intra-mart WebPlatform/AppFramework Ver.7.0

ポートレット　プログラミングガイド

2009/06/30 第 4 版

＜＜　変更履歴　＞＞

変更年月日	変更内容
2008/07/07	初版
2008/09/30	第 2 版 「３．２　汎用新着ポートレットについて」インターフェースに関する詳細な内容を追記しました。
2009/02/27	第 3 版 「６．１　画面を作成する上での注意事項」にスタイルの注意事項を追記しました。
2009/06/30	第 4 版 「２．２．１　JSP/Servlet を利用したポートレット開発」を追加し、Struts についての注意事項を追記しました。

<< 目次 >>

1 はじめに 1
1.1 目的 1
2 概要 2
2.1 intra-martポータルモジュールについて 2
2.2 ポートレットの種類 2
2.2.1 JSP/Servletを利用したポートレット開発 3
2.3 ポートレット用画面 3
3 ポートレットの開発（スクリプト開発編） 5
3.1 画面開発 5
3.1.1 通常の画面とポートレット画面を共用するには 6
3.2 汎用新着ポートレットについて 6
3.2.1 「重要なお知らせ」ポートレットについて 9
4 ポートレットの開発（JavaEE開発編） 10
4.1 画面開発 10
4.1.1 通常の画面とポートレット画面を共用するには 11
4.2 汎用新着ポートレットについて 12
4.2.1 「重要なお知らせ」ポートレットについて 13
4.2.2 スクリプト開発モデルによるプロバイダの実装 14
5 ポートレットの開発（JSR168 編） 15
5.1 画面開発 15
6 注意事項 19
6.1 画面を作成する上での注意事項 19

1 はじめに

1.1 目的

このドキュメントは、intra-mart のポータル画面に表示して利用することが可能なポートレットモジュールを作成する方法について説明します。

本書では、ポートレットプログラミングの概要と、ポートレットを作成する上での注意事項を主に取り扱います。実際に動作可能なポートレットアプリケーションを作成するためには、それぞれ開発モデルの言語仕様を理解して実装を行う必要があります。

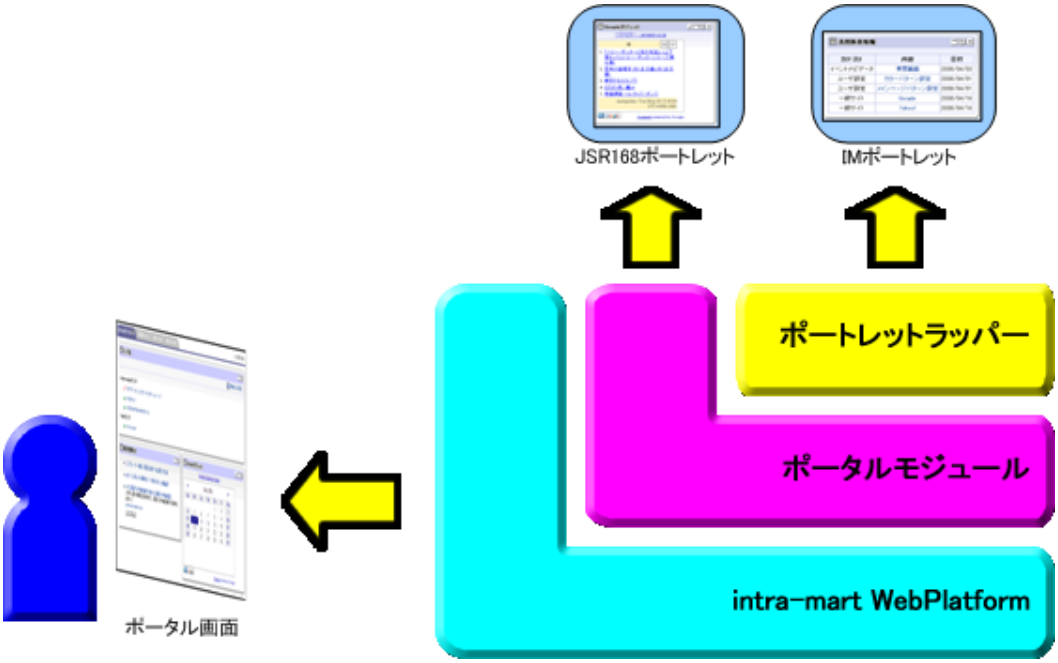
2 概要

2.1 intra-martポータルモジュールについて

ポータルモジュールは、JSR168 で規定されるポートレットコンテナとして動作します。
このため、JSR168 に準拠したポートレットであれば、そのまま利用することができます。

- JSR168
  Java 標準化団体（Java Community Process）にて策定された、ポートレットに関する標準仕様。
  ポートレットおよびポートレットコンテナとして必要な機能の定義と、ポートレットを作成するための API（Portlet API 1.0）が含まれます。

また、intra-mart アプリケーションをポートレットとして利用するためのラッパーが標準で準備されているため、旧バージョンまでで利用していたポートレットもそのまま利用することが可能です。



＜intra-mart ポータルモジュール構成＞

2.2 ポートレットの種類

ポータルモジュールで利用可能なポートレットは大きくわけて2つに分類されます。

1. intra-mart アプリケーションとして作成されたポートレット
2. JSR168 に準拠したポートレット

また、前者の場合開発モデルによってさらに2つに分類されます。

＜ポートレットの種類＞

	ポートレットの種類	開発モデル
1	intra-mart アプリケーションとして作成されたポートレット	JavaEE 開発モデル
2		スクリプト開発モデル
3	JSR168 に準拠したポートレット	JavaEE Web アプリケーション

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

次章では、これらのポートレット種別ごとにポートレットを開発する方法を説明します。

2.2.1 JSP/Servletを利用したポートレット開発

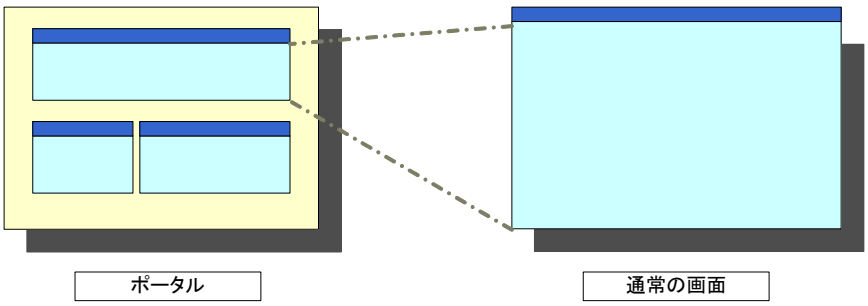
上記以外に、intra-mart のフレームワークを利用せず、JSP や Servlet で開発した Web ページをポートレットとして登録することも可能です。
これらを作成する場合は、標準的な Web アプリケーションの開発方法で作成することができます。

ただし、Struts のような Web アプリケーションフレームワークは、通常の Web ページを作成するためのフレームワークであるため、ポートレットを作成するためには適しておりません。
ほとんどの場合、そのままポートレットとして利用することはできませんので、ポートレット用に修正する必要があります。
具体的には以下の点に注意して修正してください。

- ポートレットはポータル画面からINCLUDEして呼び出されるため、ポートレット処理中にFORWARD処理することができません。
  Struts では、特に処理を記述しない限り自動的に FORWARD されるため、必ず Action クラス内で、処理内で RequestDispatcher#include()を呼び出すようにしてください。
- その他、struts-config.xml 内での FORWARD 処理も利用できませんので注意してください。
- また、ServletFilter のターゲットは FORWARD のみですので、もし Filter を利用する必要があるばあい、INCLUDE をターゲットに追加するようにしてください。

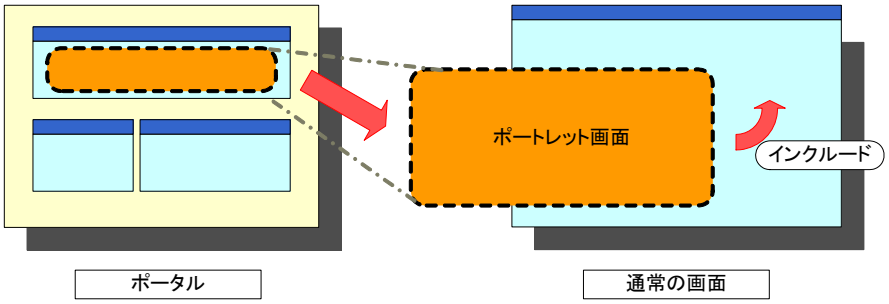
2.3 ポートレット用画面

ポートレットの画面はポータル画面の一部として表示されるため、通常の画面と同様に作成することはできません。
詳細は「6.1画面を作成する上での注意事項」を参照してください。



ポータル画面と通常画面との比較

そのため、通常の画面とポートレット画面で共通の画面を利用したい場合でも、別々に画面を作成する必要があるます。ただし、画面表示や機能は共通にできるため、以下のようにポートレット画面をインクルードするような画面を作成することで、通常の画面用に画面を作り直す必要はなくなります。



通常の画面とポートレット画面を共用する

実装方法については、それぞれの開発モデルごとに例を示します。項「3.1.1」(スクリプト開発モデルの場合)または項「4.1.1」(JavaEE開発モデルの場合)を参照してください。

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

3 ポートレットの開発（スクリプト開発編）

3.1 画面開発

スクリプト開発モデルの画面は、おおむね通常の intra-mart の画面開発と変わりません。
ポートレット新規登録／編集画面（「ポータル グループ管理者操作ガイド」参照）で設定されたページ引数は、init 関数の request オブジェクトより取得することができます。
ただし、スクリプト開発モデルで作成されたポートレットからサブミット処理を行い、 処理終了後に再びポータル画面を表示する機能を作成する場合には、以下の制約があります。

- サブミット処理後にポータル画面へ URL リダイレクトを行う。
- request オブジェクトにポータルコードが設定されている場合は、表示するポータルの情報を設定する。
- 遷移先のウインドウ名 "IM_MAIN" を target 属性で指定する。

以下に、スクリプト開発モデルで開発されたポートレットでサブミットされた後に、再びポータル画面を表示するサンプルを示します。

① プレゼンテーションページ

```
<!― ウインドウを target 属性で指定 ―>
<IMART type="form" action="insertData" target="IM_MAIN">

    <!― サブミット後に表示するポータルの情報 ―>
    <IMART type="hidden" portal_cd=portal_cd></IMART>

    <!-- サブミット処理で使用する値 -->
    <INPUT type="text" name="param" value="">

    <INPUT type="submit" value="実行">
</IMART type="form">
```

② ファンクションコンテナ

```
// ポータルコード
var portal_cd = "";

function init(request) {
    if (!isUndefined(request.portal_cd)) {
        // ポータルコードをリクエストから取得する。
        portal_cd = request.portal_cd;
    }
}

function insertData(request) {

    // 登録処理で使用するパラメータを取得
    var paramValue = request.param;

    // 何らかの登録処理を実行
    ....

    // ポータル画面遷移の URL を取得
    var portalUrl = getURL(request.portal_cd);
```

```
    // ポータル画面へリダイレクト
    redirect(portalUrl);
}

function getURL(portalCode) {

    // ポータル画面の URL の生成
    var url = "portal-portal_display.service";

    if (portalCode != '') {
        // 表示するポータルの情報(ポータルコード)を設定する。
        url += ("?portal_cd=" + portalCode);

        url += "&actionId=switch";
    }

    return url;
}
```

3.1.1 通常の画面とポートレット画面を共用するには

共用するコードと通常の画面のみ必要なコードを別々のプレゼンテーションページとして作成し、通常の画面用のプレゼンテーションページでは、include タグを利用して共用ページを読み込みます。

<ポートレット画面と共用する通常画面のプレゼンテーションページ>

```
<html>
<head>
  <link src="sample.css" type="text/css">

</head>
<body>
  <div>
    ヘッダ表示情報
  </div>

  <!-- 共用コードをインクルードする -->
  <IMART type="include" page="sample/sample-portlet.jsp"></IMART>

  <div>
    フッタ表示情報
  </div>
</body>
</html>
```

3.2 汎用新着ポートレットについて

intra-mart ポータルモジュールには、汎用的な新着情報を表示するポートレットが用意されています。 この汎用新着ポートレットでは、新着情報として以下の項目を表示します。

- 新着情報の種類を示すためのカテゴリ
- 新着情報の内容(概要)
- 配信日付
- 新着情報に紐付く遷移先の情報(リンク)

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

汎用新着ポートレットはJavaEE開発モデルを利用して作成されていますが、スクリプト開発モデルで利用できるようにインターフェースが用意されています。
汎用新着ポートレットに独自の新着情報を表示させるためには、下記の手順でポートレットを登録します。

1. 新着情報を取得するプロバイダを実装する。

汎用新着ポートレットに表示される情報の配列を取得する以下のファンクションを実装したファンクションコンテナを作成します。

- ファンクション名 ： getNewArrivedList
- パラメータ

パラメータ名	JavaScript の型	内容
user	String	ユーザ ID
group	String	ログイングループ ID
properties	Object	設定ファイルに定義された、初期パラメータ

このファンクションは、以下の新着情報オブジェクトの配列を返却する必要があります。

- 新着情報オブジェクト

プロパティ	JavaScript の型	内容
category	String	新着情報の種類を示すためのカテゴリ
contents	String	新着情報の内容（概要）
url	String	新着情報の詳細情報へ遷移する URL（リンク）
popup	Boolean	詳細情報へ遷移する際にポップアップするかどうかのフラグ
date	String	新着情報の配信日付

<sample_provider.js>

```
// getNewArrivedList 関数を実装します。
function getNewArrivedList(user, group, properties) {

  // 新着情報の配列を作成します。
var values = new Array();

  for (var i = 0; i < result.countRow; i++) {
        values[i] = new Object();
        values[i].category = result.data[i].category;   // 新着情報の種類を示すためのカテゴリ
        values[i].contents = result.data[i].contents;   // 新着情報の内容（概要）
        values[i].date = result.data[i].arrived_date;   // 配信日付
        values[i].url = result.data[i].url;             // 新着情報に紐付く遷移先の情報（リンク）
        values[i].popup = ("1" == result.data[i].popup);        // リンクへ参照時にポップアップを呼び出すかどうか
  }

  // 新着情報の配列を返却します。
  return values;
}
```

2. プロバイダの実装を定義する。

実装したプロバイダクラスについての情報は、Storage Service 配下の任意の場所に設定ファイルを XML 形式で作成して定義します。 設定ファイルには以下の情報を定義します。

<プロバイダ設定ファイル>

要素	説明
new-arrived-portlet/sort	取得した新着情報をカテゴリでソートするかどうかを定義します。
new-arrived-portlet/provider/provider-class	プロバイダの実装クラス。固定です。
new-arrived-portlet/provider/init-param	新着情報を取得するファンクションコンテナを「pagePath」パラメータに定義します。 それ以外の任意のパラメータを設定して、ファンクションコンテナで利用することが可能です。

たとえば、上記のスクリプト開発モデルプログラムより新着情報を取得するプロバイダを使って新着情報を取得する場合は、以下のように定義を行います。

<sample_portlet.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <!-- スクリプト開発モデル用のプロバイダクラス。固定とする -->
    <provider-class>
      jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider
    </provider-class>
    <init-param>
      <!-- プロバイダを定義する -->
      <param-name>pagePath</param-name>
      <param-value>/hoge/sample_provider.js</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```

プロバイダを複数登録して、複数の新着情報を1つのポートレットに表示することも可能です。
設定ファイルについての詳細な定義については、「intra-mart WebPlatform/AppFramework ポータル設定ガイド」を参照して下さい。

3. ポートレットを登録する。

グループ管理者の[ポータル] - [ポートレット]編集画面で、新着情報を表示するためのポートレットを登録します。
汎用新着ポートレットを設定する場合は、「ページ種別」、「画面パスあるいは URL 種類」、「ページ引数」の設定が必要になります。たとえば、サンプルでは、以下のように設定されます。

- ページ種別 : Servlet/JSP
- 画面パスあるいは URL 種類 : /portal/portlets/newly_arrived_view.jsp
- ページ引数 : page_param=/portal/sample_portlet.xml

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

「画面パスあるいは URL 種類」で指定された JSP には、プロバイダが取得した新着情報を表示するための JSP などを指定します。サンプルで指定している JSP は、ポータルモジュールで用意されたデフォルトの実装です。
「ページ引数」には、手順2で作成した、プロバイダの情報を定義した設定ファイルのパスを page_param= ... の形式で指定します。para_param の値には、Storage Service のルートディレクトリ配下のパスを指定します。

3.2.1 「重要なお知らせ」ポートレットについて

「重要なお知らせ」ポートレットは、汎用新着ポートレットの仕組みを使って実装されたポートレットです。
現在は JavaEE 開発モデルで作成された以下のプロバイダクラスが使用され、パスワード履歴管理の情報のみが扱われています。

- jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider

その定義は、Storage Service 配下の設定ファイル portal/system_notice.xml で行われています。

JavaEE 開発モデルとスクリプト開発モデルのプロバイダも混在可能ですので、スクリプト開発モデルでプロバイダクラスを実装して設定ファイルへ定義を追加すれば、パスワード履歴以外の情報も「重要なお知らせ」として扱うことができるようになります。

<重要なお知らせポートレット定義ファイル：system_notice.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
    </provider-class>
  </provider>
</new-arrived-portlet>
```

4 ポートレットの開発（JavaEE 開発編）

4.1 画面開発

JavaEE 開発モデルあるいは JavaEE フレームワークの開発は、おおむね通常の intra-mart の画面開発と変わりません。
ポートレット新規登録／編集画面（「intra-mart WebPlatform/AppFramework ポータル グループ管理者操作ガイド」参照）で設定されたページ引数は、HttpServletRequest オブジェクトの getParameter メソッドより取得することができます。
ただし、JavaEE 開発モデルで作成されたポートレットからサブミット処理を行い、 処理終了後に再びポータルを表示する機能を作成する場合には、以下の制約があります。

- サブミット処理後にポータル画面を表示するサービスを呼び出す。
- HttpServletRequest オブジェクトにポータルコードが設定されている場合は、表示ポータルの情報としてポータルコードなどの情報を引き渡す。
- 遷移先のウインドウ名前 "IM_MAIN" を target 属性で指定する。

以下に、JavaEE フレームワークで開発されたポートレットでサブミットされた後に、再びポータル画面を表示するサンプルを示します。

① **JSP**

```
<%@ taglib prefix="portal"
      uri="http://www.intra-mart.co.jp/taglib/core/framework" %>

<%
   String portalCd = "";
   if (request.getParameter("portal_cd") != null) {
         // ポータルコードをリクエストから取得する。
         portalCd = request.getParameter("portal_cd");
   }
%>

<!-- 何らかの登録処理を行うサービスを呼び出す -->
<!-- ウインドウを target 属性で指定 -->
<portal:Form application="sample" service="post_action" target="IM_MAIN">

   <!-- 登録処理後に表示するポータルの情報 -->
   <input type="hidden" name="portal_cd" value="<%= portalCd %>">
   <input type="hidden" name="account" value="<%= (portalCd != "") ? actionId : "" %>">

   <!-- 登録処理で使用する値 -->
   <input type="text" name="param" value="">

   <input type="submit" value="実行">
</portal:Form>
```


Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

② 登録処理を行うサービスの定義

```
<?xml version="1.0" encoding="UTF-8"?>
<service-config>
  <service>
    <!-- 登録処理を行うコントローラ -->
    <service-id>post_action</service-id>
    <controller-class>
      sample.portal.PortalSampleServiceController
    </controller-class>

    <!-- ポータル画面遷移のトランジション -->
    <transition-class>
      sample.portal.PortalReloadTransition
    </transition-class>
  </service>
</service-config>
```

③ ポータル表示のサービスを呼び出すトランジション

```
package sample.portal;
import jp.co.intra_mart.framework.base.service.DefaultTransition;

public class PortalReloadTransition extends DefaultTransition {

    public String getNextPage() {
        // ポータルを表示する URL を返す
        return "portal-portal_display.service";
    }
}
```

4.1.1 通常の画面とポートレット画面を共用するには

共用するコードと通常の画面のみ必要なコードを別々の JSP ファイルとして作成し、通常の画面用の JSP ファイルでは、jsp:include タグを利用して共用ページを読み込みます。

<ポートレット画面と共用する通常画面の JSP>

```
<html>
<head>
  <link src="sample.css" type="text/css">

</head>
<body>
  <div>
    ヘッダ表示情報
  </div>

  <!-- 共用コードをインクルードする -->
  <jsp:include page="sample/sample-portlet.jsp" flush="true"/>

  <div>
    フッタ表示情報
  </div>
</body>
</html>
```

4.2 汎用新着ポートレットについて

intra-mart ポータルモジュールには、汎用的な新着情報を表示するポートレットが用意されています。 この汎用新着ポートレットでは、新着情報として以下の項目を表示します。

- 新着情報の種類を示すためのカテゴリ
- 新着情報の内容(概要)
- 配信日付
- 新着情報に紐付く遷移先の情報(リンク)

汎用新着ポートレットに独自の新着情報を表示させるためには、下記の手順でポートレットを登録します。

1. 新着情報を取得するプロバイダを実装する。

新着ポートレットに表示される情報は、以下のインターフェースを実装したプロバイダより取得されます。

- 汎用新着情報のプロバイダーのインターフェース
  jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.NewArrivedProvider

ポータルモジュールでは、汎用新着ポートレットを利用したポートレットとして、重要なお知らせポートレットが用意されており、以下のプロバイダを利用して、パスワード履歴管理情報を表示しています。独自のプロバイダを実装する場合の参考にしてください。

- パスワード履歴管理汎用新着情報のプロバイダ
  jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider

プロバイダのインターフェースおよびプロバイダの実装についての詳細は API リストを参照して下さい。

2. プロバイダの実装を定義する。

実装したプロバイダクラスについての情報は、Storage Service 配下の任意の場所に設定ファイルを作成して定義します。 設定ファイルには以下の情報を定義します。

<プロバイダ設定ファイル>

要素	説明
new-arrived-portlet/sort	取得した新着情報をカテゴリでソートするかどうかを定義します。
new-arrived-portlet/provider/provider-class	プロバイダの実装クラスを定義します。
new-arrived-portlet/provider/init-param	プロバイダクラスに設定する初期化パラメータを定義します。

以下に設定ファイルの記述例を示します。

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

<sample_portlet.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <provider-class>
      sample.SampleProvider
    </provider-class>
    <init-param>
      <param-name>param1</param-name>
      <param-value>hogehoge</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```

プロバイダを複数登録して、複数の新着情報を1つのポートレットに表示することも可能です。
設定ファイルについての詳細な定義については、「intra-mart WebPlatform/AppFramework ポータル設定ガイド」を参照して下さい。

3. ポートレットを登録する。

グループ管理者の[ポータル] - [ポートレット]編集画面で、新着情報を表示するためのポートレットを登録します。
汎用新着ポートレットを設定する場合は、「ページ種別」、「画面パスあるいは URL 種類」、「ページ引数」の設定が必要になります。たとえば、サンプルでは、以下のように設定されます。

- ページ種別 ： Servlet/JSP
- 画面パスあるいは URL 種類 ： /portal/portlets/newly_arrived_view.jsp
- ページ引数 ： page_param=/portal/sample_portet.xml

「画面パスあるいは URL 種類」で指定された JSP には、プロバイダが取得した新着情報を表示するための JSP などを指定します。サンプルで指定している JSP は、ポータルモジュールで用意されたデフォルトの実装です。
「ページ引数」には、手順2で作成した、プロバイダの情報を定義した設定ファイルのパスを page_param= ... の形式で指定します。para_param の値には、Storage Service のルートディレクトリ配下のパスを指定します。

4.2.1 「重要なお知らせ」ポートレットについて

「重要なお知らせ」ポートレットは、汎用新着ポートレットの仕組みを使って実装されたポートレットです。
現在は以下のプロバイダクラスが使用され、パスワード履歴管理の情報のみが扱われています。

- jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider

その定義は、Storage Service 配下の設定ファイル portal/system_notice.xml で行われています。

プロバイダクラスを実装して設定ファイルへ定義を追加すれば、パスワード履歴以外の情報も「重要なお知らせ」として扱うことができるようになります。

<定義ファイル：system_notice.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
    </provider-class>
  </provider>
</new-arrived-portlet>
```

4.2.2 スクリプト開発モデルによるプロバイダの実装

ポータルモジュールでは、スクリプト開発モデルプログラムより新着情報を取得する以下のプロバイダの実装クラスがあらかじめ用意されています。

- jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider

JavaEE 開発モデルとスクリプト開発モデルのプロバイダも混在可能ですので、スクリプト開発モデルでプロバイダクラスを実装して設定ファイルへ定義を追加すれば、ポートレットに表示する汎用新着情報を追加することが可能となります。

<スクリプト開発モデルによるプロバイダ定義の例>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <!-- スクリプト開発モデル用のプロバイダクラス。固定とする -->
    <provider-class>
      jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider
    </provider-class>
    <init-param>
      <!-- スクリプト開発モデルで作成したプロバイダを定義する -->
      <param-name>pagePath</param-name>
      <param-value>/hoge/sample_provider.js</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```


Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

5 ポートレットの開発（JSR168 編）

5.1 画面開発

JSR168 準拠のポートレットは、サーブレットに類似したモジュールとして定義されており、サーブレットを利用する Web アプリケーションと同様にして作成され、WAR ファイルとしてポータルサーバに配備されて利用されます。
ただし、ポートレットの定義情報は WEB-INF 配下の portlet.xml に記述されます。
それ以外は通常の Web アプリケーションを作成する手順で開発を行いますが、サーブレットではないため、サーブレットを利用する Jakarta Struts のようなフレームワークは利用できません。

以下に、JSR168 対応のポートレットの作成方法を簡単に記述します。
JSR168 に関する詳細な情報は専門のドキュメントおよびAPIドキュメント（Portlet API 1.0）を参照してください。

- http://jcp.org/en/jsr/detail?id=168（JSR168 仕様のページ）

1. Portlet クラスの作成

JSR168 準拠のポートレットは、javax.portlet.Portlet インターフェースを実装する必要があります。
ただし PortletAPI では、ポートレットを簡単に作成するために基本的な処理を実装した GenericPortlet クラスが用意されています。

```
package sample.portlet;
import javax.portlet.GeneralPortlet;

// GenericPortlet を継承する。
public class SamplePortlet extends GenericPortlet {

  // GenericPortlet ではモードごとに表示処理メソッドが呼ばれる。
  // VIEW モードでの処理を記述する。
  protected void doView(RenderRequest request, RenderResponse response)

      // VIEW モードでの処理を記述する。

     // JSP を呼び出す。
     // ServletContext の代わりに、PortletContext を利用する。
     PortletContext context = getPortletContext();
     // RequestDispatcher の代わりに、PortletRequestDispatcher を利用する。
     PortletRequestDispatcher rd = context.getRequestDispatcher("/WEB-INF/jsp/sample_viewjsp");
     // ポートレットでは、forward メソッドは利用できない。Include のみ利用可能。
     rd.include(request, response);
  }

  // EDIT モードでの処理を記述する。
  protected void doEdit(RenderRequest request, RenderResponse response)

      // EDIT モードでの処理を記述する。
  }

  // HELP モードでの処理を記述する。
  protected void doHelp(RenderRequest request, RenderResponse response)

      // HELP モードでの処理を記述する。
```

```
    }

    // サブミット時の処理を記述する。
    public void processAction(ActionRequest request, ActionResponse response)

        // サブミット時の処理を記述する。
        // この後、表示メソッドが自動的に呼ばれるため、JSP の呼び出しは不要。
    }

}
```

2. JSP の作成

```
// portlet タグライブラリが利用できます。
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>

<%
    // ポートレットでは、HttpServletRequest、HttpServletResponse の代わりに、
    // RenderRequest、RenderResponse を利用します。
    // 使い方はほぼ同じです。
    RenderRequest rRequest = (RenderRequest)request.getAttribute("javax.portlet.request");
    RenderRespnse rResponse = (RenderResponse)request.getAttribute("javax.portlet.response");

    //リクエストからパラメータを取得する。
    String param = request.getParameter("param");


    // ポータル画面を再表示するための URL を取得します。
    // ポータル画面を再表示するための URL(Render URL)の取得。
    PortletURL renderURL = rResponse.createRenderURL();

    // サブミット処理を行ってから、ポータル画面を再表示するための URL(Action URL)の取得。
    PortletURL actionURL = rResponse.createActionURL();
%>

<!-- 何らかの登録処理を行うサービスを呼び出す -->
<!-- Action URL を設定 -->
<!-- ウインドウを target 属性で指定 -->
<form action="<%= actionURL.toString() %>" name="sample_portlet_form" target="IM_MAIN" method="POST">

    <!-- 登録処理で使用する値 -->
    <input type="text" name="param" value="">

    <input type="submit" value="実行">
</form>
```

■ actionURL はポータル情報を含むため、データ量が多くなります。メソッドは POST を指定するようにしてください。

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

3. portlet.xml の作成

portlet.xml には、ポートレットの定義情報を記述します。
詳しい記述方法については、以下より JSR168 のドキュメントを取得して参照してください。

- http://jcp.org/en/jsr/detail?id=168

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app
    xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd /opt/SUNWportal/dtd/portlet.xsd"
    version="1.0">

    <portlet>
        <portlet-name>SamplePortlet</portlet-name>
        <portlet-class>sample.SamplePortlet</portlet-class>
        <expiration-cache>300</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
        </supports>
        <supported-locale>ja</supported-locale>
        <portlet-info>
            <title>Sample Portlet</title>
        </portlet-info>
    </portlet>

</portlet-app>
```

4. web.xml の作成

web.xml の記述は必要ありませんが、Web アプリケーションとして AP サーバにデプロイするため、空のファイルを格納します。
追加情報を記述することも可能ですが、その場合、サーブレット API によって web.xml の記述が異なります。
バージョン情報を取得できるようにバージョンの定義情報は省略しないで下さい。

- Servlet2.3　：DOCTYPE による指定
- Servlet2.4～　：XMLSchema による指定

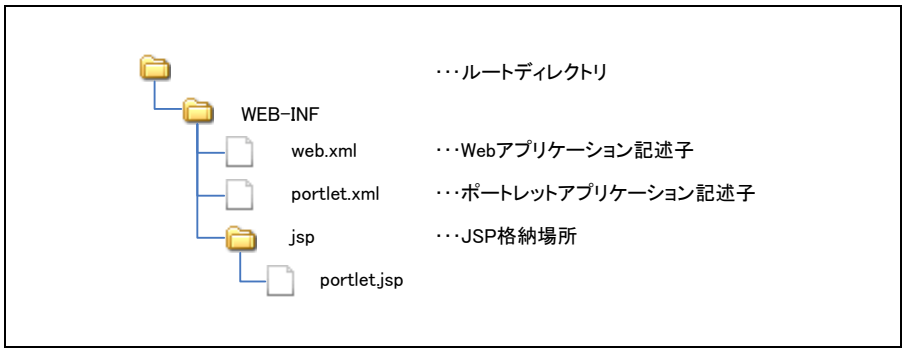
```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Servlet 2.4 の場合 -->
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
        version="2.4">

</web-app>
```

5. WAR ファイルの作成

作成したファイルをまとめて、WAR ファイルを作成してポートレットアプリケーションが完成します。
WEB-INF の中には、web.xml、portlet.xml の2つのデプロイメント記述子が格納されます。
また、JSR168 では、ポートレットを直接呼び出してはいけないと定義されていますが、AP サーバに配置した場合は JSP に直接アクセスできてしまいます。
そこで、ポートレットアプリケーションでは通常 WEB-INF 配下に JSP を格納します。



＜WARファイルの構造＞

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

6 注意事項

6.1 画面を作成する上での注意事項

ポートレットも Web アプリケーションですので、基本的には通常の開発方法と変わりません。
しかし、結果として表示されるポートレットの画面はポータル画面の一部として表示されるため、画面を作成する上で以下のような制約事項があります。

1. ポートレットは HTML の一部分として表示されるため、HTML 構造を表す以下のようなタグは利用できません。<BODY>タグ内に表示されるドキュメントの内容のみ記述するようにして下さい。
ブラウザによっては、以下のタグを記述しても正常に表示されますが、推奨はされません。

- <HEAD>
- <BODY>
- <!DOCTYPE>

2. 画面のスタイルはポータル画面で指定されます。前項に記述したように<HEAD>タグが利用できないため、スタイルクラスの定義は記述できません。タグに個別にスタイルを設定することもできますが、できるだけスタイルに依存しないシンプルな画面作成をして下さい。

- ＜STYLE>タグや<LINK>タグを用いてスタイルを指定することによって、ポータル画面のレイアウトに影響を与えることがあります。
- <IMART type="imDesignCss">／< imarttag:imartDesignCss>タグを利用した場合、動的にカラーパターンが変更されますが、ポートレットのキャッシュ機能を有効にした場合、カラーパターンの変更が反映されなくなります。
これらのタグはポータル画面で指定しており、再指定しなくても利用可能ですので、ポートレットには記述しないようにしてください。

3. JavaScript で利用するオブジェクト名や変数名、関数名をユニークな名前で定義する必要があります。
これは通常の画面開発でも同様ですが、ポートレットの場合、ポータル画面上の全てのポートレットでバッティングしないようにする必要があるため、特に注意が必要です。
また公開された共通の JavaScript を利用する場合も、そのバージョンが一致しないと思わぬ動作をする可能性があるため、注意する必要があります。ポータル画面では、以下の共通ライブラリを利用しているため、ポートレットで利用する場合は、同じバージョンのライブラリを利用するようにして下さい。

- prototype.js v1.6.0.1
- builder.js v1.8.1
- effects.js v1.8.1
- dragdrop.js v1.8.1

4. タグの ID 属性やコントロールの NAME 属性をユニークな名前で定義する必要があります。全ての ID にアプリケーションに割り当てられるプリフィックスをつけるなどして下さい。
また、以下のプリフィックスはシステムおよび intra-mart が提供するアプリケーションが使用するため、利用できません。

- "IM_"
- "_"（アンダースコア）

5. ポータル画面には複数のページが表示されるため、処理に時間のかかるページを作成すると、ポータル画面が表示されるまでにそれだけ時間がかかってしまいます。
できるだけシンプルな画面を作成することをお勧めします。

Copyright 2000-2009 株式会社 NTT データ イントラマート All rights Reserved.

intra-mart WebPlatform/AppFramework Ver.7.0
ポートレット　プログラミングガイド

2009/06/30 第4版

Copyright 2000-2009 株式会社NTTデータ イントラマート
All rights Reserved.

TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/