HITACHI
Inspire the Next

uCosminexus Portal Framework

ポートレット開発ガイド

解説・文法書

3020-3-H73-50

■対象製品

P-2443-7394 uCosminexus Portal Framework 09-00（適用 OS：Windows Server 2008 x86，Windows Server 2008 x64，Windows Server 2008 R2，Windows Server 2012，Windows Vista，Windows XP，Windows 7，Windows 8）
P-2943-7394 uCosminexus Portal Framework 09-00（適用 OS：Windows Server 2008 x64，Windows Server 2008 R2，Windows Server 2012）

■輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

■商標類

■マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

正式名称	略称
Microsoft(R) Internet Explorer(R)	Internet Explorer
Windows(R) Internet Explorer(R)	
Microsoft(R) Windows(R) 7 Enterprise	Windows 7
Microsoft(R) Windows(R) 7 Professional	

正式名称	略称
Microsoft(R) Windows(R) 7 Ultimate	
Microsoft(R) Windows(R) 8 (Core Edition)	Windows 8
Microsoft(R) Windows(R) 8 Pro	
Microsoft(R) Windows(R) 8 Enterprise	
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit 日本語版	Windows Server 2008 x86
Microsoft(R) Windows Server(R) 2008 Standard 32-bit 日本語版	
Microsoft(R) Windows Server(R) 2008 Enterprise 日本語版	Windows Server 2008 x64
Microsoft(R) Windows Server(R) 2008 Standard 日本語版	
Microsoft(R) Windows Server(R) 2008 R2 Enterprise 日本語版	Windows Server 2008 R2
Microsoft(R) Windows Server(R) 2008 R2 Standard 日本語版	
Microsoft(R) Windows Server(R) 2012 Standard	Windows Server 2012
Microsoft(R) Windows Server(R) 2012 Datacenter	
Microsoft(R) Windows Vista(R) Business	Windows Vista
Microsoft(R) Windows Vista(R) Enterprise	
Microsoft(R) Windows Vista(R) Ultimate	
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP

このマニュアルでは，Windows Server 2008 を前提として操作方法を説明しています。ご使用の環境が異なる場合は，操作方法を読み替えてください。なお，特に断りのない場合は，Windows 7，Windows 8，Windows Server 2008 x86，Windows Server 2008 x64，Windows Server 2008 R2，Windows Server 2012，Windows Vista および Windows XP を総称して Windows と表記しています。

■発行

2013 年 6 月　3020-3-H73-50

■著作権

変更内容

変更内容 (3020-3-H73-50)　uCosminexus Portal Framework　09-00

本文についての変更はありません。

はじめに

このマニュアルは，次のプログラムプロダクトに対応するポートレットの開発方法，および開発時に使用する API について説明したものです。

- uCosminexus Portal Framework

uCosminexus Portal Framework は，企業ポータルを構築および運用するソフトウェアです。

■対象読者

このマニュアルは，uCosminexus Portal Framework に対応するポートレットを開発される方を対象としています。また，次の知識を持っていることを前提としています。

- Cosminexus 上でサーブレットや JSP を運用できる知識
- HTML および XML の知識

■マニュアルの構成

このマニュアルは，次に示す編，章と付録から構成されています。

第 1 編　概要編

第 1 章　uCosminexus Portal Framework のポートレット

uCosminexus Portal Framework で使用できるポートレット，ポートレットの開発の流れおよび uCosminexus Portal Framework の特定の機能に対応するポートレットの概要について説明しています。

第 2 編　ポートレット開発編

第 2 章　ポートレット全般の前提知識

すべてのポートレットを作成する際に，前提となる知識について説明しています。

第 3 章　カスタムポートレットの作成

カスタムポートレットを作成する際の手順，および注意事項について説明しています。

第 4 章　File ポートレットの作成

File ポートレットを作成する際の手順，および注意事項について説明しています。

第 5 章　Web ポートレットの作成

Web ポートレットを作成する際の手順，前提知識，および注意事項について説明しています。

第 6 章　分散ポートレットの作成

分散ポートレットを作成する際の手順，および注意事項について説明しています。

第 3 編　特定の機能に対応するポートレットの開発編

第 7 章　シングルサインオン対応ポートレットの開発

シングルサインオン機能に対応したポートレットの開発方法について説明しています。

第 8 章　ポートレットイベント対応ポートレットの開発

ポートレットイベント機能に対応したポートレットの開発方法について説明しています。

第 9 章　クライアントサイドデータ通信対応ポートレットの開発

クライアントサイドデータ通信機能に対応したポートレットの開発方法について説明しています。

第 10 章　ナビゲーションメニュー対応ポートレットの開発

ナビゲーションメニューにメニューとして登録できるポートレットの開発方法について説明しています。

第 11 章　言語およびタイムゾーン切り替え対応ポートレットの開発

言語およびタイムゾーンの切り替え機能に対応したポートレットの開発方法について説明しています。

第 12 章　新規ウィンドウ対応ポートレットの開発

ナビゲーションメニューから表示する場合などに新規ウィンドウに表示できるポートレットの開発方法について説明しています。

第 13 章　カスタマイズ情報の任意保存対応ポートレットの開発

カスタマイズ情報の任意保存機能に対応したポートレットの開発について説明しています。

第 4 編　ライブラリ編

第 14 章　ライブラリ

uCosminexus Portal Framework で提供するライブラリについて説明しています。

付録 A　ポートレットのサンプル

uCosminexus Portal Framework で作成できるポートレットの開発例について説明しています。

付録 B　セルフユーザ登録のサンプルポートレット

セルフユーザ登録のサンプルポートレットについて説明しています。

付録 C　各バージョンの変更内容

このマニュアルでの各バージョンの変更内容について説明しています。

付録 D　このマニュアルの参考情報

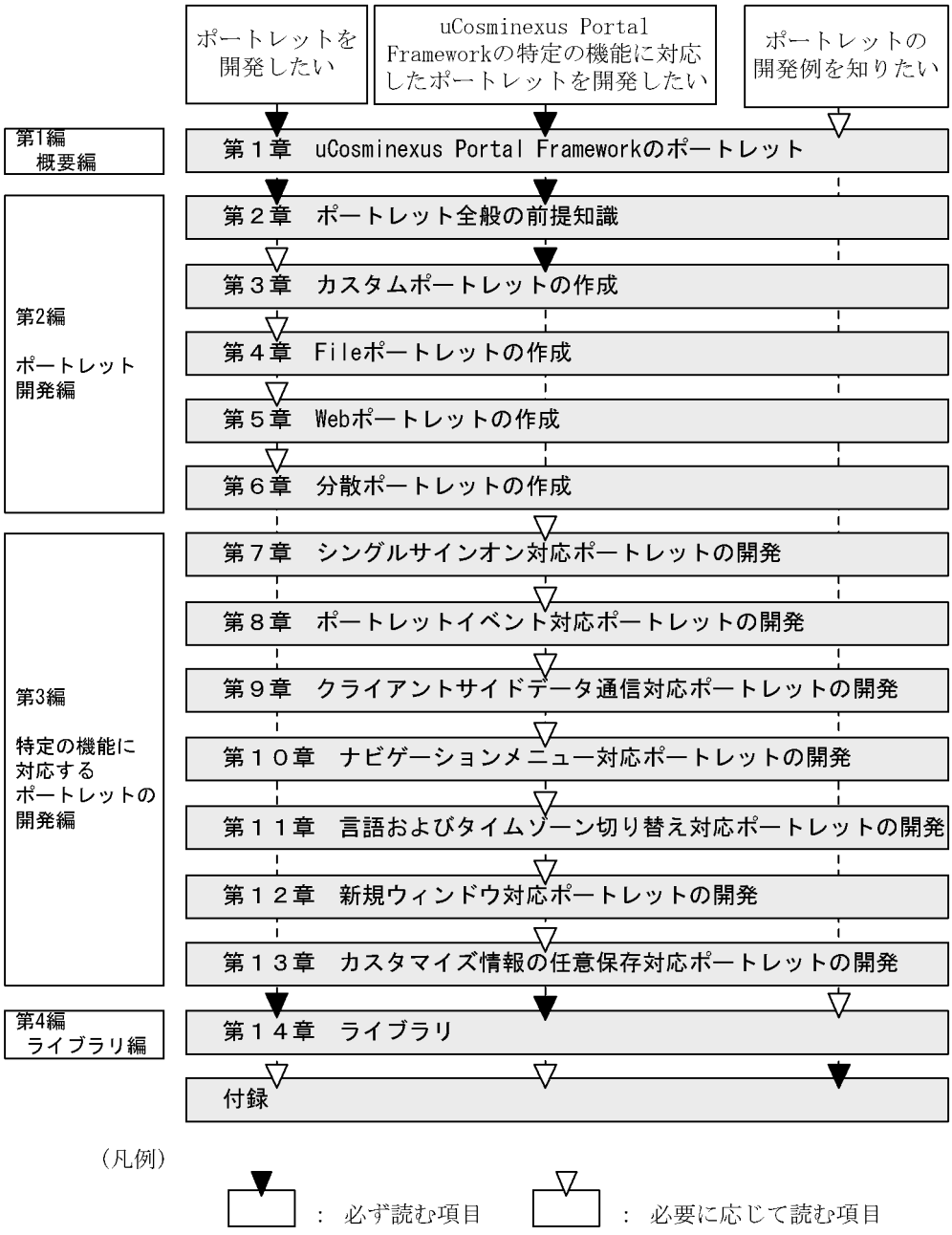
関連マニュアル，このマニュアルで使用している略語の意味などを説明しています。

付録 E　用語解説

このマニュアルで使用する用語について説明しています。

■読書手順

このマニュアルは，利用目的に合わせて次の個所をお読みいただくことをお勧めします。



■操作方法の説明で使用する記号

このマニュアルでは，次に示す記号を使用して操作方法を説明しています。

記号	意味
{ }	ユーザが指定する内容を示します。
[ ]	ウィンドウのボタン，チェックボックスの項目名，画面名，または画面内の項目名を示します。

■構文の説明で使用する記号

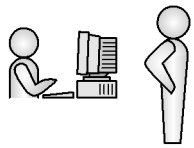
このマニュアルでは，次に示す記号を使用して構文の要素を説明しています。

記号	意味
|	横に並べられた複数の項目に対する項目間の区切りを示し，「または」の意味を表します。 （例）A | B A または B を指定することを示します。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを意味します。項目が横に並べられ，記号|で区切られている場合は，そのうちの一つを選択します。 （例）{A | B | C} A，B，または C のどれかを指定することを示します。
[ ]	この記号で囲まれている項目は省略してもよいことを示します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じくどれか一つを選択します。 （例）［A］ 「何も指定しない」か「A を指定する」ことを示します。

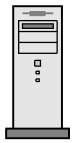
■図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。

●ユーザ

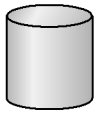


●サーバ

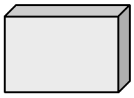


●携帯電話

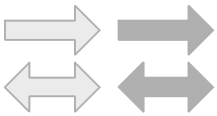
●DB



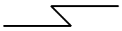
●プログラム



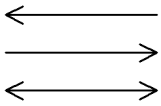
●データの流れ



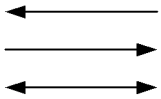
●通信回線



●制御の流れ



●その他の流れ



●ネットワーク

●画面

●ファイル

目次

第 1 編　概要編

第 2 編　ポートレット開発編

第 3 編　特定の機能に対応するポートレットの開発編

1 uCosminexus Portal Framework のポートレット

この章では，uCosminexus Portal Framework で使用できるポートレット，ポートレットの開発の流れ，および uCosminexus Portal Framework の特定の機能に対応するポートレットについて説明します。

1.1　uCosminexus Portal Framework で使用できるポートレット

uCosminexus Portal Framework とは，企業ポータルを構築および運用するフレームワークです。企業の内外にある多種多様なコンテンツ・業務システムの情報を集約する機能や，集約した情報をユーザ一人一人に対して最適な情報になるように組み合わせて提供する機能などを備えています。これらの機能によって，効果的な企業ポータルを構築できます。

uCosminexus Portal Framework では，構築した企業ポータルに設定するポートレットを開発する必要があります。このマニュアルでは，ポートレットの開発方法について説明します。

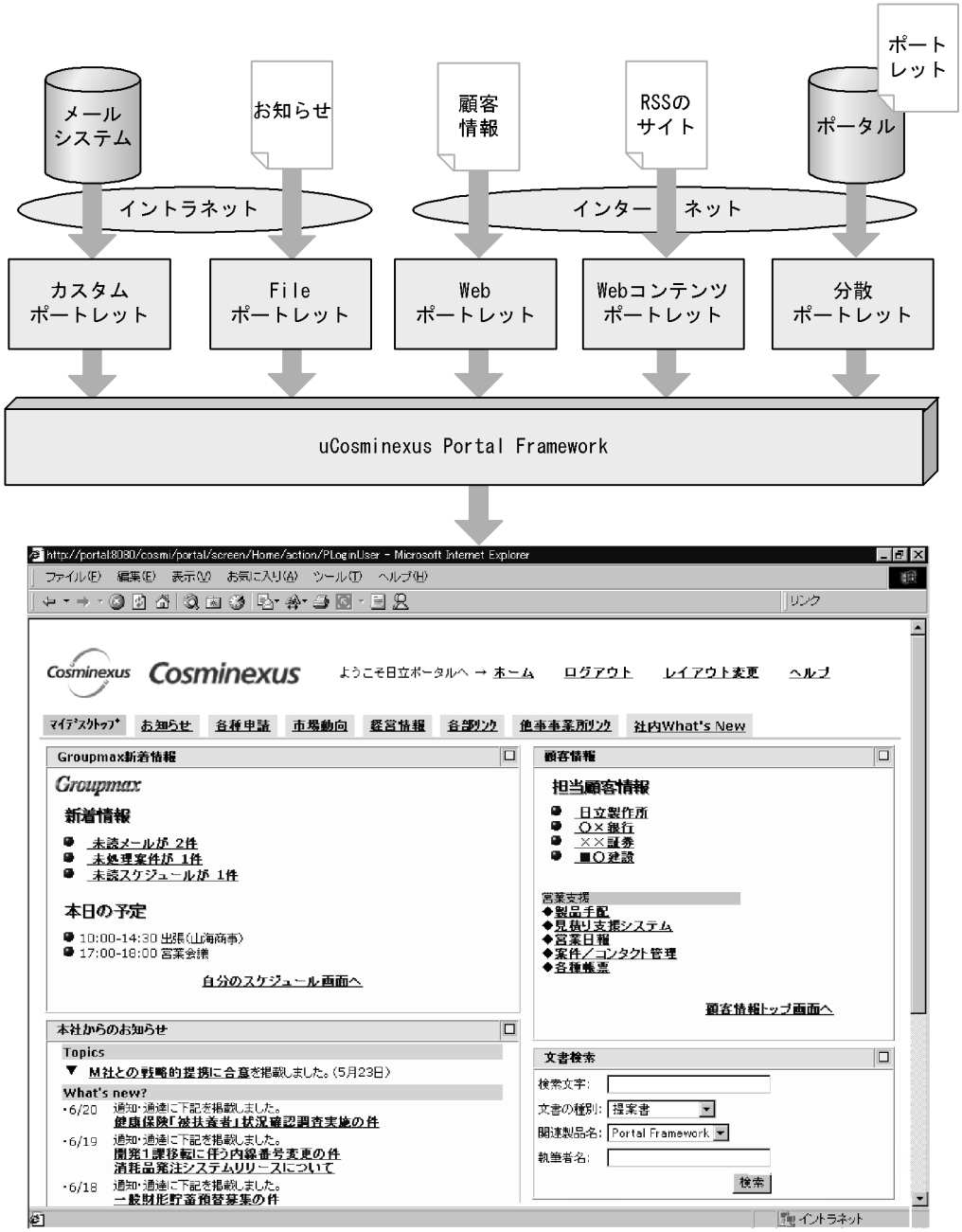
uCosminexus Portal Framework では，次のポートレットを使用できます。

- カスタムポートレット
- File ポートレット※
- Web ポートレット
- 分散ポートレット
- Web コンテンツポートレット

注※　正式名称は Multi File Portlet ですが，このマニュアルでは File ポートレットと表記します。

uCosminexus Portal Framework が各種ポートレットを取り込んで表示する処理を次の図に示します。

図 1-1 ポートレットの取り込み



それぞれのポートレットの特長を示します。

カスタムポートレット

カスタムポートレットは，JSP やサーブレットの技術を用いて独自のポートレット

を作成するポートレットです。動的なページを作成したり，企業独自の業務システムをポータルに統合したり，既存のアプリケーションと連携したりできます。例えば，既存のメールシステムをポータルに統合することで，ユーザはポータルからメールを読めます。また，携帯電話（i モードおよび EZweb）からアクセスできるポートレットも作成できます。

カスタムポートレットには次の 2 種類があります。

- 日立 API ポートレット
  uCosminexus Portal Framework の API を用いて JSP およびサーブレットで作成するポートレットです。
- 標準 API ポートレット
  Java Portlet Specification 1.0 または Struts フレームワークに従って作成するポートレットです。

カスタムポートレットのコンテンツを開発する場合は，ポートレットの開発方法，開発時に使用する API の知識が必要です。ポートレットの開発方法，開発時に使用する API の知識の詳細は，「2. ポートレット全般の前提知識」，「3. カスタムポートレットの作成」および「14. ライブラリ」を参照してください。また，日立 API ポートレットで，uCosminexus Portal Framework の特定の機能に対応する開発をしたい場合は，次の個所を参照してください。

- 「7. シングルサインオン対応ポートレットの開発」
- 「8. ポートレットイベント対応ポートレットの開発」
- 「9. クライアントサイドデータ通信対応ポートレットの開発」
- 「10. ナビゲーションメニュー対応ポートレットの開発」
- 「11. 言語およびタイムゾーン切り替え対応ポートレットの開発」
- 「12. 新規ウィンドウ対応ポートレットの開発」
- 「13. カスタマイズ情報の任意保存対応ポートレットの開発」

File ポートレット

File ポートレットは，HTML，CHTML，または HDML のどれかで記述されたファイルをポートレットとして表示します。例えば，イントラネット上の各部署のお知らせを，一つの File ポートレットに統合できます。携帯電話（i モードおよび EZweb）からアクセスできるポートレットも作成できます。

このポートレットのコンテンツを開発するには，HTML，CHTML，または HDML についての知識が必要です。

このポートレットを開発する場合は，「2. ポートレット全般の前提知識」および「4. File ポートレットの作成」を参照してください。

Web ポートレット

Web ポートレットは，HTML で記述された Web ページをポータルに取り込むポートレットです。ポータルプロジェクト内のローカルコンテンツ，および外部の Web サーバのコンテンツをポータルに統合できます。Web ページをポートレットとしてポータルに統合することで，ベーシック認証，プロキシ認証，およびフォーム認証

に対してシングルサインオンができるようになります。また，コンテンツをフィルタリングしてWebページの一部分をポータルに取り込めます。例えば，顧客情報などのインターネットの情報をポータルに取り込めます。また，携帯電話（ｉ モード，およびEZweb）からアクセスできるポートレットも作成できます。
このポートレットは，コンテンツを開発する必要はありません。Portal Managerでポータルに登録するときに，Webページのコンテンツの URLなどを設定することでポートレットが作成されます。このポートレットを作成するには，Webサイト開発についての知識が必要です。Portal ManagerでWebポートレットをポータルに登録する方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Webポートレットの設定」の説明を参照してください。
このポートレットを作成する場合は，「2.　ポートレット全般の前提知識」，および「5.　Webポートレットの作成」を参照してください。

分散ポートレット

分散ポートレットは，ほかのポータル上に配置されたポートレットを自ポータルに取り込むポートレットです。例えば，他事業所で構築しているポータル上のポートレットを，自ポータルに取り込めます。
このポートレットは，コンテンツを開発する必要はありません。Portal Managerでポータルに登録するときに，取り込む分散ポートレットを配置しているサーバのサーバアドレスなど設定することでポートレットが作成されます。このポートレットを作成するには，ポータルサイトの開発についての知識が必要です。Portal Managerで分散ポートレットをポータルに登録する方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「分散ポートレットの設定」の説明を参照してください。
このポートレットを作成する場合は，「2.　ポートレット全般の前提知識」，および「6.　分散ポートレットの作成」を参照してください。

Web コンテンツポートレット

Webコンテンツポートレットは，外部のWebサーバからコンテンツを取得し，取得したデータの表示方法を加工してポータルに表示するポートレットです。取得できるコンテンツは，HTML，RSS，またはXMLで記述されたページです。
Webコンテンツポートレットは，運用管理ポートレットでコンテンツの取得先や加工の方法などを設定することで，ポートレットが作成されます。登録の作業は必要ありません。
運用管理ポートレットでWebコンテンツポートレットを作成する方法については，マニュアル「uCosminexus Portal Framework　運用管理者ガイド」の「Webコンテンツポートレットを作成する」の説明を参照してください。

1.2 ポートレットの開発の流れ

ここでは，ポートレットの開発の流れについて説明します。ポートレットの開発の流れを次の図に示します。

図 1-2 ポートレットの開発の流れ

手順

1. **ポートレットの目的を検討します。**
   だれが何をするためのポートレットを開発したいのかを整理します。
2. **ポートレットの種類を決定します。**
   1. で整理した目的に応じて，ポートレットの種類を決定します。コンテンツの種類に応じて，カスタムポートレット，File ポートレット，Web ポートレット，または Web コンテンツポートレットのどれかを選択します。また，システムの規模に応じて，分散ポートレットが必要かどうかを検討します。
3. **ポートレットの機能を決定します。**
   カスタムポートレット，および File ポートレットの場合
   各ポートレットの機能，およびターゲットとするデバイス（PC，ｉモード，または EZweb）を決定します。
   カスタムポートレットの日立 API ポートレットのコンテンツを開発する場合，uCosminexus Portal Framework が提供する機能から必要な機能を取捨選択して，

ポートレットの機能を決定します。

- uCosminexus Portal Framework の特定の機能に対応するかどうか
  uCosminexus Portal Framework の特定の機能に対応するポートレットについては，「1.3　uCosminexus Portal Framework の機能に対応するポートレット」を参照してください。
- どのライブラリを利用して機能を実現するか
  uCosminexus Portal Framework で提供しているライブラリについては，「14.　ライブラリ」を参照してください。また，ライブラリを使用する前には，「2.　ポートレット全般の前提知識」を参照してください。

Web ポートレット，分散ポートレット，および Web コンテンツポートレットの場合

Web ポートレットを作成する場合，どの Web ページのコンテンツをどのようにポータルに取り込むかを決定します。
分散ポートレットを作成する場合，どのポータルのどのポートレットを自ポータルに取り込むかを決定します。
Web コンテンツポートレット作成する場合，どの Web ページのコンテンツをどのように加工し，どのテンプレートでポータルに取り込むかを決定します。

4. **ポートレットのコンテンツを開発します。**

   カスタムポートレット，および File ポートレットの場合

   ターゲットとするデバイスに対応する HTML/CHTML/HDML のバージョンでコンテンツを開発する必要があります。uCosminexus Portal Framework が対応している各言語のバージョンについては，「2.1.1　ターゲットとするデバイス」を参照してください。
   カスタムポートレットのコンテンツ作成の詳細は，「3.　カスタムポートレットの作成」を参照してください。
   File ポートレットのコンテンツ作成の詳細は，「4.　File ポートレットの作成」を参照してください。

   Web ポートレット，および分散ポートレットの場合

   コンテンツを作成する必要はありません。次の 5. に進んでください。

   Web コンテンツポートレットの場合

   コンテンツ，ポートレットの作成，およびポータルへの登録をする必要はありません。運用管理ポートレットでパラメタを設定して，7. へ進んでください。

5. **ポートレットを作成します。**

   ターゲットとするデバイスに，ポートレットを表示するための設定を行います。各ポートレットの詳細設定は，このマニュアルの次の個所を参照してください。

   - 「3.　カスタムポートレットの作成」
   - 「4.　File ポートレットの作成」
   - 「5.　Web ポートレットの作成」
   - 「6.　分散ポートレットの作成」

6. **ポートレットをポータルに登録します。**

   ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework

システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

7. **ポートレットをデバッグします。**
   登録したポートレットが正しく動作するかどうかを確認します。ポートレットを確認するためにはポータルサーバを再起動する必要があります。ポータルサーバの起動方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポータルの起動と終了」の説明を参照してください。

1.3 uCosminexus Portal Framework の機能に対応するポートレット

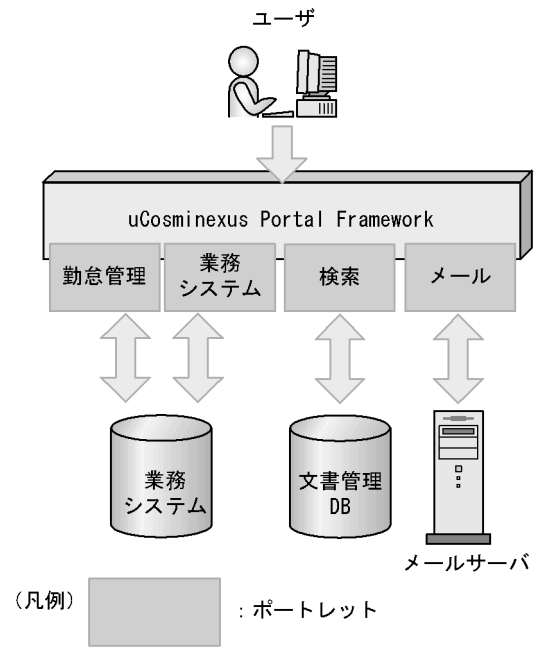
ここでは，uCosminexus Portal Framework で開発できる，特定の機能に対応するポートレットについて説明します。

1.3.1 シングルサインオン対応ポートレット

ユーザが uCosminexus Portal Framework にログインすると，ユーザ ID が異なるシステムをポータル画面から使用できます。この機能を**シングルサインオン**といいます。この機能を使用することで，ユーザは，システムごとにログインしたり，作業ごとにシステムを使い分けたりする必要がありません。

シングルサインオンの概要を次の図に示します。

図 1-3 シングルサインオンの概要



シングルサインオンに対応するポートレットの開発方法については，「7. シングルサインオン対応ポートレットの開発」を参照してください。

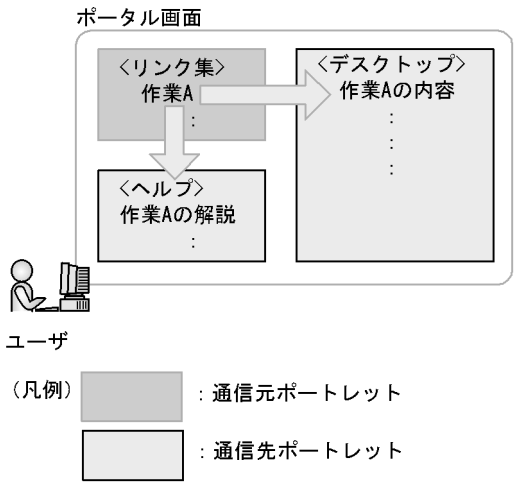
1.3.2 ポートレット間通信対応ポートレット

ポートレットのコンテンツを表示するだけではなく，ポートレット間で情報を通信できます。このため，各ポートレットで行っている業務情報の共有化および分散化を図り，

業務を連携できます。この機能を**ポートレット間通信**といいます。

ポートレット間通信の例を次の図に示します。

図 1-4　ポートレット間通信によってポートレットを連携させた例



ユーザが＜リンク集＞ポートレットから作業 A を選択すると，＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットに通知されます。この通知によって，作業 A の内容が＜デスクトップ＞ポートレットに，作業 A の解説が＜ヘルプ＞ポートレットに表示されます。

ポートレット間通信によってポートレットを連携する場合の，ポートレットの開発方法については，「8.　ポートレットイベント対応ポートレットの開発」を参照してください。

1.3.3　クライアントサイドデータ通信対応ポートレット

データを入力する項目が多くある場合，直接入力では入力ミスが発生しやすくなります。このような入力ミスを減らすため，ポートレットでデータを入力するときに，別のポートレットから該当するデータをドラッグ & ドロップしたり，コピーボタンを利用してコピーしたりして入力できます。この機能を**クライアントサイドデータ通信**といいます。

クライアントサイドデータ通信には，次の機能があります。

● ドラッグ & ドロップ機能

ポートレットにデータを入力するときに，別のポートレットからデータをドラッグ & ドロップして入力する機能です。例えば，メールのポートレットでメールを作成する場合に，電子アドレス帳のポートレットから宛先をドラッグ & ドロップしてメールに貼り付けることができます。

● データフォーム転送機能

ポートレットにデータを入力するときに，データを選択して入力する機能です。この機能には，次の二つの方法があります。

- **補助ウィンドウを使用する方法**
  別のポートレットから取得した入力候補となるデータを補助ウィンドウに表示します。このウィンドウで目的のデータを選択することで，データを入力できます。例えば，メールのポートレットでメールを作成する場合，宛先のボタンをクリックすると，補助ウィンドウが表示されます。ここに，電子アドレス帳のポートレットから取得した宛先の候補が表示されます。目的の宛先を選択して［OK］ボタンをクリックすると，選択した宛先が入力されます。
- **ボタンを使用する方法**
  別のポートレットでデータを選択して，［コピー］ボタンをクリックします。その後，入力項目があるポートレットで［貼り付け］ボタンをクリックすると，選択したデータが入力されます。例えば，電子アドレス帳のポートレットで宛先を選択し，［コピー］ボタンをクリックします。その後，メールのポートレットで［貼り付け］ボタンをクリックすると，選択した宛先が入力されます。

クライアントサイドデータ通信ができるポートレットの開発方法については，「9. クライアントサイドデータ通信対応ポートレットの開発」を参照してください。

1.3.4 ナビゲーションメニュー対応ポートレット

ポートレットをメニューバーに登録すると，ポートレットを簡単に表示できるようになります。このメニューバーを，**ナビゲーションメニュー**といいます。ナビゲーションメニューを使用すると，よく利用するポートレットをすべてポータル画面に表示しておかなくても，必要なときにだけ簡単に表示できるようになります。

ナビゲーションメニューにポートレットを登録すると，そのポートレットのメニューが表示されます。このメニューをクリックすると，ポートレットが最大化表示されます。また，新しいウィンドウを起動して，ポートレットを表示することもできます。リンク集に，よく参照する Web ページへのリンクを登録したり，ポートレットの中でも特によく利用する特定の画面へのショートカットリンクを登録したりもできます。

ナビゲーションメニューにメニューを表示できるポートレット，およびポートレットリンク集に，特定の画面へのショートカットリンクを登録できるポートレットを開発したい場合は，「10. ナビゲーションメニュー対応ポートレットの開発」を参照してください。

なお，ナビゲーションメニューに登録するポートレットは，最大化表示および新しいウィンドウでの表示ができるように開発する必要があります。

ポートレットを最大化表示できるようにするには，最大化表示画面モード時に表示するコンテンツを開発したあと，ポートレットを Portal Manager でポータルに登録するときに最大化画面に対応するよう設定します。設定方法については，「3.13.2 デプロイ定義ファイル（hportlet.xml）の作成」またはマニュアル「uCosminexus Portal Framework システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

新しいウィンドウで表示できるポートレットの開発方法については，「12. 新規ウィン

ドウ対応ポートレットの開発」を参照してください。

1.3.5 言語およびタイムゾーン切り替え対応ポートレット

海外などでポータルを使用したい場合など，必要に応じて個人ごとに使用する言語およびタイムゾーンを切り替えられます。

- **使用する言語の切り替え**
  ポータル画面に表示される文字列を，日本語または英語のどちらにするか選択できます。
- **タイムゾーンの切り替え**
  J2SE1.4 で定めるカスタムタイムゾーン ID を利用して，タイムゾーンを切り替えられます。

言語およびタイムゾーンの切り替えに対応するポートレットの開発方法については，「11. 言語およびタイムゾーン切り替え対応ポートレットの開発」を参照してください。

1.3.6 新規ウィンドウ対応ポートレット

uCosminexus Portal Framework では，ナビゲーションメニューからポートレットを表示する場合に，新規ウィンドウ画面で表示することもできます。新規ウィンドウ画面で表示できるポートレットの開発方法については，「12. 新規ウィンドウ対応ポートレットの開発」を参照してください。

1.3.7 カスタマイズ情報の任意保存対応ポートレット

カスタマイズ情報は通常ログアウト時に保存されますが，カスタマイズ情報任意保存機能を使用すると，ポートレット上で任意のタイミングで保存できるようになります。

カスタマイズ情報任意保存機能を使用できるポートレットの開発方法については，「13. カスタマイズ情報の任意保存対応ポートレットの開発」を参照してください。

2 ポートレット全般の前提知識

この章では，すべてのポートレットを作成する際に，前提となる知識について説明します。

2.1 HTML 記述上の注意

uCosminexus Portal Framework では，HTML のテーブル要素を使って複数のポートレットを一画面に配置します。そして，クライアントのデバイス（PC，ｉ モード，または EZweb）では，一つの HTML コンテンツとなります。したがって，ポートレット開発時には，次の点に注意してください。

2.1.1 ターゲットとするデバイス

ポートレットを作成する前に，ターゲットとするデバイス（PC，ｉ モード，または EZweb）を決定します。ポートレットのコンテンツは，ターゲットとするデバイスに対応する HTML/CHTML/HDML のバージョンで記述します。ターゲットとするすべてのデバイスのバージョンでテストすることを推奨します。

uCosminexus Portal Framework は次のバージョンに対応しています。

- HTML4.01
- i モード対応 HTML2.0
- Version 3.3 for HDML

また，デバイスによっては，HTML のテーブル要素の階層が深い場合，画面の表示速度が遅いことがあります。そのため，ポートレット，特にサマリ画面を表示させるときには，テーブル要素の多用を避けることを推奨します。

2.1.2 ドキュメントベース

クライアントに表示されるポータル画面では，HTML のドキュメントベースは次のとおりになります。

`http://` {ポータルサーバ名} / {ポータルプロジェクト名}

ポートレットのコンテンツで，通常のコンテンツ作成時と同様に相対パス形式でファイルを指定すると，上記ドキュメントベースが基点となるため，ファイルを正しく取得できません。そのため，ポートレット作成時には，ポートレットユティリティタグライブラリを使用します。このタグライブラリは，ポータル画面でファイルを正しく表示できるように，ポートレットに含まれる相対パス形式の URL を変換します。通常の HTML タグの代わりにタグライブラリを使用すると，相対パス形式で URL を指定できます。ただし，ポートレットの URL 定義およびハイパーリンクとして指定する JSP ファイルのパス名の長さは，Web コンテナおよび OS によって制限があります。詳細は，マニュアル「Cosminexus アプリケーションサーバ V8　システム構築・運用ガイド」，またはマニュアル「Cosminexus V9　アプリケーションサーバ　システム構築・運用ガイド」の「J2EE サーバの作業ディレクトリ」の説明を参照してください。

インラインオブジェクトの URL は，相対パス形式，または絶対 URL 形式のどちらかで

指定します。相対パス形式で指定する場合，ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用します。

ポートレットユティリティタグライブラリの使用例については，「付録 A.2　ハイパーリンクの作成」を参照してください。また，URL の変換については，「3.8.2　URL 変換」を参照してください。

2.1.3　ネームスペース

クライアントのデバイス（PC，ｉ モード，または EZweb）では一つの HTML コンテンツとなるため，HTML ファイル内の属性には，ポートレットごとに異なる名称を指定する必要があります。

クライアントサイドスクリプトを使用する場合は，関数名およびグローバル変数名にも同様に，ポートレットごとに異なる名称を指定する必要があります。また，HTML でコンテンツを作成する場合，accesskey 属性（要素のアクセスキー）の使用を推奨しません。

異なる名称を指定する必要があるネームスペースとその命名規則を次の表に示します。

表 2-1　ネームスペースの命名規則

ネームスペース	命名規則
name 属性値	hptl_user_{ポートレット名}_{name 属性値}
id 属性値	hptl_user_{ポートレット名}_{id 属性値}
class 属性値	hptl_user_{ポートレット名}_{class 属性値}
関数名	hptl_user_{ポートレット名}_{関数名}
グローバル変数名	hptl_user_{ポートレット名}_{グローバル変数名}

2.1.4　エンコーディング

ユーザの環境による文字化けを防ぐために，次に推奨するエンコーディングでコンテンツを作成してください。

- マルチデバイス環境（PC，ｉ モード，または EZweb）の場合
  Shift_JIS を推奨します。
- PC 環境でだけ使用する場合
  UTF-8 を推奨します。

Shift_JIS では表現できないコンテンツ（JISX0201 ラテン，JISX0201 カナおよび JISX0208 の定義済み文字以外を使用する場合）を作成する場合には，プライベートエリアを含まない Unicode の範囲を使用して，UTF-8 での記述を推奨します。

日立 API ポートレットのエンコーディングは，各ポートレットで指定します。日立 API

ポートレットで日本語を使用する場合には，page ディレクティブの ContentType 属性および charset 属性を正しく設定してください。

- contentType
  contentType には，MIME タイプを指定します。contentType に指定する MIME タイプを次の表に示します。

表 2-2　contentType に指定する MIME タイプ

記述言語	MIME タイプ
HTML	text/html
CHTML	text/html
HDML	text/x-hdml

- charset
  charset には，文字コードを指定します。ユーザの環境による文字化けを防ぐために，charset は JSP ファイルのエンコーディングに合わせて指定してください。

2.1.5　使用できない HTML 要素，タグ

ポートレットは，ポートレットとして表示されるコンテンツ本体部分だけを表示します。HTML，HEAD 要素などの HTML として必要な要素，およびタグは，uCosminexus Portal Framework が出力します。そのため，次の宣言，タグ，および要素は，日立 API ポートレット，および File ポートレットでは使用しないでください。

- DOCTYPE 宣言
- HTML 開始，終了タグ
- HEAD 要素
- BODY 開始，終了タグ
- FRAMESET 要素
- STYLE 要素

標準 API ポートレットで使用できないタグ，および要素については，「3.16.3　標準 API ポートレットで使用できないタグ」を参照してください。

2.1.6　使用できない CHTML 要素，タグ

ポートレットは，ポートレットとして表示されるコンテンツ本体部分だけ表示します。HTML，BODY タグ，HEAD 要素といった CHTML として必要な要素，およびタグは，uCosminexus Portal Framework が出力します。そのため，次に示すタグ，および要素はポートレットで使用しないでください。

- HTML 開始，終了タグ

- HEAD 要素
- BODY 開始，終了タグ

2.1.7 使用できない HDML 要素，タグ

ポートレットは，ポートレットとして表示されるコンテンツ本体部分だけを表示します。HDML として必要な要素，および HDML タグは，uCosminexus Portal Framework が出力します。そのため，次に示すタグはポートレットで使用しないでください。

- HDML 開始，終了タグ

2.1.8 パス名

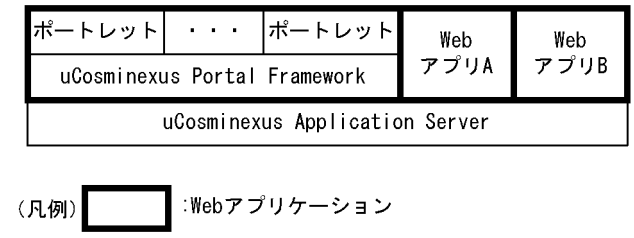
パス名には，英数字，「-」，および「_」を使用できます。

2.2 ポートレットの動作

uCosminexus Portal Framework は，uCosminexus Application Server 上で動く Web アプリケーションであり，JSP/ サーブレットで構築されています。uCosminexus Portal Framework は，複数のポートレットで構成されます。ポートレットは，Servlet API のリクエストディスパッチで呼び出され，uCosminexus Portal Framework の延長として動作します。標準 API ポートレットは，uCosminexus Portal Framework の一部として画面に表示されますが，uCosminexus Portal Framework とは別のアプリケーションとして構築されます。

uCosminexus Portal Framework とポートレットの関係を次の図に示します。

図 2-1 uCosminexus Portal Framework とポートレットの関係



uCosminexus Application Server 上では，uCosminexus Portal Framework 以外にも，JSP やサーブレットで構築された Web アプリケーションが動作します。そのため，ポートレットは，uCosminexus Portal Framework およびほかのポートレットと，Web アプリケーションのコンテキストを共有します。

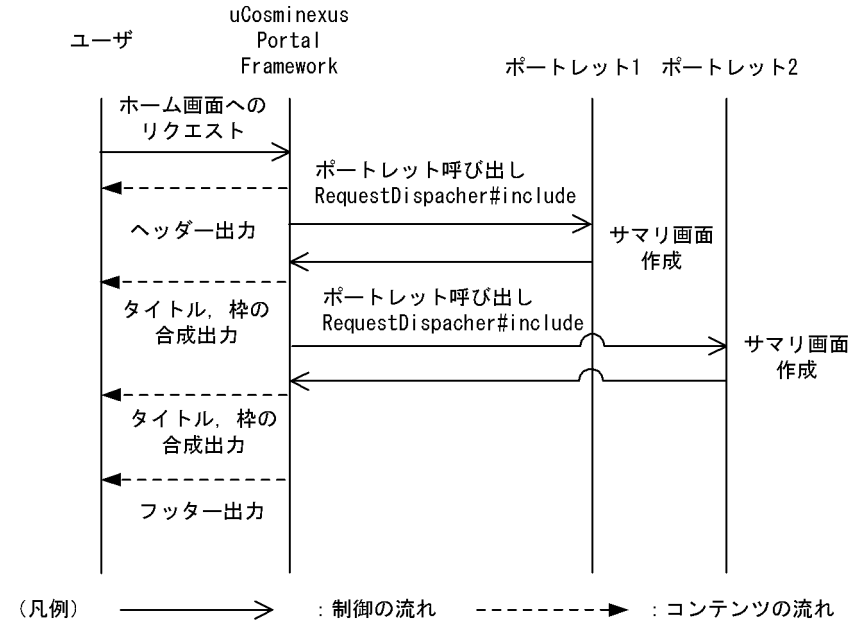
2.2.1 ポートレットの呼び出し

ポートレットは，Servlet API のリクエストディスパッチで呼び出されます。また，ポートレットは，uCosminexus Portal Framework 上で動作するほかのポートレットと，次のオブジェクトを共有します。

- ServletContext
- ServletRequest
- ServletResponse
- ServletOutputStream
- HttpSession

ユーザが初めてポータルのウェルカム画面にアクセスしたときのポートレット呼び出しシーケンスを次の図に示します。

図 2-2　ポートレット呼び出しシーケンス



日立 API ポートレットの開発では，通常の Web アプリケーションの開発に加えて，次の点に注意してください。

(1) セッション管理

セッション管理は uCosminexus Portal Framework が実施します。ポートレットでは Servlet API のセッション管理機能をそのまま利用できます。ただし，HttpSession は uCosminexus Portal Framework およびポートレットで共有するので，ポートレット内でセッションが必要ない場合でも，HttpSession#invalidate でセッションを無効にしないでください。

(2) HTTP ヘッダ操作

ポートレット内での HTTP レスポンスヘッダおよびレスポンスコードの操作はできません。Servlet API での Cookie 設定，HTTP リダイレクトなどの操作は無効となります。

また，HTTP レスポンスヘッダが登録できる場合は，Cookie の情報はまとめて登録されます。ただし，次のような場合に，同一名称の Cookie を登録しようとしたときは，エラーメッセージが出力され，ポートレットの Cookie 設定が無効となります。

- 別のポートレットで，同一名称の Cookie の登録しようとした場合
- 同一ポートレットで，再度同一名称の Cookie の情報を登録しようとした場合

(3) アトリビュート操作

HttpSession および ServletContext はすべてのポートレットで共有するため，アトリビュート名には，PortletUtils#getNamespace で取得したプレフィックスを付けます。予約語のため，プレフィックスとして次の用語は使用できません。

- jp.co.hitachi.soft.portal
- turbine
- jetspeed

アトリビュート操作の例については，「付録 A.3　アトリビュート操作」を参照してください。

(4) リクエストディスパッチ

リクエストディスパッチには，RequestDispatcher#include および RequestDispatcher#forward を使用します。ただし，forward を使用する場合，RequestDispatcher は ServletRequest から取得してください。

(5) 複数ログイン対応

uCosminexus Portal Framework では同一ユーザの複数ログインが制限されていません。同一ユーザが複数ログインした場合，それぞれに対して HttpSession が割り当てられます。ユーザセッション管理を HttpSession で実現する場合には，必要に応じて一貫性制御をする必要があります。

(6) ポートレットの呼び出し順序

ポートレットの呼び出し順序に依存したポートレット開発はできません。

3 カスタムポートレットの作成

この章では，カスタムポートレットを作成する際の手順，および注意事項について説明します。

3.1　カスタムポートレットとは

カスタムポートレットは，JSP やサーブレットの技術を用いて独自のポートレットを作成するポートレットです。動的なページを作成したり，企業独自の業務システムをポータルに統合したり，既存のアプリケーションと連携したりできます。例えば，既存のメールシステムをポータルに統合することで，ユーザはポータルからメールを読めます。また，携帯電話（ｉ モードおよび EZweb）からアクセスできるポートレットも作成できます。

カスタムポートレットには次の 2 種類があります。

日立 API ポートレット

uCosminexus Portal Framework の API を用いて作成するポートレットです。作成方法の詳細は，「3.2　日立 API ポートレットの作成手順」を参照してください。

標準 API ポートレット

Java Portlet Specification 1.0 の API を用いて作成するポートレットです。作成方法の詳細は，「3.15　標準 API ポートレットの作成手順」を参照してください。

3.2 日立 API ポートレットの作成手順

この節では，日立 API ポートレットを作成する手順について説明します。

手順

1. **日立 API ポートレットのコンテンツを開発します。**
   コンテンツを開発する際には，ターゲットとするデバイス（PC，ｉモード，またはEZweb）に対応する HTML/CHTML/HDML のバージョンで記述する必要があります。uCosminexus Portal Framework が対応している各言語のバージョンについては，「2.1.1　ターゲットとするデバイス」を参照してください。
   また，コンテンツを開発する際には，次の前提知識，および注意事項を確認してください。

   前提知識
   - 「3.6　uCosminexus Portal Framework のアクションモジュール」
   - 「3.7　日立 API ポートレットの画面モード」
   - 「3.8　日立 API ポートレットの動作」
   - 「3.9　ポートレットの無応答監視」
   - 「3.10　サーブレットを用いたポートレットの開発」
   - 「3.11　ポートレットの開発モデル」
   - 「3.14　ダイレクト呼び出し対応ポートレットの開発」

   注意事項
   - 「2.　ポートレット全般の前提知識」
   - 「3.3　HTML 記述上の注意（日立 API ポートレット）」
   - 「3.4　タグライブラリ使用時の注意」
   - 「3.5　パーソナライズ情報使用時の注意」

   なお，日立 API ポートレットからは J2EE サービスを利用できません。
   uCosminexus Portal Framework の特定の機能に対応するコンテンツを作成する場合は，このマニュアルの次の個所に記載しているコーディング例を参考にして開発してください。
   - 「7.　シングルサインオン対応ポートレットの開発」
   - 「8.　ポートレットイベント対応ポートレットの開発」
   - 「9.　クライアントサイドデータ通信対応ポートレットの開発」
   - 「10.　ナビゲーションメニュー対応ポートレットの開発」
   - 「11.　言語およびタイムゾーン切り替え対応ポートレットの開発」
   - 「12.　新規ウィンドウ対応ポートレットの開発」
   - 「13.　カスタマイズ情報の任意保存対応ポートレットの開発」

   なお，uCosminexus Portal Framework の特定の機能に対応するコンテンツを作成するときに uCosminexus Portal Framework が提供する API を使用する場合は，「14.　ライブラリ」を参照してください。

また，その他のポートレットについては，「付録 A　ポートレットのサンプル」のコーディング例を参照してください。

2. **日立 API ポートレットを作成します。**
   ターゲットとするデバイスに，ポートレットを表示するための設定を行います。この設定の詳細は，「3.12　デバイスに対応するための設定（日立 API ポートレット）」を参照してください。

3. **PAR ファイルを作成します。**

   クラスファイルをパッケージする日立 API ポートレット

   ポートレットアーカイブ（ポートレットの PAR ファイル）を作成する必要があります。PAR ファイルの作成方法については，「3.13　日立 API ポートレットのアーカイブの作成」を参照してください。

   その他の日立 API ポートレット

   PAR ファイルを作成する必要はありません。次の 4. に進んでください。

4. **日立 API ポートレットをポータルに登録します。**
   ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

3.3 HTML 記述上の注意（日立 API ポートレット）

3.3.1 予約語

uCosminexus Portal Framework では「hptl」で始まる文字列を予約語としています。このため，ポートレット開発時に「hptl」で始まる任意の文字列を指定しないでください。

「hptl」で始まる任意の文字列を使用できないファイルや関数名などについて次に示します。

- クライアントサイドデータ通信機能のタグライブラリ API
- クライアントサイドデータ通信機能の JavaScript 関数
- クライアントサイドデータ通信機能の userData 格納キー名称
- クライアントサイドデータ通信機能のフレーム ID
- ストリングリソース
- セッションにバインドするオブジェクトを指定するときの名称
- request オブジェクトのアトリビュート名

ポータルを経由してポートレットの画面を呼び出す場合，ポートレットではクエリストリング（"?query"）を自由に設計できます。ただし，次の文字列は予約語のため使用できません。

クエリストリングの予約語

action，screen，portlet，pane，mode，url，layout_id，tab_id，jsessionid，hptl_referer，および hptl で始まる文字列

ポータルを経由してのポートレットの画面呼び出しについては，「2.1.2　ドキュメントベース」を参照してください。

3.3.2 スクリプトを使用するには

Web ブラウザによって対応しているスクリプト言語が異なるため，ターゲットとする Web ブラウザを決定してから使用するスクリプト言語を決定します。対応する Web ブラウザが多いため，uCosminexus Portal Framework では，スクリプト言語として JavaScript を推奨します。

ポートレットでスクリプトを使用する場合，通常の記述方法とは次の点が異なります。

- ポートレットの URL を作成する場合，適切なポートレットユティリティタグライブラリまたはポートレットユティリティクラスライブラリを使用します。ポートレットユティリティタグライブラリの詳細は，「14.2　ポートレットユティリティタグライブ

ラリ」を，ポートレットユティリティクラスライブラリの詳細は，「14.4　ポートレットユティリティクラスライブラリ」を参照してください。

- JavaScript および JScript では，ローカル変数宣言時に "var" 宣言子を使用します。
- イベント処理をする場合，JavaScript で記述します。body タグに記述するイベント（onload や onunload など）は使用できません。フレーム内に表示するコンテンツの場合は，body タグに記述するイベントも使用できます。フレーム内にコンテンツを表示する方法については，「3.3.3　フレームを使用するには」を参照してください。
- スクリプトのオブジェクト名，関数名，およびグローバル変数名は異なる名称を指定します。
  関数名およびグローバル変数名には，"hptl_user_ ポートレット名 " をプレフィックスとして付けてください。ただし，Web ブラウザによっては名称の長さに制限があります。ターゲットとする Web ブラウザで正しく表示されない場合は，単語を省略するなどして，名称を短くしてください。

  例

  関数：function hptl_user_kinkyu_display(){...}
  変数：var hptl_user_kinkyu_cont=0;

3.3.3　フレームを使用するには

日立 API ポートレット内でフレームを使用するには，iframe 要素内でフレームを構成します。FRAMESET 要素は，iframe 内だけで使用できます。フレームを使用したポートレットのサンプルは，「付録 A.11　フレームを使用したポートレット」を参照してください。

フレーム内のコンテンツでは次の機能は使用できません。

- View の統合
- マルチデバイス対応のポートレット

フレーム内のコンテンツで次の API を使用するには，ポートレットユティリティタグライブラリを使用してフレームを生成する，またはポートレットユティリティクラスライブラリを使用して生成した URL を用いてフレームを作成してください。

- ポートレットユティリティタグライブラリ（14.2）
- ポートレットユティリティクラスライブラリ（14.4）
- ポートレット情報取得 Bean（14.5）
- ユーザ情報取得 Bean（14.6）
- UserAgent 情報取得 Bean（14.8）

3.4 タグライブラリ使用時の注意

タグライブラリには，ポートレットユティリティタグライブラリ，クライアントサイドデータ通信タグライブラリ，およびナビゲーションメニューのタグライブラリがあります。ポートレットユティリティタグライブラリは，URL 変換をするタグライブラリとそれ以外のタグライブラリに分かれています。ポートレットユティリティタグライブラリの詳細は，「14.2　ポートレットユティリティタグライブラリ」を，クライアントサイドデータ通信タグライブラリの詳細は，「14.11　クライアントサイドデータ通信タグライブラリ」を，ナビゲーションメニューのタグライブラリの詳細は，「14.15　ナビゲーションメニューのタグライブラリ」を参照してください。タグライブラリ全般の注意および URL 変換タグライブラリ使用時の注意を次に示します。

3.4.1 タグライブラリ全般の注意

- タグライブラリはすべて開始タグおよび終了タグを持ちます。終了タグを省略できません。
- 開始，終了タグおよび属性名の指定をマニュアルの記載どおりに統一します。

3.4.2 URL 変換タグライブラリ使用時の注意

URL 変換タグライブラリは，URL を指定する属性を持つ HTML/CHTML/HDML の要素をラッピングしたものです。必要に応じて，指定された相対パス形式の URL を URL 変換し，HTML/CHTML/HDML 要素として出力します。HTML/CHTML/HDML で必須属性として定義されている属性は URL 変換タグライブラリでも必須となります。

URL 変換タグライブラリは，HTML/CHTML/HDML の次のバージョンに従います。

- HTML4.01
- i モード対応 HTML2.0
- Version 3.3 for HDML

URL 変換タグライブラリでは，HttpServletResponse#encodeURL が呼び出されるので，ポートレット開発時に明示的に encodeURL を呼び出す必要はありません。また，例外発生時には，javax.servlet.jsp.JspException が発生します。

URL 変換タグライブラリは次の点で通常の使用方法と異なります。

- 開始，終了タグおよび属性名は，要素ごとに大文字または小文字に統一する必要があります。
- HTML/CHTML/HDML で属性値だけを指定できる属性は，タグライブラリでは属性名 = 属性値として記述する必要があります。
- HTML/CHTML/HDML で空要素となる要素には，内容を記述できません。空要素として記述します。

- 次の属性名は，HTML の属性名とタグライブラリの属性名が異なります。正しく指定してください。HTML の属性名とタグライブラリの属性を次の表に示します。

表 3-1 HTML の属性名とタグライブラリの属性

HTML の属性名	タグライブラリの属性名
class	hclass
id	hid
accept-charset	accept_charset

- 次の属性はタグライブラリでは使用できません。次の属性を使用するには，HTML タグを使用して記述し，PortletURI クラスで URL 変換してください。タグライブラリで使用できない属性を次の表に示します。

表 3-2 タグライブラリで使用できない属性

要素名	属性名
area	nohref
img	ismap
input	ismap，checked，disable，readonly
object	declare
script	defer

- URL を指定する属性および on で始まるイベント属性は，JSP スクリプトレット式で評価されます。JSP スクリプトレット式で評価される属性については，「14.2 ポートレットユティリティタグライブラリ」を参照してください。

3.5 パーソナライズ情報使用時の注意

PortalUserInfoBean の setCustomizeInfo メソッドを使用すると，パーソナライズ情報がシリアライズされてポータルのリポジトリ（ディレクトリサーバ，DB）に格納されます。格納されるパーソナライズ情報のデータサイズについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「DB の容量計算」の説明を参照してください。

3.6 uCosminexus Portal Framework のアクションモジュール

アクションモジュールには，次の二つがあります。

- デフォルトアクションモジュール
- アクションモジュール

3.6.1 デフォルトアクションモジュール

デフォルトアクションモジュールは，uCosminexus Portal Framework に標準で添付されているアクションモジュールです。イベント発生時の標準の処理を記述したモジュールで，すべてのポートレットの標準となるイベント処理が記述されています。

各ポートレットによってイベント処理をする場合は，次のデフォルトアクションモジュール（クラス）を継承し，サポートしているイベント（メソッド）を実装します。

デフォルトアクションモジュール（クラス）

```
jp.co.hitachi.soft.portal.portlet.api.DefaultActionModule
```

デフォルトアクションモジュールのイベントの詳細は，「14.9 ポートレットイベント API」を参照してください。

3.6.2 アクションモジュール

アクションモジュールは，各ポートレットに対してイベントが発生した場合，イベントの種別に応じて記述するモジュールです。

アクションモジュールの詳細は，「14.9 ポートレットイベント API」を参照してください。

3.7 日立 API ポートレットの画面モード

ここでは，uCosminexus Portal Framework で使用できる画面モードについて説明します。

3.7.1 日立 API ポートレットの画面モードの種類

uCosminexus Portal Framework には，次の画面モードがあります。

- DEFAULT モード
  ポータル画面に通常表示される状態です。このモードの統合画面を標準画面と呼びます。
- MAXIMIZE モード
  ポートレットを最大化表示した状態です。このモードの統合画面を最大化画面と呼びます。
- MINIMIZE モード
  ポートレットを最小化表示した状態です。このモードの統合画面を最小化画面と呼びます。
- EDIT モード
  ポートレットの状態を編集するためのモードです。このモードの統合画面を編集画面と呼びます。
- NEWWINDOW モード
  ポートレットを新規のウィンドウに表示した状態です。このモードの統合画面を新規ウィンドウ画面と呼びます。
- IFRAME モード
  Web コンテンツをインラインフレーム内に表示した状態です。このモードの統合画面を JSP/ サーブレット画面と呼びます。

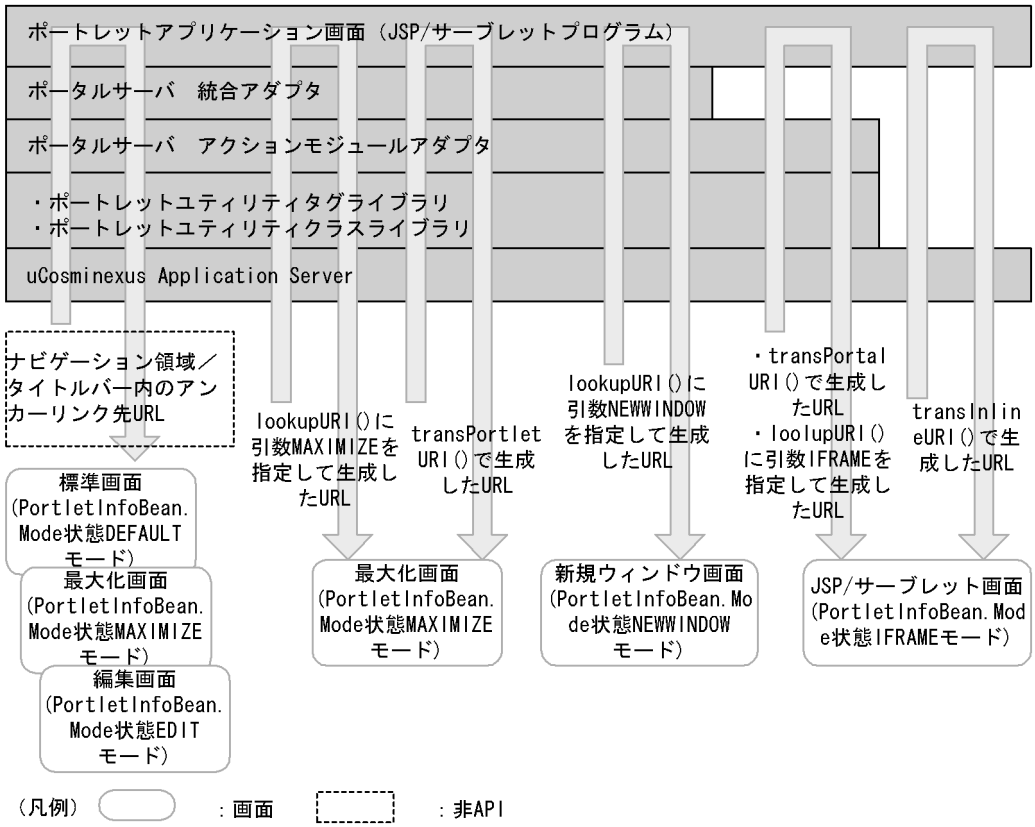
これらの画面は，ポートレットユティリティクラスライブラリの PortletInfoBean.Mode クラスで画面モードとして管理されています。PortletInfoBean.Mode クラスの詳細は，「14.5　ポートレット情報取得 Bean」を参照してください。

これらの画面の遷移図については，「3.8.3　ポートレットユティリティタグライブラリ使用時の画面遷移」，および「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」を参照してください。

(1) 画面の位置づけ

各画面モードの画面の位置づけを次に示します。

図 3-1 uCosminexus Portal Framework の画面の位置づけ



ナビゲーション領域／タイトルバー内のアンカーリンク先 URL

標準画面でナビゲーションメニューの一つのメニュー（アンカー）をクリックすると，そのリンク先 URL を取得し最大化画面を表示します。

標準画面でポートレットのタイトルバー内の編集ボタンをクリックすると，そのリンク先 URL を取得し編集画面を表示します。

lookupURI() に引数 MAXIMIZE を指定して生成した URL

標準画面で最大化画面を呼び出す操作が行われると，引数に MAXIMIZE を指定した lookupURI() メソッドが，呼び出された最大化画面の URL を生成します。その URL を基に，最大化画面を呼び出し表示します。

transPortletURI() で生成した URL

標準画面で最大化画面を呼び出す操作が行われると，transPortletURI() メソッドが，相対パス形式の URL を基に絶対 URL 形式，または絶対パス形式の URL を生成します。その URL を基に，最大化画面を呼び出し表示します。

lookupURI() に引数 NEWWINDOW を指定して生成した URL

標準画面などで新規ウィンドウ画面を呼び出す操作が行われると，引数に

NEWWINDOW を指定した lookupURI() メソッドが，呼び出された新規ウィンドウ画面の URL を生成します。その URL を基に，新規ウィンドウ画面を呼び出し表示します。

transPortalURI() で生成した URL，または lookupURI() に引数 IFRAME を指定して生成した URL

標準画面などでインラインフレームを呼び出す操作が行われると，transPortalURI() メソッド，または引数に IFRAME を指定した lookupURI() メソッドが，呼び出されたインラインフレームの URL を生成します。その URL を基に，インラインフレームを呼び出し表示します。
ここで表示されるインラインフレームは，URL を取得する際に統合アダプタを経由しないため，ポータルの統合画面内で遷移できません。

transInlineURI() で生成した URL

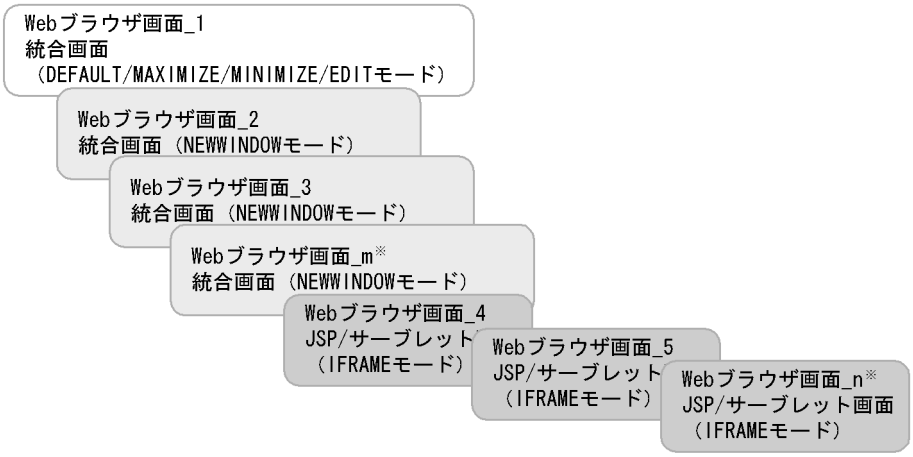
標準画面などで外部 Web サーバ上のコンテンツをインラインフレームに表示する操作が行われると，transInlineURI() メソッドが，表示するコンテンツの URL を外部 Web サーバから直接取得します。その URL を基に，外部 Web サーバ上のコンテンツをインラインフレームに表示します。

lookupURI メソッド，transPortletURI メソッド，transPortalURI メソッド，および transInlineURI メソッドの詳細は，「14.4　ポートレットユティリティクラスライブラリ」の「jp.co.hitachi.soft.portal.portlet.PortletURI」を参照してください。

(2) 複数の Web ブラウザ画面の表示

新規ウィンドウ画面を利用すると，一度に複数の Web ブラウザ画面を表示できます。一度に表示できる画面について，次の図に示します。

図 3-2 新規ウィンドウ機能の概要



1個の統合画面（DEFAULT/MAXIMIZE/MINIMIZE/EDITモード），
m個※の統合画面（NEWWINDOWモード）および
n個※のJSP/サーブレット画面（IFRAMEモード）を表示できます。

注※ m,およびnは0より大きい数字。

新規ウィンドウ画面を利用すると，次の画面を一度に表示できます。

- 1 個の統合画面（DEFAULT/MAXIMIZE/MINIMIZE/EDIT モード）
  図中の Web ブラウザ画面 1
- m 個の新規ウィンドウの統合画面（NEWWINDOW モード）
  図中の Web ブラウザ画面 2 ～ Web ブラウザ画面 m（m は 0 より大きい数字）
- n 個の JSP/ サーブレット画面（IFRAME モード）
  図中の Web ブラウザ画面 4 ～ Web ブラウザ画面 n（n は 0 より大きい数字）

3.7.2 カスタマイズできる項目

システム管理者またはポートレット開発者は，各画面モードをカスタマイズできます。
カスタマイズできる項目を次の表に示します。

表 3-3 システム管理者またはポートレット開発者がカスタマイズできる項目

分類	カスタマイズ項目	DEFAULT モード	MAXIMIZE モード	NEWWINDOW モード	IFRAME モード
http レスポンス	charset	△※1	△※1	△※1	―
head タグ	title タグ	×	×	○※2	―
	link タグ（default.css のパス）	×	×	×	―
	base タグ	―	―	―	―
body タグ	ナビゲーション	△※3	△※3	―	―
	タブ	△※3	△※3	―	―
	ポートレットのタイトル	○※4	○※4	○※5	―

（凡例）
○：ポートレットごとに設定できます。
×：カスタマイズできません。
△：ポータルプロジェクト全体で設定できます。
―：自由に開発できます（ポータルサーバの管理外）。

注※1　システム管理者の設定によって，Shift_JIS，EUC，JIS または UTF-8 の切り替えができます。

注※2　head タグの title タグは，ポータルのウィンドウタイトルと同じ名称になります。システム管理者は，デプロイ時にデプロイ定義ファイル（hportlet.xml）の <windowtitle> タグの設定を変更することによって，この名称を任意の文字列に変更できます。デプロイ定義ファイルを設定する方法，および head タグの title タグを日本語および英語で設定する方法については，「3.13.2(2) デプロイ定義ファイルで使用するタグの一覧」を参照してください。

注※3　ポートレットテンプレートを変更できます。

注※4　システム管理者の設定によって，日本語または英語のポートレットタイトルを設定できます。日本語または英語のポートレットタイトルの設定方法は，「3.13.2(2) デプロイ定義ファイルで使用するタグの一覧」を参照してください。

注※5　システム管理者は，各種設定値によって値を変更でき，ポートレット開発者は，ポータルのライブラリを使用してプログラム上から値を設定できます（デフォルトは，日本語または英語のポートレットタイトルです）。

3.7.3 アクションモジュールの呼び出し条件

アクションモジュールは，ポートレットを画面に表示する際に呼び出されます。アクションモジュールの呼び出し条件は，ポートレットの表示画面ごとに異なります。アクションモジュールの呼び出し条件を次に示します。

表 3-4 アクションモジュールの呼び出し条件

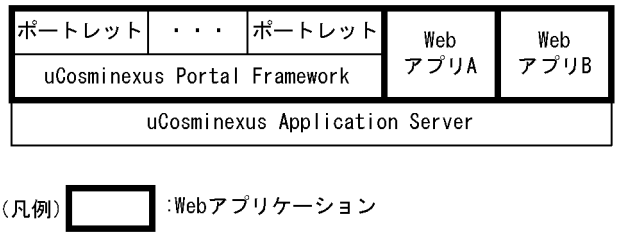
呼び出し条件の種類	標準画面（最小化画面）	編集画面	最大化画面	新規ウィンドウ画面	JSP/ サーブレット画面
呼び出す action メソッド	現在表示しようとしている標準画面内にある全ポートレットの action メソッド（現在表示しようとしているホーム画面のレイアウト内に含まれるポートレットのものです。ただし，レイアウトがタブ形式の場合は，タブ内のものになります）。	現在表示しようとしているポートレットの action メソッド	現在表示しようとしているポートレットの action メソッド	現在表示しようとしているポートレットの action メソッド	現在表示しようとしているポートレットの action メソッド
send メソッドの宛先として指定できるポートレット	ポータルプロジェクト内の全ポートレット	ポータルプロジェクト内の全ポートレット	ポータルプロジェクト内の全ポートレット	ポータルプロジェクト内の全ポートレット	ポータルプロジェクト内の全ポートレット

3.8 日立 API ポートレットの動作

uCosminexus Portal Framework は，uCosminexus Application Server 上で動く Web アプリケーションであり，JSP/ サーブレットで構築されています。uCosminexus Portal Framework は，複数のポートレットで構成されます。日立 API ポートレットは，Servlet API のリクエストディスパッチで呼び出され，uCosminexus Portal Framework の延長として動作します。

uCosminexus Portal Framework と日立 API ポートレットの関係を次の図に示します。

図 3-3 uCosminexus Portal Framework と日立 API ポートレットの関係



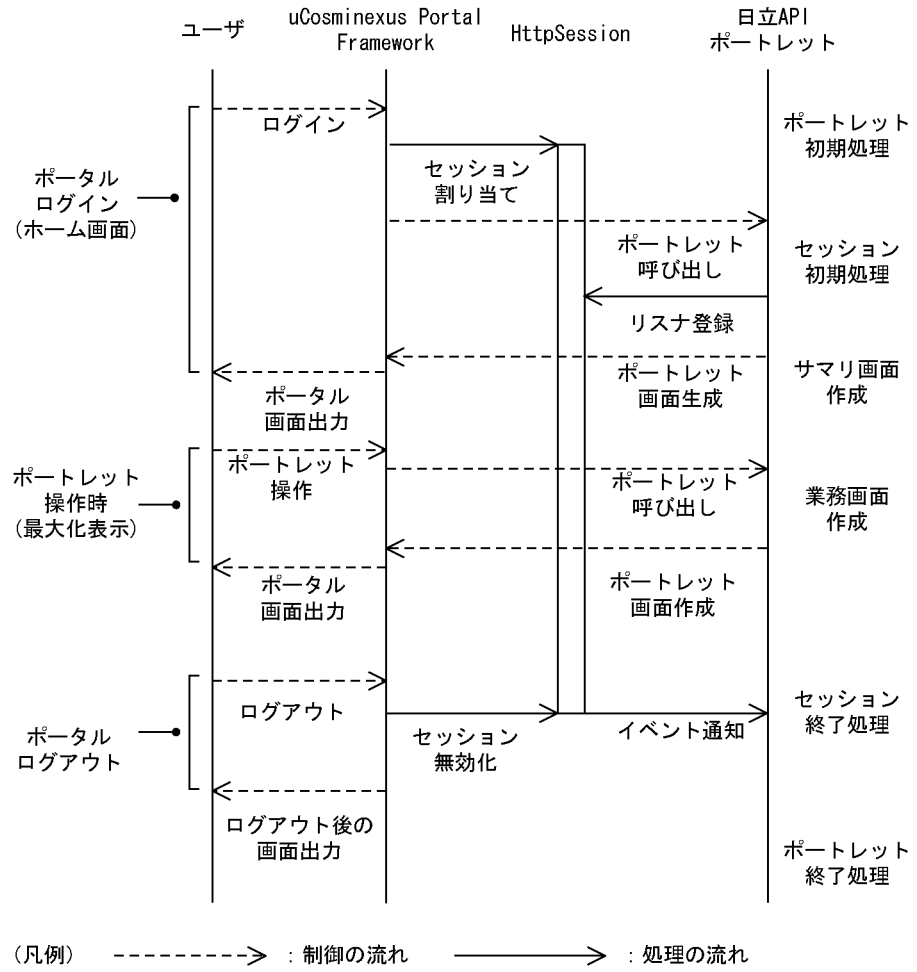
uCosminexus Application Server 上では，uCosminexus Portal Framework 以外にも，JSP やサーブレットで構築された Web アプリケーションが動作します。そのため，ポートレットは，uCosminexus Portal Framework およびほかのポートレットと，Web アプリケーションのコンテキストを共有します。

3.8.1 日立 API ポートレットのライフサイクル

uCosminexus Portal Framework では，JSP やサーブレットなどの Java の技術や HTML などの Web の技術を用いて独自のポートレットを作成できます。

日立 API ポートレットのライフサイクルを次の図に示します。

図 3-4　日立 API ポートレットのライフサイクル



ポートレット開発者は，ライブラリやユティリティクラス，および統合ユーザ管理フレームワークを利用して日立 API ポートレットを開発します。ライブラリの詳細は，「14.　ライブラリ」を参照してください。

（1）ポートレットの初期処理および終了処理

ポートレットの初期処理は，エントリポイントとして登録した JSP/ サーブレットの初期処理メソッドに記述します。JSP では，javax.servlet.jsp.JspPage#jspInit，サーブレットでは，javax.servlet.Servlet#init をオーバーライドします。初期処理では，ポートレットで共通の設定ファイルの読み込み，外部 DB とのコネクション確立など，ポートレットがサービスを提供する間使用するリソースを確保します。

ポートレットの終了処理は，エントリポイントとして登録した JSP/ サーブレットの終了処理メソッドに記述します。JSP では，javax.servlet.jsp.JspPage#jspDestroy，サーブ

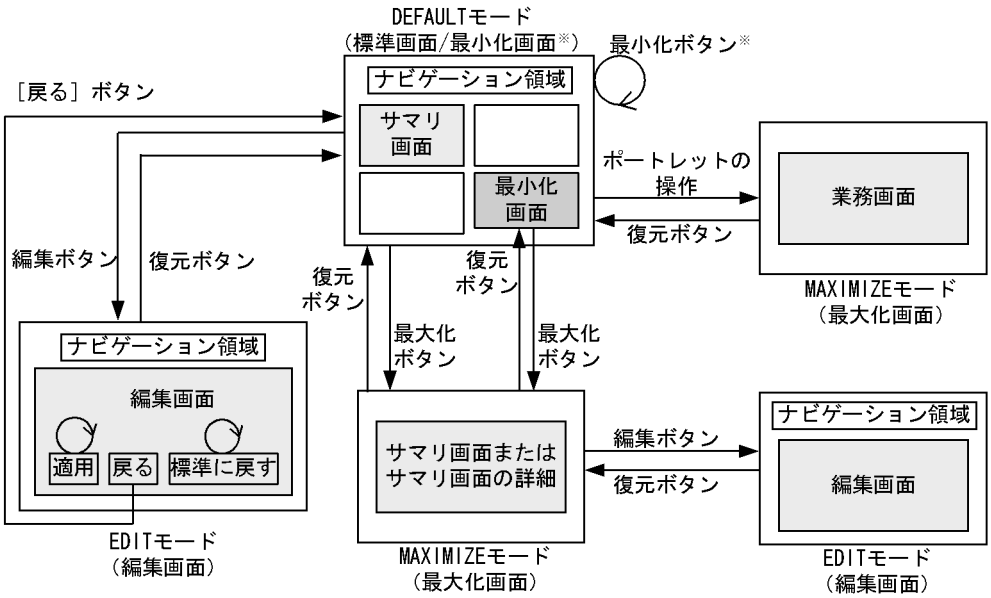
レットでは，javax.servlet.Servlet#destroy をオーバーライドします。終了処理では，初期処理時に確保したリソースを解放します。

(2) リクエスト処理

ポートレットへのリクエストに対する処理は，JSP/ サーブレットのサービスメソッドに記述します。JSP では，javax.servlet.jsp.HttpJspPage#jspService をオーバーライドします。サーブレットでは，javax.servlet.Servlet#service，または javax.servlet.http.HttpServlet の doGET もしくは doPOST のメソッドをオーバーライドします。

ボタンによるポートレットの画面遷移を次の図に示します。

図 3-5　ボタンによるポートレットの画面遷移



注※　標準画面でポートレットの最小化ボタンをクリックすると，標準画面内に最小化画面を表示します。

画面モードについては，「3.7.1　日立 API ポートレットの画面モードの種類」を参照してください。

サービスメソッド内では，PortletInfoBean#getMode で画面モードを取得します。ユーザの操作時に呼び出されるメソッドおよびポートレットの処理内容を次の表に示します。

表 3-5 ユーザ操作とポートレットの処理内容

ユーザの操作	呼び出されるメソッド	画面モード	ポートレットの処理内容
ユーザのホーム画面にアクセス	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	DEFAULT	サマリ画面を作成
最大化ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	MAXIMIZE	• サマリ画面を作成 • 必要に応じてサマリの詳細を表示
最大化画面で復元ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	DEFAULT	サマリ画面を作成
ポートレットユティリティタグライブラリで作成した JSP/ サーブレットのリンクをクリック	指定した JSP/ サーブレットのサービスメソッド	MAXIMIZE	業務処理および業務画面を作成
最大化画面で編集ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	EDIT	編集画面を表示
最大化画面の編集画面で復元ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	MAXIMIZE	最大化画面を表示
標準画面内の最小化画面で最大化ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	MAXIMIZE	最大化画面を表示
最大化画面で復元ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	DEFAULT	標準画面内に最小化画面を表示
編集ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	EDIT	編集画面を表示
編集画面で［適用］ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	EDIT	パーソナライズ情報を更新して，再度編集画面を表示
編集画面で［戻る］ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	DEFAULT	パーソナライズ情報を更新しないで，ポートレットのサマリ画面を表示
編集画面で［標準に戻す］ボタンをクリック	日立 API ポートレットに登録した JSP/ サーブレットのサービスメソッド	EDIT	システム管理者が設定した内容またはポートレットの初期設定に戻して，再度編集画面を表示
IFRAME 内のコンテンツを表示	IFRAME 内の JSP/ サーブレットのサービスメソッド	IFRAME	IFRAME 内に表示するコンテンツを作成

ユーザの操作	呼び出されるメソッド	画面モード	ポートレットの処理内容
ポートレットユティリティタグライブラリで作成した IFRAME 内の JSP/ サーブレットのリンクをクリック	IFRAME 内の JSP/ サーブレットのサービスメソッド	IFRAME	IFRAME 内に表示するコンテンツを作成

(3) セッションの初期処理および終了処理

セッションの初期処理および終了処理では，ユーザに固有のパーソナライズデータを取得したり，保存したりします。セッションの初期処理および終了処理は，セッションに対するイベントリスナ（HttpSessionBindingListener）を使用して実現します。エントリポイントとして登録した JSP/ サーブレットのサービスメソッド内で，開始・終了処理を記述したイベントリスナを HttpSession に登録します。ログインおよびログアウト処理のサンプルについては，「付録 A.4　ログインログアウトの処理」を参照してください。

3.8.2 URL 変換

日立 API ポートレット内で，ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL は，ドキュメントベースを基とした絶対 URL 形式，または {PROJECT_HOME} を基点とした絶対パス形式に変換されます。これを **URL 変換**と呼びます。ポートレットユティリティタグライブラリを使用すると，相対パス形式の URL が自動変換されます。ポートレットユティリティクラスライブラリを使用すると，任意の URL 変換種別を選択できます。ポートレットユティリティタグライブラリの詳細は，「14.2　ポートレットユティリティタグライブラリ」を，ポートレットユティリティクラスライブラリの詳細は，「14.4　ポートレットユティリティクラスライブラリ」を参照してください。

URL 変換は，部分識別子（"#fragment"），クエリ（"?query"）に対応します。";" パラメタには対応していません。また，URL 属性値が絶対 URL 形式，または絶対パス形式である場合は URL 変換されません。相対パス形式の URL だけが URL 変換されます。絶対 URL 形式，絶対パス形式，および相対パス形式の例を次に示します。

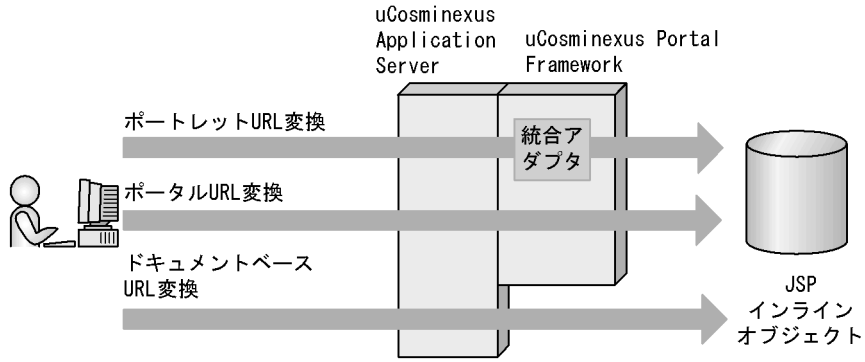
- 絶対 URL 形式：http://server/portal/sample/index.jsp
- 絶対パス形式：/portlets/sample/index.jsp
- 相対パス形式：index.jsp

なお，SSL アクセラレーターまたはリバースプロキシを使用した環境で uCosminexus Portal Framework を使用する場合には，URL 変換規則を切り替える必要があります。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「SSL アクセラレーターまたはリバースプロキシ使用時の設定」の説明を参照してください。

(1) URL 変換種別

URL 変換には，次の図に示す変換種別があります。

図 3-6 URL 変換種別



ポートレット URL 変換

ポートレット URL 変換は，ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，日立 API ポートレットをポータルの統合画面内で遷移させるための URL に変換することです。ポータル画面で View を統合するための統合アダプタを経由します。必要に応じて HttpServletResponse#encodeURL を呼び出して，URLRewriting をします。

ポータル URL 変換

ポータル URL 変換は，ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，uCosminexus Portal Framework を経由してコンテンツを取得するための URL に変換することです。uCosminexus Portal Framework を経由するので，uCosminexus Portal Framework の API を使用できます。また，ポートレットのアクセス制御も適用されます。ただし，統合アダプタを経由しないため，ポータルの統合画面内で遷移できません。

ドキュメントベース URL 変換

ドキュメントベース URL 変換は，ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，uCosminexus Portal Framework を経由しないでコンテンツを外部 Web サーバから直接取得するための URL に変換することです。URLRewriting はしません。また，uCosminexus Portal Framework の API を使用できません。

3.8.3 ポートレットユティリティタグライブラリ使用時の画面遷移

ポートレットユティリティタグライブラリを使用すると，相対パス形式の URL が自動変換されます。

ポートレットユティリティタグライブラリ使用時に適用される URL 変換を次の表に示します。

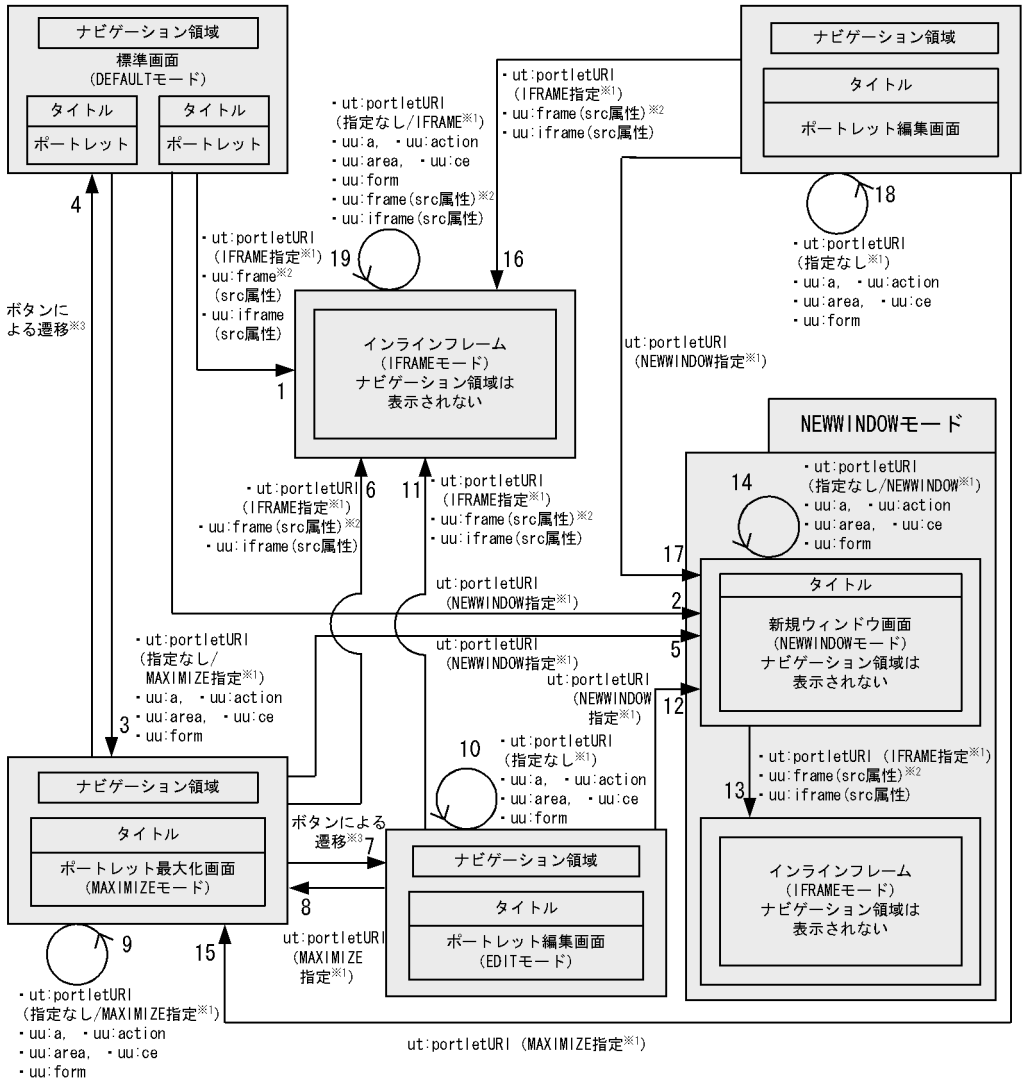
表 3-6 ポートレットユティリティタグライブラリ使用時に適用される URL 変換

タグ	適用される URL 変換
<uu:a>，<uu:action>，<uu:area>，<uu:ce>，<uu:form>	ポートレット URL 変換 （インラインフレーム内および PortletURI#transportalURI で生成された URL 内では，ポータル URL 変換）
<uu:iframe>（src 属性），<uu:frame>（src 属性）※	ポータル URL 変換
<uu:iframe>（longdesc 属性），<uu:img>，<uu:input>，<uu:object>，<uu:script>，<uu:frame>（longdesc 属性）※	ドキュメントベース URL 変換

注※ <uu:frame> タグは，iframe 内でだけ使用できます。

ポートレットユティリティタグライブラリ使用時のポートレットの画面遷移を次の図に示します。

図 3-7　ポートレットユティリティタグライブラリ使用時のポートレットの画面遷移



（凡例）　⟶　：遷移

注※1　<ut:portletURI>タグの属性に指定する画面モード
注※2　<uu:frame>タグは，インラインフレーム内だけで使用できます。
注※3　ボタンによる遷移は，「3.8.1(2)　リクエスト処理」を参照してください。

ユーザの操作時に呼び出されるタグ，および適用される URL 変換を次の表に示します。なお，項番は図中の番号に対応しています。

表 3-7 ユーザの操作時に呼び出されるタグおよび適用される URL 変換

項番	ユーザの操作	呼び出されるタグ	画面モード	URL 変換種別
1	標準画面でインラインフレームにコンテンツを表示	• ut:portletURI（IFRAME 指定） • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	ポータル URL 変換
2	標準画面から新規ウィンドウ画面を表示	ut:portletURI（NEWWINDOW 指定）	NEWWINDOW	―
3	標準画面からポートレット最大化画面を表示	• ut:portletURI（指定なし／MAXIMIZE 指定） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3	MAXIMIZE	ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換）
4	ボタンをクリック※4	―	DEFAULT	―
5	ポートレット最大化画面から新規ウィンドウを表示	ut:portletURI（NEWWINDOW 指定）	NEWWINDOW	―
6	ポートレット最大化画面でインラインフレームにコンテンツを表示	• ut:portletURI（IFRAME 指定） • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	ポータル URL 変換
7	ボタンをクリック※4	―	EDIT	―
8	ポートレット最大化画面の編集画面で［更新］ボタンなどをクリック	• ut:portletURI（指定なし） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3	EDIT	ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換）
9	ポートレット最大化画面の編集画面からポートレット最大化画面に戻る操作	ut:portletURI（MAXIMIZE 指定）	MAXIMIZE	―
10	ポートレット最大化画面で［更新］ボタンなどをクリック	• ut:portletURI（指定なし／MAXIMIZE 指定） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3	MAXIMIZE	ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換）

項番	ユーザの操作	呼び出されるタグ	画面モード	URL 変換種別
11	ポートレット最大化画面の編集画面でインラインフレームにコンテンツを表示	• ut:portletURI（IFRAME 指定） • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	ポータル URL 変換
12	ポートレット最大化画面の編集画面から新規ウィンドウ画面を表示	ut:portletURI（NEWWINDOW 指定）	NEWWINDOW	―
13	新規ウィンドウ画面でインラインフレームにコンテンツを表示	• ut:portletURI（IFRAME 指定） • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	ポータル URL 変換
14	新規ウィンドウ画面で［更新］ボタンなどをクリック	• ut:portletURI（指定なし／NEWWINDOW 指定） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3	NEWWINDOW	ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換）
15	ポートレット編集画面からポートレット最大化画面を表示	ut:portletURI（MAXIMIZE 指定）	MAXIMIZE	―
16	ポートレット編集画面でインラインフレームにコンテンツを表示	• ut:portletURI （IFRAME 指定） • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	ポータル URL 変換
17	ポートレット編集画面から新規ウィンドウ画面を表示	ut:portletURI（NEWWINDOW 指定）	NEWWINDOW	―
18	ポートレット編集画面で［適用する］ボタンなどをクリック	• ut:portletURI（指定なし） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3	EDIT	ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換）

項番	ユーザの操作	呼び出されるタグ	画面モード	URL 変換種別
19	インラインフレームで［再表示］ボタンなどをクリック	• ut:portletURI（指定なし／IFRAME 指定） • uu:a ※3 • uu:action ※3 • uu:area ※3 • uu:ce ※3 • uu:form ※3 • uu:frame（src 属性）※1※2 • uu:iframe（src 属性）※1	IFRAME	• ポートレット URL 変換（iframe 内および PortletURI#transportalURI で生成された URI 内では，ポータル URI 変換） • ポータル URL 変換

（凡例）

－：該当しません。

注※1　<uu:frame>（src 属性）および <uu:iframe>（src 属性）タグは，ポータル URL 変換を行います。

注※2　<uu:frame> タグは，インラインフレーム内でだけ使用できます。

注※3　<uu:a>，<uu:action>，<uu:area>，<uu:ce>，および <uu:form> タグは，ポートレット URL 変換を行います。ただし，インラインフレーム内および PortletURI#transportalURI で生成された URL 内では，ポータル URL 変換を行います。

注※4　ボタンによる遷移は，「3.8.1(2) リクエスト処理」を参照してください。

注意事項

各画面モードで次のタグを使用すると，ポータル外に遷移します。

<uu:frame>（longdesc 属性），<uu:img>，<uu:input>，および <uu:script> タグ

ただし，<uu:frame> タグは，インラインフレーム内でだけ使用できます。

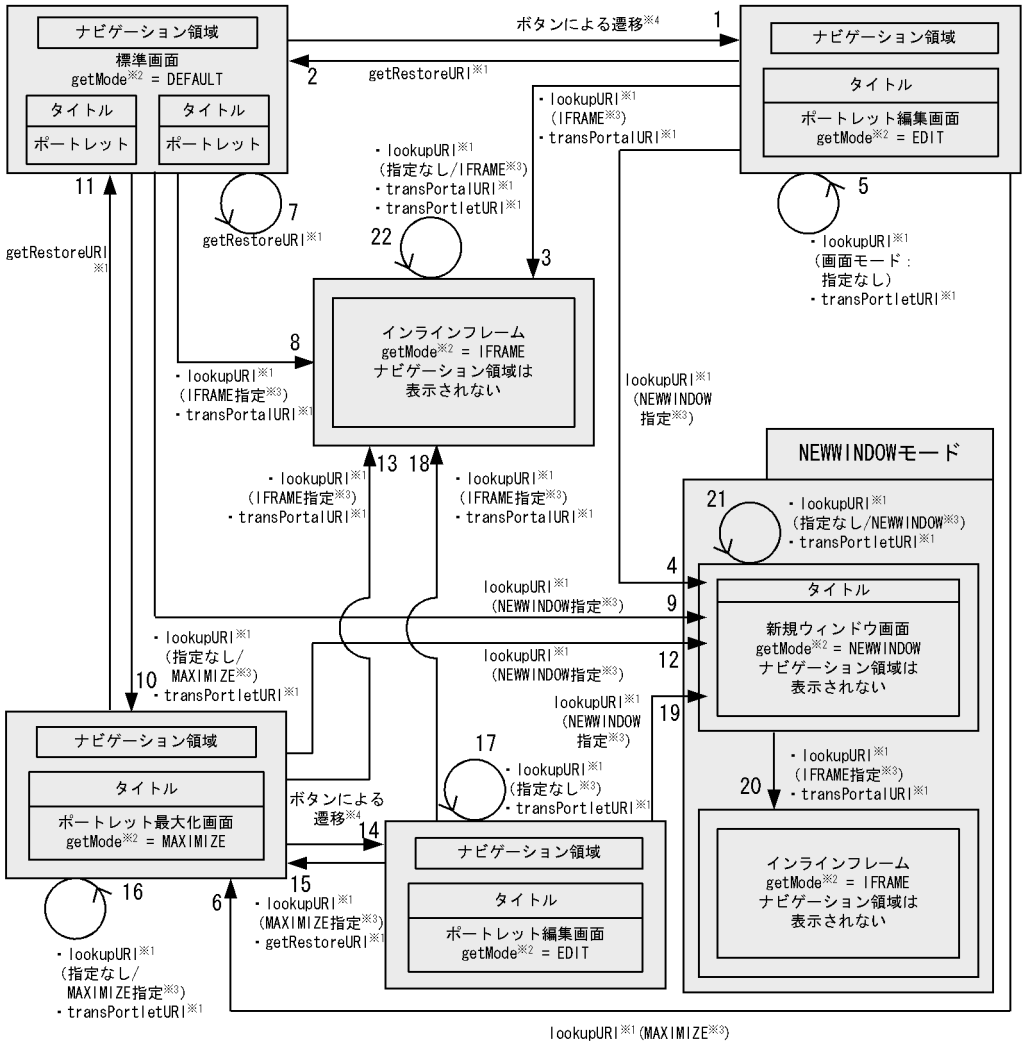
画面モードについては，「3.7.1　日立 API ポートレットの画面モードの種類」を参照してください。

3.8.4 ポートレットユティリティクラスライブラリ使用時の画面遷移

ポートレットユティリティクラスライブラリを使用すると，任意の URL 変換種別を選択できます。

ポートレットユティリティクラスライブラリ使用時のポートレットの画面遷移を次の図に示します。

図 3-8　ポートレットユティリティクラスライブラリ使用時のポートレットの画面遷移



（凡例）　⟶　：遷移

注※1　PortletURIクラスのURLを生成するメソッド
注※2　PortletInfoBeanのポートレットの画面モード名を取得するメソッド
注※3　lookupURIメソッドのパラメタに指定する画面モード
注※4　ボタンによる遷移は，「3.8.1(2)　リクエスト処理」を参照してください。

ユーザの操作時に呼び出されるメソッド，および使用する URL 変換種別を次の表に示します。なお，項番は図中の番号に対応しています。

表 3-8　ユーザの操作時に呼び出されるメソッドおよび使用する URL 変換種別

項番	ユーザの操作	呼び出されるメソッド	画面モード	URL 変換種別
1	ボタンをクリック※1	—	EDIT	—

項番	ユーザの操作	呼び出されるメソッド	画面モード	URL 変換種別
2	ポートレット編集画面から標準画面に戻る操作	getRestoreURI	DEFAULT	―
3	ポートレット編集画面でインラインフレームにコンテンツ表示	• lookupURI（IFRAME 指定） • transPortalURI ※2	IFRAME	ポータル URL 変換
4	ポートレット編集画面から新規ウィンドウを表示	lookupURI（NEWWINDOW 指定）	NEWWINDOW	―
5	ポートレット編集画面で［適用する］ボタンなどをクリック	• lookupURI（指定なし） • transPortletURI ※3	EDIT	ポートレット URL 変換
6	ポートレット編集画面から最大化画面を表示	lookupURI（MAXIMIZE 指定）	MAXIMIZE	―
7	標準画面で［更新］ボタンなどをクリック	getRestoreURI	DEFAULT	―
8	標準画面でインラインフレームにコンテンツ表示	• lookupURI（IFRAME 指定） • transPortalURI ※2	IFRAME	ポータル URL 変換
9	標準画面から新規ウィンドウ画面を表示	lookupURI（NEWWINDOW 指定）	NEWWINDOW	―
10	標準画面から最大化画面を表示	• lookupURI（指定なし／MAXIMIZE 指定） • transPortalURI ※3	MAXIMIZE	ポートレット URL 変換
11	ポートレット最大化画面で標準画面に戻る操作	getRestoreURI	DEFAULT	―
12	ポートレット最大化画面から新規ウィンドウ画面を表示	lookupURI（NEWWINDOW 指定）	NEWWINDOW	―
13	ポートレット最大化画面でインラインフレームにコンテンツを表示	• lookupURI（IFRAME 指定） • transPortalURI ※2	IFRAME	ポータル URL 変換
14	ボタンをクリック※1	―	EDIT	―
15	ポートレット最大化画面の編集画面からポートレット最大化画面に戻る操作	• lookupURI（MAXIMIZE 指定） • getRestoreURI	MAXIMIZE	―
16	ポートレット最大化画面で［更新］ボタンなどをクリック	• lookupURI（指定なし／MAXIMIZE 指定） • transPortletURI ※3	MAXIMIZE	ポートレット URL 変換

項番	ユーザの操作	呼び出されるメソッド	画面モード	URL 変換種別
17	ポートレット最大化画面の編集画面で［適用する］ボタンなどをクリック	• lookupURI（指定なし） • transPortletURI ※3	EDIT	ポートレット URL 変換
18	ポートレット最大化画面の編集画面でインラインフレームにコンテンツを表示	• lookupURI（IFRAME 指定） • transPortalURI ※2	IFRAME	ポータル URL 変換
19	ポートレット最大化画面の編集画面から新規ウィンドウ画面を表示	lookupURI（NEWWINDOW 指定）	NEWWINDOW	―
20	新規ウィンドウ画面でインラインフレームにコンテンツを表示	• lookupURI（IFRAME 指定） • transPortalURI ※2	IFRAME	ポータル URL 変換
21	新規ウィンドウ画面で［更新］ボタンなどをクリック	• lookupURI（指定なし／NEWWINDOW 指定） • transPortletURI ※3	NEWWINDOW	ポートレット URL 変換
22	インラインフレームで［再表示］ボタンなどをクリック	• lookupURI（指定なし／IFRAME 指定） • transPortalURI ※2 • transPortletURI ※3	IFRAME	• ポータル URL 変換 • ポートレット URL 変換

（凡例）

―：該当しません。

注※1　ボタンによる遷移は，「3.8.1(2)　リクエスト処理」を参照してください。

注※2　transPortalURI メソッドは，ポータル URL 変換を行います。

注※3　transPortletURI メソッドは，ポートレット URL 変換を行います。

画面モードについては，「3.7.1　日立 API ポートレットの画面モードの種類」を参照してください。

3.9 ポートレットの無応答監視

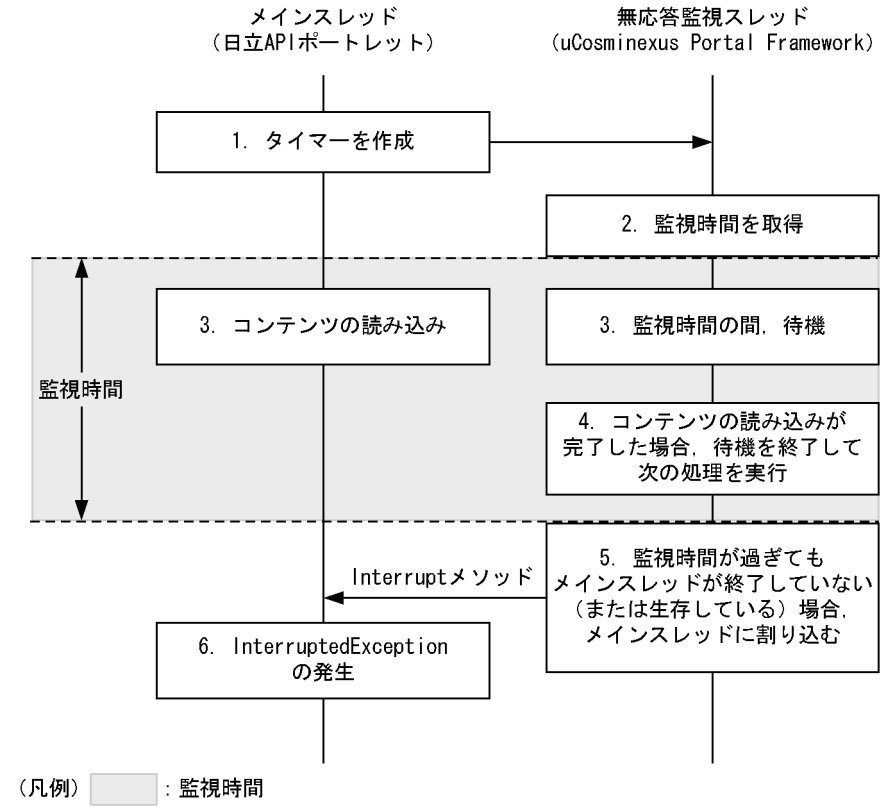
uCosminexus Portal Framework に登録したポートレットが無応答になると，uCosminexus Portal Framework 全体が停止してしまいます。そのため，ポートレット開発者は，uCosminexus Portal Framework が提供する無応答監視機能を利用して，無応答にならないポートレットを作成する必要があります。

無応答監視機能を利用するには，次に説明する無応答監視の処理シーケンスに従ってポートレット作成してください。

3.9.1 無応答監視の処理シーケンス

無応答監視の処理シーケンスを次の図に示します。

図 3-9 無応答監視の処理シーケンス



図の説明

1. 日立 API ポートレット内のメインスレッドは，uCosminexus Portal Framework に対してタイマーを作成し無応答監視スレッドを実行します。

2. uCosminexus Portal Framework 内の無応答監視スレッドは，JSP ポートレットの無応答監視時間をポートレット定義ファイルのパラメタ（hptl.MultiJSP.tags.watch）から取得します。
   ポートレット定義ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット定義ファイル（jetspeed-config.jcfg）」の説明を参照してください。
3. 日立 API ポートレット内のメインスレッドおよび uCosminexus Portal Framework 内の無応答監視スレッドでは，それぞれ次に示す処理をします。
   日立 API ポートレット内のメインスレッド
   RequestDispatcher インタフェースの forward メソッド※1 を利用して日立 API ポートレットに表示するコンテンツ（iframe を利用している場合は iframe 内のコンテンツを含む）を読み込みます。
   uCosminexus Portal Framework 内の無応答監視スレッド
   2. で取得した無応答監視時間の間，待機します。
4. 日立 API ポートレット内のメインスレッドがコンテンツ（iframe を利用している場合は iframe 内のコンテンツを含む）の読み込みを完了した場合，uCosminexus Portal Framework 内の無応答監視スレッドは，待機を終了して次の処理を実行します。
5. 2. で取得した無応答監視時間を過ぎても（タイムアウトになっても）JSP ポートレット内のメインスレッドが終了していない場合，または生存している場合，
   uCosminexus Portal Framework 内の無応答監視スレッドは，interrupt メソッド※2 を使用してメインスレッドに割り込みます（interrupt を通知します）。
6. 日立 API ポートレット内のメインスレッドで InterruptedException※2 が発生します。
   注意事項
   メインスレッドでは，例外が発生した場合でも実行する必要がある処理（共通資源の解放など）は finally ブロックに記述してください。

注※1　J2EE の API です。

注※2　J2SE の API です。

なお，uCosminexus Portal Framework が提供しているライブラリのすべてのクラスは，Cosminexus のメソッドキャンセル機能の保護区に設定されています。メソッドキャンセル機能の保護区については，マニュアル「Cosminexus　アプリケーションサーバ V8　機能解説　運用／監視／連携編」，またはマニュアル「Cosminexus V9　アプリケーションサーバ　機能解説　運用／監視／連携編」を参照してください。

3.9.2　無応答監視の interrupt の通知回数

無応答監視がタイムアウトになったとき，uCosminexus Portal Framework 内の無応答監視メソッドは，設定された回数だけ日立 API ポートレット内のメインスレッドに

interrupt を通知します。interrupt の通知回数は，ポートレットを登録するときにポートレット単位で設定できます。

次に，interrupt の通知回数の設定方法と interrupt の通知回数によって異なる uCosminexus Portal Framework の動作について説明します。

interrupt の通知回数の設定方法

interrupt の通知回数は，デプロイ定義ファイル（hportlet.xml），ポートレット定義ファイルまたは Portal Manager で設定します。

デプロイ定義ファイルで interrupt の通知回数を設定する場合，<config-param> の子要素（<param-name> および <param-value>）を使用します。次に，デプロイ定義ファイルの設定例を示します。

```
<portlet-app>
 <portlet>
         :
  <config-param>
         :
    <param-name>hptl.portlet.nonresponse.timeout.trytimes</
param-name>
    <param-value>3</param-value>
         :
  </config-param>
         :
 </portlet>
</portlet-app>
```

設定例では，interrupt を 3 回通知します。

ポートレット定義ファイルで interrupt の通知回数を設定する方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット定義ファイル（jetspeed-config.jcfg）」の説明を，Portal Manager で interrupt の通知回数を設定する方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「日立 API ポートレット（PAR 形式以外）および File ポートレットの設定」の説明を参照してください。

interrupt の通知回数によって異なる uCosminexus Portal Framework の動作

interrupt の通知回数に設定した値によって uCosminexus Portal Framework の動作は異なります。interrupt の通知回数と uCosminexus Portal Framework の動作について次の表に示します。

表 3-9　interrupt の通知回数と uCosminexus Portal Framework の動作

interrupt の通知回数（単位：回）	uCosminexus Portal Framework の動作
省略時（記述なし）	1 回だけ interrupt を通知
0 以下	uCosminexus Portal Framework 実行時にエラー※
1 ～ 10	指定した回数だけ interrupt を通知

interrupt の通知回数（単位：回）	uCosminexus Portal Framework の動作
11 以上	uCosminexus Portal Framework 実行時にエラー※

注※　ポートレットにタイムアウト時間設定不正時エラー画面を表示します。

interrupt の通知回数に 1 ～ 10 を設定した場合，uCosminexus Portal Framework 内の無応答監視スレッドは次に示すタイミングで interrupt を通知します。

- 1 回目：ポートレット実行開始から無応答の監視時間が経過したとき
- 2 回目以降：前回の interrupt 通知から無応答の監視時間が経過したとき

3.10 サーブレットを用いたポートレットの開発

uCosminexus Portal Framework ではサーブレットを使用できます。サーブレットを用いたポートレットを使用する場合のサーブレットのマッピング，およびサーブレットと JSP の連携方法について説明します。

3.10.1 サーブレットのマッピング

サーブレットを用いたポートレットを登録する場合，日立 API ポートレットを登録する前に，Web アプリケーションの DD（web.xml）でサーブレットをマッピングする必要があります。**サーブレットのマッピング**とは，登録する日立 API ポートレットで作成したサーブレットを呼び出す URL パターンを定義することです。定義した URL パターンを含む URL を開くことでサーブレットが実行されます。

なお，サーブレットのマッピング方法は，日立 API ポートレットの種類（PAR ファイルを作成する日立 API ポートレット，および PAR ファイルを作成しない日立 API ポートレット）によって異なります。

PAR ファイルを作成する日立 API ポートレット

PAR ファイルを作成する前に，次の設定を行います。

- 新規に web.xml を作成し，{PROJECT_HOME} ¥WEB-INF に格納されている web.xml に追記したいマッピング定義を記述します。
- デプロイ定義ファイル（hportlet.xml）の <device> タグの子要素 <url> タグに，サーブレット名（記述例では「/Controller」）を記述します。デプロイ定義ファイルの詳細は，「3.13.2　デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。

web.xml に，マッピングする URL としてポートレットをデプロイしたあとのディレクトリ下（記述例では「/adportlets/mvctest/Controller」）を設定します。ポートレットをデプロイした場合，次のディレクトリに作成されます。

/adportlets/<**ポートレット名**>/

web.xml の記述例を次に示します。

```
<servlet>
  <servlet-name>Controller</servlet-name>
  <servlet-class>mydomain.myapp</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Controller</servlet-name>
  <url-pattern>/adportlets/mvctest/Controller</url-pattern>
</servlet-mapping>
```

デプロイ定義ファイルでの記述例を次に示します。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<portlet-app>
  <portlet>
    <portlet-name>mvctest</portlet-name>
    <portlet-type>type</portlet-type>
    <portlet-parent>MultiJSP</portlet-parent>
    <title>title</title>
    <description>description</description>
    <windowtitle>windowtitle</windowtitle>
    <!-ポートレット起動パラメタ -->
    <config-param>
        <param-name>param-name</param-name>
        <param-value>param-value</param-value>
    </config-param>
    <supports>
        <timeout>timeout</timeout>
    </supports>
    <!-最大化・最小化等の設定 -->
    <allows>
        <maximize />
   </allows>
    <device media="HTML">
       <url>/Controller</url>
       <supports>
        <personalize />
       </supports>
    </device>
  </portlet>
</portlet-app>
```

PAR ファイルを作成しない日立 API ポートレット

web.xml にサーブレットのマッピング定義を追記します。

web.xml は，次に示すディレクトリに格納しています。

格納ディレクトリ

{PROJECT_HOME}¥WEB-INF

web.xml に，サーブレット（記述例では「Controller」）をマッピングする URL としてポートレットのディレクトリ下（記述例では「/portlets/mvctest/Controller」）を設定します。

web.xml の記述例を次に示します。

```
<servlet>
  <servlet-name>Controller</servlet-name>
  <servlet-class>mydomain.myapp</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Controller</servlet-name>
  <url-pattern>/portlets/mvctest/Controller</url-pattern>
</servlet-mapping>
```

3.10.2 サーブレットからの JSP の呼び出し

MVC モデルをポートレット開発に適用して，Controller をサーブレット，View を JSP で開発する場合，サーブレットから JSP を呼び出す必要があります。ここでは，サーブレットから JSP を呼び出す例を示します。

JSP を呼び出すには，RequestDispatcher#include を使用します。サーブレットをポートレットのエントリポイントとして登録する場合には，doPost が doGet を呼び出すようにします。

サーブレットから JSP を呼び出す例を次に示します。

```
package mydomain;
import javax.servlet.*;
import javax.servlet.http.*;
import jp.co.hitachi.soft.portal.portlet.PortletURI;
import jp.co.hitachi.soft.portal.portlet.PortletUtils;
import jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean;

public class Controller extends HttpServlet {
  /*
   * JSPの相対パスをコンテキスト内絶対パスに変換
   **/
  private String getUrl(HttpServletRequest req, String url) {
    String current =

(String)req.getAttribute("javax.servlet.include.servlet_path");
    return current.substring(0, current.lastIndexOf('/')+1) + url;
  }

 public void doGet(HttpServletRequest req, HttpServletResponse res)
{
    String uri = "error.jsp";
    try {
      PortletInfoBean pInfo = new PortletInfoBean();
      pInfo.initBean(req);
      PortletInfoBean.Mode mode = pInfo.getMode();

      if (mode == PortletInfoBean.Mode.DEFAULT) {
        /** サマリ画面用業務ロジック呼び出し **/

        // サマリ画面用JSPのパス取得
        uri = getUrl(req, "summary.jsp");
      } else if (mode == PortletInfoBean.Mode.MAXIMIZE) {
        /** 最大化画面用業務ロジック呼び出し **/

        // 最大化画面用JSPのパス取得
        uri = getUrl(req, "main.jsp");
      }
      // includeを用いたView(JSP)の呼び出し
      getServletContext().getRequestDispatcher(uri).include(req,
res);
    } catch (Exception e) {
        // エラーメッセージ出力
    }
  }

  public void doPost(HttpServletRequest req, HttpServletResponse
```

```
res) {
    doGet(req, res);
  }

}
```

3.10.3 JSP からのサーブレットの呼び出し

サーブレットをポートレットディレクトリ下の URL にマッピングしておくと，JSP を呼び出す方法と同様に，include アクションを使用して相対パス形式でサーブレットを呼び出せます。

Web アプリケーションの DD（web.xml）のマッピング例を次に示します。

```
<servlet>
  <servlet-name>Handler</servlet-name>
  <servlet-class>mydomain.myapp</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Handler</servlet-name>
  <url-pattern>/portlets/mvctest/Handler</url-pattern>
</servlet-mapping>
```

JSP からサーブレットを呼び出す例を次に示します。

```
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletUtils" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
// サーブレット呼び出し
<jsp:include page="Handler" flush="true"/>
<%
    String uri = "error.jsp";
    try {
      PortletInfoBean pInfo = new PortletInfoBean();
      pInfo.initBean(request);
      PortletInfoBean.Mode mode = pInfo.getMode();
      if (mode == PortletInfoBean.Mode.DEFAULT) {
        /** サマリ画面用業務ロジック呼び出し */
        uri = "summary.jsp";
      } else if (mode == PortletInfoBean.Mode.MAXIMIZE) {
        /** 最大化画面用業務ロジック呼び出し */
        uri = "main.jsp";
      }
    } catch (Exception ioe) {
      // エラーメッセージ出力
    }
%>
// includeを用いたJSP(View)の呼び出し
<jsp:include page="<%= uri %>" flush="true"/>
```

3.11 ポートレットの開発モデル

日立 API ポートレットを開発する場合のアプローチについて説明します。

日立 API ポートレットの開発には次に示すアプローチがあります。

- 2 層モデル
- MVC モデル

2 層モデルは，画面遷移が少ない，業務ロジックが簡潔であるなど，単純なポートレットの開発に適しています。

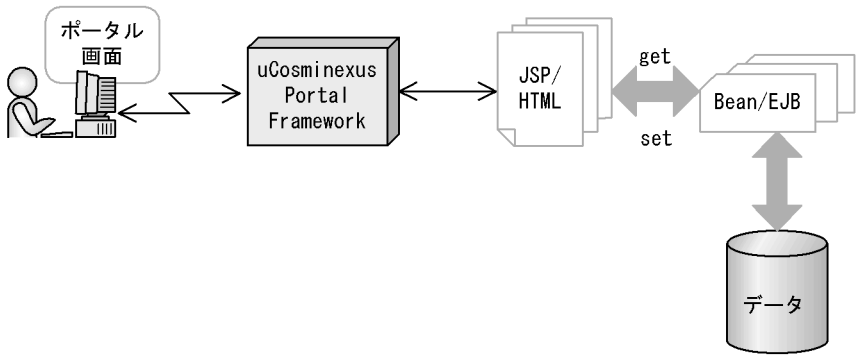
MVC モデルは，業務ロジックが複雑な場合，画面遷移が複雑な場合，または表示と業務ロジックを分離したい場合に適しています。

3.11.1 2 層モデル

2 層モデルは，JSP/HTML だけでリクエストを処理してクライアントへ送信するモデルです。画面遷移が少ないなど，単純なポートレットの開発方式です。

2 層モデルを次の図に示します。

図 3-10 2 層モデル



2 層モデルでは，JSP/HTML および Bean/EJB を使用します。JSP/HTML は，ユーザからのリクエストを基に，適切な Bean/EJB を呼び出します。Bean/EJB が業務処理をします。JSP/HTML は，リクエストや業務処理の結果を基に，適切な画面を表示します。

2 層モデルのサンプルを次に示します。このサンプルは，各ページへのハイパーリンクで構成されています。Web ブラウザからのリクエストで各ページが呼び出される形式です。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
```

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>

<UL>
  <LI><uu:a href="122501.html">「年末すす払い」に関する件</uu:a>
  <LI><uu:a href="122502.html">総務のURLが変更になりました。</uu:a>
  <LI><uu:a href="122101.html">規制改定の件</uu:a>
  <LI><uu:a href="121301.html">「給食シフト変更」のお知らせ</uu:a>
  <LI><uu:a href="120301.html">「チケット割引(12/3～)」のご案内</uu:a>
</UL>
```

3.11.2 MVC モデル

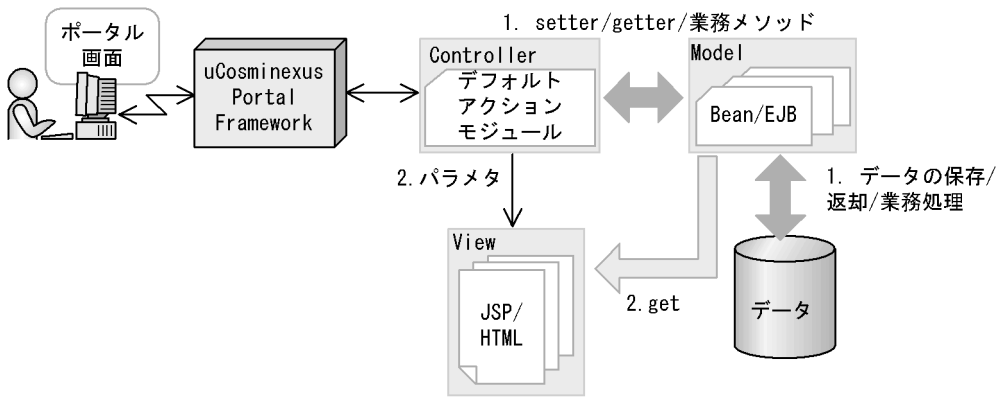
業務ロジックが複雑な場合は，MVC モデルと呼ばれるアプローチでポートレットを開発します。

MVC モデルとは，業務ロジック（Model），表示（View），制御（Controller）に分けてアプリケーションを開発する考え方です。業務と画面デザインを分離して開発できるため，効率良く開発できます。

MVC モデルをポートレット開発にあてはめた場合，Model には Bean/EJB，View には JSP/HTML，Controller にはデフォルトアクションモジュールを使用します。
Controller にデフォルトアクションモジュールを利用すると，ポータル統合画面と連携した画面を制御できます。

MVC モデルを次の図に示します。

図 3-11 MVC モデル



図の説明

1. Controller は，リクエストのパラメタ（URL のパスやクエリなど）を基に，適切な Model の setter/getter/ 業務メソッドを呼び出します。Model は，データの保存／返却／業務処理をします。
2. Controller は，リクエストのパラメタや Model の状態を基に，適切な View を選択し

呼び出します。View は，Model から必要なデータを取得し，画面を表示します。

Controller と View のサンプルを次に示します。

（1） Controller の開発

手順

1. **デフォルトアクションモジュールを用いて** Controller **を開発します。**
   Controller では，リクエストのパラメタや Model の状態を基に View へ渡すパラメタを決定します。デフォルトアクションモジュールから JSP（View）へのパラメタ渡しには，action メソッドで渡される request の属性を使用します。

   注意事項

   属性は，統合画面内の全ポートレットで共通に使用されます。そのため，属性の名前には，一意の名前を指定してください。

2. **開発したデフォルトアクションモジュールの名前を，デプロイ定義ファイル（**hportlet.xml**）の** config-param **要素に設定します。**
   config-param 要素の設定方法については，「8.3.3　ポートレットのデプロイ」を参照してください。

3. Portal Manager **でデプロイします。**
   ポートレットのデプロイについては，マニュアル「uCosminexus Portal Framework システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

Controller **のサンプル**

```
public class SampleControllerActionModule extends
DefaultActionModule
{
    public void action(HttpServletRequest req, HttpServletResponse
res) {
        req.setAttribute("page", Controller(req));
    }

    //コントローラ用メソッド
    public String Controller(HttpServletRequest req){
        //Attributeにcontrollerがセットされていたら，Stringオブジェクトに
変換する。
        String ControllerParameter =
(String)req.getParameter("controller");
        // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
        ControllerParameter = new
String(ControllerParameter.getBytes("iso-8859-1"),"Windows-31J");

        //デフォルト表示用JSP名をセットする。
        String JSPFILE = "default.jsp";

        //クエリにNEWMAILが指定されている場合
        if("NEWMAIL".equals(ControllerParameter)){
            //JSP表示を行うJSP名を返却する。
            return "newmail.jsp";
        }
        //クエリにADDRESSが指定されている場合
        else if("ADDRESS".equals(ControllerParameter)){
```

```
            //JSP表示を行うJSP名を返却する。
            return "address.jsp";
        }

        return JSPFILE;
    }
}
```

(2) View の開発

JSP を使用して，パラメタの値に応じて表示内容を切り替えるポートレットコンテンツを開発します。日立 API ポートレットのサンプルソース（index.jsp）は，パラメタ page の値（default.jsp，newmail.jsp，address.jsp）に応じて表示内容を切り替えます。

index.jsp

```
<hr>メールポートレットの画面<hr>
<%
//
//デフォルトアクションモジュールが設定した属性を判断し，画面に表示するコンテンツを切り替えます。
// ここでは，pageというAttributeを参照し， 表示コンテンツをControlします。

String view = (String)request.getAttribute("page");

if (page != null) {%>
<jsp:include page="<%= view %>" flush="true"/>
<% }%>
```

default.jsp

```
<hr>
初期状態の画面です。
```

newmail.jsp

```
<hr>
メールが到着した際の画面です。
```

address.jsp

```
<hr>
アドレス帳の画面です。
```

3.12 デバイスに対応するための設定（日立 API ポートレット）

日立 API ポートレットでは，PC だけでなく携帯電話（ｉ モード，および EZweb）からアクセスできるコンテンツも作成できます。コンテンツは，クライアントのデバイス（PC，ｉ モード，または EZweb）ごとに対応する言語で作成する必要があります。

ポータルにアクセスできるクライアントのデバイスの種類と，対応する言語を次の表に示します。

表 3-10 クライアントのデバイスの種類と対応する言語（日立 API ポートレット）

クライアントのデバイス	対応する言語
HTML4.01 に対応する Web ブラウザのある PC	HTML
i モード対応 HTML2.0 に対応する i モードの携帯電話	CHTML
Version 3.3 for HDML に対応する EZweb の携帯電話	HDML

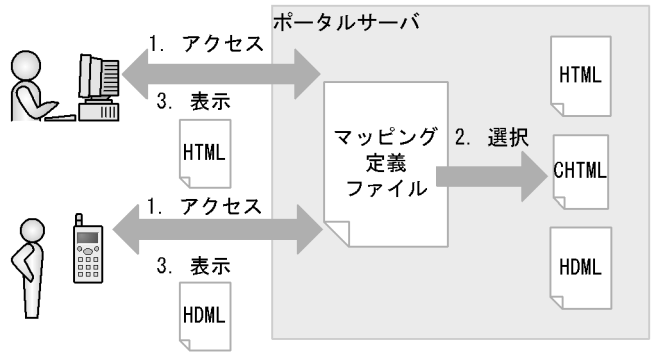
なお，クライアントのデバイスに合ったコンテンツを表示するために，対応する言語ごとに文書を書き換える必要があります。

- HTML ／ CHTML の場合：ドキュメントタイプの宣言の追加，および背景色の指定
- HDML の場合：ドキュメントタイプの宣言の追加

アクセスしてきたクライアントの種別を判別して，クライアントに合ったコンテンツを選択するために，マッピング定義ファイルを使用します。一つのポートレットに一つのマッピング定義ファイルが必要です。

マッピング定義ファイルとクライアント判別について次の図に示します。

図 3-12 マッピング定義ファイルとクライアント判別（日立 API ポートレット）



図の説明

1. ユーザがポータルにアクセスしたときに，マッピング定義ファイルでユーザのクライアント種別を判別します。
2. クライアント種別に合致したファイルを選択します。
3. 選択したファイルをポータルで表示します。

マッピング定義ファイルは，ポートレット登録時に作成されます。Portal Manager でポートレットを登録するときに，マッピング定義ファイルの作成場所を指定します。ポートレットごとに作成したポートレットディレクトリの直下を指定します。

3.13 日立 API ポートレットのアーカイブの作成

ここでは，ポートレットアーカイブの作成について説明します。ポートレットアーカイブとは，ポートレットの実行に必要な，ポートレットコンテンツ（JSP/HTML ファイル）およびデプロイ定義ファイル（hportlet.xml）などを一つにまとめた（パッケージした）ファイルのことです。このファイルを，**PAR ファイル**といいます。この PAR ファイルを利用すると，ポートレットごとにクラスやコンテンツのデプロイおよびアンデプロイができ，複数のポートレットを効率良く管理できます。

ポートレットアーカイブの作成手順を次に示します。

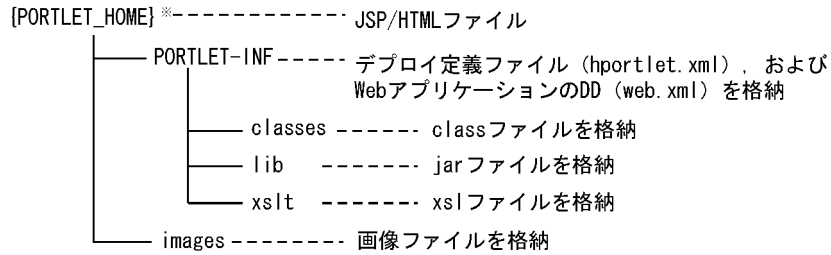
1. **PAR ファイルを作成するときのトップディレクトリ以下に，ポートレットコンテンツを格納します。**
   ここでは，PAR ファイルを作成するときのトップディレクトリを {PORTLET_HOME} と表記します。PAR ファイルを作成するときの，ポートレットのディクトリ構成については，「3.13.1　日立 API ポートレットのディレクトリ構成」を参照してください。
2. **デプロイ定義ファイル（hportlet.xml）を作成します。**
   デプロイ定義ファイルの詳細は，「3.13.2　デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。
3. **Web アプリケーションの DD（web.xml）を作成します。**
   Web アプリケーションの DD の詳細は，「3.13.3　Web アプリケーションの DD（web.xml）の作成」を参照してください。
4. **Java アーカイブツールを使用してパッケージします。**
   PAR ファイルを作成します。PAR ファイルを作成するには，コマンドプロンプト上でカレントディレクトリを {PORTLET_HOME} に移動し，次のコマンドを実行します。
   ```
   > jar cvf {parファイル名}.par
   ```
5. **Portal Manager，またはコマンドを使用して PAR ファイルをデプロイします。**
   Portal Manager，またはコマンドを使用した PAR ファイルのデプロイについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「日立 API ポートレット（PAR 形式）のデプロイ」の説明を参照してください。

3.13.1 日立 API ポートレットのディレクトリ構成

PAR ファイルを作成するときの，日立 API ポートレットのディレクトリ構成について説明します。

ポートレットを格納するときには，ポートレットごとにディレクトリを作成します。
PAR ファイル内のディレクトリ構成を次の図に示します。

図 3-13 日立 API ポートレットのディレクトリ



注※ PARファイル作成時のトップディレクトリです。
ディレクトリ名には，任意の名称を付けてください。

JSP/HTML ファイル

ポートレットディレクトリ直下に格納します。統合画面内で遷移させる JSP ファイルは同じポートレットディレクトリ下に格納します。

デプロイ定義ファイル（hportlet.xml）

デプロイ定義ファイルは，{PORTLET_HOME}¥PORTLET-INF に格納します。なお，デプロイ定義ファイルの詳細は「3.13.2 デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。

Web アプリケーションの DD（web.xml）

サーブレットのマッピング定義などに使用する Web アプリケーションの DD（web.xml）を {PORTLET_HOME}¥PORTLET-INF に格納します。このファイルは，{PROJECT_HOME}¥WEB-INF に格納されている web.xml に対して，サーブレットの定義やタグライブラリの定義を追加する場合にだけ新規に作成する必要があります。このファイルに記述した定義内容は，{PROJECT_HOME}¥WEB-INF に格納されている web.xml に追記されます。なお，Web アプリケーションの DD の詳細は，「3.13.3 Web アプリケーションの DD（web.xml）の作成」を参照してください。

クラスファイル

サーブレットやユティリティクラスなどのクラスファイルは，
{PORTLET_HOME}¥PORTLET-INF¥classes に格納します。
{PORTLET_HOME}¥PORTLET-INF¥classes に格納したクラスファイルのデプロイ後の展開先は，uCosminexus Portal Framework がポータルプロジェクト内で使用するクラスフォルダとなるため，名前が重複するおそれがあります。また，ディレクトリ名の大文字小文字の区別がない OS を利用している場合，クラスのパッケージ名が不正となる場合があります。そのため，JAR ファイルを使用して格納することを推奨します。
Portal Manager は，展開先に重複した名称のクラスファイルを発見した場合，PAR ファイルで指定されたクラスファイルを展開しません。また，クラスファイルを登録したすべてのポートレットをアンデプロイするまで，クラスファイルを削除しま

せん。なお，Portal Manager を利用しないでクラスファイルをコピーまたは削除した場合，関連するポートレットのデプロイまたはアンデプロイができなくなります。PAR ファイルにクラスファイルを含める場合は，十分に注意してください。

JAR ファイル

JAR ファイルは，{PORTLET_HOME}¥PORTLET-INF¥lib に格納します。なお，"hptl" で始まる JAR ファイル名は予約されているため使用できません。{PORTLET_HOME}¥PORTLET-INF¥lib に格納した JAR ファイルのデプロイ後の展開先は，uCosminexus Portal Framework がポータルプロジェクト内で使用するライブラリフォルダとなります。なお，JAR ファイルのファイル名は，デプロイ定義ファイル（hportlet.xml）で指定したポートレット名を接頭語とするファイル名にリネームされるため，ほかのポートレットとの名前重複を考慮する必要はありません。

画像ファイル

通常の Web アプリケーション作成時と同様に格納します。

3.13.2 デプロイ定義ファイル（hportlet.xml）の作成

デプロイ定義ファイルを作成します。**デプロイ定義ファイル**（hportlet.xml）とは，ポートレットのタイトルや URL など，uCosminexus Portal Framework で使用するポートレットの情報を設定するファイルです。デプロイ定義ファイルのサンプル，デプロイ定義ファイルで使用するタグの一覧およびデプロイ定義ファイル作成時の注意事項を次に示します。なお，この要素で指定する内容は，ポートレット定義ファイル（jetspeed-config.jcfg）の各ポートレット定義の parameter 要素で指定する内容に対応しています。ポートレット定義ファイル（jetspeed-config.jcfg）については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット定義ファイル（jetspeed-config.jcfg）」の説明を参照してください。

(1) サンプルのデプロイ定義ファイル

サンプルのデプロイ定義ファイルを次に示します。このデプロイ定義ファイルは，uCosminexus Portal Framework がサンプルとして提供している「sampleforum.par」に格納されているデプロイ定義ファイル（hportlet.xml）に，太字の要素（<windowtitle>，<support> および <timeout>）を追加した内容になっています。

「sampleforum.par」については，「10.3　ナビゲーションメニュー対応ポートレットの開発例」を参照してください。

サンプルのデプロイ定義ファイル

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<portlet-app>
  <portlet>
    <portlet-name>sampleforum</portlet-name>
    <portlet-type>ref</portlet-type>
    <portlet-parent>MultiJSP</portlet-parent>
```

```
    <title>Sample forum</title>
    <description>This is the Sample forum portlet.</description>
    <windowtitle lang="ja">Sample forum</windowtitle>
    <config-param>
      <param-name>hptl.module.action</param-name>

<param-value>jp.co.hitachi.soft.portal.accessories.sampleforum.Sam
pleForumActionModule</param-value>
      <param-name>hptl.portlet.defaulticon</param-name>
      <param-value>images/forum.gif</param-value>
      <param-name>hptl.inbox.support</param-name>
      <param-value>true</param-value>
      <param-name>hptl.inbox.filtering.support</param-name>
      <param-value>true</param-value>
      <param-name>hptl.navigationbar.support</param-name>
      <param-value>true</param-value>
    </config-param>
    <support>
      <timeout>10</timeout>
    </support>
    <allows>
      <maximize />
    </allows>
    <device media="HTML">
      <url>index.jsp</url>
      <supports>
        <personalize />
      </supports>
    </device>
  </portlet>
</portlet-app>
```

(2) デプロイ定義ファイルで使用するタグの一覧

タグの一覧を次の表に示します。

表 3-11 デプロイ定義ファイルのタグ一覧

タグ名	タグの出現回数	属性の出現回数	説明
<portlet-app>	1 回	—	デプロイ定義ファイルのルートタグ。
<portlet>	1 回	—	ポートレット定義情報の親タグ。
<portlet-name>	1 回	—	ポータルプロジェクト内で一意となるポートレット名。 指定できる最大文字数は，半角英数字 32 文字です。
<portlet-type>	1 回	—	常に「ref」を指定します。
<portlet-parent>	1 回	—	ポートレットの種類。 日立 API ポートレットの場合，「MultiJSP」を指定します。

タグ名	タグの出現回数	属性の出現回数	説明
<title lang="*言語種別*">	0 回以上	0 または 1 回	ポートレットのタイトルバーおよび利用者用レイアウトカスタマイズ画面の［ポートレット選択画面］に表示されるポートレットのタイトル。※1※2 全角および半角の文字列を使用できます。ただし，エンコード依存文字，機種依存文字，外字および制御コードの文字列は使用できません。指定できる最大文字数は，全角でも半角でも 80 文字です。 省略した場合のデフォルト値は，「No Title Set」です。 属性 lang で言語種別を指定できます。※3
<description lang="*言語種別*">	0 回以上	0 または 1 回	ポートレットの説明文。※1 全角および半角の表示可能文字列を使用できます。 全角および半角の文字列を使用できます。ただし，エンコード依存文字，機種依存文字，外字および制御コードの文字列は使用できません。指定できる最大文字数は，全角でも半角でも 80 文字です。 省略した場合のデフォルト値は，「No description found. Generic portlet」です。 属性 lang で言語種別を指定できます。※3
<windowtitle lang="*言語種別*">	0 回以上	0 または 1 回	新規ウィンドウを表示する場合のウィンドウのタイトル。※1※2 全角および半角の文字列を使用できます。ただし，エンコード依存文字，機種依存文字，外字および制御コードの文字列は使用できません。指定できる最大文字数は，全角でも半角でも 80 文字です。 省略した場合のデフォルト値は，<portlet-name> で指定したポートレット名です。 属性 lang で言語種別を指定できます。※3
<config-param>	0 または 1 回	－	ポートレット起動時のパラメタ（ポートレットの固有の情報）についての定義。
<param-name>	0 回以上	－	パラメタのプロパティ名。※4 プロパティ名は，ポートレット単位で決められます。ただし，「hptl」で始まる名称は予約語のため指定できません。
<param-value>	0 回以上	－	パラメタのプロパティ値。※4 <param-name> に指定したプロパティ名に対応する値を指定します。 プロパティ値は，ポートレット単位で決められます。ただし，「hptl」で始まる値は予約語のため指定できません。

タグ名	タグの出現回数	属性の出現回数	説明
</config-param>	0 または 1 回	−	終了タグ。
<supports>	0 または 1 回	−	<timeout> の親タグ。
<timeout>	0 または 1 回	−	ポートレットの無応答監視時間。※5 指定する値の単位は秒です。指定できる値の範囲は，0 ～ 86,400（整数）です。0 を指定した場合，無応答監視は行いません。 ポートレットの実行時間が無応答監視時間を超えた場合，エラー画面を表示します。
</supports>	0 または 1 回	−	終了タグ。
<allows>	0 または 1 回	−	画面表示モードの親タグ。
<minimize />	0 または 1 回	−	ポートレットの最小化表示の許可。 ポートレットがタイトル表示の場合，ポートレットの最小化ボタンを表示し，最小化できるようにします。 ポートレットに最小化表示を許可する場合，空要素タグ <minimize /> を記述します。なお，値を指定する必要はありません。
<close />	0 または 1 回	−	ポートレットの閉じるボタンの許可。 ポートレットがタイトル表示の場合，かつ強制表示ポートレットでない場合，ポートレットの閉じるボタンを表示し，閉じることができるようにします。 ポートレットに閉じるボタンを許可する場合，空要素タグ <close /> を記述します。なお，値を指定する必要はありません。
<maximize />	0 または 1 回	−	ポートレットの最大化表示の許可。 ポートレットがタイトル表示の場合，ポートレットの最大化ボタンを表示し，最大化できるようにします。 ポートレットに最大化表示を許可する場合，空要素タグ <maximize /> を記述します。なお，値を指定する必要はありません。
</allows>	0 または 1 回	−	終了タグ。

タグ名	タグの出現回数	属性の出現回数	説明
<device media="*デバイス種別*">	0 または 3 回まで	0 または 1 回	デバイス種別単位の情報についての定義。 属性 media でデバイス種別を指定できます。デバイス種別を指定するには，*デバイス種別*に次のどれかを指定します。 • HTML：PC 向けコンテンツ • CHTML：i モード向けコンテンツ • HDML：EZweb 向けコンテンツ 属性 media に「HTML」を指定した場合の開始タグ例を次に示します。 <device media="HTML"> 属性 media の指定は省略できません。<device> を記述する場合は必ず指定してください。
<url>	1 回	－	<device> の media 属性で指定したデバイス種別に対応するコンテンツのエントリポイント。エントリポイントには，ポートレットディレクトリからの絶対パスを指定します。
<supports>	0 または 1 回	－	<personalize /> の親タグ。
<personalize />	0 または 1 回	－	編集画面の存在についての定義。 <device> の media 属性で指定したデバイス種別に対応するポートレットが，編集画面（EDIT モード）を所有する場合，空要素タグ <personalize /> を記述します。なお，値を指定する必要はありません。
</supports>	0 または 1 回	－	終了タグ。
</device>	0 または 3 回まで	－	終了タグ。
</portlet>	1 回	－	終了タグ。
</portlet-app>	1 回	－	終了タグ。

（凡例）

－：該当しません。

注※ 1　タイトルに日本語を使用する場合は，xml 宣言文の encoding 属性に文字コードを指定する必要があります。

注※ 2　各タグで指定したタイトルが，最大化画面，および新規ウィンドウ画面で表示される場所を「(a) 最大化画面，および新規ウィンドウ画面で表示されるタイトル」に示します。

注※ 3　属性 lang で言語種別を指定するには，*言語種別*に次の形式で指定します。

言語コード［- 国コード［- バリアント］］

日本語で表示するポートレットのタイトル（<title> タグ）を指定する場合は，次のように設

定します。

```
<title lang="ja">
```

また，中国語（香港）で表示するポートレットのタイトル（<title> タグ）を指定する場合は，次のように設定します。

```
<title lang="zh-HK">
```

属性 lang の指定を省略した場合のデフォルト値は，「ja」です。言語の優先順位については，「(b) 言語の優先順位」を参照してください。

注※ 4　パラメタは，<param-name> および <param-value> を対にして，必要な数だけ <config-param> 〜 </config-param> 間に設定できます。パラメタを設定すると，次の形式で定義されます。

（<param-name> の設定内容）=（<param-value> の設定内容）

パラメタ（<param-name> および <param-value>）の記述方法については，「(1) サンプルのデプロイ定義ファイル」を参照してください。

設定できるパラメタを「(c) 設定できるパラメタ」に示します。

ポートレットイベントにポートレットを対応させたい場合は，「8.3.3　ポートレットのデプロイ」を参照し，パラメタを設定してください。ナビゲーションメニューにポートレットを対応させたい場合は，「10.1.2　デプロイ定義ファイルの設定」を参照し，パラメタを設定してください。

注※ 5　Cosminexus Portal Framework　06-10 から，iframe モードやアクションモジュール内の処理に対しても無応答監視が適用になりました。

(a) 最大化画面，および新規ウィンドウ画面で表示されるタイトル

各タグで指定したタイトルが，最大化画面，および新規ウィンドウ画面で表示される場所を次に示します。ここでは，「(1) サンプルのデプロイ定義ファイル」のデプロイ定義ファイルで設定したタイトル（<title> で指定したタイトルおよび <windowtitle> で指定したタイトルはどちらも「Sample forum」）を表示しています。

図 3-14　<title> で指定したタイトルが表示される場所（最大化画面の場合）

図 3-15 <title> および <windowtitle> で指定したタイトルが表示される場所（新規ウィンドウ画面の場合）

(b) 言語の優先順位

uCosminexus Portal Framework は，クライアントからアクセスされた際，クライアントが要求する言語種別と属性 "lang" の指定値を比較し，クライアントに応答する最適な言語を決定します。次の条件を上から順に評価し，条件を満たす定義が存在したらその定義内容（タイトルや説明文）を応答します。

- 言語コード・国コード・バリアントが一致している。
- 言語コード・国コードが一致している。
- 言語コードが一致している。

なお，言語コード・国コード・バリアントがすべて一致しない場合，ポータル画面上では「No Title Set」が，ポータルの利用者用レイアウトカスタマイズ画面の［ポートレット選択画面］，および運用管理ポートレットでは，ポートレット名称（<portlet-name> タグ）が表示されます。

(c) 設定できるパラメタ

デプロイ定義ファイルでは，ポートレットの起動時のパラメタを設定できます。パラメタは，<param-name> および <param-value> を対にして，必要な数だけ <config-param> 〜 </config-param> 間に設定します。uCosminexus Portal Framework が提供する機能で設定できるパラメタの一覧を次の表に示します。

表 3-12 設定できるパラメタの一覧

<param-name> に設定するパラメタ名※1	<param-value> に設定するパラメタ値※1	設定内容
Cache-Enable	*Cache-Enable*	キャッシュについて設定します。 • true：キャッシュを使用します（デフォルト値）。 • false：キャッシュを使用しません。
hptl.help.url	*hptl.help.url*	ポートレットアクションのヘルプの URL を指定します。

<param-name> に設定するパラメタ名※1	<param-value> に設定するパラメタ値※1	設定内容
hptl.help.url.ja	*hptl.help.url.ja*	ポートレットアクションの日本語ヘルプの URL を指定します。
hptl.help.url.en	*hptl.help.url.en*	ポートレットアクションの英語ヘルプの URL を指定します。
hptl.help.url. 言語コード［- 国コード［- バリアント］］	*hptl.help.url. 言語コード［- 国コード［- バリアント］］*	ポートレットアクションの日本語と英語以外のヘルプの URL を指定します。国コードとバリアントは省略可能です。
hptl.EditMode	*hptl.EditMode*	編集モードを設定します。 • true：編集モードを設定します。 • false：編集モードを設定しません（デフォルト値）。 なお，空要素タグ <personalize /> を指定すると true が設定されます。
hptl.MinimizeMode	*hptl.MinimizeMode*	ポートレットがタイトル表示の場合，ポートレットの最小化ボタンを表示し，最小化できるようにするか指定します。 • true：ポートレットの最小化をサポートします。 • false：ポートレットの最小化をサポートしません（デフォルト値）。※2 なお，空要素タグ <minimize /> を指定すると true が設定されます。
hptl.CloseMode	*hptl.CloseMode*	ポートレットがタイトル表示の場合，かつ強制表示ポートレットでない場合，ポートレットの閉じるボタンを表示し，閉じることができるようにするか指定します。 • true：ポートレットのクローズをサポートします。 • false：ポートレットのクローズをサポートしません（デフォルト値）。※2 なお，空要素タグ <close /> を指定すると true が設定されます。
hptl.MaximizeMode	*hptl.MaximizeMode*	ポートレットがタイトル表示の場合，ポートレットの最大化ボタンを表示し，最大化できるようにするか指定します。 • true：ポートレットの最大化をサポートします（デフォルト値）。 • false：ポートレットの最大化をサポートしません。※2 なお，空要素タグ <maximize /> を指定すると true が設定されます。

<param-name> に設定するパラメタ名※1	<param-value> に設定するパラメタ値※1	設定内容
hptl.NewWindowMode	*hptl.NewWindowMode*	ポートレットがタイトル表示の場合，ポートレットの別画面表示ボタンを表示し，新規ウィンドウにポートレットを表示できるようにするか指定します。 • true：ポートレットの新規ウィンドウ表示をサポートします。 • false：ポートレットの新規ウィンドウ表示をサポートしません（デフォルト値）。
hptl.Size	*hptl.Size*	ポートレットの大きさを指定します。レイアウトパターンの設定をしている場合に指定できます。指定できる項目を次に示します。なお，次の項目以外を指定した場合，そのポートレットは表示されません。 • large：大きいサイズ • medium：中サイズ • small：小さいサイズ • 指定無し：レイアウトパターンの設定で指定したサイズ（デフォルト値）
hptl.MultiJSP.tags.watch	*timeout*	無応答監視時間を指定します。指定する値の単位は秒です。指定できる値の範囲は，0 ～ 86,400（整数）です。0 を指定した場合，無応答監視は行いません。 なお，<timeout> タグでも無応答監視時間の指定ができます。
hptl.portlet.nonresponse.timeout.trytimes	*hptl.portlet.nonresponse.timeout.trytimes*	無応答監視がタイムアウトになったときに，uCosminexus Portal Framework が日立 API ポートレットに interrupt を通知する回数を指定します。指定できる値の範囲は，1 ～ 10（整数）です。interrupt を通知する回数の指定がない場合（記述がない場合），1 回だけ interrupt を通知します。
hptl.module.action	*hptl.module.action*	ポートレットのイベント処理を記述するためのアクションモジュールのパッケージ名称を含めたクラスのフルパス名称を指定します。
hptl.navigationbar.support	*hptl.navigationbar.support*	ナビゲーションメニューに対応するかどうかを指定します。 • true：ナビゲーションメニューに対応します。 • false：ナビゲーションメニューに対応しません（デフォルト値）。
hptl.navigationbar.version	*hptl.navigationbar.version*	対応しているナビゲーションメニューの API のバージョンを指定します。「01-00」を指定してください（デフォルト値は 01-00）。
hptl.navigationbar.alltabflag	*hptl.navigationbar.alltabflag*	連携ポートレットをデフォルトメニューに設定するかどうかを指定します。 • true：ユーザにメニュー情報がない場合，デフォルトメニューとしてメニュー登録されます。 • false：ユーザにメニュー情報がない場合，デフォルトメニューとしてメニュー登録されません（デフォルト値）。

<param-name> に設定するパラメタ名※1	<param-value> に設定するパラメタ値※1	設定内容
hptl.portlet.defaulticon	*hptl.portlet.defaulticon*	ポートレットのデフォルトアイコンの URL を指定します。ポートレットのエントリのディレクトリからの相対パスで指定します。アイコンの画像サイズは 16 × 16 ピクセルとします。
hptl.portlet.directreq.support	*hptl.portlet.directreq.support*	uCosminexus Portal Framework にログインしていない状態から，ポートレットのダイレクト呼び出し（Web ブラウザでポートレットの URL を直接指定して表示させる機能）ができるかどうかを指定します。 • true：ダイレクト呼び出しができます。ログイン後に指定したポートレットが表示されます。 • false：ダイレクト呼び出しできません。ログイン後にホーム画面が表示されます（デフォルト値）。
hptl.portlets.layout.hide	*hptl.portlets.layout.hide*	利用者用レイアウトカスタマイズ画面の［ポートレット選択画面］のポートレット一覧から選択できないポートレットに設定するかどうか（レイアウトできないポートレットにするかどうか）を指定します。 • true：レイアウトできないポートレットに設定します。 • false：レイアウトできるポートレットに設定します（デフォルト値）。
hptl.portlets.layout.not.modified	*hptl.portlets.layout.not.modified*	拡張レイアウト形式の変更不可エリアでだけ選択できるポートレットに設定するかどうかを指定します。運用管理者または部門管理者が変更不可エリアでポートレットを追加・変更するときにだけ管理者用レイアウトカスタマイズ画面の［ポートレット選択画面］のポートレット一覧に表示されます。 • true：変更不可エリアにだけ追加・変更できるポートレットに設定します。 • false：すべてのエリア，およびすべてのレイアウト形式でポートレットの追加・変更ができるポートレットに設定します（デフォルト値）。
hptl.portlet.parameters	*hptl.portlet.parameters*	他のポートレットに設定情報などのパラメタを引き渡すためのインタフェースを提供するかどうかを定義します。インタフェースを提供する場合は，このインタフェースの実装クラスの FQCN（パッケージ名を含むクラス名）を指定します。

（凡例）

－：該当しません。

注※1　パラメタ名およびパラメタ値に「'」，「"」，「|」を含む文字列は入力できません。また，パラメタ名およびパラメタ値は，大文字小文字を区別します。

注※2　ポートレットのタイトルに各ボタンを表示させるには，Portal Manager の各ポートレットの設定で指定する必要があります。

(3) デプロイ定義ファイル作成時の注意事項

- タグ名および属性名はすべて小文字で記載します。
- 空要素タグ <minimize />，<close />，または <maximize /> を記述した場合にポートレットで「最小化ボタン」,「閉じるボタン」または「最大化ボタン」を使用するためには，ポートレットテンプレートでポートレットテンプレート Bean を利用する必要があります。ポートレットテンプレートについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットテンプレート」の説明を，ポートレットテンプレート Bean については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットテンプレート Bean」の説明を参照してください。

3.13.3 Web アプリケーションの DD（web.xml）の作成

サーブレットのマッピング定義などに使用する Web アプリケーションの DD（web.xml）を作成し，{PORTLET_HOME}¥PORTLET-INF に格納します。このファイルは，{PROJECT_HOME}¥WEB-INF に格納されている web.xml に対して，デプロイするポートレットで使用するサーブレットの定義やタグライブラリの定義を追加する場合にだけ新規に作成する必要があります。

(1) {PROJECT_HOME}¥WEB-INF に格納されている web.xml へのタグの追記

作成した web.xml ファイルに記述した定義内容は，ポートレットデプロイ時に {PROJECT_HOME}¥WEB-INF に格納されている web.xml に追記されます。ただし，作成した web.xml ファイルに記述したタグと同じ名前のタグが，{PROJECT_HOME}¥WEB-INF に格納されている web.xml にすでに存在する場合は，作成した web.xml ファイルに記述したタグで上書きされます。追記されるタグおよび同じ名前のタグと判断する条件を次の表に示します。

表 3-13　追記するタグおよび同じ名前のタグと判断する条件

タグ	同じ名前のタグと判断する条件
<filter>	filter-name タグの値が同じ値の場合
<filter-mapping>	url-pattern タグまたは servlet-name タグの値が同じ値の場合
<servlet>	servlet-name タグの値が同じ値の場合
<servlet-mapping>	url-pattern タグの値が同じ値の場合
<taglib>	taglib-uri タグの値が同じ値の場合
<resource-ref>	res-ref-name タグの値が同じ値の場合
<ejb-ref>	ejb-ref-name タグの値が同じ値の場合

(2) web.xml ファイル作成時の注意事項

- uCosminexus Portal Framework で使用できる Web アプリケーションの DD は Servlet 2.2 または Servlet 2.3 の DD です。
- taglib タグ内の taglib-location タグで指定したパスに TLD ファイルが存在する場合，ポートレットデプロイ時に {PROJECT_HOME}¥WEB-INF¥cosmi¥portlets¥< ポートレット名 > ディレクトリに，指定したファイルがコピーされます。また，{PROJECT_HOME}¥WEB-INF に格納されている web.xml にはコピー後のパスで taglib-location タグが追記されます。

3.14 ダイレクト呼び出し対応ポートレットの開発

ここでは，ダイレクト呼び出しに対応したポートレットを開発する場合のデプロイ定義ファイルの設定，およびダイレクト呼び出し時の POST データについて説明します。

3.14.1 ダイレクト呼び出しとは

ダイレクト呼び出しとは，uCosminexus Portal Framework にログインしていない状態から，Web ブラウザでポートレットの URL を直接指定し，uCosminexus Portal Framework を経由してポートレットの画面を表示させる機能のことです。uCosminexus Portal Framework を経由してのポートレットの画面呼び出しについては，「2.1.2　ドキュメントベース」を参照してください。

なお，ダイレクト呼び出しをした場合，いったん uCosminexus Portal Framework の［ログイン］画面が表示され，ログインするとポートレットの画面が表示されます。

ただし，ウェルカム画面に呼び出したポートレットのアクセス権がある場合は対象外となります。

3.14.2 デプロイ定義ファイルの設定

uCosminexus Portal Framework は，ポートレットをデプロイするときの設定値を参照して，ダイレクト呼び出しに対応しているかどうかを判定します。ここでは，ダイレクト呼び出し対応ポートレットのデプロイ定義ファイルに記述する項目について説明します。

ダイレクト呼び出し対応ポートレットのデプロイ定義ファイル（ファイル名は hportlet.xml）に記述する項目を，次の図に示します。

図 3-16　ダイレクト呼出対応ポートレットのデプロイ定義ファイルに記述する項目

<portlet-app>			このファイルのルートタグです。
	<portlet>		ポートレット情報の親タグです。
	< 中略 > ※		< 中略 > ※
		<config-param>	ポートレット起動時のパラメタを設定します。 この設定は複数指定できます。
		<param-name> hptl.portlet.directreq.support </param-name>	ポートレットの対応状況を示すキーです。 この項目の設定は必須です。

		<param-value> 　true </param-value>	true：ダイレクト呼び出し対応 false：ダイレクト呼び出し非対応 省略したときは，false が設定されます。
		< 中略 > ※	< 中略 > ※

注※　タグの詳細は，「3.13.2　デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。

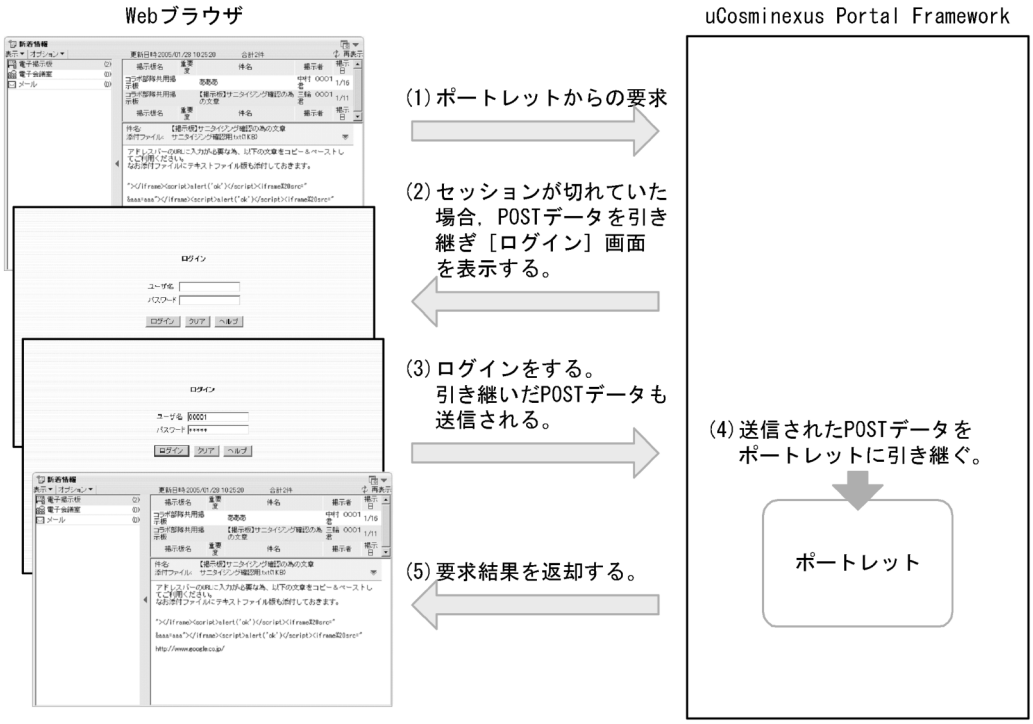
3.14.3　ダイレクト呼び出し時の POST データ

セッションタイムアウトやその他の原因によってセッションが切断されている状態で，ダイレクト呼び出しをしたときの POST データは，次の場合を除いてポートレットに送信されます。

- ファイルアップロードが実行されていた場合
- データサイズが上限を超えていた場合

ダイレクト呼び出し時の画面遷移と処理の流れを次の図に示します。

図 3-17　ダイレクト呼び出し時の画面遷移と処理の流れ



(1) ポートレットへの POST データの引き渡し

ダイレクト呼び出しをした場合に，ポートレットに渡される HttpServletRequest オブジェクトからデータを取得できます。HttpServletRequest の各メソッドで取得できるデータを次の表に示します。

表 3-14　HttpServletRequest の各メソッドで取得できるデータ

項番	メソッド名	取得データ
1	getAuthType()	(3)
2	getContextPath()	(3)
3	getCookies()	(3)
4	getDateHeader(java.lang.String name)	(3)
5	getHeader(java.lang.String name)	(3)
6	getHeaderNames()	(3)
7	getHeaders(java.lang.String name)	(3)
8	getIntHeader(java.lang.String name)	(3)
9	getMethod()	(3) ※1
10	getPathInfo()	(3) ※2
11	getPathTranslated()	(3)
12	getQueryString()	(1)
13	getRemoteUser()	(3)
14	getRequestedSessionId()	(3)
15	getRequestURI()	(3)
16	getRequestURL()	(3)
17	getServletPath()	(3)
18	getSession()	(3)
19	getSession(boolean create)	(3)
20	getUserPrincipal()	(3)
21	isRequestedSessionIdFromCookie()	(3)
22	isRequestedSessionIdFromUrl()	(3)
23	isRequestedSessionIdFromURL()	(3)
24	isRequestedSessionIdValid()	(3)
25	isUserInRole(java.lang.String role)	(3)

（凡例）

括弧付き数字は図 3-17 の括弧付き数字と対応しています。

(1)：(1) で送信したリクエスト

(3)：(3) で送信したリクエスト

注※1　"POST" となります。
注※2　使用できません。

表 3-15　ServletRequest から継承した各メソッドで取得できるデータ

項番	メソッド名	取得データ
1	getAttribute(java.lang.String name)	(3)
2	getAttributeNames()	(3)
3	getCharacterEncoding()	(3)
4	getContentLength()	(1)
5	getContentType()	(1)
6	getInputStream()	(1)
7	getLocale()	(3)
8	getLocales()	(3)
9	getParameter(java.lang.String name)	(1)
10	getParameterMap()	(1)
11	getParameterNames()	(1)
12	getParameterValues(java.lang.String name)	(1)
13	getProtocol()	(3)
14	getReader()	(1)
15	getRealPath(java.lang.String path)	(3)
16	getRemoteAddr()	(3)
17	getRemoteHost()	(3)
18	getRequestDispatcher(java.lang.String path)	(3)
19	getScheme()	(3)
20	getServerName()	(3)
21	getServerPort()	(3)
22	isSecure()	(3)
23	removeAttribute(java.lang.String name)	(3)
24	setAttribute(java.lang.String name, java.lang.Object o)	(3)
25	setCharacterEncoding(java.lang.String env)	(3)

（凡例）
括弧付き数字は図 3-17 の括弧付き数字と対応しています。
(1)：(1) で送信したリクエスト
(3)：(3) で送信したリクエスト

(a) 上限サイズ

送受信対象データが，PortalResources.properties ファイルの

jp.co.hitachi.soft.portal.directaccess.postdata.size 項目で指定されている POST データの上限サイズを超えた場合，POST データを引き継ぎません。
PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」を参照してください。

(b) 引き継ぎ範囲

POST データの引き継ぎ範囲（日立 API ポートレット内のすべてとなります）を次に示します。

- JSP 内（各画面モード）
- アクションモジュール内（各画面モード）
- ログイン画面

(c) 注意事項

- HttpServletRequest の各メソッドで取得できるデータ以外は引き継ぎの対象外となります。
- 引き継ぎ対象データにファイルアップロードデータが含まれる場合，セキュリティの観点から POST データを引き継ぎません。

(d) 引き継ぎ結果の取得

POST データの引き継ぎ結果は，ダイレクト呼び出し結果取得 API で取得できます。ダイレクト呼び出し結果取得 API の詳細は，「14.14　ダイレクト呼び出し結果取得 API」を参照してください。

3.15　標準 API ポートレットの作成手順

この節では，標準 API ポートレットを作成する手順について説明します。

手順

1. **標準 API ポートレットのコンテンツを開発します。**
   コンテンツを開発する際には，ターゲットとするデバイス（PC，ｉモード，または EZweb）に対応する HTML/CHTML/HDML のバージョンで記述する必要があります。uCosminexus Portal Framework が対応している各言語のバージョンについては，「2.1.1　ターゲットとするデバイス」を参照してください。
   また，コンテンツを開発する際には，次の前提知識，および注意事項を確認してください。
   前提知識
   - 「3.17　標準 API ポートレットの画面モード」
   - 「3.18　標準 API ポートレットの動作」
   - 「3.19　リクエストコントローラ機能」
   - 「3.20　ポートレットタイトルの変更」
   - 「3.21　コンテンツキャッシュ制御」
   - 「3.22　カスタムウィンドウステート」
   - 「3.23　Struts フレームワークの使用」

   注意事項
   - 「3.16　HTML 記述上の注意（標準 API ポートレット）」

   なお，標準 API ポートレットからは J2EE サービスを利用できません。

2. **標準 API ポートレットを開発します**
   ターゲットとするデバイスにポートレットを表示するための設定を行います。
   この設定の詳細は，「3.24　デバイスに対応するための設定（標準 API ポートレット）」を参照してください。

3. **WAR ファイルを作成します。**
   WAR ファイルの作成方法については，「3.25　標準 API ポートレットのアーカイブの作成」を参照してください。

4. **標準 API ポートレットをポータルに登録します。**
   ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

3.16 HTML 記述上の注意（標準 API ポートレット）

3.16.1 予約語

標準 API ポートレットを uCosminexus Portal Framework で表示するとき，次の文字列を予約語としています。標準 API ポートレットを開発するとき，これらの予約語で始まる任意の文字列を指定しないでください。

- java.
- javax.
- javax.portlet.
- hptl
- com.cosminexus
- jp.co.hitachi.soft.portal

予約語に指定されている文字列を使用できないファイルや関数名などについて次に示します。

- ストリングリソース
- セッションにバインドするオブジェクトを指定するときの名称
- request オブジェクトのアトリビュート名

ポータルを経由してポートレットの画面を呼び出す場合，ポートレットではクエリストリング（"?query"）を自由に設計できます。ただし，次の文字列は予約語のため使用できません。

クエリストリングの予約語

action，screen，portlet，pane，mode，url，layout_id，tab_id，jsessionid，hptl_referer，tgt_id，typ_id，p_mode，state，rp_，hptl，および javax.portlet で始まる文字列

ポータルを経由してのポートレットの画面呼び出しについては，「2.1.2 ドキュメントベース」を参照してください。

3.16.2 スクリプトを使用するには

Web ブラウザによって対応しているスクリプト言語が異なるため，ターゲットとする Web ブラウザを決定してから使用するスクリプト言語を決定します。対応する Web ブラウザが多いため，uCosminexus Portal Framework では，スクリプト言語として JavaScript を推奨します。

ポートレットでスクリプトを使用する場合，通常の記述方法とは次の点が異なります。

- ポートレットの URL を作成する場合，URL 変換を行う必要があります。詳細は Java Portlet Specification 1.0 の仕様を参照してください。
- JavaScript および JScript では，ローカル変数宣言時に "var" 宣言子を使用します。
- スクリプトのオブジェクト名，関数名，およびグローバル変数名は異なる名称を指定します。
  関数名およびグローバル変数名には，標準 API のタグライブラリを用いてプレフィックスを付けてください。ただし，Web ブラウザによっては名称の長さに制限があります。ターゲットとする Web ブラウザで正しく表示されない場合は，単語を省略するなどして，名称を短くしてください。

3.16.3 標準 API ポートレットで使用できないタグ

標準 API ポートレットでは，次のタグは使用できません。詳細は Java Portlet Specification 1.0 の仕様を参照してください。

- HTML 開始，終了タグ※ 1
- HEAD 要素※ 1
- BASE 要素※ 1
- TITLE 要素※ 1
- BODY 開始，終了タグ※ 1
- IFRAME 要素※ 1 ※ 2
- FRAME 要素※ 1
- FRAMESET 要素※ 1

注※ 1　IFRAME ステートで動作する場合は記述することができます。

注※ 2　IFRAME ステートをサポートする場合は記述することができます。

3.17 標準 API ポートレットの画面モード

uCosminexus Portal Framework では，標準 API ポートレットを表示するために次の画面モードをサポートしています。

- ポートレットモード
- ウィンドウステート

ポートレットモードは，次の 3 種類があります。

- VIEW モード
  ポートレットを閲覧するモードです。
- EDIT モード
  ポートレットを編集するモードです。
- HELP モード
  ヘルプモードです。

ウィンドウステートは，次の 5 種類があります。

- NORMAL ステート
  ポータル画面に通常表示される状態です。
- MAXIMIZE ステート
  ポートレットを最大化表示した状態です。
- MINIMIZED ステート
  ポートレットを最小化表示した状態です。
- NEWWINDOW ステート
  ポートレットを新規のウィンドウに表示した状態です。
- IFRAME ステート
  Web コンテンツをインラインフレーム内に表示した状態です。

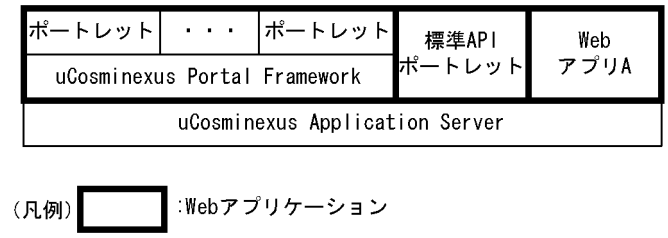
画面モードおよびウィンドウステートの詳細については，Java Portlet Specification 1.0 の仕様を参照してください。なお，NEWWINDOW ステートおよび IFRAME ステートはカスタムウィンドウステートに対応しています。詳細は，「3.22　カスタムウィンドウステート」を参照してください。

3.18 標準 API ポートレットの動作

uCosminexus Portal Framework は，Application Server 上で動く Web アプリケーションであり，JSP/ サーブレットで構築されています。uCosminexus Portal Framework は，複数のポートレットで構成されます。標準 API ポートレットは uCosminexus Portal Framework と同列の Web アプリケーションとして動作し，Servlet API のリクエストディスパッチで呼び出され，uCosminexus Portal Framework の延長として動作します。

uCosminexus Portal Framework と標準 API ポートレットの関係を次の図に示します。

図 3-18 uCosminexus Portal Framework と標準 API ポートレットの関係



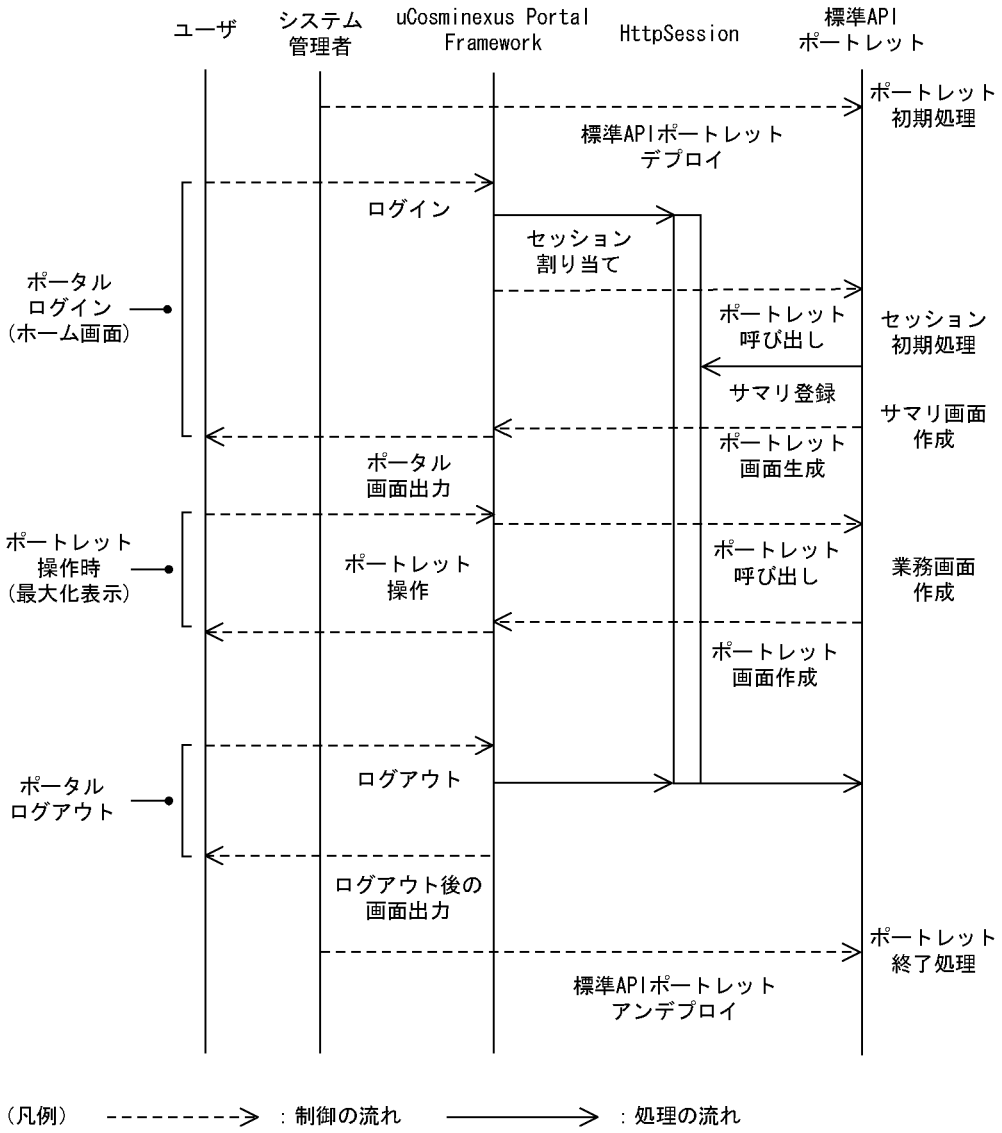
uCosminexus Application Server 上では，標準 API ポートレットは uCosminexus Portal Framework とは別のアプリケーションとして動作します。また，uCosminexus Portal Framework や標準ポートレット以外にも，JSP やサーブレットで構築された Web アプリケーションが動作します。そのため，ポートレットは，uCosminexus Portal Framework およびほかのポートレットと，Web アプリケーションのコンテキストを共有します。

3.18.1 標準 API ポートレットのライフサイクル

uCosminexus Portal Framework では，Java Portlet Specification 1.0 に従いポートレットを自由に作成することができます。

標準 API ポートレットのライフサイクルを次の図に示します。

図 3-19 標準 API ポートレットのライフサイクル



(1) 標準 API ポートレットの初期処理および終了処理

ポートレットの初期処理は，システム管理者によって uCosminexus Component Container に対しポートレットのデプロイが行われると実行されます。※

ポートレット開発者は javax.portlet.GenericPortlet.GenericPortlet#init をオーバーライドし，ポートレットで共通設定ファイルの読み込み処理などを行います。

ポートレットの終了処理は，運用管理者によってポートレットのアンデプロイが行われ

ると実行されます。ポートレット開発者は，javax.portlet.GenericPortlet.GenericPortlet#destroy をオーバーライドし，初期処理時に確保したリソースを解放します。

注※　Web アプリケーション DD の「load-on-startup」が 0 に設定されている場合は，初回アクセス時に実行されます。

(2) アクション処理およびレンダー処理

標準 API ポートレットでは，クライアントからの要求をアクション要求およびレンダー要求に分けて処理を行います。詳細は，Java Portlet Specification 1.0 の仕様を参照してください。

(3) セッションの初期処理および終了処理

セッションの初期処理および終了処理では，ユーザに固有のパーソナライズデータを取得したり，保存したりします。セッションの初期処理および終了処理は，セッションに対するイベントリスナ（HttpSessionBindingListener）を使用して実現します。エントリポイントとして登録した JSP/ サーブレットのサービスメソッド内で，開始・終了処理を記述したイベントリスナを HttpSession に登録します。

3.18.2 URL 変換

標準 API ポートレット内で，標準 API のタグライブラリを使用した相対パス形式の URL は，次のどれかの形式に変換されます。

- ドキュメントベースを基とした絶対 URL 形式
- {PROJECT_HOME} を基点とした絶対パス形式
- またはプロトコル指定を基点とした絶対 URL 形式

これを **URL 変換**と呼びます。標準 API を使用すると，現在動作している uCosminexus Portal Framework の環境を基に動的に URL 形式が変換されます。

URL 変換は，クエリ（"?query"）に対応します。部分識別子（"#fragment"），および ";" パラメタには対応していません。絶対 URL 形式，絶対パス形式，および相対パス形式の例を次に示します。

- 絶対 URL 形式：http://server/portal/sample/index.jsp
- 絶対パス形式：/portlets/sample/index.jsp
- 相対パス形式：index.jsp

なお，SSL アクセラレーターまたはリバースプロキシを使用した環境で uCosminexus Portal Framework を使用する場合には，URL 変換規則を切り替える必要があります。また，標準 API を利用して動的にセキュア属性を変更する場合にも URL 変換規則，および HTTP（S）ドメイン名を設定する必要があります。詳細は，マニュアル

「uCosminexus Portal Framework　システム管理者ガイド」の「SSL アクセラレーターまたはリバースプロキシ使用時の設定」の説明を参照してください。

3.19 リクエストコントローラ機能

リクエストコントローラ機能は，リクエストを解析して，表示するポートレットを決定し，HTTP レスポンスを構築する機能です。

リクエストコントローラ機能は，次の機能から構成されています。

- キャッシュの初期化
- セキュア通信判定
- コンテンツタイプ判定
- ユーザ情報取得
- processAction() 実行
- リダイレクト判定
- ポートレットウィンドウの情報更新

各機能について説明します。

3.19.1 キャッシュの初期化

標準 API ポートレットでは，リクエストを解析して実行情報を作成します。このとき，次のどちらかの情報があると，セッション上にあるすべてのポートレットのキャッシュ情報を初期化します。

- サブレイアウト（タブ形式で画面を表示しているときのコンテンツの表示領域）が変更されたとき
- URL にコンテンツキャッシュ削除が指定されているとき

キャッシュ初期化の対象になる情報を次の表に示します。

表 3-16 キャッシュ情報の種類

キャッシュ情報の種類	説明	サブレイアウト変更時の初期化処理※1	コンテンツキャッシュ削除指定時の初期化処理
コンテンツキャッシュ	ポートレットフラグメント領域のコンテンツ内容	○	○
レンダーパラメタ	render() 実行時に使うパラメタ	○	×
ポートレットモード	表示対象となるポートレットモード	○	×
ポートレット最小化フラグ※2	MINIMIZE ステートで表示するかどうかを示すフラグ	×	×

（凡例）

○：初期化します。　×：初期化しません。
注※1　キーボードから［Ctrl］+［N］で起動したウィンドウ上でサブレイアウトを変更した場合，同じセッションのすべてのブラウザで，操作中のポートレット表示が初期化されます。
注※2　ポートレット最小化フラグは，サブレイアウトごとにキャッシュを保持するため，サブレイアウトを切り替えたあとでも状態を保持します。セッションを破棄するまで有効です。

注意事項

標準 API ポートレットでは，上記のキャッシュ情報をセッションタイムアウト時に保持できません。

3.19.2　セキュア通信判定

標準 API ポートレットでは，ポートレットアプリケーション DD（portlet.xml）に記載された <transport-guarantee> タグの設定内容と通信プロトコルによってセキュア通信の判定をします。

標準 API ポートレットを uCosminexus Portal Framework 上で表示するとき，<transport-guarantee> タグの設定内容が「INTEGRAL」または「CONFIDENTIAL」の場合，HTTP プロトコルによる processAction() 実行をしないで，エラー画面を表示します。

ただし，SSL アクセラレーター使用時にリバースプロキシ使用プロパティを「SSL」に設定した場合，ポートレットアプリケーション DD の内容に関係なく processAction() 実行とコンテンツ表示を行います。

セキュア通信判定による動作を次の表に示します。

表 3-17　セキュア通信判定による標準 API ポートレットの動作

条件		動作	
通信プロトコル	<transport-guarantee> タグの設定	processAction 実行	コンテンツ表示
HTTP	記述なし	○	○
	NONE	○	○
	INTEGRAL	SSL アクセラレーターが無効の場合：実行しない SSL アクセラレーターが有効な場合：○	SSL アクセラレーターが無効の場合：エラー画面を表示する※ SSL アクセラレーターが有効な場合：○
	CONFIDENTIAL	SSL アクセラレーターが無効の場合：× SSL アクセラレーターが有効な場合：○	SSL アクセラレーターが無効の場合：× SSL アクセラレーターが有効な場合：○

条件		動作	
通信プロトコル	<transport-guarantee> タグの設定	processAction 実行	コンテンツ表示
HTTPS	記述なし	○	○
	NONE	○	○
	INTEGRAL	○	○
	CONFIDENTIAL	○	○

（凡例）
○：有効
注※　Web コンテナの機能では，302 レスポンスコードによる HTTPS へのリダイレクトを行いますが，uCosminexus Portal Framework ではエラー画面を表示します。HTTPS へリダイレクトしたい場合は，Web コンテナの機能を使用してください。

3.19.3 コンテンツタイプ判定

標準 API ポートレットは，コンテンツタイプを判定して，ポートレットコンテンツを表示します。コンテンツタイプは，次の項目から判定します。

- ポートレットアプリケーション DD（portlet.xml）に記載された <supports> タグの設定内容
- HTTP リクエストヘッダの User-Agent パラメタのデバイス種別

HTTP リクエストヘッダの User-Agent パラメタには，対応する応答コンテンツタイプが定義されています。uCosminexus Portal Framework は，HTTP レスポンスヘッダにこの応答コンテンツタイプを設定します。

<supports> タグに応答コンテンツタイプと一致するコンテンツタイプがない場合は，エラー画面を表示します。

注意事項

uCosminexus Portal Framework が扱う応答コンテンツタイプはリクエストごとに一つだけです。またコンテンツタイプの変換は行いません。そのため，uCosminexus Portal Framework は PortletRequest#getResponseContentType() メソッドの返却値として必ず一つのコンテンツタイプ（応答コンテンツタイプ）しか返却しません。
また，ターゲットとするデバイスに HDML を使用する場合は，ポートレットアプリケーション DD の <supports> タグには text/x-hdml を指定してください。

3.19.4 ユーザ情報取得

標準 API ポートレットでは，ポートレットアプリケーション DD（portlet.xml）に設定した <user-attribute> タグの内容（ユーザ属性名）からユーザ情報を取得します。

ポートレットアプリケーション DD の <user-attribute> タグのユーザ属性名と，uCosminexus Portal Framework のユーザ管理機能のユーザ定義項目名をマッピングしておくことで，リポジトリからユーザ情報を取得できます。マッピングの手順については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」を参照してください。

取得したユーザ情報と <user-attribute> タグのユーザ属性名を，マップに設定します。

ユーザがログインしていない場合は，null 値が返されます。また，次の場合は，空のマップを設定します。

- ポートレットアプリケーション DD にユーザ属性名を指定していない。
- リポジトリに，使用可能なユーザ情報が存在しない。

注意事項

ユーザ属性名にマルチバリュー項目が指定された場合は，次の値をユーザ属性として設定します。

ディレクトリサーバの場合
　取得した最初の値

DB の場合
　先頭のカラムの値

3.19.5 processAction() 実行

URL にアクション対象のポートレットが指定されている場合，URL とポートレットウィンドウ情報の内容を用いて，processAction() メソッドを実行します。

URL と processAction() 実行との関係を次の表に示します。

表 3-18 URL と processAction() 実行の関係

URL のターゲット ID			ポートレットの動作
コンテキストパス	画面配置	画面表示	
使用不可	配置あり	表示	processAction() を実行しないで，ポートレットウィンドウ情報にエラーメッセージ ID を設定する。
		非表示	
	配置なし	表示	
		非表示	
使用可能	配置あり	表示	レンダーパラメタのキャッシュを削除して，processAction() を実行する。
		非表示	
	配置なし	表示	
		非表示	

次の場合は，processAction() メソッドを実行しません。

- ポートレットウィンドウ情報にエラーメッセージ ID が指定されている。
- 指定されたポートレット名のコンテキストパスが使用できない。
- アクション対象のポートレットの利用権限がない。
- リクエスト種別がアクション要求以外である。
- ターゲット ID が指定されていない。

URL で指定したポートレットに利用権限がある場合は，ポートレットの配置の有無，および表示の有無に関係なく processAction() メソッドを実行します。

なお，次の場合はエラー画面を表示します。

- コンテキストパスが使用できない。
- processAction() メソッドが実装されていない。
- processAction() メソッドの実行中に例外が発生した。

3.19.6 リダイレクト判定

標準 API ポートレットでは，processAction() の実行中に ActionResponse#sendRedirect() が呼び出されると，リダイレクトが実行されて，ブラウザの画面全体が指定したリダイレクト先に切り替わります。

processAction() 実行中に例外が発生した場合は，リダイレクトを実行しません。

注意事項

リダイレクト元とリダイレクト先が同じポートレットである場合，無限ループが発生しますので，注意してください。

3.19.7 ポートレットウィンドウの情報更新

processAction()，または URL でポートレットモード，ウィンドウステート，またはレンダーパラメタが変更されたときに，ポートレットウィンドウの情報に変更内容を反映します。ここでは，変更内容を反映するときの動作を説明します。

(1) ポートレットモード変更

ポートレットモードは，ポートレットアプリケーション DD（portlet.xml）の <supports> タグに記載されているポートレットモードに変更できます。<supports> タグに記載されていないポートレットモードに変更すると，エラーになります。

URL でポートレットモードを指定する場合の，ポートレットモード変更時の動作を次の表に示します。

表 3-19 ポートレットモード変更時の動作

URL				動作
ポートレットモード	ターゲット ID			
	サポート	画面配置	画面表示	
VIEW/ EDIT/ HELP	記述あり	配置あり	表示	モードを変更します。
			非表示	
		配置なし	表示	
			非表示	
	記述なし	配置あり	表示	エラーを表示します。
			非表示	
		配置なし	表示	
			非表示	

URL が，表示対象（MINIMIZE モード以外）のポートレットに対して許可されていないポートレットモードを指定した場合は，ポートレットフラグメント領域にエラーテンプレートで定義されたエラー画面を表示します。

(2) ウィンドウステート変更

ウィンドウステートを MINIMIZE に設定した場合は，render() を実行しません。URL で指定したポートレットが画面配置されている場合だけ，ウィンドウステートを変更します。

URL でウィンドウステートを指定する場合の，ウィンドウステート変更時の動作を次の表に示します。

表 3-20 ウィンドウステート変更時の動作

URL			動作
ウィンドウステート	ターゲット ID		
	画面配置	画面表示	
NORMAL/ MINIMIZE	指定なし		レイアウト領域にエラー画面を表示します。
	配置あり	表示	ウィンドウステートを変更して，画面表示します。
		非表示	
	配置なし	表示	ウィンドウステートを変更しないで，前回レイアウトの初期画面を表示します
		非表示	
MAXIMIZE	指定なし		レイアウト領域にエラー画面を表示します。
	配置あり	表示	ウィンドウステートを変更して，画面表示します。
		非表示	
	配置なし	表示	
		非表示	

（凡例）

－：該当しません。

なお，不正なウィンドウステートが指定された場合は，ポートレットフラグメント領域にエラーテンプレートで定義されたエラー画面を表示します。

(3) レンダーパラメタ変更

URL でレンダーパラメタを指定する場合の変更時の動作を次の表に示します。

表 3-21 レンダーパラメタ変更時の動作

URL			動作
レンダーパラメタ	ターゲット ID		
	画面配置	画面表示	
指定あり	配置あり	表示	レンダーパラメタのキャッシュを削除して，レンダーパラメタを設定します。
		非表示	
	配置なし	表示	
		非表示	
指定なし	－		レンダーパラメタを設定しません。

（凡例）

－：該当しません。

3.20 ポートレットタイトルの変更

ポートレットのコンテンツ生成時に RenderResponse#setTitle() を用いて，タイトルバーのポートレットタイトルを変更できます。タイトルを変更するときに使う文字列エンコーディングは標準 API ポートレットが判断し，uCosminexus Portal Framework は設定された値をそのまま出力します。

ポートレットタイトルは，ポートレットアプリケーション DD（portlet.xml）の <title> タグでも設定できます。ポートレットアプリケーション DD の <title> タグと RenderResponse の setTitle() の両方に定義されている場合は，RenderResponse の設定値が有効となります。

setTitle() の注意事項を次に示します。

- setTitle() が実行された場合のデータは，１リクエスト内で有効です。ただし，コンテンツデータがキャッシュされる場合は，コンテンツデータと合わせてキャッシュ登録します。
- setTitle() は複数回使用できます。その場合は最後に設定した値が有効になります。
- setTitle() に「null」を設定した場合は，setTitle() の操作を無視し，ポートレットアプリケーション DD の設定を有効にします。

3.21 コンテンツキャッシュ制御

ポートレットアプリケーション DD（portlet.xml）に記述されたキャッシュの有効期間情報を基に，標準 API ポートレットの情報をセッションに登録してキャッシュ制御できます。

3.21.1 キャッシュの登録

標準 API ポートレットの情報は，キャッシュ領域にデータがない場合に登録されます。情報を登録する単位は 1 クライアント当たりのポートレットごとです。キャッシュに登録される内容を，次の表に示します。

表 3-22 キャッシュに登録されるコンテンツ情報

項番	コンテンツ情報	内容
1	コンテンツデータ	標準 API ポートレットが出力したコンテンツデータ。
2	有効期間	コンテンツを生成してから破棄の対象になるまでの有効期間。
3	ポートレットタイトル	変更されたタイトル情報。 標準 API ポートレットのコンテンツデータ生成時に，ポートレットタイトルを変更された場合だけ保存します。

3.21.2 キャッシュの破棄

キャッシュに登録されたコンテンツ情報は次のときに破棄されます。

- キャッシュの有効期間が過ぎたとき
  キャッシュ登録時に設定した有効期間が過ぎたときに，キャッシュからコンテンツ情報が破棄されます。キャッシュの有効期間は，表示対象のポートレットだけ確認されます。
- ポートレットがアクション処理対象であるとき
  アクション処理対象のポートレットでは，アクション処理結果を反映させるため，キャッシュの有効期間内であってもキャッシュの情報が破棄されます。ポートレットアクションの処理が失敗した場合もキャッシュ情報は破棄されます。
- ユーザがログアウトしたとき
  ログアウト時にセッション情報が破棄されると，キャッシュの有効期間内であってもすべてのキャッシュ情報が破棄されます。
- サブレイアウトが変更されたとき
  表示対象サブレイアウトが変更されると，古いレイアウト内のポートレット情報は有効期限に関係なくすべて破棄されます。この処理は，リクエストコントローラ機能で行われます。同一レイアウト内（ポートレットの最大表示時や，通常表示に戻ったときなど）の画面遷移，およびレイアウト編集画面，またはポートレットのデプロイ・

アンデプロイによる画面遷移の場合は，キャッシュは破棄されません。
リクエストコントローラ機能のキャッシュの初期化については，「3.19.1　キャッシュの初期化」を参照してください。

3.21.3　キャッシュの有効期間

キャッシュの有効期間は，次のどちらかに設定します。

- RenderResponse の portlet.expiration-cache プロパティ
- ポートレットアプリケーション DD（portlet.xml）に定義する <expiration-cache> タグ

RenderResponse のプロパティ項目とポートレットアプリケーション DD ファイルの両方に有効期間が定義されている場合は，RenderResponse の設定値が有効となります。

キャッシュの有効期間に設定できる値と値に対する動作を，次の表に示します。

表 3-23　キャッシュの有効期間の設定値

有効期間の設定値（秒）	キャッシュ機能の動作
-1	コンテンツ情報を無期限に保持します。ただし，次の場合は，キャッシュから破棄します。 • action メソッドが動いた場合 • 表示対象レイアウトが変わった場合 • ログアウトした場合
0 または省略	コンテンツ情報はキャッシュ登録しません。リクエストごとにコンテンツを生成します。
1 以上	設定された有効期間内であれば，キャッシュからコンテンツ情報を返却します。設定した有効期間を過ぎたときはコンテンツ生成処理を行います。

3.22 カスタムウィンドウステート

標準 API ポートレットでは，カスタムウィンドウステートとして以下のステートを使用することができます。カスタムウィンドウステートを使用する場合は，uCosminexus Portal Framework 以外のポートレットコンテナで動作しなくなりますので注意してください。

- NEWWINDOW ステート
- IFRAME ステート

この節では，カスタムウィンドウステートを使用する場合の設定を説明します。

3.22.1 カスタムウィンドウステートの設定

標準 API ポートレットでカスタムウィンドウステートを使用する場合にポートレットアプリケーション DD および PortletApp.properties の設定が必要です。

(1) ポートレットアプリケーション DD

ポートレットアプリケーション DD に <custom-window-state> の定義を次のとおりに追加します。

- NEWWINDOW ステートを使用する場合

```
<custom-window-state>
    <description>uCosminexus Portal Framework presents New window
state.</description>
     <window-state>newwindow</window-state>
</custom-window-state>
```

- IFRAME ステートを使用する場合

```
<custom-window-state>
     <description>uCosminexus Portal Framework presents IFrame
state.</description>
     <window-state>iframe</window-state>
</custom-window-state>
```

(2) PortletApp.properties ファイル

ポータルコンテキストの設定でサポートしているウィンドウ状態の設定を追加します。PortletApp.properties ファイルの設定内容についてはマニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortletApp.properties の詳細」を参照してください。

3.22.2 カスタムウィンドウステートの使用

標準 API ポートレットをカスタムウィンドウステートで実行するためには，ウィンドウステートの指定にカスタムウィンドウステートを指定し生成した URL を実行します。カスタムウィンドウステートを指定するため ,WindowState クラスのサブクラスである UCPFWindowState クラスを使用することができます。UCPFWindowState クラスの詳細は「14.20　カスタムウィンドウステート API」を参照してください。

3.22.3 カスタムウィンドウステート使用時の注意事項

カスタムウィンドウステートを使用する場合の注意事項を以下に説明します。

(1) カスタムウィンドウステート実行時のポートレットモード及びウィンドウステート

標準 API ポートレットをカスタムウィンドウステートで動作させる場合，ポートレットモードは VIEW モードになります。また，カスタムウィンドウステートで動作している場合，ポートレットモードおよびウィンドウステートを変更することはできません。変更しようとした場合は例外がスローされます。

(2) NEWWINDOW ステート

NEWWINDOW ステートは新規ウィンドウ状態で動作します。このため，
NEWWINDOW ステートを指定して生成した URL を実行する場合は，ポータル画面とは別のウィンドウに表示してください。

(3) IFRAME ステート

IFRAME ステートはインラインフレーム内に表示した状態で動作します。このため，
IFRAME ステートを指定して生成した URL を実行する場合は，インラインフレーム内に表示してください。

3.23 Struts フレームワークの使用（標準 API ポートレット）

標準 API ポートレットでは Struts フレームワークを使用したポートレットを作成することができます。サポートしている Struts フレームワークのバージョンは 1.2.9 です。この節では，Struts フレームワークを使用したポートレットを作成する方法について説明します。

3.23.1 Struts アプリケーションの開発

Struts フレームワークを使用し Struts アプリケーションを開発する場合にはポートレットに対応した開発を行う必要があります。ポートレットに対応させるためには通常の Struts アプリケーションの開発に加えて以下の対応が必要です。

(1) ウィンドウ状態について

ポートレットはポータル画面に複数のポートレットを配置するため，ウィンドウ状態に応じて画面のサイズが変わります。このため，ウィンドウ状態により画面レイアウトが崩れることがありますのでウィンドウ状態に合わせた画面設計が必要です。

- NORMAL ステートの場合
  ポータル画面の一部として表示されるため，文字サイズやテーブルサイズが大きすぎてレイアウトが崩れないように設計してください。
- MAXIMIZED ステートの場合
  ポータル画面全体に表示されるため，文字サイズやテーブルサイズが小さすぎてウィンドウの大きさに対してコンテンツが小さくなりすぎないように設計してください。

(2) 拡張 Struts タグライブラリの使用

Struts アプリケーションをポートレットとして動作させるため，拡張 Struts タグライブラリを提供しています。拡張 Struts タグライブラリを使用することでアプリケーション実行にウィンドウ状態やポートレットモードの制御を行うことができるようになります。拡張 Struts タグライブラリの詳細については「14.21　拡張 Struts タグライブラリ」を参照してください。

(3) ポートレットとして動作するための制限事項

(a) リダイレクト

ポートレットとして動作するため，Struts アプリケーションからリダイレクトの処理を行うことはできません。

(b) リクエストパラメタの予約語

uCosminexus Portal Framework では以下のリクエストパラメタを予約しています。こ

のリクエストパラメタを使用することはできません。

リクエストパラメタ

- _spage
- _sorig
- _kra

リクエストパラメタ（接頭部）

- hptl.
- java.
- javax.

(c) リクエスト属性の予約後

uCosminexus Portal Framework では以下のリクエスト属性予約しています。このリクエスト属性を使用することはできません。

リクエスト属性（接頭部）

- org.apache.portlet.bridges.struts.
- jp.co.hitachi.soft.portal.

(d) サーブレット API の制限事項

サーブレット API から取得できる情報が以下の通り制限されます。

HttpServletRequest クラス

- getPathTranslated() メソッド
  null が返却されます。
- getRealPath() メソッド
  null が返却されます。
- getParameter() メソッド
  レンダーパラメタが取得できます。レンダーパラメタはアクションモードで標準 API ポートレットを呼び出した時のリクエストパラメタが設定されます。
  getParameterNames() メソッド，getParameterValues() メソッド，getParameterMap() メソッドについても同様にレンダーパラメタが取得できます。

HttpServletResponse クラス

- EncodeRedirectUrl() メソッド
  引数に指定した値が返却されます。

(e) ポータル画面からのポートレットモードおよびウィンドウステートの変更

ポータル画面からポートレットモードおよびウィンドウステートが変更される場合があります。この場合は，レンダーモードとして実行されるため，レンダーパラメタの引継ぎは行われません。

(4) リクエストのフォワード

リクエストをフォワードしてもポータル画面全体の描画を変更することはできません。ポートレット内容表示領域の描画変更となります。

(5) ライブラリファイル

Struts アプリケーションを標準 API ポートレットとして動作させるためのライブラリおよび TLD ファイルは，標準 API ポートレットのデプロイ時にアーカイブ内にコピーされます。コピーされるファイルは，以下のディレクトリに格納されています。

{uCosminexus Portal Frameworkインストールディレクトリ}¥container¥

3.23.2 ポートレットを作成するための設定

開発した Struts アプリケーションを標準 API ポートレットとして作成するために，次に示す設定を行います。標準 API ポートレットのディレクトリ構成は「3.25.1 標準 API ポートレットのディレクトリ構成」を参照してください。

(1) web.xml の編集

以下の定義を追加します。

サーブレット定義

```
<servlet>
  <servlet-name>action</servlet-name>

<servlet-class>jp.co.hitachi.soft.portal.portal.portlets.struts
bridge.PortletServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>debug</param-name>
    <param-value>2</param-value>
  </init-param>
  <init-param>
    <param-name>detail</param-name>
    <param-value>2</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

サーブレットマッピング定義

```
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.shtml</url-pattern>
</servlet-mapping>
```

タグライブラリ定義

```
<taglib>
  <taglib-uri>http://soft.hitachi.co.jp/portal/bridges/struts/
tags-portlet-html</taglib-uri>
  <taglib-location>/WEB-INF/cosmi/portal/taglib/
struts-portlet-html-1.0.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>http://soft.hitachi.co.jp/portal/bridges/struts/
tags-portlet-html-el</taglib-uri>
  <taglib-location>/WEB-INF/cosmi/portal/taglib/
struts-portlet-html-el-1.0.tld</taglib-location>
</taglib>
```

(2) ポートレットアプリケーション DD の作成

Portlet Specification 1.0 に従いポートレットアプリケーション DD（portlet.xml）を作成してください。portlet.xml を作成する際には以下の情報を設定してください。

ポートレットクラス（portlet-class）

次の値を設定します。

```
jp.co.hitachi.soft.portal.portal.portlets.strutsbridge.StrutsPo
rtlet
```

パラメタ（init-param）

- ServletContextProvider

  次の値を設定します。必ず設定してください。

  ```
  jp.co.hitachi.soft.portal.portal.portlets.strutsbridge.Servle
  tContextProviderImpl
  ```

- ViewPage

  VIEW モード用の画面のパスを設定します。必ず設定してください。

- HelpPage

  HELP モード用の画面のパスを設定します。設定は任意です。

- EditPage

  EDIT モード用の画面のパスを設定します。設定は任意です。

定義例

```
<portlet-class>jp.co.hitachi.soft.portal.portal.portlets.struts
bridge.StrutsPortlet</portlet-class>
    <init-param>
      <name>ServletContextProvider</name>

<value>jp.co.hitachi.soft.portal.portal.portlets.strutsbridge.S
ervletContextProviderImpl</value>
    </init-param>
    <init-param>
      <name>ViewPage</name>
      <value>/index.shtml</value>
    </init-param>
    <init-param>
      <name>HelpPage</name>
```

```
    <value>/help.shtml</value>
  </init-param>
  <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>VIEW</portlet-mode>
    <portlet-mode>HELP</portlet-mode>
  </supports>
```

（3） struts-config.xml ファイルの編集

controller タグを次のとおりに設定してください。

```
<controller pagePattern="$M$P" inputForward="false"
processorClass="jp.co.hitachi.soft.portal.portal.portlets.strutsbr
idge.PortletRequestProcessor"/>
```

3.23.3 日立 API の使用

ポートレットとして開発した Struts アプリケーションでは日立 API を使用することができます。日立 API を使用する場合の注意事項を以下に説明します。

（1） uCosminexus Portal Framework 未起動時の動作

uCosminexus Portal Framework が起動していない場合に日立 API の呼び出しを行った場合，例外がスローされます。この事象を回避するためには，uCosminexus Portal Framwork を起動してからポートレットを動作させてください。

（2） 日立 API が出力するログ

日立 API が出力するログはポータル保守ログに出力されます。

（3） コンテキストパスの取得方法

uCosminexus Portal Framework が提供している JavaScript ファイルや画像ファイルなどを参照する場合は次のリクエスト属性からコンテキストパスを取得してください。リクエストクラスの getContextPath() メソッドからは取得することはできません。

リクエスト属性名：`hptl.std.portal.contextpath`

（4） 画面モードの対応

日立 API が提供している画面モードと標準 API のポートレットモードは次の通り対応しています。

表 3-24 画面モードの対応（URL 生成時の指定）

ポートレットモード	ウィンドウステート				
	NORMAL ステート	MAXIMIZED ステート	MINIMIZED ステート	NEWWINDOW ステート	IFRAME ステート
VIEW モード	DEFAULT	MAXIMIZE	—	NEWWINDOW	IFRAME
EDIT モード	—	EDIT	—	—	—
HELP モード	—	—	—	—	—

表 3-25 画面モードの対応（現在の画面モード取得時）

ポートレットモード	ウィンドウステート				
	NORMAL ステート	MAXIMIZED ステート	MINIMIZED ステート	NEWWINDOW ステート	IFRAME ステート
VIEW モード	DEFAULT	MAXIMIZE	—	NEWWINDOW	IFRAME
EDIT モード	EDIT	EDIT	—	—	—
HELP モード	—	—	—	—	—

(5) 標準 API ポートレットのポートレット ID

日立 API に指定するポートレット名に標準 API ポートレットのポートレット ID を指定する場合は，次のルールにより生成した ID を指定してください。

“ 標準 API ポートレットが動作するコンテキストパス ”+”.”+” ポートレット ID ”

3.24 デバイスに対応するための設定（標準 API ポートレット）

ポートレットを各デバイスに対応させるためには，ポートレットアプリケーション DD に <mime-type> の宣言を行う必要があります。ポートレットアプリケーション DD の詳細は Java Portlet Specification 1.0 の仕様を確認してください。

3.25 標準 API ポートレットのアーカイブの作成

ここでは，標準 API ポートレットのアーカイブの作成について説明します。標準 API ポートレットのアーカイブとは，ポートレットの実行に必要な，ポートレットコンテンツ（Java クラス /JSP/HTML ファイル）およびポートレットアプリケーション DD（portlet.xml）などを一つにまとめた（パッケージした）ファイルのことです。このファイルを利用すると，ポートレットごとにクラスやコンテンツのデプロイおよびアンデプロイができ，複数のポートレットを効率良く管理できます。

標準 API ポートレットのアーカイブの作成手順を次に示します。

1. **アーカイブファイルを作成するときのトップディレクトリ以下に，ポートレットコンテンツを格納します。**
   ここでは，アーカイブファイルを作成するときのトップディレクトリを {PORTLET_HOME} と表記します。アーカイブファイルを作成するときの，ポートレットのディクトリ構成については，「3.25.1　標準 API ポートレットのディレクトリ構成」を参照してください。
2. **ポートレットアプリケーション** DD（portlet.xml）**を作成します。**
   ポートレットアプリケーション DD の詳細は，「3.25.2　ポートレットアプリケーション DD（portlet.xml）の作成」を参照してください。
3. Java **アーカイブツールを使用してパッケージします。**
   パッケージするには，コマンドプロンプト上でカレントディレクトリを {Cosminexus インストールディレクトリ} ￥JDK￥bin に移動し，次のコマンドを実行します。

   ```
   > jar cvf {アーカイブファイル格納ディレクトリ}/{ファイル名} -C
   {PORTLET_HOME}
   ```

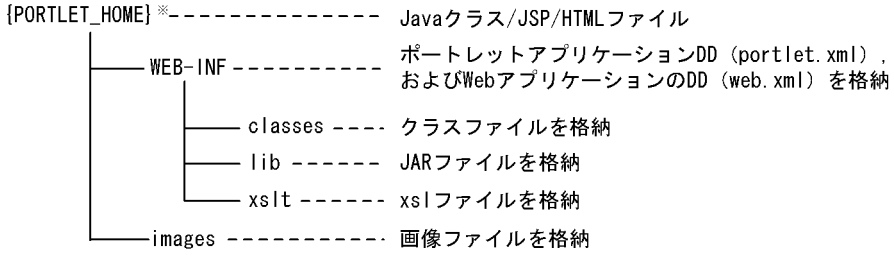
4. **デプロイ用ファイルを作成し，**EAR/WAR **ファイルをデプロイします。**
   手順 3 で作成したファイルから，デプロイ用のアプリケーションファイルを作成し，デプロイします。
   EAR/WAR ファイルのデプロイについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「標準 API ポートレットのデプロイ」の説明を参照してください。

3.25.1 標準 API ポートレットのディレクトリ構成

「3.25　標準 API ポートレットのアーカイブの作成」の手順 3. でファイルを作成するときの，標準 API ポートレットのディレクトリ構成について説明します。

ポートレットを格納するときには，ポートレットごとにディレクトリを作成します。
ファイル内のディレクトリ構成を次の図に示します。

図 3-20　標準 API ポートレットのディレクトリ



注※　アーカイブファイル作成時のトップディレクトリです。
ディレクトリ名には，任意の名称を付けてください。

Java クラス /JSP/HTML ファイル

ポートレットディレクトリ直下に格納します。統合画面内で遷移させる JSP ファイルは同じポートレットディレクトリ下に格納します。

ポートレットアプリケーション DD（portlet.xml）

ポートレットアプリケーション DD は，{PORTLET_HOME}¥WEB-INF に格納します。なお，ポートレットアプリケーション DD の詳細は「3.25.2　ポートレットアプリケーション DD（portlet.xml）の作成」を参照してください。

Web アプリケーションの DD（web.xml）

サーブレットのマッピング定義などに使用する Web アプリケーションの DD（web.xml）を {PORTLET_HOME}¥WEB-INF に格納します。

クラスファイル

サーブレットやユティリティクラスなどのクラスファイルは，
{PORTLET_HOME}¥WEB-INF¥classes に格納します。
{PORTLET_HOME}¥WEB-INF¥classes に格納したクラスファイルのデプロイ後の展開先は，uCosminexus Portal Framework がポータルプロジェクト内で使用するクラスフォルダとなるため，名前が重複するおそれがあります。また，ディレクトリ名の大文字小文字の区別がない OS を利用している場合，クラスのパッケージ名が不正となる場合があります。そのため，JAR ファイルを使用して格納することを推奨します。

JAR ファイル

JAR ファイルは，{PORTLET_HOME}¥WEB-INF¥lib に格納します。なお，"hptl" で始まる JAR ファイル名は予約されているため使用できません。
{PORTLET_HOME}¥WEB-INF¥lib に格納した JAR ファイルのデプロイ後の展開先は，uCosminexus Portal Framework がポータルプロジェクト内で使用するライブ

ラリフォルダとなります。なお，JAR ファイルのファイル名は，ポートレットアプリケーション DD（portlet.xml）で指定したポートレット名を接頭語とするファイル名にリネームされるため，ほかのポートレットとの名前重複を考慮する必要はありません。

画像ファイル

通常の Web アプリケーション作成時と同様に格納します。

3.25.2 ポートレットアプリケーション DD（portlet.xml）の作成

ポートレットアプリケーション DD（portlet.xml）とは，標準 API ポートレットの名称やクラス名を設定するファイルです。ポートレットアプリケーション DD ファイルのサンプルおよび注意事項を次に示します。なお，ポートレットアプリケーション DD で使用するタグは，Java Portlet Specification 1.0 の仕様を参照してください。

(1) ポートレットアプリケーション DD の作成例

サンプルのポートレットアプリケーション DD を次に示します。

サンプルのポートレットアプリケーション DD

```
<?xml version="1.0" encoding="UTF-8" ?>
<portlet-app
  version="1.0"
  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 http://java.sun.com/xml/ns/portlet/
portlet-app_1_0.xsd">
  <portlet>
    <description>Sample Portlet</description>
    <portlet-name>SamplePortlet</portlet-name>
    <display-name>Sample Portlet</display-name>
    <portlet-class>portlets.SamplePortlet</portlet-class>
    <expiration-cache>-1</expiration-cache>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Sample Portlet</title>
      <short-title>Sample</short-title>
      <keywords>Sample,Sample Portlet</keywords>
    </portlet-info>
  </portlet>
</portlet-app>
```

(2) ポートレットアプリケーション DD 作成時の注意事項

ポートレットアプリケーション DD に設定するタグの注意事項を次に示します。

- <portlet-name> タグ：半角英数字 1 文字以上 32 文字以内の文字列を指定してくださ

い。

- <portlet-mode> タグ：「EDIT」，「HELP」または「VIEW」のいずれかの文字列を指定できます。大文字，小文字の区別はありません。
- <read-only> タグ：「true」または「false」のいずれかの文字列を指定できます。大文字，小文字を区別します。
- <transport-guarantee> タグ：「NONE」，「INTEGRAL」または「CONFIDENTIAL」のいずれかの文字列を指定できます。大文字，小文字を区別します。

4 File ポートレットの作成

この章では，File ポートレットを作成する際の手順，および注意事項について説明します。

4.1 File ポートレットとは

File ポートレットは，HTML，CHTML，HDML のどれかで記述されたファイルをポートレットとして表示します。例えば，イントラネット上の各部署のお知らせを，一つの File ポートレットに統合できます。携帯電話（ｉ モード，および EZweb）からアクセスできるポートレットも作成できます。

4.2　File ポートレットの作成手順

この節では，File ポートレットを作成する手順について説明します。

手順

1. File **ポートレットのコンテンツを開発します。**
   コンテンツを開発する際には，ターゲットとするデバイス（PC，ｉモード，または EZweb）に対応する HTML/CHTML/HDML のバージョンで記述する必要があります。uCosminexus Portal Framework が対応している各言語のバージョンについては，「2.1.1　ターゲットとするデバイス」を参照してください。
   また，コンテンツを開発する際には，次の注意事項を確認してください。
   - 「2.　ポートレット全般の前提知識」
   - 「4.3　File ポートレット開発時の注意」
2. File **ポートレットを作成します。**
   ターゲットとするデバイスに，ポートレットを表示するための設定を行います。
   この設定の詳細は，「4.4　デバイスに対応するための設定」を参照してください。
3. File **ポートレットをポータルに登録します。**
   ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

4.3 File ポートレット開発時の注意

4.3.1 エンコーディング

File ポートレットでは，プロパティファイルのデフォルトエンコーディングで指定されている文字コードで記述します。デフォルトエンコーディングについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「デフォルトエンコーディングの設定」の説明を参照してください。

4.3.2 リンク・インラインオブジェクトの指定方法

File ポートレットで使用するリンクおよびインラインオブジェクトは，絶対 URL 形式，または {PROJECT_HOME} を基点とした絶対パス形式で指定する必要があります。相対パス形式で指定した場合，ドキュメントやインラインオブジェクトを正しく取得できません。

4.4 デバイスに対応するための設定

File ポートレットでは，PC だけでなく携帯電話（ｉ モード，および EZweb）からアクセスできるコンテンツを作成します。コンテンツは，クライアントのデバイス（PC，ｉモード，または EZweb）ごとに対応する言語で作成する必要があります。

ポータルにアクセスできるクライアントのデバイスの種類と，対応する言語を次の表に示します。

表 4-1　クライアントのデバイスの種類と対応する言語（File ポートレット）

クライアントのデバイス	対応する言語
HTML4.01 に対応する Web ブラウザのある PC	HTML
i モード対応 HTML2.0 に対応する i モードの携帯電話	CHTML
Version 3.3 for HDML に対応する EZweb の携帯電話	HDML

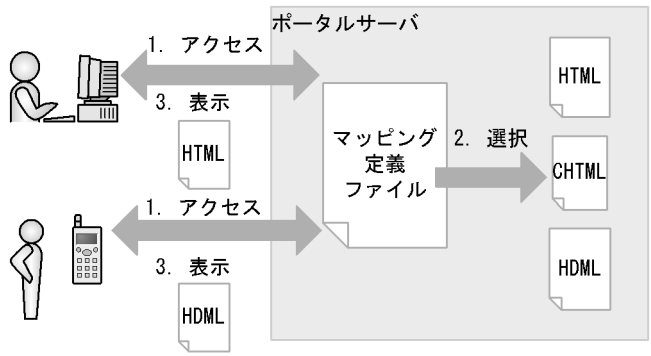
なお，クライアントのデバイスに合ったコンテンツ表示するために，対応する言語ごとに文書を書き換える必要があります。

- HTML ／ CHTML の場合：ドキュメントタイプの宣言の追加，および背景色の指定
- HDML の場合：ドキュメントタイプの宣言の追加

アクセスしてきたクライアントの種別を判別して，クライアントに合ったコンテンツを選択するために，マッピング定義ファイルを使用します。一つのポートレットに一つのマッピング定義ファイルが必要です。

マッピング定義ファイルとクライアント判別について次の図に示します。

図 4-1　マッピング定義ファイルとクライアント判別（File ポートレット）



図の説明

1. ユーザがポータルにアクセスしたときに，マッピング定義ファイルでユーザのクライアント種別を判別します。

2. クライアント種別に合致したファイルを選択します。
3. 選択したファイルをポータルで表示します。

マッピング定義ファイルは，ポートレット登録時に作成されます。Portal Manager でポートレットを登録するときに，マッピング定義ファイルの作成場所を指定します。ポートレットごとに作成したポートレットディレクトリの直下を指定します。

5 Web ポートレットの作成

この章では，Web ポートレットを作成する際の手順，前提知識，および注意事項について説明します。

5.1 Web ポートレットとは

Web ポートレットは，Web ページをポートレットとして取り込みます。Web ポートレットを使うと，ポータルプロジェクト内のローカルコンテンツおよび外部の Web サーバのコンテンツをポータルに取り込めます。

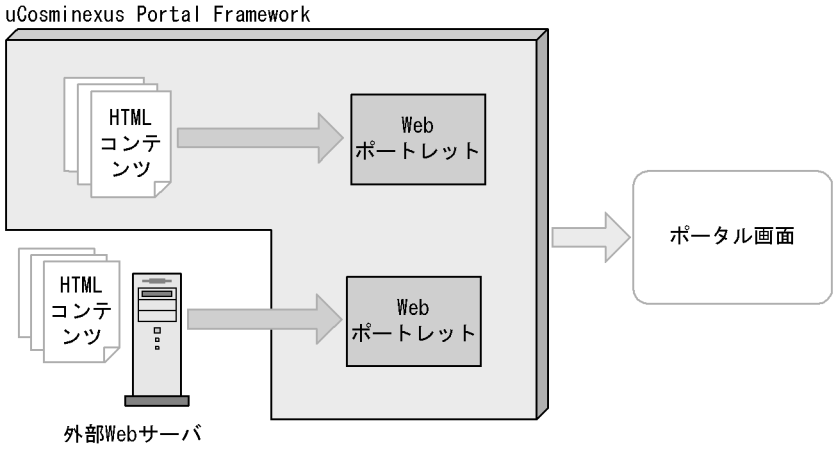
Web ポートレットは HTTP によってコンテンツを取得し，ポートレットとしてポータル画面に統合します。

注意事項

外部の Web ページを取り込む際は，取り込み先の Web ページの利用承諾条件や利用規約に基づいてください。

Web ポートレットの取り込みを次の図に示します。

図 5-1 Web ポートレットの取り込み



5.1.1 Web ポートレットの種類

Web ポートレットには，Multi Web Portlet，Web App Portlet，および Web Page Portlet があります。それぞれのポートレットの説明を次に示します。

Multi Web Portlet

- PC だけでなく携帯電話（ｉ モード，および EZweb）からアクセスできるコンテンツも取り込めます。
- Portal Manager で，ポートレット登録時にフォーム認証に必要な設定ができます。

Web App Portlet

外部の HTML コンテンツをポータル内で画面遷移させたり，動的なコンテンツを表

示したりできます。

Web Page Portlet

静的なコンテンツだけを表示できます。ただし，Web Page Portlet は認証およびセッション維持ができないので，外部の HTML リンクをポータル内で画面遷移できません。

Multi Web Portlet は，Web App Portlet と Web Page Portlet の機能を包含するため，Multi Web Portlet の使用を推奨します。

Multi Web Portlet，Web App Portlet，および Web Page Portlet の相違点を次の表に示します。

表 5-1　Multi Web Portlet，Web App Portlet，および Web Page Portlet の相違点

ポートレットの種別	対応する言語	使用する認証情報定義ファイル
Multi Web Portlet	HTML，CHTML，HDML	ポートレット定義ファイル※1
Web App Portlet	HTML	フォーム認証定義ファイル※2
Web Page Portlet	HTML	―※3

注※1　ポートレット定義ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット定義ファイル（jetspeed-config.jcfg）」の説明を参照してください。
注※2　フォーム認証定義ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「フォーム認証定義ファイル（formbase-auth.cfg）」の説明を参照してください。
注※3　Web Page Portlet は，フォーム認証は行えません。

5.2 Web ポートレットの作成手順

この節では，Web ポートレットを作成する手順について説明します。

手順

1. **Web ポートレットの機能を決定します。**
   ポータルに取り込む Web ポートレットを決定します（Web ポートレットの種類の決定）。
2. **Web ポートレットを作成します。**
   ポータルに取り込める Web ポートレットは，uCosminexus Portal Framework が決めた条件に合っている必要があります。Web ポートレットを取り込むための条件は，「5.3.2　Web ポートレットを取り込むための条件」を参照してください。

   Multi Web Portlet をポータルに取り込む場合

   コンテンツに日本語を記述する場合，エンコーディング方法を指定する必要があります。エンコーディング方法の指定については，「5.4.1(1)(b) エンコーディング」を参照してください。
   Multi Web Portlet を取り込む場合，ターゲットとするデバイス（PC，ｉモード，または EZweb）に，ポートレットを表示するための設定を行います。
   この設定の詳細は，「5.5　デバイスに対応するための設定（Multi Web Portlet）」を参照してください。

   Web App Portlet および Web Page Portlet をポータルに取り込む場合

   コンテンツ中に日本語を記述する場合，エンコーディング方法を指定する必要があります。エンコーディング方法の指定については，「5.4.2(1)(b) エンコーディング」を参照してください。
3. **Web ポートレットをポータルに登録します。**
   Web ポートレットをポータルに登録する際に，コンテンツに含まれるタグ，および要素をフィルタリングできます。Multi Web Portlet の要素フィルタリングで取り除かれるタグ，および要素については，「5.4.1(5) 要素フィルタリング」を，Web App Portlet および Web Page Portlet の要素フィルタリングで取り除かれるタグ，および要素については，「5.4.2(5) 要素フィルタリング」を参照してください。
   また，Web ポートレットをポータルに登録する際に，ポータルに取り込むコンテンツを抽出，または削除できます。Multi Web Portlet のコンテンツフィルタリングについては，「5.4.1(6) コンテンツフィルタリング」を，Web App Portlet および Web Page Portlet のコンテンツフィルタリングについては，「5.4.2(6) コンテンツフィルタリング」を参照してください。
   要素フィルタリングおよびコンテンツフィルタリングは，ポートレットの登録時に設定できます。ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

5.3 Web ポートレット全般の前提知識

この節では，Web ポートレットが対応する HTML のバージョン，およびポータルに取り込める Web ポートレットの条件について説明します。

5.3.1 HTML バージョン

Web ポートレットでは MIME タイプとして text/html に対応しています。また，HTML 4.01 に対応しています。

5.3.2 Web ポートレットを取り込むための条件

Web ポートレットを取り込むための条件を，Web ポートレットごとに説明します。次の場合は，Web ポートレットを取り込めません。

Multi Web Portlet

- **JavaScript を使用している Web サイトの場合**
  - JavaScript の onload および onunload
    JavaScript の onload および onunload は無視されます。
  - JavaScript 内に URL を記述しているとき
    JavaScript で指定した URL は，認識されません。

    （例）`window.open`
  - HTML を JavaScript で動的に更新しているとき
    FORM 要素の ACTION 属性の URL を動的に更新するときです。

    （例）`location.href`
  - ほかのポートレットと名称が重複するとき
    複数のポートレットで JavaScript を使用している場合，スクリプトの関数やグローバル変数の定義は，ほかと重複しない名称にしてください。
  - JavaScript による Target 要素の書き換えがあるとき
    LINK 要素の TARGET が，SCRIPT によって書き換えられているとき，最大画面に切り替わりません。また，TARGET 要素の書き換えによっては，ポートレットにコンテンツが正しく表示されないことがあります。
- **HTML のエンコーディング設定がない Web サイトの場合**
  Web ポートレットを取り込むには，次のどちらかのエンコーディング設定をしてください。
  - HTTP レスポンスヘッダの Content-Type フィールドの charset パラメタ
  - META 要素中の Content-Type 指定の charset パラメタ
- **スタイルシート内に URL が記述されているポータルシステムの場合**

- **ポータルシステムが認識できない部分で URL 変換する仕組みを持つ Web サイトの場合**
- **iframe タグを含む Web サイトの場合**
  次の Web ブラウザで参照することが前提となります。
  - Internet Explorer
  - Firefox
  - Netscape Navigator
- **画面の表示先の指定がポータルの画面外である Web サイトの場合**
  HTML コンテンツの画面の表示先（A 要素の TARGET 属性）に「PARENT」が指定されている場合などです。

Web App Portlet および Web Page Portlet

- **フレーム（FRAMESET 要素）を利用している Web サイトの場合**
- **JavaScript を使用している Web サイトの場合**
  - JavaScript の onload および onunload
    JavaScript の onload および onunload は無視されます。
  - JavaScript 内に URL を記述しているとき
    JavaScript で指定した URL は，認識されません。

    （例）`window.open`

  - HTML を JavaScript で動的に更新しているとき
    FORM 要素の ACTION 属性の URL を動的に更新するときです。

    （例）`location.href`

  - ほかのポートレットと名称が重複するとき
    複数のポートレットで JavaScript を使用している場合，スクリプトの関数やグローバル変数の定義は，ほかと重複しない名称にしてください。
- **HTML のエンコーディング設定がない Web サイトの場合**
  Web ポートレットを取り込むには，次のどちらかのエンコーディング設定をしてください。
  - HTTP レスポンスヘッダの Content-Type フィールドの charset パラメタ
  - META 要素中の Content-Type 指定の charset パラメタ
- **スタイルシート内に URL が記述されているポータルシステムの場合**
- **ポータルシステムが認識できない部分で URL 変換する仕組みを持つ Web サイトの場合**

5.4 Web ポートレット作成時の注意事項

この節では，Web ポートレットを作成する際に，注意していただきたいことについて説明します。

5.4.1 Multi Web Portlet 作成時の注意事項

(1) HTTP エミュレーション

(a) HTTP エミュレーション

Multi Web Portlet は，HTTP で HTML コンテンツを取得して，ポートレットとしてポータルに統合します。Multi Web Portlet は HTTP1.1 のサブセットに対応します。HTTP レスポンスコードの取り扱いを次の表に示します。

表 5-2 レスポンスコードの取り扱い（Multi Web Portlet）

レスポンスコード	内容
1xx：Informational	結果フレーズをポートレットおよびログに出力します。
2xx：Successful	200（正常終了）に対応します。201 から 206 は結果フレーズをポートレットおよびログに出力します。
3xx：Redirection	301，302，303，307（リダイレクト）に対応します。300，304，305，および GET，または HEAD 以外のリクエストによって 301，または 307 が返ってきた場合は，結果フレーズをポートレットおよびログに出力します。※
4xx：Client Error	401（Unauthorized）および 407（Proxy Authentication Required）に対応します。 401 および 407 以外は，結果フレーズをポートレットおよびログに出力します。
5xx：Server Error	結果フレーズをポートレットおよびログに出力します。

注※ Multi Web Portlet では，301 または 302 が返ってきた場合，レスポンスコードへの対応は使用する Web ブラウザによって異なります。

対応するレスポンスヘッダフィールドの取り扱いを次の表に示します。

表 5-3 レスポンスヘッダフィールドの取り扱い（Multi Web Portlet）

ヘッダフィールド	内容
Cache-Control	キャッシュ制御に使用します。キャッシュ制御の指定は，Portal Manager の Web ポートレット設定時に定義できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。
Content-Type	コンテンツのエンコーディング判定に使用します。text/html だけに対応します。
Last-Modified	コンテンツの最終更新時刻です。キャッシュ制御に使用します。

ヘッダフィールド	内容
Location	リダイレクト先の URL です。リダイレクト時に使用します。
Set-Cookie	指定された Cookie 値をサーバ内で保持します。
Set-Cookie2	指定された Cookie 値をサーバ内で保持します。
WWW-Authenticate	ベーシック認証を試みます。
Proxy-Authenticate	プロキシ認証を試みます。

クライアントからのリクエストヘッダフィールドの取り扱いを次の表に示します。

表 5-4　リクエストヘッダフィールドの取り扱い（Multi Web Portlet）

ヘッダフィールド	内容
Host	リクエスト先のホスト名を設定します。
Accept	対応する MIME タイプを設定します。
Accept-Encoding	対応するエンコーディングを設定します。
Accept-Language	対応する言語を設定します。
User-Agent	リクエストした User-Agent を設定します。
If-Modified-Since	コンテンツの最終更新時刻を設定します。
Cookie	Set-Cookie1 でサーバに保存された Cookie を設定します。
Cookie2	Set-Cookie2 でサーバに保存された Cookie を設定します。
Authorization	ベーシック認証に必要な認証情報を設定します。
Proxy-Authorization	プロキシ認証に必要な認証情報を設定します。

(b) エンコーディング

HTML コンテンツに日本語を記述する場合には，通常の HTML コンテンツ同様，エンコーディング方法を指定する必要があります。エンコーディング方法の決定は，外部の Web サイトに格納された HTML コンテンツと uCosminexus Portal Framework 内に配置した HTML コンテンツで異なります。

uCosminexus Portal Framework では，ユーザの環境による文字化けを防ぐために，ユーザの環境ごとにエンコーディング方法を推奨しています。uCosminexus Portal Framework が推奨するエンコーディングについては，「2.1.4　エンコーディング」を参照してください。

● 外部 Web サーバ上の HTML コンテンツのエンコーディング指定

エンコーディング方法は，次の優先順位で決定されます。

1. HTTP レスポンスヘッダの Content-Type フィールドの charset パラメタ
2. META 要素中の Content-Type 指定の charset パラメタ

3. **自動判定**

自動判定では，正しいエンコーディング方法を決定できないことがあるので，1.，または2. でのエンコーディング指定を推奨します。

(c) プロキシ連携

プロキシを経由してコンテンツを取得できます。プロキシと連携するには，Portal Manager の Web ポートレット設定時に定義できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(d) リダイレクト

Multi Web Portlet は，リダイレクト回数設定はできません。

(2) 認証

(a) 認証に対するシングルサインオン

PC からポータルにアクセスしている場合，ディレクトリサーバに格納されているユーザ情報（パーソナライズ情報）から認証情報を取得すれば，Web アプリケーションとのシームレスな連携ができます。

認証機能が利用できる前提条件を次に示します。

- ポートレットに表示しているアプリケーションへの，ログイン前のページの認証
  認証先が参照のチェックをするアプリケーションの場合
- JavaScript を使用している Web サイトの認証
  JavaScript で認証フィールドを動的に変更および生成している場合

● ベーシック認証

Web サーバのレスポンスコードが 401（Unauthorized）の際，次の処理をします。

- ユーザログイン時
  対象 URL に対応するユーザ ID，およびパスワードをリポジトリから取得して自動的に認証します。この機能は，ユーザがユーザ ID とパスワードを登録すれば使用できるようになります。登録されていない場合は，ユーザ ID とパスワードを登録するフォームを出力します。なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。
- ウェルカム画面（ユーザがログインしていない場合）
  ウェルカム画面での Web サーバに対応する認証はしません。

● プロキシ認証

プロキシサーバのレスポンスコードが 407（Proxy Authentication Required）の際，次の処理をします。

- ユーザログイン時

ユーザがログインすると，ユーザ情報取得 Bean を使用して，リポジトリから対象となるプロキシサーバに対応するユーザ ID およびパスワードを取得して，自動的に認証します。該当するユーザ ID，およびパスワードが登録されていない場合は，ユーザが入力できるようにフォームを出力します。なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。

- ウェルカム画面（ユーザがログインしていない場合）
  ウェルカム画面でのプロキシサーバに対応する認証はしません。

● フォーム認証

Web サーバへのリクエスト情報を基にシングルサインオンを実現します。Multi Web Portlet の場合，認証するサイトによって，認証画面を通過する必要があるとき，シングルサインオンはできません。パスワードの入力ミスなどによってフォーム認証に失敗した場合は，認証情報をクリアして，再度認証情報を入力する必要があります。

Multi Web Portlet の場合は，ポートレットの編集ボタンをクリックし，「フォームベース認証のクリア」を実行して，再度認証情報を入力してください。

フォーム認証をするためには，次に示す項目をあらかじめ設定しておくことが必要です。

- 認証するページの URL
- ログインするページの URL
- 認証時に使用するメソッド
- 認証情報を示すフィールド

なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。

Multi Web Portlet は，Portal Manager でフォーム認証情報を設定します。Multi Web Portlet の場合のフォーム認証を設定する方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

（3） URL 変換

Multi Web Portlet では，ポータル画面内で画面遷移するために，Web ページに含まれている相対パス形式および絶対パス形式の URL を，uCosminexus Portal Framework を経由する URL に変換します。BASE 要素で基準となる URL が指定されている場合，BASE 要素を基に自動的に変換します。

HTML コンテンツ中に含まれる画像などのバイナリデータを，uCosminexus Portal Framework を経由して取得するか，または Web ブラウザが直接 Web サーバから取得するかを Portal Manager でポートレットを登録するときに選択できます。この設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(4) キャッシュ制御

Multi Web Portlet では，uCosminexus Portal Framework 内に HTML，CHTML，および HDML コンテンツをキャッシングしません。ただし，クライアントと転送先の Web サーバ間では，日付情報の有効期限を基に，クライアントに蓄積されたキャッシュを使用できます。

(5) 要素フィルタリング

要素フィルタリングは，Multi Web Portlet の登録時に，フィルタリングするかしないかを設定できます。タグ，要素は取り除かれますが，コンテンツを表示するために必要な BASE 要素やエンコーディング設定，およびキャッシュ制御に必要な META 要素の一部はコンテンツ作成のために参照されます。

Multi Web Portlet は，BODY 本体内に記述できない開始，終了タグや要素を取り除きません。

Multi Web Portlet のフィルタリングは Portal Manager の［ポートレットの設定］画面で設定します。フィルタリングの設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

要素フィルタリングで取り除かれるタグ，要素を次に示します。

- APPLET 要素
- OBJECT 要素
- SCRIPT 要素

なお，SCRIPT 要素を設定すると，on で始まるイベント属性も設定されます。

(6) コンテンツフィルタリング

コンテンツフィルタリングとは，HTML ファイルを Multi Web Portlet として取り込む際に，コンテンツを抽出，または削除することです。コンテンツフィルタリングを使用すると，HTML ファイルの一部分をポートレットに取り込めます。コンテンツフィルタリングは，Web ポートレットの登録時に設定できます。コンテンツフィルタリングの設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

コンテンツフィルタリングでは，コンテンツの BODY 要素本体に含まれるディレクティブに従ってコンテンツを抽出，または削除します。ディレクティブも同時に抽出・削除します。

ディレクティブは，コメントまたは正規表現で指定できます。

なお，コンテンツフィルタリングの設定を次に示します。

表 5-5 コンテンツフィルタリングの設定（Multi Web Portlet）

フィルタリング方式	正規表現	コメント
削除	○	×
抽出	×	×

（凡例）
○：設定できます。
×：設定できません。

正規表現でのディレクティブ指定

HTML ファイルの文字列をパターンとして指定します。パターンは Web ポートレットごとに指定できます。
パターンを指定できる範囲を次に示します。

- 先頭からパターンまで
- パターンからパターンまで
- パターンから終了まで

指定するパターンは大文字小文字を区別します。また，パターンには「,」および「|」は指定できません。HTML ファイルの文字列が，正規表現と合致しない場合，コンテンツフィルタリングをしません。
パターンからパターンまでの範囲は，複数指定できます。複数のフィルタリング範囲を指定する場合，フィルタリング範囲が重複しないようにパターンを指定してください。パターンからパターンまでの範囲では，パターン 1 およびパターン 2 を指定します。パターン 2 には，パターン 1 よりファイル内で後ろに記述されている文字列を指定します。

正規表現でのコンテンツフィルタリングの使用例

正規表現で削除するときの例を次に示します。

- 特定個所のタグを消去する

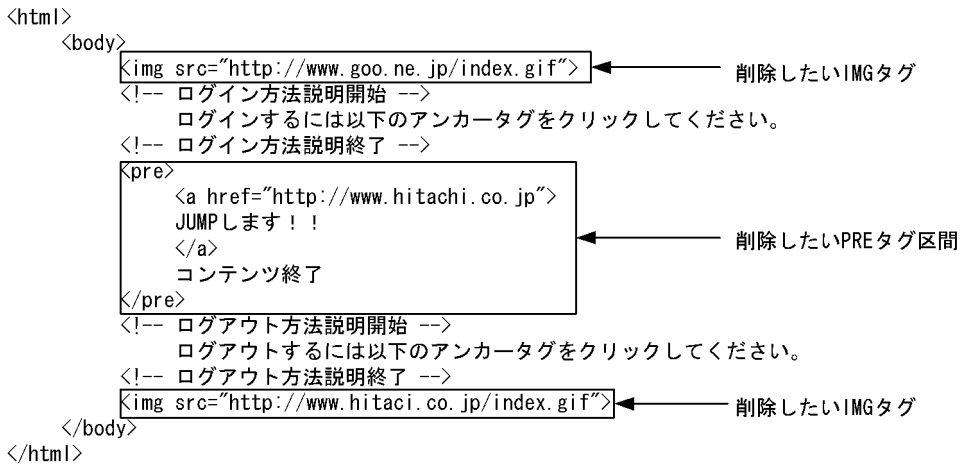
```
<html>
    <body>
        下のアンカータグをクリックしてください。
    <pre>
        <a href=http://www.hitachi.co.jp>
                JUMPします！！
        </a>                                ← 削除したいアンカータグ区間
        コンテンツ終了
    </pre>
    </body>
</html>
```

設定値

開始： ※
終了：

注※ 「/」の文字は，バックスラッシュ（「¥」または「＼」）でエスケープしてください。

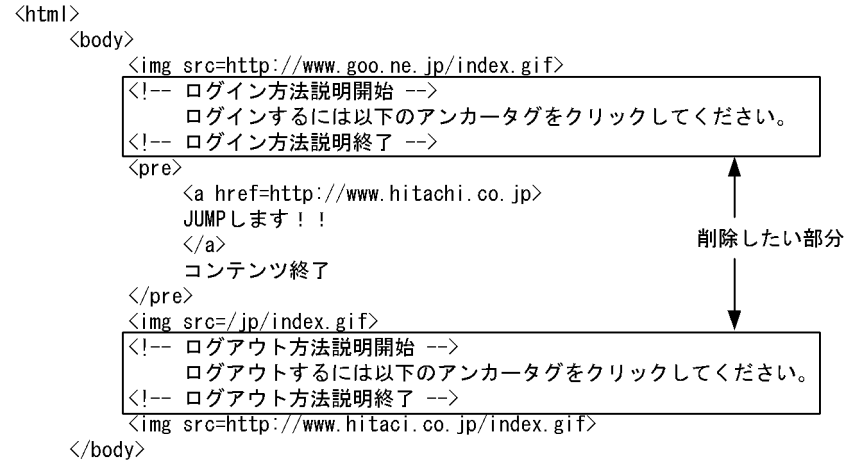
- 特定のタグを一括して消去する



設定値

開始：(<pre.*?>|<img.*?>)
終了：(</pre>|<img.*?>)

- コメント間を削除する



設定値

開始：(<!-- ログイン方法説明開始 -->|<!-- ログアウト方法説明開始 -->)
終了：(<!-- ログイン方法説明終了 -->|<!-- ログアウト方法説明終了 -->)

(7) 外部サーバとのセッション維持

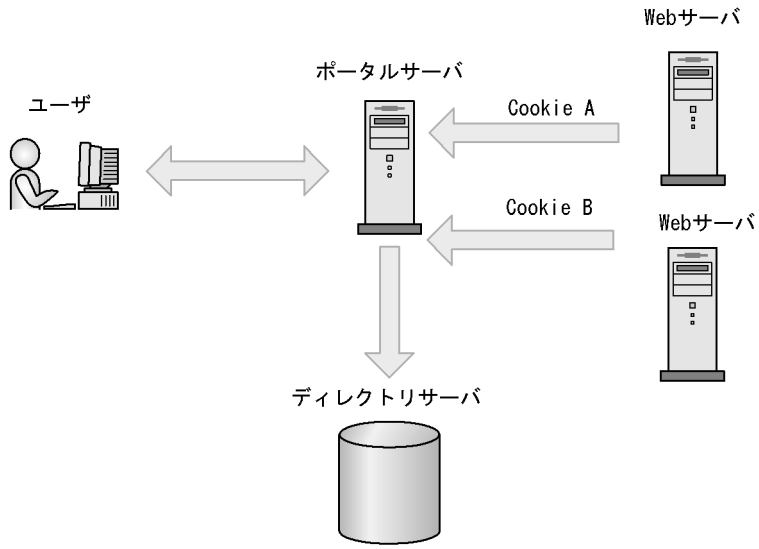
Multi Web Portlet では，外部 Web サーバとのセッションを維持できます。セッション維持の方法を次に示します。

- Cookie によるセッション維持
- URLRewriting によるセッション維持

(a) Cookie によるセッション維持

外部 Web サーバからのレスポンスから Cookie を取得します。Web ブラウザには Cookie を渡さないで，ポータルで保持します。ユーザがポータルにログインしており，かつ有効期限が永続的な Cookie は，セッションが終了するとディレクトリサーバに保存されます。Cookie によるセッション維持を次の図に示します。

図 5-2 Cookie によるセッション維持（Multi Web Portlet）



(b) URLRewriting によるセッション維持

URL を読み込んだあとに URL を書き換えて，セッションを維持します。

(8) その他の注意事項

(a) スタイル

Multi Web Portlet ではスタイルシートはページコンテンツ内のスタイルを使用します。

(b) FRAMESET

Multi Web Portlet では，FRAMESET を使用した HTML コンテンツを表示できます。

(c) スクリプトを使用するには

スクリプトを使用するには，Web ポートレットの登録の際に次の設定をしてください。

- スクリプトの関数やグローバル変数の定義は異なる名称を指定します。
- SCRIPT 要素および on で始まるイベント属性の削除オプションを［削除しない］に

します。
デフォルトでは，SCRIPT 要素および on で始まるイベント属性は削除されます。

(d) インラインオブジェクトとの連携

HTML 中に組み込まれるオブジェクトと Multi Web Portlet は連携できません。

(e) 画面遷移

Multi Web Portlet では，HTML コンテンツの場合，画面の表示先を指定する TARGET に「_PARENT」を指定しないでください。TARGET には「_TOP」，またはフレーム名を指定してください。なお，Multi Web Portlet の画面内で「_PARENT」を使用した場合，ポータル画面が遷移することがあります。

(f) 最大化表示

Multi Web Portlet の場合，次の条件のときは最大表示画面に切り替わりません。なお，最大化表示は，SCRIPT を使用している場合は設定しないでください。

- 認証画面表示直後
  通常表示画面状態で，認証画面が表示された直後は最大画面には切り替わりません。
- Frame 内のリンク要素
  Frame 内のリンク要素をクリックした場合は，最大画面に切り替わりません。
- Script による Target 要素の書き換え
  SCRIPT によって，LINK 要素の TARGET が書き換えられている場合，最大画面に切り替わりません。また，TARGET 要素の書き換えによっては，ポートレットにコンテンツが正しく表示されなくなります。
- レスポンスコードが 302 の場合
  レスポンスコードに 302 が返された場合，最大画面に切り替わりません。

(g) ポータル内のコンテンツ処理

Multi Web Portlet では，ポータル内の Web コンテンツ（HTML，CHTML，および HDML）に対して次の処理をしません。

- コンテンツ内の URL 変換
- 画面遷移時の自動最大化表示

(h) プロキシサーバ使用時の注意事項

使用するプロキシサーバの種類によっては，ソケットレベルでの通信障害が発生することがあります。ソケットレベルでの通信障害が発生した場合，ポートレット定義ファイルの Multi Web Portlet のポートレット定義に
"hptl.MultiWebPortlet.ClientCacheSwitch" パラメタを追加して，value に "NO" を指定してください。この設定によって，クライアントに蓄積されたキャッシュが使用できなくなり，通信障害を取り除けます。

なお，Portal Manager で設定を変更すると，ポートレット定義ファイルが上書きされる

ため，追加した "hptl.MultiWebPortlet.ClientCacheSwitch" パラメタは無効になります。Portal Manager で設定を変更した場合，再度，"hptl.MultiWebPortlet.ClientCacheSwitch" パラメタを追加してください。ポートレット定義ファイルについては，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット定義ファイル（jetspeed-config.jcfg）」の説明を参照してください。

ポートレット定義ファイル（jetspeed-config.jcfg）の格納場所を次に示します。

```
{PROJECT_HOME}¥WEB-INF¥conf
```

ポートレット定義ファイルに追加した場合の例，および特定のポートレットにだけ追加した場合の例を次に示します。

ポートレット定義ファイルに追加した場合の例（下線部分を追加）

```
<portlet-entry type="abstract" name="MultiWeb">

<classname>jp.co.hitachi.soft.portal.portal.portlets.MultiWebPo
rtlet</classname>
  <parameter value="NO"
name="hptl.MultiWebPortlet.ClientCacheSwitch"/>
  <parameter value="localhost" name="hptl.ProxyHost"/>
  <parameter value="8080" name="hptl.ProxyPort"/>
</portlet-entry>
```

特定のポートレットにだけ追加した場合の例（下線部分を追加）

```
<portlet-entry application="false" admin="false"
parent="MultiWeb"
                      hidden="false" type="ref" name="Hitachi">
  <url>/portlets/hitachi.xml</url>
  <parameter value="NO"
name="hptl.MultiWebPortlet.ClientCacheSwitch"/>
  <parameter value="true" name="hptl.MaximizeMode"/>
  <parameter value="false" name="hptl.MinimizeMode"/>
  <parameter value="false" name="hptl.CloseMode"/>
  <parameter value="localhost" name="hptl.ProxyHost"/>
  <parameter value="8080" name="hptl.ProxyPort"/>
  <parameter value="YES"
name="hptl.MultiWebPortlet.AppletRemove"/>
  <parameter value="YES"
name="hptl.MultiWebPortlet.ObjectRemove"/>
  <parameter value="YES"
name="hptl.MultiWebPortlet.ScriptRemove"/>
  <parameter value="Portal"

name="hptl.MultiWebPortlet.BinaryDataFetching"/>
  <parameter value="FALSE"
name="hptl.MultiWebPortlet.AutoMaxChange"/>
  <meta-info>
    <title>日立製作所</title>
    <description>日立製作所　ホームページ</description>
  </meta-info>
</portlet-entry>
```

5.4.2 Web App Portlet および Web Page Portlet 作成時の注意事項

(1) HTTP エミュレーション

(a) HTTP エミュレーション

Web ポートレットは，HTTP で HTML コンテンツを取得して，ポートレットとしてポータルに統合します。Web ポートレットは HTTP1.1 のサブセットに対応します。HTTP レスポンスコードの取り扱いを次の表に示します。

表 5-6 レスポンスコードの取り扱い（Web App Portlet および Web Page Portlet）

レスポンスコード	内容
1xx：Informational	結果フレーズをポートレットおよびログに出力します。
2xx：Successful	200（正常終了）に対応します。201 から 206 は結果フレーズをポートレットおよびログに出力します。
3xx：Redirection	301，302，303，307（リダイレクト）に対応します。300，304，305，および GET，または HEAD 以外のリクエストによって 301，または 307 が返ってきた場合は，結果フレーズをポートレットおよびログに出力します。※1
4xx：Client Error	401（Unauthorized）および 407（Proxy Authentication Required）に対応します。※2 401 および 407 以外は，結果フレーズをポートレットおよびログに出力します。
5xx：Server Error	結果フレーズをポートレットおよびログに出力します。

注※1　Web App Portlet では，POST 後に 302 が返却された場合，ユーザに確認しないでリダイレクトします。なお，一般的な Web ブラウザと同様に POST 後の 302 レスポンスコードに対応します。
注※2　Web Page Portlet は対応していません。

対応するレスポンスヘッダフィールドの取り扱いを次の表に示します。

表 5-7 レスポンスヘッダフィールドの取り扱い（Web App Portlet および Web Page Portlet）

ヘッダフィールド	内容
Cache-Control	キャッシュ制御に使用します。キャッシュ制御の指定は，Portal Manager の Web ポートレット設定時に定義できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。
Content-Type	コンテンツのエンコーディング判定に使用します。text/html だけに対応します。
Last-Modified	コンテンツの最終更新時刻です。キャッシュ制御に使用します。
Location	リダイレクト先の URL です。リダイレクト時に使用します。

ヘッダフィールド	内容
Set-Cookie	指定された Cookie 値をサーバ内で保持します。Web Page Portlet は対応していません。
Set-Cookie2	指定された Cookie 値をサーバ内で保持します。Web Page Portlet は対応していません。
WWW-Authenticate	ベーシック認証を試みます。Web Page Portlet は対応していません。
Proxy-Authenticate	プロキシ認証を試みます。Web Page Portlet は対応していません。

クライアントからのリクエストヘッダフィールドの取り扱いを次の表に示します。

表 5-8 リクエストヘッダフィールドの取り扱い（Web App Portlet および Web Page Portlet）

ヘッダフィールド	内容
Host	リクエスト先のホスト名を設定します。
Accept	対応する MIME タイプを設定します。
Accept-Encoding	対応するエンコーディングを設定します。
Accept-Language	対応する言語を設定します。
User-Agent	リクエストした User-Agent を設定します。Web App Portlet では，User-Agent の内容を uCosminexus Portal Framework 独自の User-Agent の内容に設定できます。この設定は，Portal Manager の Web ポートレット設定時に定義できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。
If-Modified-Since	コンテンツの最終更新時刻を設定します。
Cookie	Set-Cookie1 でサーバに保存された Cookie を設定します。Web Page Portlet は対応していません。
Cookie2	Set-Cookie2 でサーバに保存された Cookie を設定します。Web Page Portlet は対応していません。
Authorization	ベーシック認証に必要な認証情報を設定します。Web Page Portlet は対応していません。
Proxy-Authorization	プロキシ認証に必要な認証情報を設定します。Web Page Portlet は対応していません。

(b) エンコーディング

HTML コンテンツ中に日本語を記述する場合には，通常の HTML コンテンツ同様，エンコーディング方法を指定する必要があります。エンコーディング方法の決定は，外部の Web サイトに格納された HTML コンテンツと uCosminexus Portal Framework 内に配置した HTML コンテンツで異なります。

uCosminexus Portal Framework では，ユーザの環境による文字化けを防ぐために，ユーザの環境ごとにエンコーディング方法を推奨しています。uCosminexus Portal

Framework が推奨するエンコーディングについては，「2.1.4　エンコーディング」を参照してください。

外部 Web サーバ上の HTML コンテンツのエンコーディング指定

エンコーディング方法は，次の優先順位で決定されます。

1. HTTP レスポンスヘッダの Content-Type フィールドの charset パラメタ
2. META 要素中の Content-Type 指定の charset パラメタ
3. 自動判定

自動判定では，正しいエンコーディング方法を決定できないことがあるので，1.，または 2. でのエンコーディング指定を推奨します。

● ポータル内の HTML コンテンツのエンコーディング指定

エンコーディング方法は，次の優先順位で決定されます。

1. META 要素中の Content-Type 指定の charset パラメタ
2. 自動判定

自動判定では，正しいエンコーディング方法を決定できないことがあるので，META 要素中の Content-Type 指定の charset パラメタでのエンコーディング指定を推奨します。外部 Web サイトの方法と異なり，ポータル内の HTML コンテンツは，HTTP レスポンスヘッダでエンコーディングを指定できません。

(c) プロキシ連携

プロキシを経由してコンテンツを取得できます。プロキシと連携するには，Portal Manager の Web ポートレット設定時に定義できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(d) リダイレクト

Web App Portlet レスポンスコードの 301，302，303，307 に対応します。Web App Portlet および Web Page Portlet では，Portal Manager の Web ポートレット設定時に最大リダイレクト回数を設定できます。0 から 10 の値を指定できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(2) 認証

(a) 認証に対するシングルサインオン

PC からポータルにアクセスしている場合，ディレクトリサーバに格納されているユーザ情報（パーソナライズ情報）から認証情報を取得することによって，Web アプリケー

ションとのシームレスな連携ができます。なお，Web Page Portlet では，シングルサインオンはできません。

認証機能が利用できる前提条件を次に示します。

- ポートレットに表示しているアプリケーションへの，ログイン前のページの認証
  認証先が参照のチェックをするアプリケーションの場合
- JavaScript を使用している Web サイトの認証
  JavaScript で認証フィールドを動的に変更および生成している場合

● ベーシック認証

Web サーバのレスポンスコードが 401（Unauthorized）の際，次の処理をします。

- ユーザログイン時
  対象 URL に対応するユーザ ID，およびパスワードをリポジトリから取得して自動的に認証します。この機能は，ユーザがユーザ ID とパスワードを登録すれば使用できるようになります。登録されていない場合は，ユーザ ID とパスワードを登録するフォームを出力します。なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。
- ウェルカム画面（ユーザがログインしていない場合）
  ウェルカム画面での Web サーバに対応する認証はしません。

● プロキシ認証

プロキシサーバのレスポンスコードが 407（Proxy Authentication Required）の際，次の処理をします。

- ユーザログイン時
  ユーザがログインすると，ユーザ情報取得 Bean を使用して，リポジトリから対象となるプロキシサーバに対応するユーザ ID およびパスワードを取得して，自動的に認証します。該当するユーザ ID，およびパスワードが登録されていない場合は，ユーザが入力できるようにフォームを出力します。なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。
- ウェルカム画面（ユーザがログインしていない場合）
  ウェルカム画面でのプロキシサーバに対応する認証はしません。

● フォーム認証

Web サーバへのリクエスト情報を基にシングルサインオンを実現します。Web App Portlet の場合，シングルサインオン後は，認証の成功・失敗にかかわらず「ログイン失敗」のリンクが表示されます。認証するサイトによって，認証画面を通過する必要がある場合，シングルサインオンはできません。パスワードの入力ミスなどによってフォーム認証に失敗した場合は，認証情報をクリアして，再度認証情報を入力する必要があります。

Web App Portlet の場合は，「ログイン失敗」のリンクをクリックして表示される画面に

従って，認証情報を入力します。

フォーム認証をするためには，次に示す項目をあらかじめ設定しておくことが必要です。

- 認証するページの URL
- ログインするページの URL
- 認証時に使用するメソッド
- 認証情報を示すフィールド

なお，管理者があらかじめユーザ ID とパスワードを登録することはできません。

Web App Portlet は，フォーム認証定義ファイルで，フォーム認証情報を設定します。Web App Portlet を使用する場合のフォーム認証の設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「フォーム認証定義ファイル（formbase-auth.cfg）」の説明を参照してください。

(3) URL 変換

Web ポートレットでは，ポータル画面内で画面遷移するために，Web ページに含まれている相対パス形式および絶対パス形式の URL を，uCosminexus Portal Framework を経由する URL に変換します。BASE 要素で基準となる URL が指定されている場合，BASE 要素を基に自動的に変換します。

HTML コンテンツ中に含まれる画像などのインラインオブジェクトは，通常 uCosminexus Portal Framework を経由しないで，Web ブラウザが直接 Web サーバから取得します。Web App Portlet では，インラインオブジェクトの取得先をポートレット登録時に Portal Manager で設定できます。プロキシやコンテンツを格納する Web サーバがアクセス制御している場合，および各種認証に対してシングルサインオンをする場合は，Web App Portlet ではインラインオブジェクトの取得先を［ポータルサーバを経由して取得］に設定してください。

インラインオブジェクトの取得先の設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(4) キャッシュ制御

Web ポートレットでは，uCosminexus Portal Framework 内で HTML コンテンツをキャッシングします。キャッシングは，HTTP1.1 と同機能に基づいています。キャッシュ範囲はレスポンスのボディだけです。インラインオブジェクトは，キャッシュ対象となりません。

Web App Portlet では，ポートレットごとにキャッシュ制御の設定ができます。キャッシュ制御の設定は，Web ポートレットの登録時に Portal Manager で，［有効］または［無効］を設定できます。デフォルトでは，［有効］に設定されています。キャッシュ制御の設定方法については，マニュアル「uCosminexus Portal Framework　システム管

理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

次の場合は，キャッシュ制御を無効にしてください。サーバからの Cache-Control フィールドを無視して強制的にコンテンツを取得します。

- ベーシック認証，プロキシ認証，フォーム認証のどれかを使用する場合
- 動的コンテンツを参照する場合
- 頻繁に内容が変更されるコンテンツを利用する場合
- ユーザごとに異なるコンテンツを利用する場合

(a) コンテンツの有効期限

コンテンツの有効期限を基にキャッシュ検証されます。コンテンツの有効期限は，Portal Manager の Web ポートレット設定時に指定できます。詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

コンテンツの有効期限は，次の優先順位で決定します。

1. HTTP レスポンスヘッダの Cache-Control フィールド指定
   ローカルコンテンツの場合は指定できません。
2. META 要素中の Cache-Control 指定
3. ポートレット定義ファイルの WebPage エントリで指定されるデフォルト動作指定

(b) キャッシュ容量

キャッシュには，メモリキャッシュとディスクキャッシュがあり，それぞれ格納するコンテンツの最大数がプロパティファイルで設定できます。プロパティファイル（PortalResources.properties）の格納場所を次に示します。

```
{PROJECT_HOME}¥WEB-INF¥conf
```

プロパティファイル（PortalResources.properties）の詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortalResources.properties の詳細」の説明を参照してください。

キャッシュ容量の設定を次の表に示します。

表 5-9　キャッシュ容量の設定

属性	内容	デフォルト値
jp.co.hitachi.soft.portal.webpage.cache.directory	ディスクキャッシュの格納ディレクトリ	{PROJECT_HOME}/WEB-INF/cosmi/webpage/cache
jp.co.hitachi.soft.portal.webpage.cache.memMaxSize	メモリキャッシュに格納する最大コンテンツ数	16
jp.co.hitachi.soft.portal.webpage.cache.diskMaxSize	ディスクキャッシュに格納する最大コンテンツ数	128

メモリキャッシュおよびディスクキャッシュは，ご使用の環境に合わせた妥当値を設定してください。

(c) キャッシュクリア

キャッシュクリアでは，キャッシュ機能を使用時に登録済みのコンテンツを更新した場合，画面表示を最新にします。キャッシュクリアは，すべてのキャッシュエントリに対して実行されます。ポートレットごとのキャッシュ検証はできません。キャッシュクリアの通知は Portal Manager の［キャッシュクリアの実行］で設定します。

キャッシュクリアは，Web ポートレットにキャッシュの強制検証を通知します。キャッシュクリアはローカルホストだけに有効です。クラスタ構成の場合はホストごとにキャッシュクリアを実行する必要があります。

キャッシュクリアは，Portal Manager，または強制検証コマンドで実行します。

Portal Manager を使用してキャッシュをクリアする手順については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「キャッシュクリアの実行」の説明を参照してください。

強制検証コマンドの書式を次に示します。

```
cpfwvali.bat　プロジェクト名称
```

なお，強制検証コマンドを使用するためには次の環境変数の設定を行ってください。

サーブレットエンジンモードの場合

変数名	値
COSMI_PORTAL_HOME	{Cosminexus インストールディレクトリ}¥CC¥web¥containers¥{サーバ名称}¥webapps

J2EE モードの場合

変数名	値
COSMI_PORTAL_HOME	{Cosminexus インストールディレクトリ}¥CC¥server¥Public¥Web¥{サーバ名称}

(5) 要素フィルタリング

Web App Portlet および Web Page Portlet は，指定した Web ページをポータル画面に取り込む際，BODY 本体内に記述できない開始，終了タグや要素が取り除かれます。要素フィルタリングは，Web ポートレットの登録時に，フィルタリングするかしないかを設定できます。タグ，要素は取り除かれますが，コンテンツを表示するために必要な BASE 要素やエンコーディング設定，およびキャッシュ制御に必要な META 要素の一部はコンテンツ作成のために参照されます。

Web App Portlet および Web Page Portlet のフィルタリングは Portal Manager の

［ポートレットの設定］画面で設定します。要素フィルタリングの設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

要素フィルタリングで取り除かれるタグ，要素と設定を次の表に示します。

表 5-10　要素フィルタリングで取り除かれるタグ，要素の一覧

取り除かれるタグ，要素	フィルタリングの設定
DOCTYPE 宣言	×
HTML 開始，終了タグ	×
HEAD 要素	○
BODY 開始，終了タグ	×
FRAMESET 要素 （NOFRAMES 要素の本体は除く）	×
APPLET 要素	○
OBJECT 要素	○
SCRIPT 要素（対応する NOSCRIPT 要素の本体は除く）	○
on で始まるイベント属性	○

（凡例）
○：設定できます。
×：設定できません。

(6) コンテンツフィルタリング

コンテンツフィルタリングとは，HTML ファイルを Web ポートレットとして取り込む際に，コンテンツを抽出，または削除することです。コンテンツフィルタリングを使用すると，HTML ファイルの一部分をポートレットに取り込めます。コンテンツフィルタリングは，Web ポートレットの登録時に Portal Manager の［ポートレットの設定］画面で設定できます。コンテンツフィルタリングの設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

コンテンツフィルタリングでは，コンテンツの BODY 要素本体に含まれるディレクティブに従ってコンテンツを抽出，または削除します。ディレクティブも同時に抽出・削除します。

ディレクティブは，コメントまたは正規表現で指定できます。

なお，コンテンツフィルタリングの設定は，Web ポートレットの種類によって異なります。対応するフィルタリングの設定を次に示します。

表 5-11　コンテンツフィルタリングの設定（Web App Portlet）

フィルタリング方式	正規表現	コメント
削除	○	○
抽出	○	○

（凡例）
○：設定できます。
×：設定できません。

表 5-12　コンテンツフィルタリングの設定（Web Page Portlet）

フィルタリング方式	正規表現	コメント
削除	×	○
抽出	×	×

（凡例）
○：設定できます。
×：設定できません。

コメントでのディレクティブ指定

<!--　　--> で囲まれたコメント内にあるキーワードでフィルタリング範囲を指定します。BODY 本体の既存のコメント，または新規に追加したコメントをディレクティブとして使用できます。要素フィルタリングで取り除かれる要素内にあるディレクティブは，使用できません。

フィルタリング範囲は，開始キーワードを含むコメントから，それ以降で最初に対応する終了キーワードを含むコメントまでです。キーワードはポートレットごとに指定できます。

開始キーワードと終了キーワードを一組で指定します。コメント中に含まれる開始キーワードと終了キーワードは，コンマで区切って指定します。

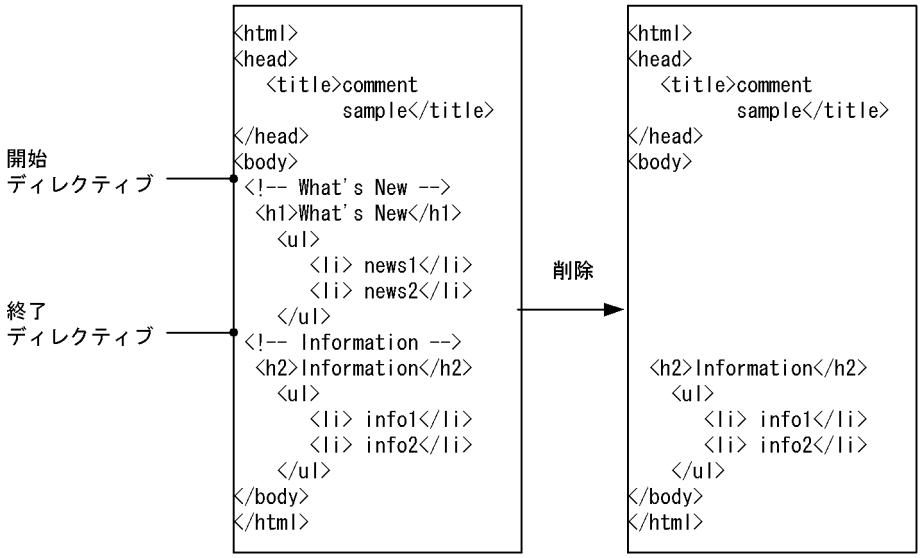
同一コメント中に開始キーワードが複数ある場合は，先に合致したキーワードを使用します。キーワードとして「,」，「"」，および「|」は使用できません。

ディレクティブを複数指定できます。フィルタリング範囲が重複しないようにキーワードを設定します。

開始ディレクティブに対応する有効な終了ディレクティブがない場合は，開始ディレクティブ以降のコンテンツすべてがフィルタリング対象となります。

ディレクティブとして，開始キーワードに "What's New" を，終了キーワードに "Information" を指定したフィルタリング削除の例を次の図に示します。

図 5-3 コメントでのフィルタリング削除



正規表現でのディレクティブ指定

HTML ファイルの文字列をパターンとして指定します。パターンは Web ポートレットごとに指定できます。

パターンを指定できる範囲を次に示します。

- 先頭からパターンまで
- パターンからパターンまで
- パターンから終了まで

指定するパターンは大文字小文字を区別します。また，パターンには「,」および「|」は指定できません。HTML ファイルの文字列が，正規表現と合致しない場合，コンテンツフィルタリングをしません。

パターンからパターンまでの範囲は，複数指定できます。複数のフィルタリング範囲を指定する場合，フィルタリング範囲が重複しないようにパターンを指定してください。パターンからパターンまでの範囲では，パターン 1 およびパターン 2 を指定します。パターン 2 には，パターン 1 よりファイル内で後ろに記述されている文字列を指定します。

(7) 外部サーバとのセッション維持

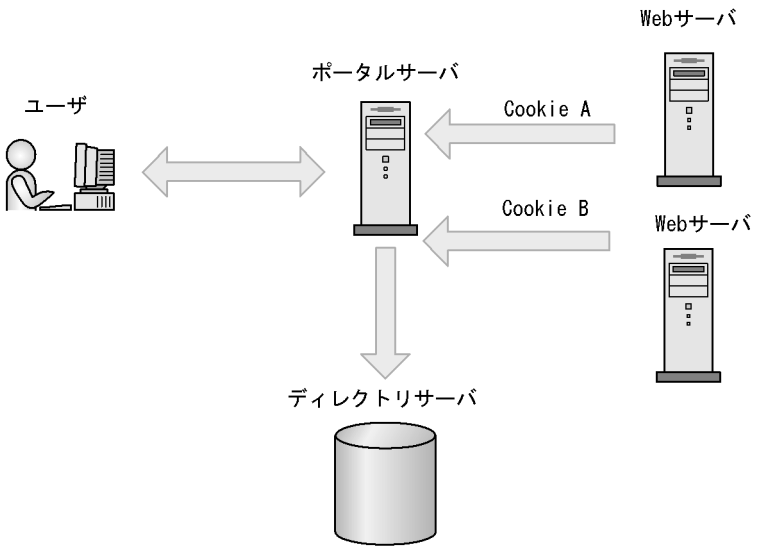
Web App Portlet では，外部 Web サーバとのセッションを維持できます。Web Page Portlet では外部サーバとセッション維持できません。セッション維持の方法を次に示します。

- Cookie によるセッション維持
- URLRewriting によるセッション維持

(a) Cookie によるセッション維持

外部 Web サーバからのレスポンスから Cookie を取得します。Web ブラウザには Cookie を渡さないで，ポータルで保持します。ユーザがポータルにログインしており，かつ有効期限が永続的な Cookie は，セッションが終了するとディレクトリサーバに保存されます。Cookie によるセッション維持を次の図に示します。

図 5-4 Cookie によるセッション維持（Web App Portlet）



(b) URLRewriting によるセッション維持

URL を読み込んだあとに URL を書き換えて，セッションを維持します。

(8) その他の注意事項

(a) ネームスペース

Web App Portlet および Web Page Portlet では，ポータル画面は複数のポートレットで構成されているためポートレット間で名称が重複する場合があります。そのため，次に示す属性はポートレット間で異なる値を指定してください。

- accesskey 属性（要素のアクセスキー）
- class 属性（要素のクラス名）
- id 属性（要素の識別子）

また，Web App Portlet では，Name 属性名にポートレット名称を自動的に追加して，異なる名称に変更できます。スクリプトを使用する場合は，Web ポートレット登録時に Portal Manager の［ポートレットの設定］画面でネームスペース自動解決機能を［無効］に設定します。ネームスペース自動解決機能の設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設

定」の説明を参照してください。

ポートレットから POST による外部 Web サーバへのリクエスト時には，属性名からポートレット名称を削除します。

(b) スタイル

ポータルの画面イメージはスタイルシートで決定します。そのため，特別に強調する場合以外で，インラインスタイルなどの個別のスタイルを指定しないでください。ポータル画面のスタイルやフォントを変更するには，次のスタイルシートを変更してください。

```
{PROJECT_HOME}¥css¥default.css
```

(c) FRAMESET

Web App Portlet および Web Page Portlet では，FRAMESET を使用した HTML コンテンツは，表示できません。

FRAMESET を使用した外部の Web サイトを取り込んだ場合，NOFRAMES の内容を表示します。NOFRAMES がない場合は何も表示しません。

(d) スクリプトを使用するには

スクリプトを使用するには，Web ポートレットの登録の際に次の設定をしてください。

- スクリプトの関数やグローバル変数の定義は異なる名称を指定します。
- SCRIPT 要素および on で始まるイベント属性の削除オプションを［削除しない］にします。
- デフォルトでは，SCRIPT 要素および on で始まるイベント属性は削除されます。
- ネームスペース自動解決機能を［無効］にします。
- HEAD 要素内にスクリプトを記述する場合は，HEAD 要素の削除オプションを［削除しない］にします。

(e) インラインオブジェクトとの連携

HTML 中に組み込まれるオブジェクトと Web ポートレットは連携できません。

(f) 最大化表示

最大化表示は，SCRIPT を使用している場合は設定しないでください。

(g) プロキシサーバ使用時の注意事項

Web App Portlet を Portal Manager で登録する際に指定するプロキシサーバは，Web App Portlet が使用するすべてのページを参照できるプロキシサーバを指定してください。プロキシサーバの指定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

使用するマシン環境によっては，インターネット向けプロキシサーバとイントラネット

向けプロキシサーバを区別している場合があります。この場合，インターネットのコンテンツとイントラネットのコンテンツを Web App Portlet で統合できません。

(h) Web App Portlet に取り込む Web コンテンツ

Web App Portlet に取り込む Web コンテンツで，HTML4.01 に従わないタグ（
 など）がある場合正しく表示されないことがあります。そのため，Web App Portlet に取り込む Web コンテンツには，HTML4.01 に従わないタグを使用しないでください。
HTML4.01 に従わないタグを使用した場合，uCosminexus Portal Framework は次に示す例のようにタグを変換します。

（例）

 のように "/" で終わるタグは，
> のように "/" を ">" に変換します。

(i) Web App Portlet を使用する場合の URL 形式

Web App Portlet を使用する場合，ポートレット定義の URL 形式やコンテンツの URL 形式が次に示す二つの条件に合うとき，コンテンツは正しく表示されません。

- URL が http:// で始まりディレクトリを指している
- URL の末尾にクエリが与えられている

（例）

```
http://hostname?querya=aaa&queryb=bbb
```

この問題は，次の例のようにディレクトリのウェルカムメッセージに相当するコンテンツのファイル名を指定することで回避できます。

（例）

```
http://hostname/index.htm?querya=aaa&queryb=bbb
```

(j) Web App Portlet でのインラインオブジェクトの取得先

Web App Portlet を使用する場合，Portal Manager でインラインオブジェクトの取得方法を［ポータルサーバを経由して取得］に設定したとき，インラインオブジェクトを取得できないことがあります。この場合，Portal Manager でインラインオブジェクトの取得方法を［web サーバから直接取得］に変更することで取得できるようになります。インラインオブジェクトの取得先の設定方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

(k) Web App Portlet および Web Page Portlet でのコンテンツ不正表示

Web App Portlet または Web Page Portlet を使用している場合に，相手先コンテンツから「<TABLE> タグと <FORM> タグの対応が誤っているコンテンツ」が返却されたとき，Web App Portlet または Web Page Portlet のコンテンツは正しく表示されません。

この問題は，コンテンツの <TABLE> タグと <FORM> タグの対応を正しく修正することで

回避できます。

5.5 デバイスに対応するための設定（Multi Web Portlet）

Multi Web Portlet では，PC だけでなく携帯電話（ｉ モード，および EZweb）からアクセスできるコンテンツを作成します。コンテンツは，クライアントのデバイス（PC，ｉモード，または EZweb）ごとに対応する言語で作成する必要があります。ただし，PC の場合は，iframe タグをサポートした Web ブラウザ（Internet Explorer，Firefox，または Netscape Navigator）で表示されます。

ポータルにアクセスできるクライアントのデバイスの種類と，対応する言語を次の表に示します。

表 5-13　クライアントのデバイスの種類と対応する言語（Multi Web Portlet）

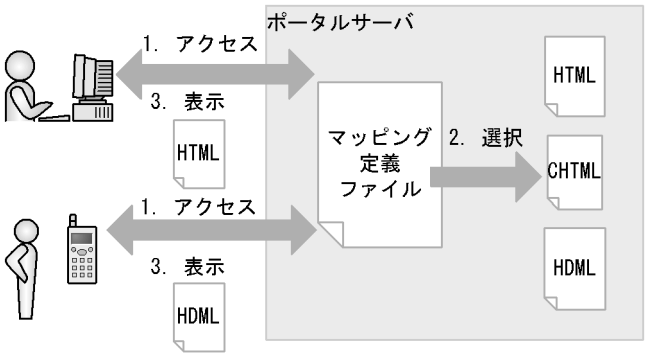
クライアントデバイス	対応する言語
HTML4.01 に対応する Web ブラウザのある PC	HTML
i モード対応 HTML2.0 に対応する i モードの携帯電話	CHTML
Version 3.3 for HDML に対応する EZweb の携帯電話	HDML

クライアントのデバイスに合ったコンテンツ表示するために，各デバイスに対応した URL を設定する必要があります。この設定は，ポートレットを登録する際に Portal Manager で行えます。Multi Web Portlet の登録については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「Web ポートレットの設定」の説明を参照してください。

アクセスしてきたクライアントのデバイスの種別を判別して，クライアントに合ったコンテンツを選択するために，マッピング定義ファイルを使用します。一つのポートレットに一つのマッピング定義ファイルが必要です。

マッピング定義ファイルとクライアント判別について次の図に示します。

図 5-5 マッピング定義ファイルとクライアント判別（Multi Web Portlet）



図の説明

1. ユーザがポータルにアクセスしたときに，マッピング定義ファイルでユーザのクライアント種別を判別します。
2. クライアント種別に合致したファイルを選択します。
3. 選択したファイルをポータルで表示します。

マッピング定義ファイルは，ポートレット登録時に作成されます。ポートレット登録時に，マッピング定義ファイルの作成場所を指定します。ポートレットごとに作成したポートレットディレクトリの直下を指定します。

6 分散ポートレットの作成

この章では，分散ポートレットを作成する際の手順，および注意事項について説明しています。

6.1 分散ポートレットとは

分散ポートレットは，ほかのポータル上に配置されたポートレットを自ポータルに取り込むポートレットです。例えば，他事業所で構築しているポータル上のポートレットを，自ポータルに取り込めます。

6.2 分散ポートレットの作成手順

この節では，分散ポートレットを作成する手順について説明します。

手順

1. **分散ポートレットの機能を決定します。**
   ポータルに取り込む分散ポートレットを決定します（取り込む分散ポートレットを配置している分散サーバの決定）。ただし，取り込めるポートレットの種類は決まっています。取り込めるポートレットの種類に種類については，「6.3.1　制限事項」を参照してください。
2. **分散ポートレットを作成します。**
   ポータルに取り込める分散ポートレットは，uCosminexus Portal Framework が決めた記述方法に従って開発されている必要があります。記述方法については，「6.4　分散ポートレット作成のガイドライン」を参照し，必要に応じて修正してください。
3. **分散ポートレットをポータルに登録します。**
   ポートレットの登録方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレットの登録」の説明を参照してください。

6.3 分散ポートレットを使用するときの注意事項

6.3.1 制限事項

分散ポートレットを使用する場合，次の制限事項があります。

- 分散ポートレットを使用して，標準 API ポートレット，Web ポートレットおよび Web コンテンツポートレットを呼び出すことはできません。

分散ポートレットで表示できる HTML タグの種類には制限があります。表示できるタグの種類は，Web App Portlet と同じです。

6.3.2 運用上の注意

分散ポートレット運用時には次のことに注意してください。

- 分散サーバと分散クライアント間の通信で使用できるプロトコルは "HTTP" だけです。そのため，分散サーバと分散クライアント間のネットワーク環境はセキュアな状態に保ってください。
- 分散ポータル環境での文字化けを防ぐため，ポートレット，コンテンツ，分散サーバ，および分散クライアントのデフォルトのエンコーディングは，すべて統一してください。
- 分散ポートレットを使用している場合に，相手先コンテンツから「<TABLE> タグと <FORM> タグの対応が誤っているコンテンツ」が返却されたとき，分散ポートレットのコンテンツは正しく表示されません。
  この問題は，コンテンツの <TABLE> タグと <FORM> タグの対応を正しく修正することで回避できます。

6.4 分散ポートレット作成のガイドライン

uCosminexus Portal Framework では，次に示す記述方法に従ったポートレットを分散ポートレットとして表示できます。分散ポートレットを作成するときの参考にしてください。

6.4.1 使用できる形式

分散ポートレットで使用できる HTML の形式と，使用できる uCosminexus Portal Framework のクラスライブラリおよび Bean を次に説明します。

(1) 分散ポートレットで使用できる HTML

分散ポートレットでは，次のように記述した HTML を表示できます。

● 画面遷移を記述した HTML

画面遷移を記述した HTML で，使用できる形式を次に示します。

1. ～

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:a href="url" ...>sample</uu:a>
```

上記の代わりに，PortletURI を使用する書き方もできます。PortletURI を使用する書き方を次に示します。

```
<a href="<%= PortletURI.transPortletURI("url")%>">～</a>
```

2. <area shape="..." coords="..." href="..." alt="...">

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:area href="url" alt="sample" ...>
...
</uu:area>
```

上記の代わりに，PortletURI を使用する書き方もできます。PortletURI を使用する書き方を次に示します。

```
<area href="<%= PortletURI.transPortletURI("url")%>">
```

3. <form action="..."> ～ </form>

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:form action="url" ...>
...
</uu:form>
```

上記の代わりに，PortletURI を使用する書き方もできます。PortletURI を使用する書き方を次に示します。

```
<form action= "<%=PortletURI.transPortletURI("url")%>">〜</form>
```

● インラインオブジェクトを表示させる記述をした HTML

インラインオブジェクトを表示させる記述をした HTML で，使用できる形式を次に示します。

1. <iframe src="..."> 〜 </iframe>

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:iframe src="url" longdesc="url" ...>
...
</uu:iframe>
```

2. <img src="...">

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:img src="url" alt="sample" longdesc="url" .../>
```

3. <input src="...">

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:input src="url" ...>
...
</uu:input>
```

● ログイン直後の画面に戻る指定をした HTML

ログイン直後の画面に戻る指定をした HTML を記述できます。

1. 戻るボタンで表示する場合
   戻るボタンで，ログイン直後の画面に戻る指定を HTML に記述できます。

```
<form method="post" action="<portal:uriLookup type="Home" />">
    <input type=submit value="戻　る">
</form>
```

2. アンカータグで表示する場合：
   アンカータグで，ログイン直後の画面に戻る指定を HTML に記述できます。

```
<a href="<portal:uriLookup type="Home" />">
    Homeに戻ります。
</a>
```

(2) 分散ポートレットで使用できるクラスライブラリおよび Bean

uCosminexus Portal Framework で提供するクラスライブラリおよび Bean のうち，分散ポートレットで使用できるものを次に示します。

クラスライブラリ

- Java のクラスライブラリ
- サーブレットのクラスライブラリ
- jp.co.hitachi.soft.portal.portlet.PortletURI
- jp.co.hitachi.soft.portal.portlet.PortletUtils

- jp.co.hitachi.soft.portal.portlet.PortletException

jp.co.hitachi.soft.portal.portlet.PortletURI，
jp.co.hitachi.soft.portal.portlet.PortletUtils，および
jp.co.hitachi.soft.portal.portlet.PortletException の詳細は，「14.4　ポートレットユティリティクラスライブラリ」を参照してください。

Bean

- ポートレット情報取得 Bean
- ユーザ情報取得 Bean
- ログ出力 Bean ※

注※　ログ出力 Bean は分散サーバのログに出力します。

ポートレット情報取得 Bean の詳細は，「14.5　ポートレット情報取得 Bean」を，ユーザ情報取得 Bean の詳細は，「14.6　ユーザ情報取得 Bean」を，ログ出力 Bean の詳細は，「14.7　ログ出力 Bean」を参照してください。

6.4.2　使用できない形式

分散ポートレットで使用できない HTML の形式と，使用できない uCosminexus Portal Framework の Bean を次に説明します。

(1) 分散ポートレットで使用できない HTML

分散ポートレットでは，次の HTML 要素は使用できません。

- <script> ～ </script>
- <applet code="..."> ～ </applet>
- <object> ～ </object>
- <frameset> ～ <frame src="..."> ～ </frameset>

また，次に示す制限事項があります。

- 次のタグ内で target 要素を指定して，別ウィンドウを表示することはできません。
  - <a> ～
  - <area> ～ </area>
  - <form> ～ </form>
- ポートレットのナビゲーションテンプレートを表示する URL も記述できません。
- iframe タグ (<iframe src="..."> ～ </iframe>) などを使用して，コンテンツ自身を表示することはできません。

(2) 分散ポートレットで使用できない Bean

uCosminexus Portal Framework で提供する Bean のうち，分散ポートレットで使用で

きない Bean を次に示します。

- ユーザ管理メッセージ表示タグライブラリ
- カスタマイズ Bean
- ポートレットテンプレート Bean
- ログインログアウト Bean

これらのタグライブラリ，および Bean の詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「カスタマイズに使用するライブラリ」の説明を参照してください。

7 シングルサインオン対応ポートレットの開発

この章では，シングルサインオン機能に対応したポートレットの開発方法について説明します。

7.1 シングルサインオンとは

シングルサインオンは，Cosminexus の統合ユーザ管理フレームワークの機能を利用してポートレットを開発することで実現します。シングルサインオンの設定および使用方法については，マニュアル「Cosminexus　アプリケーションサーバ V8　機能解説　拡張編」，またはマニュアル「Cosminexus V9　アプリケーションサーバ　機能解説　拡張編」の「シングルサインオンの利用方法」の説明を参照してください。

7.2　バックエンドシステムとの連携

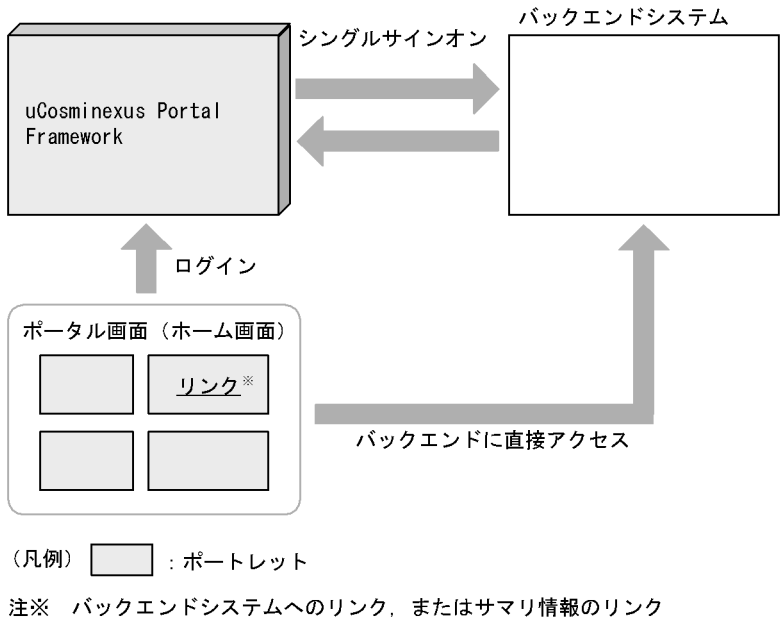
uCosminexus Portal Framework とバックエンドシステムの連携について説明します。ポータル画面とバックエンドシステムとの連携方式には，疎連携と密連携があります。

ここでは，疎連携および密連携について説明します。

7.2.1　疎連携

疎連携では，uCosminexus Portal Framework にログインしたあと，バックエンドシステムにシングルサインオンして，ポータル画面上のポートレットにバックエンドシステムへのリンク，またはサマリ情報のリンクを表示します。疎連携の場合，バックエンドシステムにシングルサインオンしたあとは，ポータル画面上のポートレットに表示されたバックエンドシステムへのリンク，またはサマリ情報のリンクをクリックすると，uCosminexus Portal Framework を経由しないでバックエンドシステムに直接アクセスします。

図 7-1　疎連携



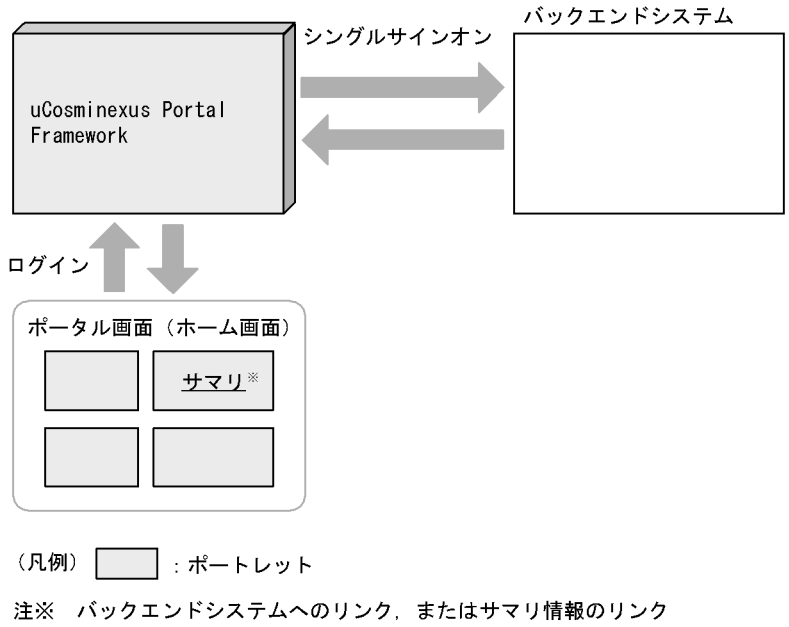
注※　バックエンドシステムへのリンク，またはサマリ情報のリンク

7.2.2　密連携

密連携では，uCosminexus Portal Framework にログインしたあと，バックエンドシステムにシングルサインオンして，ポータル画面上のポートレットにバックエンドシステムへのリンク，またはサマリ情報のリンクを表示します。密連携では，ログイン時にバックエンドシステムにシングルサインオンしたあとでも，バックエンドシステムへの

アクセスはすべて uCosminexus Portal Framework を経由して，バックエンドシステムの情報を取得します。

図 7-2 密連携



注※ バックエンドシステムへのリンク，またはサマリ情報のリンク

シングルサインオン対応ポートレットの開発には，開発工数が比較的小さくてすむ疎連携のサマリ情報を表示する方式を推奨します。密連携では，ポータル画面に対応したビューを作成する開発工数が大きくなります。また，密連携では必ず uCosminexus Portal Framework を経由するため，性能面での考慮が必要となります。

7.3　シングルサインオン対応ポートレットの流れ

疎連携の接続方法には，セッション ID を引き継ぐ方法と，DB 連携の方法があります。

ここでは，セッション ID 引き継ぎ方式と DB 連携方式について説明します。

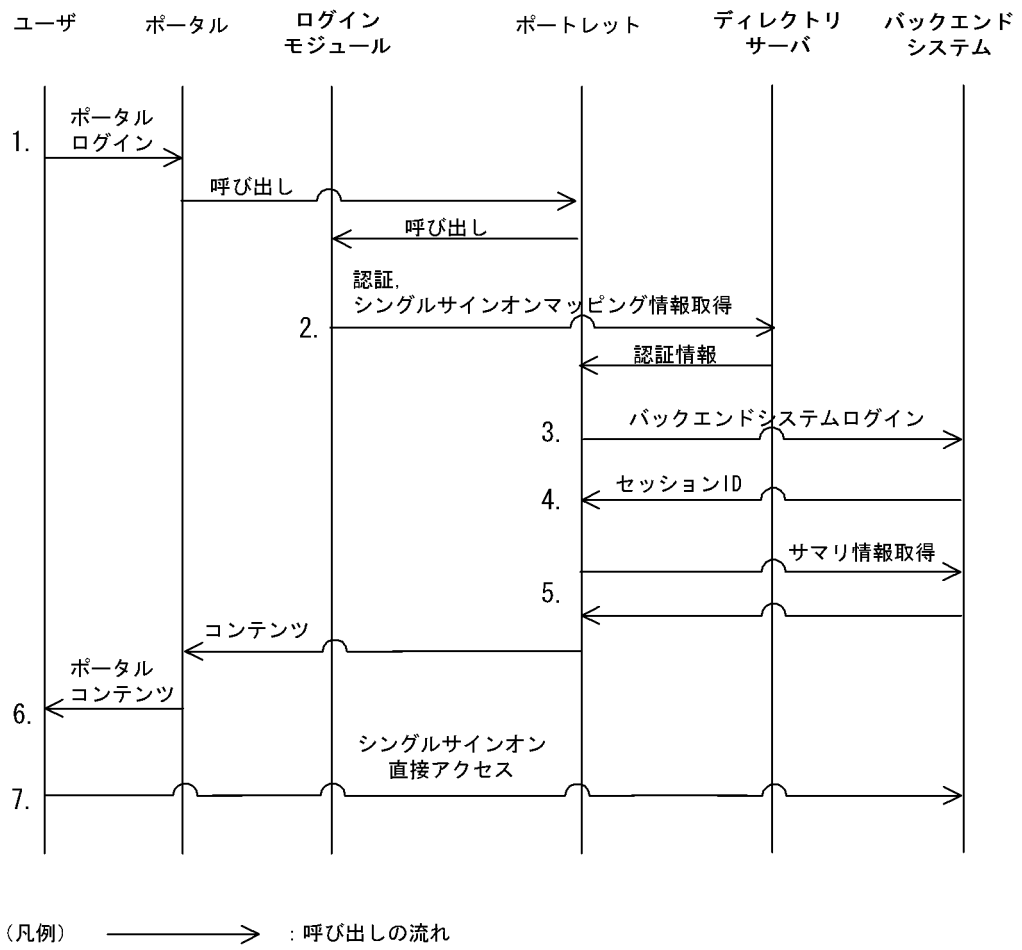
7.3.1　セッション ID 引き継ぎ

セッション ID 引き継ぎの場合，バックエンドが発行するセッション ID を，ポートレット内に記述することでシングルサインオンが実現されます。セッション ID 引き継ぎの場合，ポータルログイン中にバックエンドシステムにもログインします。なお，シングルサインオンを使用するには，バックエンドシステムがセッション ID を利用してセッショントラッキングを実現（Cookie，URL-Rewriting，hidden）している必要があります。

バックエンドシステムの認証方式がフォーム認証の場合だけ接続できます。

シングルサインオン対応ポートレットの接続の流れを次のシーケンスに示します。

図 7-3 シングルサインオン対応ポートレットの流れ（セッション ID 引き継ぎ）



図の説明

1. ポータルへのログインに成功するとポートレットを呼び出します。ポートレットは，シングルサインオン専用のログインモジュールを呼び出します。
2. ログインモジュールでは，ディレクトリサーバのシングルサインオンマッピング情報からバックエンドシステムのユーザ ID とパスワードを取得し，ポートレットに渡します。
3. ポートレットは，ログインモジュールから引き継いだユーザ ID とパスワードを使用して，バックエンドシステムにログインします。
4. バックエンドシステムは，認証に成功するとセッション ID を発行します。
5. ポートレットは，サマリ情報をバックエンドシステムから取得し，コンテンツを生成します。このとき，バックエンドシステムへのハイパーリンクにバックエンドシステムのセッション ID を記述しておきます。

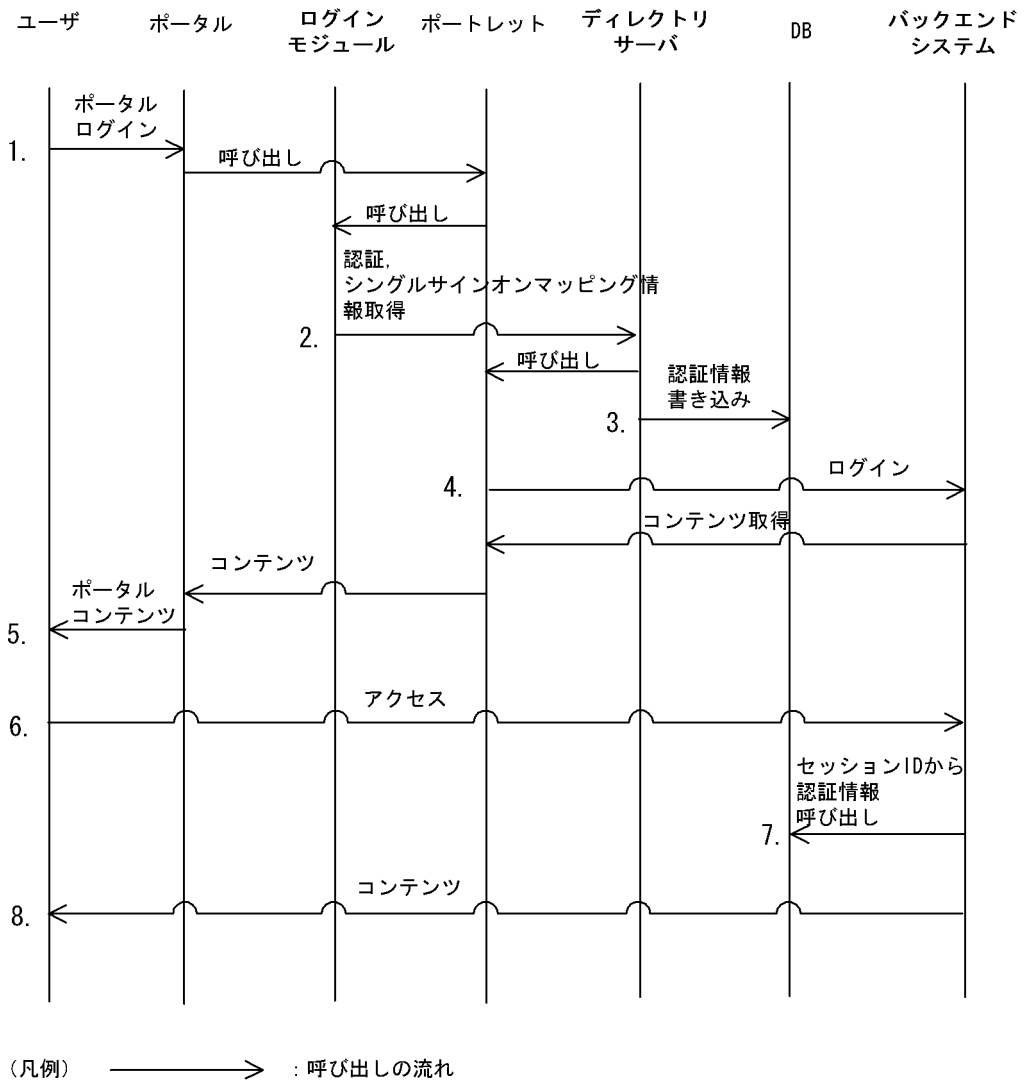
6. ポータルはすべてのポートレットの処理が終わると，コンテンツを合成して Web ブラウザに応答を返します。
7. ポートレットに表示されたハイパーリンクをクリックすると，ポータルを経由しないで直接バックエンドシステムにアクセスします。ハイパーリンクに記載されたセッション ID によって，シングルサインオンが実現できます。

7.3.2 DB 連携

DB 連携の場合，ポータルログイン時に，ポータルのセッション ID などの認証情報，バックエンドシステムのユーザ ID，およびパスワードを対応づけて，DB に格納します。ポートレットに表示されたリンクをクリックしてバックエンドシステムに直接アクセスする場合は，ポータルの認証情報をリクエスト中に含めて，バックエンドシステム側でリクエスト中のポータルの認証情報をキーとして，DB からユーザ ID およびパスワードを取得してシングルサインオンを行います。DB 連携の場合，バックエンドシステムに直接アクセスするたびに，バックエンドシステムへのログイン・ログアウトを行います。

なお，バックエンドシステムの認証方式がフォーム認証，またはベーシック認証の場合に接続できます。

図 7-4 シングルサインオン対応ポートレットの流れ（DB 連携）



図の説明

1. ポータルへのログインに成功するとポートレットを呼び出します。ポートレットは，シングルサインオン専用のログインモジュールを呼び出します。
2. ログインモジュールでは，ディレクトリサーバのシングルサインオンマッピング情報からバックエンドシステムのユーザ ID とパスワードを取得します。
3. ログインモジュールは，2. で取得したポータル認証情報，バックエンドシステムのユーザ ID とパスワードを認証情報 DB に書き込みます。
4. ポートレットは，ログインモジュールから引き継いだユーザ ID とパスワードを使用

して，バックエンドシステムにログインし，コンテンツを取得します。このとき，バックエンドシステムへのハイパーリンクには，ポータルのセッション ID を記述しておきます。

5. ポータルはすべてのポートレットの処理が終わると，コンテンツを合成して Web ブラウザに応答を返します。
6. ポートレットに表示されたハイパーリンクをクリックすると，ポータルを経由しないで直接バックエンドシステムにアクセスします。
7. バックエンドシステムは，URL に記載されたポータルのセッション ID をキーに，認証情報 DB からユーザ ID およびパスワードを取得し，認証を行います。
8. Web ブラウザにコンテンツを生成します。

7.4 バックエンドシステムのポートレット対応

疎連携の場合，バックエンドシステムにシングルサインオン対応ポートレットに対応したビュー，およびログイン・ログアウト処理を作成すると容易に連携できます。

作成する処理は，バックエンドシステムとの接続方法，および認証方法によって異なります。

7.4.1 バックエンドシステムの対応（セッション ID 引き継ぎ）

セッション ID 引き継ぎの場合に作成するページを次に示します。なお，接続方法がセッション ID 引き継ぎの場合，バックエンドシステムとの認証方式はフォーム認証となります。

ログインページ

ユーザ認証を行います。セッショントラッキング用のセッション ID をポートレットに返します。

サマリページ

サマリ画面を出力します。サマリ画面内のバックエンドシステムへのリンクは URL 表記を絶対パスで記述し，リクエスト中にセッション ID を記述します。

7.4.2 バックエンドシステムの対応（DB 連携）

DB 連携の場合に作成するページを認証方式ごとに説明します。なお，DB 連携の場合，バックエンドシステムとの認証方式はフォーム認証・ベーシック認証となります。

(1) フォーム認証

フォーム認証の場合に作成するページを次に示します。

サマリページ

サマリ画面を出力します。サマリ画面内のバックエンドシステムへのリンクは URL 表記を絶対パスで記述し，リクエスト中にポータルのセッション ID などの認証情報を記述します。

ログインページ

リクエスト中に記述しているポータルの認証情報をキーに，バックエンドシステムのユーザ ID，およびパスワードを DB から取得して，ユーザ認証を行います。認証が完了した場合，リクエスト先へフォワード，またはリダイレクトします。

(2) ベーシック認証

ベーシック認証の場合に作成するページを次に示します。

サマリページ

サマリ画面を出力します。サマリ画面内のバックエンドシステムへのリンクは URL 表記を絶対パスで記述し，リクエスト中にポータルのセッション ID などの認証情報を記述します。

HTTP ヘッダ変更モジュール

ベーシック認証の場合，リクエスト中に認証用の Authorization ヘッダを付ける必要があります。ヘッダを付けるため，次のどちらかのモジュールを設置します。

- Web サーバの前段に Authorization ヘッダを付ける Gateway を設置する
- Web サーバに Authorization ヘッダを付ける機能拡張モジュールを追加する

このモジュールでの処理を説明します。
初回リクエスト時は，リクエスト中に記述されているポータルの認証情報をキーに，バックエンドシステムのユーザ ID とパスワードを認証情報 DB から取得し，Authorization ヘッダを付けます。
応答返却時には，ポータルの認証情報を Cookie としてクライアントに設定します。
2 回目以降のリクエストでは，Cookie にある認証情報を参照してバックエンドシステムへシングルサインオンを行います。

7.5 シングルサインオン対応ポートレットの作成

セッション ID 引き継ぎでのシングルサインオンをサンプルとして説明します。このサンプルでは，ログインモジュール，ポートレットの作成方法，バックエンドシステムの修正方法，および設定ファイルの記述方法を説明します。

ログインモジュール
- ログインモジュール（WebLoginModule）
- ログインモジュール（WebPrincipal）

ポートレット
- シングルサインオンポートレット（index.jsp）

設定ファイル
- 統合ユーザ管理フレームワークのコンフィグレーションファイル（ua.conf）
- JAAS のコンフィグレーションファイル（jaas.conf）

バックエンドシステム
- ログインページ（login.jsp）
- ログアウトページ（logout.jsp）
- サマリページ（portlet.jsp）

それぞれのファイルの変更例を次に示します。

7.5.1 ログインモジュール

ログインモジュールは，WebLoginModule と WebPrincipal を作成します。

（1）ログインモジュール（WebLoginModule）

```
package web.login;

import com.cosminexus.admin.auth.*;
import javax.security.auth.*, javax.security.auth.login.*,
javax.security.auth.callback.*, javax.security.auth.spi.*;
import java.io.*, java.util.*, java.net.*;
import java.util.Properties;
import web.WebPrincipal;

public class WebLoginModule implements LoginModule {
    private Subject subject;
    private CallbackHandler callbackHandler;
    private Map sharedState;
    private Map options;

    // 接続先Webアプリパラメータ名(jaas.conf)
    private static final String PORTLET_URL = "web.portlet.url";
    private static final String LOGOUT_URL = "web.logout.url";
```

```
    private static final String LOGIN_URL = "web.login.url";
    private static final String LOGIN_FORM_UID =
"web.login.form.uid";
    private static final String LOGIN_FORM_PASSWD =
"web.login.form.passwd";

    // SSOパラメータ名(ua.conf)
    private static final String USERNAME = "web.login.username";
    private static final String PASSWORD = "web.login.password";

    // エラーメッセージ
   private static final String MSG_USERNOTFOUND = "user not found!";
   private static final String MSG_INVALID_URL = "invalid url option
specifed";
    private static final String MSG_INVALID_LOGININFO = "invalid id/
password/method option specifed";

    private boolean commitSucceeded = false;

    private String sid; // セッションID
    private String username;

    public void initialize(Subject subject, CallbackHandler
callbackHandler,
                           Map sharedState, Map options) {
        this.subject = subject;
        this.callbackHandler = callbackHandler;
        this.sharedState = sharedState;
        this.options = options;
    }

    public boolean login() throws LoginException {
        // ユーザ名とパスワードを取得
        username = (String)sharedState.get(USERNAME);
        if (username == null) throw new
LoginException(MSG_USERNOTFOUND);
        String passwd = (String)this.sharedState.get(PASSWORD);

        URL url;
        String form_uid_name;
        String form_passwd_name;
        String method;

        try {
            url = new URL((String)options.get(LOGIN_URL));
        } catch (Exception e) {
            throw new LoginException(MSG_INVALID_URL);
        }

        form_uid_name = (String)options.get(LOGIN_FORM_UID);
        form_passwd_name = (String)options.get(LOGIN_FORM_PASSWD);
        if (form_uid_name == null || form_passwd_name == null) {
            throw new LoginException(MSG_INVALID_LOGININFO);
        }

        // バックエンドログイン
        try {
            HttpURLConnection conn =
(HttpURLConnection)url.openConnection();
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);
            conn.setRequestProperty("Content-Type",
                                    "application/x-www-form-urlencoded");
```

```
            StringBuffer sb = new StringBuffer();

sb.append(form_uid_name).append("=").append(username).append("&")
              .append(form_passwd_name).append("=").append(passwd)
              .append("&action=login");

            conn.setRequestProperty("Content-Length",
                                    Integer.toString(sb.length()));

            BufferedWriter out =
              new BufferedWriter(new
OutputStreamWriter(conn.getOutputStream()));
            out.write(sb.toString(), 0, sb.length());
            out.flush();

            String cookie = conn.getHeaderField("Set-Cookie");
            sid = cookie.substring(0, cookie.indexOf(";"));
            sid = "jsessionid" + sid.substring(sid.indexOf("="),
sid.length());
        } catch (Exception e) {
            throw new LoginException(e.toString());
        }

        return true;
    }

    public boolean commit() throws LoginException {
        subject.getPrincipals().add( new WebPrincipal(username) );

        Vector tmp = new Vector();
        tmp.add(sid);
        tmp.add(options.get(PORTLET_URL));
        subject.getPublicCredentials().add( tmp );

        return this.commitSucceeded = true;
    }

    public boolean abort() throws LoginException {
        if (commitSucceeded) {
            logout();
        }
        return true;
    }

    public boolean logout() throws LoginException {
        try {
            URL url = new URL(options.get(LOGOUT_URL)+";"+sid);
            BufferedInputStream in =
                new
BufferedInputStream(url.openConnection().getInputStream());
            int BUFLEN = 1024;
            byte[] buf = new byte[BUFLEN];
            while (in.read(buf, 0, BUFLEN) > 0) { }
            in.close();
        } catch (Exception e) {
        }

        commitSucceeded = false;

        return true;
    }
}
```

注意事項

本サンプルでは Cookie の先頭の値をセッション ID（変数名は sid）として引き継いでいます。Cookie の先頭にセッション ID 以外の値が設定される場合や，セッション ID 以外の Cookie（負荷分散のサーバ ID など）を引き継ぐ必要がある場合は，使用する環境に合わせて処理を追加・変更してください。

(2) ログインモジュール（WebPrincipal）

ログインモジュール（WebPrincipal）の作成を次に示します。

```
package web;

import java.security.Principal;
import java.io.Serializable;

public class WebPrincipal implements Principal, Serializable {
  private String name;

  public WebPrincipal(String name) {
    if (name == null) throw new NullPointerException();
    this.name = name;
  }

  public String getName() { return name; }

  public String toString() { return getName(); }

  public boolean equals(Object o) {
    if (o == null) return false;
    if (this == o) return true;
    if (!(o instanceof WebPrincipal)) return false;
    WebPrincipal rhs = (WebPrincipal)o;
    if (getName().equals(rhs.getName())) return true;

    return false;
  }

  public int hashCode() { return getName().hashCode(); }
}
```

7.5.2 ポートレット（index.jsp）

ポートレット（index.jsp）の作成を次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ taglib uri="http://cosminexus.com/admin/auth/uatags"
prefix="ua" %>
<%@ page import="com.cosminexus.admin.auth.*" %>
<%@ page import="com.cosminexus.admin.auth.sso.callback.*" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletUtils" %>
<%@ page import="javax.security.auth.*" %>
<%@ page import="javax.security.auth.login.*" %>
<%@ page import="java.security.*" %>
<%@ page import="java.util.Vector" %>
<%@ page import="java.io.*" %>
<%@ page import="java.net.*" %>
<%!
```

```
  class SessionListener implements HttpSessionBindingListener {
    LoginContext lc;
    SessionListener(LoginContext lc) { this.lc = lc; }
    public void valueBound(HttpSessionBindingEvent evt) { }
    public void valueUnbound(HttpSessionBindingEvent evt) {
      try {
        lc.logout();
      } catch (LoginException e) {
      }
    }
  }

%>
<%
  String sid;  // バックエンドセッションID
  String portletURL; // バックエンドポートレット用URL

  String ns = PortletUtils.getNamespace(request, response);
  Object isBackendLoggedIn = session.getAttribute(ns+"isLoggedIn");

  if (isBackendLoggedIn == null) { // 初回ログイン時(バックエンドにログ
イン)
    LoginContext lc = null;
    try {
      WebSSOHandler h =
        new WebSSOHandler(request, response, null);
      lc = new LoginContext("Web", h);
      lc.login();
    } catch (LoginException e) {
      out.println("Webへのログインに失敗しました。<br>");
      out.println("詳細 : " + e);
      return;
    } catch (Exception e) {
      out.println("Webへのログインで障害が発生しました。<br>");
      out.println("詳細 : " + e);
      return;
    }

    Subject subject = lc.getSubject();
    Principal principal =
(Principal)subject.getPrincipals().iterator().next();
    Vector tmp =
(Vector)subject.getPublicCredentials().iterator().next();
    sid = (String)tmp.get(0);  // セッションID
    portletURL = (String)tmp.get(1); // ポートレットURL

    session.setAttribute(ns+"sid", sid);
    session.setAttribute(ns+"portletURL", portletURL);

    session.setAttribute(ns+"isLoggedIn", new Object());
    session.setAttribute(ns+"logoutCB", new SessionListener(lc));
  } else {
    sid = (String)session.getAttribute(ns+"sid");
    portletURL = (String)session.getAttribute(ns+"portletURL");
  }

  // ポートレット用コンテンツ作成
  try {
    URL url = new URL(portletURL+";"+sid);

    HttpURLConnection conn =
(HttpURLConnection)url.openConnection();
    //conn.setRequestProperty("Cookie", sid);
```

```
    BufferedReader in =
      new BufferedReader(new
InputStreamReader(conn.getInputStream(),
                                                "JISAutoDetect"));
    int BUFLEN = 512;
    char[] buf = new char[BUFLEN];
    while (in.read(buf, 0, BUFLEN) > 0) {
      out.println(buf);
    }

    in.close();
    conn.disconnect();

  } catch (Exception e) {
    out.println("バックエンドアクセス時に障害が発生しました。<br>");
    out.println("詳細 : " + e);
  }
%>
```

7.5.3 設定ファイルの記述例

ポートレットで変更するファイルと記述例を次に示します。

(1) 統合ユーザ管理フレームワークのコンフィグレーションファイル (ua.conf)

ua.confi ファイルに追加する記述を次に示します。

```
com.cosminexus.admin.auth.sso.lm.WebLM=web.login.WebLoginModule
com.cosminexus.admin.auth.sso.param.userid.WebLM=web.login.usernam
e
com.cosminexus.admin.auth.sso.param.secdat.WebLM=web.login.passwor
d
```

なお，統合ユーザ管理フレームワークのコンフィグレーションファイル（ua.conf）は，次に示すディレクトリに格納されています。

格納ディレクトリ

{Cosminexusインストールディレクトリ}¥manager¥config

(2) JAAS のコンフィグレーションファイル（jaas.conf）

jaas.conf ファイルに追加する記述を次に示します。

```
Web {
  com.cosminexus.admin.auth.sso.login.WebSSOLoginModule required
  com.cosminexus.admin.auth.sso="WebLM"
  com.cosminexus.admin.auth.realm="Web"
  web.portlet.url="http://localhost:8080/backend/portlet.jsp" //
ポートレットURL
  web.login.url="http://localhost:8080/backend/login.jsp" // ログイ
ンURL
  web.login.form.uid="uid"         // ユーザID変数名
  web.login.form.passwd="passwd"   // パスワード変数名
```

```
  web.logout.url="http://localhost:8080/backend/logout.jsp" // ログアウトURL
  ;
};
```

なお，JAAS のコンフィグレーションファイル（jaas.conf）は，次に示すディレクトリに格納されています。

格納ディレクトリ
　　{Cosminexusインストールディレクトリ} ¥manager¥config

7.5.4 バックエンドシステム

バックエンドシステムで作成するファイルと記述例を次に示します。

（1） ログインページ（login.jsp）

```
<%@  page session="true" contentType="text/html; charset=Shift_JIS"
          pageEncoding="Shift_JIS" %>
<%
  String uid = request.getParameter("uid");
  // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
  uid = new String(uid.getBytes("iso-8859-1"),"Windows-31J");
  String passwd = request.getParameter("passwd");
  // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
  passwd = new String(passwd.getBytes("iso-8859-1"),"Windows-31J");

  if (ユーザ認証(uid, passwd)) { // ユーザ認証
    // 認証OK
    session.setAttribute("uid", uid);
  }
%>
```

（2） ログアウトページ（logout.jsp）

```
<%@  page session="true" contentType="text/html; charset=Shift_JIS"
          pageEncoding="Shift_JIS" %>
<%
  // バックエンドはURL-Rewritingによって連携
  String url = response.encodeURL(バックエンドのコンテンツの絶対形式URL);
%>

<h2>ポートレット用画面</h2>
<a target="_blank" href="<%= url %>">バックエンドSSOアクセス</a>
```

（3） サマリページ（portlet.jsp）

```
<%@  page session="true" contentType="text/html; charset=Shift_JIS"
          pageEncoding="Shift_JIS" %>
<%
  session.removeAttribute("uid");
%>
```

8 ポートレットイベント対応ポートレットの開発

この章では，ポートレットイベント機能に対応したポートレットの開発方法について説明します。

8.1 ポートレットイベント機能とは

ポートレットイベント機能とは，何らかのイベントが発生した場合に，各ポートレットへ通知する機能です。この機能を利用すると，単にポートレットコンテンツを表示するだけでなく，ポートレットの状態に応じて処理を制御できるため，より柔軟なポータルサイトを提案・構築できます。

また，ポートレットイベント機能には，次の二つの機能があります。

ポートレットアクションイベント機能

クライアント（Web ブラウザ）からポートレットに対して通知する機能です。この機能によって，クライアント側で要求した情報を解析して，処理を実行し，その処理が反映された内容をポートレット上に表示できます。

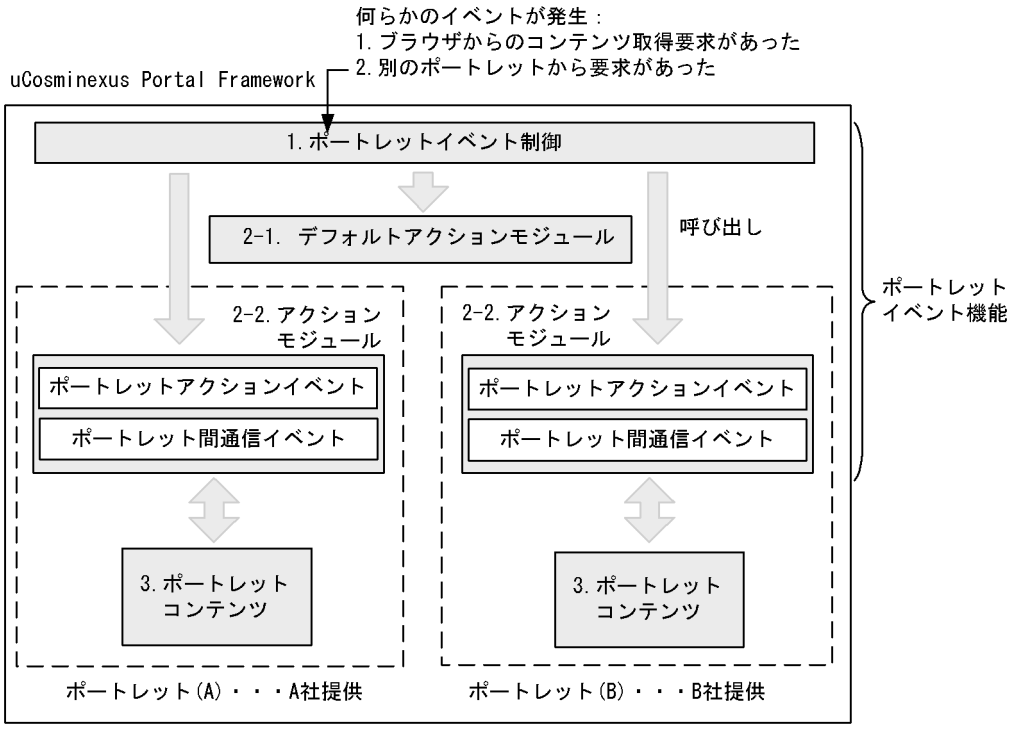
ポートレット間通信イベント機能（ポートレットメッセージ機能）

ポートレット間で情報をやり取りする機能です。情報をやり取りすることで，ポートレット間で情報を共有したり，分散したりできるので，業務効率を向上させるポートレットを開発できます。

8.2　ポートレットイベント機能の処理の流れ

ポートレットイベント機能の処理の流れを次の図に示します。図中の番号は，処理の順序を示します。

図 8-1　ポートレットイベント機能の処理の流れ



図中の項目の概要を，次の表に示します。なお，項番は図中の番号に対応しています。

表 8-1　ポートレットイベント機能の各項目の概要

項番	項目名	概要
1	ポートレットイベント制御	ポートレットイベントを制御する機能です。
2-1	デフォルトアクションモジュール	uCosminexus Portal Framework が標準で添付しているアクションモジュールです。
2-2	アクションモジュール	各ポートレットがイベントの種別に応じて記述するモジュール（クラス）です。 アクションモジュールには，次の二つのイベント（メソッド）があります。 **ポートレット間通信イベント** ポートレット間通信イベント機能で提供するイベント（メソッド）です。 **ポートレットアクションイベント** ポートレットアクションイベント機能で提供するイベント（メソッド）です。

項番	項目名	概要
3	ポートレットコンテンツ	ポートレットコンテンツとは，ポートレット定義で指定した JSP ポートレットや Web ポートレットのことです。最終的に画面に表示される内容になります。

ポートレットイベント制御，デフォルトアクションモジュール，およびアクションモジュールの詳細を，次に説明します。

8.2.1 ポートレットイベント制御

ポートレットイベント制御はすべてのポートレットイベントを制御します。

ポートレットイベント機能は仕様に従ってイベントを制御するため，ポートレット開発者はこの制御を踏まえてイベントの発生を考慮し，処理を記述してください。

ポートレットイベントの仕様を次に示します。

- 次の優先順位で各ポートレットに対してイベントが処理されます。

優先順位	イベント名称
1	ポートレットアクションイベント
2	ポートレット間通信イベント

- 各イベント単位に，イベントがキューイングされます。
- イベントはポートレットに関係なくイベント発生順にキューイングされ，順番に実行されます（優先キューといわれる，イベントを優先的に実行できる機能はありません）。
  ただし，ポートレット並列化機能を使用しているときは，ポートレット単位に並列して実行されます。
  例えば，ポートレット通信イベントのキューにポートレット A，B，C，A の順に 4 個キューングされているとします。この場合，1 番目のポートレット A の処理が完了すれば，ポートレット B，C の処理が完了していなくても，4 番目のポートレット A の処理が実行できます。
- 優先順位が高いイベントが一つでもある場合は，そのイベントが優先的に処理・実行されます。
- すべてのイベント処理はシングルスレッドで実行されます。したがって，異なるイベントが別のスレッドで並行して呼び出されたり，同一のイベントが並行して呼び出されたりすることはありません。
  ただし，ポートレット並列化機能を使用するときはポートレット単位にイベントが実行できます。
- イベントは要求を受けるたびに無条件にキューイングされるため，同一のイベントが一つのポートレットで 2 回以上呼び出されることがあります。

注意事項

- ポートレットの実行順序は規定できないため，画面レイアウトやポートレットの処理速度を考慮してイベントの処理を実装してはなりません。
- uCosminexus Portal Framework に障害が発生した場合，必ずしもすべてのイベント処理が呼び出されるわけではありません。初期処理や終了処理が実行されないおそれがあることを考慮してください。

8.2.2 デフォルトアクションモジュール

デフォルトアクションモジュールは，uCosminexus Portal Framework に標準で添付されているアクションモジュールです。イベント発生時の標準の処理を記述したモジュールで，すべてのポートレットの標準となるイベント処理が記述されています。

各ポートレットによってイベント処理をする場合は，次のデフォルトアクションモジュール（クラス）を継承し，サポートしているイベント（メソッド）を実装します。

デフォルトアクションモジュール（クラス）

```
jp.co.hitachi.soft.portal.portlet.api.DefaultActionModule
```

デフォルトアクションモジュールのイベントの詳細は，「14.9　ポートレットイベント API」を参照してください。

8.2.3 アクションモジュール

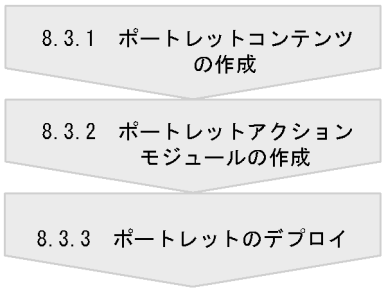
アクションモジュールは各ポートレットに対してイベントが発生した場合，各イベント種別に応じて記述するモジュールです。

アクションモジュールの詳細は，「14.9　ポートレットイベント API」を参照してください。

8.3　ポートレットイベント対応ポートレットの開発手順

ポートレットイベント機能を利用するには，次の手順でポートレットを作成します。

図 8-2　ポートレットイベント対応ポートレットの開発手順



図中の各手順の内容を，次に説明します。

8.3.1　ポートレットコンテンツの作成

ポートレットコンテンツを作成します。

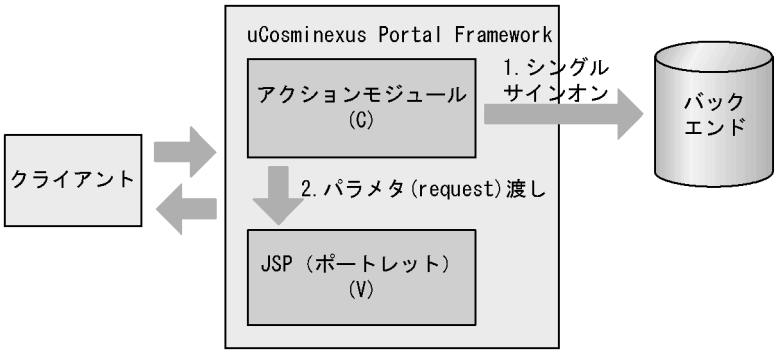
ポートレットアクションモジュールでの処理結果をポートレットコンテンツに対して通知し，画面の表示内容を変更したい場合は，リクエストパラメタに情報を設定します。

8.3.2　ポートレットアクションモジュールの作成

各ポートレットで処理したいイベントがある場合，ポートレットアクションモジュールを作成します。作成したポートレットアクションモジュールを利用すると，MVC モデルに基づいた形式でポートレットを開発できます。MVC モデルの詳細は，「3.11.2　MVC モデル」を参照してください。

ポートレットアクションモジュールを利用して開発すると，次に示すような MVC モデルを形成できます。

図 8-3　アクションモジュールを Controller，JSP を View とする MVC モデル

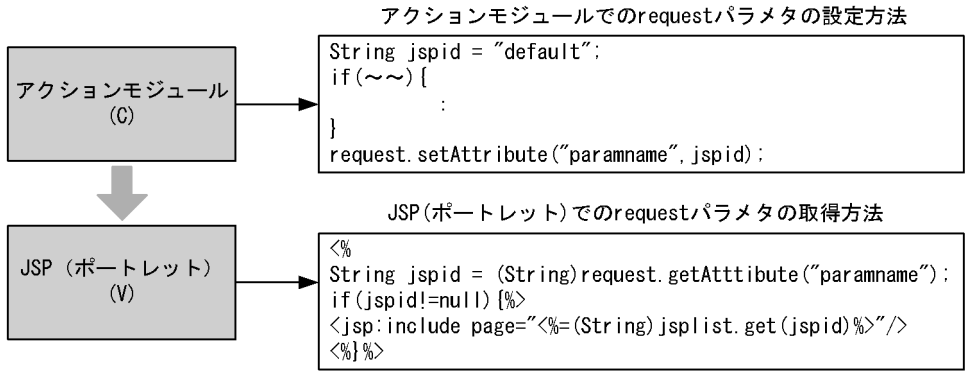


図の説明

1. action メソッドの ServletResponse パラメタによって，アクションモジュールからバックエンドへのシングルサインオンを実現します。
2. アクションモジュールで設定されたパラメタは，JSP（ポートレット）で取得されます。このとき，action メソッドで渡される request パラメタの属性が使用されます。アクションモジュールでは，request パラメタに名前と値を組にして属性を設定します。JSP では，request パラメタの属性から名前をキーにして値を取得し，呼び出す JSP を include で切り替えます。JSP 側では，request パラメタの属性をすべてのポートレットが使用するため，一意の名前を指定してください。

アクションモジュールでのパラメタの設定方法および JSP でのパラメタの取得方法を次に示します。

図 8-4　アクションモジュールでの request パラメタの設定方法および JSP での request パラメタの取得方法



アクションモジュールでの request パラメタの設定方法

action メソッドでは，名前（"paramname"）を値（"default"）を組にして，request パラメタに属性として setAttribute メソッドで設定します。

JSP（ポートレット）での request パラメタの取得方法

JSP（ポートレット）では，名前（"paramname"）をキーに getAttribute メソッドを使用して，request パラメタの属性の値を取得します。取得した属性の値（"default"）を基に，呼び出す JSP（ポートレット）を include で切り替えます。

アクションモジュールの詳細は，「14.9　ポートレットイベント API」を参照してください。

作成したポートレットアクションモジュールは，所定の場所に格納してください。

8.3.3　ポートレットのデプロイ

ポートレットをデプロイする際に，アクションモジュールを指定します。アクションモジュールは，デプロイ定義ファイル，または Portal Manager の［ポートレットの設定］画面の［その他の項目］で config-param 要素に指定します。

config-param 要素の子要素である param-name および param-value に次の値を設定します。

- param-name に設定する内容
  hptl.module.action
- param-value に設定する内容
  アクションモジュールのパッケージ名称を含めたクラスのフルパス名称を指定します。

デプロイ定義ファイルの設定例を次に示します。Portal Manager での設定方法は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「日立 API ポートレット（PAR 形式以外）および File ポートレットの設定」の説明を参照してください。ポートレットのデプロイ方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「日立 API ポートレット（PAR 形式）のデプロイ」の説明を参照してください。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<portlet-app>
  <portlet>
    <portlet-name>portletName</portlet-name>
    <portlet-type>ref</portlet-type>
    <portlet-parent>MultiJSP</portlet-parent>
    <config-param>
      <param-name>hptl.module.action</param-name>
      <param-value>sample.portal.portlet.api.SampleActionModule</
param-value>
    </config-param>
    <title>サンプルポートレット</title>
    <description>ポートレットイベント機能を用いたポートレットです。</
description>
    <device media='HTML'>
      <url>/Sample</url>
    </device>
  </portlet>
</portlet-app>
```

8.4　アクションモジュールで利用できる API

アクションモジュールでは API を利用してさまざまな処理を実行できます。

API は大きく分けて次の二つに分類されます。

- イベント処理を実行するための API
- uCosminexus Portal Framework が提供している Bean を利用するための API

8.4.1　イベント処理を実行するための API

アクションモジュールではイベント処理を実行するための API を利用できます。

対応する API を次の表に示します。

表 8-2　イベント処理を実行するための API

項目	対応する API（メソッド）	利用目的
ポートレットの名称	getPortletName	現在処理中のポートレット名称が取得できます。

API の仕様の詳細は，「14.9　ポートレットイベント API」を参照してください。

8.4.2　uCosminexus Portal Framework が提供している Bean を利用するための API

アクションモジュール内では，JSP やサーブレットでも利用できた，uCosminexus Portal Framework が提供している Bean を API として利用できます。このため，従来と変わらない開発環境で開発できます。

アクションモジュール内で取得できる API を次の表に示します。

表 8-3　アクションモジュール内で取得できる API

Bean 名称	対応する API（メソッド）	利用目的
ポートレット情報取得 Bean	getPortletInfo	各ポートレット単位に設定しているパラメタ情報が取得できます。
ユーザ情報取得 Bean	getUserInfo	ログインしているユーザ単位のリポジトリ情報（各利用者のユーザ，組織，グループ情報，カスタマイズ情報など）が取得できます。
ログ出力 Bean	getPortletLog	ポートレットログに，ポートレット単位の情報を出力できます。

Bean 名称	対応する API（メソッド）	利用目的
UserAgent 情報取得 Bean	getUserAgentInfoBean	現在アクセスしているクライアントに対して，クライアント情報定義ファイルで設定されている UserAgentType の情報が取得できます。

API の詳細は「14.9　ポートレットイベント API」を参照してください。

8.5 ポートレットアクションイベント機能

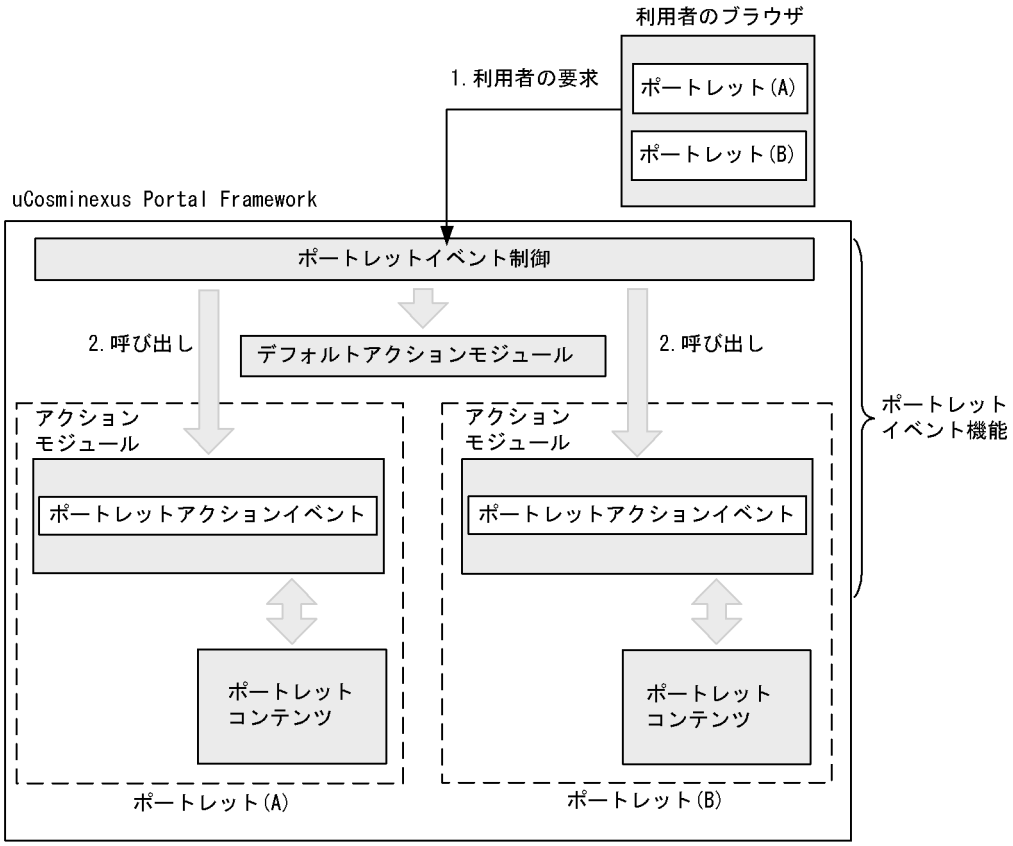
ここでは，ポートレットアクションイベント機能の概要，詳細，利用方法，および利用できる API について説明します。

8.5.1 概要

ポートレットアクションイベント機能は，クライアント側（Web ブラウザ）からコンテンツの取得要求（リクエスト）が発生した場合に呼び出されるイベントを提供します。この機能を利用することで，ポートレットコンテンツを出力する前にさまざまな前処理を実行できます。

ポートレットアクションイベント機能の概要を次の図に示します。

図 8-5 ポートレットアクションイベント機能の概要



ポートレットアクションイベントが呼び出される制御の流れ
1. 利用者の要求によって，ポートレットイベント制御にポートレットアクションイベントが登録されます。
2. 各ポートレットのポートレットアクションイベントが呼び出されます。

8.5.2 詳細

ポートレットアクションイベントは，クライアント側（Web ブラウザ）からコンテンツの取得要求があった場合に発生するイベントです。

このイベントは，現在画面に表示しようとしている，すべてのポートレットに対して発生します。

ポートレットアクションイベントが発生する対象のポートレットを次に示します。

- 画面レイアウトがタブ形式の場合，選択しているタブ内のすべてのポートレット
- 画面レイアウトが行列形式などのタブ形式以外の場合，表示されているすべてのポートレット
- ポートレットが最大化表示されている場合，最大化表示されているポートレット

なお，ポートレットが最小化状態であってもポートレットアクションイベントは発生します。

また，ポートレットのクローズボタンを押したときは，ポートレットアクションイベントは発生しません。

8.5.3　利用方法

ポートレットアクションイベントを利用する場合，ポートレットアクションモジュールに，次に示すポートレットアクションイベントを実装します。

```
public void action(javax.servlet.http.HttpServletRequest request) {
・
・
}
```

API の詳細は「14.9　ポートレットイベント API」を参照してください。

8.5.4　ポートレットアクションイベントで利用できる API

ポートレットイベント機能で提供する API のほかに，次の API をサポートしています。

表 8-4　ポートレットアクションイベントで利用できる API

項目	対応する API（メソッド）	利用目的
ポートレットへの通知	send	ポートレット間通信をするときに相手ポートレットに通知したい場合に呼び出します。

API の詳細は「14.9　ポートレットイベント API」を参照してください。

8.6 ポートレット間通信イベント機能

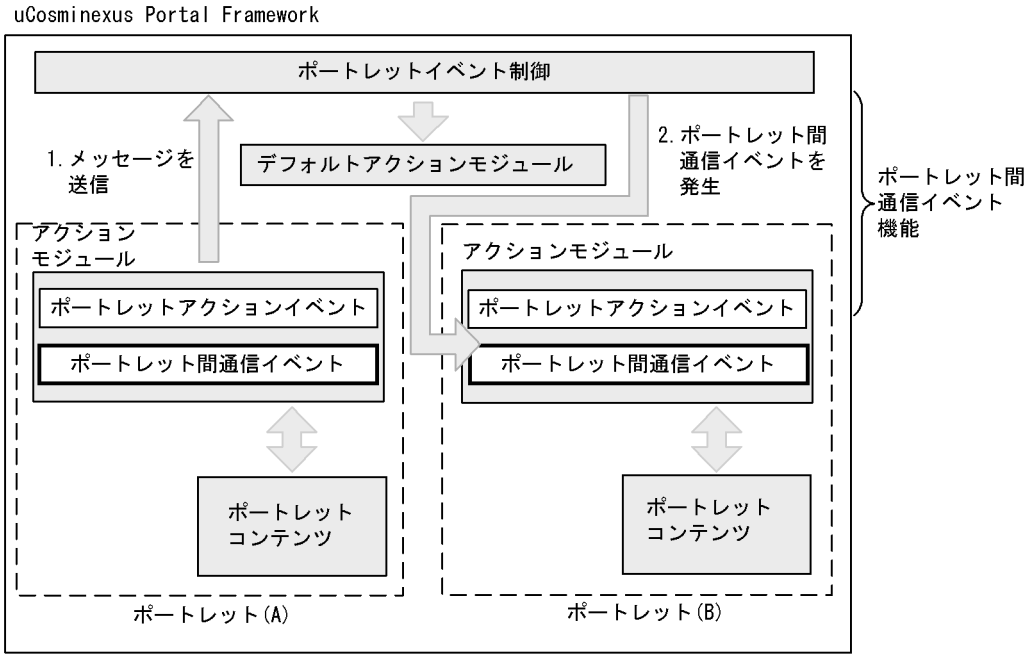
ここでは，ポートレット間通信イベント機能の概要，詳細，利用方法，および利用できる API について説明します。

8.6.1 概要

ポートレット間通信イベント機能は，あるポートレットから，別のポートレットに対してメッセージを送信する機能です。この機能によって，ポートレットの状態変化を検知した場合，ほかのポートレットへ状態を通知したり，処理を指示したりできます。

ポートレット間通信イベント機能の概要を次の図に示します。

図 8-6 ポートレット間通信イベント機能の概要



ポートレット間通信イベントが呼び出される制御の流れ
1. メッセージを送りたいポートレット(A)は，通信先（ポートレット(B)）を指定してメッセージを送信します。
2. ポートレットイベント制御が，通信先に対してポートレット間通信イベントを発生させます。

8.6.2 詳細

ポートレット間通信イベントは，通信元ポートレットが通信要求メソッドを呼び出したときに，通信先で発生します。

ポートレット間通信イベントが発生する対象ポートレットは，次の条件をすべて満たしている必要があります。

- uCosminexus Portal Framework に登録されているポートレット
- その利用者にアクセス権があるポートレット

通信要求は次の二つのイベントで発生させることができます。

- ポートレットアクションイベント
- ポートレット間通信イベント

通信要求メソッドについては，「14.9　ポートレットイベント API」を参照してください。

8.6.3　利用方法

通信要求元ポートレットおよび通信要求先ポートレットの利用方法を説明します。

(1) 通信要求元のポートレット

1. ポートレットアクションモジュールで通信要求メソッドを呼び出します。
2. 通信要求メソッドの応答は，相手側のイベント処理の完了を待たないで，すぐに返却されます。

(2) 通信要求先のポートレット

ポートレット間通信イベントを受信したい場合，ポートレットアクションモジュールに，次に示すポートレット間通信イベントを実装します。

```
public void receive(javax.servlet.http.HttpServletRequest request)
{
・
・
}
```

API の詳細は，「14.9　ポートレットイベント API」を参照してください。

8.6.4　ポートレット間通信イベントで利用できる API

ポートレットイベント機能で提供する API のほかに，次の API をサポートしています。

表 8-5　ポートレット間通信イベントで利用できる API

項目	対応する API（メソッド）	利用目的
ポートレット名称の取得	getSrcPortletName	通信元のポートレット名称を取得します。通信元によって処理を振り分けることができます。
メッセージの取得	getMessage	通信元から通知があったメッセージを取得します。

項目	対応する API（メソッド）	利用目的
ポートレットへの通知	send	ポートレット間通信をするときに相手ポートレットに通知したい場合に呼び出します。

API の詳細は,「14.9　ポートレットイベント API」を参照してください。

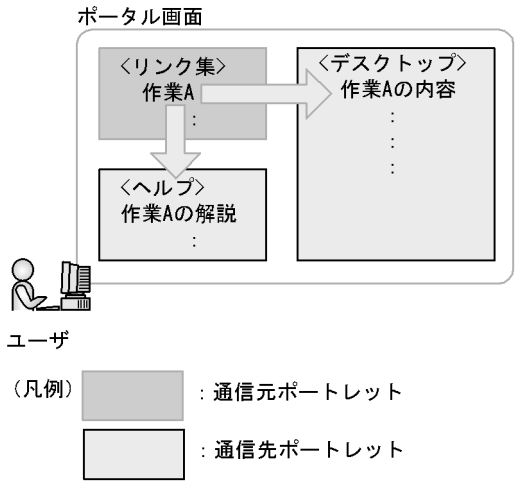
8.7　ポートレットイベント対応ポートレットの開発例

ここでは，ポートレットイベント対応ポートレットの開発例を説明します。

この開発例では，ユーザの好みに応じてヘルプ画面を表示するポータル業務画面の開発方法について説明します。

ポータル業務画面の概要を次の図に示します。

図 8-7　ポータル業務画面例の概要



これは，あるポートレット（ここでは，＜リンク集＞ポートレット）から複数のポートレット（ここでは，＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレット）に情報が送信されるポータル業務画面の例です。

ユーザが＜リンク集＞ポートレットから作業 A を選択すると，＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットに通知されます。この通知によって，作業 A の内容が＜デスクトップ＞ポートレットに，作業 A の解説が「ヘルプ」ポートレットに表示されます。

ユーザは，ポータルの習熟度に応じて，＜ヘルプ＞ポートレットの表示／非表示を切り替えたり，最小化の状態にしておいて必要に応じて復元したりするなどの使い方ができます。また，＜ヘルプ＞ポートレットに「作業に関する緊急のお知らせ」など，その作業をするときに必ず知っておいてほしい情報を表示するような使い方もできます。

＜リンク集＞ポートレットには，業務関連のリンクのほかにも，業務に関連した「経済」や「金融」などのポートレットへのリンクをそろえられます。

開発例の開発手順を次に示します。

手順

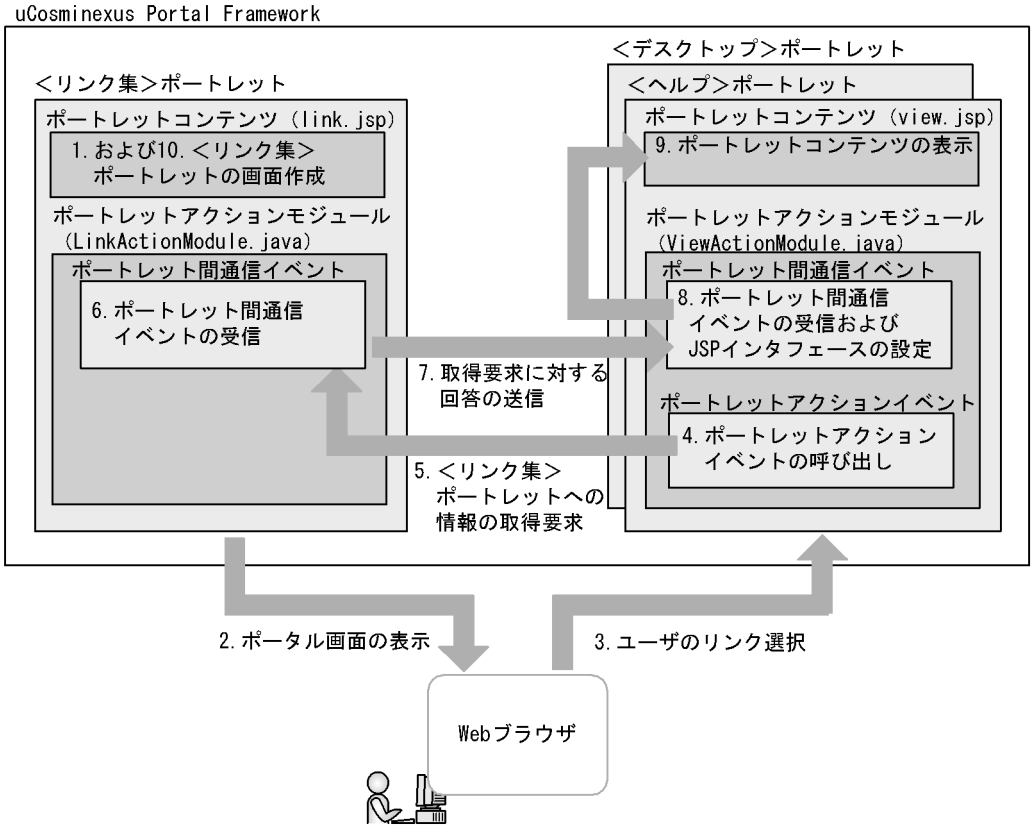
1. **ポータル業務画面の処理を設計します。**
   処理の内容，および流れを設計します。
2. **インタフェースを設定します。**
   ポートレット間のインタフェース，およびポートレットアクションモジュールとポートレットコンテンツ間のインタフェースを設定します。
3. **ポートレットアクションモジュールを開発します。**
   各ポートレットの，ポートレットアクションモジュールの内容を記述します。

各開発手順の詳細を，次に説明します。

8.7.1 ポータル業務画面の処理の設計

開発するポータル業務画面の処理の内容および流れを設計します。処理の内容および流れの設計イメージを次の図に示します。なお，図中の番号は，処理が流れる順序を示します。

図 8-8 処理の内容および流れの設計イメージ



図の説明

1. **＜リンク集＞ポートレットの画面作成**
   ＜リンク集＞ポートレットの画面を作成します。このとき，各リンクに LinkActionModule に対するインタフェースとして，次のようなリクエストパラメタを設定します。
   「識別名+ポートレット名称1」=「表示するJSPファイル名1」，...，「識別名+ポートレット名称n」=「表示するJSPファイル名n」

   なお，「識別名」とは，ほかのポートレットのリクエストパラメタと値が競合しないために必要な設定です。

2. **ポータル画面の表示**
   ユーザに，処理 1. で作成した＜リンク集＞ポートレットを含むポータル画面を表示します。

3. **ユーザのリンク選択**
   ユーザが，ポータル画面の＜リンク集＞ポートレットでリンクを選択することで，uCosminexus Portal Framework に対してコンテンツの取得を要求します。

4. **ポートレットアクションイベントの呼び出し**
   処理 3. でユーザがリンクをクリックすることで，＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットのポートレットアクションモジュール（ViewActionModule.java）が呼び出されます。

5. **＜リンク集＞ポートレットへの情報の取得要求**
   ＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットは，＜リンク集＞ポートレットとの間で取り決めたインタフェースに基づいて，＜リンク集＞ポートレットに対して「画面に表示する JSP ファイル名」の取得を要求するメッセージを送信します。

6. **ポートレット間通信イベントの受信**
   ＜リンク集＞ポートレットのポートレットアクションモジュール（LinkActionModule.java）が，＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットからメッセージを受信します。このとき，処理 1. で取得したリクエストパラメタの内容を解析し，「ポートレット名称 n」が「メッセージ送信元のポートレット名称」と一致しているかどうかを確認します。

7. **取得要求に対する回答の送信**
   処理 6. で「ポートレット名称 n」が「メッセージ送信元のポートレット名称」と一致していれば，「画面に表示する JSP ファイル名 n」を＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットのポートレットアクションモジュール（ViewActionModule.java）に返却します。

8. **ポートレット間通信イベントの受信および JSP インタフェースの設定**
   ＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットのポートレットアクションモジュール（ViewActionModule.java）は，＜リンク集＞ポートレットからのメッセージを受信します。受信したメッセージに格納されている，「画面に表示する

JSP ファイル名 n」を自分自身の JSP に対するリクエストインタフェースとして設定します。

9. **ポートレットコンテンツの表示**
＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットのポートレットコンテンツ（view.jsp）で，処理 8. で指定されたリクエストインタフェースの情報を取り出し，その JSP ファイルの内容を表示します。

10. **＜リンク集＞ポートレットの画面作成**
＜リンク集＞ポートレットの画面を再度作成し，処理 9. で設定された＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットの画面と統合して，ユーザにポータル画面を表示します。

8.7.2 インタフェースの設定

ポートレット間のインタフェース，およびポートレットアクションモジュールとポートレットコンテンツ間のインタフェースを設定します。

（1）ポートレット間のインタフェース

＜リンク集＞ポートレットと，＜ヘルプ＞ポートレットおよび＜デスクトップ＞ポートレットとのインタフェースを設定します。

この開発例では，＜リンク集＞ポートレットと＜ヘルプ＞ポートレット，および＜リンク集＞ポートレットと＜デスクトップ＞ポートレットとのインタフェースを設定します。

ポートレット間通信イベントでのインタフェースを次の表に示します。

表 8-6 ポートレット間通信イベントでのインタフェース

クラス	内容	意味	利用するポートレット
なし (null)	－	表示する JSP ファイル名の取得要求	**通知元：** ＜デスクトップ＞ポートレット，または＜ヘルプ＞ポートレット **通知先：** ＜リンク集＞ポートレット
java.lang.String	表示する JSP ファイル名	表示する JSP ファイル名	**通知元：** ＜リンク集＞ポートレット **通知先：** ＜デスクトップ＞ポートレット，または＜ヘルプ＞ポートレット

ポートレットアクションモジュールのインタフェースを次の表に示します。

表 8-7 ポートレットアクションモジュールのインタフェース

パラメタ名称	パラメタ設定値	利用するポートレット
hptl.ex.comlink.send.name	JSP ファイルの内容を表示するポートレットが取得要求するポートレット名称を指定します。	＜デスクトップ＞ポートレットまたは＜ヘルプ＞ポートレット

注 ポートレットの定義内容は，ポートレット情報取得 Bean で取得できます。ポートレット情報取得 Bean の詳細は，「14.5 ポートレット情報取得 Bean」を参照してください。

(2) ポートレットアクションモジュールとポートレットコンテンツとのインタフェース

＜デスクトップ＞ポートレットおよび＜ヘルプ＞ポートレットは，次のインタフェースを持ちます。

パラメタ名	意味	利用するポートレット
hptl.ex.comlink.include.name	表示する JSP ファイル名	＜デスクトップ＞ポートレットまたは＜ヘルプ＞ポートレット

8.7.3 ポートレットアクションモジュールの開発

ここでは，開発例の，ポートレットアクションモジュールの内容を説明します。

(1) ＜リンク集＞ポートレットのポートレットアクションモジュール

＜リンク集＞ポートレットの，ポートレットアクションモジュールの内容を次に示します。

```
package jp.co.hitachi.soft.portal.portlet.api;

import javax.servlet.http.HttpServletRequest;
import jp.co.hitachi.soft.portal.api.log.PortletLog;

public class LinkActionModule extends DefaultActionModule {

  /**
   * リクエストパラメタの名称<br>
   * ・形式<br>
   * JSPで，以下の形式でリクエストパラメタを設定してください。<br>
   * &識別名ポートレット名称1=表示するJSPファイル名1+識別名ポートレット名称2=
表示するJSPファイル名2<br>
   * ・例<br>
   *
&hptl.ex.comlink.url.PortletName1=a.jsp+hptl.ex.comlink.url.Portle
tName2=b.jsp+...
   */
  public static final String PARAM_REQUEST_LINK =
"hptl.ex.comlink.url.";

  /**
```

```
    * ポートレット間通信イベント処理<br>
    * @param request イベント処理に渡されたリクエストオブジェクト
    */
   public void receive(HttpServletRequest request) {
     try {
       // Webブラウザからのリクエストパラメタを取得します。
       String value = request.getParameter(PARAM_REQUEST_LINK +
getSrcPortletName(request));
       // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
       value = new
String(value.getBytes("iso-8859-1"),"Windows-31J");
       // もし，リクエストパラメタが存在していれば，メッセージを送信します。
       if (value != null) {
         // 実際にメッセージ(表示するJSPファイル名称)を送信します
         send(request, getSrcPortletName(request), value);
       }
     }catch (java.lang.Exception e) {
       // アプリケーションログにエラー情報を出力します
       getPortletLog(request, "log").error("receive error", e);
     }
   }
}
```

(2) ＜デスクトップ＞ポートレットまたは＜ヘルプ＞ポートレットのポートレットアクションモジュール

＜デスクトップ＞ポートレットまたは＜ヘルプ＞ポートレットの，ポートレットアクションモジュールの内容を次に示します。

```
package jp.co.hitachi.soft.portal.portlet.api;

import jp.co.hitachi.soft.portal.portlet.api.*;
import javax.servlet.http.HttpServletRequest;
import jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean;
/**
* アクションイベント機能が提供するAPIです。<br>
*/
public class ViewActionModule extends DefaultActionModule {

  /**
   * ポートレットの定義に指定するパラメタ。<br>
   * 通知するポートレット名称を指定します。<br>
   */
  public static final String PORTLET_PARAMETER_SEND_NAME =
"hptl.ex.comlink.send.name";
  /**
   * ポートレット情報取得Beanのbeanidを設定します。<br>
   */
  public static final String BEAN_ID_PORTLET_INFO =
"hptl.ex.comlink.bean.ViewActionModule";
  /**
   * JSPに設定するパラメタ名称を設定します。<br>
   */
  public static final String CONTENT_PARAMETER_INCLUDE_NAME =
"hptl.ex.comlink.include.name";
  /**
   * ポートレットアクションイベント処理<br>
```

```
   * @param request イベント処理に渡されたリクエストオブジェクト
   */
  public void action(HttpServletRequest request) {
    try {
      // 送信先ポートレット名称を取得する
      String portletName = getPortletInfo(request,
BEAN_ID_PORTLET_INFO).
getParameter(PORTLET_PARAMETER_SEND_NAME);
      // メッセージを送信する
      send(request, portletName, null);
    }catch (java.lang.Exception e) {
      // アプリケーションログにエラー情報を出力します
     getPortletLog(request, "log").error("ViewActionModule.action
error", e);
    }
  }
  /**
   * ポートレット間通信イベント処理<br>
   * @param request イベント処理に渡されたリクエストオブジェクト
   */
  public void receive(HttpServletRequest request) {
    try {
      // 送信先ポートレット名称を取得する
      String portletName = getPortletInfo(request,
BEAN_ID_PORTLET_INFO).
getParameter(PORTLET_PARAMETER_SEND_NAME);
      // 送信先ポートレット名称と，受信したポートレットが一致している
      if (getSrcPortletName(request).equals(portletName)) {
        // メッセージ(JSPファイル名)を取得する
        String name = (String)getMessage(request);
        // JSP/サーブレットに対するインタフェースを設定します
        request.setAttribute(CONTENT_PARAMETER_INCLUDE_NAME, name);
      }
    }catch (java.lang.Exception e) {
      // アプリケーションログにエラー情報を出力します
     getPortletLog(request, "log").error("ViewActionModule.receive
error", e);
    }
  }
}
```

9 クライアントサイドデータ通信対応ポートレットの開発

この章では，クライアントサイドデータ通信機能に対応したポートレットの開発方法について説明します。

9.1 クライアントサイドデータ通信とは

データを入力する項目が多くある場合，直接入力では入力ミスが発生する可能性が高くなります。このような入力ミスを削減するため，ポートレットでデータを入力する場合に，別のポートレットから該当するデータをドラッグ & ドロップしたり，コピーボタンを利用してコピーしたりして入力することができます。この機能を**クライアントサイドデータ通信**といいます。

9.1.1 クライアントサイドデータ通信の処理概要

クライアントサイドデータ通信機能には，次の二つの機能があります。

- ドラッグ & ドロップ
- データフォーム転送

この二つの機能は，同一の HTML 部品上で実現でき，共通の API を使用します。HTML 部品の詳細は，「9.1.4　クライアントサイドデータ通信機能がサポートするHTML」を参照してください。API の詳細は，「14.11　クライアントサイドデータ通信タグライブラリ」，「14.12　クライアントサイドデータ通信 JavaScript」，および「14.13　クライアントサイドデータ通信クラスライブラリ」を参照してください。

9.1.2 ドラッグ & ドロップ

ドラッグ & ドロップ機能を使用すると，入力操作しないでクライアントサイドでのポートレット間データ通信ができます。

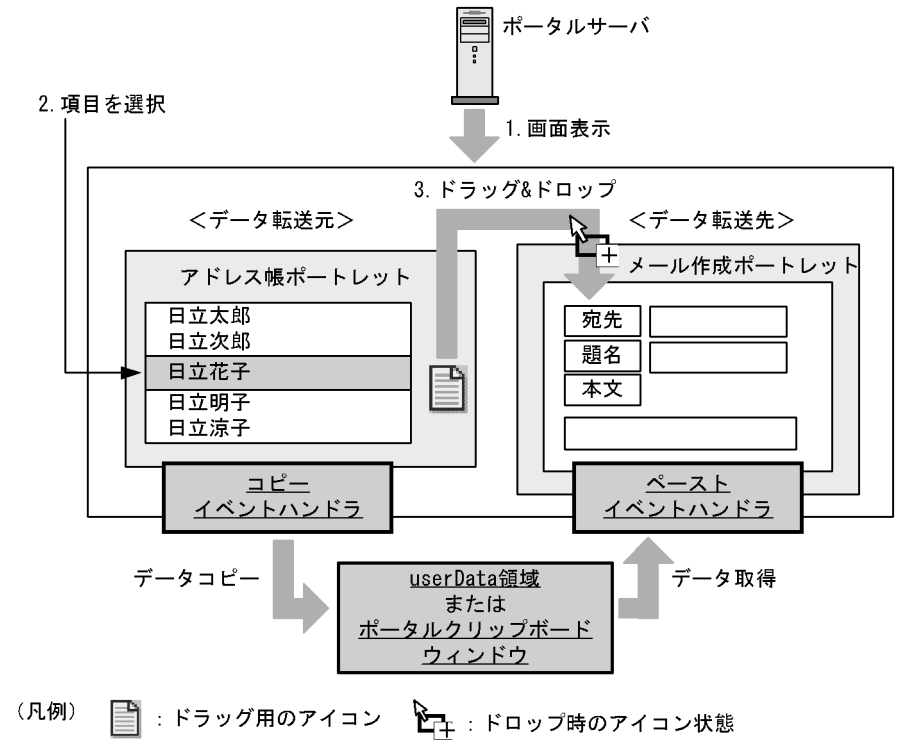
ドラッグ & ドロップ機能を実現するための前提条件を次に示します。

- データ転送先のポートレットには，「ペーストイベントハンドラ」が定義されている必要があります。**ペーストイベントハンドラ**とは，データ転送先にデータを貼り付ける JavaScript イベントハンドラのことです。ペーストイベントハンドラは，ドロップ時に呼ばれます。
- データ転送元には，「コピーイベントハンドラ」が定義されている必要があります。**コピーイベントハンドラ**とは，HTML 上に格納されているシリアライズされた Java オブジェクトである文字列を取得し，ポータルクリップボードウィンドウにその文字列をコピーする JavaScript イベントハンドラのことです。コピーイベントハンドラは，ドラッグ時に呼ばれます。
- ドラッグ & ドロップは，アイコンを使用して実現します。ポートレット開発者は，任意のアイコンを使用できます。アイコンの指定方法については，「14.11.3(3) csdc:dragicon」を参照してください。アイコンをドラッグする際に，マウスカーソルがドロップするフォーム領域に入っていない場合，またはデータ転送先とデータ転送元の datatype が異なる場合は，ドロップ抑止機能が働きます。

なお，ドラッグ & ドロップする際のデータの保管場所には，クライアント PC の userData **領域**，または**ポータルクリップボードウィンドウ**を使用します。userData 領域およびポータルクリップボードウィンドウの詳細は，「9.1.5　データ保管領域」を参照してください。

ドラッグ & ドロップ機能の概要を次の図に示します。

図 9-1　ドラッグ & ドロップ機能の概要



図の説明

1. 画面表示
   Java オブジェクトがシリアライズされ，HTML に格納されます。
   ポータルクリップボードウィンドウだけを使用する方式の場合，ポータルクリップボードウィンドウを開きます。
2. 項目を選択
   ドラッグ対象項目を選択します。ドラッグする項目を選択していない場合は，ドラッグできません。
3. ドラッグ & ドロップ
   ドラッグ処理
   2. で選択した項目に対応するシリアライズされた文字列を「コピーイベントハンドラ」でデシリアライズし，userData 領域またはポータルクリップボードウィンドウにコピーします。

userData 領域に格納するデータを再シリアライズします。シリアライズ後のデータが，PortalResources.properties ファイルの jp.co.hitachi.soft.portal.csdc.datasize.threshold 項目に指定されている値を超えているかどうかによって次のようになります。PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortalResources.properties の詳細」の説明を参照してください。

< 指定値以内の場合 >

userData 領域にデータを格納します。

< 指定値を超えている場合 >

ポータルクリップボードウィンドウを開き，ポータルクリップボードウィンドウにデータを格納します。

ドロップ処理

userData 領域またはポータルクリップウィンドウから取得したデータを「ペーストイベントハンドラ」で入力フォームに反映します。

ポータルクリップボードウィンドウが開かれている場合，ポータルクリップボードウィンドウを閉じます。

9.1.3 データフォーム転送

データフォーム転送機能を使用すると，入力操作またはドラッグ & ドロップを使用しないでクライアントサイドでのポートレット間データ通信ができます。

データフォーム転送機能には，次の二つの方法があります。

- 補助ウィンドウを使用する方法
- ボタンを使用する方法

(1) 補助ウィンドウを使用する方法

補助ウィンドウを使用する方法とは，入力項目が表示されている補助ウィンドウから入力項目を選択する方法です。入力項目が表示されている補助ウィンドウを開き，入力項目を選択したあと［OK］ボタンをクリックすることにより，補助ウィンドウで選択した入力項目のデータを入力フォームに出力します。

補助ウィンドウを使用する方法を実現するための前提条件を次に示します。

- データ転送先のメイン画面には，「ペーストイベントハンドラ」が定義されている必要があります。**ペーストイベントハンドラ**とは，データ転送先にデータを貼り付けるための JavaScript ハンドラのことです。
- データ転送元には，「コピーイベントハンドラ」が定義されている必要があります。**コピーイベントハンドラ**とは，HTML 上に格納されているシリアライズされた Java オブジェクトである文字列を取得し，ポータルクリップボードウィンドウにその文字列をコピーする JavaScript イベントハンドラのことです。

補助ウィンドウを使用する方法の場合，ペーストイベントハンドラおよびコピーイベントハンドラは，［OK］ボタンをクリックしたときに呼ばれます。

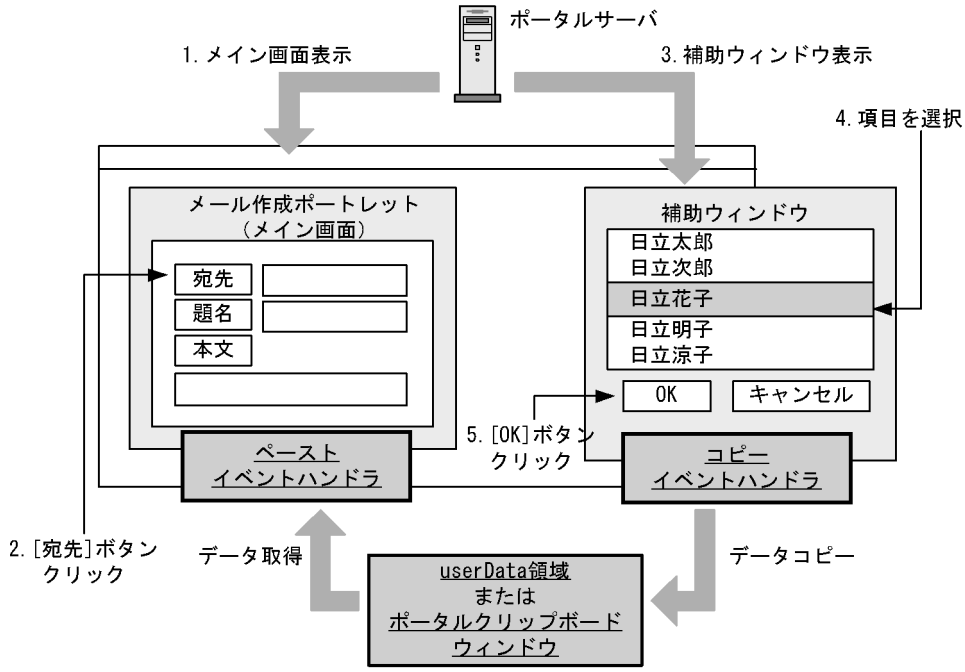
なお，ドラッグ & ドロップする際のデータの保管場所には，クライアント PC の userData **領域**，または**ポータルクリップボードウィンドウ**を使用します。userData 領域およびポータルクリップボードウィンドウの詳細は，「9.1.5　データ保管領域」を参照してください。

補助ウィンドウを使用する方法の実現方法を次に示します。

- 補助ウィンドウを使用する方法では，コピーイベントハンドラおよびペーストイベントハンドラを一度に実行するため，補助ウィンドウは，データ転送先に定義されているペーストイベントハンドラの名称を認識している必要があります。
- データ転送先では，補助ウィンドウを開く際の URL に，実行するペーストイベントハンドラ名称をクエリとして埋め込む必要があります。クエリ内のペーストイベントハンドラを認識するキー名称は，「hptl.pasteHandler」です。

補助ウィンドウを使用する方法の概念を次の図に示します。

図 9-2　補助ウィンドウを使用する方法の概念（データフォーム転送）



図の説明

1. メイン画面表示
2. ［宛先］ボタンをクリック
3. 補助ウィンドウを表示

補助ウィンドウが表示されます。URL のクエリ部分に，実行するペーストイベントハンドラの名称を指定します。クエリ内のペーストハンドラを認識するキー名称は，「hptl.pasteHandler」です。
ポータルクリップボードウィンドウだけを使用する方式の場合，ポータルクリップボードウィンドウを開きます。

4. 項目を選択
データ転送の対象となる項目を選択します。データ転送の対象となる項目を選択していない場合は，データを転送できません。

5. ［OK］ボタンをクリック
4. で選択した項目に対応するシリアライズされた文字列を「コピーイベントハンドラ」でデシリアライズし，userData 領域またはポータルクリップボードウィンドウにコピーします。
userData 領域に格納するデータを再シリアライズします。シリアライズ後のデータが，PortalResources.properties ファイルの jp.co.hitachi.soft.portal.csdc.datasize.threshold 項目に指定されている値を超えているかどうかによって次のようになります。PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortalResources.properties の詳細」の説明を参照してください。

< 指定値以内の場合 >

userData 領域にデータを格納します。

< 指定値を超えている場合 >

処理を続行するかどうかを示すダイアログボックスが表示されます。
処理を続行するときは，ポータルクリップボードウィンドウを開き，ポータルクリップボードウィンドウにデータを格納します。
処理を中止するときは終了します。
URL のクエリ部分から「hptl.pasteHandler」のキー値を取得します。これを実行する**ペーストイベントハンドラ**とします。
ペーストイベントハンドラを実行し，userData 領域またはポータルクリップボードウィンドウから取得したデータを入力フォームに反映します。
ポータルクリップボードウィンドウが開かれている場合，ポータルクリップボードウィンドウを閉じます。

(2) ボタンを使用する方法

ボタンを使用する方法とは，入力項目を選択して［コピー］ボタンをクリックし，データ転送先で［貼り付け］ボタンをクリックすることにより選択した入力項目のデータを入力フォームに入力する方法です。

ボタンを使用する方法を実現するための前提条件を次に示します。

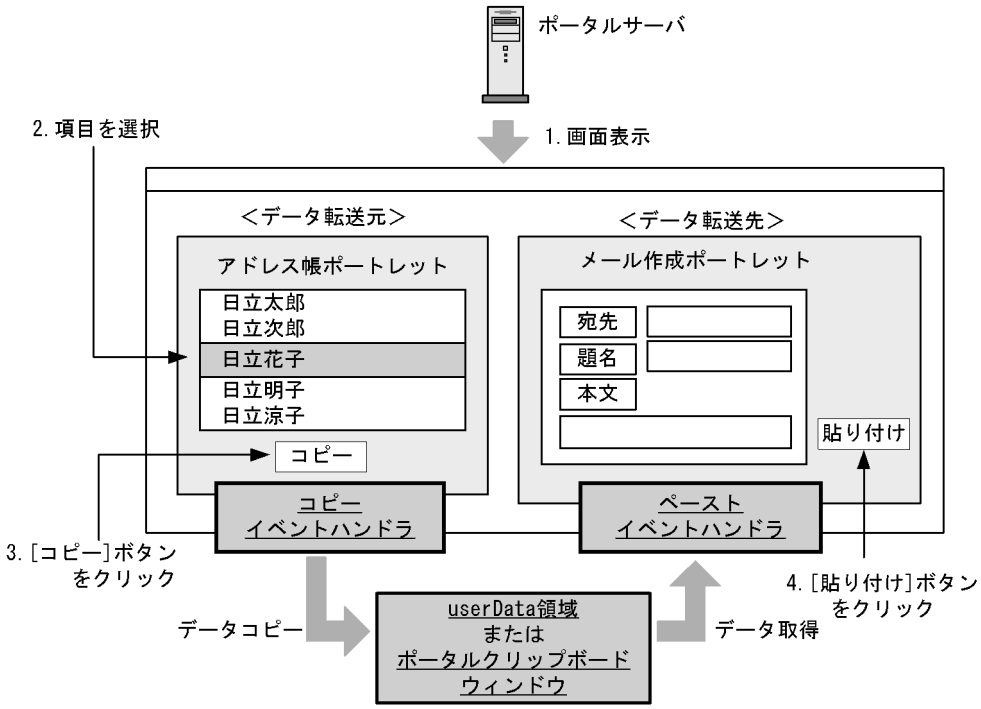
- データ転送先のポートレットには，「ペーストイベントハンドラ」が定義されている必要があります。**ペーストイベントハンドラ**とは，データ転送先にデータを貼り付けるための JavaScript ハンドラのことです。ペーストイベントハンドラは，［貼り付け］ボタンがクリックされたときに呼ばれます。

● データ転送元には，「コピーイベントハンドラ」が定義されている必要があります。**コピーイベントハンドラ**とは，HTML 上に格納されているシリアライズされた Java オブジェクトである文字列を取得し，ポータルクリップボードウィンドウにその文字列をコピーする JavaScript イベントハンドラのことです。コピーイベントハンドラは，［コピー］ボタンがクリックされたときに呼ばれます。

なお，ドラッグ & ドロップする際のデータの保管場所には，クライアント PC の userData **領域**，または**ポータルクリップボードウィンドウ**を使用します。userData 領域およびポータルクリップボードウィンドウの詳細は，「9.1.5　データ保管領域」を参照してください。

ボタンを使用する方法の概念を次の図に示します。

図 9-3　ボタンを使用する方法の概念（データフォーム転送）



図の説明

1. 画面表示
   Java オブジェクトがシリアライズされ，HTML に格納されます。
   ポータルクリップボードウィンドウだけを使用する方式の場合，ポータルクリップボードウィンドウを開きます。
2. 項目を選択
   データ転送の対象となる項目を選択します。データ転送の対象となる項目を選択していない場合は，データを転送できません。

3. ［コピー］ボタンをクリック
   2. で選択された項目に対応するシリアライズされた文字列を「コピーイベントハンドラ」でデシリアライズし，userData 領域またはポータルクリップボードウィンドウにコピーします。
   userData 領域に格納するデータを再シリアライズします。シリアライズ後のデータが，PortalResources.properties ファイルの jp.co.hitachi.soft.portal.csdc.datasize.threshold 項目に指定されている値を超えているかどうかによって次のようになります。PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortalResources.properties の詳細」の説明を参照してください。
   < 指定値以内の場合 >
   userData 領域にデータを格納します。
   < 指定値を超えている場合 >
   処理を続行するかどうかを示すダイアログボックスが表示されます。
   処理を続行するときは，ポータルクリップボードウィンドウを開き，ポータルクリップボードウィンドウにデータを格納します。
   処理を中止するときは終了します。
4. ［貼り付け］ボタンをクリック
   userData 領域またはポータルクリップボードウィンドウから取得したデータを「ペーストイベントハンドラ」で，入力フォームに反映します。
   ポータルクリップボードウィンドウが開かれている場合，ポータルクリップボードウィンドウを閉じます。

9.1.4 クライアントサイドデータ通信機能がサポートする HTML

クライアントサイドデータ通信機能がサポートする HTML 部品を次に示します。

図 9-4　サポート対象 HTML 部品

転送先 転送元	HTML タグ 名称	input					text area	select	
HTML タグ 名称	属性	text	file	check box	radio	hidden	—	—	multiple
input	text	○	○	×	×	○	○	×	×
	file	×	×	×	×	×	×	×	×
	check box	○	○	×	×	○	○	×	×
	radio	○	○	×	×	○	○	×	×
	hidden	○	○	×	×	○	○	×	×
text area	—	○	○	×	×	○	○	×	×
select	—	○	○	×	×	○	○	×	×
	multiple	○	○	×	×	○	○	×	×

（凡例）
　○：サポートします。
　×：サポートしません。
　—：該当しません。

注　クライアントサイドデータ通信機能では，すべてのHTML部品での動作を想定しています。
　　ただし，動作を保証するHTML部品は，図に示しているとおりです。

9.1.5　データ保管領域

クライアントサイドデータ通信を実現する際には，データの保管場所として，userData **領域**と**ポータルクリップボードウィンドウ**を使用できます。

(1) データ保管領域の設定

データ保管領域の設定は，PortalResources.properties ファイルで行います。

(a) データ格納方式の設定

クライアントサイドデータ通信機能では，次に示す方式でデータを格納できます。

- userData 領域とポータルクリップボードウィンドウを併用する方式
- ポータルクリップボードウィンドウだけを使用する方式

データ格納方式は，PortalResources.properties ファイルの jp.co.hitachi.soft.portal.csdc.datatype.userData 項目で指定します。PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」を参照してください。

(b) 転送データサイズの設定

データの保管場所として userData 領域とポータルクリップボードウィンドウを併用する場合，データの保管場所に userData 領域を使用するか，またはポータルクリップボードウィンドウを使用するかは，保管するデータが転送データサイズの指定値以内かどうかで決まります。

転送データサイズは，PortalResources.properties ファイルの jp.co.hitachi.soft.portal.csdc.datasize.threshold 項目で指定します。PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」を参照してください。保管するデータが，転送データサイズの指定値以内かどうかによって次のようになります。

- 保管するデータが転送データサイズの指定値以内の場合
  userData 領域を使用します。
- 保管するデータが転送データサイズの指定値を超える場合
  ポータルクリップボードウィンドウを使用します。

なお，HTTPS 環境でクライアントサイドデータ通信機能を使用している場合，転送データサイズが指定値を超えると，セキュリティに関する警告ダイアログボックスが 2 回表示されます。

(2) userData 領域

userData 領域は，クライアントサイドデータ通信を実現する際に使用するデータの保管場所です。コピー処理では，通常 userData 領域をデータの保管場所として使用します。

userData 領域には文字列だけ格納できます。そのため，データ格納時には，ポートレットで指定した JavaScript 連想配列オブジェクトはシリアライズされ，データ取得時には，userData 領域から取得した文字列はデシリアライズされます。

userData 領域へのデータの格納に失敗した場合，次のようになります。

- userData 領域より格納するデータが大きいとき
  ポータルクリップボードウィンドウを使用して，データ転送をします。
- 上記以外のとき（環境不正など）
  警告を示すダイアログボックスが表示され，データ転送をしないで終了します。

(3) ポータルクリップボードウィンドウ

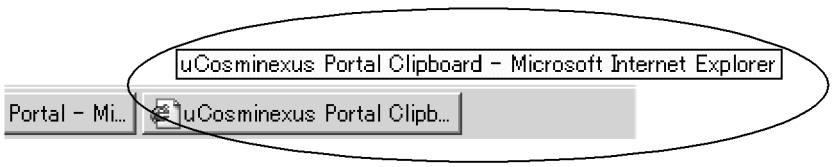
ポータルクリップボードウィンドウは，HTML 上で開かれる一般のウィンドウで，データを保管する場所としての機能だけを持ちます。

ポータルクリップボードウィンドウは，csdc.js をインクルードするタイミングで開かれます。そのため，ポートレット開発者は，ポータルクリップボードウィンドウを開いたり閉じたりする処理を意識する必要はありません。

ポータルクリップボードウィンドウは，コピーしたデータを貼り付けるまで開かれている必要があります。エンドユーザの操作向上およびエンドユーザによって閉じられることを抑止するため，ポータルクリップボードウィンドウは，Window 画面の表示領域外に表示されます（エンドユーザには見えないウィンドウとして表示します)。なお，ポータルクリップボードウィンドウが表示されていない状態でデータ転送をした場合には，再操作を促すダイアログボックスが表示されます。

ポータルクリップボードウィンドウが表示されている間，タスクバーには次のように表示されます。

図 9-5　ポータルクリップウィンドウが表示されている間のタスクバー



9.2 クライアントサイドデータ通信対応ポートレットを開発する前に

ここでは，クライアントサイドデータ通信に対応するポートレットを開発する前に知っておいていただきたい内容について説明します。

9.2.1 前提条件

(1) ポータルサーバの前提条件

ポータルサーバは，Web サーバとの連携が設定されている必要があります。

J2EE サーバモードを利用する場合，SecurityManager 定義ファイルに「permission java.lang.reflect.ReflectPermission」定義があることを確認してください。セキュリティポリシーファイル（server.policy）の内容については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「uCosminexus Portal Framework のセットアップ」の説明を参照してください。

(2) Web ブラウザの前提条件

前提とする Web ブラウザ，およびクライアントサイドデータ通信が提供する機能を次に示します。

表 9-1　Web ブラウザの前提条件

Web ブラウザ	提供機能※1	提供 API
Internet Explorer	ドラッグ & ドロップ	クライアントサイドデータ通信が提供する API※2
	データフォーム転送	

注※ 1　Windows XP SP2 のポップアップブロックを設定している場合，ポータルクリップボードを使用したデータ転送はできません。
注※ 2　クライアントサイドデータ通信が提供する API の詳細は，「14.11　クライアントサイドデータ通信タグライブラリ」，「14.12　クライアントサイドデータ通信 JavaScript」，および「14.13　クライアントサイドデータ通信クラスライブラリ」を参照してください。

(3) その他の前提条件

(a) セキュリティゾーン

クライアントサイドデータ通信機能を使用できるかどうかは，ポータルサイトが含まれるセキュリティゾーンによって異なります。セキュリティゾーンごとのクライアントサイドデータ通信機能の使用可否を次に示します。

表 9-2　セキュリティゾーンごとの使用可否

項番	セキュリティゾーン	使用可否
1	インターネット	○
2	イントラネット	○
3	信頼済みサイト	○
4	制限付きサイト	×

（凡例）
○：クライアントサイドデータ通信機能を使用できます。
×：クライアントサイドデータ通信機能を使用できません。

(b) ポートレットの表示モード

クライアントサイドデータ通信機能は，default.jsp や SinglePortlet.jsp を経由しているコンテンツで使用できます。ポートレットの表示モードごとの使用可否を次に示します。

表 9-3　ポートレットの表示モードごとの使用可否

項番	表示モード	その他の条件	使用可否
1	通常	－	○
2	最大化	－	○
3	編集	－	○
4	iframe	ポータルコンテンツのフレーム内に表示	○
5		フレームを介さずブラウザに直接表示	×※
6	新規ウィンドウ	－	○
7	－	ポータル外コンテンツ	×※

（凡例）
－：該当しません。
○：クライアントサイドデータ通信機能を使用できます。
×：クライアントサイドデータ通信機能を使用できません。
注※　コンテンツ内に iframe を追加して，ポータル内のビヘイビア定義用コンテンツを取り込むと，クライアントサイドデータ通信機能を使用できます。この場合，ポートレット側を変更します。

(c) ポータル画面の表示方法

クライアントサイドデータ通信機能は，ポータル全体のコンテンツをフレーム内に表示する場合には使用できません。

9.2.2　提供ファイル

uCosminexus Portal Framework は，クライアントサイドデータ通信対応ポートレットのために，次のファイルを提供します。

```
uCosminexus Portal Framework  インストールディレクトリ
└── webapp
    ├── WEB-INF
    │   └── lib
    │       └── hitachiportal.jar・・・・・提供タグライブラリ
    ├── js
    │   ├── csdc.jsp ・・・・・・・・・・・・提供JavaScriptファイル
    │   ├── csdc_common.jsp ・・・・・・・・提供JavaScriptファイル
    │   ├── csdc_userdata.jsp・・・・・・・・提供JavaScriptファイル
    │   └── csdc_clipwin.jsp ・・・・・・・・提供JavaScriptファイル
    ├── images
    │   └── item.gif ・・・・・・・・・・・・<csdc:dragicon>デフォルト表示gifファイル
    └── html
        ├── clipboard.html ・・・・・・・・・ポータルクリップボードウィンドウ
        ├── userData.html  ・・・・・・・・・userData定義HTML
        ├── blank.html ・・・・・・・・・・・シリアライズ用テンポラリファイル
        └── close_clipboard.html ・・・・・・ポータルクリップボードウィンドウを閉じる
                                              ためのモジュール
```

9.2.3 クライアントサイドデータ通信対応ポートレットを開発するための定義

クライアントサイドデータ通信機能には，前述したように次の三つの種別があります。

- ドラッグ & ドロップ
- 補助ウィンドウを使用したデータフォーム転送
- ボタンを使用したデータフォーム転送

ポートレット開発者がクライアントサイドデータ通信対応ポートレットを開発する際，種別ごとに従う必要のある定義パターンがあります。

定義パターンの詳細は，「9.3　ドラッグ & ドロップの定義方法（パターン 1)」，「9.4 補助ウィンドウを使用したデータフォーム転送の定義方法（パターン 2)」，および「9.5　ボタンを使用したデータフォーム転送の定義方法（パターン 3)」を参照してください。

9.3 ドラッグ & ドロップの定義方法（パターン 1）

ここでは，ドラッグ & ドロップをするための定義方法について説明します。

9.3.1 定義規則

(1) ドラッグ元画面

1. <csdc:src> の datatype 属性を指定します。ドラッグ先の <csdc:dest> の datatype と一致するよう定義します。
   複数 datatype を指定する場合は，コンマ区切りで指定してください。
2. <csdc:oncopy> タグ内にコピーイベントハンドラの処理を定義します。
   (I) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用し，ドラッグ対象 HTML オブジェクトを取得します。
   (II) JavaScript 公開関数「hptl_deserializeH2JS」を使用し，(I) で取得した値を JavaScript 配列オブジェクトに格納します。
   (III) JavaScript 公開関数「hptl_setPortalClipboardData」を使用し，(II) で作成した JavaScript 配列オブジェクトを，userData 領域またはポータルクリップボードウィンドウに格納します。
   複数 datatype を指定する場合は，「hptl_setPortalClipboardData」を複数回呼んでください。
3. <csdc:body> タグ内に HTML 構成を定義します。
   (I) HTML 部品の value 値に <csdc:serializeJ2H> を定義し，Java オブジェクトをシリアライズして HTML に保存します。
   (II) タグ内のアイコンを表示したい位置に，<csdc:dragicon> タグを定義します。

(2) ドロップ先画面

1. <csdc:dest> の datatype 属性を指定します。ドラッグ元の <csdc:src> の datatype と一致するように定義してください。ただし，このバージョンでは，<csdc:dest> の datatype 属性に指定した，先頭 datatype だけ有効とします。
2. <csdc:onpaste> タグ内にペーストイベントハンドラの処理を定義します。
   (I) JavaScript 公開関数「hptl_getPortalClipboardData」を使用して，userData 領域またはポータルクリップボードウィンドウから JavaScript 連想配列オブジェクトを取得します。
   (II) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用して，ドロップ先の HTML オブジェクトを取得します。
   (III) (I) で取得した JavaScript 連想配列オブジェクトを展開して値を取得します。
   (IV) (II) で取得した HTML オブジェクトの value に (III) の値を格納します。
3. <csdc:body> タグ内にドロップ先の HTML 構成を定義します。

9.3.2 定義例

(1) ドラッグ元画面およびドロップ先画面共通の定義例

```
<%!
    public class Employee {
        public String name, mail, id;
        public Employee(String name, String mail, String id) {
            this.name = name;
            this.mail = mail;
            this.id = id;
        }
        public String getName() { return name; }
        public String getMail() { return mail; }
        public String getId() { return id; }
    }
%>
<%
    int n = 0;
    Employee[] e = new Employee[5];
    e[n++] = new Employee("日立太郎", "taro@crl.hitachi.co.jp", "123456");
    e[n++] = new Employee("日立花子", "hanako@crl.hitachi.co.jp", "765432");
    e[n++] = new Employee("日立次郎", "aigasa@domain.com", "1111xx");
%>
```

(2) ドラッグ元画面の定義例

例の中の番号は,「9.3.1 定義規則」の該当個所を示します。

```
<csdc:src datatype="Employee">                                          ・・・(1)-1
    <csdc:oncopy name="copy">
        var items = []; // データソース格納配列
        // HTMLオブジェクトからユーザ選択データを抽出
        var list = document.getElementById("addrlist");            ・・・(1)-2-(I)
        n = 0;
        for (var i = 0; i < list.options.length; i++) {
          if  (list.options[i].selected) {
           items[n] = hptl_deserializeH2JS("bean", list.options[i].value);・・(1)-2-(II)
           n++;
          }
        }

        // userData領域またはポータルクリップボードウィンドウにitemsを設定
        hptl_setPortalClipboardData("Employee", items);            ・・・(1)-2-(III)
    </csdc:oncopy>
    <csdc:body>
        //従業員一覧(Humanオブジェクト)
        <select id="addrlist" multiple="multiple">
<%  for (int i = 0; i < e.length; i++) { %>
     <option value="
         <csdc:serializeJ2H type="bean" data="<%= e[i] %>"/>       ・・・(1)-3-(I)
      ">
<%= e[i].getName() %></option>
<%  } %>
        </select>
        <csdc:dragicon desc="ドラッグ＆ドロップできます"/>          ・・・(1)-3-(II)
        <input type=button value="コピー" onClick="javascript:copy(null);">
    </csdc:body>
</csdc:src>
```

(3) ドロップ先画面の定義例

例の中の番号は,「9.3.1　定義規則」の該当個所を示します。

```
<form>
    <csdc:dest datatype="Employee" nodetype="inline">                ・・・(2)-1
        <csdc:onpaste name="myhandler">
            // userData領域またはポータルクリップボードからアイテムを取得
            var items = hptl_getPortalClipboardData( “Employee” );  ・・・(2)-2-(I)

            var obj = document.getElementById("to");                 ・・・(2)-2-(II)
            var result = "";
            if(items != null){                                       ┐
                for (var i = 0; i < items.length; i++) {             │・・・(2)-2-(III)
                    var item = items[i];                             │
                    result += item["name"] + "<" + item["mail"] +">;";┘
                }
                obj.value = result;                                  ・・・(2)-2-(IV)
            }
        </csdc:onpaste>
        <csdc:body>                                                  ┐
            <input id="to" type="text" name="to" size="50"> <br>     ┘・・・(2)-3
        </csdc:body>

    </csdc:dest>
</form>
```

9.4 補助ウィンドウを使用したデータフォーム転送の定義方法（パターン 2）

ここでは，補助ウィンドウを使用したデータフォーム転送の定義方法について説明します。

9.4.1 定義規則

(1) データ転送元画面（補助ウィンドウ）

1. <csdc:src> の datatype 属性を指定します。データ転送先の <csdc:dest> の datatype と一致するよう定義してください。
   複数 datatype を指定する場合は，コンマ区切りで指定してください。
2. <csdc:oncopy> タグ内にコピーイベントハンドラの処理を定義します。
   (I) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用して，データ転送元 HTML オブジェクトを取得します。
   (II) JavaScript 公開関数「hptl_deserializeH2JS」を使用して，(I) で取得した値を JavaScript 配列オブジェクトに格納します。
   (III) JavaScript 公開関数「hptl_setPortalClipboardData」を使用して，(II) で作成した JavaScript 配列オブジェクトを，userData 領域またはポータルクリップボードウィンドウに格納します。複数 datatype を指定する場合は，
   「hptl_setPortalClipboardData」を複数回呼んでください。
3. <csdc:body> タグ内にデータ転送元の HTML 構成を定義します。
   (I) このタグに <csdc:serializeJ2H> タグを定義し，Java オブジェクトをシリアライズして文字列として保存します。
4. ［OK］ボタンをクリックしたときに実行される関数を作成します。
   (I) コピーイベントハンドラを実行する関数 (hptl_oncopy) を定義します。
   複数 datatype を指定する場合は，第一引数にコンマ区切りの文字列を指定してください。
   (II) ペーストイベントハンドラを実行する関数 (hptl_onpaste) 定義します。
   複数 datatype を指定する場合は，第一引数にコンマ区切りの文字列を指定してください。ペーストイベントハンドラを実行する関数 (hptl_onpaste) の詳細は，「14.12 クライアントサイドデータ通信 JavaScript」を参照してください。
   (III) 補助ウィンドウを閉じます。

(2) データ転送先画面 (補助ウィンドウを開く画面)

1. 補助ウィンドウを開きます。
   (I) 補助ウィンドウの URL を作成します。
   (II) 補助ウィンドウを開く JavaScript を定義します。
2. <csdc:dest> タグの datatype 属性を指定します。補助ウィンドウ画面の <csdc:src>

タグの datatype と一致するよう定義してください。ただし，このバージョンでは，<csdc:dest> の datatype 属性に指定した先頭 datatype だけを有効とします。

3. <csdc:onpaste> タグ内にペーストイベントハンドラの処理を定義します。
   (I) JavaScript 公開関数「hptl_getPortalClipboardData」を使用して，userData 領域またはポータルクリップボードウィンドウから JavaScript 連想配列オブジェクトを取得します。
   (II) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用して，データ転送先の HTML オブジェクトを取得します。
   (III) (I) で取得した JavaScript 連想配列オブジェクトを展開して，値を取得します。
   (IV) (II) で取得した HTML オブジェクトの value に (III) の値を格納します。
4. <csdc:body> タグにデータ転送先の HTML 構成を定義します。

注意事項

- データ転送先画面の定義 <csdc:dest> の datatype が，データ転送元画面の定義 <csdc:src> の datatype に含まれていない場合は何の処理も行われません。
- 複数階層にわたって補助ウィンドウを開いた場合に，途中のウィンドウが閉じられた場合は，親ウィンドウを見つけることができなくなります。その場合は，各ポートレット開発者が，適切なエラーメッセージを表示するなどの対応をする必要があります。

9.4.2 定義例

(1) データ転送元（補助ウィンドウ）

例の中の番号は，「9.4.1 定義規則」の該当個所を示します。

```
<%@ page contentType=" text/html; charset=UTF-8"  %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/api/csdc" prefix="csdc" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<meta http-equiv="Content-Script-Type" content="text/javascript">
</head>
<body>
<script src="<%=request.getContextPath()%>/js/csdc.js" type="text/javascript"></script>
          <% String query = request.getQueryString();
             String key = query.substring(0,query.indexOf("="));
             String value = query.substring(query.indexOf("=") + 1,query.length());
             String pasteHandler = "";
             if (key.equals("hptl.pasteHandler")) {
               pasteHandler = value;
             }%>
<script type="text/javascript">
function done() {
  if(window.opener.closed == true){ return; }
  hptl_oncopy( “Employee” ,myhandler” ,null);                          ・・・(1)-4-(I)
  var func = window.opener.<%= pasteHandler %>                         ・・・(1)-4-(II)
  hptl_onpaste( “Employee” ,func,null);                                ・・・(1)-4-(III)
  window.close();
 }
</script>
<csdc:src datatype="Employee">
 <csdc:oncopy name="myhandler">                                        ・・・(1)-1
     var items = []; // データソース格納配列
     // HTMLオブジェクトからユーザ選択データを抽出
     var list = document.getElementById("addrlist");                   ・・・(1)-2-(I)
     n = 0;
     for (var i = 0; i < list.options.length; i++) {
        if  (list.options[i].selected) {
         items[n] = hptl_deserializeH2JS("bean", list.options[i].value); ・・(1)-2-(II)
         n++;
        }
      }
    // userData領域またはポータルクリップボードにitemを設定
   hptl_setPortalClipboardData("Employee", items);                     ・・・(1)-2-(III)
 </csdc:oncopy>

<csdc:body>
 <select id="addrlist" multiple="multiple">
 <%  for (int i = 0; i < e.length; i++) { %>
   <option value="
    <csdc:serializeJ2H type="bean" data="<%= e[i] %>"/>                ・・・(1)-3-(I)
   ">
   <%= e[i].getName() %>
   </option>
<%  } %>
 </select>
 <csdc:dragicon/>
</csdc:body>
</csdc:src>

 <button onclick="done()">完了</button>
```

(2) データ転送先画面

例の中の番号は,「9.4.1　定義規則」の該当個所を示します。

```
<%
  String uri = PortletURI.lookupURI(request,response," testportlet",
    PortletInfoBean.Mode.NEWWINDOW," hptl.pasteHandler=myhandler");     ・・(2)-1-(I)
%>
<script language=" JavaScript" >
function winOpen() {
  var win = window.open( "<%= uri %>" ," hojyo" );                      ・・・(2)-1-(II)
}
</script>
<form>
    <csdc:dest datatype="Employee" nodetype="inline">                   ・・・(2)-2
        <csdc:onpaste name="myhandler">
            // userData領域またはポータルクリップボードからアイテムを取得
            var items = hptl_getPortalClipboardData( "Employee" );      ・・・(2)-3-(I)

            var obj = document.getElementById("to");                    ・・・(2)-3-(II)
            var result = "";
            if(items != null){
                for (var i = 0; i < items.length; i++) {
                    var item = items[i];                                ・・(2)-3-(III)
                    result += item["name"] + "<" + item["mail"] +">;";
                }
                obj.value = result;                                     ・・・(2)-3-(IV)
            }
        </csdc:onpaste>
        <csdc:body>
        <input id="to" type="text" name="to" size="50"> <br>            ・・(2)-4
        <input type=" button"  value=" 宛先"  onClick=" javascript:winOpen();" >
        </csdc:body>

    </csdc:dest>
</form>
```

9.5 ボタンを使用したデータフォーム転送の定義方法（パターン 3）

ここでは，ボタンを使用したデータフォーム転送の定義方法について説明します。

9.5.1 定義規則

(1) データ転送元画面

1. <csdc:src> の datatype 属性を指定します。データ転送先画面の <csdc:dest> の datatype と一致するよう定義してください。
   複数 datatype を指定する場合は，コンマ区切りで指定してください。
2. <csdc:oncopy> タグ内にコピーイベントハンドラの処理を定義します。
   (I) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用して，データ転送元 HTML オブジェクトを取得します。
   (II) JavaScript 公開関数「hptl_deserializeH2JS」を使用して，(I) で取得した値を JavaScript 配列オブジェクトに格納します。
   (III) JavaScript 公開関数「hptl_setPortalClipboardData」を使用して，(II) で作成した JavaScript 配列オブジェクトを，userData 領域またはポータルクリップボードウィンドウに格納します。複数 datatype を指定する場合は，「hptl_setPortalClipboardData」を複数回呼んでください。
3. <csdc:body> タグ内にデータ転送元の HTML 構成を定義します。
   (I) HTML 部品の value 値に <csdc:serializeJ2H> を定義し，Java オブジェクトをシリアライズして，HTML に保存します。
   (II) ［コピー］ボタンをクリックすると実行される関数を作成し，その関数内でコピーイベントハンドラを呼ぶ関数 (hptl_oncopy) を定義します。複数 datatype を指定する場合は，第一引数にコンマ区切りの文字列を指定してください。

(2) データ転送先画面

1. <csdc:dest> の datatype 属性を指定します。データ転送元の <csdc:src> の datatype と一致するよう定義してください。ただし，このバージョンでは，<csdc:dest> の datatype 属性に指定した先頭 datatype だけを有効とします。
2. <csdc:onpaste> タグ内にペーストイベントハンドラの処理を定義します。
   (I) JavaScript 公開関数「hptl_getPortalClipboardData」を使用して，userData 領域またはポータルクリップボードウィンドウから JavaScript 連想配列オブジェクトを取得します。
   (II) JavaScript 関数「getElementById(“ html 部品名称 ”)」を使用して，データ転送先の HTML オブジェクトを取得します。
   (III) (I) で取得した JavaScript 連想配列オブジェクトを展開して，値を取得します。
   (IV) (II) で取得した HTML オブジェクトの value に (III) の値を格納します。

3. <csdc:body> タグ内にデータ転送元の HTML 構成を定義します。
(l) ［貼り付け］ボタンをクリックすると実行される関数を作成し，その関数内でペーストイベントハンドラを呼ぶ関数 (hptl_onpaste) を定義します。
複数 datatype を指定する場合は，第一引数にコンマ区切りの文字列を指定してください。
ペーストイベントハンドラを呼ぶ関数 (hptl_onpaste) の詳細は，「14.12　クライアントサイドデータ通信 JavaScript」を参照してください。

注意事項

- ［コピー］ボタンがクリックされないで［貼り付け］ボタンがクリックされた場合は何も処理をしません。
- データ転送先画面の定義 <csdc:dest> の datatype が，データ転送元画面の定義 <csdc:src> の datatype に含まれていない場合は何も処理をしません。

9.5.2　定義例

(1) データ転送元画面

例の中の番号は，「9.5.1　定義規則」の該当個所を示します。

```
<csdc:src datatype="Employee">                                          ・・・(1)-1
    <csdc:oncopy name="copy">
        var items = []; // データソース格納配列
        // HTMLオブジェクトからユーザ選択データを抽出
        var list = document.getElementById("addrlist");                 ・・・(1)-2-(I)
        n = 0;
        for (var i = 0; i < list.options.length; i++) {
          if  (list.options[i].selected) {
           items[n] = hptl_deserializeH2JS("bean", list.options[i].value); ・・(1)-2-(II)
           n++;
            }
        }
        // userData領域またはポータルクリップボードにitemsを設定
        hptl_setPortalClipboardData("Employee", items);
    </csdc:oncopy>
    <csdc:body>
        従業員一覧(Humanオブジェクト)
        <select id="addrlist" multiple="multiple">
<%  for (int i = 0; i < e.length; i++) { %>
      <option value="
       <csdc:serializeJ2H type="bean" data="<%= e[i] %>"/>       ・・・(1)-3-(I)
      ">
<%= e[i].getName() %></option>
<%  } %>
        </select>
<script language=" JavaScript" >                                        ┐
function oncopy() {                                                     ・・・(1)-3-(II)
var r= hptl_oncopy( "Employee" ,copy,null);                             ┘
}
</script>
        <input type=button value="コピー" onClick="javascript:oncopy();">
    </csdc:body>
</csdc:src>
```

(2) データ転送先

例の中の番号は,「9.5.1　定義規則」の該当個所を示します。

```
<form>
    <csdc:dest datatype="Employee" nodetype="inline">                ・・・(2)-1
        <csdc:onpaste name="paste">
            // userData領域またはポータルクリップボードからアイテムを取得
            var items = hptl_getPortalClipboardData( “Employee” );   ・・・(2)-2-(I)
            var obj = document.getElementById("to");                 ・・・(2)-2-(II)
            var result = "";
            if(items != null){
                for (var i = 0; i < items.length; i++) {
                    var item = items[i];                             ・・(2)-2-(III)
                    result += item["name"] + "<" + item["mail"] +">;";
                }
                obj.value = result;                                  ・・(2)-2-(IV)
            }
 </csdc:onpaste>
 <csdc:body>
  <input id="to" type="text" name="to" size="50"> <br>
<script language=" JavaScript" >
 function onpaste() {                                                ・・・(2)-3-(I)
  hptl_onpaste( “Employee” ,paste,null);
}
</script>
 </csdc:body>
</csdc:dest>
```

10 ナビゲーションメニュー対応ポートレットの開発

この章では，ナビゲーションメニューにメニューとして登録できるポートレットの開発方法について説明します。

10.1　ナビゲーションメニュー対応ポートレットを開発する前に

ここでは，ナビゲーションメニューに対応するポートレットを開発する前に知っておいていただきたい内容について説明します。

10.1.1　ナビゲーションメニュー対応ポートレットに設定する内容

ナビゲーションメニューに登録できるポートレットを開発するには，あらかじめ次の項目を開発してください。

- メニューに対応するための開発項目
- リンク集に対応するための開発項目

(1) メニューに対応するための開発項目

ポータルサーバは，ナビゲーションメニュー内にポートレット画面を呼び出すアンカーを自動的に生成します。このアンカーのことを**最大化画面アンカー**といいます。

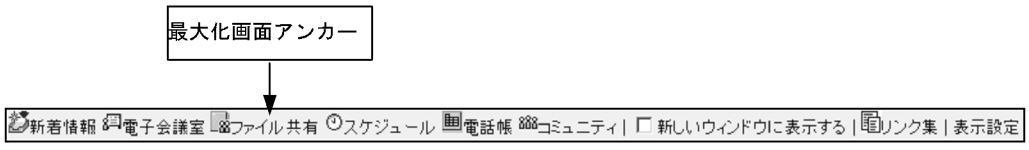
ポータルサーバのこの機能を利用すると，次の項目を開発または追記するだけでナビゲーションメニュー内に最大化画面アンカーを表示できます。

1. 最大化画面モードおよび新規ウィンドウ画面モードに対応したポートレット画面の開発
2. デプロイ定義ファイル（hportlet.xml）にナビゲーションメニューに対応させるためのパラメタを追記

新規ウィンドウ画面モードに対応したポートレット画面開発の詳細は，「12.　新規ウィンドウ対応ポートレットの開発」を参照してください。デプロイ定義ファイルの設定については，「10.1.2　デプロイ定義ファイルの設定」を参照してください。

最大化画面アンカーを表示しているナビゲーションメニューを次の図に示します。

図 10-1　最大化画面アンカーを表示しているナビゲーションメニュー



最大化画面アンカーの左側には，対応ポートレットが指定したアイコンが表示されます。ナビゲーションメニューに対応しないポートレット，またはアイコンを指定していない対応ポートレットの場合は，ナビゲーションメニューが提供するデフォルトアイコンが表示されます。

対応ポートレットは，ポートレットのタイトルバーの最大化ボタンをクリックすることによって表示される場合と同様に，ナビゲーションメニューから呼ばれた場合でも，画面モードを取得できます。

ナビゲーションメニューから呼ばれた場合，ナビゲーションメニュー画面上の最大化画面アンカーの URL リンク内容には，次の**クエリストリング**が付け加えられます。

表 10-1　ナビゲーションメニューが付加するクエリストリング

変数	値	説明
hptl_referer	navi	ナビゲーションメニューの操作によってリクエストが発生しました。

対応ポートレットでこのクエリストリングの有無を判定することによって，統合画面の最大化ボタンがクリックされた場合か，またはナビゲーションメニューの最大化画面アンカーがクリックされた場合かの区別ができるようになります。また，統合画面の最大化ボタンがクリックされた場合，またはナビゲーションメニューの最大化画面アンカーがクリックされた場合で，画面に表示する内容を変更できます。

なお，リンク集に登録された特定の画面へのアンカーでは，クエリストリングは付け加えられません。これは，特定の画面を含むポートレットで，任意に URL リンク内容を設定できるためです。このため，リンク集から表示した場合は，その画面がどこから表示されたか判定できません。

(2) リンク集に対応するための開発項目

ポートレット中の特定の画面をリンク集に登録できるようにするには，ナビゲーションメニューが提供するタグライブラリを使用してポートレットを開発します。タグライブラリの詳細は，「14.15　ナビゲーションメニューのタグライブラリ」を参照してください。

10.1.2　デプロイ定義ファイルの設定

ナビゲーションメニューは，ポートレットをデプロイするときの設定値を参照して，ナビゲーションメニュー対応しているかどうかを判定します。ここでは，ナビゲーションメニュー対応ポートレットのデプロイ定義ファイルに記述する項目について説明します。

ナビゲーションメニュー対応ポートレットのデプロイ定義ファイル（ファイル名は hportlet.xml）に記述する項目を，次の図に示します。

図 10-2 ナビゲーションメニュー対応ポートレットのデプロイ定義ファイルに記述する項目

<portlet-app>	このファイルのルートタグです。
<portlet>	ポートレット情報の親タグです。
＜中略＞※	＜中略＞※
<config-param>	ポートレット起動時のパラメタを設定します。 この設定は複数指定できます。
<param-name> hptl.navigationbar.support </param-name>	ポートレット対応状況を示すキーです。 この項目の設定は必須です。 ナビゲーションメニューに対応するポートレットは，新規ウィンドウを表示できるポートレットが前提となっています。
<param-value> true </param-value>	true：ナビゲーションメニュー対応 false：ナビゲーションメニュー非対応 省略したときは，falseが設定されます。
<param-name> hptl.navigationbar.version </param-name>	対応しているナビゲーションメニューのAPIバージョンを示すキーです。 ナビゲーションメニューに対応しているポートレットは，この設定が必須です。
<param-value> 01-00 </param-value>	現在のナビゲーションメニューのAPIバージョンである「01-00」を指定します。 省略したときは，「01-00」が設定されます。
<param-name> hptl.navigationbar.alltabflag </param-name>	連携ポートレットをデフォルトメニューに設定するキーです。 ナビゲーションメニューに対応しているポートレットは，この設定は任意です。
<param-value> true </param-value>	true：ユーザにメニュー情報がない場合，デフォルトメニューとしてメニューに登録 false：省略した場合に設定
<param-name> hptl.portlet.defaulticon </param-name>	ナビゲーションメニューに表示するデフォルトのアイコンを指定します。
<param-value> images/image.jpg </param-value>	アイコンファイルのURIを指定します。 ポートレットがエントリされているディレクトリからの相対パスを指定します。
＜中略＞※	＜中略＞※

注※ タグの詳細は，「3.13.2 デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。

注意事項

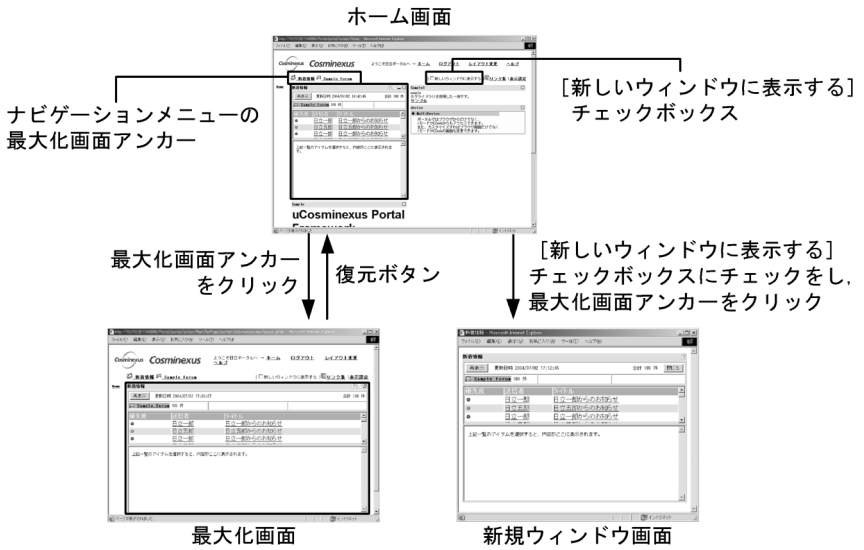
hptl.navigationbar.support 項目の設定は必須です。

hptl.navigationbar.support 項目に true を設定した場合は，ほかの項目が設定されていなくてもナビゲーションメニュー対応ポートレットと判定されます。また，hptl.navigationbar.support 項目に false を設定した場合は，ほかの項目が設定されていてもナビゲーションメニュー非対応ポートレットと判定されます。

10.2　ナビゲーションメニュー対応ポートレットの画面遷移

ナビゲーションメニューからナビゲーションメニュー対応ポートレットの画面呼び出しがある場合の画面遷移図を次に示します。

図 10-3　ナビゲーションメニューからナビゲーションメニュー対応ポートレットの画面遷移図



ナビゲーションメニューから最大化画面アンカーをクリックしてポートレットを呼び出すと，クエリストリング（hptl_referer=navi）がURLリンク内容に付け加えられます。呼び出されたポートレットは，最大化画面に表示されます。

［新しいウィンドウに表示する］チェックボックスにチェックをした状態で，ナビゲーションメニューから最大化画面アンカーをクリックしてポートレットを呼び出すと，クエリストリング（hptl_referer=navi）がURLリンク内容に付け加えられます。呼び出されたポートレットは，新規ウィンドウ画面に表示されます。

10.3　ナビゲーションメニュー対応ポートレットの開発例

uCosminexus Portal Framework で提供しているサンプルを基に，ナビゲーションメニューに対応するポートレットの開発方法について説明します。

10.3.1　サンプルの提供方法および動作確認方法

(1) サンプルの提供方法

uCosminexus Portal Framework は，par 形式ファイルの「sampleforum.par」をサンプルとして提供しています。

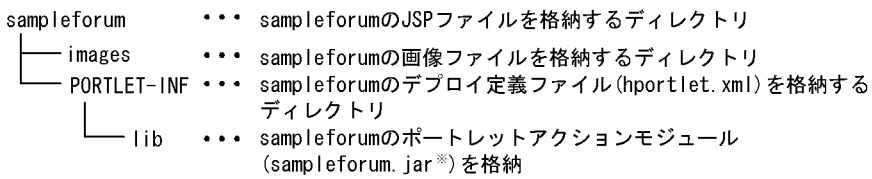
「sampleforum.par」の格納場所を次に示します。

```
{uCosminexus Portal Frameworkインストールディレクトリ}
¥samples¥SampleForum¥portlets
```

「sampleforum.par」を解凍すると，プログラムソースを参照できます。

「sampleforum.par」を解凍したあとのディレクトリ構成およびファイル構成を次の図に示します。

図 10-4 「sampleforum.par」のディレクトリ構成およびファイル構成



注※　解凍するとJavaソースとクラスを参照できます。

「sampleforum」ディレクトリに格納されている JSP ファイルを次に示します。

- done.jsp
- edit.jsp
- index.jsp
- main.jsp
- view.jsp

このサンプルでは，view.jsp を使用します。view.jsp の詳細は，「10.3.2　画面の開発」を参照してください。

(2) サンプルの動作確認方法

サンプルを使用して動作を確認するには，「sampleforum.par」をデプロイする必要があります。「sampleforum.par」をデプロイすると，サンプルは次の表に示すようにポータルサーバに登録されます。

表 10-2　デプロイされるポートレット

ポートレット名	ポートレットのタイトル※（英語／日本語）	ポートレットが提供する API クラス
sampleforum	Sample forum ／ Sample forum	sampleforum.par 内の hportlet.xml を参照してください。

注※　デプロイしたときのデフォルト値です。

このサンプルは通常の日立 API ポートレットです。そのため，特別なライブラリなどの前提条件はありません。

デプロイの方法については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「日立 API ポートレット（PAR 形式）のデプロイ」の説明を参照してください。

10.3.2　画面の開発

(1) view.jsp

表示している画面をリンク集に登録するためのボタンおよびアンカーを開発します。ボタンやアンカーを開発するには，<navimenu:linkregisturl> タグを使用します。

このサンプルでは，［リンク集に登録］がリンク集に登録するためのアンカーとなります。

view.jsp を新規ウィンドウモードで表示した場合の画面を次の図に示します。

図 10-5　view.jsp を新規ウィンドウモードで表示した画面

この画面上で［リンク集に登録］アンカーをクリックしたあとに，ナビゲーションメニューが表示する画面を次の図に示します。

図 10-6 ［リンク集登録確認画面］

10.3.3 デプロイ定義ファイルの開発

ポートレットの動作を設定するために，デプロイ定義ファイルを開発します。このサンプルのデプロイ定義ファイル（hportlet.xml）の内容を次に示します。ここで，ポートレットをナビゲーションメニューに対応するために必要な項目（<param-name> および <param-value>）を設定しています。

サンプルのデプロイ定義ファイル（hportlet.xml）の内容

```
<portlet-app>
    <portlet>
                ＜中略＞
        <config-param>
            <param-name>hptl.portlet.defaulticon</param-name>
            <param-value>images/forum.gif</param-value>
            <param-name>hptl.inbox.support</param-name>
            <param-value>true</param-value>
          <param-name>hptl.inbox.filtering.support</param-name>
            <param-value>true</param-value>
            <param-name>hptl.navigationbar.support</param-name>
            <param-value>true</param-value>
        </config-param>
                ＜中略＞
    </portlet>
</portlet-app>
```

各項目の内容を説明します。

- hptl.portlet.defaulticon
  ナビゲーションメニューの最大化画面アンカーの左側に表示する，デフォルトアイコンの URL を指定します。
- hptl.inbox.support
  ポートレットを Information View に対応させるかどうかを設定します。Information View に対応させる場合は，true を設定します。
- hptl.inbox.filtering.support
  ポートレットを新着情報の取得条件を設定する機能に対応させるかどうかを設定します。新着情報の取得条件を設定する機能に対応させる場合は，true を設定します。
- hptl.navigationbar.support
  ポートレットをナビゲーションメニューに対応させるかどうかを設定します。ナビゲーションメニューに対応させる場合は，true を設定します。ナビゲーションメ

ニュー対応ポートレットでは，この項目の設定は必須です。

なお，デプロイ定義ファイルの詳細は，「10.1.2　デプロイ定義ファイルの設定」を参照してください。

11 言語およびタイムゾーン切り替え対応ポートレットの開発

この章では，言語およびタイムゾーンの切り替え機能に対応したポートレットの開発方法について説明します。

11.1　言語切り替え対応ポートレットの開発方法

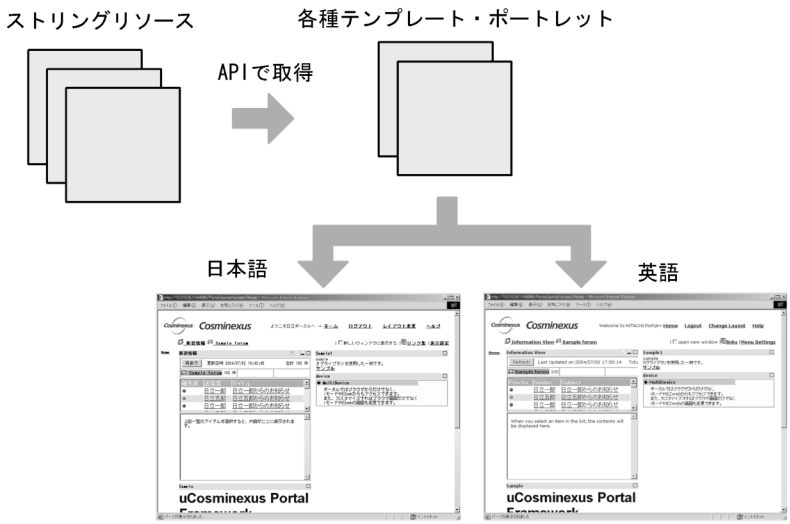
ここでは，個人ごとに使用する言語およびタイムゾーンを切り替えられるポートレットの開発方法を説明します。

11.1.1　言語切り替え対応ポートレットの開発

言語切り替え対応ポートレットでは，uCosminexus Portal Framework が提供する API によって使用する必要がある言語を判別し，最適な言語でコンテンツを作成します。uCosminexus Portal Framework は，標準で日本語と英語に対応していますが，システム管理者の設定により中国語等のほかの言語に対応することが可能です。使用する言語の追加については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ストリングリソースのカスタマイズ」の説明を参照してください。

言語切り替え対応ポートレットの処理概要を次の図に示します。

図 11-1　言語切り替え対応ポートレットの処理概要



ポートレットでは，次の処理を行います。

- 使用する言語の判定
  使用する言語は，ユーザが使用している Web ブラウザの言語設定か，またはポータル画面の利用者用レイアウトカスタマイズ画面で設定されます。
  言語切り替え対応ポートレットでは，ユーザロケール情報取得 API の LocaleData クラスを使用して，設定された言語の識別子を取得します。
- 該当する言語のストリングの作成
  ストリングリソース取得 API 等を使用して，LocaleData クラスが示した判定に応じ

た適切なストリングを作成します。

ユーザロケール情報取得 API の詳細は,「14.16 ユーザロケール情報取得 API」を参照してください。また,ストリングリソース取得 API の詳細は,「14.17 ストリングリソース取得 API」を参照してください。

11.1.2 言語切り替え対応ポートレットの開発例

使用する言語を判定する JSP ファイルのサンプルコードを次に示します。

サンプルコード

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean
" %>
<%@ page import="jp.co.hitachi.soft.portal.api.user.LocaleData"
%>
<jsp:useBean id="configbean"
class="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
scope="page"/>

<%
 configbean.initBean(request);

 String langType = LocaleData.getLangType(request) ;

 if(langType.equals("ja"))
  configbean.setCustomizeTitle("日本語用のタイトルメニューです") ;
 else if(langType.equals("en"))
  configbean.setCustomizeTitle("For English Title Menu") ;
 else
  configbean.setCustomizeTitle("Other Language Title Menu") ;

%>
サンプルポートレット
```

11.2　タイムゾーン切り替え対応ポートレットの開発方法

11.2.1　タイムゾーン切り替え対応ポートレットの開発

タイムゾーン切り替え対応ポートレットでは，使用するタイムゾーンを取得・設定して，適切な時刻をポートレットに表示します。

使用するタイムゾーンは，ユーザが使用している Web ブラウザの言語設定か，またはポータル画面の利用者用レイアウトカスタマイズ画面で設定されます。また，タイムゾーンの識別には，J2SE1.4 で定めるカスタムタイムゾーン ID を使用します（サマータイムを考慮した地域は，サポートされません）。

タイムゾーン切り替え対応ポートレットでは，ユーザロケール情報取得 API で使用するタイムゾーンを取得・設定します。ユーザロケール情報取得 API の詳細は，「14.16 ユーザロケール情報取得 API」を参照してください。

11.2.2　タイムゾーン切り替え対応ポートレットの開発例

ユーザのタイムゾーンに応じて日時を表示する JSP ファイルのサンプルコードを次に示します。

サンプルコード

```
<%@ page import="jp.co.hitachi.soft.portal.api.user.LocaleData"
%>
<%@ page import="java.util.Date" %>
<%@ page import="java.util.TimeZone" %>
<%@ page import="java.text.SimpleDateFormat" %>
<%
SimpleDateFormat formatter = new SimpleDateFormat("yyyy/MM/dd
HH:mm:ss z",LocaleData.getLocale(request));
formatter.setTimeZone(TimeZone.getTimeZone(LocaleData.getTimeZo
ne(request)));
String dateString = formatter.format(new Date());
%>
サンプルポートレット<%=dateString%>
```

12 新規ウィンドウ対応ポートレットの開発

uCosminexus Portal Framework には，ポートレットを新規ウィンドウ画面で表示する機能があります。この章では，ナビゲーションメニューから表示する場合などに新規ウィンドウに表示できるポートレットの開発方法について説明します。

12.1　新規ウィンドウ対応ポートレットを開発する前に

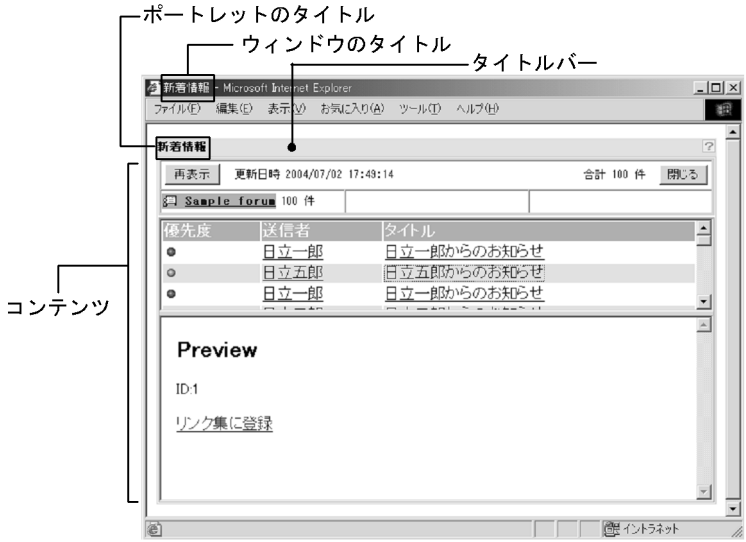
ここでは，新規ウィンドウ対応ポートレットを開発する前に，知っておいていただきたい内容について説明します。

12.1.1　新規ウィンドウの画面構成

新規ウィンドウ画面は，ポートレット開発者が作成したコンテンツにタイトルバーを付けた構成になります。このタイトルバーにポートレットのタイトルが表示されます。新規ウィンドウ画面は，JavaScript の window.open などで作成した Web ブラウザウィンドウ画面内に統合画面を表示する場合に適しています。

新規ウィンドウ画面の例を次の図に示します。

図 12-1　新規ウィンドウ画面



新規ウィンドウ画面のタイトル

新規ウィンドウ画面では，次の二つのタイトルを登録できます。

- ポートレットのタイトル
  ポートレットのタイトルを登録して，新規ウィンドウ画面のタイトルバーに表示できます（最大化画面と同様）。
- ウィンドウのタイトル
  ウィンドウのタイトルを登録して，Web ブラウザのタイトルバーに表示できます。

なお，PortletInfoBean の setCustomizeTitle メソッドを使用すると，ポートレットのタイトルの表示を動的にできます。PortletInfoBean の setCustomizeTitle メソッ

ドについては，「14.5　ポートレット情報取得 Bean」の「jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean」を参照してください。ポートレットのタイトルの登録方法については，「3.13.2　デプロイ定義ファイル（hportlet.xml）の作成」を参照してください。

12.1.2　新規ウィンドウ対応ポートレットの開発上の注意

新規ウィンドウ画面を開発する場合は，次の点に注意して開発してください。

- 新規ウィンドウ画面内に不正なリンク（新規ウィンドウ画面から標準画面／最大化画面／編集画面へのリンク）を出現させないようにしてください。
- 利用者のコンピュータ上に複数の「標準画面／最大化画面／編集画面」が表示されないようにしてください。複数の「標準画面／最大化画面／編集画面」が表示された場合，一つの画面を編集したときに，表示されている複数の画面上の情報に矛盾が発生するおそれがあります。

12.2 新規ウィンドウ対応ポートレットの開発例

ここでは，新規ウィンドウ対応ポートレットの開発例について説明します。

12.2.1 新規ウィンドウ画面へのリンクを表示するコンテンツ

新規ウィンドウ画面へのリンクを表示する日立 API ポートレットのソース（default.jsp）のコーディング例を次に示します。

コーディング例（default.jsp）

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>

<H1>デフォルト画面</H1>

<%
PortletInfoBean configBean = new PortletInfoBean();
configBean.initBean(request);
%>

<!% メール新規作成画面を新規ウィンドウ画面モードで表示するリンク %>
<A HREF="<ut:portletURI name="<%=configBean.getName()%>"
mode="<%=PortletInfoBean.Mode.NEWWINDOW.toString()%>"
parameter="controller=NEWMAIL" />" target="_blank">メール新規作成</A>
```

12.2.2 新規ウィンドウ画面を表示するコンテンツ

新規ウィンドウ画面モードで表示するリンクがクリックされた際に，新規ウィンドウ画面を表示する日立 API ポートレットのソース（Controller.jsp）のコーディング例を次に示します。このサンプルでは，最大化画面モード時および新規ウィンドウ画面モード時のコンテンツを開発します。また，新規ウィンドウ画面モード時には，ポートレットのタイトルをカスタマイズしています。

コーディング例（Controller.jsp）

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>

<%
    PortletInfoBean pInfo = new PortletInfoBean();
    pInfo.initBean(request);
    PortletInfoBean.Mode pMode = pInfo.getMode();
```

```
%>
<%
    // 標準画面モード
    view = "default.jsp";

    // 画面モード判定
    if (pMode == PortletInfoBean.Mode.MAXIMIZE) {
        // 最大化画面モード
        view = "maximized.jsp";
    } else if (pMode == PortletInfoBean.Mode.NEWWINDOW) {
        // 新規ウィンドウ画面モード
        pInfo.setCustomizeTitle("メール新規作成画面");
        view = "newwindow.jsp";
    }
%>
<jsp:include page="<%= view %>" flush="true"/>
```

default.jsp

```
<hr>
```
標準画面モード時に表示する画面です。

maximized.jsp

```
<hr>
```
最大化画面モード時に表示する画面です。

newwindow.jsp

```
<hr>
```
新規ウィンドウモード時に表示する画面です。

13 カスタマイズ情報の任意保存対応ポートレットの開発

この章では，カスタマイズ情報の任意保存機能（カスタマイズ情報を任意のタイミングで保存できる機能）に対応したポートレットの開発について説明します。

13.1 カスタマイズ情報任意保存機能の使用方法

カスタマイズ情報は，通常ログアウト時に保存されますが，カスタマイズ情報任意保存機能を使用すると，ポートレット上で任意のタイミングで保存できるようになります。

カスタマイズ情報任意保存機能を使用する場合，カスタマイズ情報を追加・変更する処理部分に「排他区間」を設定する必要があります。

なお，排他区間の開始処理を行った場合は，必ず排他区間の終了処理をしてください。

ポートレット並列化機能を使用している場合

ポートレット並列化機能を使用している場合，自身のポートレットだけではなく，ほかのポートレットも同時にカスタマイズ情報にアクセスします。このため，各ポートレットで排他区間を必ず設定してください。排他区間を設定すると，排他区間中はほかのポートレットの保存処理が保留されます。

ポートレット並列化機能を使用している場合の，排他の仕様を次の表に示します。

表 13-1 ポートレット並列化機能を使用している場合の排他の仕様

自身のポートレットの状態	ほかのポートレットの状態	
	排他区間中	保存処理中
排他区間中	○	×
保存処理中	×	×

（凡例）

○：並列実行できます。

×：並列実行できません。

一方のポートレットが排他区間中の場合は，他方のポートレットも排他区間を開始できます。

一方のポートレットが追加・変更したカスタマイズ情報を保存処理中の場合は，他方のポートレットが排他区間を開始することはできません。また，他方のポートレットの保存処理は，排他区間が終了するまで保留されます。

13.2　カスタマイズ情報任意保存機能使用時のポートレットの開発例

ここでは，カスタマイズ情報任意保存機能を使用する場合のポートレットの開発例を説明します。

カスタマイズ情報任意保存機能を使用する場合，変更区間処理全体を try ブロックで囲って，ブロック内の先頭で排他区間の開始処理を記述し，finally ブロックで排他区間の終了処理を記述します。

日立 API ポートレットでのコーディング例を次に示します。

コーディング例

```
  <jsp:useBean id="UserInfo"
class="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean"
scope="page"/>
  <%
  UserInfo.setRequest(request);

  try{

    UserInfo.customizeInfoStart();  //排他区間開始
    UserInfo.setCustomizeInfo("DataA","UserID:ABC");
    UserInfo.setCustomizeInfo("DataB","UserPassword:XYZ");
    UserInfo.setCustomizeInfo("DataC","UserOu:XYZ");

   }finally{
     UserInfo.customizeInfoEnd();     //排他区間終了

   }

   //LDAP/DBへのデータの保存処理
   boolean isSave = UserInfo.commitData();
   if( isSave ){
    out.print("データ保存に成功しました");

   }else{
     out.print("データ保存に失敗しました");

   }
  %>
```

14 ライブラリ

この章では，uCosminexus Portal Framework で提供するライブラリについて説明します。

14.1 ポートレットを開発するためのライブラリ

uCosminexus Portal Framework では次のライブラリを使用します。

ポートレット API

ポートレットを作成するときのライブラリです。

- ポートレットユティリティタグライブラリ（14.2）
- ポートレットモードタグライブラリ（14.3）
- ポートレットユティリティクラスライブラリ（14.4）
- ポートレット情報取得 Bean（14.5）
- ユーザ情報取得 Bean（14.6）
- ログ出力 Bean（14.7）
- UserAgent 情報取得 Bean（14.8）
- ポートレットイベント API（14.9）
- ポートレットパラメタ取得インタフェース（14.10）

クライアントサイドデータ通信 API

クライアントサイドデータ通信機能を実現するためのライブラリです。

- クライアントサイドデータ通信タグライブラリ（14.11）
- クライアントサイドデータ通信 JavaScript（14.12）
- クライアントサイドデータ通信クラスライブラリ（14.13）

ダイレクト呼び出し結果取得 API

ダイレクト呼び出し実行時のデータの引き継ぎ結果を取得するためのライブラリです。

- ダイレクト呼び出し結果取得 API（14.14）

ナビゲーションメニュー API

ナビゲーションメニューに対応したポートレット作成するためのライブラリです。

- ナビゲーションメニューのタグライブラリ（14.15）

ユーザロケール情報取得 API

ユーザのロケール情報（言語識別子）を取得，設定するためのライブラリです。

- ユーザロケール情報取得 API（14.16）

ストリングリソース取得 API

ストリングリソースを取得するときに使用するライブラリです。

- ストリングリソース取得 API（14.17）

セルフユーザ登録ポートレット用 API

セルフユーザ登録ポートレットを使用するためのライブラリです。

- ユーザ登録 API（14.18）

- ユーザアカウント情報取得 API（14.19）

カスタムウィンドウステート API

標準 API ポートレットでカスタムウィンドウステートを使用するためのライブラリです。

- カスタムウィンドウステート API（14.20）

拡張 Struts タグライブラリ

標準 API ポートレットで Struts フレームワークを使用する場合に使用するタグライブラリです。

- 拡張 Struts タグライブラリ（14.21）

14.2 ポートレットユティリティタグライブラリ

ポートレットユティリティタグライブラリの一覧を次の表に示します。

表 14-1 ポートレットユティリティタグライブラリの一覧

タグ名	説明	JSP スクリプトレット式で評価される属性
ut:cache	日立 API ポートレットの出力するコンテンツの一部に対し，キャッシュを使用します。	name，time，scope
ut:convert	XSLT スタイルシートに基づいて，XML データをポートレットに取り込みます。	xslt
ut:logindata	ログイン画面でダイレクト呼び出しパラメタおよび POST データを保持する <HIDDEN> タグを生成します。	－
ut:portletURI	登録されているポートレットを，画面モードを指定して呼び出すためのアンカーを作成します。	name，mode，parameter
ut:secureURI	現在の URL のプロトコルを「HTTPS」に変更し，コンテンツキャッシュ削除の指定を追加して出力します。	－
uu:a	HTML および CHTML では，href 属性を URL 変換します。	href，lang，name，onblur，onclick，ondblclick，onfocus，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，shape，target
	HDML では，dest 属性を URL 変換します。	dest，next，cancel
uu:action	dest，next，cancel 属性を URL 変換します。	dest，next，cancel
uu:area	href 属性を URL 変換します。	href，alt，lang，name，onblur，onclick，ondblclick，onfocus，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，shape，target
uu:ce	ce 要素の dest 属性を URL 変換します。	dest，next，cancel
uu:choice	bookmark 属性を URL 変換します。終了タグの直前にログイン／ログアウトアンカーを挿入します。	－

タグ名	説明	JSP スクリプトレット式で評価される属性
uu:display	bookmark 属性を URL 変換します。終了タグの直前にログイン／ログアウトアンカーを挿入します。	－
uu:entry	bookmark 属性を URL 変換します。開始タグの直後にログイン／ログアウトアンカーを挿入します。	－
uu:form	action 属性を URL 変換します。	action，accept，accept_charset，enctype，lang，name，onclick，ondblclick，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，onreset，onsubmit，target
uu:frame	src，longdesc 属性を URL 変換します。	src，longdesc，frameborder，marginheight，marginwidth，name，noresize，scrolling，target
uu:iframe	src，longdesc 属性を URL 変換します。	src，longdesc，align，frameborder，height，marginheight，marginwidth，name，scrolling，width
uu:img	src，longdesc 属性を URL 変換します。longdesc 属性は HTML の場合だけ URL 変換します。	src，longdesc，alt，align，border，height，hspace，lang，name，onclick，ondblclick，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，usemap，vspace，width
uu:input	src 属性を URL 変換します。	src，accept，align，alt，lang，name，onblur，onchange，onclick，ondblclick，onfocus，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，onselect，size，usemap，value
uu:object	data，archive，classid，codebase 属性を URL 変換します。	codebase，archive，classid，data，align，border，height，hspace，lang，name，onclick，ondblclick，onkeydown，onkeypress，onkeyup，onmousedown，onmousemove，onmouseout，onmouseover，onmouseup，type，usemap，vspace，width
uu:script	src 属性を URL 変換します。	src，type

（凡例）

－：該当しません。

注意事項

ポートレットユーティリティクラスライブラリの jp.co.hitachi.soft.portal.portlet.PortletURI クラスを使用して取得した URL をポートレットユーティリティタグライブラリの URL 指定属性に指定すると，一度変換された URL に対して再度 URL 変換が行われるため，変換後の URL が不正となる場合があります。

ut:cache

機能

日立 API ポートレットの出力するコンテンツの一部に対し，キャッシュを使用します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>
<ut:cache name="CacheKey" time="3" scope="session">
キャッシュを使用するコンテンツ
</ut:cache>
```

対応する言語

HTML，CHTML，HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
name	キャッシュ内でのキー文字列を指定します。	必須	○
time	キャッシュの有効期間を指定します。単位は分です。	任意	○
scope	キャッシュの有効範囲を指定します。session だけに対応します。	必須	○

（凡例）

○：JSP スクリプトレット式で評価されます。

ut:convert

機能

XSLT スタイルシートに基づいて，XML データをポートレットに取り込みます。このタグ内には XML 形式のデータを記述します。

構文

```
<%@ taglib uri=" http://soft.hitachi.co.jp/portal/utils"
prefix="ut" %>
<ut:convert xslt = "html_default.xsl">
XSLTスタイルシートに基づいて取り込むXMLデータ
</ut:convert>
```

対応する言語

HTML，CHTML，HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
xslt	タグ内の XML データを変換する XSLT スタイルシート名を指定します。	任意	○

（凡例）
○：JSP スクリプトレット式で評価されます。

ut:logindata

機能

ログイン画面でダイレクト呼び出しパラメタおよび POST データを保持する <HIDDEN> タグを生成します。

このタグライブラリは，ログイン画面の FORM 内に定義します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>
<ut:logindata/>
```

対応する言語

HTML，CHTML，HDML

属性

なし

ut:portletURI

機能

jetspeed-config.jcfg ファイルに登録されているポートレットを，画面モードを指定して呼び出すためのアンカーを作成します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>
...
<a href="<ut:portletURI name='portletA'/ >">portletA</a>
```

対応する言語

HTML，CHTML，HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
name	呼び出されるポートレット名を指定します。省略した場合は呼び出したポートレットとなります。	任意	○
mode	呼び出す画面モードを示す文字列 MAXIMIZE，NEWWINDOW，および IFRAME を指定します。省略時は呼出元の画面モードとなります。ただし，呼出元の画面モードが DEFAULT である場合は MAXIMIZE になります。	任意	○
parameter	クエリ文字列（URL エンコード済みの文字列）を指定します。"?" の付加は不要です。省略時はクエリなしとして扱います。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

例外

JSPException - 次の場合に発生します。

● mode 引数の値が MAXIMIZE，NEWWINDOW，および IFRAME 以外の場合

ut:secureURI

機能

現在の URL のプロトコルを「HTTPS」に変更し，コンテンツキャッシュ削除の指定を追加して出力します。現在の URL がすでに HTTPS プロトコルの場合は，現在の URL

をそのまま表示します。

URL のホスト名とポート番号は，HTTPS ドメイン名プロパティの設定値を利用します。

構文

```
<%@ taglib uri='http://soft.hitachi.co.jp/portal/utils' prefix='ut'
%><%@ page contentType="text/html; charset=Shift_JIS" %>
...
<A HREF="<ut:secureURI/ >">portletA</A>
```

対応する言語

HTML，CHTML，HDML

uu:a

機能

HTML，CHTML では，a 要素の href 属性を URL 変換します。

HDML では，a 要素の dest 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:a href="url" ...>
...
</uu:a>
```

対応する言語

HTML，CHTML，HDML

属性（HTML，CHTML）

属性	説明	必須 / 任意	JSP スクリプトレット式評価
href	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
accesskey	HTML4.01 の仕様に従います。	任意	×
charset	HTML4.01 の仕様に従います。	任意	×
coords	HTML4.01 の仕様に従います。	任意	×
dir	HTML4.01 の仕様に従います。	任意	×
hclass ※1	HTML4.01 の仕様に従います。	任意	×
hid ※2	HTML4.01 の仕様に従います。	任意	×

属性	説明	必須 / 任意	JSP スクリプトレット式評価
hreflang	HTML4.01 の仕様に従います。	任意	×
lang	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
onblur	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onfocus	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
rel	HTML4.01 の仕様に従います。	任意	×
rev	HTML4.01 の仕様に従います。	任意	×
shape	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
tabindex	HTML4.01 の仕様に従います。	任意	×
target	HTML4.01 の仕様に従います。	任意	○
title	HTML4.01 の仕様に従います。	任意	×
type	HTML4.01 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

属性（HDML）

属性	説明	必須 / 任意	JSP スクリプトレット式評価
dest	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
next	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
cancel	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
accesskey	HDML3.3 の仕様に従います。	任意	×
accept_charset	HDML3.3 の仕様に従います。	任意	×
clear	HDML3.3 の仕様に従います。	任意	×
friend	HDML3.3 の仕様に従います。	任意	×
label	HDML3.3 の仕様に従います。	任意	×
method	HDML3.3 の仕様に従います。	任意	×
number	HDML3.3 の仕様に従います。	任意	×
postdata	HDML3.3 の仕様に従います。	任意	×
receive	HDML3.3 の仕様に従います。	任意	×
rel	HDML3.3 の仕様に従います。	任意	×
retvals	HDML3.3 の仕様に従います。	任意	×
sendreferer	HDML3.3 の仕様に従います。	任意	×
task	HDML3.3 の仕様に従います。	任意	×
vars	HDML3.3 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

uu:action

機能

action 要素の dest 属性，next 属性，および cancel 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:action task="gosub" dest="url" next="url" cancel="url"/>
```

対応する言語

HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
dest	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
next	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
cancel	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
accept_charset	HDML3.3 の仕様に従います。	任意	×
clear	HDML3.3 の仕様に従います。	任意	×
friend	HDML3.3 の仕様に従います。	任意	×
icon	HDML3.3 の仕様に従います。	任意	×
label	HDML3.3 の仕様に従います。	任意	×
method	HDML3.3 の仕様に従います。	任意	×
number	HDML3.3 の仕様に従います。	任意	×
postdata	HDML3.3 の仕様に従います。	任意	×
receive	HDML3.3 の仕様に従います。	任意	×
rel	HDML3.3 の仕様に従います。	任意	×
retvals	HDML3.3 の仕様に従います。	任意	×
sendreferer	HDML3.3 の仕様に従います。	任意	×
src	HDML3.3 の仕様に従います。	任意	×
task	HDML3.3 の仕様に従います。	任意	×
type	HDML3.3 の仕様に従います。	任意	×
vars	HDML3.3 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

uu:area

機能

area 要素の href 属性を URL 変換します。なお，nohref 属性は使用できません。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:area href="url" alt="sample" ... />
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
href	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
alt	HTML4.01 の仕様に従います。	必須	○
accesskey	HTML4.01 の仕様に従います。	任意	×
hclass ※1	HTML4.01 の仕様に従います。	任意	×
coords	HTML4.01 の仕様に従います。	任意	×
dir	HTML4.01 の仕様に従います。	任意	×
hid ※2	HTML4.01 の仕様に従います。	任意	×
lang	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
onblur	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onfocus	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
shape	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
tabindex	HTML4.01 の仕様に従います。	任意	×
target	HTML4.01 の仕様に従います。	任意	○
title	HTML4.01 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:ce

機能

ce 要素の dest 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:ce task="gosub" dest="url" .../>
```

対応する言語

HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
dest	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
next	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
cancel	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
accept_charset	HDML3.3 の仕様に従います。	任意	×
clear	HDML3.3 の仕様に従います。	任意	×
friend	HDML3.3 の仕様に従います。	任意	×
label	HDML3.3 の仕様に従います。	任意	×
method	HDML3.3 の仕様に従います。	任意	×
number	HDML3.3 の仕様に従います。	任意	×
postdata	HDML3.3 の仕様に従います。	任意	×
receive	HDML3.3 の仕様に従います。	任意	×
rel	HDML3.3 の仕様に従います。	任意	×
retvals	HDML3.3 の仕様に従います。	任意	×
sendreferer	HDML3.3 の仕様に従います。	任意	×
task	HDML3.3 の仕様に従います。	任意	×
value	HDML3.3 の仕様に従います。	任意	×
vars	HDML3.3 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

uu:choice

機能

choice 要素の bookmark 属性を URL 変換します。終了タグの直前にログイン／ログアウトアンカーを挿入します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:choice name="choicetag" bookmark="url" ...>
...
</uu:choice>
```

対応する言語

HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
bookmark	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	×
default	HDML3.3 の仕様に従います。	任意	×
idefault	HDML3.3 の仕様に従います。	任意	×
ikey	HDML3.3 の仕様に従います。	任意	×
key	HDML3.3 の仕様に従います。	任意	×
markable	HDML3.3 の仕様に従います。	任意	×
method	HDML3.3 の仕様に従います。	任意	×
name	HDML3.3 の仕様に従います。	任意	×
public	HDML3.3 の仕様に従います。	任意	×
title	HDML3.3 の仕様に従います。	任意	×

（凡例）

×：JSP スクリプトレット式で評価されません。

uu:display

機能

display 要素の bookmark 属性を URL 変換します。終了タグの直前にログイン／ログアウトアンカーを挿入します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:display name="displaytag" bookmark="url" ...>
...
</uu:display>
```

対応する言語

HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
bookmark	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	×
markable	HDML3.3 の仕様に従います。	任意	×
name	HDML3.3 の仕様に従います。	任意	×
public	HDML3.3 の仕様に従います。	任意	×
title	HDML3.3 の仕様に従います。	任意	×

（凡例）

×：JSP スクリプトレット式で評価されません。

uu:entry

機能

entry 要素の bookmark 属性を URL 変換します。開始タグの直後にログイン／ログアウトアンカーを挿入します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:entry name="entrytag" src="url" ...>
...
</uu:entry>
```

対応する言語

HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
bookmark	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	×
default	HDML3.3 の仕様に従います。	任意	×
emptyok	HDML3.3 の仕様に従います。	任意	×
fill	HDML3.3 の仕様に従います。	任意	×
format	HDML3.3 の仕様に従います。	任意	×

属性	説明	必須 / 任意	JSP スクリプトレット式評価
key	HDML3.3 の仕様に従います。	任意	×
markable	HDML3.3 の仕様に従います。	任意	×
name	HDML3.3 の仕様に従います。	任意	×
noecho	HDML3.3 の仕様に従います。	任意	×
public	HDML3.3 の仕様に従います。	任意	×
title	HDML3.3 の仕様に従います。	任意	×

（凡例）

×：JSP スクリプトレット式で評価されません。

uu:form

機能

form 要素の action 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:form action="url" ...>
...
</uu:form>
```

対応する言語

HTML，CHTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
action	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	必須	○
accept	HTML4.01 の仕様に従います。	任意	○
accept_charset ※1	HTML4.01 の仕様に従います。	任意	○
dir	HTML4.01 の仕様に従います。	任意	×
enctype	HTML4.01 の仕様に従います。	任意	○
hclass ※2	HTML4.01 の仕様に従います。	任意	×

属性	説明	必須 / 任意	JSP スクリプトレット式評価
hid ※3	HTML4.01 の仕様に従います。	任意	×
lang	HTML4.01 の仕様に従います。	任意	○
method	HTML4.01 の仕様に従います。	任意	×
name	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
onreset	HTML4.01 の仕様に従います。	任意	○
onsubmit	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
target	HTML4.01 の仕様に従います。	任意	○
title	HTML4.01 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　accept_charset：HTML 属性の accept-charset です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 3　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:frame

機能

frame 要素の src および longdesc 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:frame src="url" longdesc="url" ... />
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
src	uCosminexus Portal Framework を経由してコンテンツを出力するように変換します。	任意	○
longdesc	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
frameborder	HTML4.01 の仕様に従います。	任意	○
hclass※1	HTML4.01 の仕様に従います。	任意	×
hid※2	HTML4.01 の仕様に従います。	任意	×
marginheight	HTML4.01 の仕様に従います。	任意	○
marginwidth	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
noresize	HTML4.01 の仕様に従います。	任意	○
scrolling	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
target	HTML4.01 の仕様に従います。	任意	○
title	HTML4.01 の仕様に従います。	任意	×

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:iframe

機能

iframe 要素の src 属性および longdesc 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:iframe src="url" longdesc="url" ...>
...
</uu:iframe>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
src	uCosminexus Portal Framework を経由してインラインフレーム内に表示するコンテンツを出力するように変換します。	任意	○
longdesc	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
align	HTML4.01 の仕様に従います。	任意	○
frameborder	HTML4.01 の仕様に従います。	任意	○
hclass ※ 1	HTML4.01 の仕様に従います。	任意	×
height	HTML4.01 の仕様に従います。	任意	○
hid ※ 2	HTML4.01 の仕様に従います。	任意	×
marginheight	HTML4.01 の仕様に従います。	任意	○
marginwidth	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
scrolling	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
title	HTML4.01 の仕様に従います。	任意	×
width	HTML4.01 の仕様に従います。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:img

機能

img 要素の src 属性および longdesc 属性を URL 変換します。なお，ismap 属性は使用できません。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:img src="url" alt="sample" longdesc="url" ... />
```

対応する言語

HTML，CHTML，HDML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
src	インラインオブジェクトを直接参照するように変換します。	必須	○
longdesc	指定された相対パス形式の URL をポータル画面内に遷移するように変換します。	任意	○
alt	HTML4.01 の仕様に従います。	必須	○
align	HTML4.01 の仕様に従います。	任意	○
border	HTML4.01 の仕様に従います。	任意	○
hclass ※1	HTML4.01 の仕様に従います。	任意	×
dir	HTML4.01 の仕様に従います。	任意	×
height	HTML4.01 の仕様に従います。	任意	○
hspace	HTML4.01 の仕様に従います。	任意	○
hid ※2	HTML4.01 の仕様に従います。	任意	×
lang	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
title	HTML4.01 の仕様に従います。	任意	×
usemap	HTML4.01 の仕様に従います。	任意	○
vspace	HTML4.01 の仕様に従います。	任意	○
width	HTML4.01 の仕様に従います。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

×：JSP スクリプトレット式で評価されません。

注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:input

機能

input 要素の src 属性を URL 変換します。なお，次の属性は使用できません。

- checked
- disabled
- ismap
- readonly

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:input src="url" ... />
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
src	インラインオブジェクトを直接参照するように変換します。	任意	○
accept	HTML4.01 の仕様に従います。	任意	○
accesskey	HTML4.01 の仕様に従います。	任意	×
align	HTML4.01 の仕様に従います。	任意	○
alt	HTML4.01 の仕様に従います。	任意	○
dir	HTML4.01 の仕様に従います。	任意	×
hclass ※1	HTML4.01 の仕様に従います。	任意	×
hid ※2	HTML4.01 の仕様に従います。	任意	×
lang	HTML4.01 の仕様に従います。	任意	○
maxlength	HTML4.01 の仕様に従います。	任意	×
name	HTML4.01 の仕様に従います。	任意	○
onblur	HTML4.01 の仕様に従います。	任意	○
onchange	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onfocus	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
onselect	HTML4.01 の仕様に従います。	任意	○
size	HTML4.01 の仕様に従います。	任意	○
style	HTML4.01 の仕様に従います。	任意	×
tabindex	HTML4.01 の仕様に従います。	任意	×
title	HTML4.01 の仕様に従います。	任意	×
type	HTML4.01 の仕様に従います。	任意	×

属性	説明	必須 / 任意	JSP スクリプトレット式評価
usemap	HTML4.01 の仕様に従います。	任意	○
value	HTML4.01 の仕様に従います。	任意	○

（凡例）
○：JSP スクリプトレット式で評価されます。
×：JSP スクリプトレット式で評価されません。
注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。
注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:object

機能

object 要素の codebase 属性，classid 属性，data 属性，および archive 属性を URL 変換します。なお，declare 属性は使用できません。

codebase 属性が指定された場合

codebase 属性だけを URL 変換します。

codebase 属性が指定されていない場合

classid 属性，data 属性，archive 属性を URL 変換します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:object codebase="url" classid="url" data="url" archive="url
list">
...
</uu:object>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
codebase	指定された相対パス形式の URL を直接参照するように変換します。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
archive	codebase 属性が指定されなかった場合，指定された相対パス形式の URL を直接参照するように変換します。	任意	○
classid	codebase 属性が指定されなかった場合，指定された相対パス形式の URL を直接参照するように変換します。	任意	○
data	codebase 属性が指定されなかった場合，指定された相対パス形式の URL を直接参照するように変換します。	任意	○
align	HTML4.01 の仕様に従います。	任意	○
border	HTML4.01 の仕様に従います。	任意	○
codetype	HTML4.01 の仕様に従います。	任意	×
dir	HTML4.01 の仕様に従います。	任意	×
hclass ※ 1	HTML4.01 の仕様に従います。	任意	×
height	HTML4.01 の仕様に従います。	任意	○
hid ※ 2	HTML4.01 の仕様に従います。	任意	×
hspace	HTML4.01 の仕様に従います。	任意	○
lang	HTML4.01 の仕様に従います。	任意	○
name	HTML4.01 の仕様に従います。	任意	○
onclick	HTML4.01 の仕様に従います。	任意	○
ondblclick	HTML4.01 の仕様に従います。	任意	○
onkeydown	HTML4.01 の仕様に従います。	任意	○
onkeypress	HTML4.01 の仕様に従います。	任意	○
onkeyup	HTML4.01 の仕様に従います。	任意	○
onmousedown	HTML4.01 の仕様に従います。	任意	○
onmousemove	HTML4.01 の仕様に従います。	任意	○
onmouseout	HTML4.01 の仕様に従います。	任意	○
onmouseover	HTML4.01 の仕様に従います。	任意	○
onmouseup	HTML4.01 の仕様に従います。	任意	○
standby	HTML4.01 の仕様に従います。	任意	×
style	HTML4.01 の仕様に従います。	任意	×
tabindex	HTML4.01 の仕様に従います。	任意	×
title	HTML4.01 の仕様に従います。	任意	×
type	HTML4.01 の仕様に従います。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
usemap	HTML4.01 の仕様に従います。	任意	○
vspace	HTML4.01 の仕様に従います。	任意	○
width	HTML4.01 の仕様に従います。	任意	○

（凡例）
○：JSP スクリプトレット式で評価されます。
×：JSP スクリプトレット式で評価されません。
注※ 1　hclass：HTML 属性の class です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。
注※ 2　hid：HTML 属性の id です。詳細は「3.4　タグライブラリ使用時の注意」を参照してください。

uu:script

機能

script 要素の src 属性を URL 変換します。なお，defer 属性は使用できません。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<uu:script type="text/javascript" src="url" ...>
...
</uu:script>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
src	指定された相対パス形式の URL を直接参照するように変換します。	任意	○
type	HTML4.01 の仕様に従います。	必須	○
charset	HTML4.01 の仕様に従います。	任意	×
language	HTML4.01 の仕様に従います。	任意	×

（凡例）
○：JSP スクリプトレット式で評価されます。
×：JSP スクリプトレット式で評価されません。

14.3 ポートレットモードタグライブラリ

ポートレットモードタグライブラリの一覧を次の表に示します。

表 14-2 ポートレットモードタグライブラリの一覧

タグ名	説明
PortalAC:ModeDeniedList	画面に表示されていないポートレットで，ポートレットモードの変更が拒否されたポートレットのリストを表示します。
PortalAC:isModeDenied	画面に表示されていないポートレットで，ポートレットモードの変更が拒否されたポートレットが一つ以上ある場合だけボディを表示します。

PortalAC:ModeDeniedList

機能

画面に表示されていないポートレットで，ポートレットモードの変更が拒否されたポートレットのリストを表示します。

注意事項

このタグライブラリはナビゲーションテンプレートの下部設定 JSP ファイル（bottom.jsp）だけで使用できます。

構文

```
<%@ taglib uri='http://soft.hitachi.co.jp/portal/portalac'
prefix='PortalAC' %><%@ page contentType="text/html;
charset=Shift_JIS" %>
...
<PortalAC:ModeDeniedList separator="," />
```

対応する言語

HTML，CHTML，HDML

属性

属性	説明	必須 / 任意
separator	ポートレット名を区切る文字列。省略時は半角コンマ記号が区切り文字になる。	任意

PortalAC:isModeDenied

機能

画面に表示されていないポートレットで，ポートレットモードの変更が拒否されたポートレットが一つ以上ある場合だけボディを表示します。

注意事項

このタグライブラリは，ナビゲーションテンプレートの下部設定 JSP ファイル（bottom.jsp）だけで使用できます。

構文

```
<%@ taglib uri='http://soft.hitachi.co.jp/portal/portalac'
prefix='PortalAC' %><%@ page contentType="text/html;
charset=Shift_JIS" %>
...
<PortalAC:isModeDenied>画面に表示されていないポートレットモード変更不可ポートレットがあるときだけ表示する内容</PortalAC:isModeDenied>
```

対応する言語

HTML，CHTML，HDML

14.4　ポートレットユティリティクラスライブラリ

ポートレットユティリティクラスライブラリの一覧を次の表に示します。

表 14-3　ポートレットユティリティクラスライブラリの一覧

クラス名	説明
jp.co.hitachi.soft.portal.portlet.PortletException	uCosminexus Portal Framework で使用するタグライブラリおよびクラスライブラリで発生した例外を格納します。
jp.co.hitachi.soft.portal.portlet.PortletURI	指定された相対パスを基に，ポートレット URL 変換，ポータル URL 変換，ドキュメントベース URL 変換をします。
jp.co.hitachi.soft.portal.portlet.PortletUtils	日立 API ポートレット用ユティリティクラスです。

jp.co.hitachi.soft.portal.portlet.PortletException

機能

uCosminexus Portal Framework で使用するタグライブラリおよびクラスライブラリで発生した例外を格納します。日立 API ポートレットの開発者は，通常はこのクラスのインスタンスを作成する必要はありません。

コンストラクタの説明

PortletException〔詳細メッセージなし〕

形式

```
public PortletException()
```

機能

詳細メッセージを指定しないで PortletException を生成します。

PortletException〔詳細メッセージ指定〕

形式

```
public PortletException(String msg)
```

機能

指定された詳細メッセージを持つ PortletException を生成します。

パラメタ

msg - 詳細メッセージ

jp.co.hitachi.soft.portal.portlet.PortletURI

機能

ポートレット URL 変換，ポータル URL 変換，およびドキュメントベース URL 変換をします。

URL 変換種別，変換対象および変換内容を次の表に示します。

表 14-4　URL の変換種別，変換対象および変換内容

URL 変換種別	変換対象	変換内容
ポートレット URL 変換	• 相対パス形式 例：index.jsp	統合アダプタを経由してポータル画面内に遷移するように，URL 変換をします。必要に応じて HttpServletResponse#encodeURL を呼び出して，URLRewriting をします。
ドキュメントベース URL 変換	• 相対パス形式 例：index.jsp	指定された URL を Web サーバから直接参照するように URL 変換をします。なお，URLRewriting はしません。
ポータル URL 変換	• 相対パス形式 例：index.jsp	ポータルを経由してコンテンツを取得するように URL 変換をします。統合アダプタは経由しません。
無変換	• 絶対 URL 形式 例：http://server/portal/sample/index.jsp • ホスト内絶対パス形式 例：/portlets/sample/index.jsp	URL 変換をしません。

注意事項

本クラスを使用して取得した URL をポートレットユーティリティタグライブラリの URL 指定属性に指定すると，一度変換された URL に対して再度 URL 変換が行われるため，変換後の URL が不正となる場合があります。

コンストラクタの説明

PortletURI

形式

```
public PortletURI(ServletRequest req,
                  ServletResponse res)
```

機能

PortletURI オブジェクトを生成します。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

res - 日立 API ポートレットに渡された ServletResponse オブジェクト

メソッドの一覧

戻り値のデータ型	メソッド	説明
static String	getRestoreURI(ServletRequest req, ServletResponse res)	ポートレット編集状態から復元する URL を取得する static メソッドです。
String	getRestoreURI()	ポートレット編集状態から復元する URL を取得します。
static String	lookupURI(ServletRequest req, ServletResponse res, String portletname, PortletInfoBean.Mode mode, String parameter)	ポートレットを呼び出す URL を取得する static メソッドです。
String	lookupURI(String portletname, PortletInfoBean.Mode mode, String parameter)	ポートレットを呼び出す URL を取得します。
static String	transInlineURI(ServletRequest req,ServletResponse res,String org_uri)	ドキュメントベース URL 変換をする static メソッドです。
String	transInlineURI(String org_uri)	ドキュメントベース URL 変換をします。
static String	transInlineURIEx(ServletRequest req,ServletResponse res,String org_uri)	ドキュメントベース URL 変換をする static メソッドです。
String	transInlineURIEx(String org_uri)	ドキュメントベース URL 変換をします。
static String	transPortalURI(ServletRequest req,ServletResponse res, String org_uri)	ポータル URL 変換をする static メソッドです。
String	transPortalURI(String org_uri)	ポータル URL 変換をします。
static String	transPortletURI(ServletRequest req, ServletResponse res, String org_uri)	ポートレット URL 変換をする static メソッドです。
String	transPortletURI(String org_uri)	ポートレット URL 変換をします。
static String	transContextURI(ServletRequest req, ServletResponse res, String org_uri)	ポータルサーバのコンテキストルートからのコンテンツを取得する URL に変換をする static メソッドです。
String	transContextURI(String org_uri)	ポータルサーバのコンテキストルートからのコンテンツを取得する URL に変換をします。

メソッドの説明

getRestoreURI〔static メソッド〕

形式

```
public static String getRestoreURI(ServletRequest req,
                                   ServletResponse res)
```

機能

ポートレット編集状態から復元する URL を取得します。

- 通常画面から呼び出された編集画面で呼び出したとき
  通常画面に遷移する URL を取得します。
- 最大化画面から呼び出された編集画面で呼び出したとき
  最大化画面に遷移する URL を取得します。
- 通常画面（最小化表示時）から呼び出された編集画面で呼び出したとき
  最大化画面に遷移する URL を取得します。
- 編集画面以外から呼び出したとき
  通常画面に遷移する URL を取得します。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト
res - 日立 API ポートレットに渡された ServletResponse オブジェクト

戻り値

生成された URL（最大 2,048 文字）

例外

PortletException - 次の場合に発生します。

- 引数 req，res の値が不正な場合

注意事項

- このメソッドを使用した場合，表示する JSP ファイルを指定できません。そのため，マッピング定義ファイルで指定した JSP ファイル（メイン画面）に編集内容を反映する処理を入れる必要があります。
- このメソッドは，IFRAME モード，および NEWWINDOW モードには対応していません。そのため，これらの画面モードではこのメソッドを呼び出さないようにしてください。

getRestoreURI

形式

```
public String getRestoreURI()
```

機能

ポートレット編集状態から復元する URL を取得します。

- 通常画面から呼び出された編集画面で呼び出したとき
  通常画面に遷移する URL を取得します。
- 最大化画面から呼び出された編集画面で呼び出したとき

最大化画面に遷移する URL を取得します。
- 通常画面（最小化表示時）から呼び出された編集画面で呼び出したとき
  最大化画面に遷移する URL を取得します。
- 編集画面以外から呼び出したとき
  通常画面に遷移する URL を取得します。

パラメタ
なし

戻り値
生成された URL（最大 2,048 文字）

例外
PortletException - 次の場合に発生します。
- コンストラクタで渡された引数 req，res の値が不正な場合

注意事項
- このメソッドは，PortletURI クラスのコンストラクタで初期化したあとに使用してください。
- このメソッドを使用した場合，表示する JSP ファイルを指定できません。そのため，マッピング定義ファイルで指定した JSP ファイル（メイン画面）に編集内容を反映する処理を入れる必要があります。
- このメソッドは，IFRAME モード，および NEWWINDOW モードには対応していません。そのため，これらの画面モードではこのメソッドを呼び出さないようにしてください。

lookupURI〔static メソッド〕

形式
```
public static String lookupURI(ServletRequest req,
                                ServletResponse res,
                                String portletname,
                                PortletInfoBean.Mode mode,
                                String parameter)
```

機能
ポートレットを呼び出す URL を取得します。PortletURI クラスのインスタンス生成後は，次の形式でも利用できます。
```
public String lookupURI(String portletname,
                        PortletInfoBean.Mode mode,
                               String parameter)
```

パラメタ
req - 日立 API ポートレットに渡された ServletRequest オブジェクト
res - 日立 API ポートレットに渡された ServletResponse オブジェクト
portletname - 呼び出すポートレットの名称。指定しない場合は null を設定します。その場合には自ポートレットとみなします。空文字列を指定した場合は，空のポートレットとして URL を生成するため，クリック時にはエラーとなります。

mode - 呼び出す画面モード。指定できる画面モードを次に示します。

- MAXIMIZE
- IFRAME
- NEWWINDOW

これ以外を指定すると例外が発生します。指定しない場合は null を設定します。その場合には呼び出し元の画面モードとなります。ただし，呼び出し元の画面モードが DEFAULT の場合は MAXIMIZE になります。呼び出し元とは，このメソッドの実行時のことです。

parameter - 呼び出すポートレットに渡すパラメタ。クエリ文字列で指定します（URL エンコード済みの文字列を指定）。指定できる最大バイト数は 1,024 バイトです。"?" の付加は不要です。指定しない場合は null を設定します。

戻り値

生成された URL（最大 2,046 文字）

例外

PortletException - 次の場合に発生します。

- 引数 mode の指定に誤りがある場合
- 引数 req，res の値が不正な場合

注意事項

デプロイされていない portletname を指定した場合，および無効な parameter を指定しても URL は生成されますが，ユーザが URL をクリックした際にエラー画面を表示します。

lookupURI

形式

```
public String lookupURI(String portletname,
                        PortletInfoBean.Mode mode,
                        String parameter)
```

機能

ポートレットを呼び出す URL を取得します。PortletURI クラスのコンストラクタで初期化したあとにこのメソッドを利用してください。処理については static 版と同様です。

パラメタ

portletname - 呼び出すポートレットの名称。指定しない場合は null を設定します。その場合には自ポートレットとみなします。空文字列を指定した場合は，空ポートレットとして URL を生成するため，クリック時にはエラーとなります。

mode - 呼び出す画面モード。指定できる画面モードを次に示します。

- MAXIMIZE
- IFRAME
- NEWWINDOW

これ以外を指定すると例外が発生します。指定しない場合は null を設定します。その場合には呼び出し元の画面モードとなります。ただし，呼び出し元の画面モードが DEFAULT の場合は MAXIMIZE になります。呼び出し元とは，このメソッドの実行時のことです。
parameter - 呼び出すポートレットに渡すパラメタ。クエリ文字列で指定します（URL エンコード済みの文字列を指定）。指定できる最大バイト数は 1,024 バイトです。"?" の付加は不要です。指定しない場合は null を設定します。

戻り値

生成された URL（最大 2,046 文字）

例外

PortletException - 次の場合に発生します。
- 引数 mode の指定に誤りがある場合

transInlineURI〔static メソッド〕

形式

```
public static String transInlineURI(ServletRequest req,
                                    ServletResponse res,
                                    String org_uri)
```

機能

このメソッドは，画像ファイルを指定するとき，およびポータル外に画面遷移するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。
「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってドキュメントベース URL 変換をします。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト
res - 日立 API ポートレットに渡された ServletResponse オブジェクト
org_uri - 変換対象 URL

戻り値

変換後 URL。このメソッドは，プロパティ「jp.co.hitachi.soft.portal.transurlflag」の設定に関わらず，プロトコルとサーバ名称を含まない URL が返却されます。プロトコルとサーバ名称を含む URL を取得する場合は，transInlineURIEx メソッドを使用してください。URL 変換規則の詳細については，マニュアル「uCosminexus Portal Framework システム管理者ガイド」を参照してください。

例外

PortletException - req または res が不正な場合
java.net.MalformedURLException - 不正な URL を指定した場合

transInlineURI〔static ではないメソッド〕

形式

```
public String transInlineURI(String org_uri)
```

機能

このメソッドは，画像ファイルを指定するとき，およびポータル外に画面遷移するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。

パラメタ

org_uri - 変換対象 URL

戻り値

変換後 URL。このメソッドは，プロパティ「jp.co.hitachi.soft.portal.transurlflag」の設定に関わらず，プロトコルとサーバ名称を含まない URL が返却されます。プロトコルとサーバ名称を含む URL を取得する場合は，transInlineURIEx メソッドを使用してください。URL 変換規則の詳細については，マニュアル「uCosminexus Portal Framework システム管理者ガイド」を参照してください。

例外

PortletException - req または res が不正な場合

java.net.MalformedURLException - 不正な URL を指定した場合

transInlineURIEx〔static メソッド〕

形式

```
public static String transInlineURIEx(ServletRequest req,
                                       ServletResponse res,
                                       String org_uri)
```

機能

このメソッドは，画像ファイルを指定するとき，およびポータル外に画面遷移するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってドキュメントベース URL 変換をします。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

res - 日立 API ポートレットに渡された ServletResponse オブジェクト

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

transInlineURIEx〔static ではないメソッド〕

形式

```
public String transInlineURIEx(String org_uri)
```

機能

このメソッドは，画像ファイルを指定するとき，およびポータル外に画面遷移するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってドキュメントベース URL 変換をします。

パラメタ

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

transPortalURI〔static メソッド〕

形式

```
public static String transPortalURI(ServletRequest req,
                                    ServletResponse res,
                                    String org_uri)
```

機能

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってポータル URL 変換をします。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

res - 日立 API ポートレットに渡された ServletResponse オブジェクト

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

transPortalURI〔static ではないメソッド〕

形式

```
public String transPortalURI(String org_uri)
```

機能

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってポータル URL 変換をします。

パラメタ

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

transPortletURI〔static メソッド〕

形式

```
public static String transPortletURI(ServletRequest req,
                                     ServletResponse res,
                                     String org_uri)
```

機能

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってポートレット URL 変換をします。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

res - 日立 API ポートレットに渡された ServletResponse オブジェクト

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

java.net.MalformedURLException - 不正な URL を指定した場合

transPortletURI〔static ではないメソッド〕

形式

```
public String transPortletURI(String org_uri)
```

機能

「3.8.4　ポートレットユティリティクラスライブラリ使用時の画面遷移」に従ってポートレット URL 変換をします。

パラメタ

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または res が不正な場合

java.net.MalformedURLException - 不正な URL を指定した場合

transContextURI〔static メソッド〕

形式

```
public static String transContextURI(ServletRequest req,
```

```
                                        ServletResponse res,
                                        String org_uri)
```

機能

このメソッドは，コンテキストルートからのコンテンツの URL を指定するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト
res - 日立 API ポートレットに渡された ServletResponse オブジェクト
org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - req または org_uri が不正な場合

transContextURI〔static ではないメソッド〕

形式

```
public String transContextURI(String org_uri)
```

機能

このメソッドは，コンテキストルートからのコンテンツの URL を指定するときに使用する URL を生成します。このメソッドで生成された URL は，ポータルを経由しません。

パラメタ

org_uri - 変換対象 URL

戻り値

変換後 URL

例外

PortletException - org_uri が不正な場合

jp.co.hitachi.soft.portal.portlet.PortletUtils

機能

日立 API ポートレット用ユティリティクラスです。

コンストラクタの説明

PortletUtils

形式

```
public PortletUtils()
```

機能

空の PortletUtils オブジェクトを作成します。通常は作成の必要はありません。

メソッドの一覧

戻り値のデータ型	メソッド	説明
static String	getNamespace(ServletRequest req, ServletResponse res)	ポータルでユニークなポートレットネームスペースを取得します。

メソッドの説明

getNamespace

形式

```
public static String getNamespace(ServletRequest req,
                                  ServletResponse res)
```

機能

ポータルでユニークなポートレットネームスペースを取得します。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

res - 日立 API ポートレットに渡された ServletResponse オブジェクト

戻り値

ポータル内でユニークなネームスペース

例外

PortletException - req または res が不正な場合

14.5 ポートレット情報取得 Bean

ポートレット情報取得 Bean は，ポートレット固有の情報や画面モード情報を取得するための API です。

ポートレット情報取得 Bean の一覧を次の表に示します。

表 14-5 ポートレット情報取得 Bean の一覧

クラス名	説明
jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean	ポートレット固有の情報を取得します。
jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean.Mode	画面モード情報を示すクラスです。

ポートレット情報取得 Bean の詳細を説明します。

jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean

機能

ポートレット固有の情報を取得します。

Bean の項目

宣言

```
<jsp:useBean id="configbean"
class="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
scope="page"/>
```

対象スコープ

```
page
```

メソッドの一覧

戻り値のデータ型	メソッド	説明
String	getDescription()	ポートレットの説明文を取得します。
PortletInfoBean.Mode	getMode()	現在の画面モードを取得します。
String	getName()	ポートレット名称を取得します。
String	getParameter(String name)	パラメタ設定値を取得します。
String	getTitle()	ポートレットのタイトル文字列を取得します。

戻り値のデータ型	メソッド	説明
void	initBean(ServletRequest req)	Bean を初期化します。
void	initBean(ServletRequest req, String portletName)	指定したポートレット名称で PortletInfoBean を初期化します。
void	setCustomizeTitle(String titleString)	新規ウィンドウのタイトルを，一時的に変更します。

メソッドの説明

getDescription

形式

```
public String getDescription()
```

機能

ポートレットの説明文を取得します。

パラメタ

なし

戻り値

ポートレットの説明文

説明文が設定されていない場合，空文字列を戻します。

例外

なし

getMode

形式

```
public PortletInfoBean.Mode getMode()
```

機能

現在の画面モードを取得します。

パラメタ

なし

戻り値

現在の画面モード

次のどれかの値を戻します。

戻り値	モード名称	説明
PortletInfoBean.Mode.DEFAULT	標準モード	複数ポートレット表示状態
PortletInfoBean.Mode.MAXIMIZE	最大化モード	最大化表示状態
PortletInfoBean.Mode.EDIT	編集モード	編集画面表示状態
PortletInfoBean.Mode.NEWWINDOW	新規ウィンドウモード	新規ウィンドウ表示状態

戻り値	モード名称	説明
PortletInfoBean.Mode.IFRAME	インラインモード	インラインフレーム表示状態
PortletInfoBean.Mode.IRREGULAR	規格外モード	画面モードが取得できない状態

例外
なし

getName

形式
```
public String getName()
```

機能
ポートレット名称を取得します。

パラメタ
なし

戻り値
ポートレット名称
initBean() が呼ばれていない，または initBean() に失敗している場合，null を戻します。

例外
なし

getParameter

形式
```
public String getParameter(String name)
```

機能
name で指定されたパラメタ設定値を取得します。

パラメタ
name - パラメタ名称

戻り値
パラメタ設定値

例外
IllegalArgumentException - 入力パラメタが不正な場合

getTitle

形式
public String getTitle()

機能
ポートレットのタイトル文字列を取得します。

パラメタ
なし

戻り値
ポートレットのタイトル文字列
ポートレットのタイトルが設定されていない場合，"No title set" を戻します。

例外
なし

initBean〔PortletInfoBean を初期化〕

形式
```
public void initBean(ServletRequest req)
```

機能
PortletInfoBean を初期化します。

パラメタ
req - JSP に渡された ServletRequest オブジェクト

戻り値
なし

例外
IllegalArgumentException - 引数 req が null または不正なオブジェクトである場合

initBean〔指定したポートレット名称で PortletInfoBean を初期化〕

形式
```
public void initBean(ServletRequest req,
                     String portletName)
```

機能
指定したポートレット名称で PortletInfoBean を初期化します。以降，指定したポートレットの情報を取得します。

パラメタ
req - JSP に渡された ServletRequest オブジェクト
portletName - ポートレット名称。指定できる最大文字数は 80 文字です。

戻り値
なし

例外
IllegalArgumentException - 次の場合に発生します。
- 引数 req が null または不正なオブジェクトである場合
- 引数 portletName が null の場合

IllegalStateException - 引数 portletName に指定したポートレットがデプロイされていない場合

setCustomizeTitle

形式

```
public void setCustomoizeTitle(String titleString)
```

機能

新規ウィンドウのタイトルを，一時的に変更します。

パラメタ

titleString - 新規ウィンドウのタイトルに表示する文字列。指定できる最大文字数は 64 文字です。

戻り値

なし

例外

IllegalAugumentException - 次の場合に発生します。

- 引数 titleString が null の場合
- 引数 titleString の文字数が 64 文字を超えた場合

InitializationException - 初期化されていない場合

jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean.Mode

機能

画面モード情報を示すクラスです。

フィールドの説明

フィールド	モード名称
PortletInfoBean.Mode DEFAULT	標準モード
PortletInfoBean.Mode MAXIMIZE	最大化モード
PortletInfoBean.Mode EDIT	編集モード
PortletInfoBean.Mode NEWWINDOW	新規ウィンドウモード
PortletInfoBean.Mode IFRAME	インラインモード
PortletInfoBean.Mode IRREGULAR	規格外モード

メソッドの一覧

戻り値のデータ型	メソッド	説明
String	toString()	画面モードの文字列表現を戻します。

メソッドの説明

toString

形式

```
public String toString()
```

機能

画面モードの文字列表現を戻します。

パラメタ

なし

戻り値

画面モードの文字列表現
次のどれかの値を戻します。

戻り値	モード名称
DEFAULT	標準モード
MAXIMIZE	最大化モード
EDIT	編集モード
NEWWINDOW	新規ウィンドウモード
IFRAME	インラインモード
IRREGULAR	規格外モード

例外

なし

14.6 ユーザ情報取得 Bean

ユーザ情報取得 Bean は，リポジトリに格納されているユーザ情報を取得する API です。

ユーザ情報取得 Bean の一覧を次の表に示します。

表 14-6 ユーザ情報取得 Bean の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean	リポジトリに格納されているユーザ情報を取得します。

ユーザ情報取得 Bean の詳細を説明します。

jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean

機能

リポジトリに格納されているユーザ情報を取得します。

Bean の項目

宣言

```
<jsp:useBean id="user"
class="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean"
scope="session" />
```

対象スコープ

```
session, request
```

`request` は性能が低下するため `session` を推奨します。

メソッドの一覧

戻り値のデータ型	メソッド	説明
boolean	checkGroup(String groupName)	ユーザがグループに所属しているかをチェックします。
boolean	checkLoggedIn()	ユーザがログイン済みかをチェックします。
boolean	checkOu(String ouName)	ユーザが組織に所属しているかをチェックします。
boolean	commitData()	LDAP または DB にデータを書き込みます。
void	customizeInfoEnd()	排他区間の終了を設定します。
void	customizeInfoStart()	排他区間の開始を設定します。区間中では，LDAP/DB へのデータの保存処理は保留されます。

戻り値のデータ型	メソッド	説明
Object	getCustomizeInfo(String appKey)	ポートレットのカスタマイズ情報を取得します。
String[]	getGroupName()	ユーザのグループ名をすべて取得します。
String[]	getOuName()	ユーザの組織単位名をすべて取得します。
Object	getParameter(String column)	ユーザ独自に定義している属性値を戻します。
Object	getUserCustomizeInfo(String appKey)	ユーザのカスタマイズ情報を取得します。
String	getUserId()	ユーザのユーザ ID を取得します。
boolean	isAccessible(String portletName)	ポートレットのアクセス権を判定します。
void	removeCustomizeInfo(String appKey)	ポートレットのカスタマイズ情報を削除します。
void	removeUserCustomizeInfo(String appKey)	ユーザのカスタマイズ情報を削除します。
void	setCustomizeInfo(String appKey, Object appData)	ポートレットのカスタマイズ情報を保存します。
void	setRequest(HttpServletRequest req)	リクエストオブジェクトを設定します。必ず最初にこのメソッドを呼び出してください。
void	setUserCustomizeInfo(String appKey, Object appData)	ユーザのカスタマイズ情報を保存します。

メソッドの説明

checkGroup

形式

```
public boolean checkGroup(String groupName)
```

機能

ユーザがグループに所属しているかをチェックします。<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。

パラメタ

groupName - チェックするグループ名

戻り値

グループに所属しているかどうか

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

checkLoggedIn

形式

```
public boolean checkLoggedIn()
```

機能

ユーザがログイン済みかをチェックします。<jsp:getProperty> タグ, <jsp:setProperty> タグでの操作はできません。

パラメタ

なし

戻り値

ログインしているかどうか

例外

なし

checkOu

形式

```
public boolean checkOu(String ouName)
```

機能

ユーザが組織に所属しているかをチェックします。<jsp:getProperty> タグ, <jsp:setProperty> タグでの操作はできません。

パラメタ

ouName - チェックする組織単位名

戻り値

組織単位に所属しているかどうか

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

commitData

形式

```
public boolean commitData()
```

機能

LDAP または DB にデータを書き込みます。

パラメタ

なし

戻り値

true - データ書き込み成功

false - データ書き込み失敗

例外

なし

customizeInfoEnd

形式

```
public void customizeInfoEnd()
```

機能
排他区間の終了を設定します。

パラメタ
なし

戻り値
なし

例外
なし

customizeInfoStart

形式
```
public void customizeInfoStart()
```

機能
排他区間の開始を設定します。区間中では，LDAP/DB へのデータの保存処理は保留されます。

パラメタ
なし

戻り値
なし

例外
なし

getCustomizeInfo

形式
```
public Object getCustomizeInfo(String appKey)
```

機能
ポートレットのカスタマイズ情報を取得します。アプリケーションキーに対応するポートレットのカスタマイズ情報を取得します。指定されたカスタマイズ情報が見つからない場合は null を戻します。<jsp:getProperty> タグ，<jsp:setProperty> タグでの操作はできません。

パラメタ
appKey - アプリケーションキー

戻り値
カスタマイズ情報

例外
IllegalArgumentException - 入力パラメタが null または不正な場合

注意事項
このメソッドを複数回呼び出すとき（複数のカスタマイズ情報を取得するとき）にエラーが発生した場合は，エラーの発生したカスタマイズ情報だけが取得で

きません。それ以外のカスタマイズ情報については，正常に取得できます。

getGroupName

形式

```
public String[] getGroupName()
```

機能

ユーザのグループ名をすべて取得します。ユーザのグループ名が見つからないときは null を戻します。

パラメタ

なし

戻り値

ユーザのグループ名

例外

なし

getOuName

形式

```
public String[] getOuName()
```

機能

ユーザの組織単位名をすべて取得します。ユーザの組織単位名が見つからないときは null を戻します。

パラメタ

なし

戻り値

ユーザの組織単位名

例外

なし

getParameter

形式

```
public Object getParameter(String column)
```

機能

ユーザ独自に定義している属性値を戻します。multivalue の場合は Enumeration 型で戻します。singlevalue の場合は Object 型で戻します。指定した属性値が見つからない場合は null を戻します。<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。
パラメタには，リポジトリファイルのマッピング情報の共通項目名（dest）を指定します。getParameter を用いて属性値を取得するには，リポジトリファイルにディレクトリ情報の属性と共通項目名を設定する必要があります。設定方法については，マニュアル「uCosminexus Portal Framework　システム管理

者ガイド」の「リポジトリファイル（Repository.xml）」の説明を参照してください。

パラメタ

column - リポジトリファイルの共通項目名（dest）

戻り値

ディレクトリサーバ上の属性値

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getUserCustomizeInfo

形式

```
public Object getUserCustomizeInfo(String appKey)
```

機能

ユーザのカスタマイズ情報を取得します。アプリケーションキーに対応するユーザのカスタマイズ情報を取得します。指定されたカスタマイズ情報が見つからない場合は null を返却します。<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。
ログインしていない場合，null を戻します。

パラメタ

appKey - アプリケーションキー（null 以外）

戻り値

カスタマイズ情報（シリアライズできる値）

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

注意事項

このメソッドを複数回呼び出すとき（複数のカスタマイズ情報を取得するとき）にエラーが発生した場合は，エラーの発生したカスタマイズ情報だけが取得できません。それ以外のカスタマイズ情報については，正常に取得できます。

getUserId

形式

```
public String getUserId()
```

機能

ユーザのユーザ ID を取得します。この API で取得できるユーザ ID は，ユーザ情報のマッピング項目のユーザ ID で指定した属性の値です。
ユーザのユーザ ID が見つからないときは null を戻します。

パラメタ

なし

戻り値

ユーザのユーザ ID

例外

なし

isAccessible

形式

```
public boolean isAccessible(String portletName)
```

機能

PortalUserInfoBean に結び付けられたユーザ（対象ユーザ）が，引数 portletName に指定したポートレット（対象ポートレット）に対してアクセス権があるかどうかを判定します。
なお，対象ユーザがログインしていない状態でこのメソッドを呼び出した場合（ウェルカム画面に配置されたポートレットからこのメソッドを呼び出すなど），対象ユーザを匿名のユーザとみなしてアクセス権を判定します。このとき，ウェルカム画面に対象ポートレットが配置されているかどうか（デプロイされているかどうか）も判定します。
「アクセス権がある」とは，複数ポートレットが表示されている状態で対象ポートレットを表示できること，および対象ポートレットを最大化状態で表示できることを示します。
アクセス権は，対象ポートレットがデプロイされているかどうか，および対象ユーザが対象ポートレットに対してアクセス権があるかどうかの組み合わせにより次のように決定されます。

対象ポートレットのデプロイ状況	対象ユーザの対象ポートレットへのアクセス権	このメソッドの戻り値
未デプロイ	あり	false
	なし	
デプロイ済み	あり	true
	なし	false

利用手順

1. HTTP リクエストを受信してからこのメソッドを使用するまでの間に setRequest メソッドで PortalUserInfoBean を初期化します。
2. このメソッドを呼び出します。

パラメタ

portletName - アクセス権を判定するポートレットの名称（null 以外）。指定できる最大文字数は，標準 API ポートレットの場合は 64 文字，標準 API ポートレット以外の場合は 32 文字です。

戻り値

true - パラメタに指定したポートレットにアクセス権がある
false - パラメタに指定したポートレットにアクセス権がない

例外

IllegalArgumentException - 次の場合に発生します。

- 引数 portletName に null を指定した場合
- 引数 portletName に空文字列を指定した場合
- 引数 portletName に 33 文字以上の文字数を指定した場合（標準 API ポートレットの場合は 65 文字以上の文字数を指定した場合）

IllegalStateException - setRequest メソッドで初期化する前にこのメソッドが呼ばれたとき

注意事項

このメソッドを使用してアクセス権を判定する場合，ページスコープまたはリクエストスコープ（推奨）でキャッシュしてください。
isAccessible メソッドの結果は，ページまたはリクエストの開始時点で boolean 型の変数に格納され，アプリケーションは，その変数の値を用いてアクセス権を判断します。そのため，複数リクエストにわたってキャッシュしないでください。複数リクエストにわたってキャッシュすると，不正な情報を表示するおそれがあります。
この注意事項には，次に示す理由があります。

- 呼び出しオーバヘッドは無視できない
  boolean 型の変数参照と public メソッドの呼び出しでは，実行速度が数倍異なるため，オーバヘッドは無視できません。
- 計算量
  単体コストは，（コミュニティ数）×（コミュニティ単位の平均所属ポートレット数）に比例します。計算量は，O(n) です。したがって，コミュニティ数がおおむね 2,000 を超える環境でこのメソッドを頻繁に呼び出すと，性能に影響を及ぼします。
  このメソッドを呼び出す回数の目安を次に示します。
  推奨値：リクエスト内で１回／調査対象ポートレット　以内
  最大値：リクエスト内で５回／調査対象ポートレット　以内

removeCustomizeInfo

形式

```
public void removeCustomizeInfo(String appKey)
```

機能

アプリケーションキーに対応するポートレットのカスタマイズ情報を削除します。

パラメタ

appKey - アプリケーションキー

戻り値

なし

例外

IllegalArgumentException - 引数が不正または null の場合

removeUserCustomizeInfo

形式

```
public void removeUserCustomizeInfo(String appKey)
```

機能

アプリケーションキーに対応するユーザのカスタマイズ情報を削除します。<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。
ログインしていない場合，何もしません。

パラメタ

appKey - アプリケーションキー（null 以外）

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

setCustomizeInfo

形式

```
public void setCustomizeInfo(String appKey,
                             Object appData)
```

機能

アプリケーションキーに対応するポートレットのカスタマイズ情報をセットします。<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。ログインしていないときは何もしません。

パラメタ

appKey - アプリケーションキー
appData - カスタマイズ情報（シリアライズできる値）

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

setRequest

形式

```
public void setRequest(HttpServletRequest req)
```

機能

リクエストオブジェクトを設定します。PortalUserInfoBean を使用する場合，必ず最初にこのメソッドを呼び出してください。リクエストの 2 回目以降は呼び出す必要はありません。ただし，2 回目以降は正しいリクエストオブジェク

トを設定してください。

パラメタ

req - イベント処理に渡されたリクエストオブジェクト

戻り値

なし

例外

IllegalStateException - 引数が不正または null の場合

setUserCustomizeInfo

形式

```
public void setUserCustomizeInfo(String appKey,
                                 Object appData)
```

機能

アプリケーションキーに対応するユーザのカスタマイズ情報を保存します。
<jsp:getProperty> タグ，および <jsp:setProperty> タグでの操作はできません。
ログインしていない場合，何もしません。
ユーザ単位に情報を保存するため，ポートレット間でデータを受け渡せますが，ポートレットごとにアプリケーションキーが衝突しないように注意する必要があります。また，"hptl" で始まるキー名は予約されています。

パラメタ

appKey - アプリケーションキー（null 以外）
appData - カスタマイズ情報（シリアライズできる値）

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

14.7 ログ出力 Bean

ログ出力 Bean は，ポートレットからログを出力するための API です。

ログ出力 Bean の一覧を次の表に示します。

表 14-7 ログ出力 Bean の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.log.PortletLog	ポートレットからログを出力します。

ポートレットログの出力レベルによって，出力されるログの種類が異なります。そのため，ログ出力 Bean のメソッドは，次の方針に従って使用します。方針に従わない場合，必要なログが得られない，または大量のログが出力され性能が著しく低下するおそれがあるので注意してください。

ログ出力 Bean のメソッドと出力するログ内容を次の表に示します。

表 14-8 ログ出力 Bean のメソッドと出力するログ内容

使用するメソッド	出力レベル	内容
error，printStackTrace	0	ポートレットの起動・終了のメッセージ，および業務に支障があるエラーメッセージを出力します。トランジェントオブジェクトの起動や終了のメッセージはこのレベルでは出力しません。
warn	10	主要なリクエストをトレースし，システムの大まかな動きが把握できるレベルのログを出力します。発生頻度の低い障害の再発監視や本番前の現地テストへの適用を想定しています。トレースの出力量をシステムの性能に影響を与えない範囲に抑える必要があります。
note	20	再現性のある障害に対し，障害の部位を明確に分類するために使用します。ほかのプロセスとの通信など，ほかのプログラムとの関連を把握するのに十分な情報，およびプログラムの大体の動作を把握できるトレース情報を出力します。
debug	30	障害の個所を特定するために使用します。主要なメンバ関数の開始や終了が取得およびトレースできる内容を出力する必要があります。

ログ出力 Bean の詳細を説明します。

jp.co.hitachi.soft.portal.api.log.PortletLog

機能

ポートレットからログを出力します。

Bean の項目

宣言

```
<jsp:useBean id="Log"
class="jp.co.hitachi.soft.portal.api.log.PortletLog"
scope="session" />
```

対象スコープ

```
session, request
```

request は性能が低下するため session を推奨します。

メソッドの一覧

戻り値のデータ型	メソッド	説明
void	debug(Object obj)	デバッグログを出力します。
void	error(Object obj)	エラーログを出力します。
void	error(Object obj, Throwable t)	エラーログおよびスタックトレースを出力します。
void	note(Object obj)	情報ログを出力します。
void	printStackTrace(Throwable t)	スタックトレースを出力します。
void	**setApplication(String appName)**	**アプリケーション名を設定します。必ず最初にこのメソッドを呼び出してください。**
void	warn(Object obj)	警告ログを出力します。
void	warn(Object obj, Throwable t)	警告ログおよびスタックトレースを出力します。

メソッドの説明

debug

形式

```
public void debug(Object obj)
```

機能

ログに対してデバッグログとして obj.toString() の結果を出力します。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。setApplication() メソッドを呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。

ログの出力条件を次に示します。

- メッセージ ID：KDPF99990-I
- ログ出力レベルが 30 のときに出力
- 種別：なし

パラメタ

obj - ログに出力する内容

戻り値
なし

例外
なし

error〔obj.toString() の結果をエラーログとして出力〕

形式

```
public void error(Object obj)
```

機能
ログに対してエラーログとして，obj.toString() の結果を出力します。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。setApplication() メソッドを呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。
ログの出力条件を次に示します。
- メッセージ ID：KDPF99993-E
- ログ出力レベルが 0 のときに出力
- 種別：常に "ER"

パラメタ
obj - ログに出力する内容

戻り値
なし

例外
なし

error〔String.valueOf(obj) の結果をエラーログとして出力〕

形式

```
public void error(Object obj,
                  Throwable t)
```

機能
ログに対してエラーログとして String.valueOf(obj) の結果を出力し，その次の行にスタックトレースを出力します。ただし，t が null の場合はスタックトレースを出力しません。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。setApplication() メソッドを呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。
ログの出力条件を次に示します。
- メッセージ ID：KDPF99993-E
- ログ出力レベルが 0 のときに出力
- 種別：常に "ER"

パラメタ
obj - ログに出力する内容

t - 警告を出力するきっかけとなった Throwable

戻り値
なし

例外
なし

note

形式
```
public void note(Object obj)
```

機能
ログに対して情報ログとして obj.toString() の結果を出力します。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。
setApplication() メソッドを呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。
ログの出力条件を次に示します。
- メッセージ ID：KDPF99991-I
- ログ出力レベルが 20 のときに出力
- 種別：なし "

パラメタ
obj - ログに出力する内容

戻り値
なし

例外
なし

printStackTrace

形式
```
public void printStackTrace(Throwable t)
```

機能
ログに対してデバッグログとしてスタックトレースを出力します。ただし，t が null の場合は何もしません。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。setApplication() メソッドを呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。
ログの出力条件を次に示します。
- メッセージ ID：KDPF99990-I
- ログ出力レベルが 30 のときに出力
- 種別：なし

パラメタ
t - 警告を出力するきっかけとなった Throwable

戻り値
なし

例外
なし

setApplication

形式
```
public void setApplication(String appName)
```

機能
ログ出力 Bean のアプリケーション名を設定します。このアプリケーション名は，ログ中のアプリケーション名の文字列として表示されます。appName が null または英数字以外を含む場合，アプリケーション名は設定されません。このとき，setApplication() メソッド後に呼ばれたログ出力メソッドについても何も行われません。このメソッドはインスタンスに対して 1 回だけ有効で，2 回目以降は何もしないで戻ります。

パラメタ
appName - アプリケーション名（NULL 以外の英数字）

戻り値
なし

例外
IllegalArgumentException - appName に英数字以外が渡された場合：詳細メッセージは " There is an illegal letter in appName."
IllegalArgumentException - appName が null の場合：詳細メッセージは "appName is null."

warn〔obj.toString() の結果を警告ログとして出力〕

形式
```
public void warn(Object obj)
```

機能
ログに対して警告ログとして，obj.toString() の結果を出力します。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。
setApplication() を呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。
ログの出力条件を次に示します。
- メッセージ ID：KDPF99992-W
- ログ出力レベルが 10 のときに出力
- 種別：なし

パラメタ
obj - ログに出力する内容

戻り値

なし

例外

なし

warn〔String.valueOf(obj) の結果を警告ログとして出力し，次の行にスタックトレースを出力〕

形式

```
public void warn(Object obj,
                 Throwable t)
```

機能

ログに対して String.valueOf(obj) の結果を出力し，その次の行にスタックトレースを出力します。ただし，t が null の場合はスタックトレースを出力しません。このメソッドを呼び出す前に，setApplication() メソッドを呼び出す必要があります。setApplication() を呼び出す前にこのメソッドを呼び出すと，このメソッドは何もしないで戻ります。

ログの出力条件を次に示します。

- メッセージ ID：KDPF99992-W
- ログ出力レベルが 10 のときに出力
- 種別：t が null 以外の場合は "EC"，t が null の場合はなし

パラメタ

obj - ログに出力する内容

t - 警告を出力するきっかけとなった Throwable

戻り値

なし

例外

なし

14.8 UserAgent 情報取得 Bean

UserAgent 情報取得 Bean は，現在アクセスしているクライアントに対して，クライアント情報定義ファイルで設定されている UserAgentType を取得するための API です。ポートレットユティリティタグライブラリにある <ut:convert> 内の xslt 属性で，UserAgentType 属性によって XSLT スタイルシートを変換するときに使用します。

UserAgent 情報取得 Bean の一覧を次の表に示します。

表 14-9 UserAgent 情報取得 Bean の一覧

クラス名	説明
jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean	UserAgentType の情報を取得します。

UserAgent 情報取得 Bean の詳細を説明します。

jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean

機能

UserAgentType の情報を取得します。

Bean の項目

宣言

```
<jsp:useBean id="useragentinfobean"
class="jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBea
n"
scope="session"/>
```

対象スコープ

```
session
```

メソッドの一覧

戻り値のデータ型	メソッド	説明
String	getUserAgentGroup()	UsarAgentType の値を取得します。
void	initBean(ServletRequest req)	Bean を初期化します。

メソッドの説明

getUserAgentGroup

形式

```
public String getUserAgentGroup()
```

機能

現在のクライアントで設定されている useragentType オブジェクトの値を取得します。

パラメタ

なし

戻り値

現在のクライアントの useragentType 値
useragentType に値がない場合は，null を戻します。

例外

なし

initBean

形式

```
public void initBean(ServletRequest req)
```

機能

UserAgentType の値を取得する Bean を使用する前に初期化します。
UserAgentInfoBean を使用する場合は，必ずこのメソッドで初期化します。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

戻り値

なし

例外

IllegalArgumentException - 引数 req が null または不正なオブジェクトである場合

14.9 ポートレットイベント API

ポートレットイベント API は，アクションイベント機能が提供する API です。

ポートレットイベント API の一覧を次の表に示します。

表 14-10 ポートレットイベント API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.portlet.api.DefaultActionModule	ポートレット間通信機能を提供するクラスです。

ポートレットイベント API の詳細を説明します。

jp.co.hitachi.soft.portal.portlet.api.DefaultActionModule

機能

ポートレット間通信機能を提供するクラスです。

使用方法

必要に応じて次に示すメソッドをオーバーライドしてください。

1. action メソッド
   ポートレットアクションイベント発生時に呼び出されます。
   利用者から uCosminexus Portal Framework に対して要求が発生したとき，そのポートレットが画面上に表示されていれば呼び出されます。
2. receive メソッド
   ポートレット間通信イベント発生時に呼び出されます。
   あるポートレットのアクションモジュール内から，このポートレットを対象として send メソッドが呼び出された場合に，この receive メソッドが呼び出されます。

なお，次の場合は，後述のデフォルトアクションモジュールの動作をします。

- イベント（メソッド）を定義しなかった場合
- アクションモジュールが存在しない場合

アクションモジュールは，ポートレット単位で指定できます。

注意事項

- uCosminexus Portal Framework はアクションモジュールごとにインスタンスを保持します。
  したがって，アクションモジュールでインスタンス変数を定義しても，それらは static 変数のように共有化されています。

- アクションモジュールは利用者要求（リクエスト）単位のシングルスレッドで実行されます。
  イベント内で処理負荷が掛かる記述をすると，uCosminexus Portal Framework 全体の性能を低下させるおそれがあるので注意してください。
  ただし，ポートレット並列表示機能使用時は，ポートレットごとに並列動作ができます。ポートレット並列表示機能については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ポートレット並列表示」の説明を参照してください。

メソッドの一覧

戻り値のデータ型	メソッド	説明
void	action(HttpServletRequest req, HttpServletResponse res)	ポートレットアクションイベント処理（推奨）
void	action(HttpServletRequest req)	ポートレットアクションイベント処理（非推奨）
final java.lang.Object	getMessage(javax.servlet.http.HttpServletRequest request)	ポートレット間通信イベントの場合に通信元が設定したメッセージを取得します。
final jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean	getPortletInfo(javax.servlet.http.HttpServletRequest request,java.lang.String beanid)	ポートレット情報取得 Bean 取得メソッド
final jp.co.hitachi.soft.portal.api.log.PortletLog	getPortletLog(javax.servlet.http.HttpServletRequest request,java.lang.String beanid)	ポートレットログ出力 Bean 取得メソッド
final java.lang.String	getPortletName(javax.servlet.http.HttpServletRequest request)	処理中のポートレット名称を取得します。
final java.lang.String	getSrcPortletName(javax.servlet.http.HttpServletRequest request)	ポートレット間通信イベントの場合に通信元のポートレット名称を取得します。
final jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean	getUserAgentInfoBean(javax.servlet.http.HttpServletRequest request,java.lang.String beanid)	UserAgent 情報取得 Bean 取得メソッド
final jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean	getUserInfo(javax.servlet.http.HttpServletRequest request,java.lang.String beanid)	ユーザ情報取得 Bean 取得メソッド

戻り値のデータ型	メソッド	説明
void	receive(HttpServletRequest req, HttpServletResponse res)	ポートレット間通信イベント処理（推奨）
void	receive(HttpServletRequest req)	ポートレット間通信イベント処理（非推奨）
final void	send(javax.servlet.http.HttpServletRequest request,java.lang.String portletName,java.lang.Object message)	ポートレット間通信をしたい場合にメッセージを送信します。

メソッドの説明

action〔使用を推奨するメソッド〕

形式

```
public void action(HttpServletRequest request,
HttpServletResponse response)
```

機能

Web ブラウザからフォームデータを入力した，アンカーをクリックしたなどの利用者要求が発生した場合に呼び出されます。
ただし，そのポートレットが画面上に表示されている場合だけです。

利用方法

利用者要求が発生した場合で，何らかの処理が必要なときはこのメソッドをオーバーライドしてください。

利用手順

1. どのポートレットに対して要求があったのかをチェックします。
   getPortletName()
   注　この処理は推奨しません。ポートレットを判断しなくてもよい処理にしてください。
2. インタフェースの内容を参照して処理を実行します。
   request.getAttribute(key) の内容
3. インタフェースを設定します。
   Web ブラウザからの要求を取得する場合
   次のパラメタ取得メソッドを使用します。
   getParameter(String)，getParameterMap()，getParameterNames()，getParameterValues(String)
   ほかのポートレットのインタフェースを設定する場合
   send メソッドにメッセージを設定します。
   相手側ポートレットは，receive メソッドで受信します。
   JSP やサーブレットに対してインタフェースを設定する場合
   次の値を利用します。

[リクエスト] に設定
request.setAttribute(key, value)
[セッション] に設定
request.getSession().setAttribute(key, value)
なお，request.setAttribute(key, value) した結果は，各ポートレットに独立して引き渡されます。
例えば，ポートレット A のポートレットアクションモジュールで，request.setAttribute("test", "a") と設定したあと，ポートレット B のポートレットアクションモジュールで，request.setAttribute("test", "b") と設定したとします。各ポートレットの JSP やサーブレットに対してポートレット A には，request.getAttribute("test") で，"a" を返します。そして，ポートレット B には，request.getAttribute("test") で，"b" を返します。
なお，そのほかのポートレット（例えば，ポートレット C など）には，最後に設定したポートレットの情報である "b" が返されます。したがって，全ポートレット共通にパラメタを持つ場合には，一意となるようなパラメタ名称を設定してください。

4. 特定のポートレットに対して処理を依頼したり，状態変化を通知したい場合，次の処理を実行します。

- このメソッド内で通知先ポートレットと，メッセージ内容を決定して send メソッドを呼び出します。

パラメタ
request - イベント処理に渡されたリクエストオブジェクト
response - イベント処理に渡されたレスポンスオブジェクト

戻り値
なし

例外
なし

注意事項
- response に何らかの操作を行っても無視されます。
- response パラメタの getOutpuStream() メソッドまたは getWriter() メソッドを呼び出した場合，JSP（ポートレット）側に制御を渡す処理で java.lang.IllegalStateException が発生する場合があります。

action〔使用を推奨しないメソッド〕

形式
```
public void action(HttpServletRequest req)
```

機能
旧バージョンとの互換のためのメソッドです。そのため，このメソッドを使用することは，推奨しません。

パラメタ

req - イベント処理に渡されたリクエストオブジェクト

戻り値

なし

例外

java.lang.IllegalArgumentException - request が null の場合
ログにエラー情報が出力されますので確認してください。

getMessage

形式

```
public final java.lang.Object
getMessage(javax.servlet.http.HttpServletRequest request)
```

機能

ポートレット間通信イベントの場合，通信元が設定したメッセージを取得します。receive メソッドが呼ばれた場合に使用してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

戻り値

通信元のメッセージ内容
メッセージが設定されていない場合は null を返します。

例外

java.lang.IllegalArgumentException - request が null の場合

getPortletInfo

形式

```
public final
jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean
   getPortletInfo(javax.servlet.http.HttpServletRequest request,
                  java.lang.String beanid)
```

機能

ポートレット情報取得 Bean 取得メソッド
ポートレットの情報を取得するための API オブジェクトを取得します。
ポートレットのパラメタを取得したい場合は，ポートレット定義ファイル (jetspeed-config.jcfg) または Portal Manager で設定します。
そのほかに，ポートレットのタイトル，説明文を取得できます。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト
beanid - JSP などで定義している bean 名称を指定してください。

- JSP ファイルの jsp:useBean アクションで指定する scope 属性は，page でなければなりません。
- ポートレット情報取得 Bean オブジェクトが存在しない場合は，新たに生成

します。
- 新たに bean を生成した場合，その情報は page に保持されます。

戻り値

PortletInfoBean ポートレット情報取得 Bean オブジェクト（null では返却されません）

例外

java.lang.IllegalArgumentException - 次の場合に発生します。
- uCosminexus Portal Framework が状態不正を検出した場合
- 引数の request または beanid が null の場合

ログにエラー情報が出力されますので確認してください。

getPortletLog

形式

```
public final jp.co.hitachi.soft.portal.api.log.PortletLog
    getPortletLog(javax.servlet.http.HttpServletRequest request,
                  java.lang.String beanid)
```

機能

ポートレットログ出力 Bean 取得メソッド
- ポートレット単位でログをファイルに出力するための情報が取得できます。

利用方法

1. jp.co.hitachi.soft.portal.api.log.PortletLog パッケージをインポートしてください。
2. action および receive メソッド内で，getPortletLog メソッドを呼び出すことによって，ポートレットログ出力 Bean オブジェクトを取得できます。
3. ログ出力 Bean のメソッドを利用して，目的の情報にアクセスしてください。
   ログ出力 Bean のメソッドについては，「14.7　ログ出力 Bean」を参照してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

beanid - JSP 等で定義している bean 名称を指定してください。
- JSP ファイルの jsp:useBean アクションで指定する scope 属性は，session または request でなければなりません。
- session と request 双方に，指定した beanid が設定されている場合は session を優先的に利用します。
- session にポートレットログ出力 Bean でないオブジェクトが設定されていて，request にポートレットログ出力 Bean が設定されている場合は，request に設定されているポートレットログ出力 Bean を取得します。
- session と request 双方に存在しない場合は，ポートレットログ出力 Bean オブジェクトを保持しません。

戻り値

PortletLog ポートレットログ出力 Bean オブジェクト（null では返却されません）

例外

java.lang.IllegalArgumentException - 次の場合発生します。

- 引数の beanid または request が null の場合

ログにエラー情報が出力されますので確認してください。

getPortletName

形式

```
public final java.lang.String
getPortletName(javax.servlet.http.HttpServletRequest request)
```

機能

処理中のポートレット名称を取得します。action または receive メソッドが呼ばれた場合に使用してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

戻り値

処理中のポートレット名称

例外

java.lang.IllegalArgumentException - request が null の場合

getSrcPortletName

形式

```
public final java.lang.String
getSrcPortletName(javax.servlet.http.HttpServletRequest
request)
```

機能

ポートレット間通信イベントの場合，通信元のポートレット名称を取得します。receive メソッドが呼ばれた場合に使用してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

戻り値

通信元のポートレット名称

例外

java.lang.IllegalArgumentException - request が null の場合

getUserAgentInfoBean

形式

```
public final
jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean
```

```
getUserAgentInfoBean(javax.servlet.http.HttpServletRequest
request,
                              java.lang.String beanid)
```

機能

UserAgent 情報取得 Bean 取得メソッド

- 要求されたリクエストに関して，ユーザエージェントを判定します。
- クライアント情報定義ファイル（UserAgentType.xml）で定義している useragentType を取得できます。

利用方法

1. jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean パッケージをインポートしてください。
2. action および receive メソッド内で，getUserAgentInfoBean メソッドを呼び出すことによって，UserAgent 情報取得 Bean オブジェクトを取得できます。
   Bean 初期化のための initBean メソッドを明示的に呼び出す必要はありません。
3. UserAgent 情報取得 Bean のメソッドを利用して，目的の情報にアクセスしてください。
   UserAgent 情報取得 Bean のメソッドについては，「14.8　UserAgent 情報取得 Bean」を参照してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

beanid - JSP などで定義している bean 名称を指定してください。

- JSP ファイルの jsp:useBean アクションで指定する scope 属性は，session でなければなりません。
- 指定した beanid の UserAgent 情報取得 Bean オブジェクトが存在しない場合は，新たに生成します。
- 新たに bean を生成した場合，その情報は session に保持されます。

戻り値

UserAgent 情報取得 Bean オブジェクト（null で返却されることはありません）

例外

java.lang.IllegalArgumentException - 次の場合発生します。

- 引数の beanid または request が null の場合
- uCosminexus Portal Framework が状態不正を検出した場合

ログにエラー情報が出力されますので確認してください。

getUserInfo

形式

```
public final
jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean
          getUserInfo(javax.servlet.http.HttpServletRequest
```

```
request,
                                java.lang.String beanid)
```

機能

ユーザ情報取得 Bean 取得メソッド

現在ログインしているユーザの情報を取得したり，各ユーザのポートレット固有の情報にアクセスしたりできます。

- 現在ログインしているユーザのユーザ情報，組織情報，およびグループ情報が取得できます。
  これは，リポジトリファイル（Repository.xml）に定義された LDAP/DB で定義している情報です。
- 各ユーザのポートレット固有の情報にアクセスできます。
  これは，LDAP/DB に保持している情報です。

利用方法

1. jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean パッケージをインポートしてください。
2. action および receive メソッド内で，getUserInfo メソッドを呼び出すことによって，ユーザ情報取得 Bean オブジェクトを取得できます。
   Bean 初期化のための setRequest メソッドを明示的に呼び出す必要はありません。
3. ユーザ情報取得 Bean のメソッドを利用して，目的の情報にアクセスしてください。
   ユーザ情報取得 Bean のメソッドについては，「14.6　ユーザ情報取得 Bean」を参照してください。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

beanid - JSP などで定義している bean 名称を指定してください。

- JSP ファイルの jsp:useBean アクションで指定する scope 属性は，session でなければなりません。
- 指定した beanid のユーザ情報取得 Bean オブジェクトが存在しない場合は，新たに生成します。
- 新たに bean を生成した場合，その情報は session に保持されます。

戻り値

ユーザ情報取得 Bean オブジェクト（null では返却されません）

例外

java.lang.IllegalStateException -

uCosminexus Portal Framework が状態不正を検出した場合に発生します。ログにエラー情報が出力されますので確認してください。

java.lang.IllegalArgumentException -

引数の beanid または request が null の場合に発生します。ログにエラー情報が出力されますので確認してください。

receive〔使用を推奨するメソッド〕

形式

```
public void receive(HttpServletRequest request,
                    HttpServletResponse response)
```

機能

ポートレット間通信イベント処理
ほかのポートレットから send メソッドを呼ばれた場合に，このメソッドが呼び出されます。

利用方法

ほかのポートレットから send メソッドを呼ばれた場合，何らかの処理が必要なときはこのメソッドをオーバーライドしてください。

利用手順

1. どのポートレットに対して要求があったのかチェックします。
   getPortletName()
2. どのポートレットから要求があったのかチェックします。
   getSrcPortletName()
3. どのようなメッセージをポートレットから受信したのかチェックします。
   getMessage()
4. インタフェースの内容を参照して処理を実行します。
   request.getAttribute(key) の内容
5. インタフェースを設定します。
   詳細は action メソッドを参照してください。
6. 特定のポートレットに対して処理を依頼したり，状態変化を通知したい場合，次の処理を実行します。

- このメソッド内で send メソッドを呼び出します。
  send（リクエストオブジェクト，送信先ポートレット名称，送信メッセージ）

パラメタ

request - イベント処理に渡されたリクエストオブジェクト
response - イベント処理に渡されたレスポンスオブジェクト

戻り値

なし

例外

java.lang.IllegalStateException - response パラメタの getOutpuStream() メソッドまたは getWriter() メソッドを呼び出した場合，JSP（ポートレット）側に制御を渡す処理で発生する場合があります。

注意事項

response に何らかの操作を行っても無視されます。

receive〔使用を推奨しないメソッド〕

形式

```
public void receive(HttpServletRequest req)
```

機能

ポートレット間通信イベント処理
ほかのポートレットから send メソッドを呼ばれた場合に，このメソッドが呼び出されます。
このメソッドを使用することは，推奨しません。

利用方法

ほかのポートレットから send メソッドを呼ばれた場合，何らかの処理が必要なときはこのメソッドをオーバーライドしてください。

利用手順

1. どのポートレットに対して要求があったのかチェックします。
   getPortletName()
2. どのポートレットから要求があったのかチェックします。
   getSrcPortletName()
3. どのようなメッセージをポートレットから受信したのかチェックします。
   getMessage()
4. インタフェースの内容を参照して処理を実行します。
   request.getAttribute(key) の内容
5. インタフェースを設定します。
   詳細は action メソッドを参照してください。
6. 特定のポートレットに対して処理を依頼したり，状態変化を通知したりしたい場合，次の処理を実行します。

- このメソッド内で send メソッドを呼び出します。
  send（リクエストオブジェクト，送信先ポートレット名称，送信メッセージ）

パラメタ

req - イベント処理に渡されたリクエストオブジェクト

戻り値

なし

例外

java.lang.IllegalArgumentException - request が null の場合

ログにエラー情報が出力されますので確認してください。

send

形式

```
public final void send(javax.servlet.http.HttpServletRequest
request,
                       java.lang.String portletName,
                       java.lang.Object message)
```

機能

ポートレット間通信をしたい場合，メッセージを送信します。

action メソッドまたは receive メソッドから呼び出せます。

パラメタ

request - イベント処理に渡されたリクエストオブジェクト

portletName - 通知先ポートレット名称

message - 相手側に通知するメッセージオブジェクト

戻り値

なし

例外

java.lang.IllegalArgumentException - 次の場合，例外が発生します。

- request，portletName が null の場合
- portletName で指定されたポートレットが，Cosminexus Portal Framework に未登録である場合
- portletName で指定されたポートレットを，この利用者が利用できない（アクセス権限がない）場合
- デフォルトアクションモジュールを不正に生成した場合

ログにエラー情報が出力されますので確認してください。

14.10 ポートレットパラメタ取得インタフェース

ポートレットパラメタ取得インタフェースは，ほかのポートレットに設定情報などのパラメタを受け渡すためのインタフェースです。

ポートレットパラメタ取得インタフェースの一覧を次の表に示します。

クラス名	説明
jp.co.hitachi.soft.portal.api.portlets.PortletParameters	ポートレット間でパラメタを受け渡すためのインタフェースです。

ポートレットパラメタ取得インタフェースの詳細を説明します。

jp.co.hitachi.soft.portal.api.portlets.PortletParameters

機能

ポートレット間でパラメタを受け渡すためのインタフェースです。

使用方法

ポートレットでこのインタフェースを実装した場合，各ポートレットデプロイ定義ファイル（hportlet.xml）の中に以下の形式で実装クラスを指定します。

<param-name>hptl.portlet.parameters</param-name>

<param-value>PortletParameters 実装クラスの FQCN（パッケージ名を含むクラス名）</param-value>

PortletParameters 実装クラスの呼び出し側では，上記より取得した FQCN から Class.forName でオブジェクトを取得して，newInstance メソッドでインスタンス化します。そのインスタンスに対して setRequest メソッドを発行後，ウィンドウサイズ情報の取得用のキーなどを指定して getParameter メソッドをコールすることで，対応する情報を取得します。newInstance で PortletParameters のインスタンスを生成する呼び出し側では，JSP ページ内や request スコープ内でインスタンスを解放し，このインスタンスを複数ページ／複数スレッド間で共有してはいけません。

メソッドの一覧

戻り値のデータ型	メソッド	説明
void	setRequest(javax.servlet.http.HttpServletRequest req)	リクエストオブジェクトを設定します。必ず最初にこのメソッドを呼び出してください。

戻り値のデータ型	メソッド	説明
java.lang.String	getParameter(java.lang.String key)	指定したキーに対応するポートレットの設定値を取得します。

メソッドの説明

setRequest

形式

```
public void setRequest(javax.servlet.http.HttpServletRequest req)
```

機能

このメッソドを発行することで，PortletParameters 実装クラスから HttpServletRequest オブジェクトを利用できるようになります。

PortletParameters 実装クラスの呼び出し側は，必ず最初にこのメソッドを呼び出す必要があります。

パラメタ

req - HttpServletRequest オブジェクト

戻り値

なし

getParameter

形式

```
public java.lang.String getParameter(java.lang.String key)
```

機能

指定したキーに対応するポートレットの設定値を取得します。

パラメタ

key - 設定値を取得するためのキー値

戻り値

キーに対応する設定値。設定値を返せない場合には null を返す。

14.11 クライアントサイドデータ通信タグライブラリ

14.11.1 タグライブラリの概要

クライアントサイドデータ通信タグライブラリは，シンプルで統一感のあるユーザインタフェースを実現するためのAPIを提供します。このAPIを利用して，ドラッグ＆ドロップ機能およびデータフォーム転送機能の二つの機能を実現できます。

このタグライブラリを使用する際の前提となる条件を次に示します。

1. Webアプリケーションの DD（web.xml）内のタグライブラリに関する要素の追加
   uCosminexus Portal Framework にポートレットをインストールする際，<web-app>の子要素に次の要素を追加します。なお，この要素は，新規にポータルサーバを構築した場合に追加されます。

```
<taglib>
    <taglib-uri>http://soft.hitachi.co.jp/portal/api/csdc</
taglib-uri>
    <taglib-location>/WEB-INF/cosmi/portal/taglib/csdc.tld</
taglib-location>
</taglib>
```

   web.xmlの格納場所を次に示します。

   {PROJECT_HOME}¥WEB-INF

   注意事項

   <web-app>の文書型定義（DTD定義）に沿った順番で定義してください。

2. taglibディレクティブの追加
   該当するJSPファイルに，次に示すtaglibディレクティブの定義を記述します。

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/api/csdc"
prefix="csdc" %>
```

3. csdc.jspへのマッピング定義に関する要素の追加
   uCosminexus Portal Frameworkにポートレットをインストールする際，
   <web-app>の子要素に次の要素を設定します。

```
  <servlet>
    <servlet-name>csdc</servlet-name>
    <jsp-file>/js/csdc.jsp</jsp-file>
  </servlet>

  <servlet-mapping>
    <servlet-name>csdc</servlet-name>
    <url-pattern>/js/csdc.js</url-pattern>
  </servlet-mapping>
```

   注意事項

   <web-app>の文書型定義（DTD定義）に沿った順番で定義してください。

クライアントサイドデータ通信タグライブラリの一覧を次の表に示します。

表 14-11 クライアントサイドデータ通信タグライブラリの一覧

タグ名	説明	JSP スクリプトレット式で評価される属性値
csdc:body	コピー元およびコピー先の HTML 構成を定義します。<csdc:src> および <csdc:dest> の子要素として利用できます。	―
csdc:dest	データ転送先オブジェクトを定義します。	datatype，nodetype
csdc:dragicon	ドラッグ用のアイコン（img 要素）を出力します。<csdc:src> の <csdc:body> の子要素としてだけ利用できます。	desc，escape，hid，src，align，type
csdc:oncopy	コピーイベントハンドラを作成します。<csdc:src> の子要素としてだけ利用できます。	name
csdc:onpaste	ペーストイベントハンドラを定義します。<csdc:dest> の子要素としてだけ利用できます。	name
csdc:serializeJ2H	Java オブジェクトをシリアライズします。次の Java データ型をサポートします。 • JavaBeans • HashMap クラス これら以外の Java データ型はサポートしません。	data，type
csdc:src	データ転送元のオブジェクトを定義します。	datatype
csdc:use	クライアントサイドデータ通信機能を使用するための初期定義をします。このタグライブラリは，ポートレット／テンプレート上で <body> タグ内に 1 回だけ定義します。また，フレームで使用する場合には，そのフレームを取り込んでいるウィンドウの <body> タグ内に定義します。	type

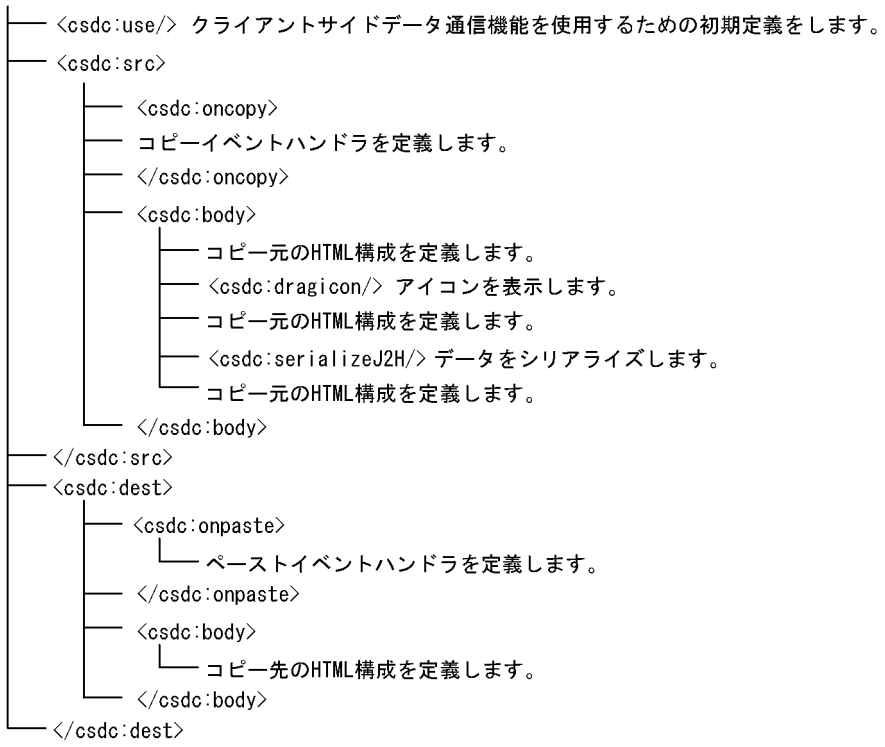
（凡例）

―：該当しません。

14.11.2 タグライブラリのタグ階層

クライアントサイドデータ通信機能を実現するために使用するタグライブラリのタグ階層を次に示します。ポートレット開発者は，階層に従ってタグライブラリを定義する必要があります。

なお，クライアントサイドデータ通信 API で例外が発生すると，javax.servlet.jsp.JspException が発生します。また，クライアントサイドデータ通信 API はライブラリ名と例外内容をログファイルに出力します。



注　タグライブラリのcsdcプレフィックスは，「Client Side Data Communication」の略です。

14.11.3　タグライブラリの説明

(1) csdc:body

機能

コピー元およびコピー先の HTML 構成を定義します。<csdc:src> および <csdc:dest> の子要素として利用できます。

構文

<csdc:src> の子要素の場合

```
<csdc:body>
    コピー元のHTML構成定義
    <csdc:dragicon />
    コピー元のHTML構成定義
    <csdc:serializeJ2H />
</csdc:body>
```

<csdc:dest> の子要素の場合

```
<csdc:body>
    コピー先のHTML構成定義
    <csdc:dragicon />
    コピー先のHTML構成定義
```

```
        <csdc:serializeJ2H />
    </csdc:body>
```

属性

なし

HTML 変換規則

<csdc:src> の子要素の場合

<csdc:body> タグ内に記述された内容が表示されます。

<csdc:dest> の子要素の場合

1. ドラッグ & ドロップの場合

- nodetype == block **の場合**

```
<div ondrop="hptl_onpaste(‘(I)’,(II),(III))"
ondragenter="cancelDefault('(I)')"
ondragover="cancelDefault(‘(I)')">
<csdc:body>タグ内に記述された内容が表示される。
</div>
```

- nodetype == inline **の場合**

```
<span ondrop="hptl_onpaste(‘(I)’,(II),(III))"
ondragenter="cancelDefault('(I)')"
ondragover="cancelDefault(‘(I)')">
<csdc:body>タグ内に記述された内容が表示される。
</span>
```

（凡例）

(I)：<csdc:dest> タグの datatype 属性に指定した値

(II)：ペーストイベントハンドラ名称

(III)：ペーストハンドラに渡すオブジェクト

2. データフォーム転送の場合

<csdc:body> タグ内に記述された内容が表示されます。

(2) csdc:dest

機能

データ転送先オブジェクトを定義します。

構文

属性を定義する順序は任意です。

```
<csdc:dest datatype="data type" nodetype="{block|inline}">
    <csdc:onpaste> // 子要素
    <csdc:body>    // 子要素
</csdc:dest>
```

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
datatype	String	・この属性には，単一の文字列だけ指定できます。 ・定義フォーマットを次に示します。 「**ネームスペース名 . 通信データオブジェクト名称**」 ・ネームスペース名が Java クラスライブラリ名と同じ場合は，ドメインの文字列を反転させたものを記述します。	必須	○
nodetype	String	<csdc:body> の HTML タグの要素を指定します（ブロックレベル要素またはインライン要素）。	必須	○

（凡例）

○：JSP スクリプトレット式で評価されます。

HTML 変換規則

<csdc:dest> タグ内に記述された内容が表示されます。

(3) csdc:dragicon

機能

ドラッグ用のアイコン（img 要素）を出力します。<csdc:src> の <csdc:body> の子要素としてだけ利用できます。

構文

属性を定義する順序は任意です。

```
<csdc:dragicon [type="ポートレット任意の文字列"] [src="path to
icon"]
[desc="アイコンの説明"] hid="任意文字列" escape="single-quote |
double-quote"
[hclass="ドラッグ用のアイコン(img要素)のclass属性"]
[align="ドラッグ用のアイコン(img要素)のalign属性"]>
```

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
desc	String	アイコンの説明を記述します。HTML では，ツールチップとして出力されます。	任意	○

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
escape	String	このタグライブラリが出力する HTML の文字列の文字列囲み文字「”」,「’」を，ポートレットで制御したい場合に指定します。省略した場合は，「single-quote」が指定されます。通常は指定しないでください。 • single-quote：文字列囲み文字「’」を利用して出力する場合に指定します。 • double-quote：文字列囲み文字「”」を利用して出力する場合に指定します。	任意	○
hclass	String	アイコンのスタイルシートのクラスを指定するための class 属性を指定します。	任意	○
hid	String	ドラッグ & ドロップに使用するアイコンを，ポートレットで一意に識別したい場合に付け加える ID です。通常は指定しないでください。	任意	○
src	String	ユーザが定義したアイコンの URL を指定します。省略した場合は，「item.gif」を表示します。「item.gif」のドラッグ用アイコンについては，「図 9-1 ドラッグ & ドロップ機能の概要」に示す「ドラッグ用アイコン」を参照してください。	任意	○
align	String	アイコンの表示位置を指定するための align 属性を指定します。	任意	○
type	String	• ポートレットごとの組み込み型を指定します。（例：people） • この属性は，src 属性で指定するアイコンがどのような種別であるかを示すコメントの役割を果たします。 • この属性に定義された文字列は，このタグライブラリから参照されることはないため，不正な種別文字列が定義されていてもエラーにはなりません。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

HTML 変換規則

<csdc:dragicon> の HTML 変換規則を次に示します。

表 14-12 <csdc:dragicon> の HTML 変換規則

変換前 / 後	変換規則
変換前	`<csdc:dragicon type="people" hid=”0” src=”http://○○○/×××/people.gif” desc=”人の情報です” />`
変換後※	`<img id=”1” src=”http://○○○/×××/people.gif" alt=”人の情報です” ondragstart="hptl_oncopy('1.',2.,3.)" >`

（凡例）

1.：<csdc:src> の name 属性に指定した値

2.：コピーイベントハンドラ名称

3.：イベント対象オブジェクト（this）

注※　変換前の属性の定義順に関係なく，表中の順序で属性は表示されます。

（4） csdc:oncopy

機能

コピーイベントハンドラを作成します。<csdc:src> の子要素としてだけ利用できます。

構文

```
<csdc:oncopy [name="function name"]>
    コピーイベントハンドラの処理※
</csdc:oncopy>
```

注※　ここでは，JavaScript を使用し，次の処理を定義します。

1. Java オブジェクトがシリアライズされた文字列を，HTML から取得します。
2. 1. で取得した文字列を JavaScript 関数「hptl_deserializeH2JS」により，JavaScript オブジェクトにデシリアライズします。
3. JavaScript 関数「hptl_setPortalClipboardData」により，2. のオブジェクトをポータルクリップボードウィンドウにセットします。

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
name	String	・「コピーイベントハンドラ」の関数名称を定義します。関数名称を定義することで，任意の JavaScript コードから呼出しができるようになります。ただし，ページ内で関数名をユニークとなるように注意する必要があります。 ・他 JavaScript コードから呼び出しがない場合は省略できます。 ・省略した場合は，イベントハンドラの名称は「**”** hptl_uap_oncopy **”+ ページ内でユニークな通番**」となります。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

HTML 変換規則

<csdc:oncopy> の HTML 変換規則を次に示します。

表 14-13 <csdc:oncopy> の HTML 変換規則

変換前 / 後	name 属性が定義されていない場合	name 属性が定義されている場合
変換前	<csdc:oncopy> コピーイベントハンドラの処理※ </csdc:oncopy>	<csdc:oncopy name="samplehandler"> コピーイベントハンドラの処理※ </csdc:oncopy>

変換後	<script type="text/
	javascript">
	function
	hptl_uap_oncopy1(){
	コピーイベントハンドラの処理※
	}
	</script>
		<script type="text/
		javascript">
		function samplehandler()
		{
		コピーイベントハンドラの処理※
		}
		</script>

注※ コピーイベントハンドラの処理の詳細は，このタグライブラリの構文の注を参照してください。

(5) csdc:onpaste

機能

ペーストイベントハンドラを定義します。<csdc:dest> の子要素としてだけ利用できます。

構文

```
<csdc:onpaste [name="function name"]>
    ペーストイベントハンドラの処理※
</csdc:onpaste>
```

注※ ここでは，JavaScript を使用し，次の処理を定義します。

1. JavaScript 関数「hptl_getPortalClipboardData」を使用し，ポータルクリップボードウィンドウから JavaScript オブジェクトを取得します。
2. 転送先 HTML 部品のオブジェクトを取得します。
3. 1. で取得した JavaScript オブジェクトを 2. で取得した転送先 HTML 部品のオブジェクトに格納します。

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
name	String	ハンドラの関数名を定義します。関数名を定義することで，任意の JavaScript コードから呼び出しができるようになります。ただし，ページ内で関数名をユニークにする必要があります。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

HTML 変換規則

<csdc:onpaste> の HTML 変換規則を次に示します。

表 14-14 <csdc:onpaste> の HTML 変換規則

変換前 / 後	name 属性が定義されていない場合	name 属性が定義されている場合
変換前	<csdc:onpaste> ペーストイベントハンドラの処理※ </csdc:onpaste>	<csdc:onpaste name="samplehandler2"> ペーストイベントハンドラの処理※ </csdc:onpaste>

変換後	<script type="text/javascript">
	function hptl_uap_oncopy2()
	{
	ペーストイベントハンドラの処理※
	}
	</script>
		<script type="text/
		javascript">
		function samplehandler2()
		{
		ペーストイベントハンドラの処理※
		}
		</script>

注※ ペーストイベントハンドラの処理の詳細は，このタグライブラリの構文の注を参照してください。

(6) csdc:serializeJ2H

機能

Java オブジェクトをシリアライズします。このタグライブラリでは，次の Java データ型をサポートします。

- JavaBeans
- HashMap クラス

これら以外の Java データ型はサポートしません。

構文

属性を定義する順序は任意です。

```
<csdc:serializeJ2H type="{bean|map}" data="java data object" />
```

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
data	Object	シリアライズ対象のオブジェクトを指定します。 ・JavaBeans の場合：JavaBeans インスタンスを指定 ・HashMap の場合：HashMap インスタンスを指定	必須	○
type	String	Java オブジェクトのデータ型を指定します。 ・JavaBeans の場合："bean" を指定 ・HashMap の場合："map" を指定	必須	○

（凡例）
○：JSP スクリプトレット式で評価されます。

HTML 変換規則

変換前

```
<csdc:serializeJ2H type="bean" data="object" />
```

変換後

data 属性に指定されているインスタンス ”object ” がシリアライズされた文字列

（7） csdc:src

機能

データ転送元のオブジェクトを定義します。

構文

```
<csdc:src datatype="data type">
    <csdc:oncopy> // 子要素
    <csdc:body>   // 子要素
</csdc:src>
```

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価
datatype	String	・この属性には，複数の文字列を指定できます。文字列を複数指定する場合は，文字列をコンマ区切りで記述します。指定する文字列の中にコンマが含まれる場合は，動作しなくなるおそれがあるため注意が必要です。 ・指定できる文字列は，最大 16 文字です。 ・定義フォーマットを次に示します。 「**ネームスペース名 . 通信データオブジェクト名称**」 ・ネームスペース名が Java クラスライブラリ名と同じ場合は，ドメインの文字列を反転させたものを記述します。 ・uCosminexus Portal Framework では，「hptl」で始まる文字列を予約語としているため，開発ユーザは，「hptl」で始まる文字列以外を指定する必要があります。	必須	○

（凡例）
○：JSP スクリプトレット式で評価されます。

HTML 変換規則

<csdc:src> タグ内に記述された内容が表示されます。

（8） csdc:use

機能

クライアントサイドデータ通信機能を使用するための初期定義をします。このタグライブラリは，ポートレット／テンプレート上で <body> タグ内に 1 回だけ定義します。また，フレームで使用する場合には，そのフレームを取り込んでいるウィンドウの <body> タグ内に定義します。

なお，複数定義した場合，HTML 変換は実行されますが，iframe タグの ID が重複するため，クライアントサイドデータ通信機能の動作を保証できません。また，フレーム内で定義した場合，HTML 変換は実行されますが，初期定義として使用されません。

構文

```
<csdc:use/>
```

属性

なし

HTML 変換規則

変換前

```
<csdc:use/>
```

変換後

<PortalResources.properties **ファイル**※**の**
jp.co.hitachi.soft.portal.csdc.datatype.userData **項目が「**true**」の場合 >**

```
<iframe id="hptl_csdc_behavior_frame" src="{uCosminexus
Portal Frameworkのコンテキストルート}/html/userData.html"
width="0" height="0" frameborder="no" scrolling="no"
marginwidth="0" marginheight="0">
</iframe>
<iframe id="hptl_csdc_serialize_temporary_frame" src=”
{uCosminexus Portal Frameworkのコンテキストルート}/html/
blank.html” width="0" height="0" frameborder="no"
scrolling="no" marginwidth="0" marginheight="0">
</iframe>
```

<PortalResources.properties **ファイル**※**の**
jp.co.hitachi.soft.portal.csdc.datatype.userData **項目が「**true**」以外の場合 >**

何も記述されません。

注※　PortalResources.properties ファイルの詳細は，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「PortalResources.properties の詳細」の説明を参照してください。

14.12 クライアントサイドデータ通信 JavaScript

14.12.1 クライアントサイドデータ通信 JavaScript の概要

クライアントサイドデータ通信 JavaScript は，ポートレット開発者がクライアントサイドデータ通信機能を実現するために使用する JavaScript 関数を提供します。ポートレット開発者は，クライアントサイドデータ通信機能を使用するためには JSP ファイルに JavaScript ファイル「csdc.js」への絶対パス参照を次のように記述する必要があります。

```
<script src="/ポータルプロジェクト名称js/csdc.js">
</script>
```

上記の“ / ポータルプロジェクト名称 “ は，「request.getContextPath()」を使用して取得できます。

クライアントサイドデータ通信 JavaScript 関数の一覧を次の表に示します。

表 14-15 クライアントサイドデータ通信 JavaScript 関数の一覧

JavaScript 関数	説明
hptl_deserializeH2JS	<csdc:serializeJ2H> タグライブラリでシリアライズされたデータをデシリアライズし，JavaScript オブジェクトに変換します。
hptl_getPortalClipboardData	• userData 領域またはポータルクリップボードウィンドウ内の JavaScript グローバル変数に設定されている JavaScript 配列オブジェクトへの参照を取得します。 • ペーストイベントハンドラ内には，この関数を定義する必要があります。
hptl_oncopy	• コピーイベントハンドラを実行します。 • コピーイベントハンドラを実行する場合は，この関数を使用する必要があります。
hptl_onpaste	• ペーストイベントハンドラを実行します。 • ペーストイベントハンドラを実行する場合は，この関数を使用する必要があります。
hptl_setPortalClipboardData	• userData 領域またはポータルクリップボードウィンドウに存る JavaScript グローバル変数に，JavaScript オブジェクトを設定します。 • コピーイベントハンドラ内にこの関数を定義する必要があります。

14.12.2 関数の説明

hptl_deserializeH2JS

形式

```
function hptl_deserializeH2JS(type, data)
```

機能

<csdc:serializeJ2H> タグライブラリでシリアライズされたデータをデシリアライズし，JavaScript オブジェクトに変換します。

パラメタ

type - デシリアライズの方式。<csdc:serializeJ2H> タグライブラリの type 属性で指定した文字列を指定してください。
data - デシリアライズ対象のデータ（文字列）

戻り値

- 戻り値 == null でない場合
  JavaScript 連想配列オブジェクト
- 戻り値 == null の場合
  null

hptl_getPortalClipboardData

形式

```
function hptl_getPortalClipboardData(datatype)
```

機能

userData 領域またはポータルクリップボードウィンドウ内の JavaScript グローバル変数に設定されている JavaScript 配列オブジェクトへの参照を取得します。
ペーストイベントハンドラ内には，この関数を定義する必要があります。

パラメタ

datatype - <csdc:dest> タグライブラリの datatype 属性に指定する文字列

戻り値

- 戻り値 == null でない場合
  userData 領域またはポータルクリップボードウィンドウにセットされている JavaScript データ
- 戻り値 == null の場合
  null

hptl_oncopy

形式

```
function hptl_oncopy(datatype, handler, obj)
```

機能

コピーイベントハンドラを実行します。
コピーイベントハンドラを実行する場合は，この関数を使用する必要があります。

パラメタ

datatype - <csdc:src> タグライブラリの datatype 属性に指定してある文字列。

複数の datatype 属性を指定する場合は，コンマ区切りで文字列を指定してください。
handler - コピーイベントハンドラ
obj - イベント対象オブジェクト（this）

戻り値
なし

hptl_onpaste

形式
```
function hptl_onpaste(datatype, handler, obj)
```

機能
ペーストイベントハンドラを実行します。
ペーストイベントハンドラを実行する場合は，この関数を使用する必要があります。

パラメタ
datatype - <csdc:dest> タグライブラリの datatype 属性に指定してある文字列。
複数の datatype を指定する場合は，コンマ区切りで文字列を指定してください。
handler - ペーストイベントハンドラ
obj - イベント対象オブジェクト（this）

注意事項
このバージョンでは，データ転送先で複数の datatype を指定することはサポートされていません。そのため，第一パラメタに複数の datatype を指定することはできますが，動作しません。

戻り値
なし

hptl_setPortalClipboardData

形式
```
function hptl_setPortalClipboardData(datatype, data)
```

機能
userData 領域またはポータルクリップボードウィンドウに存る JavaScript グローバル変数に，JavaScript オブジェクトを設定します。
コピーイベントハンドラ内にこの関数を定義する必要があります。

パラメタ
datatype - <csdc:src> タグライブラリの datatype 属性に指定する文字列
data - デシリアライズされたデータを格納した JavaScript 配列オブジェクト

戻り値
なし

14.13 クライアントサイドデータ通信クラスライブラリ

クライアントサイドデータ通信クラスライブラリは，ドラッグアイコン取得 API だけを提供します。

ドラッグアイコン取得 API の一覧を次の表に示します。

表 14-16 ドラッグアイコン取得 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.csdc.lib.CsdcUtil	クライアントサイドデータ通信機能のドラッグ用アイコン（img 要素）を示す HTML 文字列を取得します。

jp.co.hitachi.soft.portal.api.csdc.lib.CsdcUtil

機能

クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。

この API は，クライアントサイドデータ通信タグライブラリの <csdc:src> の <csdc:body> 内で使用します。

メソッドの一覧

戻り値のデータ型	メソッド	説明
static String	getDragIcon(HttpServletRequest request, String desc, String hclass, String src)	クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。 HTML 内に格納するコードを出力する場合に使用します。ただし，align パラメタは指定できません。
static String	getDragIcon(HttpServletRequest request, String desc, int escape, String hclass, String hid, String src, String type)	クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。 JavaScript 内に記載するためのコードを出力する場合に使用します。ただし，align パラメタは指定できません。

戻り値のデータ型	メソッド	説明
static String	getDragIcon(HttpServletRequest request, String desc, String hclass, String src, String align)	クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。 HTML 内に格納するコードを出力する場合に使用します。align パラメタを指定できます。
static String	getDragIcon(HttpServletRequest request, String desc, int escape, String hclass, String hid, String src, String align, String type)	クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。 JavaScript 内に記載するためのコードを出力する場合に使用します。align パラメタを指定できます。

メソッドの説明

getDragIcon〔HTML 用　align パラメタなし〕

形式

```
public static String getDragIcon(HttpServletRequest request,
                                  String desc, String hclass,
                                  String src)
```

機能

指定したパラメタに従って，クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。

HTML 内に格納するコードを出力する場合に使用します。ただし，align パラメタは指定できません。

パラメタ

request - JSP 表示時のリクエストオブジェクト（null 以外）

desc - <img> タグの alt 属性に設定する文字列を指定してください。null または空文字列を指定した場合，alt 属性は設定されません。

hclass - <img> タグの class 属性に設定する文字列を指定してください。null または空文字列を指定した場合，class 属性は設定されません。

src - <img> タグの src 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。

戻り値

ドラッグ用のアイコンを示す HTML 文字列

例外

PortletException - 入力パラメタが不正な場合，またはこの API の使用個所が不正な場合

getDragIcon〔JavaScript 用　align パラメタなし〕

形式

```
public static String getDragIcon(HttpServletRequest request,
                                  String desc, int escape,
                                 String hclass, String hid,
                                  String src, String type)
```

機能

指定したパラメタに従って，クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。

JavaScript 内に記載するためのコードを出力する場合に使用します。ただし，align パラメタは指定できません。

パラメタ

request - JSP 表示時のリクエストオブジェクト（null 以外）

desc - <img> タグの alt 属性に設定する文字列を指定してください。null または空文字列を指定した場合，alt 属性は設定されません。

escape - JavaScript 内で使用する場合に，区切り文字を変更するために指定します。

- CSDC_DRAGICON_ESCAPE_TYPE_SINGLE_QUOTE（数値は 0 とする）：このメソッドが出力する HTML を JavaScript 上で文字列リテラルとして埋め込む場合で，文字列リテラルを「'」で囲むときに指定します。
- CSDC_DRAGICON_ESCAPE_TYPE_DOUBLE_QUOTE（数値は 1 とする）：このメソッドが出力する HTML を JavaScript 上で文字列リテラルとして埋め込む場合で，文字列リテラルを「"」で囲むときに指定します。
- CSDC_DRAGICON_ESCAPE_TYPE_NONE（数値は -1 とする）：このメソッドが出力する HTML を，HTML 上に直接埋め込むときに指定します。
- 上記以外の値の場合：CSDC_DRAGICON_ESCAPE_TYPE_NONE が指定されたものとして処理します。

hclass - <img> タグの class 属性に設定する文字列を指定してください。null または空文字列を指定した場合，class 属性は設定されません。

hid - <img> タグの id 属性に設定する文字列を指定してください。null または空文字列を指定した場合，id 属性は設定されません。

src - <img> タグの src 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。

type - ポートレットごとの組み込み型を指定します（例：people）。

このパラメタに指定した文字列は，src 属性で指定するアイコンがどのような種別であるかを示すコメントになります。なお，このパラメタに指定した文字列は，クライアントサイドデータ通信クラスライブラリから参照されることがないため，不正な文字列が定義されていてもエラーにはなりません。

戻り値

ドラッグ用のアイコンを示す HTML 文字列

例外

PortletException - 入力パラメタが不正な場合，またはこの API の使用個所が

不正な場合

getDragIcon〔HTML 用　align パラメタあり〕

形式

```
public static String getDragIcon(HttpServletRequest request,
                                  String desc, String hclass,
                                   String src, String align)
```

機能

指定したパラメタに従って，クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。

HTML 内に格納するコードを出力する場合に使用します。align パラメタを指定できます。

パラメタ

request - JSP 表示時のリクエストオブジェクト（null 以外）

desc - <img> タグの alt 属性に設定する文字列を指定してください。null または空文字列を指定した場合，alt 属性は設定されません。

hclass - <img> タグの class 属性に設定する文字列を指定してください。null または空文字列を指定した場合，class 属性は設定されません。

src - <img> タグの src 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。

align - <img> タグの align 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。

戻り値

ドラッグ用のアイコンを示す HTML 文字列

例外

PortletException - 入力パラメタが不正な場合，またはこの API の使用個所が不正な場合

getDragIcon〔JavaScript 用　align パラメタあり〕

形式

```
public static String getDragIcon(HttpServletRequest request,
                                   String desc, int escape,
                                  String hclass, String hid,
                                   String src, String align,
                                   String type)
```

機能

指定したパラメタに従って，クライアントサイドデータ通信機能のドラッグ用アイコンの画像を示す HTML 文字列を取得します。

JavaScript 内に記載するためのコードを出力する場合に使用します。align パラメタを指定できます。

パラメタ

request - JSP 表示時のリクエストオブジェクト（null 以外）

desc - <img> タグの alt 属性に設定する文字列を指定してください。null または空文字列を指定した場合，alt 属性は設定されません。
escape - JavaScript 内で使用する場合に，区切り文字を変更するために指定します。

- CSDC_DRAGICON_ESCAPE_TYPE_SINGLE_QUOTE（数値は 0 とする）：このメソッドが出力する HTML を JavaScript 上で文字列リテラルとして埋め込む場合で，文字列リテラルを「'」で囲むときに指定します。
- CSDC_DRAGICON_ESCAPE_TYPE_DOUBLE_QUOTE（数値は 1 とする）：このメソッドが出力する HTML を JavaScript 上で文字列リテラルとして埋め込む場合で，文字列リテラルを「"」で囲むときに指定します。
- CSDC_DRAGICON_ESCAPE_TYPE_NONE（数値は -1 とする）：このメソッドが出力する HTML を，HTML 上に直接埋め込むときに指定します。
- 上記以外の値の場合：CSDC_DRAGICON_ESCAPE_TYPE_NONE が指定されたものとして処理します。

hclass - <img> タグの class 属性に設定する文字列を指定してください。null または空文字列を指定した場合，class 属性は設定されません。
hid - <img> タグの id 属性に設定する文字列を指定してください。null または空文字列を指定した場合，id 属性は設定されません。
src - <img> タグの src 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。
align - <img> タグの align 属性に設定する文字列を指定してください。null または空文字列を指定した場合，src 属性は設定されません。
type - ポートレットごとの組み込み型を指定します（例：people）。
このパラメタに指定した文字列は，src 属性で指定するアイコンがどのような種別であるかを示すコメントになります。なお，このパラメタに指定した文字列は，クライアントサイドデータ通信クラスライブラリから参照されることがないため，不正な文字列が定義されていてもエラーにはなりません。

戻り値

ドラッグ用のアイコンを示す HTML 文字列

例外

PortletException - 入力パラメタが不正な場合，またはこの API の使用個所が不正な場合

14.14 ダイレクト呼び出し結果取得 API

ダイレクト呼び出し結果取得 API の一覧を次の表に示します。

表 14-17 ダイレクト呼び出し結果取得 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.request.PortalHttpRequest	ダイレクト呼び出し実行時のデータの引き継ぎ結果を取得します。

ダイレクト呼び出し結果取得 API の詳細を説明します。

jp.co.hitachi.soft.portal.api.request.PortalHttpRequest

機能

ダイレクト呼び出し実行時のデータの引き継ぎ結果を取得するクラスです。

使用方法

ダイレクト呼び出し実行時のデータの引き継ぎ結果を取得する場合に使用します。

メソッドの一覧

戻り値のデータ型	メソッド	説明
int	getSuccessionResult(ServletRequest)	データの引き継ぎ結果を取得します。

メソッドの説明

getSuccessionResult

形式

```
public int getSuccessionResult(ServletRequest req)
```

機能

データの引き継ぎ結果を取得します。

パラメタ

req - 日立 API ポートレットに渡された ServletRequest オブジェクト

戻り値

以下の int 型整数値の変数名

RETURN_SUCCESS_DIRECT - POST データの引き継ぎに成功した

RETURN_NOT_DIRECT - ダイレクト呼び出しされていない

RETURN_MAX_ERROR - POST データのサイズが上限を超えている

RETURN_MIME_ERROR - MIME タイプが multipart/form-data; である

RETURN_NOT_HEADER - 引き継ぎデータに HTTP ヘッダがない
RETURN_OTHER_ERROR - その他の例外的な処理によって POST データの引き継ぎに失敗した

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

14.15 ナビゲーションメニューのタグライブラリ

各ポートレットが，ナビゲーションメニューのリンク集機能を使用する際に，リンク集の登録をするリクエスト先 URL と，登録情報を引き渡すインタフェースが必要となります。ここでは，ナビゲーションメニューが提供するリンク集登録用 URL を生成するタグライブラリについて説明します。

提供方法

このタグライブラリは，ナビゲーションメニューの jar ファイル内に同梱して提供されます。

利用手順

このタグライブラリを使用する前に以下の設定が必要です。

1. Web アプリケーションの DD（web.xml）内のタグライブラリに関する要素を追加します。

この要素は，新規にポータルサーバを構築した場合に追加されます。

```
<taglib>
<taglib-uri>http://soft.hitachi.co.jp/portal/api/navigationbar</taglib-uri>
<taglib-location>/portlets/navigationmenu/navimenu.tld</taglib-location>
</taglib>
```

web.xml の格納場所を次に示します。

{PROJECT_HOME} ¥WEB-INF

2. 該当する JSP ファイルに，次に示す taglib ディレクティブの定義を記述します。

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/api/navigationbar"
prefix="navimenu" %>
```

14.15.1 タグライブラリの一覧

ナビゲーションメニューで提供するタグライブラリ一覧を次の表に示します。

表 14-18 ナビゲーションメニューで提供するタグライブラリ一覧

タグ名	説明	属性
navimenu:linkregisturl /	ナビゲーションメニューのリンク集にリンクを登録するための URL を出力します。	label，description，url，target，portletname

14.15.2 タグライブラリの説明

<navimenu:linkregisturl />

機能

ナビゲーションメニューのリンク集にリンクを登録するための URL を出力します。

構文

```
<navimenu:linkregisturl label="labelString" url="urlString"
target={"other"} portletname="name"/>
```

属性

属性	型	説明	必須 / 任意	JSP スクリプトレット式評価	最大値
description	String	リンク集に表示するラベル名のツールチップに表示する説明文を指定します。	必須	○	32 文字
label	String	リンク集に表示するラベル名の文字列を指定します。	必須	○	16 文字
portletname	String	リンク集のリンクに表示するアイコン（hportlet.xml で指定した defaulticon）を取得するための，キーとなる対応ポートレット名を指定します。	必須	○	－
target	String	リンク集に登録された URL の参照先を開く Web ブラウザを指定します。 • other：新しい Web ブラウザのウィンドウに表示します。このバージョンでは，other を必ず指定してください。 指定できる URL は，ポートレットの NEWWINDOW モードまたは IFRAME モードです。MAXIMIZE モードの URL を指定すると，HOME 画面が複数現れてしまい動作の保証外となります。	必須	○	－
url	String	リンク集に登録する URL を指定します。	必須	○	670 バイト

（凡例）
○：JSP スクリプトレット式で評価されます。
－：該当しません。

HTML 変換規則

各属性に指定したリンクの情報をリンク集に登録する URL が出力されます。

- このタグライブラリは，属性で指定したデータをクエリとして，引き渡す URL を出力します。
- このタグライブラリで生成される URL は，登録確認画面を表示します。そのため，HTML の a タグ，または JavaScript の window.open() で，ターゲットに一意な値を代入して参照する必要があります。（例：target="_blank"）

制限および注意事項

- ポートレットユティリティクラスライブラリが使用できる JSP 内でだけ利用できます。ポートレットユティリティクラスライブラリの詳細は,「14.4 ポートレットユティリティクラスライブラリ」を参照してください。
- 属性値の最大値を超えて利用した場合，Internet Explorer では GET リクエストの発行ができなくなります。
- 属性値の最大値を超えて利用したい場合
  label，description，および url の属性値の最大値を超えて利用したい場合は,次の計算式を満たす範囲で拡張して利用してください。

  1100 **バイト＝（**label **文字数＋** description **文字数）×** 9 **＋** url **バイト数**

  < 備考 >

  label および description は，URLEnecoding 対象のフィールドです。ASCII 文字以外は，1 文字当たり 9 バイトの変換文字列となるため 9 を掛けています。

14.16 ユーザロケール情報取得 API

ユーザロケール情報取得 API の一覧を次の表に示します。

表 14-19 ユーザロケール情報取得 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.user.LocaleData	ユーザのロケール情報を取得，設定します。

ユーザロケール情報取得 API の詳細を説明します。

jp.co.hitachi.soft.portal.api.user.LocaleData

機能

カスタマイズ情報に保存されているユーザのロケール情報（言語およびタイムゾーン）を取得，設定するクラスです。

使用方法

ユーザの利用言語に応じたコンテンツを作成する場合に使用します。

注意事項

カスタマイズ情報が取得できない場合に，Web ブラウザの設定値を取得する場合があります。しかし，この API から Web ブラウザの設定を変更することはできません。

メソッドの一覧

戻り値のデータ型	メソッド	説明
static final String	getLangType(HttpServletRequest req)	ユーザが表示可能な言語の中で最も優先度の高い言語の識別子を ISO 言語コードに従った形式で取得します。
public static final Enumeration	getLangTypes(HttpServletRequest req)	ユーザが表示可能な言語のすべての識別子を取得します。
public static final String	getLanguage(HttpServletRequest req)	ユーザが表示可能な言語の中で最も優先度の高い言語の識別子を文字列表現で取得します。
static final String	getTimeZone(HttpServletRequest req)	カスタマイズ情報に設定されているタイムゾーン ID を取得します。
static final void	setLangType(HttpServletRequest req,String langtype)	カスタマイズ情報に言語 ID を設定します。
static final void	setTimeZone(HttpServletRequest req,String timezone)	カスタマイズ情報にタイムゾーン ID を設定します。

戻り値のデータ型	メソッド	説明
public static final Locale	getLocale(HttpServletRequest req)	ユーザが表示可能な言語の中で最も優先度の高い言語に対応するロケールを取得します。
public static final Enumeration	getLocales(HttpServletRequest req)	ユーザが表示可能な言語に対応するロケールをすべて取得します。
public static final Enumeration	getSupportLocales()	uCosminexus Portal Framework がサポートしているすべての言語に対応するロケールを取得します。
public static final Enumeration	getSupportLanguagesOrder()	uCosminexus Portal Framework がサポートしているすべての言語の文字列表現を取得します。

メソッドの説明

getLangType

形式

```
public static final String getLangType(HttpServletRequest req)
```

機能

ユーザが表示可能な言語の中で最も優先度の高い言語の識別子を ISO 言語コードに従った形式で取得します。ユーザがカスタマイズ画面で特定の言語を選択している場合は，その言語の識別子を取得します。ユーザが特定の言語を選択していない場合は，ブラウザで設定されている言語の識別子の中で最も優先度が高いものを取得します。どちらも設定されていない場合は，uCosminexus Portal Framework のデフォルトの言語を取得します。言語の識別子は，ISO 言語コード（RFC1766）に準拠しています。取得できる言語 ID 値を次の表に示します。

ユーザがカスタマイズ情報に設定している言語設定	ユーザが Web ブラウザに設定している言語設定		
	日本語（ja）	英語（en）	その他の言語
日本語	ja	ja	ja
英語	en	en	en
設定なし※1	ja	en	en※2

注※1　ユーザがポータル画面の利用者用レイアウトカスタマイズ画面で言語を設定していない場合，およびウェルカム画面などのログイン前の画面でこのメソッドを発行した場合です。
注※2　プロパティで変更できます。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

ユーザが表示可能な言語の中で最も優先度が高い言語の ISO 言語コード（null

以外）

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getLangTypes

形式

```
public static final Enumeration
getLangTypes(HttpServletRequest req)
```

機能

ユーザが表示可能な言語を次のルールで決定し，該当する言語の配列を取得します。

- ユーザがカスタマイズ画面で特定の言語を選択している場合は，その言語の識別子一つだけが格納された配列を取得します。
- ユーザが特定の言語を選択していない場合は，ブラウザで設定されている言語の識別子が優先度順に格納された配列を取得します。
- どちらも設定が行われていない場合は，uCosminexus Portal Framework のデフォルトの言語の識別子が格納された配列を取得します。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

ユーザが表示可能な言語（String 型，ISO 言語コード）が優先度順に格納された配列（Enumeration 型）

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getLanguage

形式

```
public static final String getLanguage(HttpServletRequest req)
```

機能

ユーザが表示可能な言語の中で最も優先度の高い言語の識別子を文字列表現で取得します。ユーザがカスタマイズ画面で特定の言語を選択している場合は，その言語の識別子を取得します。ユーザが特定の言語を選択していない場合は，ブラウザで設定されている言語の識別子の中で最も優先度が高いものを取得します。どちらも設定されていない場合は，uCosminexus Portal Framework のデフォルトの言語を取得します。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

ユーザが表示可能な言語の中で最も優先度が高い言語の文字列表現

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getTimeZone

形式

```
public static final String getTimeZone(HttpServletRequest req)
```

機能

ユーザがカスタマイズ情報に設定したタイムゾーン ID を取得します。ID は，J2SE1.4 の定めたカスタムタイムゾーン ID に準拠しています。ユーザが何も設定していない場合は，ポータルサーバのデフォルト値が返却されます。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

タイムゾーン ID

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

setLangType

形式

```
public static final void setLangType(HttpServletRequest req,
                                     String langtype)
```

機能

ユーザが使用する言語の ISO 言語コードを設定します。設定された言語コードがサポート外の場合，例外（IllegalArgumentException）が発生します。

パラメタ

req - サーブレットリクエスト（null 以外）

langtype - ISO 言語コード（null 以外）

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

setTimeZone

形式

```
public static final void setTimeZone(HttpServletRequest req,
                                     String timezone)
```

機能

ユーザのタイムゾーン（カスタムタイムゾーン ID）を設定します。無効な ID を指定した場合，例外（IllegalArgumentException）が発生します。

パラメタ

req - サーブレットリクエスト（null 以外）

timezone - カスタムタイムゾーン ID（null 以外）

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getLocale

形式

```
public static final Locale getLocale(HttpServletRequest req)
```

機能

ユーザが表示可能な言語の中で最も優先度の高い言語に対応するロケールを取得します。ユーザがカスタマイズ画面で特定の言語を選択している場合は，その言語に対応するロケールを取得します。ユーザが特定の言語を選択していない場合は，ブラウザで設定されている言語の中で最も優先度が高いものに対応するロケールを取得します。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

ユーザが表示可能な言語の中で最も優先度の高い言語に対応するロケール

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getLocales

形式

```
public static final Enumeration getLocales(HttpServletRequest
req)
```

機能

ユーザが表示可能なロケールを次のルールで決定し，該当するロケールの配列を取得します。

- ユーザがカスタマイズ画面で特定の言語を選択している場合は，その言語に対応するロケール一つだけが格納された配列を取得します。
- ユーザが特定の言語を選択していない場合は，ブラウザで設定されている言語に対応するロケールが優先度順に格納された配列を取得します。
- どちらも設定が行われていない場合は，uCosminexus Portal Framework のデフォルトの言語に対応するロケールが格納された配列を取得します。

パラメタ

req - サーブレットリクエスト（null 以外）

戻り値

ユーザが表示可能なロケール（java.util.Locale 型）が優先度順に格納された配列（Enumeration 型）

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getSupportLocales

形式

```
public static final Enumeration getSupportLocales()
```

機能

uCosminexus Portal Framework がサポートしているすべての言語に対応するロケールを優先度順に取得します。優先度は，uCosminexus Portal Framework の jp.co.hitachi.soft.portal.i18n.supports プロパティに定義された順になります。

パラメタ

なし

戻り値

uCosminexus Portal Framework がサポートしているすべての言語に対応するロケール（java.util.Locale 型）が優先度順に格納された配列

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getSupportLanguagesOrder

形式

```
public static final Enumeration getSupportLanguagesOrder()
```

機能

uCosminexus Portal Framework がサポートしているすべての言語の文字列表現を優先度順に取得します。優先度は，uCosminexus Portal Framework の jp.co.hitachi.soft.portal.i18n.supports プロパティに定義された順になります。

パラメタ

なし

戻り値

uCosminexus Portal Framework がサポートしているすべての言語の文字列表現（String 型）が優先度順に格納された配列

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

14.17 ストリングリソース取得 API

ストリングリソース取得 API の一覧を次の表に示します。

表 14-20 ストリングリソース取得 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.util.StringResource	ストリングリソースを取得します。

ストリングリソース取得 API の詳細を説明します。

jp.co.hitachi.soft.portal.api.util.StringResource

機能

ストリングリソースを取得するクラスです。

使用方法

getStringResource メソッドでインスタンスを取得して使用してください。

メソッドの一覧

戻り値のデータ型	メソッド	説明
Enumeration	getKeys()	ストリングリソースのキーの一覧を取得します。
String	getString(String key)	ストリングリソースから指定したキーに対する値を取得します。
static final StringResource	getStringResource(HttpServletRequest req,String appkey)	ユーザの使用言語に合わせたストリングリソース（StringResource クラス）を取得します。
public static final StringResource	getStringResource(java.util.Locale locale, String appkey)	指定した言語に合わせたストリングリソース（StringResource クラス）を取得します。

メソッドの説明

getKeys

形式

```
public Enumeration getKeys()
```

機能

ストリングリソースファイルからメッセージキーの一覧を取得します。メッセージキーが存在しない場合，要素数 0 の配列を返却します。このメソッドを

呼び出す前に，getStringResource() メソッドでインスタンスを取得してください。

パラメタ

なし

戻り値

ストリングリソース内のキー一覧

例外

なし

getString

形式

```
public String getString(String key)
```

機能

ストリングリソースファイルからメッセージを取得します。このメソッドを呼び出す前に，getStringResource() メソッドでインスタンスを取得してください。ストリングリソースファイルに該当するメッセージキーが存在しない場合，null を戻します。

パラメタ

key - メッセージキー（null 以外）

戻り値

ストリングリソースファイルから取得したメッセージ

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

getStringResource(HttpServletRequest req, String appkey)

形式

```
public static final StringResource
getStringResource(HttpServletRequest req,
                  String appkey)
```

機能

指定したアプリケーションキー，およびユーザの設定言語に対応する StringResource クラスのインスタンスを取得します。該当するストリングリソースファイルが存在しない場合，およびデフォルト言語のストリングリソースファイルを検索しても取得できなかった場合，例外（MissingResourceException）が発生します。

パラメタ

req - サーブレットリクエスト（null 以外）

appkey - アプリケーションキー（null 以外）

戻り値

ストリングリソースクラス

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

MissingResourceException - ストリングリソースファイルが見つからない場合

getStringResource(java.util.Locale locale, String appkey)

形式

```
public static final StringResource
   getStringResource(java.util.Locale locale, String appkey)
```

機能

指定したアプリケーションキー，およびロケールに対応する StringResource クラスのインスタンスを取得します。該当するストリングリソースファイルが存在しない場合，およびデフォルト言語のストリングリソースファイルを検索しても取得できなかった場合，例外（MissingResourceException）が発生します。

パラメタ

locale - ロケール（null 以外）

appkey - アプリケーションキー（null 以外）

戻り値

ストリングリソースクラス

例外

IllegalArgumentException - 入力パラメタが null または不正な場合

MissingResourceException - ストリングリソースファイルが見つからない場合

14.18 ユーザ登録 API

ユーザ登録 API は，ポータル利用者自身が，ユーザ情報を登録するときに使用する API です。

ユーザ登録 API の一覧を次の表に示します。

表 14-21 ユーザ登録 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.user.PortalUserAccountBean	ポータル利用者自身が，ユーザ情報を登録します。

ユーザ登録 API の詳細を説明します。

jp.co.hitachi.soft.portal.api.user.PortalUserAccountBean

機能

ポータル利用者自身が，ユーザ情報を登録します。

Bean の項目

宣言

```
<jsp:useBean id="puab"
class="jp.co.hitachi.soft.portal.api.user.PortalUserAccountBean
"
scope="request"/>
```

対象スコープ

```
request
```

メソッドの一覧

戻り値のデータ型	メソッド	説明
PortalUserAccount	getUserAccount(String userId)	ユーザ情報を取得します。
String[]	searchUser(String key_colmn, String key_value, String target_colmn)	ユーザ情報を検索します。
void	addUserAccount(PortalUserAccount user, String password)	ユーザを新規に登録します。
void	changeUserAccount(PortalUserAccount user)	ユーザ情報を変更します。

戻り値のデータ型	メソッド	説明
void	removeUserAccount(String userId)	ユーザを削除します。
void	changePassword(String userId, String password)	指定したユーザのパスワードを変更します。

メソッドの説明

getUserAccount

形式

```
public PortalUserAccount getUserAccount(String userId)
```

機能

ユーザ ID（userId）に指定したユーザのアカウント情報を取得します。

パラメタ

userId - リポジトリに登録されているユーザ ID

戻り値

リポジトリにアクセスできるユーザアカウント情報

例外

IllegalArgumentException - 入力パラメタが null の場合

PortalException - 次の場合に発生します。

- リポジトリからユーザ情報の取得に失敗した場合
- 対象のユーザ情報がリポジトリに存在しない場合

searchUser

形式

```
public String[] searchUser(String key_colmn, String key_value,
String target_colmn)
```

機能

検索対象の項目名（key_colmn），検索値（key_value），および取得対象の項目名（target_colmn）を指定して，ユーザ情報を検索します。なお，検索結果が 0 件の場合は空の配列が戻り値となります。

パラメタ

key_colmn - 検索対象の項目名※

key_value - 検索値（ワイルドカードは指定できない）

target_colmn - 取得対象の項目名※

注※

ユーザ情報のマッピング項目を指定してください。なお，検索対象の項目名と取得対象の項目名のどちらかには，ユニークキー（ユーザ情報のマッピング項目の接続先（構成情報）のキーカラム（属性））にマッピングされた項目を指定する必要があります。ユニークキーにマッピングされた項目以外を指定した場

合には，取得カラムの指定は無視され，ユニークキーの値が戻り値となります。

戻り値

取得したカラムの値を格納した配列

例外

IllegalArgumentException - 入力パラメタ（key_colmn，target_colmn）が null の場合

PortalException - データリソース参照中に障害が発生した場合

addUserAccount

形式

```
public void addUserAccount(PortalUserAccount user, String password)
```

機能

ユーザアカウント情報（user）およびパスワード（password）に指定したユーザ情報を，マッピング定義で指定したリポジトリに登録します。

パラメタ

user - 追加対象のユーザアカウント情報

password - 追加対象のユーザのパスワード

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null の場合

PortalException - 次の場合に発生します。

- データリソース参照中に障害が発生した場合
- すでにユーザ情報が存在する場合

changeUserAccount

形式

```
public void changeUserAccount(PortalUserAccount user)
```

機能

リポジトリに登録されているユーザアカウント情報を，ユーザアカウント情報（user）に指定した情報に変更します。

パラメタ

user - 変更対象のユーザアカウント情報

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null の場合

PortalException - 次の場合に発生します。

- リポジトリへの情報の登録，またはリポジトリからの情報の取得に失敗した

場合
- 対象のユーザ情報がリポジトリに存在しない場合
- 対象ユーザの所属組織に存在しない組織を指定した場合

removeUserAccount

形式

```
public void removeUserAccount(String userId)
```

機能

ユーザ ID（userId）に指定したユーザのアカウント情報を，リポジトリから削除します。
削除処理は，Cosminexus のリポジトリと uCosminexus Portal Framework のリポジトリの両方に対して実施されます。uCosminexus Portal Framework のリポジトリの削除処理では，指定されたユーザ ID に対応するユーザアカウント情報が登録されていない場合もエラーにはならないで，処理を続行されます。

パラメタ

userId - 削除対象のユーザ ID

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null の場合
PortalException - 次の場合に発生します。
- リポジトリからの情報の削除に失敗した場合
- 対象のユーザ情報が，すべてのリポジトリ（Cosminexus のリポジトリおよび uCosminexus Portal Framework のリポジトリ）に存在しない場合

changePassword

形式

```
public void changePassword(String userId, String password)
```

機能

ユーザ ID（userId）に指定したユーザのパスワードを，パスワード（password）に指定した文字列に変更します。

パラメタ

userId - パスワードを変更するユーザのユーザ ID
password - 変更後のパスワード

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが null または対象のユーザ情報がリポジトリに存在しない場合
PortalException - リポジトリの情報の更新に失敗した場合

14.19 ユーザアカウント情報取得 API

ユーザアカウント情報取得 API は，ユーザのアカウント情報を取得，設定する API です。

ユーザアカウント情報取得 API の一覧を次の表に示します。

表 14-22 ユーザアカウント情報取得 API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.om.security.PortalUserAccount	ポータルのユーザアカウント情報を取得，設定します。

ユーザアカウント情報取得 API の詳細を説明します。

jp.co.hitachi.soft.portal.om.security.PortalUserAccount

機能

ポータルのユーザアカウント情報を取得，設定するクラスです。

メソッドの一覧

戻り値のデータ型	メソッド	説明
String	getUserName()	ユーザ ID を取得します。
String	getDispName()	ユーザ名を取得します。
String[]	getTitle()	役職名を取得します。
String[]	getOU()	所属組織 ID を取得します。
String[]	getParameter(String key)	ユーザに対する特定のマッピング項目を取得します。
void	setDispName(String name)	ユーザ名を設定します。
void	setTitle(String[] titles)	役職名を設定します。
void	setOU(String[] ou)	所属組織 ID を設定します。
void	setParameter(String key, String[] value)	ユーザに対する特定のマッピング項目を設定します。

メソッドの説明

getUserName

形式

String `getUserName()`

機能

ユーザアカウント情報からユーザ ID を取得します。
このオブジェクトが持っているキャッシュからユーザ ID を取得します。

パラメタ
なし

戻り値
ユーザ ID

例外
なし

getDispName

形式
```
String getDispName()
```

機能
ユーザアカウント情報からユーザ名を取得します。
このオブジェクトが持っているキャッシュからユーザ名を取得します。キャッシュにユーザ名がない場合は，ユーザ ID をキーにしてリポジトリから取得します。なお，ユーザ名が設定されていない場合は，ユーザ ID が戻り値となります。

パラメタ
なし

戻り値
ユーザ名

例外
なし

getTitle

形式
```
String[] getTitle()
```

機能
ユーザアカウント情報からユーザの役職名を取得します。
このオブジェクトが持っているキャッシュから役職名を取得します。キャッシュに役職名がない場合は，ユーザ ID をキーにしてリポジトリから取得します。役職名が設定されていない場合，およびマッピング情報の設定で役職が設定されていない場合は，空のリストが戻り値となります。

パラメタ
なし

戻り値
役職名のリスト

例外

なし

getOU

形式

```
String[] getOU()
```

機能

ユーザアカウント情報から所属組織 ID（ユーザが所属している組織の ID）を取得します。
このオブジェクトが持っているキャッシュから所属組織 ID を取得します。
キャッシュに所属組織 ID がない場合は，ユーザ ID をキーにしてリポジトリから取得します。所属組織 ID は，ユーザが所属している最下位の組織の ID だけを取得し，上位組織の ID は取得しません。ユーザが組織に所属していない場合，およびマッピング情報の設定で組織情報が設定されていない場合は，空のリストが戻り値となります。

パラメタ

なし

戻り値

所属組織 ID

例外

なし

getParameter

形式

```
String[] getParameter(String key)
```

機能

ユーザアカウント情報から特定のマッピング項目を取得します。
値が一つだけの場合，長さ 1 の String 配列の最初（0 番目）に格納されます。
このオブジェクトが持っているキャッシュからマッピング項目を取得します。
キャッシュにマッピング項目がない場合は，ユーザ ID をキーにしてリポジトリから取得します。リポジトリにユーザ情報または対象データがない場合は，空のリストが戻り値となります。
なお，このオブジェクトの戻り値は，PortalUserAccount オブジェクトのメンバそのものでなく，オブジェクトのコピーとなります。

パラメタ

key - ユーザのマッピング情報のキー項目名

戻り値

キー項目名にマッピングされている値を格納したリスト

例外

IllegalArgumentException - 入力パラメタが次の値の場合に発生します。

- null

- LOGIN_NAME
- NAME
- ParsonalizeKey
- TITLE
- DEPARTMENT

setDispName

形式

```
void setDispName(String name)
```

機能

ユーザ名を設定します。

パラメタ

name - ユーザ名

戻り値

なし

例外

なし

setTitle

形式

```
void setTitle(String[] titles)
```

機能

役職名を設定します。

パラメタ

titles - 役職名のリスト

戻り値

なし

例外

なし

setOU

形式

```
void setOU(String[] ou)
```

機能

所属組織 ID（ユーザが所属している組織の ID）を設定します。

パラメタ

ou - 所属組織 ID のリスト

戻り値

なし

例外

なし

setParameter

形式

```
void setParameter(String key, String[] value)
```

機能

ユーザアカウント情報に特定のマッピング項目を設定します。
値が一つだけの場合，長さ 1 の String 配列の最初（0 番目）に格納します。
このオブジェクトが持っているキャッシュだけにマッピング項目を設定し，リポジトリには設定しません。

パラメタ

key - 値を格納するキー項目名

value - キー項目名にマッピングする値のリスト

戻り値

なし

例外

IllegalArgumentException - 入力パラメタが次の値の場合に発生します。

- null
- LOGIN_NAME
- NAME
- ParsonalizeKey
- TITLE
- DEPARTMENT
- リポジトリ情報ファイルに未定義のキー項目名

14.20 カスタムウィンドウステート API

カスタムウィンドウステート API は，標準 API ポートレットでカスタムウィンドウステートを指定する場合に使用する API です。

カスタムウィンドウステート API の一覧を次の表に示します。

表 14-23 カスタムウィンドウステート API の一覧

クラス名	説明
jp.co.hitachi.soft.portal.api.UCPFWindowState	javax.portlet.WindowState クラスにカスタムウィンドウステートを拡張したクラス。

カスタムウィンドウステート API の詳細を説明します。

jp.co.hitachi.soft.portal.api.UCPFWindowState

機能

カスタムウィンドウステートを定義するクラスです。

フィールドの説明

フィールド	説明
javax.portlet.WindowState NEW_WINDOW	NEWWINDOW ステート用の WinwowState の定義
javax.portlet.WindowState IFRAME	IFRAME ステート用の WindowState の定義

14.21 拡張 Struts タグライブラリ

拡張 Struts タグライブラリは Struts タグライブラリにポートレットとして動作させるための属性を追加しています。ここでは，追加される属性を説明します。拡張 Struts タグライブラリの一覧を次の表に示します。

表 14-24 拡張 Struts タグライブラリの一覧

タグ名	説明	JSP スクリプトレット式で評価される属性
html:form	ウィンドウステートおよびポートレットモードの指定を追加します。	state，mode
html:link	アクションモードおよびレンダーモードの指定を追加します。 リソース URL の指定を追加します。 ウィンドウステートおよびポートレットモードの指定を追加します。	actionURL，renderURL，resouceURL，state，mode
html:rewrite	アクションモードおよびレンダーモードの指定を追加します。 リソース URL の指定を追加します。 ウィンドウステートおよびポートレットモードの指定を追加します。	actionURL，renderURL，resouceURL，state，mode

html:form

機能

FORM タグを生成します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/bridges/struts/
tags-portlet-html" prefix="html" %>
<html:form action="/html-link-submit" state="maximized" >
フォームデータ
</html:form>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
state	標準 API ポートレットのウィンドウステートを指定する場合に次の値を設定します。 NORMAL ステート：normal MAXIMIZED ステート：maximized MINIMIZED ステート：minimized NEWWINDOW ステート：newwindow IFRAME ステート：ifarme 省略した場合は現在のステートが引き継がれます。	任意	○
mode	標準 API ポートレットのポートレットモードを指定する場合に次の値を設定します。 VIEW モード：view HELP モード：help EDIT モード：edit 省略した場合は現在のポートレットモードが引き継がれます。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

html:link

機能

A タグを生成します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/bridges/struts/
tags-portlet-html" prefix="html" %>
<html:link action="/html-link-submit" state="maximized" >
アンカー文字列
</html: link>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
actionURL	アクション URL を生成する場合に true を設定します。 指定しない場合は属性を省略するか false を設定します。 renderURL 属性，resourceURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
renderURL	レンダー URL を生成する場合に true を設定します。 指定しない場合は属性を省略するか false を設定します。 actionURL 属性，resourceURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
resouceURL	リソース（画像データなど）用の URL を生成する場合に true に設定します。 指定しない場合は属性を省略するか false を設定します。 actionURL 属性，renderURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
state	標準 API ポートレットのウィンドウステートを指定する場合に次の値を設定します。 NORMAL ステート：normal MAXIMIZED ステート：maximized MINIMIZED ステート：minimized NEWWINDOW ステート：newwindowI FRAME ステート：ifarme 省略した場合は現在のステートが引き継がれます。	任意	○
mode	標準 API ポートレットのポートレットモードを指定する場合に次の値を設定します。 VIEW モード：view HELP モード：help EDIT モード：edit 省略した場合は現在のポートレットモードが引き継がれます。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

html:rewrite

機能

URL を生成します。

構文

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/bridges/struts/
tags-portlet-html" prefix="html" %>
<html:rewrite href="/html-link-submit" state="maximized"/>
```

対応する言語

HTML

属性

属性	説明	必須 / 任意	JSP スクリプトレット式評価
actionURL	アクション URL を生成する場合に true を設定します。 指定しない場合は属性を省略するか false を設定します。 renderURL 属性，resourceURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
renderURL	レンダー URL を生成する場合に true を設定します。 指定しない場合は属性を省略するか false を設定します。 actionURL 属性，resourceURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
resouceURL	リソース（画像データなど）用の URL を生成する場合に true に設定します。 指定しない場合は属性を省略するか false を設定します。 actionURL 属性，renderURL 属性と同時に指定した場合は後に設定した属性が有効となります。	任意	○
state	標準 API ポートレットのウィンドウステートを指定する場合に次の値を設定します。 NORMAL ステート：normal MAXIMIZED ステート：maximized MINIMIZED ステート：minimized NEWWINDOW ステート：newwindow IFRAME ステート：ifarme 省略した場合は現在のステートが引き継がれます。	任意	○

属性	説明	必須 / 任意	JSP スクリプトレット式評価
mode	標準 API ポートレットのポートレットモードを指定する場合に次の値を設定します。 VIEW モード：view HELP モード：help EDIT モード：edit 省略した場合は現在のポートレットモードが引き継がれます。	任意	○

（凡例）

○：JSP スクリプトレット式で評価されます。

付録

付録 A　ポートレットのサンプル

uCosminexus Portal Framework では，JSP やサーブレットなどの Java の技術や HTML などの Web の技術を用いて独自のポートレット（日立 API ポートレット）を作成できます。ここでは，uCosminexus Portal Framework で作成できるポートレットの開発例について説明します。

なお，このマニュアルに記載されているサンプルは，ポートレットの作成例を示すために作成されたものです。実際の業務での利用に際してはセキュリティや性能を考慮した処理の作り込みを行ってください。

付録 A.1　Hello World ポートレット

日立 API ポートレットのサンプルとして，［Hello World］ポートレットの例を次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<H1>Hello World</H1>
```

ポータル画面で表示される［Hello World］ポートレットを次の図に示します。

図 A-1　［Hello World］ポートレット

付録 A.2　ハイパーリンクの作成

ポータル画面内で画面遷移する日立 API ポートレットを作成するには，ポートレットユティリティタグライブラリまたはポートレットユティリティクラスライブラリを使用します。ポートレットユティリティタグライブラリの詳細は，「14.2　ポートレットユティリティタグライブラリ」を，ポートレットユティリティクラスライブラリの詳細は，「14.4　ポートレットユティリティクラスライブラリ」を参照してください。

ポートレットユティリティタグライブラリの使用例として，検索ポートレットのサンプルを次に示します。このサンプルでは，トップページに検索キー入力画面を表示して，検索結果を出力します。

{PROJECT_HOME}/portlets/search/index.jsp

```
<%@ page contentType="text/html;  charset=Shift_JIS" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
```

```
prefix="uu" %>

検索キーワードを入力してください。
<uu:form action ="search.jsp" accept_charset="Shift_JIS">
   <label><input type="text"
name="hptl_user_search_key"><label><br>
   <input type="submit" value="検索">
</uu:form>
```

{PROJECT_HOME}/portlets/search/search.jsp

```
<%@ page contentType="text/html;  charset=Shift_JIS" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>

<% String key=new
String(request.getParameter("hptl_user_search_key").getBytes("8859
_1"), "Windows-31J");
検索処理;
%>
<p>
キーワード：<%= key %><br>
<!--検索結果-->
...
</p>
<uu:a href="index.jsp">検索画面に戻る</uu:a>
```

付録 A.3　アトリビュート操作

uCosminexus Portal Framework およびポートレットは同一のサーブレット空間で動作します。そのため，HttpSession および ServletContext に設定するアトリビュート名は，すべてのポートレットで共有します。

ポートレット間で異なるアトリビュート名を指定するには，ポートレットユティリティクラスライブラリの PortletUtils#getNamespace を使用してアトリビュート名を修飾する必要があります。PortletUtils クラスの詳細は「14.4　ポートレットユティリティクラスライブラリ」を参照してください。また，アトリビュート名のプレフィックスとして "jp.co.hitachi.soft.portal"，"turbine"，"jetspeed" は使用できません。

アトリビュート設定と取得方法を次の表に示します。

表 A-1　アトリビュート設定と取得方法

アトリビュート格納先	スコープ	ポートレット開発者の対応
PageContext	page	JSP ページ単位なので，ポートレットとしての対応は必要ありません。
ServletRequest	request	ポートレットで設定されたアトリビュートはポートレットを戻したあとに削除されます。ポートレット内でユニークなネームスペースが保証されるのでポートレットとしての対応は必要ありません。

アトリビュート格納先	スコープ	ポートレット開発者の対応
HttpSession	session	PortletUtils#getNamespace でユニークなネームスペースを取得してアトリビュート名を修飾します。
ServletContext	application	PortletUtils#getNamespace でユニークなネームスペースを取得してアトリビュート名を修飾します。

JSP の jsp:useBean アクションでスコープ属性が session および application の Bean は，それぞれ HttpSession および ServletContext にアトリビュートとして格納されます。そのため，該当する jsp:useBean アクションは，ネームスペースを設定するようにスクリプトレットに置き換える必要があります。

アトリビュート操作の例を次に示します。

アトリビュート設定

```
String msg = "Hello";
String namespace = PortletUtils.getNamespace(request, response);
session.setAttribute(namespace+"アトリビュート名", msg);
```

アトリビュート取得

```
String namespace = PortletUtils.getNamespace(request, response);
String msg = (String)session.getAttribute(namespace+"アトリビュート名");
```

付録 A.4　ログインログアウトの処理

ログインおよびログアウト処理のサンプルを示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletUtils" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%!
  class SessionListener implements HttpSessionBindingListener {
    public void valueBound(HttpSessionBindingEvent e) {
      // ログイン時の処理を記述。
    }
    public void valueUnbound(HttpSessionBindingEvent e) {
      // ログアウト時の処理を記述。
    }
  }
%>
<%
  String ns = PortletUtils.getNamespace(request, response);
  SessionListener sl =
(SessionListener)session.getAttribute(ns+"sl");

  if (sl == null) { // 初回アクセス時にセッションリスナを登録
```

```
    session.setAttribute(ns+"sl", new SessionListener());
  }
%>
<jsp:useBean id="pInfo" scope="page"
class="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"/>
<%
  pInfo.initBean(request);
  PortletInfoBean.Mode mode = pInfo.getMode();
  if (mode == PortletInfoBean.Mode.DEFAULT) {
    //サマリ画面
  } else if (mode == PortletInfoBean.Mode.MAXIMIZE) {
    // 最大化画面
  } else {
    // エラー画面
  }
%>
```

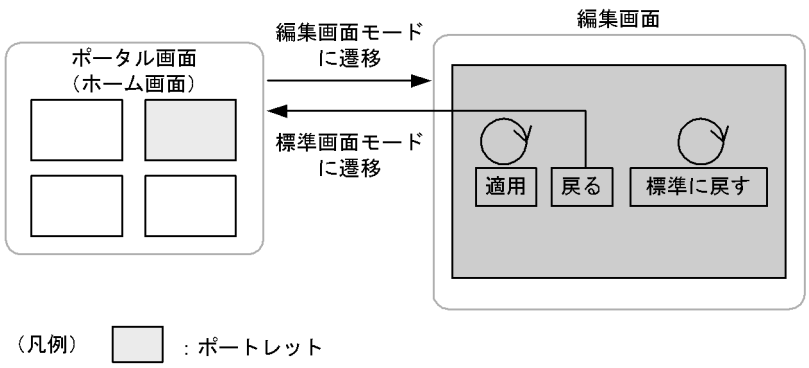
付録 A.5　ユーザごとにパーソナライズするポートレット

ユーザごとにパーソナライズするポートレットを開発するには，ユーザ情報取得 Bean を使用します。ユーザ情報取得 Bean の詳細は，「14.6　ユーザ情報取得 Bean」を参照してください。ユーザごとにパーソナライズするには，リポジトリの設定が必要です。必要な設定については，マニュアル「uCosminexus Portal Framework　システム管理者ガイド」の「ユーザ管理情報の検討」の説明を参照してください。

パーソナライズ情報は，リポジトリに保管されます。そのため，パーソナライズ情報はシリアライズできるオブジェクトに限られます。

パーソナライズするポートレットでは，編集画面を使用します。編集画面の遷移を次の図に示します。

図 A-2　編集画面の遷移



ポータル全体で編集画面のインタフェースを統一するため，ポートレットの編集画面では，次の表に示すボタンおよび動作を実装します。

表 A-2　ポートレットの編集画面でのボタンおよび動作

ボタン	必須	動作
適用	○	パーソナライズ情報を更新して，再度編集画面を表示します。
戻る	○	パーソナライズ情報を更新しないでパーソナライズを終了します。終了後は，ポートレットのサマリ画面を表示します。
標準に戻す	×	システム管理者が設定した内容またはポートレットの初期設定に戻して，再度編集画面を表示します。

（凡例）
○：必須
×：任意

パーソナライズするポートレットのサンプルを次に示します。

このサンプルには，メイン（main.jsp），サマリ（summary.jsp），コントローラ（controller.jsp），およびエディタ（edit.jsp）のファイルがあります。メインおよびサマリは，それぞれメイン画面用およびサマリ画面用のファイルです。コントローラは，ログインチェック，ロジックの呼び出し，および画面モードに応じて View を呼び出します。エディタは，パーソナライズの結果を保存したり，編集画面を作成したりします。

（1）コントローラ（controller.jsp）

{PROJECT_HOME}¥portlets¥customize¥controller.jsp

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%@ page
import="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean" %>
<%
  String uri = "error.jsp";
  try {
    PortletInfoBean pInfo = new PortletInfoBean();
    pInfo.initBean(request);

    PortalUserInfoBean userInfo = new PortalUserInfoBean();
    userInfo.setRequest(request);

    // ログイン済みかチェック
    if (userInfo.checkLoggedIn()) {
      PortletInfoBean.Mode mode = pInfo.getMode();

      if (mode == PortletInfoBean.Mode.DEFAULT) {
        // ここにサマリ画面用業務ロジック呼び出しを追加する

        uri = "summary.jsp";
      } else if (mode == PortletInfoBean.Mode.MAXIMIZE) {
        // ここに最大化画面用の業務ロジック呼び出し追加する

        uri = "main.jsp";
      } else if (mode == PortletInfoBean.Mode.EDIT) {
        uri = "edit.jsp";
```

```
      }
    }
  } catch (Exception ioe) {
    // エラーメッセージ出力
    uri = "error.jsp";
  }
%>
<!-- includeを用いたビューの呼び出し -->
<jsp:include page="<%= uri %>" flush="true"/>
```

(2) エディタ（edit.jsp）

{PROJECT_HOME}¥portlets¥customize¥edit.jsp

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/standard"
prefix="portal" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%
  String def1 = "default1";  String def2 = "defautl2";
  String prop1, prop2;  // プロパティ
  PortalUserInfoBean userInfo = new PortalUserInfoBean();
  userInfo.setRequest(request);

  // 編集情報保存処理
  String editType = request.getParameter("edittype");
  // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
  editType = new
String(editType.getBytes("iso-8859-1"),"Windows-31J");
  if ("apply".equals(editType)) {
    // 適用ボタンの処理(POSTデータを参照し変更を適用)
    prop1 = request.getParameter("prop1");
    prop2 = request.getParameter("prop2");
    userInfo.setCustomizeInfo("prop1", prop1);
userInfo.setCustomizeInfo("prop2", prop2);
  } else if ("restore".equals(editType)) {
    // デフォルトボタン(初期値に変更)
    prop1 = def1;                                    prop2 = def2;
    userInfo.setCustomizeInfo("prop1", prop1);
userInfo.setCustomizeInfo("prop2", prop2);
  } else {
    prop1 = (String)userInfo.getCustomizeInfo("prop1");
    prop2 = (String)userInfo.getCustomizeInfo("prop2");
    if (prop1 == null) { prop1 = def1; }
    if (prop2 == null) { prop2 = def2; }
  }
%>
<script type="text/JavaScript">
<!--
  function returnHome() {
    document.location = "<portal:uriLookup type="Home" />";
  }
  function restore() {
    document.editForm.edittype.value = "restore";
    document.editForm.mysubmit.click();
  }
-->
</script>
<!-- コントローラ経由でPOSTする -->
```

```
<uu:form name="editForm" action="controller.jsp" method="POST">
プロパティ 1 <input type="text" name="prop1" value="<%= prop1 %>">
<br>
プロパティ 2 <input type="text" name="prop2" value="<%= prop2 %>">
<br>
<p>
<input type="hidden" name="edittype" value="apply">
<input type="submit" value="適用" name="mysubmit">
<input type="button" value="戻る" onclick="returnHome()">
<input type="button" value="標準に戻す" onclick="restore()">
</p>
</uu:form>
```

付録 A.6　キャッシュを使用したポートレット

ポータルのスループットやレスポンスタイムの向上には，キャッシュの使用が有効です。

ポートレットでキャッシュを使用するには，タグライブラリを使用します。ユーザのセッション単位で，一定期間ごとにキャッシングできます。タグライブラリの詳細は，「14.2　ポートレットユティリティタグライブラリ」を参照してください。

次にキャッシュを使用したポートレットを示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>

<%
  String nameSpace = PortletUtils.getNamespace(request, response);
  String cacheKey = nameSpace + "key1";
%>
<ut:cache name=<%= cacheKey %> time="10" scope="session">
 <%
   // キャッシングするコンテンツ
 %>
</ut:cache>
```

付録 A.7　ログを出力するポートレット

ログ出力 Bean を使用して，ログを出力するポートレットを開発できます。ポートレットログの出力レベルによって，出力されるログの種類が異なります。出力レベルはデフォルトでは 0 が設定されています。ログ出力 Bean の詳細は，「14.7　ログ出力 Bean」を参照してください。

ポートレットログの出力レベルと出力されるログの内容を次の表に示します。

表 A-3　ポートレットログの出力レベルと出力されるログの内容

出力レベル	出力されるログ	内容
0	error，printStackTrace	通常運用でのトレースレベルです。

出力レベル	出力されるログ	内容
10	warn	監視対象を必要とするトレースレベルです。プログラムの実行性能に影響を与えない範囲で出力します。
20	note	障害調査時に設定するトレースレベルです。プログラムの動作シーケンスを把握できる内容を出力します。
30	debug	障害調査時に設定するトレースレベルです。プログラムの動作が完全に把握できる内容を出力します。

ログ出力 Bean のメソッドは，次の方針に従って使用します。方針に従わない場合，必要なログが得られない，または大量のログが出力されるなどの問題が起きるので注意してください。

error **および** printStackTrace

ポートレットの起動・終了のメッセージ，および業務に支障があるエラーメッセージを出力します。トランジェントオブジェクトの起動や終了のメッセージはこのレベルでは出力しません。

warn

主要なリクエストをトレースし，システムの大まかな動きが把握できるレベルのトレースを出力します。発生頻度の低い障害の再発監視や本番前の現地テストへの適用を想定しています。トレースの出力量をシステムの性能に影響を与えない範囲に抑える必要があります。

note

再現性のある障害に対し，障害の部位を明確に分類するために使用します。ほかのプロセスとの通信など，ほかのプログラムとの関連を把握するのに十分な情報，およびプログラムの大体の動作を把握できるトレース情報を出力します。

debug

障害の個所を特定するために使用します。主要なメンバ関数の開始や終了が取得およびトレースできる内容を出力する必要があります。

ログを出力するポートレットの例を次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%@ page
import="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean" %>
<%@ page import="jp.co.hitachi.soft.portal.api.log.PortletLog" %>
<%
  String uri = "error.jsp";
  PortletLog logger = new PortletLog();

  try {
    logger.setApplication("logging");

    PortletInfoBean pInfo = new PortletInfoBean();
```

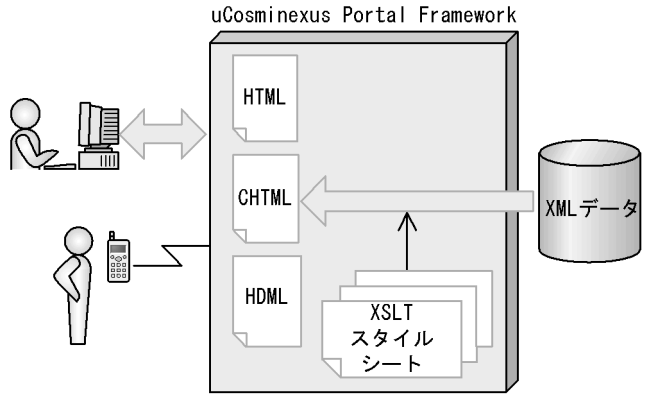
```
    pInfo.initBean(request);

    PortalUserInfoBean userInfo = new PortalUserInfoBean();
    userInfo.setRequest(request);

    // ログイン済みかチェック
    if (userInfo.checkLoggedIn()) {
      PortletInfoBean.Mode mode = pInfo.getMode();

      if (mode == PortletInfoBean.Mode.DEFAULT) {
        logger.note("CALL logic1");
        // ここにサマリ画面用業務ロジック呼び出しを追加する
        logger.note("RET logic1");

        uri = "summary.jsp";
      } else if (mode == PortletInfoBean.Mode.MAXIMIZE) {
        logger.note("CALL logic2");

        // ここに最大化画面用の業務ロジック呼び出し追加する
        logger.note("RET logic2");
        uri = "main.jsp";
      }
    }
  } catch (Exception e) {
    // エラーメッセージ出力
    logger.error("Error occured", e);
    uri = "error.jsp";
  }
%>
<!-- includeを用いたビューの呼び出し -->
<jsp:include page="<%= uri %>" flush="true"/>
```

付録 A.8　XSLT を用いたポートレット

XSLT スタイルシートを介して，XML データをポータルに取り込めます。

XML データの取り込みを次の図に示します。

図 A-3　XML データの取り込み



アクセスするクライアントに応じて，XML データを変換します。また，UserAgent 情報

取得 Bean を使用すると，アクセスしてきたクライアントに応じて，XSLT スタイルシートを変更できます。UserAgent 情報取得 Bean の詳細は，「14.8　UserAgent 情報取得 Bean」を参照してください。

XSLT スタイルシートは，次のディレクトリに格納します。

{PROJECT_HOME}¥portlets¥ ポートレット名 ¥PORTLET-INF¥xslt

日立 API ポートレットでは，<ut:convert> タグライブラリ内に，XML データを記述します。また，<ut:convert> の xslt 属性には，XSLT スタイルシート名を指定します。タグライブラリの詳細は，「14.2　ポートレットユティリティタグライブラリ」を参照してください。

XSLT を用いたポートレットのサンプルを次に示します。このサンプルでは，ユーザに割り当てられた仕事一覧を表示します。データベースなどから，仕事一覧を XML データとして取得します。XSLT スタイルシートを使用して，クライアントが PC の場合はテーブルで，携帯電話（ｉ モード，および EZweb）の場合はリストで表示します。また，XML データの取得および <ut:convert> タグライブラリでは，<ut:cache> タグライブラリを使用して，60 分ごとにデータを更新します。

（1）日立 API ポートレットの例

日立 API ポートレットで XML データを取り込む例を次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletUtils" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%@ page
import="jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean" %>
<jsp:useBean id="useragentInfo" scope="session"

class="jp.co.hitachi.soft.portal.portlet.beans.UserAgentInfoBean"
/>
<%@ taglib uri="http://soft.hitachi.co.jp/portal/utils" prefix="ut"
%>
<%
  String ns = PortletUtils.getNamespace(request, response);
  String xslt;
  try {
    PortalUserInfoBean userInfo = new PortalUserInfoBean();
    userInfo.setRequest(request);

    useragentInfo.initBean(request);
    if (useragentInfo.getUserAgentGroup().equals("PCBrowser")) {
      xslt = "table.xsl";  // PCはtable形式で出力
    } else {
      xslt = "list.xsl";  // PC以外はlist形式で出力
    }

    // ログイン済みかチェック
    if (userInfo.checkLoggedIn()) {
      PortletInfoBean pInfo = new PortletInfoBean();
```

```
      pInfo.initBean(request);
      PortletInfoBean.Mode mode = pInfo.getMode();
      if (mode == PortletInfoBean.Mode.DEFAULT ||
          mode == PortletInfoBean.Mode.MAXIMIZE) {
        String cacheKey = ns+"key";
%>
<ut:cache name="<%= cacheKey %>" time="60" scope="session">
  <ut:convert xslt="<%= xslt %>">
    <!-- XMLデータ(ログインユーザの仕事一覧)読み込み処理 -->
  </ut:convert>
</ut:cache>
<%
      } else { throw new Exception("未サポートの画面モード"); }
    } else { throw new Exception("ログインが必要です"); }
  } catch (Exception e) {
      // エラー出力
  }
%>
```

(2) XML データ

XML データの例として，ユーザに割り当てられた仕事一覧を示します。

```
<task-list>
  <item>
    <title>提案書作成</title>
    <deadline>2002/04/01</deadline>
    <priority>高</priority>
    <detail>http://mytask/task.cgi?id=11111</detail>
  </item>

  <item>
    <title>営業方針検討資料作成</title>
    <deadline>2002/04/11</deadline>
    <priority>標準</priority>
    <detail>http://mytask/task.cgi?id=12111</detail>
  </item>
</task-list>
```

(3) PC 用 XSLT スタイルシート

PC 用に，テーブルに変換する XSLT スタイルシートの例を次に示します。

{PROJECT_HOME}¥portlets¥weather¥xslt¥table.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
  <xsl:output method="html" encoding="Shift_JIS"/>

  <xsl:template match="//task-list">
    <table border="1">
    <tr><th>タイトル</th><th>期限</th><th>優先度</th></tr>
    <xsl:for-each select="item">
      <tr>
     <td><a href="{detail}"><xsl:value-of select="title"/></a></td>
      <td><xsl:value-of select="deadline"/></td>
      <td><xsl:value-of select="priority"/></td>
```

```
      </tr>
    </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

（4）i モード用 XSLT スタイルシート

i モード用に，リストに変換する XSLT スタイルシートの例を次に示します。

{PROJECT_HOME}¥portlets¥weather¥xslt¥list.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
  <xsl:output method="html" encoding="Shift_JIS"/>

  <xsl:template match="//task-list">
    <ol>
    <xsl:for-each select="item">
      <li>
        <xsl:variable name="number"><xsl:number/></xsl:variable>
        <a href="{detail}" accesskey="{$number}">
          <xsl:value-of select="title"/></a>
        (<xsl:value-of select="deadline"/>)
      </li>
    </xsl:for-each>
    </ol>
  </xsl:template>
</xsl:stylesheet>
```

付録 A.9　統合ユーザ管理フレームワークとの連携

日立 API ポートレット内で統合ユーザ管理フレームワークの JSP タグライブラリを使用できます。統合ユーザ管理フレームワークの JSP タグライブラリについては，マニュアル「Cosminexus　アプリケーションサーバ V8　リファレンス　API 編」，またはマニュアル「Cosminexus V9　アプリケーションサーバ　リファレンス　API 編」の「統合ユーザ管理フレームワークで使用するタグライブラリ」の説明を参照してください。

ユーザ属性取得のサンプルコードを次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ taglib uri="http://cosminexus.com/admin/auth/uatags"
prefix="ua" %>

<ua:attributeEntries id="ae">
  <ua:attributeEntry attrName="cn" alias="fullname" />
</ua:attributeEntries>

<!-- 画面表示 -->
<ua:login id="lc" entry="RealmA" attrEntName="ae" excepId="ex" />
こんにちは<ua:getAttribute name="lc" attrName="fullname" />
>(<ua:getPrincipalName name="lc" />)さん。
```

```
<ua:logout name="lc" />

<!-- 例外の処理 -->
<ua:exception name="ex" type="java.lang.Exception">
Exception例外が発生しました:<br />
<%= ex.toString() %><hr />
</ua:exception>
```

付録 A.10　シングルサインオンのためのポートレット

この例では，統合ユーザ管理フレームワークの提供の機能を利用して，メールサーバにシングルサインオンするポートレットを示します。

(1) カスタムログインモジュールの作成

まず，メールサーバとのシングルサインオンを実現するために，メールサーバと認証するカスタムログインモジュールを作成します。カスタムログインモジュールでは，sharedState オブジェクトから，ユーザ ID とパスワードを取得して認証します。次の二つのファイルを作成します。

ファイル 1

```
package mail;

import java.security.Principal;
import java.io.Serializable;

public class MailPrincipal implements Principal, Serializable
{
  private String name;

  public MailPrincipal(String name) {
    if (name == null) throw new NullPointerException();
    this.name = name;
  }
  public String getName() { return name; }
  public String toString() { return getName(); }
  public boolean equals(Object o) {
    if (o == null) return false;
    if (this == o) return true;
    if (!(o instanceof MailPrincipal)) return false;
    MailPrincipal rhs = (MailPrincipal)o;
    if (getName().equals(rhs.getName())) return true;
    return false;
  }
  public int hashCode() { return getName().hashCode(); }
}
```

ファイル 2

```
package mail.login;

import com.cosminexus.admin.auth.*;

import javax.security.auth.*;
import javax.security.auth.login.*;
import javax.security.auth.callback.*;
```

```
import javax.security.auth.spi.*;
import java.text.*;
import java.util.*;
import javax.mail.*;
import java.util.Properties;

import mail.MailPrincipal;

public class MailLoginModule implements LoginModule {
    private Subject subject;
    private CallbackHandler callbackHandler;
    private Map sharedState;
    private Map options;

    private static final String USER_HOST_OPT =
"mail.login.host";
    private static final String USER_PROV_OPT =
"mail.login.provider";

   private static final String USERNAME = "mail.login.username";
   private static final String PASSWORD = "mail.login.password";

    private static final String DEFAULT_PROV = "imap";

    private static final String MSG_USERNOTFOUND = "user not
found!";
    private static final String MSG_NOHOSTOPT = "not host
option!";

    private boolean commitSucceeded = false;

    private Store store;
    private String username;

    public void initialize(
        Subject subject,
        CallbackHandler callbackHandler,
        Map sharedState,
        Map options)
    {
        this.subject = subject;
        this.callbackHandler = callbackHandler;
        this.sharedState = sharedState;
        this.options = options;
    }

    public boolean login()
        throws LoginException
    {
        // get user-name & password
        this.username = (String)this.sharedState.get(USERNAME);
        if (username == null) throw new
LoginException(MSG_USERNOTFOUND);
      String password = (String)this.sharedState.get(PASSWORD);

        // option check
        String host = (String)this.options.get(USER_HOST_OPT);
        if (host == null || host.length()==0) throw new
LoginException(MSG_NOHOSTOPT);

        String provider =
(String)this.options.get(USER_PROV_OPT);
        if (provider == null ) provider = DEFAULT_PROV;

        // login
```

```
            Properties props = new Properties();
            Session session = Session.getInstance(props, null);
            try {
                store = session.getStore(provider);
                store.connect(host, username, password);
            } catch(Exception ex) {
                throw new LoginException(ex.toString());
            }
            return true;
        }

        public boolean commit() throws LoginException {
            this.subject.getPrincipals().add( new
    MailPrincipal(this.username) );
            this.subject.getPublicCredentials().add( this.store );
            return this.commitSucceeded = true;
        }

        public boolean abort() throws LoginException {
            try {
                if (store != null) store.close();
            } catch(MessagingException ex) {
                // Disregard
            }
            if (this.commitSucceeded) {
                logout();
            }
            return true;
        }

        public boolean logout() throws LoginException {
            try {
                if (store != null) store.close();
            } catch(MessagingException ex) {
                // Disregard
            }
            this.commitSucceeded = false;
            return true;
        }
    }
```

(2) 日立 API ポートレット

次に，メール情報を表示する日立 API ポートレットを作成します。このポートレットでは，ログインに成功したあと，subject からメール情報を取得しています。メール情報を表示するポートレットのサンプルコードを次に示します。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ taglib uri="http://cosminexus.com/admin/auth/uatags"
prefix="ua" %>
<%@ page import="com.cosminexus.admin.auth.*" %>
<%@ page import="com.cosminexus.admin.auth.sso.callback.*" %>
<%@ page import="javax.security.auth.*" %>
<%@ page import="javax.security.auth.login.*" %>
<%@ page import="java.security.*" %>
<%@ page import="javax.mail.*" %>
<%@ page import="java.util.Properties" %>

<%
  LoginContext lc = null;
  try {
```

```
      WebSSOHandler h =
        new WebSSOHandler(request, response, null);
      lc = new LoginContext("Mail", h);
      lc.login();
    }
    catch (LoginException e) {
      out.println("Mailへのログインに失敗しました。");
      return;
    }
    catch (Exception e) {
      out.println("Mailへのログインで障害が発生しました。");
      return;
    }

    try {
        Subject subject = lc.getSubject();
        Principal principal = (Principal)subject.getPrincipals().
          iterator().next();
        String mailid = principal.getName();
        Store store =
          (Store)subject.getPublicCredentials().iterator().next();
        out.println("メールID="+mailid+"<br />");
        Folder folder = null;
        try {
            folder = store.getFolder("INBOX");
            folder.open(Folder.READ_ONLY);

            Message msg[] = folder.getMessages();
            out.println("メール総数="+msg.length+"<br />");
            out.println("<br />");
            if (msg.length == 0) {
                out.println("メールはありません<br />");
            }
            else {
                out.println("メール一覧<br />");
                out.println("<table border>");
                {
                    // 先頭行
                    out.println("<tr>");
                    out.println(
                       "<th><font size=¥"4¥">項番</th>"
                      +"<th><font size=¥"4¥">送信者</th>"
                      +"<th><font size=¥"4¥">メールタイトル</th>");
                    out.println("</tr>");

                    // メールのタイトル表示
                    int loop = msg.length>10 ? 10 : msg.length;
                    for (int i = 0; i < loop ; ++i) {
                        out.println("<tr>");
                     out.print("<td align=¥"right¥">"+(i+1)+"</td>");
                        Address[] addrs = msg[i].getFrom();
                        String requestName = (addrs != null &&
addrs.length > 0) ?
                          addrs[0].toString() : "不明";
                        out.print("<td>"+requestName+"</td>");
                      out.print("<td>"+msg[i].getSubject()+"</td>");
                        out.println("</tr>");
                    } // for i
                }
                out.println("</table>");
            }
        }
        catch (Exception e) {
```

```
                out.println("Mailの表示中に障害が発生しました。");
                return;
            }
            finally {
                try {
                    if (folder != null) folder.close(false);
                }
                catch (MessagingException e) { e.printStackTrace(); }
            }
        }
        finally {
            try {
                lc.logout();
            }
            catch (Exception e) {
              out.println("Mailのログオフに障害が発生しました。");
              return;
            }
        }

    %>
    <hr />
```

(3) ポートレットを動作させるための設定

(2) の日立 API ポートレットを動作させるための設定について説明します。

1. (1) **で作成したカスタムログインモジュールをコンパイルして，{**Cosminexus **インストールディレクトリ}** ¥manager¥modules **に格納する**

   格納先のディレクトリはユーザ管理コンフィグレーションファイル（ua.conf）の com.cosminexus.admin.auth.custom.modules パラメタで変更できます。

2. JAAS **コンフィグレーションファイル（**jaas.conf**）を変更する**

   作成したカスタムログインモジュールとポータルのログインモジュールを関連づけるため JAAS コンフィグレーションファイル変更します。変更例を次に示します。

   ```
   Portal {
     com.cosminexus.admin.auth.sso.login.WebSSOLoginModule required
       com.cosminexus.admin.auth.realm="Portal"
       com.cosminexus.admin.auth.ldap.r="0"
       com.cosminexus.admin.auth.ldap.w="1"
       ;
   };
   Mail {
     com.cosminexus.admin.auth.sso.login.WebSSOLoginModule required
       com.cosminexus.admin.auth.sso="MailLM"
       com.cosminexus.admin.auth.realm="Mail"
       mail.login.host="localhost"  /* MailLoginModuleのオプション */
       ;
   };
   ```

3. **カスタムログインモジュールを統合ユーザ管理に認識させるため，ユーザ管理コンフィグレーションファイル（**ua.conf**）に次の内容を追記する**

   追加内容を次に示します。

   ```
   com.cosminexus.admin.auth.sso.lm.MailLM=mail.login.MailLoginModu
   le
   ```

```
com.cosminexus.admin.auth.sso.param.userid.MailLM=mail.login.use
rname
com.cosminexus.admin.auth.sso.param.secdat.MailLM=mail.login.pas
sword
```

4. **ポータルのユーザとメールのユーザのマッピングを** CSV **ファイルに作成し，** ssoimport **コマンドで登録する**

登録例を次に示します。この例では，ポータルのユーザの user1 は，メールのユーザの muser1 に対応していることを示しています。ポートレットの場合，最初にポータルにログインするため，ポータルのパスワードはマッピング情報に登録する必要がありません。

```
REALMNAME,USERID,SECRETDATA,PUBLICDATA,LINK_Mail
Portal,user1,,,muser1
Mail,muser1,pass1,,
```

付録 A.11　フレームを使用したポートレット

uCosminexus Portal Framework では，HTML のテーブル要素を使用して複数のポートレットをポータル画面に統合しています。そのため，Multi Web Portlet 以外のポートレット内で FRAMESET 要素を直接使用できません。ポートレットでフレームを表示するには，IFRAME 要素を使用します。

ポートレットでは，単一のフレームまたは複数のフレームを表示できます。単一のフレームを表示するには IFRAME 要素を使用します。複数のフレームを表示するには，IFRAME 内で FRAMESET 要素を使用します。

(1) 単一のフレーム（iframe）

単一のフレームを表示するサンプルを次に示します。HTML で iframe を使用する場合と同様に，iframe を使用できます。

{PROJECT_HOME}¥portlets¥iframe¥index.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI" %>
<%@ page contentType="text/html; charset=Shift_JIS" %>

<uu:iframe src="contents.jsp" width="100%">
IFRAMEをサポートしたWebブラウザを利用願います。
</uu:iframe><br>
```

{PROJECT_HOME}¥portlets¥iframe¥contents.jsp

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
                          "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Inline Contents</title>
</head>
<body>
<%
  // タグライブラリなどを利用してコンテンツを作成します
%>
</body>
</html>
```

(2) 複数のフレーム（iframe+frameset）

複数のフレームを表示するサンプルを次に示します。複数のフレームを使用したポートレットは，最大化時だけに使用することを推奨します。

{PROJECT_HOME}¥portlets¥frameset¥index.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page
import="jp.co.hitachi.soft.portal.portlet.beans.PortletInfoBean"
%>
<%
   PortletInfoBean pInfo = new PortletInfoBean();
   pInfo.initBean(request);
   PortletInfoBean.Mode mode = pInfo.getMode();
   if (mode == PortletInfoBean.Mode.DEFAULT) { // サマリ画面作成
     ...
   } else if (mode == PortletInfoBean.Mode.MAXIMIZE) { // 最大化画面
作成(frameset)
%>
<uu:iframe src="frameset.jsp" width="100%">
IFRAMEをサポートしたWebブラウザを利用願います。
</uu:iframe><br>
<%
   } else { // エラー画面
     ...
   }
%>
```

{PROJECT_HOME}¥portlets¥frameset¥frameset.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page contentType="text/html; charset=Shift_JIS" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head><title></title></head>
<frameset cols="20%,80%">
 <uu:frame src="navigation.jsp"/>
 <uu:frame src="main.jsp"/>
</frameset>
</html>
```

付録 A.12　セキュアなポートレットの開発

通常の日立 API ポートレットの登録方法では，ポータルのユーザが URL を直接指定してポートレットおよびインラインオブジェクトにアクセスするおそれがあります。その場合，ポータルにアクセスしたユーザに JSP ファイル，および JSP ファイルに含まれるインラインオブジェクトが表示されるおそれがあります。

ポートレットおよびインラインオブジェクトを，セキュリティを考慮してポータルで使用する方法について説明します。

(1) アクセス判定するサーブレットの作成

セキュアなポートレットを作成するには，アクセス判定するサーブレットを作成します。サーブレット内のコードのうち，アクセス判定の基準として，ユーザ情報取得 Bean も使用できます。ユーザ情報取得 Bean については，「14.6　ユーザ情報取得 Bean」を参照してください。

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import jp.co.hitachi.soft.portal.api.user.PortalUserInfoBean;
import java.net.*;
public class securecontents extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse
res){
      try{
        ServletOutputStream out = res.getOutputStream();
        String url = req.getParameter("url");
          // サイトのエンコーディングがShift_JISの場合，次の処理を行う。
       url = new String(url.getBytes("iso-8859-1"),"Windows-31J");
        PortalUserInfoBean puibean = new PortalUserInfoBean();
        puibean.setRequest(req);
        String userid = puibean.getUserId();
        if (  アクセス判定 == true ){
            if ( url.endsWith("jsp") ){
                req.getRequestDispatcher(url).forward(req,res);
            }else{
                java.net.URL nurl =
getServletContext().getResource(url);
              java.net.URLConnection urlc = nurl.openConnection();
               res.setContentLength(urlc.getContentLength());
               res.setContentType(urlc.getContentType());
              byte  barray[] = new byte[urlc.getContentLength()];
               InputStream is = urlc.getInputStream();
               is.read(barray);
               out.write(barray);
          }
        }else{
               // アクセス拒否の応答を返却する
               res.setStatus(404);
               out.print("access error");
        }
      }catch(Exception e){}
```

```
    }
}
```

(2) セキュアなポートレットの設定

セキュアなポートレットを作成するには，次の方法があります。

- すべてのコンテンツをセキュアにする方法
- 一部のコンテンツをセキュアにする方法

すべてのコンテンツをセキュアにする場合，セキュリティ強度は上がりますが，Web サーバの負荷も上がります。

環境に応じて，どちらの方法を選択するかを決定します。

(a) すべてのコンテンツをセキュアにする方法

1. **Web アプリケーションの DD（web.xml）を設定する**

   web.xml に次の設定を追加します。

```
<web-app>
    ...
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected Area</web-resource-name>
        <url-pattern>(URL)</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <auth-constraint>
            <role-name>dummy</role-name>
        </auth-constraint>
    </web-resource-collection>
</security-constraint>
</web-app>
```

   <url-pattern>(URL)</url-pattern>

   (URL) には，ポートレットディレクトリ下のすべてのディレクトリを指定します。

   例：/portlets/mailportlet/*

   <auth-constraint>

   <role-name> には実在しないロールを指定します。

   web.xml の格納場所を次に示します。

   {PROJECT_HOME} ¥WEB-INF

2. **URL の記述形式を変更する**

   アクセス判定するサーブレット経由でコンテンツを呼び出す設定に変更します。
   ポートレットユティリティタグライブラリを用いて画面内遷移しているポートレットは，変更する必要はありません。
   セキュアにするコンテンツを参照しているすべてのファイルで，参照先 URL を次の形式に変更します。変更後は，ポートレットユティリティタグライブラリは使用でき

ません。

/ SecureContents?url=(URL)

変更例を次に示します。

変更前

```
<img src="/portlets/mailportlet/images/icon.gif">
```

変更後

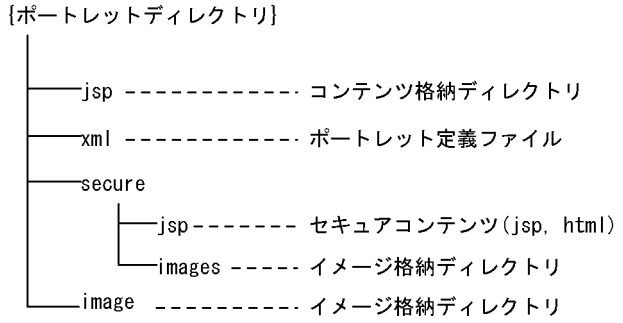
```
<img src="/SecureContents?url=/portlets/mailportlet/images/
icon.gif">
```

(b) 一部のコンテンツをセキュアにする方法

1. **ディレクトリ構成を次のとおりに作成する**

一部のコンテンツをセキュアにするためのディレクトリ構成を次の図に示します。

図 A-4　一部のコンテンツをセキュアにするためのディレクトリ構成



2. **Web アプリケーションの DD（web.xml）を設定する**

web.xml に次の設定を追加します。

```
<web-app>
...
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected Area</web-resource-name>
        <url-pattern>(URL)</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <auth-constraint>
            <role-name>dummy</role-name>
        </auth-constraint>
    </web-resource-collection>
</security-constraint>
</web-app>
```

<url-pattern>(URL)</url-pattern>

(URL) には，セキュアにするコンテンツのディレクトリを指定します。

例：/portlets/mailportlet/secure/*

<auth-constraint>

<role-name> には実在しないロールを指定します。

web.xml の格納場所を次に示します。

{PROJECT_HOME} ¥WEB-INF

3. URL の記述形式を変更する

アクセス判定するサーブレット経由でコンテンツを呼び出す設定に変更します。
ポートレットユティリティタグライブラリを用いて画面内遷移しているポートレットは，変更する必要はありません。
セキュアにするコンテンツを参照しているすべてのファイルで，参照先 URL を次の形式に変更します。変更後は，ポートレットユティリティタグライブラリは使用できません。

/ SecureContents?url=(URL)

変更例を次に示します。

変更前

```
<img src="/portlets/mailportlet/images/icon.gif">
```

変更後

```
<img src="/SecureContents?url=/portlets/mailportlet/images/
icon.gif">
```

付録 A.13　バックエンドシステム連携

バックエンドシステムと連携するポートレットの開発について説明します。ポートレットとバックエンドシステムを連携させる場合，エンドユーザのリクエストのたびにバックエンドにログインして問い合わせる方法（ステートレス）と，バックエンドとのコネクションを保持したまま問い合わせる方法（ステートフル）があります。

(1) ステートレス接続

ステートレスな接続の場合，あらかじめ取得しておいた LoginContext からログイン情報を取得し，バックエンドへの問い合わせ，およびログアウトをします。ステートレスな連携の場合，状態を保持しないのでポータルのログアウトやポータルのユーザセッションタイムアウトは意識する必要はありません。

次にサンプルコードを示します。

```
<%@ page import="javax.security.auth.login.LoginContext" %>
<%@ page import="javax.security.auth.Subject" %>
<%@ page import="javax.security.auth.Principal" %>
<%@ page
import="com.cosminexus.admin.auth.callback.WebPasswordHandler" %>
<%@ page import="com.cosminexus.admin.auth.UserAttributes" %>

<%
  /*
  * ユーザID，属性取得処理
  */
  ユーザID，属性取得処理;

  //  バックエンドシステム問い合わせ
  バックエンドシステムログイン；
```

```
  問い合わせ処理；
  バックエンドシステムログアウト；
%>

<!--バックエンドシステムの結果を出力 -->
ポートレットコンテンツ出力
```

(2) ステートフル接続

ステートフルな接続の場合，バックエンドとのコネクションを保持しておく必要があります。ポータルのユーザセッションに対応する HttpSession を利用すると，コネクションを保持できます。ログアウトは，HttpSessionBindingListener を利用します。HttpSessionBindingListener を利用すると，ユーザがポータルをログアウトしたタイミングでバックエンドシステムのログアウト処理をします。

MyLoginContext クラスはログイン情報を保持するクラスで，HttpSessionBindingListener インタフェースを実装しています。また，valueUnbound メソッドでバックエンドシステムのログアウト処理を記述します。valueUnbound メソッドは，MyLoginContext が HttpSession から削除されるとき，つまり，ログアウトのタイミングで呼び出されます。

次にサンプルコードを示します。

```
import javax.servlet.http.*;
import javax.security.auth.login.LoginContext;

class MyLoginContext
    implements HttpSessionBindingListener, java.io.Serializable {
  public MyLoginContex(...) {
    ...
  }

  public void valueBound(HttpSessionBindingEvent event) { ... }

  /**
  * ログアウト処理をする。このメソッドはセッションが無効(ユーザログアウト時,
  * セッションタイムアウト時)に呼び出される。
  */
  public void valueUnbound(HttpSessionBindingEvent event) {
    HttpSession event.getSession();
    バックエンドシステムログアウト処理；
  }
}

<%@ page import="javax.security.auth.*" %>
  ...

<%
  if (ログイン必要) {
    LoginContext取得処理;
    ユーザID，属性取得処理;
    バックエンドシステムログイン；

    バックエンドシステムとの連携に必要な情報をMyLoginContextに設定；
    HttpSessionにMyLoginContextを格納；
```

```
  }
%>

<%
  HttpSessionからログイン情報取得；
  問い合わせ処理；
%>
<!--バックエンドシステムの結果を出力 -->
ポートレットコンテンツ出力
```

付録 A.14　ファイルをアップロードするポートレット

文書管理システムを構築する場合，ファイルをサーバにアップロードする必要があります。ポートレットでは，通常の Web でファイルをアップロードするのと同様に，ファイルをアップロードできます。

ファイルをアップロードするポートレットを次に示します。

{PROJECT_HOME}¥portlets¥fileupload¥index.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page contentType="text/html; charset=Shift_JIS" %>
<uu:form action="upload.jsp" enctype="multipart/form-data"
method="POST">
アップロードするファイル：<br>
<input type="file" name="Uploaded file">
<input type="submit" value="アップロード">
</uu:form>
```

{PROJECT_HOME}¥portlets¥fileupload¥upload.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%
  try {
    String ct = request.getHeader("Content-Type");
    String mimeType = ct.substring(0, ct.indexOf(";")).trim();
    if (mimeType.equals("multipart/form-data")) {
      String boundary = ct.substring(ct.indexOf("boundary=")+9,
ct.length());
      // ファイルを取り出して必要な処理をする。
    } else {
      new Exception("Unsupported MIME type");
    }
    // 受け付け画面生成
  } catch (Exception e) {
    // エラー画面生成
  }
%>
```

付録 A.15　プッシュ型ポートレット

ポートレットでプッシュ配信するには，プッシュ配信したいコンテンツを iframe 内に配

置して，配置したコンテンツを，Web ブラウザに定期的に更新させるようにします。

ポートレットでプッシュ配信するサンプルを次に示します。このサンプルでは，META タグの http-equiv 属性を使用しています。

{PROJECT_HOME}¥portlets¥emergency¥index.jsp

```
<%@ taglib uri="http://soft.hitachi.co.jp/portal/urlutils"
prefix="uu" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI"
%>
<%@ page contentType="text/html; charset=Shift_JIS" %>

<uu:iframe src="contents.jsp" width="100%">
IFRAMEをサポートしたWebブラウザを利用願います。
</uu:iframe><br>
```

{PROJECT_HOME}¥portlets¥emergency¥contents.jsp

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI"
%>
<html>
<head>
  <title>緊急連絡事項</title>
  <meta http-equiv="refresh" content="600"><!-- 10分置きに更新 -->
</head>
<body>
<%
  // ファイル，DBなどから連絡事項を取得して出力
%>
</body>
</html>
```

付録 A.16　プラグインモジュールと連携するポートレット

Java Applet や Macromedia Flash など，Web ブラウザにロードされるプラグインモジュールと連携するポートレットを作成する方法について説明します。

プラグインモジュールと連携するには，ポートレットの URL が必要です。また，セッションを維持して連携するには，セッション ID が必要です。OBJECT タグのパラメタを使用して，ポートレットの URL やセッション ID をプラグインモジュールに渡すと，プラグインモジュールと連携できます。

ここでは，Macromedia Flash と連携するサンプルを示します。Java Applet の場合は，PARAM タグを使用して，必要な情報を渡せます。それ以外のプラグインモジュールも同じ方法で連携できます。パラメタを引き渡す方法については，各プラグインモジュールのマニュアルを参照してください。

Macromedia Flash と連携するサンプルファイルの構成を次に示します。

サンプルファイルの構成

```
   {PROJECT_HOME}¥portlets¥flash¥index.xml
                                ¥flash.jsp
                                ¥sample.swf     FLASHファイル
```

flash.jsp を次に示します。

```
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletURI" %>
<%@ page import="jp.co.hitachi.soft.portal.portlet.PortletUtils" %>
<%
  String nameSpace = PortletUtils.getNamespace(request, response);
  String flashUrl = (String)session.getAttribute(namespace+"url");
  if (Mode. == DEFAULT || Mode= MAXIMIZED) {
  if (flashUrl == null) {
     String me = PortletURI.transInlineURI(request, response, "");
     String baseUrl = me.substring(0, me.lastIndexOf('/'));
     session.setAttribute(nameSpace+"base", baseUrl);

     Cookie cookies[] = request.getCookies();
     for (int i = 0; i < cookies.length; i++) {
       if (cookies[i].getName().equalsIgnoreCase("jsessionid")) {
          sessionID = cookies[i].getValue();
          break;
       }
     }
     flashUrl = PortletURI.transInlineURI(request, response,
              new
StringBuffer(64).append("sample.swf").append("?PortletURL=").appen
d(baseUrl)

.append("&SessionID=").append(sessionID).toString());
     session.setAttribute(nameSpace+"url", flashUrl);
  }
  }
%>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
swflash.cab#version=4,0,2,0"
 ID="sample" WIDTH="100" HEIGHT="100">
   <param name="movie" value="<%= flushUrl %>">
</OBJECT>
```

付録 B　セルフユーザ登録のサンプルポートレット

uCosminexus Portal Framework では，ポータル利用者自身が，ユーザ情報を登録して利用するためのサンプルポートレット（セルフユーザ登録ポートレット）を提供しています。

セルフユーザ登録ポートレットでは，次に示す操作が実行できます。

- ユーザの新規登録
- ユーザ情報の変更
- パスワードの変更

セルフユーザ登録ポートレットはそのままでも使用できますが，実際の業務で使用する場合には，必要なユーザ情報の追加や変更，セキュリティや性能を考慮したカスタマイズを行ってください。サンプルポートレットのカスタマイズ方法および登録方法の詳細は，サンプルポートレットの「Readme.txt」を参照してください。

サンプルポートレットおよびサンプルポートレットの「Readme.txt」は，次に示すディレクトリに格納されています。

格納ディレクトリ

```
{uCosminexus Portal Framework インストールディレクトリ
}¥samples¥portlets¥AccountManager
```

なお，セルフユーザ登録ポートレットでは，セルフユーザ登録ポートレット用 API を使用します。セルフユーザ登録ポートレット用 API の詳細は，「14.18　ユーザ登録 API」および「14.19　ユーザアカウント情報取得 API」を参照してください。

付録 C　各バージョンの変更内容

変更内容 (3020-3-H73-40)　uCosminexus Portal Framework　08-70，**および**
uCosminexus Portal Framework - Light　08-70

追加・変更内容
PortletURI クラスに transInlineURIEx メソッドを追加しました。

変更内容 (3020-3-H73-30)　uCosminexus Portal Framework　08-03，**および**
uCosminexus Portal Framework - Light　08-03

追加・変更内容
PortletURI クラスに transContextURI メソッドを追加しました。

変更内容 (3020-3-H73-20)　uCosminexus Portal Framework　08-02，**および**
uCosminexus Portal Framework - Light　08-02

追加・変更内容
Web ポートレットの作成で iframe タグをサポートする Web ブラウザに Firefox を追加しました。
分散ポートレットの制限事項の記述を修正しました。
クライアントサイドデータ通信対応ポートレットの定義規則（パターン 2 およびパターン 3）の説明を変更しました。

付録 D　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 D.1　関連マニュアル

関連マニュアルを次に示します。必要に応じてお読みください。

＜ uCosminexus Portal Framework のマニュアル＞

uCosminexus Portal Framework についての，他マニュアルを次に示します。

uCosminexus Portal Framework　**システム管理者ガイド**（3020-3-H71）

uCosminexus Portal Framework 全体の機能概要について説明しています。また，ポータル全体の管理者（システム管理者）が実施する作業（ポータルの構築方法，ポータルのカスタマイズ方法，ポートレットの登録方法，およびポータルの起動方法）について説明しています。

uCosminexus Portal Framework　**運用管理者ガイド**（3020-3-H72）

ポータルを運用する管理者（運用管理者および部門管理者）が実施する作業（ポータルの運用管理および標準画面レイアウトの作成）について説明しています。

uCosminexus Portal Framework　**ユーザーズガイド**（3020-3-H74）

エンドユーザがポータルを使用するときの操作方法（ポータルへのログイン方法，ポータル画面のカスタマイズ方法，およびナビゲーションメニューの操作方法）について説明しています。

＜他製品のマニュアル＞

このマニュアルで参照している他製品のマニュアルを次に示します。

Collaboration　**導入ガイド**（3020-3-H01）

Collaboration 製品を導入するための，システム構築，環境設定，および運用方法について説明しています。

Collaboration - Online Community Management　**システム管理者ガイド**（3020-3-H03）

コミュニティ管理を利用するための環境設定および運用方法について説明しています。

Collaboration - Online Community Management　**ユーザーズガイド**（3020-3-H04）

コミュニティ管理の機能および操作方法について説明しています。

Cosminexus　**アプリケーションサーバ** V8　**システム構築・運用ガイド**（3020-3-U04）

Cosminexus Version 8 のシステムを構築・運用する方法について説明しています。

Cosminexus　**アプリケーションサーバ** V8　**機能解説　拡張編**（3020-3-U08）

Cosminexus Version 8 のアプリケーションサーバの機能の詳細について，アプリケーションの実装方法や実行環境で必要な設定などを含めて解説しています。

Cosminexus　**アプリケーションサーバ** V8　**機能解説　運用／監視／連携編**
（3020-3-U09）

Cosminexus Version 8 のアプリケーションサーバの機能の詳細について，アプリケーションの実装方法や実行環境で必要な設定などを含めて解説しています。

Cosminexus　**アプリケーションサーバ** V8　**リファレンス**　API **編**　**（3020-3-U26）**

Cosminexus Version 8 のアプリケーションの開発で使用する API およびタグについて説明しています。

Cosminexus V9　**アプリケーションサーバ　システム構築・運用ガイド**　**（3020-3-Y02）**

Cosminexus Version 9 のシステムを構築・運用する方法について説明しています。

Cosminexus V9　**アプリケーションサーバ　機能解説　拡張編**　**（3020-3-Y08）**

Cosminexus Version 9 のアプリケーションサーバの機能の詳細について，アプリケーションの実装方法や実行環境で必要な設定などを含めて解説しています。

Cosminexus V9　**アプリケーションサーバ　機能解説　運用／監視／連携編**
（3020-3-Y10）

Cosminexus Version 9 のアプリケーションサーバの機能の詳細について，アプリケーションの実装方法や実行環境で必要な設定などを含めて解説しています。

Cosminexus V9　**アプリケーションサーバ　リファレンス**　API **編**　**（3020-3-Y21）**

Cosminexus Version 9 のアプリケーションの開発で使用する API およびタグについて説明しています。

付録 D.2　このマニュアルでの表記

このマニュアルでは，製品名称を次に示す略称で表記しています。

正式名称	略称
Enterprise JavaBeans(TM)	EJB
Java(TM)	Java
Java(TM) 2 Platform, Enterprise Edition	J2EE
Java(TM) 2 Platform, Standard Edition	J2SE
Java(TM) Authentication and Authorization Service	JAAS
Java(TM) Servlet	Servlet またはサーブレット
JavaServer Pages(TM)	JSP

付録 D.3　英略語

このマニュアルで使用する主な英略語を次に示します。

英略語	説明
API	Application Programming Interface
CHTML	Compact Hypertext Markup Language
DB	Database
DD	Deployment Descriptor
DTD	Document Type Definition
EAR	Enterprise Archive
EUC	Extended UNIX Code
HDML	Handheld Device Markup Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
JAR	Java Archive
JIS	Japan Industrial Standard
JSP	JavaServer Pages
LDAP	Lightweight Directory Access Protocol
MIME	Multipurpose Internet Mail Extension
MVC	Model View Controller
OS	Operating System
PAR	Portlet Application Archive
PC	Personal Computer
RDF	Resource Description Framework
RSS	RDF Site Summary
SGML	Standard Generalized Mark-up Language
SP	Service Pack
SSL	Secure Socket Layer
SSO	Single Sign-On
UCS	Universal multi-octet coded Character Set
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	UCS Transformation Format
WAR	Web Archive
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

付録 E　用語解説

（英字）

DTD（Document Type Definition）

「文書型定義」のことです。SGML や XML で文書を記述する際，その文書中でどのようなタグや属性が使われているかを定義したものです。

File ポートレット

HTML，CHTML，HDML のどれかで記述されたコンテンツを，ポータル画面に表示するためのポートレットです。

LDAP（Lightweight Directory Access Protocol）

ディレクトリサーバにアクセスするための通信プロトコルです。

RSS（RDF Site Summary）

RSS は RDF 形式で記述した Web サイトのコンテンツを要約したファイルです。XML で記述されています。なお，RDF は Web ページでコンテンツの構成を記述するためのメタデータの規格です。

SSO（Single Sign-On）

ユーザが一度認証を受けるだけで，許可されているすべての機能を利用できるようになるシステムです。

UserAgent

アクセス端末ごとの識別情報です。

Web コンテナ

uCosminexus Application Server の Web コンテナ機能を指します。Web コンテナは JSP およびサーブレットを実行します。

Web コンテンツポートレット

外部の Web サーバからコンテンツを取得し，ポータル画面に表示するためのポートレットです。取得したデータは，あらかじめ設定した抽出方法やソート方法などに従って加工して表示されます。取得できるコンテンツは，HTML，RSS，または XML で記述されたページです。

Web ポートレット

HTML で記述された Web ページをポータルに取り込み，ポータル画面に表示するためのポートレットです。ポータルプロジェクト内のローカルコンテンツ，および外部の Web サーバのコンテンツをポータルに統合できます。

（ア行）

アプリケーションキー
ストリングリソースを区別するためのキーのことです。

インラインオブジェクト
HTML ファイルなどのコンテンツ中に含まれる画像などのことです。

ウェルカム画面
ポータルに最初にアクセスしたときに表示される画面（ログイン前の画面）です。ポータルユーザ全員に知らせたい情報などを表示します。

運用管理者
ポータルの運用を管理する権限を持つユーザです。システム管理者によって任命されます。運用管理ポートレットを使用して，ポータルの運用管理に必要なすべての情報（ポータル管理グループ，ポートレット，および画面レイアウト）を設定できます。
また，ほかのユーザを「部門管理者」に任命して，運用管理権限の一部を委譲できます。

運用管理ポートレット
運用管理者がポータルを運用管理するためのポートレットです。日立 API ポートレットにより作成されます。

エリア
拡張レイアウト形式の，ポートレットを配置する一つの行または列のことです。エリアには，ユーザがポートレットの配置を変更できる変更可能エリアと，配置を変更できない変更不可エリアがあります。変更不可エリアのポートレットの配置を変更できるのは，運用管理者と部門管理者だけです。

（カ行）

拡張レイアウト形式
行列形式を組み合わせてポートレットを表示できるレイアウト形式です。拡張レイアウト形式では，例えば，行の中に列を入れ子にするなど，より複雑にポートレットを配置できます。また，ユーザがポートレットの配置を変更できるかどうかを，ポートレットを配置する一つの行または列（エリア）ごとに設定できます。

カスタマイズ
ログインしたあとに表示される標準のポータル画面のコンテンツやレイアウトを，ユーザごとに変更することです。

カスタムポートレット
JSP やサーブレットの技術を用いて独自に作成したコンテンツを，ポータル画面に表示するためのポートレットです。動的なページを作成したり，企業独自の業務システムをポータルに統合したり，既存のアプリケーションと連携したりできます。標準 API ポートレットと日立 API ポートレットがあります。

管理者用レイアウトカスタマイズ画面

運用管理ポートレットから表示される画面です。この画面で，運用管理者または部門管理者が，同じポータル管理グループに属するユーザのポータル画面に表示される共通のレイアウト（標準画面レイアウト）をカスタマイズします。

クライアントサイドデータ通信

「ドラッグ & ドロップ」および「データフォーム転送」の総称です。

コピーイベントハンドラ

シリアライズされた Java オブジェクトのデータを，ポータルクリップボードにコピーする JavaScript の定義のことです。

コミュニティ

Collaboration - Online Community Management の管理情報で，コミュニティメンバとワークプレースをまとめた集合体のことです。

コミュニティメンバ（省略時はメンバ）

Collaboration - Online Community Management 管理情報で，コミュニティに参加中または脱退依頼中の人のことです。

コンテンツ表示領域

ポートレットが出力するコンテンツ領域のことです。

コンテンツフィルタリング

Web ポートレットが表示するコンテンツを一定の条件で部分的に削除または抽出することです。

（サ行）

最大化ボタン

ポートレットの最大化ボタン（最大化時には復元ボタン）のことです。

システム管理者

ポータル全体を管理する権限を持つユーザです。ポータルの構築，ポートレットの登録，および Portal Manager でポータル全体に関する情報を設定します。また，ほかのユーザを運用管理者に任命できます。

シリアライズ

メモリ上のデータをファイルなどに保存できる形式に変換することです（対義語はデシリアライズ）。

ストリングリソース

ポータル画面に表示する文字列を言語別にまとめたリソースのことです。

セルフユーザ登録ポートレット

ポータル利用者自身が，ユーザ情報を登録して利用するためのポートレットです。

（タ行）

タイトルバー

uCosminexus Portal Framework が合成するタイトルバーのことです。

ディレクティブ

コンテンツフィルタリング機能でフィルタリングする位置を指定する指示子です。HTML ファイルのコメント，および正規表現で検索した文字列をディレクティブとして使用します。

ディレクトリサーバ

情報を階層構造で一元管理するサーバです。uCosminexus Portal Framework では，LDAP を使用したディレクトリサーバで，ユーザ情報を管理します。

デシリアライズ

ファイルなどに保存できるデータ形式から，メモリ上のオブジェクトに変換することです（対義語はシリアライズ）。

統合画面

ポータルサーバが提供する View（ナビゲーション領域とタイトルバー）とポートレットとして開発されたコンテンツの組み合わせによって表示される画面です。標準画面（DEFAULT モード），最大化画面（MAXIMIZE モード），編集画面（EDIT モード），および新規ウィンドウ画面（NEWWINDOW モード）の 4 種類があります。

統合ユーザ管理フレームワーク

Cosminexus で，JAAS を用いてユーザを管理するフレームワークです。uCosminexus Portal Framework は，統合ユーザ管理フレームワークを使用してディレクトリサーバと連携してユーザを管理します。

ドキュメントベース URL 変換

ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，uCosminexus Portal Framework を経由しないでコンテンツを外部 Web サーバから直接取得するための URL に変換することです。URLRewriting はしません。また，uCosminexus Portal Framework の API を使用できません。

ドロップ抑止機能

ドラッグ & ドロップをする際に，マウスカーソルがドロップ対象領域に入っていない場合はマウスカーソルを禁止マークにし，マウスカーソルがドロップ対象領域に入っている場合はドロップ可能マークにする機能です。ただし，ドロップ対象領域に入っていてもドラッグ元とドロップ先の datatype が一致していない場合は，マウスカーソルは禁止マークになります。

（ハ行）

パーソナライズ

パーソナライズとは，ポータル画面の表示形式および表示するコンテンツを，ユーザごとまたはグループごとに変更することです。パーソナライズには，運用管理者または部門管理者によるポータルの標準画面の設定と，各ユーザによるポータルの標準画面のカスタマイズがあります。

日立 API ポートレット

カスタムポートレットの一つで，uCosminexus Portal Framework の API を用いて作成したポートレットです。

標準 API ポートレット

カスタムポートレットの一つで，Java Portlet Specification 1.0 に従って作成されたポートレットです。

部門管理者

ポータルの運用を管理する権限を持つユーザです。運用管理者によって，ポータル管理グループごとに設定されます。運用管理ポートレットを使用して，ポータル管理グループ内のリソース（ポートレットおよび画面レイアウト）についての情報を設定できます。ただし，リソースの管理権限は設定できません。

分散ポートレット

ほかのポータル上に配置されたポートレットを自ポータルに取り込み，ポータル画面に表示するためのポートレットです。

ペーストイベントハンドラ

ポータルクリップボードからデータを取得し，デシリアライズする JavaScript の定義のことです。

変更可能エリア

拡張レイアウト形式のエリアの一つです。変更可能エリアでは，ユーザがポートレットを追加，移動および削除できます。

変更不可エリア

拡張レイアウト形式のエリアの一つです。変更不可エリアは，管理者がユーザに必ず参照させたいポートレットを配置するための領域です。このため，変更不可エリアでは，ユーザがポートレットを追加，移動および削除することはできません。

ポータル

幾つかのポートレットで構成された Web サイトです。さまざまな情報を統合したインターネットやイントラネットの入り口になります。

ポータル URL 変換

ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，uCosminexus Portal Framework を経由してコンテンツを取得するための URL に変換することです。uCosminexus Portal Framework を経由するので uCosminexus Portal Framework の API を使用できます。また，ポートレットのアクセス制御も適用されます。ただし，統合アダプタを経由しないためポータルの統合画面内で遷移できません。

ポータル管理グループ

リソース（ポートレットおよび画面レイアウト）のアクセス権を管理するためのグループです。リソースの管理者および利用者を定義します。リソースの管理者は，リソースの内容を編集したり，アクセス権を変更したりできます。リソースの利用者は，ポータル画面にリソースが表示されて，リソースを参照したり使用したりできます。

ポータルクリップボードウィンドウ
ポータルクリップボードを実現するためのウィンドウです。

ポータルサーバ
uCosminexus Portal Framework を実行して動作している servlet のことです。

ポータルプロジェクト
ポータルを識別するためのディレクトリ名です。一つのポータルをポータルプロジェクトとして運用します。

ポートレット
ポータル上で動作するアプリケーションコンポーネントです。

ポートレット URL 変換
ポートレットユティリティタグライブラリおよびポートレットユティリティクラスライブラリを使用した相対パス形式の URL から，日立 API ポートレットをポータルの統合画面内で遷移させるための URL に変換することです。ポータル画面で View を統合するための統合アダプタを経由します。必要に応じて HttpServletResponse#encodeURL を呼び出して，URLRewriting をします。

ポートレット開発者
ポータルに表示するコンテンツを開発するユーザです。

ポートレットコンテンツ
ポートレットとして定義されているコンテンツのことです。

ポートレットタイトル
ポートレットのタイトルのことです。

ホーム画面
ウェルカム画面からログインしたときに表示されるユーザのホーム画面です。
uCosminexus Portal Framework では，運用管理者または部門管理者によって，標準のホーム画面を設定することもできます。標準のホーム画面を設定すると，ユーザが所属している組織や役職ごとに異なるホーム画面を表示できます。

（マ行）

メッセージキー
ストリングリソース内のメッセージを識別するためのキーのことです。

（ヤ行）

ユーザ
ポータルを使用する人のことです。

要素フィルタリング

Web ページをポータル画面に取り込む際に，開始タグ，終了タグ，などの要素を削除することです。

（ラ行）

リポジトリ

ユーザ情報，組織単位情報などが格納されているデータストアです。ディレクトリサーバ，DB などの総称です。

利用者用レイアウトカスタマイズ画面

各ユーザのポータル画面の［レイアウト変更］アンカーまたはボタンから表示されます。この画面では，各ユーザが自分のポータル画面を利用しやすいようにカスタマイズします。

レイアウト

利用者に表示する画面のことです。レイアウトは，複数のポートレットを集約して構成されます。レイアウトは，パーソナライズ機能で示した行列形式だけではなく，タブ形式など複数のレイアウトを持つ形式も含みます。

レイアウトカスタマイズ機能

ポータル画面のレイアウトをカスタマイズする機能です。ポータル画面に表示するポートレットやポートレットの配置などを設定できます。
レイアウトカスタマイズ機能の画面（レイアウトカスタマイズ画面）には，管理者用レイアウトカスタマイズ画面と利用者用レイアウトカスタマイズ画面の 2 種類があります。

レイアウト形式

レイアウトにどのようにポートレットを配置するか定義したものです。レイアウト形式には，タブ形式，行列形式，ユーザ登録形式，グリッド形式，フロー形式，拡張レイアウト形式などがあります。

レイアウト情報

ディレクトリサーバのユーザエントリ内にある uCosminexus Portal Framework 専用オブジェクトクラス hptluser 下の hptlPersonalizeInfo 属性内の「レイアウトグループ」（layoutid）の内容です。

（ワ行）

ワークプレース

Collaboration - Online Community Management の管理情報で，同じコミュニティに参加しているコミュニティメンバ間で共有するコンテンツを管理するものです。

索引

記号

数字

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

あ

い

う

え

か

き

く

け

こ

さ

し

す

せ

そ

た

て

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

み

む

め

ゆ

よ

ら

り

れ

ろ

わ