabaseScheduleManagerService**

ConnectionFactory

JDBC

Connection

ScheduleMaker

Schedule

ScheduleMaster

データベース

以下にテーブルの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 論理テーブル名 | 説明 |
| 1 | スケジュールマスタテーブル | ScheduleMasterを登録する |
| 2 | スケジュールグループマスタテーブル | ScheduleMasterのグルーピングを登録する |
| 3 | スケジュール依存関係マスタテーブル | ScheduleMasterの依存関係を登録する |
| 4 | スケジュールグループ依存関係マスタテーブル | ScheduleMasterのグループ依存関係を登録する |
| 5 | スケジュールテーブル | 作成されたScheduleを管理する |
| 6 | スケジュールグループテーブル | 作成されたScheduleのグループを管理する |
| 7 | スケジュール依存関係テーブル | 作成されたScheduleの依存関係を管理する |
| 8 | スケジュールグループ依存関係テーブル | 作成されたScheduleのグループ依存関係を管理する |

各テーブルのスキーマを示す。テーブル名、列名ともに、物理的な名前は、変更可能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールマスタテーブル  /SCHEDULE\_MST | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | スケジュールマスタID  /ID | VARCHAR | **主キー** | 値の説明は、2.1を参照 |
| 2 | タスク名  /TASK\_NAME | VARCHAR | **NOT NULL** | 値の説明は、2.1を参照 |
| 3 | スケジュール種別  /SCHEDULE\_TYPE | VARCHAR |  | 値の説明は、2.1を参照 |
| 4 | 入力  /INPUT | VARCHAR  CLOB  BLOB |  | 列の型は、ScheduleMasterTableSchema.inputColumnType属性で指定可能で、デフォルトは、VARCHAR。  値の説明は、2.1を参照 |
| 5 | 開始時刻  /START\_TIME | CHAR |  | フォーマットは、TimeFormat属性で設定可能で、デフォルトは、HHmmssSSS。  値の説明は、2.1を参照 |
| 6 | 終了時刻  /END\_TIME | CHAR |  | フォーマットは、TimeFormat属性で設定可能で、デフォルトは、HHmmssSSS。  値の説明は、2.1を参照 |
| 7 | 繰り返し間隔  /REPEAT\_INTERVAL | NUMERIC |  | 値の説明は、2.1を参照 |
| 8 | リトライ間隔  /RETRY\_INTERVAL | NUMERIC |  | 値の説明は、2.1を参照 |
| 9 | リトライ終了時刻  /RETRY\_END\_TIME | CHAR |  | フォーマットは、TimeFormat属性で設定可能で、デフォルトは、HHmmssSSS。  値の説明は、2.1を参照 |
| 10 | 最大遅延時間  /MAX\_DELAY\_TIME | NUMERIC |  | 値の説明は、2.1を参照 |
| 11 | 有効フラグ  /ENABLE | CHAR | **NOT NULL** | 1か0で、有効な場合1。  値の説明は、2.1を参照 |
| 12 | 実行キー  /EXECUTOR\_KEY | VARCHAR |  | 値の説明は、2.1を参照 |
| 13 | 実行種類  /EXECUTOR\_TYPE | VARCHAR |  | 値の説明は、2.1を参照 |
| 14 | テンプレート  /TEMPLATE | CHAR | **NOT NULL** | 1か0で、テンプレートな場合1。  値の説明は、2.1を参照 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールグループマスタテーブル  /SCHEDULE\_GROUP\_MST | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | スケジュールマスタID  /ID | VARCHAR | **主キー** | スケジュールマスタテーブルのスケジュールマスタID |
| 2 | グループID  /GROUP\_ID | VARCHAR | **主キー** | スケジュールマスタが所属するグループID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュール依存関係マスタテーブル  /SCHEDULE\_DEPENDS\_MST | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | 依存されるスケジュールマスタID  /ID | VARCHAR | **NOT NULL** | スケジュールマスタテーブルのスケジュールマスタID |
| 2 | 依存するスケジュールマスタID  /DEPENDS\_ID | VARCHAR |  | スケジュールマスタテーブルのスケジュールマスタID。  スケジュールとスケジュールマスタの依存関係を定義する際、または同一スケジュールグループ内でのスケジュールとスケジュールマスタの依存関係を定義する際に使用する。 |
| 3 | 依存するスケジュールグループマスタID  /DEPENDS\_GROUP\_ID | VARCHAR |  | スケジュールグループマスタテーブルのグループID。  スケジュールとスケジュールグループの依存関係を定義する際に使用する。 |
| 4 | グループ内依存を示す場合のスケジュールグループマスタID  /GROUP\_ID | VARCHAR |  | スケジュールグループマスタテーブルのグループID。  同一スケジュールグループ内でのスケジュールとスケジュールマスタの依存関係を定義する際に使用する。 |
| 5 | エラー無視  /IGNORE\_ERROR | CHAR |  | 1か0で、異常終了のスケジュールを終了したとみなす場合1。 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールグループ依存関係マスタテーブル  /SCHEDULE\_GROUP\_DEPENDS\_MST | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | 依存されるスケジュールマスタグループID  /GROUP\_ID | VARCHAR | **主キー** | スケジュールグループマスタテーブルのグループID |
| 2 | 依存するスケジュールマスタグループID  /DEPENDS\_GROUP\_ID | VARCHAR | **主キー** | スケジュールグループマスタテーブルのグループID。  スケジュールグループとスケジュールグループの依存関係を定義する際に使用する。 |
| 3 | エラー無視  /IGNORE\_ERROR | CHAR |  | 1か0で、異常終了のスケジュールを終了したとみなす場合1。 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールテーブル  /SCHEDULE | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | スケジュールID  /ID | VARCHAR | **主キー** | 値の説明は、2.3を参照 |
| 2 | スケジュールマスタID  /MASTER\_ID | VARCHAR | **NOT NULL** | スケジュールマスタテーブルのスケジュールマスタID。  値の説明は、2.3を参照 |
| 3 | スケジュール日付  /S\_DATE | CHAR | **NOT NULL** | スケジュール日付と開始時刻を連結して、Scheduleの開始日時とする。  フォーマットは、DateFormat属性で設定可能で、デフォルトは、yyyyMMdd。  値の説明は、2.3を参照 |
| 4 | 開始時刻  /S\_TIME | CHAR | **NOT NULL** | スケジュール日付と開始時刻を連結して、Scheduleの開始日時とする。  フォーマットは、TimeFormat属性で設定可能で、デフォルトは、HHmmssSSS。  値の説明は、2.3を参照 |
| 5 | 初期スケジュール日付  /INITIAL\_DATE | CHAR | **NOT NULL** | 初期スケジュール日付と初期開始時刻を連結して、Scheduleの初期開始日時とする。  フォーマットは、DateFormat属性で設定可能で、デフォルトは、yyyyMMdd。  値の説明は、2.3を参照 |
| 6 | 初期開始時刻  /INITIAL\_TIME | CHAR | **NOT NULL** | 初期スケジュール日付と初期開始時刻を連結して、Scheduleの初期開始日時とする。  フォーマットは、TimeFormat属性で設定可能で、デフォルトは、HHmmssSSS。  値の説明は、2.3を参照 |
| 7 | タスク名  /TASK\_NAME | VARCHAR | **NOT NULL** | 値の説明は、2.3を参照 |
| 8 | 入力  /INPUT | VARCHAR  CLOB  BLOB |  | 列の型は、ScheduleTableSchema.inputColumnType属性で指定可能で、デフォルトは、VARCHAR。  値の説明は、2.3を参照 |
| 9 | 出力  /OUTPUT | VARCHAR  CLOB  BLOB |  | 列の型は、ScheduleTableSchema.outputColumnType属性で指定可能で、デフォルトは、VARCHAR。  値の説明は、2.3を参照 |
| 10 | リトライ間隔  /RETRY\_INTERVAL | NUMERIC |  | 値の説明は、2.3を参照 |
| 11 | リトライ終了時刻  /RETRY\_END\_TIME | CHAR |  | フォーマットは、DateFormat属性とTimeFormat属性で設定可能で、デフォルトは、yyyyMMddHHmmssSSS。  値の説明は、2.3を参照 |
| 12 | 最大遅延時間  /MAX\_DELAY\_TIME | NUMERIC |  | 値の説明は、2.3を参照 |
| 13 | 状態  /STATE | CHAR | **NOT NULL** | 以下の状態：値を取りうる。  初期：I  投入：E  実行中：R  正常終了：N  異常終了：F  中断中：P  強制終了：A  リトライ中：T  無効化：D  各状態の値は、デフォルト値が上記のように決まっているが、サービス定義で変更可能である。  値の説明は、2.3を参照 |
| 14 | 制御状態  /CONTROL\_STATE | CHAR | **NOT NULL** | 以下の状態：値を取りうる。  初期：I  中断：P  再開：R  強制終了：A  制御失敗：F  各状態の値は、デフォルト値が上記のように決まっているが、サービス定義で変更可能である。  値の説明は、2.3を参照 |
| 15 | チェック状態  /CHECK\_STATE | CHAR | **NOT NULL** | 以下の状態：値を取りうる。  初期：I  タイムオーバー：O  各状態の値は、デフォルト値が上記のように決まっているが、サービス定義で変更可能である。  値の説明は、2.3を参照 |
| 16 | 実行キー  /EXECUTOR\_KEY | VARCHAR |  | 値の説明は、2.3を参照 |
| 17 | 実行種類  /EXECUTOR\_TYPE | VARCHAR |  | 値の説明は、2.3を参照 |
| 18 | 実行開始日時  /EXEC\_S\_TIME | CHAR |  | フォーマットは、DateFormat属性とTimeFormat属性で設定可能で、デフォルトは、yyyyMMddHHmmssSSS。  値の説明は、2.3を参照 |
| 19 | 実行終了日時  /EXEC\_E\_TIME | CHAR |  | フォーマットは、DateFormat属性とTimeFormat属性で設定可能で、デフォルトは、yyyyMMddHHmmssSSS。  値の説明は、2.3を参照 |
| 20 | レコード更新バージョン番号  /ROWVERSION | INTEGER |  | レコードの更新回数を示す |
| 21 | レコード更新ユーザID  /UPDATEUSERID | VARCHAR |  | レコードの更新ユーザを示す  UpdateUserId属性で設定できる。指定しない場合は、ホスト名。 |
| 22 | レコード更新日時  /UPDATETIME | TIMESTAMP |  | レコードの更新日時を示す |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールグループテーブル  /SCHEDULE\_GROUP | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | スケジュールID  /ID | VARCHAR | **主キー** | スケジュールテーブルのスケジュールID |
| 2 | グループID  /GROUP\_ID | VARCHAR | **主キー** | スケジュールが所属するグループID |
| 3 | マスタグループID  /MASTER\_GROUP\_ID | VARCHAR | **NOT NULL** | スケジュールマスタが所属するスケジュールグループマスタのグループID |
| 4 | レコード更新バージョン番号  /ROWVERSION | INTEGER |  | レコードの更新回数を示す |
| 5 | レコード更新ユーザID  /UPDATEUSERID | VARCHAR |  | レコードの更新ユーザを示す  UpdateUserId属性で設定できる。指定しない場合は、ホスト名。 |
| 6 | レコード更新日時  /UPDATETIME | TIMESTAMP |  | レコードの更新日時を示す |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュール依存関係テーブル  /SCHEDULE\_DEPENDS | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | 依存されるスケジュールID  /ID | VARCHAR | **NOT NULL** | スケジュールテーブルのスケジュールID |
| 2 | 依存するスケジュールマスタID  /DEPENDS\_ID | VARCHAR |  | スケジュールマスタテーブルのスケジュールマスタID。 |
| 3 | 依存するスケジュールグループマスタID  /DEPENDS\_GROUP\_ID | VARCHAR |  | スケジュールグループマスタテーブルのグループID。 |
| 4 | グループ内依存を示す場合のスケジュールグループID  /GROUP\_ID | VARCHAR |  | スケジュールグループテーブルのグループID。 |
| 5 | エラー無視  /IGNORE\_ERROR | CHAR |  | 1か0で、異常終了のスケジュールを終了したとみなす場合1。 |
| 6 | レコード更新バージョン番号  /ROWVERSION | INTEGER |  | レコードの更新回数を示す |
| 7 | レコード更新ユーザID  /UPDATEUSERID | VARCHAR |  | レコードの更新ユーザを示す  UpdateUserId属性で設定できる。指定しない場合は、ホスト名。 |
| 8 | レコード更新日時  /UPDATETIME | TIMESTAMP |  | レコードの更新日時を示す |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 論理テーブル名  /物理テーブル名 | | | スケジュールグループ依存関係テーブル  /SCHEDULE\_GROUP\_DEPENDS | |
| # | 論理列名  /物理列名 | 型 | 制約 | 説明 |
| 1 | 依存されるスケジュールグループID  /GROUP\_ID | VARCHAR | **主キー** | スケジュールグループテーブルのグループID |
| 2 | 依存するスケジュールマスタグループID  /DEPENDS\_GROUP\_ID | VARCHAR | **主キー** | スケジュールグループマスタテーブルのグループID。 |
| 3 | エラー無視  /IGNORE\_ERROR | CHAR |  | 1か0で、異常終了のスケジュールを終了したとみなす場合1。 |
| 4 | レコード更新バージョン番号  /ROWVERSION | INTEGER |  | レコードの更新回数を示す |
| 5 | レコード更新ユーザID  /UPDATEUSERID | VARCHAR |  | レコードの更新ユーザを示す  UpdateUserId属性で設定できる。指定しない場合は、ホスト名。 |
| 6 | レコード更新日時  /UPDATETIME | TIMESTAMP |  | レコードの更新日時を示す |

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleMakerTypeMapping | Properties | スケジュール種別とjp.ossc.nimbus.service.scheduler2.ScheduleMakerインタフェースを実装したサービスのサービス名のマッピングを設定する。 |
| 2 | DefaultScheduleMakerServiceName | ServiceName | ScheduleMakerTypeMapping属性で指定されていないスケジュール種別のScheduleMasterの場合に、使用するjp.ossc.nimbus.service.scheduler2.ScheduleMakerインタフェースを実装したサービスのサービス名 |
| 3 | ScheduleMakerTypeRegexEnabled | boolean | ScheduleMakerTypeMappingで設定したスケジュール種別を正規表現として評価するかどうかを設定する。  デフォルトは、false |
| 4 | ConnectionFactoryServiceName | ServiceName | JDBCコネクションを取得するjp.ossc.nimbus.service.connection.ConnectionFactoryサービスのサービス名を設定する。 |
| 5 | DateFormat | String | データベースの日付系カラムのフォーマットを設定する。  デフォルトは、yyyyMMdd |
| 6 | TimeFormat | String | データベースの時刻系カラムのフォーマットを設定する。  デフォルトは、HHmmssSSS |
| 7 | UpdateUserId | String | データベースの「更新ユーザ」カラムの値を設定する。  デフォルトは、ホスト名。 |
| 8 | NextScheduleIdSelectQuery | String | スケジュールIDを発番するSQLを設定する。 |
| 9 | SequenceServiceName | ServiceName | スケジュールIDを発番するjp.ossc.nimbus.service.sequence.Sequenceサービスのサービス名を設定する。 |
| 10 | ControlStateCheckInterval | long | 制御状態をチェックする間隔[ms]を設定する。  デフォルトは、1000[ms]。 |
| 11 | TimeoverCheckInterval | long | 最大遅延時間の監視を行う間隔[ms]を設定する。  デフォルトは、1000[ms] |
| 12 | MakeScheduleOnStart | boolean | サービスの開始時に、日次スケジュール作成を行うかどうかを設定する。但し、当日のスケジュールが作成されている場合は、作成しない。  デフォルトは、true |
| 13 | LockForFindExecutable | boolean | 実行可能なスケジュールを検索するときに、行ロックをかけるかどうかを設定する。  デフォルトは、false。 |
| 14 | ClusterServiceName | ServiceName | ScheduleManagerの監視系デーモンをクラスタ化するjp.ossc.nimbus.service.keepalive.ClusrterServiceのサービス名を設定する。 |
| 15 | UseConcatFunction | boolean | 内部で使用するSQLで文字列連結にCONCAT関数を使うかどうかを設定する。  デフォルトは、falseで、"||"で連結する。 |
| 16 | JSONInput | boolean | スケジュールの入力のJSONフォーマットをサポートするかどうかを設定する。  デフォルトは、falseで、スケジュールの入力は単なる文字列として扱う。 |
| 17 | ScheduleMasterTableSchema.table | String | スケジュールマスタテーブルのテーブル名を設定する。 |
| 18 | ScheduleMasterTableSchema.カラム名プロパティ | String | スケジュールマスタテーブルの各カラム名を設定する。 |
| 19 | ScheduleMasterTableSchema.inputColumnType | int | スケジュールマスタテーブルの「入力」カラムのJDBC型を設定する。  デフォルトは、java.sq.Types.VARCHAR。 |
| 20 | ScheduleGroupMasterTableSchema.table | String | スケジュールグループマスタテーブルのテーブル名を設定する。 |
| 21 | ScheduleGroupMasterTableSchema.カラム名プロパティ | String | スケジュールグループマスタテーブルの各カラム名を設定する。 |
| 22 | ScheduleDependsMasterTableSchema.table | String | スケジュール依存関係マスタテーブルのテーブル名を設定する。 |
| 23 | ScheduleDependsMasterTableSchema.カラム名プロパティ | String | スケジュール依存関係マスタテーブルの各カラム名を設定する。 |
| 24 | ScheduleGroupDependsMasterTableSchema.table | String | スケジュールグループ依存関係マスタテーブルのテーブル名を設定する。 |
| 25 | ScheduleGroupDependsMasterTableSchema.カラム名プロパティ | String | スケジュールグループ依存関係マスタテーブルの各カラム名を設定する。 |
| 26 | ScheduleTableSchema.table | String | スケジュールテーブルのテーブル名を設定する。 |
| 27 | ScheduleTableSchema.カラム名プロパティ | String | スケジュールテーブルの各カラム名を設定する。 |
| 28 | ScheduleTableSchema.inputColumnType | int | スケジュールテーブルの「入力」カラムのJDBC型を設定する。  デフォルトは、java.sq.Types.VARCHAR。 |
| 29 | ScheduleTableSchema.outputColumnType | int | スケジュールテーブルの「出力」カラムのJDBC型を設定する。  デフォルトは、java.sq.Types.VARCHAR。 |
| 30 | ScheduleTableSchema.stateString\_状態 | String | スケジュールテーブルの「状態」カラムの各状態値を設定する。 |
| 31 | ScheduleTableSchema.controlStateString\_状態 | String | スケジュールテーブルの「制御状態」カラムの各状態値を設定する。 |
| 32 | ScheduleTableSchema.checkStateString\_状態 | String | スケジュールテーブルの「チェック状態」カラムの各状態値を設定する。 |
| 33 | ScheduleGroupTableSchema.table | String | スケジュールグループテーブルのテーブル名を設定する。 |
| 34 | ScheduleGroupTableSchema.カラム名プロパティ | String | スケジュールグループテーブルの各カラム名を設定する。 |
| 35 | ScheduleDependsTableSchema.table | String | スケジュール依存関係テーブルのテーブル名を設定する。 |
| 36 | ScheduleDependsTableSchema.カラム名プロパティ | String | スケジュール依存関係テーブルの各カラム名を設定する。 |
| 37 | ScheduleGroupDependsTableSchema.table | String | スケジュールグループ依存関係テーブルのテーブル名を設定する。 |
| 38 | ScheduleGroupDependsTableSchema.カラム名プロパティ | String | スケジュールグループ依存関係テーブルの各カラム名を設定する。 |

## Scheduler

### DefaultSchedulerService

Schedulerインタフェースの基本的な機能を実装するデフォルト実装クラスである。

実行可能スケジュール監視デーモンを持ち、ScheduleManagerに対して定期的に実行可能スケジュールを検索し、実行可能なスケジュールが存在する場合には、Nimbus Queueにスケジュールを投入する。

また、スケジュールを投入したNimbus Queueを待ち受けるスケジュール実行スレッドプールを持ち、Queueから引き抜いたスケジュールを、ScheduleExecutorに実行依頼する。

内部にScheduleControlListenerを持ち、ScheduleManagerに登録しておくことで、ScheduleManagerから制御状態の変更要求が通知された場合に、ScheduleExecutorに連携する役目も持つ。

**DefaultSchedulerService**

ScheduleManager

ScheduleControlListener

Schedule

Queue

Schedule

ScheduleExecutor

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleTickerInterval | long | 実行可能スケジュール監視デーモンの監視間隔[ms]を設定する。  デフォルトは、1000[ms] |
| 2 | ScheduleManagerServiceName | ServiceName | ScheduleManagerサービスのサービス名を設定する。 |
| 3 | ScheduleExecutorServiceName | ServiceName | ScheduleExecutorサービスのサービス名を設定する。  但し、実行種別が1種類しか存在しない場合。 |
| 4 | ScheduleExecutorServiceNames | ServiceName[] | ScheduleExecutorサービスのサービス名の配列を設定する。  実行種別が複数存在する場合。  この属性を設定した場合、実行可能スケジュールをScheduleManagerから取得する際に、ここで設定されたScheduleExecutorの実行種別を指定する。 |
| 5 | ExecutorKey | String | 実行可能スケジュールをScheduleManagerから取得する際に指定する実行キーを設定する。 |
| 6 | ThreadContextServiceName | ServiceName | スケジュール実行通番を乗せるjp.ossc.nimbus.service.context.Contextサービスのサービス名を設定する。 |
| 7 | SequenceServiceName | ServiceName | スケジュール実行通番を発行するjp.ossc.nimbus.service.sequence.Sequenceサービスのサービス名を設定する。 |
| 8 | TransactionControl | boolean | スケジュールの投入処理でトランザクションを使用するかどうかを設定する。  デフォルトは、false。 |
| 9 | TransactionManagerFactoryServiceName | ServiceName | トランザクションを使用する場合に、TransactionManagerを取得するjp.ossc.nimbus.service.transaction.TransactionManagerFactoryサービスのサービス名を設定する。  この属性が設定されていない場合は、JNDI名"java:/TransactionManager"をlookupして使用する。 |
| 10 | ClusterServiceName | ServiceName | 実行可能スケジュール監視デーモンをクラスタ化するjp.ossc.nimbus.service.keepalive.ClusterServiceのサービス名を設定する。 |
| 11 | QueueServiceName | ServiceName | スケジュールを非同期実行するためのjp.ossc.nimbus.service.queue.Queueサービスのサービス名を設定する。  設定しない場合、内部でjp.ossc.nimbus.service.queue.DefaultQueueServiceを自動生成する。 |
| 12 | ScheduleDispatcherSize | int | スケジュール実行スレッドプールのプール数を設定する。  デフォルトは、1。 |
| 13 | DaemonScheduleDispatcher | boolean | スケジュール実行スレッドをデーモンスレッドにするかどうかを設定する。  デフォルトは、true。 |

### JMSSchedulerService

JMS Queueを使用するSchedulerインタフェースの実装クラスである。

基本的な機能は、DefaultSchedulerServiceに準じるが、Nimbus Queueの代わりにJMS Queueを使用する。

スケジューラサーバとJMSサーバを分ける事で、JMS Queueに投入された状態でスケジューラサーバに障害が発生した場合に、スケジューラサーバの復旧後に処理を続行可能になる。また、JMSの永続化機能を利用すれば、JMSサーバの障害時にも、復旧後にQueueの状態を復元可能になる。

**JMSSchedulerService**

JMSMessageProducerFactory

JMS MessageProducer

Schedule

ScheduleControlListener

ScheduleManager

JMS Queue

JMSSessionFactory

JMS Session

JMSMessageConsumerFactory

JMS MessageConsumer

Schedule

ScheduleExecutor

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleTickerInterval | long | 実行可能スケジュール監視デーモンの監視間隔[ms]を設定する。  デフォルトは、1000[ms] |
| 2 | ScheduleManagerServiceName | ServiceName | ScheduleManagerサービスのサービス名を設定する。 |
| 3 | ScheduleExecutorServiceName | ServiceName | ScheduleExecutorサービスのサービス名を設定する。  但し、実行種別が1種類しか存在しない場合。 |
| 4 | ScheduleExecutorServiceNames | ServiceName[] | ScheduleExecutorサービスのサービス名の配列を設定する。  実行種別が複数存在する場合。  この属性を設定した場合、実行可能スケジュールをScheduleManagerから取得する際に、ここで設定されたScheduleExecutorの実行種別を指定する。 |
| 5 | ExecutorKey | String | 実行可能スケジュールをScheduleManagerから取得する際に指定する実行キーを設定する。 |
| 6 | ThreadContextServiceName | ServiceName | スケジュール実行通番を乗せるjp.ossc.nimbus.service.context.Contextサービスのサービス名を設定する。 |
| 7 | SequenceServiceName | ServiceName | スケジュール実行通番を発行するjp.ossc.nimbus.service.sequence.Sequenceサービスのサービス名を設定する。 |
| 8 | TransactionControl | boolean | スケジュールの投入処理でトランザクションを使用するかどうかを設定する。  デフォルトは、false。 |
| 9 | TransactionManagerFactoryServiceName | ServiceName | トランザクションを使用する場合に、TransactionManagerを取得するjp.ossc.nimbus.service.transaction.TransactionManagerFactoryサービスのサービス名を設定する。  この属性が設定されていない場合は、JNDI名"java:/TransactionManager"をlookupして使用する。 |
| 10 | ClusterServiceName | ServiceName | 実行可能スケジュール監視デーモンをクラスタ化するjp.ossc.nimbus.service.keepalive.ClusterServiceのサービス名を設定する。 |
| 11 | JMSSessionFactoryServiceName | ServiceName | JMS Sessionを取得するjp.ossc.nimbus.service.jms.JMSSessionFactoryサービスのサービス名を設定する。 |
| 12 | JMSMessageProducerFactoryServiceName | ServiceName | JMS MessageProducerを取得するjp.ossc.nimbus.service.jms.JMSMessageProducerFactoryサービスのサービス名を設定する。 |
| 13 | JMSMessageConsumerFactoryServiceName | ServiceName | JMS MessageConsumerを取得するjp.ossc.nimbus.service.jms.JMSMessageConsumerFactoryサービスのサービス名を設定する。 |
| 14 | MessageConsumerSize | int | スケジュール実行スレッドプールのプール数を設定する。  デフォルトは、1。 |
| 15 | DeliveryMode | int | JMS Queueに投入する際の配信モードを設定する。  デフォルトは、javax.jms.Message.DEFAULT\_DELIVERY\_MODE |
| 16 | Priority | int | JMS Queueに投入する際の優先順位を設定する。  デフォルトは、javax.jms.Message.DEFAULT\_PRIORITY |
| 17 | TimeToLive | long | JMS Queueに投入する際の生存期間[ms]を設定する。  デフォルトは、javax.jms.Message.DEFAULT\_TIME\_TO\_LIVE |

## ScheduleExecutor

### BeanFlowScheduleExecutorService

スケジュールで行うタスクをBeanFlowで実行するScheduleExecutorインタフェースの実装クラスである。

Scheduleの状態遷移を、ScheduleManagerに伝えながら、Scheduleの実行を行う。また、実行結果がリトライ要求の場合は、リトライするためにスケジュールの再登録を行う。

Schedulerから実行依頼されたScheduleのタスク名を取得して、BeanFlow名と解釈し、BeanFlowInvokerFactoryからBeanFlowInvokerを取得する。BeanFlowInvokerにScheduleを引数として渡して、タスクの実行を依頼する。

また、Schedulerから、制御状態の変更を要求されると、BeanFlowモニタの機能を使って、中断、再開、中止などの制御を行う。

Scheduler

Schedule

ScheduleManager

BeanFlow

定義XML

BeanFlowInvokerFactory

**BeanFlowScheduleExecutorService**

Schedule

BeanFlowInvoker

タスク

BeanFlow

定義XML

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleManagerServiceName | ServiceName | スケジュールの状態遷移を行うScheduleManagerサービスのサービス名を設定する。 |
| 2 | Key | String | 実行キーを設定する。  デフォルトは、ホスト名。 |
| 3 | Type | String | 実行種別を設定する。  デフォルトは、BEANFLOW |
| 4 | JournalServiceName | ServiceName | ジャーナルを出力するjp.ossc.nimbus.service.journal.Journalサービスのサービス名を設定する。 |
| 5 | EditorFinderServiceName | ServiceName | ジャーナルを編集するJournalEditorを検索するjp.ossc.nimbus.service.journal.editorfinder.EditorFinderサービスのサービス名を設定する。 |
| 6 | ThreadContextServiceName | ServiceName | スケジュール実行通番を取得するjp.ossc.nimbus.service.context.Contextサービスのサービス名を設定する。 |
| 7 | BeanFlowInvokerFactoryServiceName | ServiceName | BeanFlowInvokerを取得するjp.ossc.nimbus.service.beancontrol.interfaces. BeanFlowInvokerFactoryサービスのサービス名を設定する。 |
| 8 | ControlStateChangingWaitInterval | long | 制御状態が変わるのを確認する間隔[ms]を設定する。  デフォルトは、500[ms] |
| 9 | ControlStateChangingWaitTimeout | long | 制御状態が変わるのを待つ時間[ms]を設定する。  デフォルトは、-1で無限待ち。 |

### CommandScheduleExecutorService

スケジュールで行うタスクを別プロセスのコマンドとして実行するScheduleExecutorインタフェースの実装クラスである。

Scheduleの状態遷移を、ScheduleManagerに伝えながら、Scheduleの実行を行う。また、実行結果がリトライ要求の場合は、リトライするためにスケジュールの再登録を行う。

Schedulerから実行依頼されたScheduleのタスク名を取得して、別プロセスを起動しコマンドとして実行する。コマンドに引数を指定する場合は、Scheduleの入力にCSV形式で指定する。そのCSVの最後の要素として、””&”を指定すると、コマンドプロセスの終了を待たずに終了する。コマンドプロセスの終了を待つ場合は、コマンドの終了コードが0の場合は、正常終了、それ以外は異常終了とみなして、スケジュール状態に反映させる。また、標準出力およびエラー出力は、スケジュールの出力に記録される。

Schedulerから、制御状態の変更を要求されると、コマンドの終了待ちをしている場合に限って、中止（コマンドの終了待ちを諦める）の制御のみ行う。

ScheduleManager

Schedule

Scheduler

プロセス

**CommandScheduleExecutorService**

コマンド　引数

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleManagerServiceName | ServiceName | スケジュールの状態遷移を行うScheduleManagerサービスのサービス名を設定する。 |
| 2 | Key | String | 実行キーを設定する。  デフォルトは、ホスト名。 |
| 3 | Type | String | 実行種別を設定する。  デフォルトは、COMMAND |
| 4 | JournalServiceName | ServiceName | ジャーナルを出力するjp.ossc.nimbus.service.journal.Journalサービスのサービス名を設定する。 |
| 5 | EditorFinderServiceName | ServiceName | ジャーナルを編集するJournalEditorを検索するjp.ossc.nimbus.service.journal.editorfinder.EditorFinderサービスのサービス名を設定する。 |
| 6 | ThreadContextServiceName | ServiceName | スケジュール実行通番を取得するjp.ossc.nimbus.service.context.Contextサービスのサービス名を設定する。 |
| 7 | WorkDirectory | String | コマンドの作業ディレクトリを設定する。  デフォルトは、JavaVMの作業ディレクトリ。 |
| 8 | EnvironmentVariables | String[] | コマンドプロセスに与える環境変数を設定する。 |

### ConcentrateScheduleExecutorService

スケジュールで行うタスクを集配信タスクとして実行するScheduleExecutorインタフェースの実装クラスである。このクラスは、BeanFlowScheduleExecutorのサブクラスとなるため、集配信タスクの実行は、BeanFlowで行う。

Scheduleの状態遷移を、ScheduleManagerに伝えながら、Scheduleの実行を行う。また、実行結果がリトライ要求の場合は、リトライするためにスケジュールの再登録を行う。

タスク実行の前処理として、Schedulerから実行依頼されたScheduleの入力に指定された文字列を解析して、ConcentrateRequestに格納し、Scheduleの入力に再設定する。ここで、 入力文字列のフォーマットは、"GET|PUT,source,destination"である。

また、空のConcentrateResponseを生成し、Scheduleの出力に設定する。

さらに、Scheduleのタスク名に指定された文字列を解析して、BeanFlow名とConcentrateRequestのキーを取得してConcentrateRequestに設定し、BeanFlowを実行する。ここで、タスク名文字列のフォーマットは、"BeanFlow名[:ConcentrateRequestのキー]"である。

タスク実行処理であるBeanFlowでは、スケジュールの入力として渡されたConcentrateRequestの情報を使って、集配信処理を行い、その結果をスケジュールの出力として渡されたConcentrateResponseに格納する。

収集（GET）のBeanFlowでは、ConcentrateRequest#getSource()を収集元として、ファイルを取得し、ローカルに保存する。ConcentrateResponse#addFile(File)を使って、ローカルに保存したファイルパスを収集結果として格納する。その後、タスク実行後の後処理として、ConcentrateResponseからローカルに保存されたファイルを取得し、ConcentrateBackupManagerによってバックアップが行われた後、ConcentrateRequest#getDestination()の宛先へとファイルを移動する。

配信（PUT）のBeanFlowでは、ConcentrateRequest#getSource()を配信元として、ローカルのファイルを取得し、ConcentrateRequest#getDestination()の宛先へと配信する。ConcentrateResponse#addFile(File)を使って、配信元のローカルファイルパスを配信結果として格納する。その後、タスク実行後の後処理として、ConcentrateResponseから配信元のローカルファイルを取得し、ConcentrateBackupManagerによってバックアップが行われた後、削除する。

実行後のScheduleの出力には、BeanFlowで明示的に設定されていなければ、ConcentrateBackupManagerによってバックアップされたファイルの一覧が出力される。

Scheduler

Schedule

ScheduleManager

バックアップ

**ConcentrateScheduleExecutorService**

ConcentrateBackupManager

BeanFlow

定義XML

ConcentrateResponse

BeanFlowInvoker

BeanFlowInvokerFactory

Schedule

ConcentrateRequest

集配信

BeanFlow

定義XML

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ScheduleManagerServiceName | ServiceName | スケジュールの状態遷移を行うScheduleManagerサービスのサービス名を設定する。 |
| 2 | Key | String | 実行キーを設定する。  デフォルトは、ホスト名。 |
| 3 | Type | String | 実行種別を設定する。  デフォルトは、CONCENTRATE |
| 4 | JournalServiceName | ServiceName | ジャーナルを出力するjp.ossc.nimbus.service.journal.Journalサービスのサービス名を設定する。 |
| 5 | EditorFinderServiceName | ServiceName | ジャーナルを編集するJournalEditorを検索するjp.ossc.nimbus.service.journal.editorfinder.EditorFinderサービスのサービス名を設定する。 |
| 6 | ThreadContextServiceName | ServiceName | スケジュール実行通番を取得するjp.ossc.nimbus.service.context.Contextサービスのサービス名を設定する。 |
| 7 | BeanFlowInvokerFactoryServiceName | ServiceName | BeanFlowInvokerを取得するjp.ossc.nimbus.service.beancontrol.interfaces. BeanFlowInvokerFactoryサービスのサービス名を設定する。 |
| 8 | ControlStateChangingWaitInterval | long | 制御状態が変わるのを確認する間隔[ms]を設定する。  デフォルトは、500[ms] |
| 9 | ControlStateChangingWaitTimeout | long | 制御状態が変わるのを待つ時間[ms]を設定する。  デフォルトは、-1で無限待ち。 |
| 10 | ConcentrateBackupManagerServiceName | ServiceName | 集配信ファイルのバックアップを行うConcentrateBackupManagerサービスのサービス名を設定する。 |

# 分散構成

スケジュールの実行負荷を分散させるために、分散構成を採る事ができる。

以下に、その構成例を示す。

スケジュールサーバ

ScheduleManager

Scheduler

ScheduleExecuter

ScheduleExecuter

（プロキシ）

分散サーバ２

分散サーバ１

ScheduleExecuter

ScheduleExecuter

ScheduleManager

（プロキシ）

ScheduleManager

（プロキシ）

ScheduleExecuterをスケジュールサーバと、分散サーバ１、分散サーバ２の３台に分散させた構成例である。各ScheduleExecuterへの振り分けは、Nimbus Proxyのクラスタ機能により振り分け方式を選択可能で、ラウンドロビンとファーストアベイラブル等の実装が用意されている。

# クラスタ構成

スケジュールサーバで障害が発生しても、フェイルオーバしてスケジュールを実行し続けるクラスタ構成を採る事ができる。

以下に、その構成例を示す。

ScheduleManager

スケジュールサーバ（稼働系）

Scheduler

ScheduleExecuter

Cluster

DBサーバ

DBサーバ

DB

JMS Queue

スケジュールサーバ（待機系）

Cluster

ScheduleExecuter

ScheduleManager

Scheduler

（Created）

スケジュールサーバは、稼働系、待機系とも同じサービスの構成とする。先に起動したサーバのClusterサービスが、自身を稼働系と位置付けSchedulerを起動させる。後に起動したサーバのClusterサービスは、自身を待機系と位置付けSchedulerをCreatedな状態のまま待機させておく。Clusterサービスは、相互にハートビート通信を行っており、稼働系からのハートビートが途絶えた場合は、待機系のClusterサービスがSchedulerを起動して、スケジュール実行を引き継ぐ。

スケジュールの投入は、Schedulerの実装にJMSSchedulerServiceを適用し、JMS Queueに投入する事で、トランザクション保障と永続化を行う。この例では、永続化先をDBとした。

スケジュールの状態管理は、ScheduleMannagerの実装にDatabaseScheduleManagerServiceを適用し、DBでスケジュール管理を行う事で、トランザクション保障と永続化を行う。

また、この構成において、ScheduleExecuterを分散構成のプロキシ方式とすれば、分散クラスタ構成も可能である。

# スケジュール管理画面

スケジュールの、検索、変更、削除、追加等を行うスケジュール管理機能をWebアプリケーション及び、Webサービスとして提供する。

Webアプリケーション及びWebサービス共に、サーブレット実装で提供するため、以下のようにWebアプリケーション配置記述子web.xmlを記述し、サーブレットコンテナに配置する事で、利用可能となる。

<?xml version="1.0" encoding="Shift\_JIS"?>

<web-app version="2.5"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<servlet>

<servlet-name>ScheduleManagerServlet</servlet-name>

<servlet-class>jp.ossc.nimbus.servlet.ScheduleManagerServlet</servlet-class>

<init-param>

<param-name>ScheduleManagerServiceName</param-name>

<param-value>SchedulerAdmin#ScheduleManager</param-value>

</init-param>

<init-param>

<param-name>SchedulerServiceName</param-name>

<param-value>SchedulerAdmin#Scheduler</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>ScheduleManagerServlet</servlet-name>

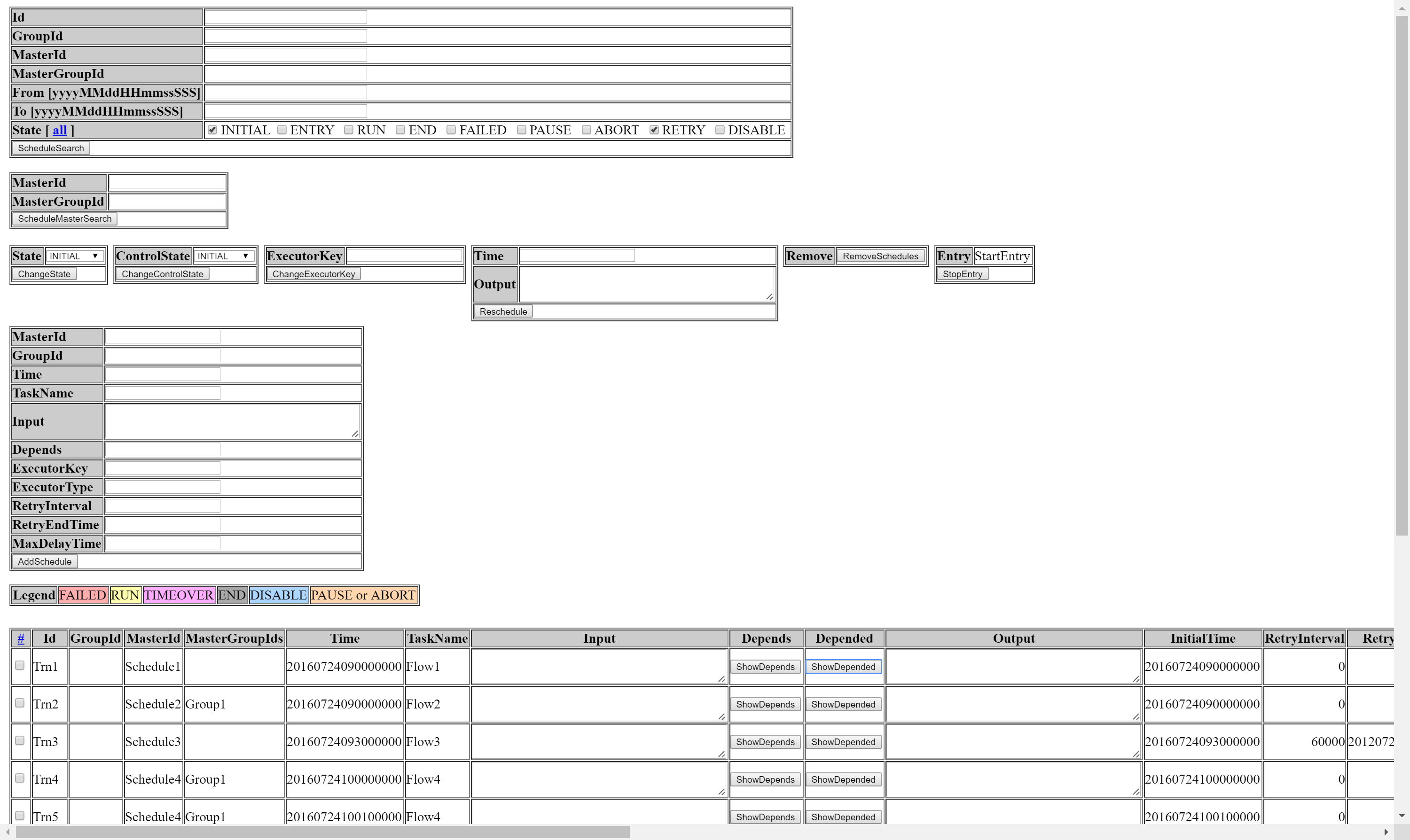
<url-pattern>/schedule-console</url-pattern>

</servlet-mapping>

</web-app>

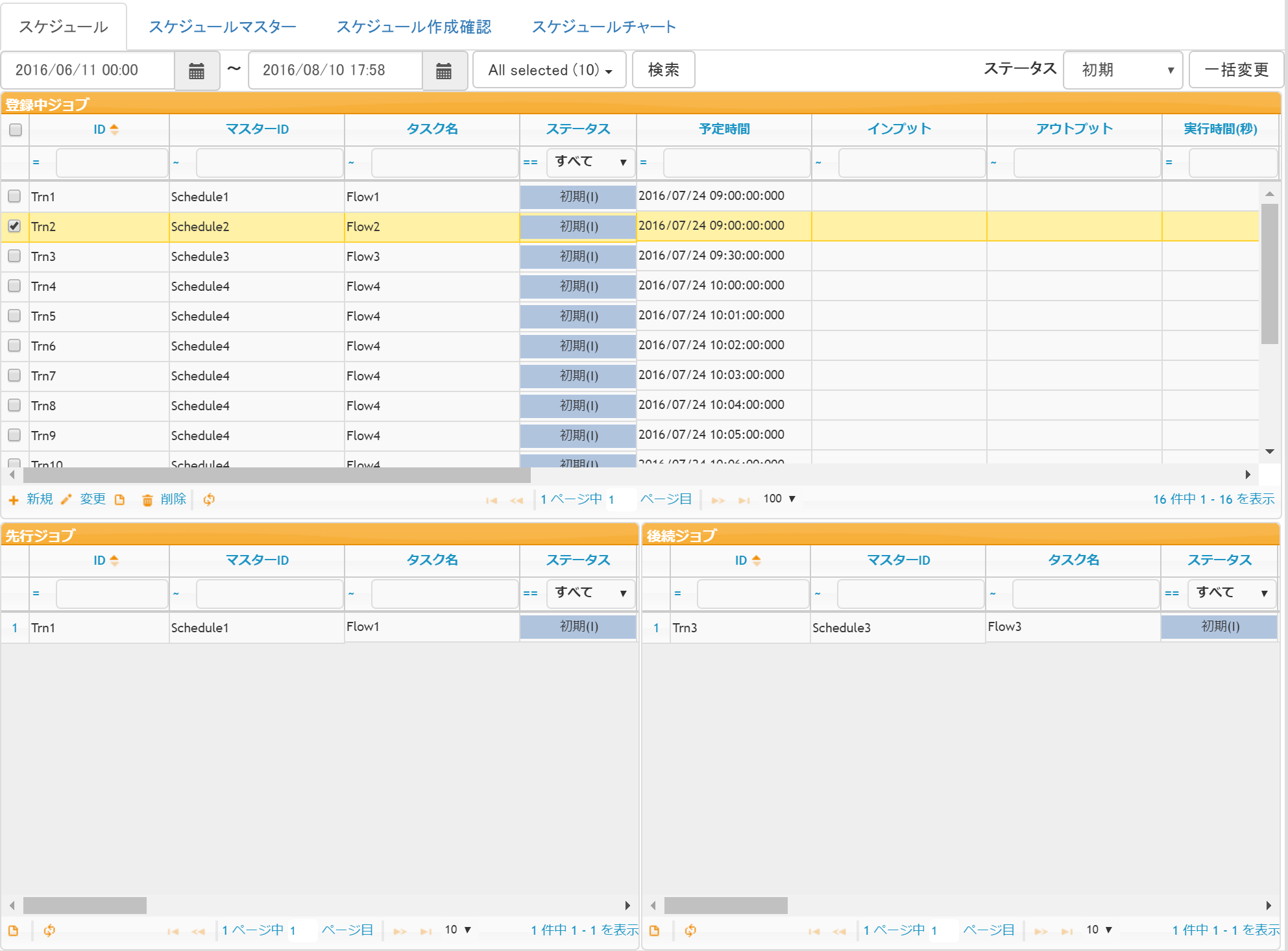
ScheduleManagerServletを配置するだけで、簡易なHTMLを出力するWebアプリケーションを利用可能だが、同時に利用可能になるWebサービスの機能を使った、リッチなHTML版もサンプルとして提供している。

## 簡易画面

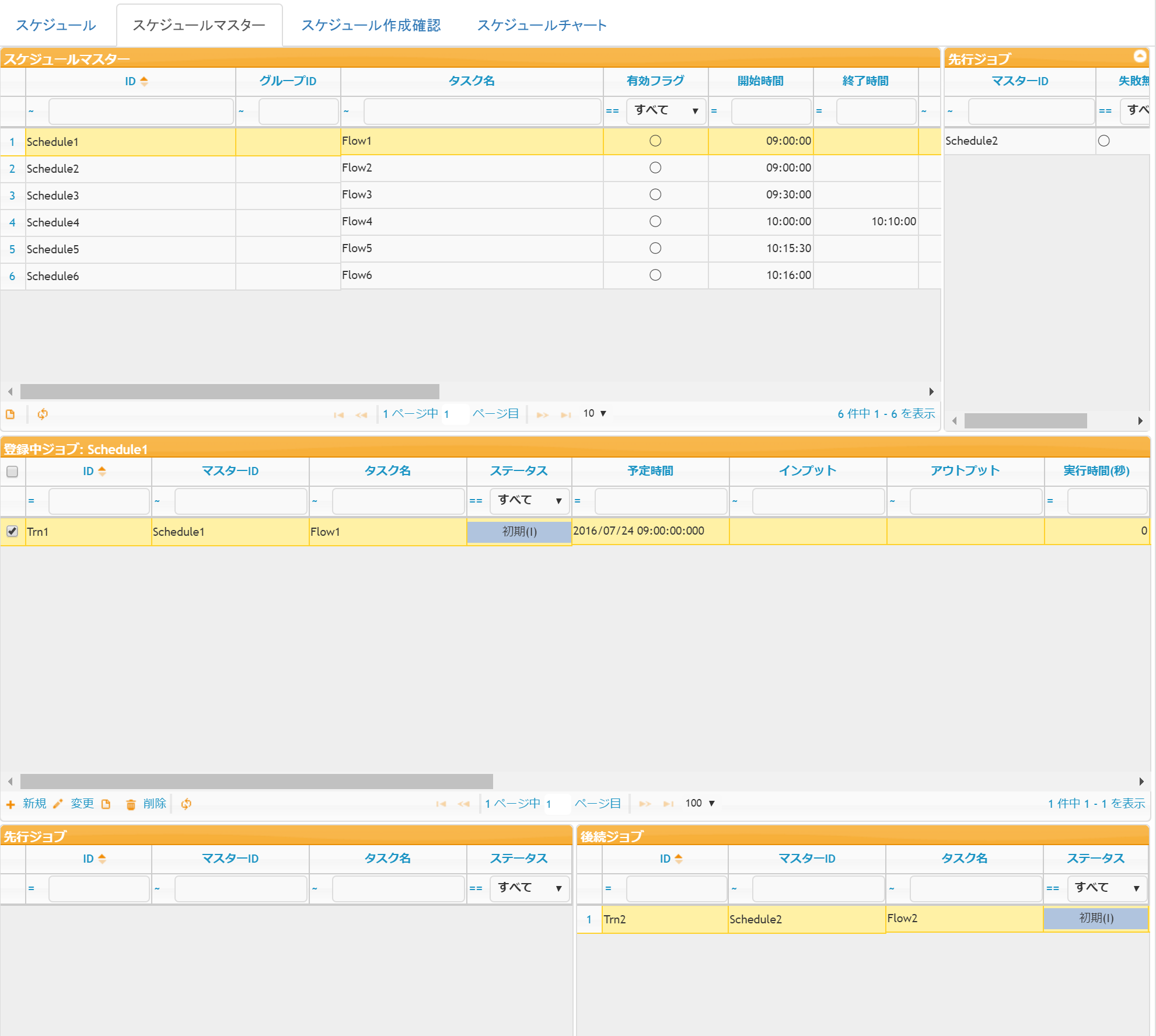


## リッチ管理画面

### スケジュール管理画面



### スケジュールマスタ管理画面



### スケジュール作成確認画面



### スケジュールチャート

