Nimbusサーブレット仕様書

2016年10月4日

目次

[1. 概要 1](#_Toc464037228)

[2. インタフェース設計 3](#_Toc464037229)

[2.1. jp.ossc.nimbus.servlet.BeanFlowSelector 3](#_Toc464037230)

[2.2. jp.ossc.nimbus.service.rest.RestServer 4](#_Toc464037231)

[3. 実装設計 5](#_Toc464037232)

[3.1. javax.servlet.Filter 5](#_Toc464037233)

[3.1.1. jp.ossc.nimbus.servlet.InterceptorChainCallFilter 5](#_Toc464037234)

[3.2. javax.servlet.http.HttpServlet 7](#_Toc464037235)

[3.2.1. jp.ossc.nimbus.servlet.ServiceManagerFactoryServlet 7](#_Toc464037236)

[3.2.2. jp.ossc.nimbus.servlet.ScheduleManagerServlet 10](#_Toc464037237)

[3.2.3. jp.ossc.nimbus.servlet.SharedContextServlet 13](#_Toc464037238)

[3.2.4. jp.ossc.nimbus.servlet.BeanFlowServlet 16](#_Toc464037239)

[3.2.5. jp.ossc.nimbus.servlet.RestServlet 19](#_Toc464037240)

[3.3. jp.ossc.nimbus.service.rest.RestServer 21](#_Toc464037241)

[3.3.1. jp.ossc.nimbus.service.rest.BeanFlowRestServerService 21](#_Toc464037242)

[3.4. jp.ossc.nimbus.servlet.BeanFlowSelector 26](#_Toc464037243)

[3.4.1. jp.ossc.nimbus.servlet.DefaultBeanFlowSelectorService 26](#_Toc464037244)

[3.4.2. jp.ossc.nimbus.servlet.MappingBeanFlowSelectorService 26](#_Toc464037245)

# 概要

Nimbusをサーブレットコンテナ上で利用するための機能を提供する。

大きく２つに分けられ、１つは、NimbusでWebアプリケーションを構築するためのフレームワーク機能。もう１つは、NimbusのサービスをWebアプリケーションとして利用する機能である。

Webアプリケーションフレームワーク機能は、フィルター層とサーブレット層に分かれる。

Nimbus業務フロー

サーブレット

サーブレットフィルタ

Interceptor

Interceptor

Interceptor

アプリケーション

サーブレットフィルタでは、Nimbus AOPを使って、Interceptorサービスを階層的に挟み込む事ができる。これにより、サーブレットまでリクエストが到達する前及び、サーブレットから返ってきた応答の後に、様々な処理を付加する事ができる。例えば、ジャーナルを出力したり、リクエスト情報をDTO（Data Transfer Object）に変換したり、DTOからレスポンス情報に変換したり、例外処理を行ったり、応答を圧縮したり、様々な処理を付加して、HTTPの入出力に対する処理フレームワークを構築する事が可能になる。

サーブレットでは、リクエスト情報に応じて、Nimbus業務フローを呼び出し、リクエストに応じたアプリケーションロジックを実行してレスポンスを返す処理を行う。

サービスWebアプリケーション機能は、Nimbusの特定の機能をWebアプリケーションとして利用可能にしたもので、サービスの管理機能と言える。いくつかの特定のサービスに対するサーブレットが提供されているが、その中でも、ServiceManagerFactoryをWebアプリケーション化するサーブレットは、Nimbus自体の管理コンソールと言える。また、サーブレットコンテナ上に、Nimbusのサービスを配置するエントリポイントとしての役目も持つ。

サーブレット

特定のサービス

ブラウザ

管理コンソール

JavaDocは、以下を参照。

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

サンプルコードは、以下のzipファイル内のservlet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

## jp.ossc.nimbus.servlet.BeanFlowSelector

HTTPリクエストから実行する業務フロー名を選択する機能を提供するインタフェースである。

selectBeanFlow(request)

**BeanFlowSelector**

selectBeanFlow()にHTTPリクエストを渡すと、実行する業務フロー名を返す。

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

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.servlet.DefaultBeanFlowSelectorService | リクエストされたパスを業務フロー名として解釈するデフォルト実装クラスである。 |
| 2 | jp.ossc.nimbus.servlet.MappingBeanFlowSelectorService | リクエストされたパスと業務フロー名をマッピング定義することができる実装クラスである。 |

## jp.ossc.nimbus.service.rest.RestServer

RESTfulなWebサービスの実装を提供するインタフェースである。RESTfulなWebサービスのAPIは、HTTPメソッド毎に提供する機能が分類されている。そのため、HTTPメソッド単位で、RestServerのメソッドも分かれている。また、HTTPメソッド毎に、入力や出力の渡し方が規定されている。そのため、メソッド毎に、入力及び出力のDTOの型も異なる。

process～(request, response)

**RestServer**

～RestRequest

～RestResponse

このインタフェースで、サポートするHTTPメソッドは、HEAD、POST、PUT、GET、DELETE、OPTIONSである。

以下に、RestServerインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.rest.BeanFlowRestServerService | RESTfulなWebサービスの実装を、Nimbus業務フローで実装するフレームワークを提供する実装クラスである。 |

# 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実装クラスの機能を説明する。また、Nimbusサーブレットは、JavaEEのサーブレット規格のインタフェースや、クラスを実装及び継承するクラスが、主となるためそれらのクラスについても説明する。

## javax.servlet.Filter

### jp.ossc.nimbus.servlet.InterceptorChainCallFilter

JavaEEのサーブレット規格のjavax.servlet.Filterインタフェースの実装クラスで、サーブレットフィルタからNimbusアスペクト指向のInterceptorChainFactoryサービスを呼び出し、HTTPリクエストのパスに応じた、任意のInterceptorのチェーンを挟み込むことができる。

サーブレット

**InterceptorChainCallFiler**

InterceptorChain

InterceptorChainList

Interceptor

Interceptor

Interceptor

getInterceptorChainList(path)

InterceptorChainList

Interceptor

Interceptor

Interceptor

InterceptorChainFactory

InterceptorChainList

Interceptor

Interceptor

Interceptor

InterceptorChainCallFilter は、サーブレットフィルタの実装なので、Webアプリケーションの配置記述子web.xmlにて配置する。web.xmlにて、フィルターの初期化パラメータとして、InterceptorChainListのサービス名、または、InterceptorChainFactoryのサービス名を設定できる。InterceptorChainListが設定されている場合は、全てのHTTPリクエストに対して、同じInterceptorのチェーンを適用する。InterceptorChainFactoryが設定されている場合は、HTTPリクエストのパスを、InterceptorChainFactoryのgetInterceptorChainList(path)に渡すことで、パスに応じたInterceptorのチェーンを適用する。

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<filter>

<filter-name>InterceptorChainCallFilter</filter-name>

<filter-class>jp.ossc.nimbus.servlet.InterceptorChainCallFilter</filter-class>

<init-param>

<param-name>InterceptorChainListServiceName</param-name>

<param-value>Nimbus#InterceptorChainList</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>InterceptorChainCallFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | InterceptorChainListServiceName | Nimbusアスペクト指向のInterceptorChainListサービスのサービス名を設定する。 |
| 2 | InterceptorChainFactoryServiceName | Nimbusアスペクト指向のInterceptorChainFactoryサービスのサービス名を設定する。 |
| 3 | UseThreadLocalInterceptorChain | 再利用可能なInterceptorChainの実装を利用するかどうかを設定する。  デフォルトは、trueで利用する。 |

## javax.servlet.http.HttpServlet

### jp.ossc.nimbus.servlet.ServiceManagerFactoryServlet

JavaEEのサーブレット規格のjavax.servlet.http.HttpServletクラスのサブクラスで、NimbusコアのServiceManagerFactoryの機能をWebアプリケーションとして提供することで、NimbusのサービスをWebアプリケーションで管理可能にする。また、Webサービスとして、動作させる事も可能である。

サーブレットコンテナ上に、サービス定義を読み込み、サービスを配置するためのエントリポイントとして、動作させる事も可能である。

Webサービス

JSON

**ServiceManagerFactoryServlet**

ServiceManagerFactory

ServiceManager

ServiceManager

Service

ServiceManager

サービス定義

ブラウザ

Webアプリケーションとしては、以下のような機能がある。

* ServiceManagerの一覧表示
* ServiceManagerに登録されているServiceの一覧表示
* Serviceの属性及び操作の一覧表示
* Serviceの属性値の変更
* Serviceの操作の実行

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<servlet>

<servlet-name>NimbusServlet</servlet-name>

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

<init-param>

<param-name>ServicePaths</param-name>

<param-value>sample1-service.xml,sample2-service.xml</param-value>

</init-param>

<init-param>

<param-name>CheckLoadManagerCompleted</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>ConsoleEnabled</param-name>

<param-value>true</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>NimbusServlet</servlet-name>

<url-pattern>/nimbus-console</url-pattern>

</servlet-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | ServicePaths | ロードするサービス定義ファイルのパスをカンマ区切りで指定する。  パスは、絶対パス、相対パス、クラスパスが指定できる。  ここで指定されたサービス定義は、サーブレットの初期化時に指定された順番にロードされ、サーブレットの破棄時に指定された順と逆順でアンロードされる。 |
| 2 | CheckLoadManagerCompleted | サービス定義のロード完了チェックを行うかどうかを指定する。  デフォルトは、falseでチェックを行わない。 |
| 3 | CheckLoadManagerCompletedBy | サービス定義のロード完了チェックをServiceManager単位で行いたい場合に、ServiceManagerの名前をカンマ区切りで指定する。  初期化パラメータCheckLoadManagerCompletedがtrueの場合だけ、有効である。 |
| 4 | Validate | サービス定義の検証を行うかどうかを指定する。  デフォルトは、falseで検証を行わない。 |
| 5 | ConsoleEnabled | このサーブレットが提供する管理コンソール及びWebサービスを有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 6 | AttributeSetEnabled | このサーブレットが提供する管理コンソール及びWebサービスで、サービスの属性を変更する機能を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 7 | MethodCallEnabled | このサーブレットが提供する管理コンソール及びWebサービスで、サービスの操作を実行する機能を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 8 | IgnoreMethods | このサーブレットが提供する管理コンソール及びWebサービスで、無効にしたいサービスのメソッドを指定する。  メソッド指定のフォーマットは、jp.ossc.nimbus.beans.MethodArrayEditorの仕様に準じる。 |
| 9 | JSONConverterServiceName | JSON形式での応答を要求する場合に使用するBeanJSONConverterサービスのサービス名を指定する。  指定しない場合は、内部生成される。 |
| 10 | UnicodeEscape | JSON形式での応答を要求する場合に、２バイト文字をユニコードエスケープするかどうかを指定する。  デフォルトは、trueで、ユニコードエスケープされる。 |

### jp.ossc.nimbus.servlet.ScheduleManagerServlet

JavaEEのサーブレット規格のjavax.servlet.http.HttpServletクラスのサブクラスで、NimbusスケジューラのScheduleManagerや、Schedulerの機能をWebアプリケーションとして提供することで、スケジュールをWebアプリケーションで管理可能にする。また、Webサービスとして、動作させる事も可能である。

ServiceManager

Schedule

ScheduleManager

Webサービス

JSON

**ScheduleManagerServlet**

ブラウザ

Scheduler

Webアプリケーションとしては、以下のような機能がある。

* スケジュールの一覧表示及び条件指定での検索
* 先行、後続スケジュールの表示
* スケジュールマスタの表示
* スケジュールの作成
* スケジュールの変更（状態、制御状態、時刻など）
* スケジュールの削除
* スケジュール投入の開始・停止

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<servlet>

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

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

<init-param>

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

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

</init-param>

<init-param>

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

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

</init-param>

</servlet>

<servlet-mapping>

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

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

</servlet-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | ScheduleManagerServiceName | 対象とするScheduleManagerサービスのサービス名を指定する。 |
| 2 | SchedulerServiceName | 対象とするSchedulerサービスのサービス名を指定する。 |
| 3 | MakeEnabled | スケジュール作成操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 4 | AddEnabled | スケジュール追加操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 5 | RescheduleEnabled | スケジュール時刻変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 6 | RemoveEnabled | スケジュール削除操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 7 | RemoveAllEnabled | スケジュール削除操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 8 | ChangeStateEnabled | スケジュール状態変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 9 | ChangeControlStateEnabled | の制御状態変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 10 | ChangeExecutorKeyEnabled | 実行キー変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 11 | ChangeRetryEndTimeEnabled | リトライ終了時刻変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 12 | ChangeMaxDelayTimeEnabled | 最大遅延時間変更操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 13 | StopEntryEnabled | スケジュール投入停止操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 14 | JSONConverterServiceName | JSON形式での応答を要求する場合に使用するBeanJSONConverterサービスのサービス名を指定する。  指定しない場合は、内部生成される。 |
| 15 | UnicodeEscape | JSON形式での応答を要求する場合に、２バイト文字をユニコードエスケープするかどうかを指定する。  デフォルトは、trueでユニコードエスケープする。 |

### jp.ossc.nimbus.servlet.SharedContextServlet

JavaEEのサーブレット規格のjavax.servlet.http.HttpServletクラスのサブクラスで、NimbusコンテキストのSharedContextやDistributedSharedContextの機能をWebアプリケーションとして提供することで、メモリグリッド上のデータをWebアプリケーションで管理可能にする。また、Webサービスとして、動作させる事も可能である。

Webサービス

JSON

**SharedContextServlet**

SharedContext

SharedContext

ブラウザ

DistributedSharedContext

DistributedSharedContext

Webアプリケーションとしては、以下のような機能がある。

* SharedContext及びDistributedSharedContextの一覧表示
* SharedContext及びDistributedSharedContextの属性表示
* SharedContext及びDistributedSharedContext上のキー一覧表示
* SharedContext及びDistributedSharedContext上のキー指定でのデータ表示
* SharedContext及びDistributedSharedContext上のキー指定でのデータ更新
* SharedContext及びDistributedSharedContextへのデータ登録
* SharedContext及びDistributedSharedContextからのデータ削除
* SharedContext及びDistributedSharedContextへのクエリ実行
* SharedContext及びDistributedSharedContextの永続化及びロード

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<servlet>

<servlet-name>SharedContextServlet</servlet-name>

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

</servlet>

<servlet-mapping>

<servlet-name>SharedContextServlet</servlet-name>

<url-pattern>/context-console</url-pattern>

</servlet-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | ServiceNames | 対象とするSharedContext及びDistributedSharedContextサービスのサービス名をカンマ区切りで指定する。  指定しない場合は、存在する全ての共有コンテキストサービス。 |
| 2 | InterpreterServiceName | 入力フィールドでスクリプトを入力された場合に、その解釈を行うインタープリタのサービス名を指定する。  指定しない場合は、内部的に生成されたBeanShellInterpreterService。 |
| 3 | PutEnabled | 追加操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 4 | UpdateEnabled | 更新操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 5 | RemoveEnabled | 削除操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 6 | ClearEnabled | 全削除操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 7 | LoadEnabled | 読込操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 8 | SaveEnabled | 永続化操作を有効にするかどうかを指定する。  デフォルトは、falseで無効。 |
| 9 | JSONConverterServiceName | JSON形式での応答を要求する場合に使用するBeanJSONConverterサービスのサービス名を指定する。  指定しない場合は、内部生成される。 |
| 10 | UnicodeEscape | JSON形式での応答を要求する場合に、２バイト文字をユニコードエスケープするかどうかを指定する。  デフォルトは、trueでユニコードエスケープする。 |

### jp.ossc.nimbus.servlet.BeanFlowServlet

JavaEEのサーブレット規格のjavax.servlet.http.HttpServletクラスのサブクラスで、HTTPリクエストに応じて、Nimbus業務フローを使ったアプリケーションを実行する。3.1.1のInterceptorChainCallFilterと組み合わせることで、自由度の高いWebアプリケーションフレームワークを構築できる。

BeanFlowSelector

selectBeanFlow(request)

BeanFlowInvoker

(入力検証用)

BeanFlowServletContext

**BeanFlowServlet**

InterceptorChainCallFiler

InterceptorChain

InterceptorChainList

Interceptor

Interceptor

Interceptor

BeanFlowInvoker

(リクエスト処理用)

createFlow(flowName)

BeanFlowInvokerFactory

BeanFlowServletにHTTPリクエストが到達すると、BeanFlowSelectorのselectBeanFlow(request)にHTTPリクエストを渡して、実行すべきリクエスト処理を実装した業務フロー名を選択する。その業務フロー名に、固定の前置詞（デフォルトでは、”validate“）を付けた名前で、入力の検証処理を実装した業務フローが存在するか調べる。存在する場合は、その入力検証用の業務フローを実行する。その際に、HTTPリクエスト、HTTPレスポンス、Interceptorで変換した入力のDTOなどを格納したBeanFlowServletContextを生成し、引数情報として渡す。入力検証用の業務フローの戻り値がfalseの場合は、処理を中断して、レスポンスを返す。また、戻り値がtrueの場合は、リクエスト処理用の業務フローを実行して、レスポンスを返す。

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<servlet>

<servlet-name>BeanFlowServlet</servlet-name>

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

<init-param>

<param-name>Validate</param-name>

<param-value>true</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>BeanFlowServlet</servlet-name>

<url-pattern>\*.bf</url-pattern>

</servlet-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | BeanFlowInvokerFactoryServiceName | jp.ossc.nimbus.service.beancontrol.interfaces.BeanFlowInvokerFactoryサービスのサービス名を指定する。 |
| 2 | BeanFlowSelectorServiceName | BeanFlowSelectorサービスのサービス名を指定する。  指定しない場合は、DefaultBeanFlowSelectorServiceが内部で生成される。 |
| 3 | JournalServiceName | jp.ossc.nimbus.service.journal.Journalサービスのサービス名を指定する。 |
| 4 | ContextServiceName | ジャーナルに設定するリクエストIDの取得元となるjp.ossc.nimbus.service.context.Contextサービスのサービス名を指定する。 |
| 5 | Validate | 入力検証を行うかどうかを設定する。  デフォルトは、falseで、入力検証を行わない。 |
| 6 | ValidateFlowPrefix | 入力検証用の業務フロー名を特定する前置詞を設定する。  デフォルトは、“validate”。 |
| 7 | InputAttributeName | 入力DTOの取得するHTTPリクエストの属性名を設定する。  デフォルトでは、Nimbusアスペクト指向のjp.ossc.nimbus.service.aop.interceptor.servlet.StreamExchangeInterceptorServiceがHTTPリクエストの入力ストリームから変換した入力DTOを、HTTPリクエストの属性に設定する属性名と同じ。 |
| 8 | OutputAttributeName | リクエスト処理用の業務フローの結果として、戻り値または、BeanFlowServletContext#getOutput()で取得されるオブジェクトを、出力DTOとして、HTTPリクエストの属性に設定する属性名を設定する。  デフォルトでは、Nimbusアスペクト指向のjp.ossc.nimbus.service.aop.interceptor.servlet.StreamExchangeInterceptorServiceが出力DTOを変換してHTTPレスポンスの出力ストリームに書き込むために、HTTPリクエストの属性から出力DTOを取得する際の属性名と同じ。 |

### jp.ossc.nimbus.servlet.RestServlet

JavaEEのサーブレット規格のjavax.servlet.http.HttpServletクラスのサブクラスで、HTTPリクエストのメソッドに応じて、jp.ossc.nimbus.service.rest.RestServerの各メソッドを実行するRESTfulなWebサービスを実装するためのサーブレットである。3.1.1のInterceptorChainCallFilterと組み合わせることで、自由度の高いWebサービスフレームワークを構築できる。

process～(request, response)

RestRequest

RestServer

**RestServerServlet**

InterceptorChainCallFiler

InterceptorChain

InterceptorChainList

Interceptor

Interceptor

Interceptor

RestResponse

RestServerServletにHTTPリクエストが到達すると、HTTPメソッドに対応するRestServerサービスのメソッドを呼び出し、処理を委譲する。

以下に、web.xmlの記述例を示す。

webアプリケーション配置記述子web.xml

<servlet>

<servlet-name>RestServlet</servlet-name>

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

<init-param>

<param-name>RestServerServiceName</param-name>

<param-value>Nimbus#RestServer</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>RestServlet</servlet-name>

<url-pattern>\*</url-pattern>

</servlet-mapping>

以下に初期化パラメータの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | パラメータ名 | 説明 |
| 1 | RestServerServiceName | jp.ossc.nimbus.service.rest.RestServerサービスのサービス名を指定する。 |

## jp.ossc.nimbus.service.rest.RestServer

### jp.ossc.nimbus.service.rest.BeanFlowRestServerService

RESTfulなWebサービスをNimbus業務フローで実装するフレームワークを提供する実装クラスである。

RESTサーバ定義

XMLファイル

BeanFlowInvokerFactory

出力DTO

入力DTO

アプリケーション

**BeanFlowRestServerService**

BindingStreamConverter

パス

パラメータ

入力ストリーム

RestContext

RestRequest

BeanFlowInvoker

process～(request, response)

RestResponse

ステータス

メッセージ

出力ストリーム

StreamConverter

RESTサーバ定義XMLファイルで、Webサービスのリソース、リソースに対して提供されるHTTPメソッド、メソッドにおける入力及び出力DTOを定義する。サービスの起動時に、RESTサーバ定義XMLファイルを読み込み、Webサービスのリソースを準備する。

HTTPリクエストのHTTPメソッドに応じて、process～(request,response)が呼び出されると、リクエストされたパスに該当するリソースを特定し、呼びされたメソッドに該当するHTTPメソッドがリソースとして定義されていれば、該当する業務フローを呼び出す。その際、リソースに宣言された入力DTOを生成し、パス、パラメータを入力DTOに設定する。また、入力ストリームでデータが渡ってきた場合は、リクエストされたメディアタイプ（Content-Typeヘッダのメディアタイプ）に該当するBindingStreamConverterを使って、入力DTOに変換して格納する。入力DTOをRestRequestに格納し、さらにRestContextに格納して、業務フローに渡す。

業務フローでは、アプリケーション処理を行った結果を、HTTPステータス、メッセージ、出力DTOとして、RestContext内のRestResponseに設定する。出力DTOが設定されている場合は、リクエストされたメディアタイプ（Acceptヘッダのメディアタイプ）に該当するStreamConverterを使って、出力ストリームに変換して書き込む。

以下に、RESTサーバ定義ＸＭＬファイルの仕様を定義する。

子要素の”+”は、１個以上。”\*”は、０個以上。”?”は0個か１個、現れる事を意味する。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 要素名 | 親要素 | 子要素 | 説明 | |
| 1 | restserver | なし | resource+ | ルート要素。 | |
| 2 | resource | restserver | post?  get?  head?  put?  delete?  options? | RESTサーバに存在するリソースを定義する。 | |
| 属性名 | | 必須/任意 | 説明 |
| name | | 必須 | RESTサーバのリソースを示すURIを定義する。  URI中に、“{パラメータ名}”を指定する事で、パラメータを含める事ができる。 |
| 3 | post | resource | description?  request?  response? | リソースのPOSTメソッドを定義する。 | |
| 4 | get | resource | description?  request?  response | リソースのGETメソッドを定義する。 | |
| 5 | head | resource | description?  request? | リソースのHEADメソッドを定義する。 | |
| 6 | put | resource | description?  request?  response? | リソースのPUTメソッドを定義する。 | |
| 7 | delete | resource | description?  request?  response? | リソースのDELETEメソッドを定義する。 | |
| 8 | options | resource | なし | リソースのOPTIONSメソッドを定義する。 | |
| 9 | description | post  get  head  put  delete | なし | メソッドの説明を定義する。  機能的な意味はない。 | |
| 10 | request | post  get  head  put  delete | なし | メソッドのリクエスト情報を定義する。 | |
| 属性名 | | 必須/任意 | 説明 |
| code | | 必須 | リクエスト情報を格納するBeanのクラス名を定義する。 |
| 11 | response | post  get  put  delete | なし | メソッドのレスポンス情報を定義する。 | |
| 属性名 | | 必須/任意 | 説明 |
| code | | 必須 | レスポンス情報を格納するBeanのクラス名を定義する。 |

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | BeanFlowInvokerFactoryServiceName | ServiceName | jp.ossc.nimbus.service.beancontrol.interfaces.BeanFlowInvokerFactoryサービスのサービス名を設定する。 |
| 2 | JournalServiceName | ServiceName | jp.ossc.nimbus.service.journal.Journalサービスのサービス名を設定する。 |
| 3 | EdiorFinderServiceName | ServiceName | jp.ossc.nimbus.service.journal.editorfinder.EditorFinderサービスのサービス名を設定する。 |
| 4 | ContextServiceName | ServiceName | ジャーナルに設定するリクエストID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。 |
| 5 | SequenceServiceName | ServiceName | ジャーナルに設定するリクエストID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。 |
| 6 | RequestConverterServiceName | Map<String,ServiceName> | メディアタイプ毎に、HTTPリクエストの入力ストリームをDTOに変換するjp.ossc.nimbus.util.converter.Converterサービスのサービス名を設定する。 |
| 7 | ResponseConverterServiceName | Map<String,ServiceName> | メディアタイプ毎に、DTOを変換してHTTPレスポンスの出力ストリームに書き込むためのjp.ossc.nimbus.util.converter.Converterサービスのサービス名を設定する。 |
| 8 | ServerDefinitionPath | String | RESTサーバ定義ファイルのパスを設定する。  パスは、絶対パス、サービス定義ファイルからの相対パス、クラスパスを指定できる。 |
| 9 | DocumentBuilderFactoryClassName | String | RESTサーバ定義ファイルをパースするjavax.xml.parsers.DocumentBuilderFactoryのクラス名を設定する。  指定しない場合は、DocumentBuilderFactory.newInstance()でDocumentBuilderFactoryを生成する。 |
| 10 | Validate | boolean | RESTサーバ定義ファイルを検証するかどうかを設定する。  デフォルトは、falseで検証しない。 |
| 11 | ValidateFlowPrefix | String | リクエストオブジェクト検証フローのフロー名の前置詞を設定する。  デフォルトは、“validate”。 |
| 12 | PostMethodFlowPostfix | String | POSTメソッド処理フローのフロー名の後置詞を設定する。  デフォルトは、“$POST”。 |
| 13 | GetMethodFlowPostfix | String | GETメソッド処理フローのフロー名の後置詞を設定する。  デフォルトは、“$GET”。 |
| 14 | HeadMethodFlowPostfix | String | HEADメソッド処理フローのフロー名の後置詞を設定する。  デフォルトは、“$HEAD”。 |
| 15 | PutMethodFlowPostfix | String | PUTメソッド処理フローのフロー名の後置詞を設定する。  デフォルトは、“$PUT”。 |
| 16 | DeleteMethodFlowPostfix | String | DELETEメソッド処理フローのフロー名の後置詞を設定する。  デフォルトは、“$DELETE”。 |

## jp.ossc.nimbus.servlet.BeanFlowSelector

### jp.ossc.nimbus.servlet.DefaultBeanFlowSelectorService

HTTPリクエストのパス情報を業務フロー名として使用するデフォルト実装クラスである。

http://host/context/path/action.bf

selectBeanFlow(request)

**DefaultBeanFlowSelectorService**

/path/action

リクエストされたURIのうち、コンテキストパスから後ろのパスから、後置詞（デフォルトでは“.bf”）を取り除いたものを、業務フロー名とみなす。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | BeanFlowPathPostfix | String | アクションBeanFlowパスの後置詞を設定する。  デフォルトは、“.bf”。 |

### jp.ossc.nimbus.servlet.MappingBeanFlowSelectorService

HTTPリクエストのパス情報に対して業務フロー名をマッピングする実装クラスである。

また、DefaultBeanFlowSelectorServiceのサブクラスとなっているため、マッピング定義されていないパスがリクエストされた場合は、親クラスの機能を利用する。

http://host/context/hoge.html

selectBeanFlow(request)

**MappingBeanFlowSelectorService**

業務フロー名

パス

action/hoge

/hoge.html

action/hoge

action/fuga

/fuga.html

リクエストされたURIのうち、コンテキストパスから後ろのパスを取得し、そのパスとマッピングされている業務フロー名を返す。パスのマッチングには、正規表現を利用する事もできる。また、マッピング定義に該当する業務フロー名が存在しない場合は、親クラスの機能を利用する。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | BeanFlowPathPostfix | String | アクションBeanFlowパスの後置詞を設定する。  デフォルトは、“.bf”。 |
| 2 | Mapping | Properties | リクエストパスとアクションフロー名のマッピングを設定する。  リクエストパス=アクションフロー名 |
| 3 | RegexEnabled | boolean | リクエストパスとアクションフロー名のマッピングにおいて、リクエストパスに正規表現を指定できるようにするかどうかを設定する。  デフォルトは、falseで正規表現を使用しない。 |
| 4 | RegexMatchFlag | int | 正規表現比較を行う場合に使用するマッチフラグを設定する。  但し、RegexEnabled属性がtrueの場合のみ有効である。  デフォルトは、0。 |