PALポータル 設定ガイド v.1.1-rev2

Configuration Guide

Portal Application Laboratory

2008/11/04 06:52

Table of Contents

1.1 はじめに

対象読者

このドキュメントは、PALポータルの設定を担当するユーザーを対象にしています。

お読みになる前に

このドキュメントでは、PALポータルの詳細な設定方法を示しているため、PALポータルの基礎的な知識が必要になります。

オンラインでのアクセス

ダウンロード、専門的サービス、サポート、その他の開発者情報については、次にアクセスしてください。

- プロジェクトサイト: http://pal.sourceforge.jp/

技術的なサポートの連絡先

本製品に関する技術的質問で、ドキュメント内に解決策が得られない場合は、次にアクセスしてください。

- 公開フォーラム: http://sourceforge.jp/forum/?group_id=1972
- メーリングリスト: http://sourceforge.jp/mail/?group_id=1972

関連サードパーティー Web サイトの参照

Portal Application Laboratoryプロジェクトでは、このドキュメントに記載されているサードパーティーの Web サイトの有効性については責任を持ち ません。 Portal Application Laboratoryプロジェクトはそのようなサイトやリソースを通じて、利用可能なコンテンツ、広告、製品、サービス、その他のドキ>ュメントなどについて、保証、責任、義務を負いません。 Portal Application Laboratoryプロジェクトはそのようなサイトやリソースと通じて、利用可能なコンテンツ、広告、製品、サービス、その他のドキ>ュメントなどを、使用または信用したり、それに関連して発生または申し立てられた、一切の損傷や損害に対しては責任または義務を負いません。

コメントおよび提案の送付方法

Portal Application Laboratoryプロジェクトは、このドキュメントの改善に努めており、読者からのコメントおよび提案などを歓迎しています。

- メーリングリスト: http://sourceforge.jp/mail/?group_id=1972

2.1 ポータルの概要

ポータルについて

ビジネスに必要な情報が企業の内外に散在し、その情報量は日々増え続けている今日において、情報を容易に検索・共有できる環境を整え、生産性を高め、的確で迅速なビジネスをすることは今日の課題です。 その現実的な解の一つとして、企業ポータル（Enterprise Information Portal：EIP）が注目を集めています。

企業情報ポータルを導入することで、企業内外に散在する様々なリソースを 1 箇所に統合し、利用者に適した形態でサービスを提供することが可能になります。 PALポータルは、企業情報ポータルとして、利用者に必要としているサービスを迅速に提供し、集約されたリソースを最大限に利用できる環境を構築します。 また、PALポータルは、オープンな仕様に基づき、適切なアーキテクチャで設計されているため、開発作業から配備・運用に至る、すべての作業が容易に行うことができ、コスト削減や生産性を向上することができます。

PALポータルは、Apache Software Foundation にて開発をされている、企業情報ポータルのオープンソースの実装 Jetspeed 2 をベースとして構築されています。 そのため、企業ポータルの基盤となる部分の開発は多くの開発者により、適切な仕様に基づき、柔軟で拡張が可能な設計が行われています。 柔軟に機能を拡張や変更ができるので、PALポータルは企業内ポータルに限らず、オンラインショップや SNS などの様々なユーザーポータル構築に対応することができます。

PALポータルでは、1 つのサイトを通して、エンドユーザへアプリケーションサービス、データベースの情報やその他の利用可能なデータソースを提供できます。 PALポータルは、各利用者へ提供される情報や機能が、利用者または利用者が属する役割を元にカスタマイズ可能なセキュリティ基盤を提供します。 利用者は、ウェブブラウザや携帯電話などの多くのデバイスで、ポータルへアクセスすることが可能です。

ポートレットについて

ポータル上では、様々なコンテンツを集約することができます。 ポートレットがその集約されたコンテンツの一部を形成します。

ポートレットは、Java ベースの UI コンポーネントです。 ポートレットは、ポータル上でコンテンツの一部を生成するウェブアプリケーションになります。 ですので、ポータルを稼働させたまま、ポートレットを配備または配備解除をすることができます。

PALポータルは、Java のポートレットの標準仕様である JSR 168 に準拠したポータルサーバーです。 ですので、他ベンダーのポータルサーバー上で動作している JSR 168 のポー

トレットも配備して、動作させることが可能です。

ポートレットAPI 1.0（JSR 168）

ポートレットAPI 1.0（JSR 168）は、Java Community Process（ http://jcp.org/ ）により定義されているポートレットの標準仕様になります。 以下のことなどを定義しています。

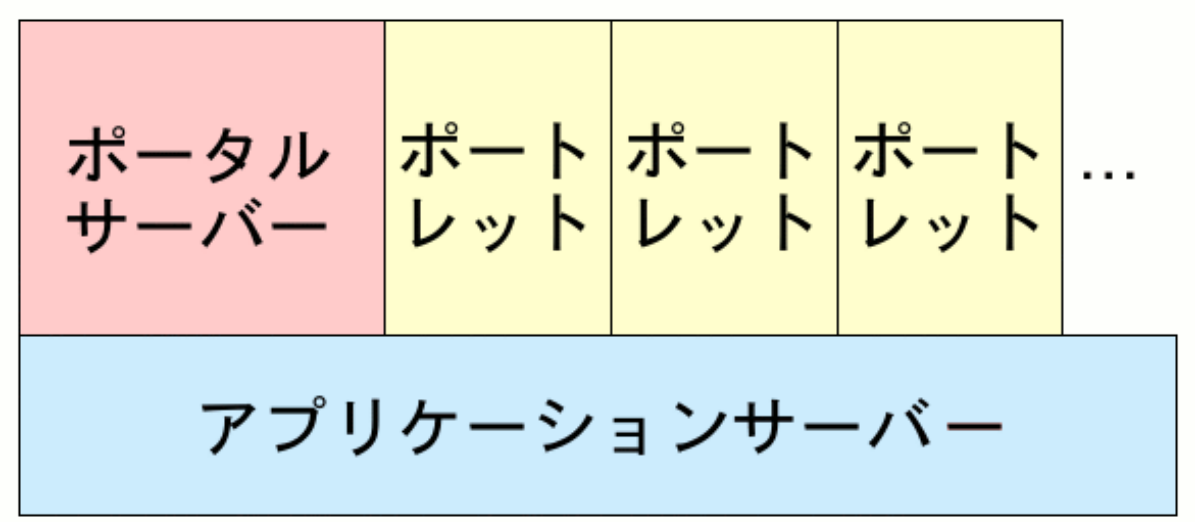
- ポートレットの実行環境を定義
- コンテナとポートレット間のAPIを定義
- ポートレットのデータを保存する機能を定義
- サーブレットやJSPを呼び出し
- パッケージング方法を定義

JSR 168 の登場によって、ベンダー間を越えた UI コンポーネントのやりとりが可能になります。

2.2 アーキテクチャ

アーキテクチャについて

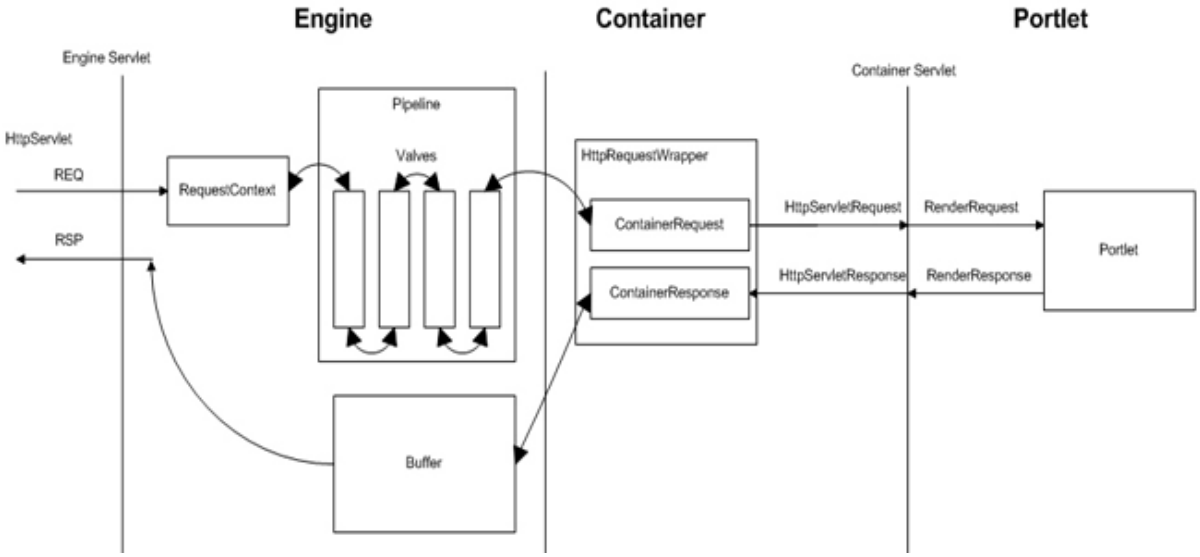
ポータルサーバーは一つのウェブアプリケーションです。 そのため、ポータルサーバーを動作させるためにはアプリケーションサーバーが必要になります。



ポートレットも同様に一つのウェブアプリケーションです。 ポートレットの配備はポータルサーバーに対して配備を実行しますが、最終的にはアプリケーションサーバー上に配備されることになります。 クライアントからのアクセスは、通常のウェブアプリケーションと同様にポータルサーバーにアクセスされた後に、ポータルサーバーからポートレットへのクロスコンテキストのアクセスにより、ポートレットのコンテンツを取得し、その結果を集約して、クライアントへ出力します。

ポータル内の処理

PALポータルは、コンポーネントのパイプライン処理に基づいて、ユーザーからのリクエストを処理します。



クライアントからのアクセスがあると、ポータルサーバー内ではそのリクエストを RequestContext というインスタンスとして扱います。 その RequestContext を Spring で設定したコンポーネントバルブに順に渡して処理していきます。 その中でポートレットコンテナを呼び出し、ポートレットを処理します。 処理の終了後、その結果を集約して、クライアントへ出力します。

構成コンポーネント

PALポータルのアーキテクチャは Martin Fowler によって以下のように定義されたコンポーネントアーキテクチャに基づいて構築されています。 コンポーネントという言葉は、変更なしに、コンポーネントの作者の管理から外れたアプリケーションとして使われることを意図したソフトウェア群を意味しています。 「変更なしに」という言葉は、ユーザーがコンポーネントの作者によって許された方法でコンポーネントを拡張し、その動作を変化させることはあるかもしれないけれども、アプリケーションを使う場合に、コンポーネントのソースコードを変えることなくアプリケーションを使うことを意味します。

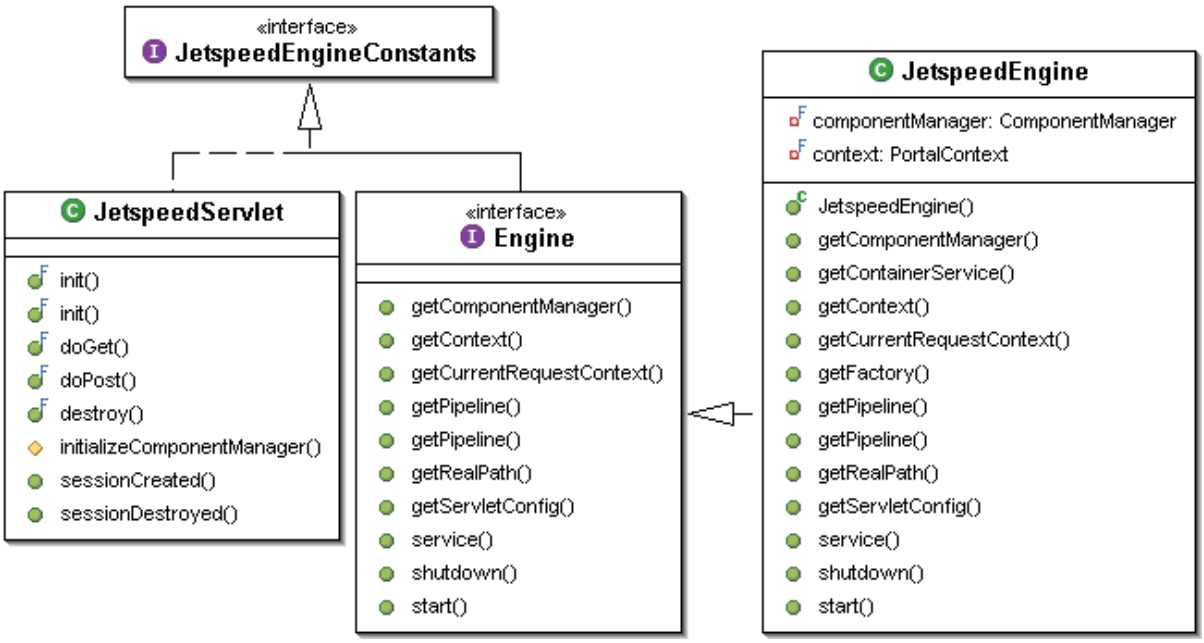
PALポータルはプログラミングのデザインパターンと構造上のモデルとして、依存性注入を使います。 これは、公開されているインターフェースを通して抽象化のレベルを構築するためであり、そしてコンポーネントの実装における依存性をなくすためです。 この構造はコンポーネントがそれ自身とリンクしたり、お互いにリンクする構造でなく、コンポーネントと結合します。 依存性注入はオブジェクトの作成やオブジェクトのリンクの機能をオブジェクト自身から削除し、ファクトリへ移したパターンです。 それゆえ、依存性注入は明らかにオブジェクトの作成とリンクの制御を反転させており、Inversion of Controls(制御の反転, IOC) の形であると考えることができます。

コンポーネントフレームワーク

PALポータルはデフォルトのコンポーネントフレームワークとして Spring フレームワークを利用しています。 しかし、PALポータルはコンポーネントフレームワークを、たとえば Pico のような代替可能なコンポーネントフレームワークに容易に置き換えることが可能な

構造をしています。

PALポータルのコンポーネントフレームワークの組み立ては JetspeedServlet 内に実装されています。



JetspeedServlet は起動時のロードのためにポータルアプリケーションの web.xml で設定されます。 initializeComponentManager メソッドは与えられたコンポーネントフレームワークのコンポーネント群をロードします。 initializeComponentManager のデフォルトの実装はコンポーネントフレームワークとして Spring フレームワーク をサポートします。 そして Spring エンジンを初期化するために webapps/palportal/WEB-INF/assembly 以下の xml ファイルを解釈します。 JetspeedEngine は適切なコンポーネントマネージャを使って構成されます。

```
engine = new JetspeedEngine(properties, applicationRoot, config,
    initializeComponentManager(config, applicationRoot, properties));
```

他のコンポーネントフレームワークをサポートするためには、開発者は initializeComponentManager の実装をオーバーライドしてください。

コアコンポーネント

クライアント識別 - アーチファクト ID: jetspeed-capability

Capabilities	Capabilities コンポーネントはクライアントを、サポートされる MIME タイプやメディアタイプにマップします。このコンポーネントは CapabilityMap を生成します。CapabilityMap はポータルエンジンを通して、対象となるクライアント用にポータルコンテンツを表示するために利用されます。

コンポーネットマネージャ - アーチファクト ID: jetspeed-cm

ComponentManager	ComponentManager は、使用するコンポーネントフレームワークのトップに位置する抽象的な標準レイヤーを提供します。PALポータルのデフォルトの ComponentManager の実装は SpringComponentManager です。

配備ツール - アーチファクト ID: jetspeed-deploy-tools

JetspeedDeploy	JetspeedDeploy は PALポータルに配備される前にポートレットアプリケーションの準備をします。
DeploymentManager	DeploymentManager は配備される新しいポータル資産（ポートレットやデコレータ）を待機します。

ポータル - アーチファクト ID: jetspeed-portal

Pipeline	Pipeline は PALポータルのリクエストで最小単位の動作である Valve を統合します。

プリファレンス - アーチファクト ID: jetspeed-prefs

PreferencesProvider	PreferencesProviderは PALポータルの java.util.Preferences API の実装です。

RDBMS - アーチファクト ID: jetspeed-rdbms

ConnectionRepositoryEntry	ConnectionRepositoryEntry は PALポータルの java.util.Preferences API の実装です。
InitablePersistenceBrokerDaoSupport	InitablePersistenceBrokerDaoSupport は PALポータルにおいてデータアクセスと永続性のサポートを提供します。

セキュリティ - アーチファクト ID: jetspeed-security

DefaultLoginModule RdbmsPolicy	PALポータルにおける JAAS サービス のデフォルトの実装。PALポータルはポータルエンジンにセキュリティの機能を公開するための標準のセキュリティフレームワークとして JAAS を利用します。JAAS サービスは PALポータルのセキュリティ SPI を通して特定の実装が提供されるために PALポータルの粗い単位のサービスを利用します。
UserManager RoleManager GroupManager PermissionManager	PALポータルのセキュリティ管理 API を公開する粗い単位のセキュリティコンポーネント
UserSecurityHandler CredentialHandler GroupSecurityHandler RoleSecurityHandler SecurityMappingHandler	特定の実装を PALポータルのセキュリティエンジンに公開する細かい単位のセキュリティ SPI コンポーネント。この仕組みは高レベルのセキュリティサービスに影響することなく、複数のセキュリティ実装をサポートするために柔軟なフレームワークを提供します。

統計 - アーチファクト ID: jetspeed-statistics

PortalStatistics	PortalStatistics はデータのコレクションとデータの検索のための PALポータルのデータコレクション API です。
BatchedStatistics	BatchedStatistics は与えられた期間の統計データのバッチコレクションに対する処理を行います。
AggregateStatistics StatisticsQueryCriteria	AggregateStatistics は、StatisticsQueryCriteria が提供する特定のイベントと検索基準に対する集約ポータルデータを提供するために、PortalStatistics コンポーネントとやりとりを行います。

2.3 管理ツール

管理ツールについて

PALポータルのインストール後、pal-admin ポートレットおよび pal-wcm ポートレットの 2 つが初回起動時に配備されます。pal-admin ポートレットでは、ユーザー・ロール・グループやサイトを管理するポートレット群を提供します。pal-wcm ポートレットでは、ポートレット内にコンテンツを表示できるポートレットを提供します。

pal-admin ポートレット

pal-admin ポートレットは以下のポートレットを含みます。

- ユーザー管理: ユーザー・ロール・グループの管理ツールです。
- ポートレット管理: ポータル内のポートレットを管理するツールです。
- ユーザー登録: 任意のユーザーからのユーザー登録を受け付けるポートレットです。
- パスワード再発行: 任意のユーザーからパスワード再発行要求を受け付けるポートレットです。
- サイトエディター: ポータル内のサイトを管理するツールです。
- パスワード変更: ユーザーのパスワード変更要求を受け付けるポートレットです。
- パーミッションエディター: リソースのパーミッションを編集するツールです。(拡張機能になります)
- セキュリティ制約エディター: セキュリティ制約を設定するツールです。(拡張機能になります)
- プロファイリングルールエディター: プロファイリングルールを設定するツールです。(拡張機能になります)

利用方法については、「管理ガイド」を参照してください。

pal-wcm ポートレット

pal-wcm ポートレットは以下のポートレットを含みます。

- WCMビューアー: 登録されているコンテンツを表示するポートレットです。
- WCMエディター: 表示するコンテンツを編集するポートレットです。

利用方法については、「管理ガイド」を参照してください。

3.1 インストールについて

インストールの概要

本章ではインストール関連の情報を提供します。

システム要件

最小システム要件

- JDK(TM) 5 以上
- 1 GHz 以上の 32 ビットまたは 64 ビットプロセッサを推奨
- 512 MB 以上のメモリー
- 100 MB 以上のディスクスペース

サポートする OS

PALポータルは 100% Pure Java(TM) のポータルサーバーです。 ですので、Linux、Windows、Unix、Mac OS X など Java Virtual Machine(JVM(TM)) が動作するオペレーティングシステムで動作可能です。

アプリケーションサーバー

PALポータルは Java のウェブアプリケーションが動作可能なアプリケーションサーバー上で稼働します。 様々なアプリケーションさーバーで動作可能ですが、以下のアプリケーションサーバーで動作確認をしています。

- Tomcat 5.5
- JBoss AS 4.0/4.2
- WebLogic Server 9.2/10
- GlassFish 2

データベース

PALポータルはポータル内のデータを保存するためのデータベースを必要とします。 様々なデータベースをサポートしていますが、以下のデータベースで動作確認をしています。

- Derby
- Oracle
- MySQL
- Postgres
- HSQL

データベースの接続に関しては、Apache DB Project の OJB（ http://db.apache.org/ojb/ ）を利用しています。

インストール方法

PALポータルでは、新規インストールおよびアップグレードインストールを提供しています。 また、新規インストールでは、Tomcat 5.5 のインストールを省略するなどのオプションを提供しています。 詳しいインストール方法に関しては、「インストールガイド」を参照してください。

インストール内容

PALポータルは、Apache Portals の Jetspeed 2（ http://portals.apache.org/jetspeed-2/ ）ベースのポータルサーバーです。 ですので、標準の新規インストールでは、Tomcat 5.5 と共に PALポータルとしてカスタマイズされた Jetspeed 2 がインストールされます。 また、PALポータルでは、管理ツールとして pal-admin ポートレットと pal-wcm ポートレットが PAL ポータルの初回起動時に配備されます。

JSR 168 準拠のポートレットを追加で配備したい場合は、PALポータルのインストール後に「ポートレット管理」画面から配備するか、webapps/palportal/WEB-INF/deploy にポートレットの war ファイルを配置してください。

3.2 ディレクトリ構成の概要

ディレクトリ構成について

ここでは、ディレクトリの構成について説明します。主に以下のディレクトリについての説明となります。 これらのうちのいくつかのものについては、別なページで詳しい説明がされます。

- webapps/palportal
  - decorations
    - layout
    - portlet
  - WEB-INF
    - assembly
    - deploy
    - lib
    - logs
    - pages
    - templates

webapps/palportal/decorations

画面デザインに関するファイルがこのディレクトリ以下に置かれます。 PALポータルでは、デザインをデコレータと呼ばれる単位で扱います。

layout

レイアウトデコレータが配置されます。 レイアウトデコレータは画面の周辺部分のデザインに関するファイル群になります。

ディレクトリ名が、レイアウトデコレータ名に対応しています。たとえば、simple というデコレータ名に対応するファイルは layout/simple というディレクトリに置かれています。

画面の周辺部分のデザインを追加したい場合、この layout ディレクトリに置くことで追加が可能です。詳しい設定については、「レイアウトデコレータ」を参照して下さい。

portlet

ポートレットデコレータが配置されます。 ポートレットデコレータはポートレットのデザインに関するファイル群になります。

ディレクトリ名が、ポートレットデコレータ名に対応しています。たとえば、simple というデコレータ名に対応するファイルは portlet/simple というディレクトリに置かれています。

ポートレットのデザインを追加したい場合、この portlet ディレクトリに置くことで追加が可能です。詳しい設定については、「ポートレットデコレータ」を参照して下さい。

webapps/palportal/WEB-INF

assembly

PALポータルはデフォルトのコンポーネントフレームワークとして Spring フレームワークを使用しています。 Spring エンジンを初期化するために assembly 以下の xml ファイルが読み込まれます。

deploy

deploy ディレクトリ以下にポートレットの war ファイルを置くことで、ポートレットを配備することができます。 ポートレットの war ファイルは自動配備されます。

lib

PALポータルが使用する jar ファイルが置かれます。

logs

デフォルトの設定ではこのディレクトリにログが出力されます。ログの出力のレベルや出力先ディレクトリは /WEB-INF/conf/Log4j.properties を変更することで設定できます。

pages

各ユーザのページ情報が置かれます。

templates

ログインページや、エラーページなど、共通して使用されるファイルがこのディレクトリ以下に置かれます。

4.1 基本設定の概要

設定概要

ポータルの設定に関する情報は、webapps/palportal/WEB-INF 以下の様々なファイルに保存されています。 ポータルの機能をカスタマイズしていくためには、対象のファイルの設定値を変更することで、変更することができます。

基本的な設定情報は、webapps/palportal/WEB-INF/conf のプロパティファイル内に定義されています。 また、PALポータルは、コンポーネントアーキテクチャにより構築されているので、コンポーネントの構成データに関しては、Spring の設定ファイルとして記述されています。それらのファイルは、webapps/palportal/WEB-INF/assembly に保存されています。

4.2 基本設定情報

基本設定情報について

PALポータルの基本的な設定情報は、webapps/palportal/WEB-INF/conf に保存されています。ベースとなる設定情報は、jetspeed.properties に保存されています。 これらの設定値を変更したい場合は、override.properties に定義して、jetspeed.properties の設定情報を上書きします。

4.3 コンポーネントの設定

コンポーネントの設定について

PALポータルは、コンポーネントアーキテクチャを採用しており、コンポーネントの管理には Spring フレームワークを利用しています。 そのため、コンポーネントを管理する設定ファイルは、Spring の設定ファイルに基づいて記述されています。 それらの設定ファイルは、webapps/palportal/WEB-INF/assembly に保存されています。

コンポーネント一覧

インストールされるコンポーネント設定ファイル一覧は以下の通りです。

設定ファイル名	説明
administration.xml	ポータルの管理用コンポーネントを定義しています
aggregation.xml	コンテンツ集約を処理するコンポーネントを定義しています
ajax-layout.xml	AJAX API で使用されるコンポーネントを定義しています
ajax.xml	AJAX サービスを提供するコンポーネントを定義しています
cache.xml	キャッシュを管理するコンポーネントを定義しています
capabilities.xml	クライアントを識別するコンポーネントを定義しています
cluster-node.xml	クラスタノードを管理するコンポーネントを定義しています
deployment.xml	配備処理をするコンポーネントを定義しています
headtag.xml	headタグで出力するタグを管理するコンポーネントを定義しています
hierarchical-principal-names.xml	階層的なロール、グループ、ユーザーを制御します
importer-page-manager.xml	PSMLデータのインポートを管理するコンポーネントを定義しています
interceptors.xml	インタセプタを設定するコンポーネントを定義しています
jetspeed-base.xml	ポータルで利用するコンポーネントを定義しています
jetspeed-production.xml	jetspeed-production.properties を利用するときに使うコンポーネントを定義しています

設定ファイル名	説明
jetspeed-services.xml	ポータル内で利用するサービスをまとめたコンポーネントを定義しています
jetspeed-spring.xml	ポータル内で必要なコンポーネントを定義しています
multiple-action.xml	複数の Ajax アクションを利用するコンポーネントを定義しています
page-manager.xml	ページ情報を管理するコンポーネントを定義しています
pipelines.xml	パイプラインとバルブのコンポーネントを定義しています
pluto-factories.xml	Pluto を管理するコンポーネントを定義しています
portal-url-generation.xml	ポータル URL を管理するコンポーネントを定義しています
prefs.xml	プリファレンスを管理するコンポーネントを定義しています
profiler.xml	プロファイリングルールを管理するコンポーネントを定義しています
registry.xml	ポートレットの登録情報を管理するコンポーネントを定義しています
request-context-objects.xml	RequestContext で利用するオブジェクトを管理するコンポーネントを定義しています
search.xml	検索コンポーネントを定義しています
security-atn.xml	ログインモジュールを管理するコンポーネントを定義しています
security-atz.xml	JAAS 管理情報を管理するコンポーネントを定義しています
security-managers.xml	セキュリティ管理を処理するコンポーネントを定義しています
security-providers.xml	ユーザー認証を管理するコンポーネントを定義しています
security-spi-atn.xml	ユーザー認証を制御するコンポーネントを定義しています
security-spi-atz.xml	アクセスを制御するコンポーネントを定義しています
security-spi.xml	セキュリティアクセスを管理するコンポーネントを定義しています
serializer.xml	データベース内のデータをインポート・エクスポートするコンポーネントを定義しています
sso.xml	SSOを管理するコンポーネントを定義しています
statistics.xml	統計情報を管理するコンポーネントを定義しています
theme-engine.xml	デコレータを制御するコンポーネントを定義しています
transaction.xml	トランザクションを管理するコンポーネントを定義しています
userinfo.xml	ユーザー属性を管理するコンポーネントを定義しています
wps.xml	Websphere で必要なコンポーネントを定義しています

4.4 ポートの変更

ポートの変更について

PALポータルは、標準で Tomcat 5.5 をバンドルして利用しています。 そのため、PALポータルへのアクセスポートを変更するためには、Tomcat のリクエスト受信ポートを変更する必要があります。

PALポータルを Tomcat 以外のアプリケーションサーバーで動作させる場合は、アクセスポートの変更方法はそのアプリケーションサーバーの設定方法に従います。

受信ポートの変更

デフォルトの受信ポートは、8080 を利用しています。 受信ポートを変更する場合は、conf/server.xml の以下の 8080 を利用したいポート番号に変更します。

```
<Connector port="8080" maxHttpHeaderSize="8192"
           emptySessionPath="true"
           maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
           enableLookups="false" redirectPort="8443" acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true" />
```

値を変更後、設定を保存し、ポータルを再起動後に有効になります。

停止ポートの変更

PALポータルを同じサーバー上などで同時に複数上げたい場合、受信ポート以外に停止ポートも変更する必要があります。 停止ポートを変更する場合は、conf/server.xml の以下の 8005 を利用したいポート番号に変更します。

```
<Server port="8005" shutdown="SHUTDOWN">
```

値を変更後、設定を保存し、ポータルを再起動後に有効になります。

4.5 プロキシの設定

プロキシの設定について

PALポータルをイントラネット環境に構築し、ポートレットなどでインターネットにアクセスする必要がある場合は、プロキシ情報の設定が必要になります。 通常、JAVA_OPTS 環境変数に -Dhttp.proxyHost=<YOUR_PROXY_HOST> -Dhttp.proxyPort=<YOUR_PROXY_PORT> を設定して、PALポータルを起動すると有効になります。

4.6 コンテキストの変更

コンテキストの変更について

PALポータルは、標準で Tomcat 5.5 をインストールして、palportal コンテキストにインストールします。 インストール後、コンテキストを変更したい場合は、本節の設定で変更することができます。

コンテキストの変更

PALポータルの停止

PALポータルを起動している場合は、停止します。

コンテキストの変更

本手続きでは http://localhost:8080/palportal/ でアクセスしていたものを http://localhost:8080/ に変更することを考えます。 ルートコンテキスト(ROOT) ではない場合は、利用したいコンテキスト名に置き換えて読んでください。

既存の ROOT を削除します。

```
$ rm -r webapps/ROOT
```

palportal を ROOT に名前変更します。

```
$ mv webapps/palportal  webapps/ROOT
```

conf 以下にある palportal.xml も ROOT.xml に名前変更します。 既に Tomcat を起動している場合は、work などにキャッシュが存在する場合があるので、削除します。

```
$ rm -rf work/*
```

PALポータルの起動

PALポータルを起動します。 上記の設定により http://localhost:8080/ でアクセスすることができるようになります。

4.7 ログの設定

ログの設定について

PALポータルは、Log4j を利用して、ログの制御を行っています。 ログの設定ファイルは、webapps/palportal/WEB-INF/conf/Log4j.properties に記述されています。 そこで指定されている値を変更することで、ログファイルの保存先を変更したり、出力するログレベルを変更することができます。

設定ファイル

Log4j.properties のデフォルトの設定は以下の通りです(一部略)。

```
# ----------------------------------------------------------------------
#
# Logging Configuration
#
# ----------------------------------------------------------------------

# If we don't know the logging facility, put it into the jetspeed.log
log4j.rootLogger = ERROR, jetspeed

# Jetspeed goes into Jetspeed Log
log4j.category.org.apache.jetspeed = ERROR, jetspeed
log4j.additivity.org.apache.jetspeed = false


log4j.category.org.apache.jetspeed.tools = ERROR, jetspeed
log4j.additivity.org.apache.jetspeed.tools = false


# Velocity Logfile
log4j.category.velocity = ERROR, velocity
log4j.additivity.velocity = false

# Deployment Category
log4j.category.deployment = ERROR, DEPLOYMENT
log4j.additivity.deployment = false
```

```
#############################################################################
#
# Logfile definitions
#
#############################################################################

# jetspeed.log
log4j.appender.jetspeed = org.apache.log4j.RollingFileAppender
log4j.appender.jetspeed.File = ${applicationRoot}/WEB-INF/logs/jetspeed.log
log4j.appender.jetspeed.layout = org.apache.log4j.PatternLayout
log4j.appender.jetspeed.layout.ConversionPattern = %d [%t] %-5p %c - %m%n
log4j.appender.jetspeed.Append = true
log4j.appender.jetspeed.MaxFileSize = 10MB
log4j.appender.jetspeed.MaxBackupIndex = 5

# deployment.log
log4j.appender.DEPLOYMENT = org.apache.log4j.RollingFileAppender
log4j.appender.DEPLOYMENT.File = ${applicationRoot}/WEB-INF/logs/deployment.log
log4j.appender.DEPLOYMENT.layout = org.apache.log4j.PatternLayout
log4j.appender.DEPLOYMENT.layout.ConversionPattern = %d [%t] %-5p %c - %m%n
log4j.appender.DEPLOYMENT.Append = true
log4j.appender.DEPLOYMENT.MaxFileSize = 10MB
log4j.appender.DEPLOYMENT.MaxBackupIndex = 5

# Velocity gets configured to write its output onto the velocity category.
log4j.appender.velocity = org.apache.log4j.RollingFileAppender
log4j.appender.velocity.File = ${applicationRoot}/WEB-INF/logs/velocity.log
log4j.appender.velocity.layout = org.apache.log4j.PatternLayout
log4j.appender.velocity.layout.ConversionPattern = %d [%t] %-5p %c - %m%n
log4j.appender.velocity.Append = true
log4j.appender.velocity.MaxFileSize = 10MB
log4j.appender.velocity.MaxBackupIndex = 5
```

デフォルトのログレベルは ERROR に設定されています。 出力されるログファイルの保存先は、webapps/palportal/WEB-INF/logs に保存されます。 ログファイルの内容は、PALポータルを再起動しても保存されたままで、ログを追記します。 各ログファイルのファイルサイズは、10MB まで保存し、それ以上のサイズになった場合は〈ファイル名〉.〈数字〉 として保存され、最大 5 つまで保持します。 それ以上になった場合は古いものから削除されます。

jetspeed.log

jetspeed.log は、PALポータルの処理中に出力するログが保存されます。

deployment.log

deployment.log は、PALポータルにポートレットを配備の処理中に出力するログが保存されます。

velocity.log

velocity.log は、レイアウトデコレータやポートレットデコレータなどで Velocity の処理中に出力するログが保存されます。

ログレベルの変更

Log4j.properties で、対象ログの log4j.category.* プロパティで設定します。 デフォルトでは ERROR が設定されているので、この値を DEBUG などに変更すると、ログレベルがデバッグになります。 利用可能なログレベルは以下の通りです。

- DEBUG: アプリケーションのデバッグ用のログを出力する。以下のログレベルで指定したものは出力される。
- INFO: アプリケーションの必要な情報を出力する用などのログを出力する。以下のログレベルで指定したものは出力される。
- WARN: 警告メッセージなどの情報を出力するログレベル。以下のログレベルで指定したものは出力される。
- ERROR: エラーメッセージなどの情報を出力するログレベル。以下のログレベルで指定したものは出力される。
- FATAL: 致命的なエラーメッセージを表示するログレベル。

値を変更後ポータルを再起動することで変更が有効になります。

保存先の変更

各ログファイルの保存先を変更することができます。 Log4j.properties で、対象ログの log4j.appender.*.File プロパティで設定します。 ${applicationRoot} は PALポータルのルートディレクトリである webapps/palportal を示しています。 値を変更後ポータルを再起動することで変更が有効になります。

4.8 メールサーバーの設定

メールサーバーの設定について

PALポータルは、ゲストアクセスからのユーザー登録処理やパスワード再発行時にメールサーバーが必要になります。 メールサーバーが正しく設定されていない場合は、それらの機能を利用することができません。

メールサーバーの設定は管理コンポーネントにより設定されています。 コンポーネントの設定ファイルでメールサーバーの情報を変更することができます。

メールサーバーの変更

通常、インストール時の設定において、メールサーバーの値が更新されます。 インストール後にメールサーバーの情報を変更したい場合は、webapps/palportal/WEB-INF/assembly 内の administration.xml の以下の値を変更します。

```
<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
  <property name="host"><value>localhost</value></property>
  <property name="username"><value></value></property>
  <property name="password"><value></value></property>
    <property name="javaMailProperties">
        <props>
            <prop key="mail.smtp.auth">false</prop>
        </props>
    </property>
</bean>
```

host がメールサーバー名、username と password はメールサーバーのユーザー名とパスワードで、認証が必要な場合は mail.smtp.auth を true にして有効にします。 値を更新した場合は、PALポータルを再起動後に有効になります。

4.9 セッション共有

セッション共有について

PALポータルも各ポートレットもアプリケーションサーバー上では 1 つのウェブアプリケーションとして動作しています。 ポートレットへの呼び出しは、ポータルを経由するアクセスであるので、クロスコンテキストなアクセスになります。 そのため、通常の Tomcat では、ポートレットでセッションを取得して、アプリケーションスコープで値を保存しても、そのポートレットのコンテキストにあるサーブレットにアクセスしてセッションからその値を取得できません。

PALポータルでは、ポートレットのコンテキストにある、ポートレットのセッションとサーブレットのセッションにある値を共有できるように、デフォルトで Tomcat に対して、emptySessionPath を有効にしています。 もし、PALポータルにバンドルされた Tomcat を利用しない場合は、その値を有効にする必要があります。

emptySessionPath の設定

PALポータルにバンドルされた Tomcat では、emptySessionPath を true で設定されています。 emptySessionPath を設定する場合は、conf/server.xml の以下の箇所に emptySessionPath を追加します。

```
<Connector port="8080" maxHttpHeaderSize="8192"
           emptySessionPath="true"
           maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
           enableLookups="false" redirectPort="8443" acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true" />
```

値を変更後、設定を保存し、ポータルを再起動後に有効になります。

4.10 SSL リダイレクト

SSL リダイレクトについて

特定のページアクセスした場合に HTTPS のページにリダイレクトをさせたり、また逆にあるページにアクセスした場合に HTTP を表示したい場合があるかと思います。 そのような場合に対応するため、PALポータルでは SSL リダイレクトバルブを提供しています。 SSL リダイレクトバルブをパイプラインに追加することで、ページ単位で SSL ページにするかどうかの指定ができます。

設定方法

webapps/palportal/WEB-INF/assembly/pipelines.xml に
jp.sf.pal.portal.redirect.impl.SSLRedirectValveImpl を追加します。

```
...
  <bean id="sslRedirectValve"
        class="jp.sf.pal.portal.redirect.impl.SSLRedirectValveImpl">
    <constructor-arg>
      <value>redirect.to.ssl.page</value>
    </constructor-arg>
    <constructor-arg>
      <value>80</value>
    </constructor-arg>
    <constructor-arg>
      <value>443</value>
    </constructor-arg>
  </bean>

  <bean id="jetspeed-pipeline"
        class="org.apache.jetspeed.pipeline.JetspeedPipeline"
        init-method="initialize"
  >
   <constructor-arg>
      <value>JetspeedPipeline</value>
   </constructor-arg>
   <constructor-arg>
    <list>
      <ref bean="capabilityValve"/>
```

```
      <ref bean="portalURLValve"/>
      <ref bean="securityValve"/>
      <ref bean="localizationValve"/>
      <ref bean="passwordCredentialValve"/>
      <ref bean="loginValidationValve"/>
      <ref bean="profilerValve"/>
      <ref bean="sslRedirectValve"/>
...
```

sslRedirectValve を bean 要素を追加して、jetspeed-pipeline の profilerValve の後に sslRedirectValve の ref 要素を追加します。SSLRedirectValveImplのコンストラクタ引数は、一番目の引数がページで指定するキーの名前、2番目が非SSLページで利用するポート、３番目がSSLページで利用するポートです。 設定を保存し、ポータルを再起動後に有効になります。

設定後、サイトエディターで各ページ(PSML)の直下にあるレイアウトのプロパティに redirect.to.ssl.page のキーで true または false を設定します。true の場合は SSL ページにリダイレクとして、false の場合は非 SSL ページにリダイレクトします。 redirect.to.ssl.pageを指定しない場合は、指定された URL をそのまま表示します。 サイトエディターの使い方については、「管理ガイド」を参照してください。

4.11 非同期コンテンツ集約

非同期コンテンツ集約について

ポートレットも一つのウェブアプリケーションです。 ポータルページ内に複数のポートレットを配置した場合、そのページにアクセスすると、ポートレットが配置してある数だけウェブアプリケーションにアクセスすることと同じことになります。

PALポータルでは、デフォルトでポータルページ内にあるポートレットに対して、1 つのポータルページへのアクセスでは複数のポートレットをマルチスレッドでアクセスします。マルチスレッドでアクセスしたくない場合は、パラメータを変更することで同期的にポートレットを呼び出すことも可能です。

設定方法

設定ファイルの変更

webapps/palportal/WEB-INF/assembly/pipelines.xml で aggregatorValve に引数を変更できます。

```
...
  <bean id="aggregatorValve"
        class="org.apache.jetspeed.aggregator.AggregatorValve"
        init-method="initialize"
  >
   <constructor-arg>
       <ref bean="org.apache.jetspeed.aggregator.AsyncPageAggregator"/>
   </constructor-arg>
  </bean>
...
```

org.apache.jetspeed.aggregator.AsyncPageAggregator は非同期にコンテンツを集約します。 同期的にコンテンツ集約をしたい場合は、
org.apache.jetspeed.aggregator.PageAggregator を利用してください。

タイムアウトの設定

pipelines.xmlの設定後、非同期で処理したいポートレットに対して、タイムアウト値を設定する必要があります。全てのポートレットをマルチスレッドで呼び出す場合は以下の変更を webapps/palportal/WEB-INF/assembly/aggregation.xml に加えます。

```
...
    <bean id="org.apache.jetspeed.aggregator.PortletTrackingManager"
class="org.apache.jetspeed.aggregator.impl.PortletTrackingManagerImpl">
...
      <!-- Default portlet timeout in milliseconds:
       Zero means no portlet timeout option by default.
       -->
       <constructor-arg index='1'>
           <value>5000</value>
       </constructor-arg>
...
    </bean>
...
```

上記の設定の場合、5000ms のタイムアウト値が設定されます。

各ポートレットで個別に設定する場合には、対象のポートレットアプリケーションの jetspeed-portlet.xml で以下のような設定をします。

```
...
   <portlet>
       <portlet-name>PALWcmViewer</portlet-name>
       <js:metadata name="timeout">5000</js:metadata>
   </portlet>
...
```

5.1 データベースについて

データベース設定の概要

PALポータルは、ポータルに必要なデータを設定されたデータベースに保存します。 PALポータルからのデータベースへのアクセスは、JNDI を用いてデータソースを取得して行います。 O/R マッパーには、Apache DB の OJB を利用しています。 PALポータルは、特定のデータベースに依存していないので、Oracle や MySQL など様々なデータベースを利用することが可能です。

5.2 設定

設定について

PALポータルは、データベースのアクセスにはアプリケーションサーバーからデータソースを取得して処理します。 デフォルトの構成では、インストール時にセットアップされる Tomcat にデータソースの設定がされています。 インストール後にデータベースを変更したい場合は、Tomcat のデータソースの設定を変更します。

Tomcat のデータソースの設定は、conf/Catalina/localhost/palportal.xml に記述されています。 Derby をデータベースとして利用している場合は、以下のような設定が記述されています。

```
<Resource name="jdbc/jetspeed" auth="Container"
        factory="org.apache.commons.dbcp.BasicDataSourceFactory"
        type="javax.sql.DataSource" username="" password=""
        driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
        url="jdbc:derby:/tmp/Portal/database/derby/productiondb;create=true"
        maxActive="100" maxIdle="30" maxWait="10000"/>
```

設定方法

PALポータルは、インストール時にデータベースを構成するスクリプトもインストールします。 ここでは、インストール時に Derby を利用するデータベースとして選択して、インストール後に MySQL に変更する方法を説明します。 今回は、データの移行などはせず、新規にデータベースを作りなおすこととします(データの移行が必要な場合は各データベースのドキュメントを参考にして、DML などを作成してください)。 データベース構成スクリプトの実行には、Apache Ant が必要になります。

PALポータルの停止

PALポータルが起動している場合は、停止してください。

jar の配置

データベースのアクセスに必要な JDBC ライブラリの jar ファイルを shared/lib に配置します。

```
$ cp <somewhere>/mysql-connector-java-5.0.4.jar shared/lib
```

設定ファイルの変更

データベースの構成スクリプトは、database ディレクトリにあります。

```
$ cd database
```

データベースの構成スクリプトの設定ファイル database.properties を修正します。

```
; Jetspeed Enterprise Portal 2.1.1 Database setup configuration

; db.type supported values: db2, derby, mssql, mysql, oracle, postgres, sapdb
db.type=mysql

: for db.type other than derby, the properties below need to be specified
db.username=<DB のユーザー名>
db.password=<DB のパスワード>
jdbc.url=jdbc:mysql://localhost/<DB 名>?useUnicode=true\&amp;characterEncoding=UTF-8
jdbc.driver.class=com.mysql.jdbc.Driver

; boolean flag (true/false) indicating if psml is to be imported
; in the database or will be read from file system
dbImportPsml = true
```

ご利用の環境に応じて、DB 名、DB のユーザー名とパスワードを設定してください。
dbImportPsml を true にすることで、PSML ファイルをデータベースに格納して利用することができます。 データベースに格納した場合は、webapps/palportal/WEB-INF/pages の PSML ファイルは利用されなくなります。

実行

上記を設定後、Ant を実行します。

```
$ ant
```

BUILD SUCCESSFUL と表示されれば、データベースの設定は完了です。

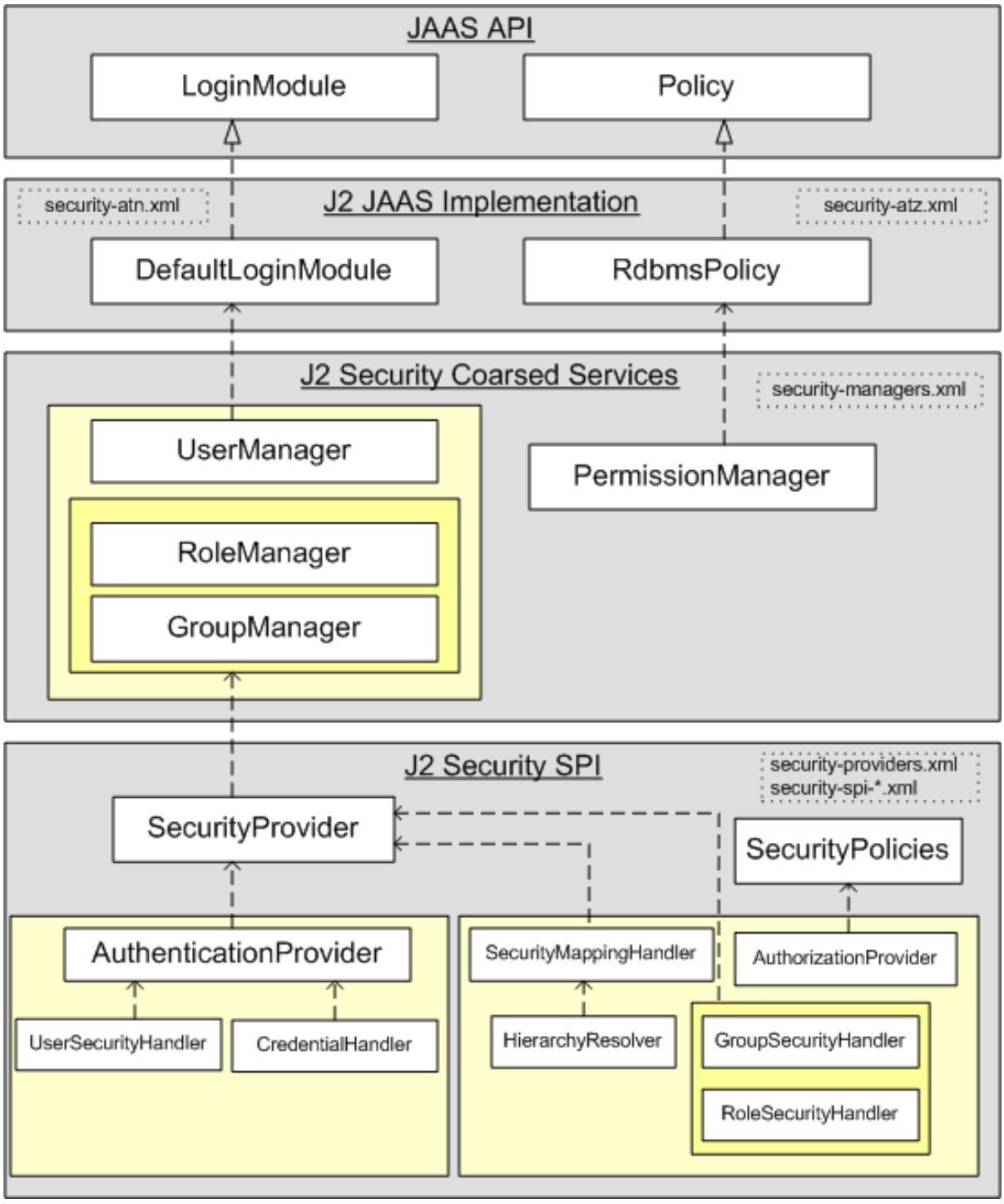
conf/Catalina/localhost/palportal.xml も新しく設定したデータベースを利用するように更新されています。 以上の設定で、新しいデータベースを用いて、PALポータルを起動することができます。

6.1 セキュリティの概要

セキュリティの概要

PALポータルは、LoginModule と Policy の実装により、J2EE の認証・承認サービスを利用しています。 認証はユーザー ID により確立して、すべてのユーザー主体を持つ Subject を生成します。 ポータルコンテキスト内で、生成された Subject は org.apache.jetspeed.security.SecurityValve の実装でセッションへ格納されます。 Subject の主体は、適切なリソースへのユーザーアクセスを承認するためなどに使われます。 それにより、AccessController でユーザーのパーミッションを確認することで、JAAS 承認を利用しています。

以下にセキュリティアーキテクチャの概要図を示します。

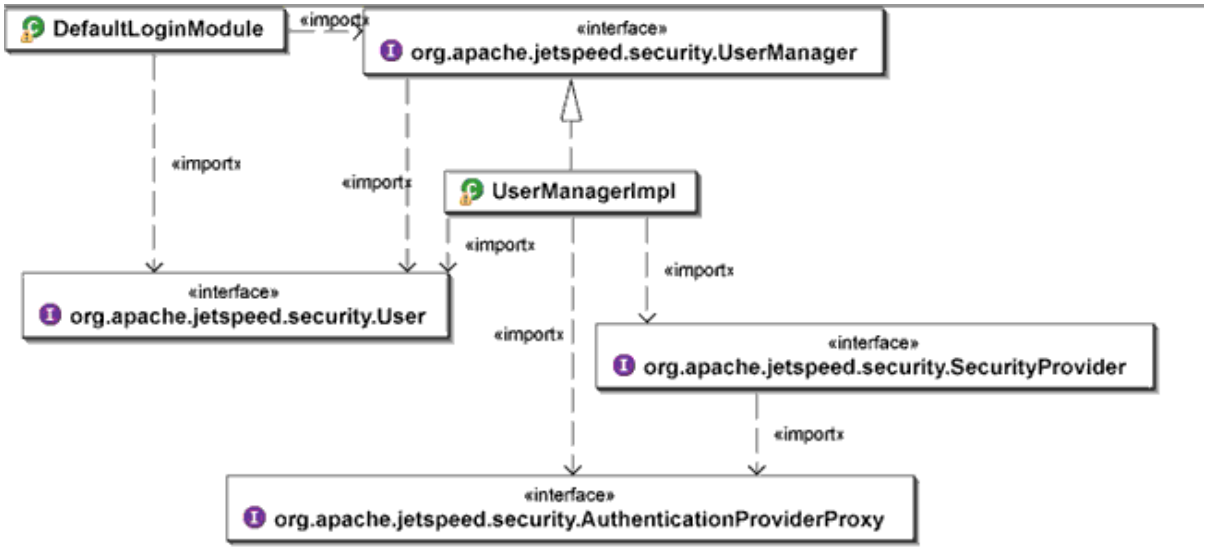


各コンポーネントはコンポーネント設定ファイルで指定されています。

PALポータルのセキュリティアーキテクチャは JAAS に準拠しています。 必要に応じて、LoginModule や Policy の実装は変更可能です。 PALポータルの実装では、機能拡張しやすいように SPI モデルを採用しています。

認証概要

認証に関して、PALポータルでは Java の LoginModule 実装を利用しています。 LoginModule は、複数の認証システムからユーザー認証ができるように、DefaultLoginModule として柔軟な実装をしています。 DefaultLoginModule 内では、UserManager を利用して、さまざまな認証サービスへアクセスしています。 そのクラスダイアグラムは、以下の図のように、認証サービスへのアクセスを実装しています。



上記のコンポーネントについては、以下のとおりです。

コンポーネント	説明
DefaultLoginModule	PALポータル標準の LoginModule の実装で、UserManager の authenticate() メソッドを利用しています。 そのメソッドにより、設定された様々な AuthenticationProvider に対して、認証を提供することができます。
UserManager	認証とユーザー管理を提供するコンポーネントです。 AuthenticationProviderProxy や SecurityProvider を通して、様々な AuthenticationProvider が利用することができます。
SecurityProvider	SPI の実装で利用可能なセキュリティプロバイダーを提供します。
AuthenticationProviderProxy	複数の AuthenticationProvider 実装のプロキシとして動作します。 AuthenticationProviderProx は、認証およびユーザー管理のために適切な AuthenticationProvider を呼び出すことができます。

承認概要

PALポータルは、主体とパーミッション間の関係を管理するデータベースを利用するための java.security.Policy を実装しています。

PermissionManager は与えられた主体に関連したパーミッションへのアクセスを提供します。

セキュリティサービス

PALポータルでは以下のセキュリティサービスへのインターフェースを提供しています。

- UserManager: ユーザー管理機能を提供します。
- GroupManager: グループ管理機能を提供します。
- RoleManager: ロール管理機能を提供します。
- PermissionManager: パーミッション管理機能を提供します。

6.2 設定ファイル

設定ファイル

PALポータル標準のセキュリティサービスの設定は、独自のデータベースで管理する実装を利用しています。 セキュリティサービスは、その独自のデータベース管理を LDAP など任意のサービスに切り替え可能な仕組みとして実装してあります。 ですので、必要に応じて、設定ファイルを変更することでセキュリティサービスを変更することが可能です。

全ての設定ファイルは webapps/palportal/WEB-INF/assembly に配置されています。

security-atn.xml

この設定ファイルはログインモジュールの設定を提供しています。 ログインモジュールを変更したい場合に編集します。

security-atz.xml

この設定ファイルは、承認ポリシーの設定を提供します。

security-managers.xml

この設定ファイルは、セキュリティに関するマネージャ機能を提供します。

security-providers.xml

この設定ファイルは、様々なプロバイダーを提供しています。

AuthenticationProviderProxy は AuthenticationProvider のリストと、デフォルトの認証方法名を設定します。 これにより、複数の認証方法が適用することができます。

```
<bean id="org.apache.jetspeed.security.AuthenticationProviderProxy"
    class="org.apache.jetspeed.security.impl.AuthenticationProviderProxyImpl">
    <constructor-arg >
        <list>
            <ref bean="org.apache.jetspeed.security.AuthenticationProvider"/>
        </list>
    </constructor-arg>
    <constructor-arg ><value>DefaultAuthenticator</value></constructor-arg>
```

```
</bean>
```

AuthenticationProvider は、ポータル上で利用する認証プロバイダーを設定しています。以下の例では、ユーザー情報の管理にデータベースを利用する認証方法を設定しています。

```
<bean id="org.apache.jetspeed.security.AuthenticationProvider"
    class="org.apache.jetspeed.security.impl.AuthenticationProviderImpl">
    <constructor-arg index="0"><value>DefaultAuthenticator</value></constructor-arg>
    <constructor-arg index="1"><value>The default authenticator</value></constructor-arg>
    <constructor-arg index="2"><value>login.conf</value></constructor-arg>
    <constructor-arg index="3">
        <ref bean="org.apache.jetspeed.security.spi.CredentialHandler"/>
    </constructor-arg>
    <constructor-arg index="4">
        <ref bean="org.apache.jetspeed.security.spi.UserSecurityHandler"/>
    </constructor-arg>
</bean>
```

AuthorizationProvider はパーミッションを適用するために使用される SecurityPolicies を設定します。

```
<bean id="org.apache.jetspeed.security.AuthorizationProvider"
    class="org.apache.jetspeed.security.impl.AuthorizationProviderImpl">
    <constructor-arg index="0">
        <ref bean="org.apache.jetspeed.security.impl.RdbmsPolicy"/>
    </constructor-arg>
    <!-- Does not use the default policy as a default behavior -->
    <constructor-arg index="1"><value>false</value></constructor-arg>
</bean>
```

security-spi.xml

この設定ファイルは、共通の認証・承認 SPI の設定を提供します。

security-spi-atn.xml

この設定ファイルは、認証 SPI の設定を提供します。

コンポーネント名	説明
org.apache.jetspeed.security.spi.CredentialHandler	CredentialHandler は資格に関する操作を内包します。その処理は、PasswordCredentialProvider や InternalPasswordCredentialInterceptor により定義されたパスワード検証処理の実装を持ちます。
org.apache.jetspeed.security.spi.UserSecurityHandler	UserSecurityHandler はユーザー主体まわりの操作を内包します。

security-spi-atz.xml

この設定ファイルは、承認 SPI の設定を提供します。

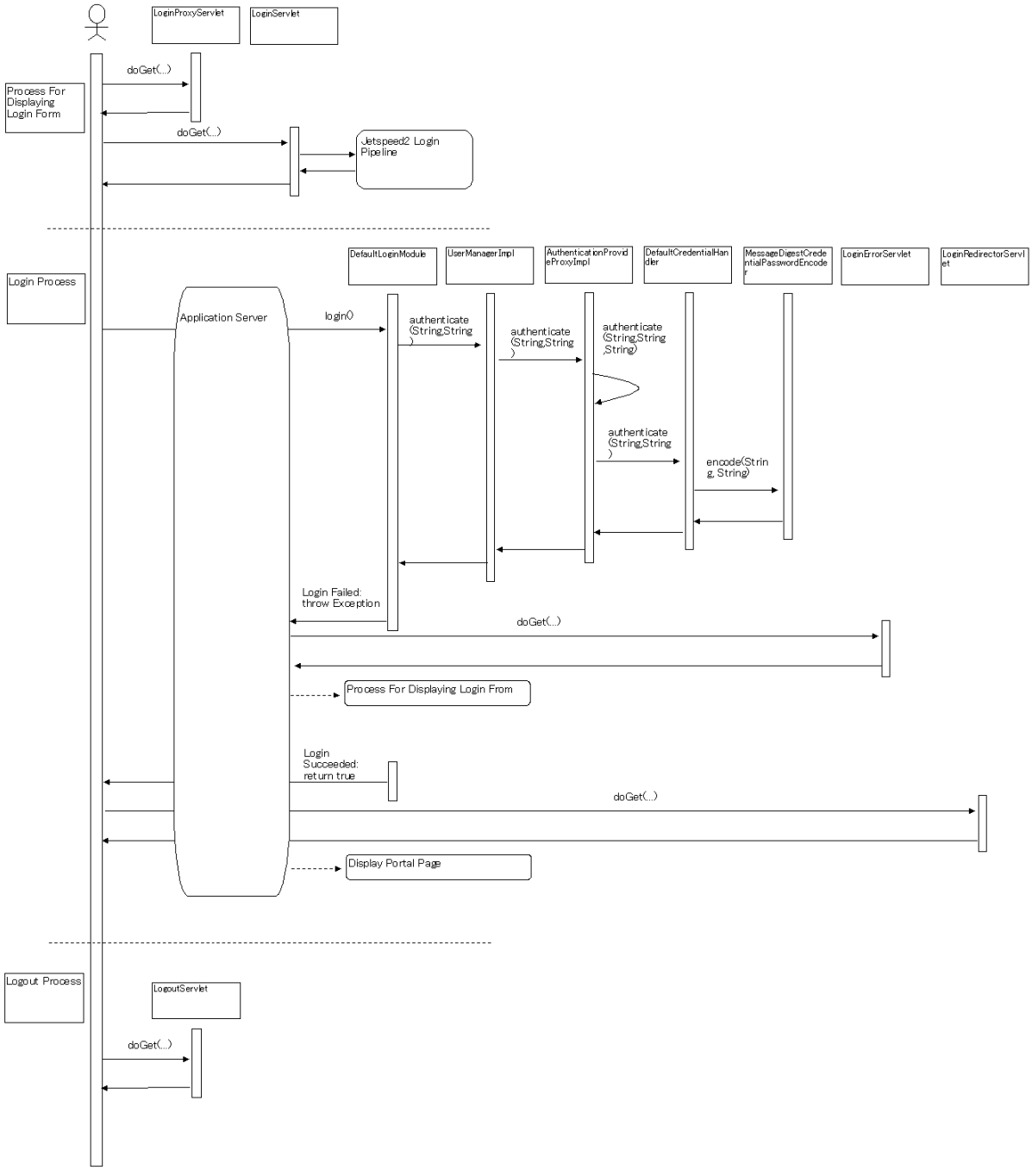
コンポーネント名	説明
org.apache.jetspeed.security.spi.RoleSecurityHandler	RoleSecurityHandler はロール主体まわりの処理を内包しています。
org.apache.jetspeed.security.spi.GroupSecurityHandler	GroupSecurityHandler はグループ主体まわりの処理を内包しています。
org.apache.jetspeed.security.spi.SecurityMappingHandler	SecurityMappingHandler は主体間のマッピング操作を内包しています。

6.3 ログイン

ログインについて

ユーザー認証について、PALポータルでは JAAS のログインモジュールの実装を提供しています。 そのため、PALポータルで提供しているログインモジュールは、標準仕様に基づいているので、認証サービスに関する詳しい情報については、JDK のドキュメントなどを参照してください。

PALポータルのログイン処理の流れについては以下のようになります。



LoginProxyServlet にアクセスして、ログイン画面を作成するための初期値を設定後、LoginServlet でログイン画面が生成されます。 ログイン画面で、ユーザー名とパスワードを JAAS のモデルに基づき、アプリケーションサーバーに送信して、DefaultLoginModule でログイン処理が行われます。 実際のパスワードの検証処理などは、UserManager により行われます。 ログインに失敗であれば、LoginErrorServlet から再度ログイン画面が表示され、成功していれば、LoginRedirectorServlet からユーザーのポータルページを表示します。 ログアウトについては、LogoutServlet にアクセスすること

で処理されます。

ログインモジュールの設定

設定は、通常の JAAS 認証の設定になります。 PALポータルでは、デフォルトで org.apache.jetspeed.security.impl.DefaultLoginModule を利用します。 この設定ファイルは login.conf であり、jetspeed2-security-{version}.jar の中に保存されています。 login.conf の内容は以下の通りです。

```
Jetspeed {
   org.apache.jetspeed.security.impl.DefaultLoginModule required;
};
```

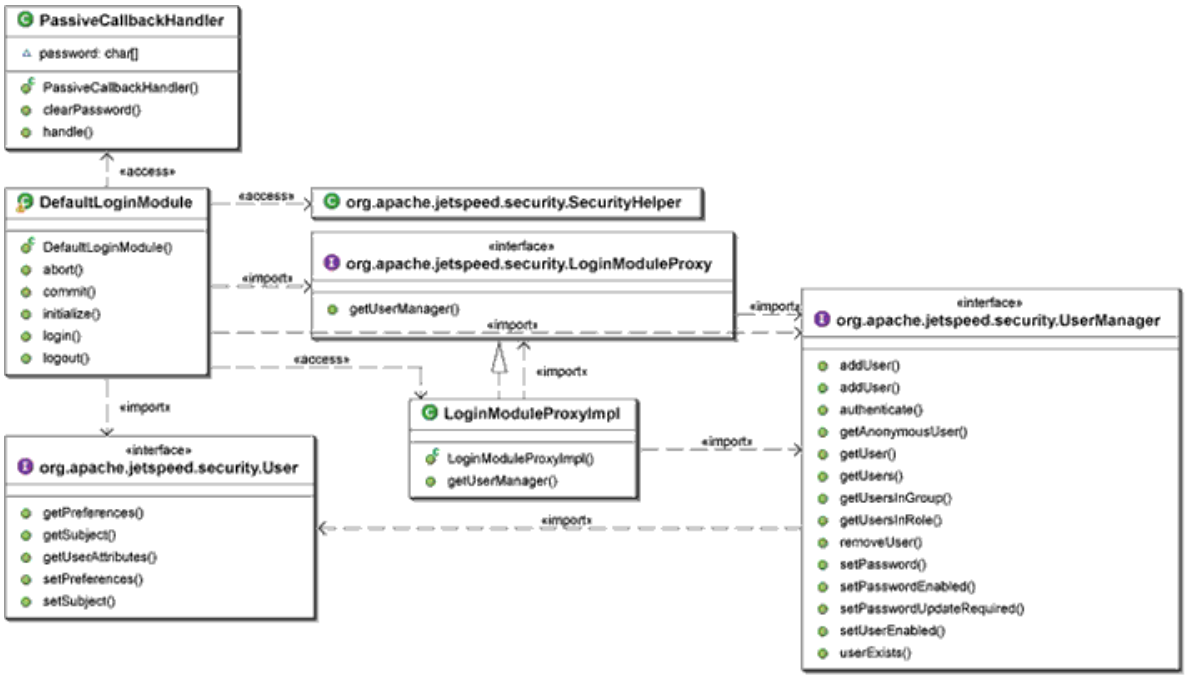
この設定を上書きして変更するためには、webapps/palportal/WEB-INF/classes に login.conf を作成します。 login.conf のファイル名などを変更したい場合などは、security-providers.xml で login.conf を指定しているので、その値を変更してください。

```
<bean id="org.apache.jetspeed.security.AuthenticationProvider"
  class="org.apache.jetspeed.security.impl.AuthenticationProviderImpl">
  <constructor-arg index="0"><value>DefaultAuthenticator</value></constructor-arg>
  <constructor-arg index="1"><value>The default authenticator</value></constructor-arg>
  <constructor-arg index="2"><value>login.conf</value></constructor-arg>
  <constructor-arg index="3">
    <ref bean="org.apache.jetspeed.security.spi.CredentialHandler"/>
  </constructor-arg>
  <constructor-arg index="4">
    <ref bean="org.apache.jetspeed.security.spi.UserSecurityHandler"/>
  </constructor-arg>
</bean>
```

AuthenticationProvider は、システムプロパティの java.security.auth.login.config の値を security-providers.xml で指定した login.conf のパスを設定して、使用される LoginModule を設定します。

ログインモジュールの実装

DefaultLoginModule の実装は、以下の図のようになります。



DefaultLoginModule で使用されるクラスの役割は以下の通りです。

クラス名	説明
org.apache.jetspeed.security.impl.DefaultLoginModule	javax.security.auth.spi.LoginModule の実装です。DefaultLoginModule での認証確認は、UserManager を利用しています。
org.apache.jetspeed.security.LoginModuleProxy	UserManager を DefaultLoginModule で利用するためのユーティリティコンポーネントです。
org.apache.jetspeed.security.User	User は javax.security.auth.Subject と java.util.prefs.Preferences を保持するインターフェースです。PALポータルでは、UserPrincipal、RolePrincipal、GroupPrincipal の 3 種類の主体を扱います。
org.apache.jetspeed.security.UserManager	ユーザーに関する操作を提供するインターフェースです。 このインターフェースは、様々な SPI を集約したもので、SPI で定義された機能を利用することができます。

6.4 資格管理

資格管理について

この節では、パスワードまわりの処理について説明します。

DefaultCredentialHandler の機能

DefaultCredentialHandler により、パスワード資格の様々な管理を簡単に実現できます。PasswordCredentialProvider と InternalPasswordCredentialInterceptor コンポーネントを利用して、パスワード検証を柔軟に設定することができます。

PALポータルでは、パスワード資格を扱うために PasswordCredential の実装を提供して、認証に利用しています。

パスワードエンコード

PasswordCredentialProvider から CredentialPasswordEncoder が利用可能であれば、パスワードは保存される前にエンコードされます。 提供している MessageDigestCredentialPasswordEncoder はパスワードの暗号化にメッセージダイジェストのハッシュアルゴリズムを適用します。 たとえば、SHA-1 や Base64 などが利用可能です。

パスワード検証

PasswordCredentialProvider から CredentialPasswordValidator が利用可能であれば、パスワードは保存される前に検証されます。 たとえば、DefaultCredentialPasswordValidator はパスワードを必須にします。 また、SimpleCredentialPasswordValidator は最小の文字列数や最小の数値文字数などを検証することもできます。

InternalCredential では、ライフサイクルイベントで割り込んで検証します。DefaultCredentialHandler が InternalPasswordCredentialInterceptor で提供されていれば、以下の状況で呼ばれます。

- 保存場所から資格を読み取った後
- ユーザー認証した後
- 新しい資格を保存する前
- 新しいパスワードを資格に保存する前

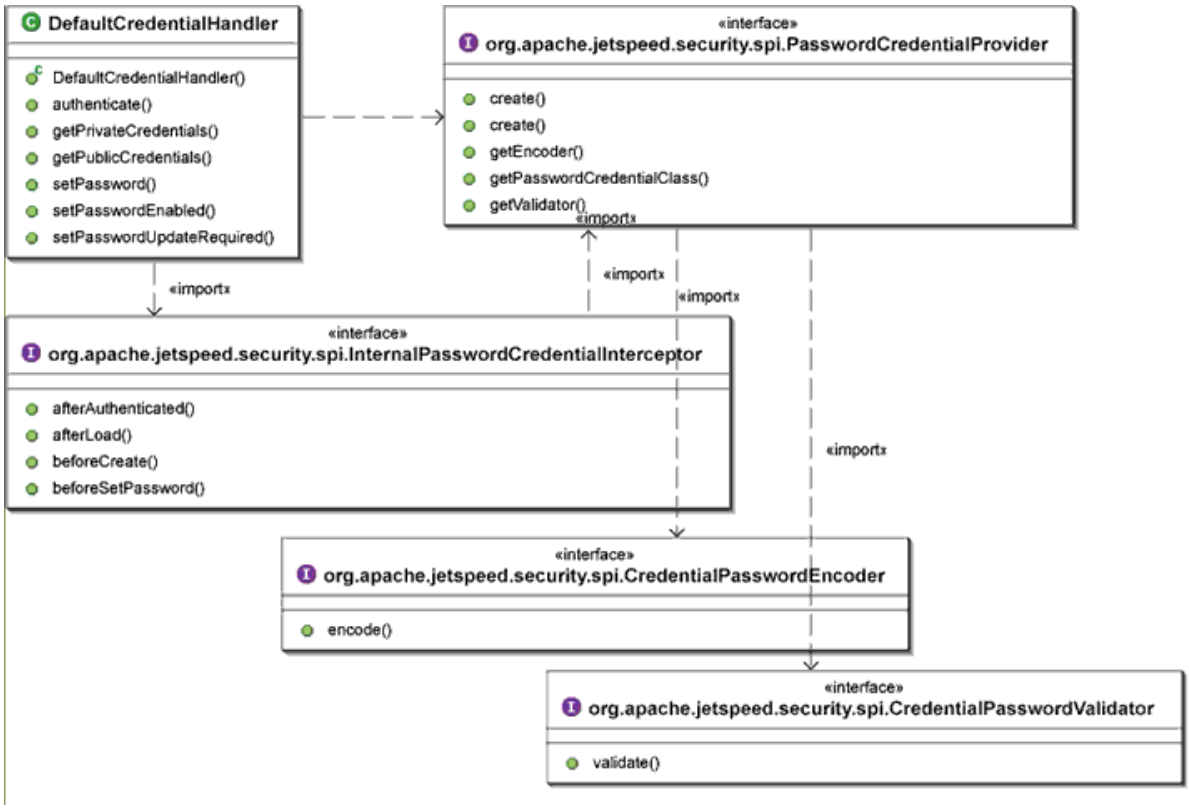
PALポータルは、基本的なインターセプターを提供しています。

- ValidatePasswordOnLoadInterceptor: このインターセプターは保存時にパスワードを検証し、正しくなければ変更を要求します。 それは PasswordCredentialProvider の CredentialPasswordValidator により構成されます。
- EncodePasswordOnFirstLoadInterceptor: このインターセプターはパスワード保存先から暗号化されていないパスワードを読み取ったときに実行されます。 PasswordCredentialProvider の CredentialPasswordEncoder を使用して、データベースから平文パスワードを初回読み込み時に自動的にエンコードします。
- PasswordExpirationInterceptor: このインターセプターはパスワードの有効期間を確認します。 それは、InternalCredential の expiration_date と is_expired の値で管理され、期限切れの場合は期限切れフラグを立てます。
- MaxPasswordAuthenticationFailuresInterceptor: このインターセプターは不正パスワード入力の最大回数を確認して、パスワードハッキングを防ぎます。 最大失敗回数に到達すると、その資格は無効になります。 認証に成功すると、失敗回数はリセットされます。
- PasswordHistoryInterceptor: このインターセプターは過去にパスワードが利用したものでないかを確認します。 新いパスワードを設定すると、現在のパスワードを FIFO スタックに保存します。 以前に利用したものであれば、 PasswordAlreadyUsedException が発生します。

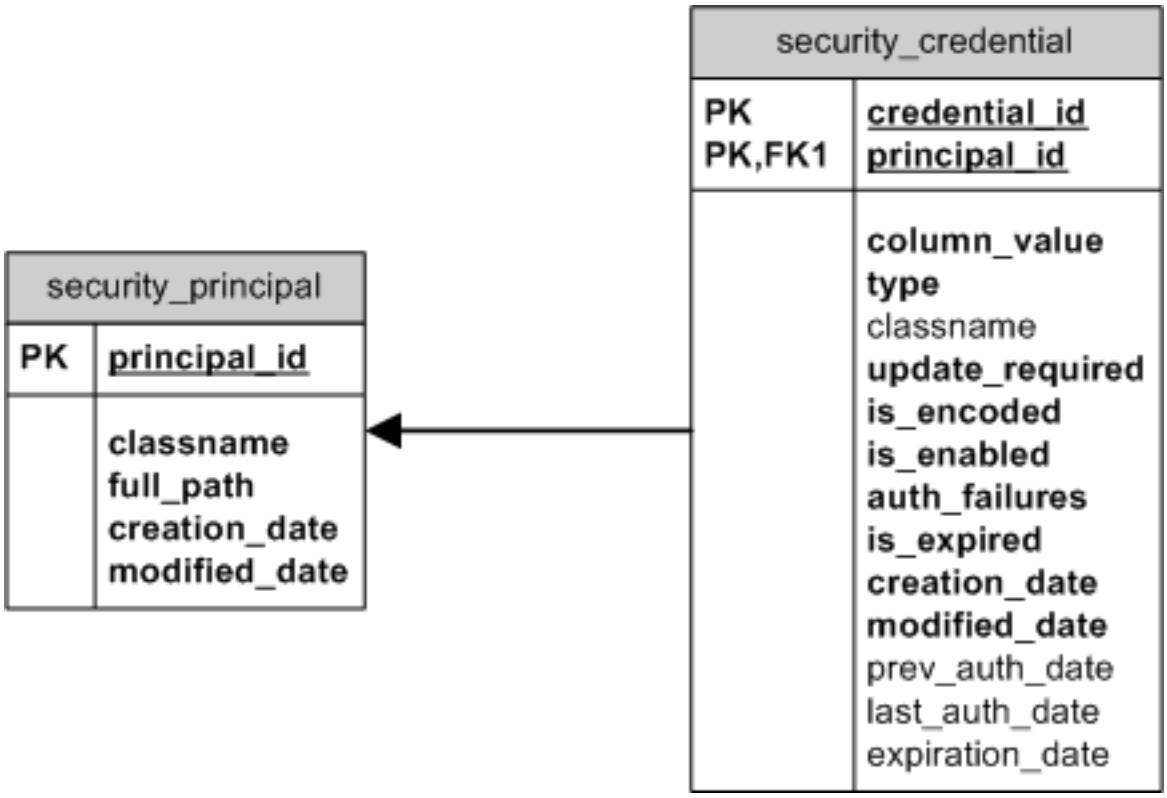
DefaultCredentialHandler は 1 つだけのインターセプターをサポートしています。 しかし、InternalPasswordCredentialInterceptorsProxy を用いることで、複数のインターセプターを利用することができます。

資格管理の実装

DefaultCredentialHandler の実装で利用されるコンポーネントを以下の図に示します。



標準の資格実装では、security_repository.xml で指定される OJB によるマッピングで実現されています。 InternalCredential はSECURITY_CREDENTIAL テーブルにマップされています。 以下が対象部分のデータベーススキーマになります。



6.5 ポートレットからの利用

ポートレットからの利用

UserManager などのセキュリティサービスのコンポーネントは、ポートレットからも呼び出し可能です。 ポートレット上でそれらを利用するためには、拡張設定ファイルの jetspeed-portlet.xml をポートレットの WEB-INF に配置してください。 jetspeed-portlet.xml で js:services タグで利用するコンポーネント名を指定することで、PortletContext からそのコンポーネントを取得することができます。

利用方法については、「ポートレット」章の「ポータル機能の利用」を参照してください。

6.6 認証フィルター

認証フィルターについて

PALポータルは J2EE のフォーム認証をサポートしていますが、様々な認証環境にサポートするため、認証フィルター群を提供しています。 認証フィルターを利用することで、リクエストパラメータやクッキーなどからユーザー名とパスワードを取得して、認証処理を実行できます。 たとえば、他のシステムで認証後、そのシステムでクッキーに認証するための情報を入れて、PALポータルにアクセスすることで、ログイン画面を表示することなしにログインすることができます。

認証フィルターの種類

PALポータルでは、次の認証フィルターを提供しています。

- jp.sf.pal.portal.filter.CookieAuthFilter: クッキーからユーザー名とパスワードを取得する。
- jp.sf.pal.portal.filter.RequestHeaderAuthFilter: リクエストヘッダーからユーザー名とパスワードを取得する。
- jp.sf.pal.portal.filter.RequestParameterAuthFilter: リクエストパラメータからからユーザー名とパスワードを取得する。

初期値

web.xml で filter 要素内の init-param 要素で値を指定できます。

username.key

- ユーザー名を取得するためのキー
- デフォルト値: org.apache.jetspeed.login.username

password.key

- パスワードを取得するためのキー
- デフォルト値: org.apache.jetspeed.login.password

skip.password.check

- パスワード確認をスキップするかどうか。
- デフォルト値: false

設定方法

web.xml に以下のように記述します。

```
...
  <filter>
    <filter-name>AuthFilter</filter-name>
    <filter-class>jp.sf.pal.portal.filter.RequestHeaderAuthFilter</filter-class>
    <init-param>
      <param-name>username.key</param-name>
      <param-value>USERNAME</param-value>
    </init-param>
    <init-param>
      <param-name>password.key</param-name>
      <param-value>PASSWORD</param-value>
    </init-param>
    <init-param>
      <param-name>skip.password.check</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>
...
  <filter-mapping>
    <filter-name>AuthFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
...
```

設定を保存し、ポータルを再起動後に有効になります。

6.7 情報転送フィルター

情報転送フィルターについて

PALポータルにログイン後、そのログインユーザーの情報を他アプリケーションに転送するためのフィルターを提供しています。 このフィルターを利用することで、他アプリケーションに対して、シングルサインオン環境を実現することができます。

認証フィルターの種類

PALポータルでは、次の認証フィルターを提供しています。

- jp.sf.pal.portal.filter.CookieTransferFilter: ユーザー名やユーザー情報をクッキーに設定します。

初期値

web.xml で filter 要素内の init-param 要素で値を指定できます。

transferred.info

- 転送対象の情報を指定する。転送対象となる情報は、ユーザー情報(JSR 168 PLT.17)として格納されている情報です。それ以外にユーザーIDをusernameとして定義しています。
- 「名前キー=値キー」を , 区切りで指定する。名前キーがクッキーでキーとして利用されます。値キーはユーザー情報で利用されるキーの部分または username を利用します。

path

- クッキーに指定する際の path として渡される情報。

domain

- クッキーに指定する際の domain として渡される情報。

max.age

- クッキーに指定する際の maxAge として渡される情報。

secure

- クッキーに指定する際の secure として渡される情報。

設定方法

web.xml に以下のように記述します。

```
...
  <filter>
    <filter-name>CookieTransferFilter</filter-name>
    <filter-class>jp.sf.pal.portal.filter.CookieTransferFilter</filter-class>
    <init-param>
      <param-name>transferred.info</param-name>
      <param-value>H=username,NAME=user.name.given</param-value>
    </init-param>
    <init-param>
      <param-name>path</param-name>
      <param-value>/</param-value>
    </init-param>
    <init-param>
      <param-name>max.age</param-name>
      <param-value>-1</param-value>
    </init-param>
  </filter>
...
  <filter-mapping>
    <filter-name>CookieTransferFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
...
```

設定を保存し、ポータルを再起動後に有効になります。

7.1 デザインについて

デザインの概要

ポートレット API（JSR 168）は、ポートレットプログラミングの Java 標準インターフェースを定義しています。 ポートレット API 仕様により、ポートレットとポータルの間の相互の接続が可能になります。 しかし、Java ポートレット API は、ポータルのデザインについての標準は定義していません。 本章では、デザインに関する事項について説明します。

デザインテンプレート

ポートレットページ内の 1 つ以上のポートレットを描画するということは、描画するためのテンプレートをを用意して、それらのポートレットを動的なコンテンツとして集約する処理を行うことになります。 PALポータルは、ページの描画を Apache Velocity テンプレートを使って生成します。 PALポータルの構造は、Velocity 以外に JSP をテンプレートとして使うことが可能な構造をしていますが、Velocity の利用をサポートしています。

PALポータルには、レイアウトテンプレートと 2 種類のデコレータが存在しています。 レイアウトテンプレートは、ポートレットの表示列数などページの基本構成を描画する処理をします。 デコレータは、レイアウトデコレータとポートレットデコレータがあり、前者はページ全体のデザインを構成し、後者は表示するポートレットのデザインを構成します。

レイアウトテンプレート

レイアウトテンプレートは、ポートレットの表示列数を表現するなどの、ポータルページの集約処理を制御します。 レイアウトテンプレートの機能的な部分は、PALポータル用の配備可能なレイアウトポートレットアプリケーションとしてまとめられています。 レイアウトテンプレートは、コンテンツ集約のためのコンポーネントモデルを提供し、ポートレットが生成したコンテンツを統合してポータルページを生成します。 PALポータルは一列、または二列、または三列のレイアウトを含むいくつかのレイアウトコンポーネントを同梱しています。 必要に応じて、新規のレイアウトを定義することも可能です。

デコレータ

デコレータには、レイアウトデコレータとポートレットデコレータがあります。 レイアウトデコレータでは、ページのヘッダーやフッターなどのページ全体のでデザインを構成します。 ポートレットデコレータでは、ポートレットの枠の部分やポートレット内で表示す

るコンテンツのスタイルを構成します。 デコレータは、Velocity テンプレートにより記述され、利用する CSS や画像などのデータを含みます。 PALポータルでは、デコレータの記述テンプレートとして JSP も利用可能ですが、標準では Velocity をサポートします。デコレータは、PALポータル用の配備可能なアーカイブとしてまとめることも可能です。

7.2 レイアウトテンプレート

レイアウトテンプレート

レイアウトテンプレートはポータルページのコンテンツ集約処理を制御しています。 レイアウトテンプレートの機能的な部分は PALポータル用の配備可能なレイアウトポートレットアプリケーション内にパッケージされています。 レイアウトテンプレートはレイアウトポートレットで指定された集約処理を元にポートレットのコンテンツを集約して、ポータルページを生成します。 PALポータルは、標準で 1 列、2 列、3 列のレイアウトを含むいくつかのレイアウトコンポーネントを持っています。 デフォルトの PALポータルで指定可能なレイアウトは後述の「レイアウトの種類」を参照してください。 必要に応じて、新規にレイアウトを定義したり、変更したりすることも可能です。

レイアウトはどのように 1 つのポータルページがコンテンツ集約するのかを定義します。レイアウトテンプレートはコンテンツの配置をどのようにするかを処理し、ポータルへリクエストされたコンテンツが集約される方法を定義します。 レイアウトは、ポータルページのコンテンツを生成するために、テンプレート処理のアルゴリズムを実行するポートレットにより定義されます。 典型的なレイアウト構成のアルゴリズムには 2 列のもの、3 列のもの、階層化されたものがあります。

レイアウトテンプレートは以下により構成されます。

- レイアウトポートレット
- レイアウトテンプレートファイル

レイアウトポートレット

レイアウトポートレットは、jetspeed-layouts ポートレットと呼ばれ、PALポータル用の JSR 168 ポートレットとして構成されています。 レイアウトポートレットは、jetspeed-layouts.war というファイルでパッケージ化され、PALポータルに配備すると webapps/palportal/WEB-INF/apps/jetspeed-layouts に配置されます。 レイアウトポートレットは、JSR 168 のポートレットですが、jetspeed で始まるポートレット名は PALポータルではポータル用のポートレットとして認識され、webapps/palportal/WEB-INF/apps 以下に配備されます。

レイアウトポートレットは標準的なポートレットアプリケーション同様に、以下の操作をサポートします。

- 配備
- 配備解除
- 再配備

レイアウトの種類

PALポータルは 1 列、2 列、3 列のレイアウトを含むいくつかのレイアウトコンポーネントを標準で含んでいます。 必要に応じて、新規にレイアウトを定義したり、変更したりすることも可能です。 以下の表は PALポータルで利用可能なレイアウトコンポーネントのリストです。 ページのレイアウトを変更する際には、利用したいレイアウトコンポーネントを選択します。 PSML ページごとに 1 つのルートレイアウトを指定することが可能です。

ポートレット名	レイアウト名	カラム数	サイズ	説明
VelocityOneColumn	1 列レイアウト	1	100%	ポートレットの表示エリアの 100% 幅を占める 1 列表示。
VelocityTwoColumns	2 列レイアウト(50/50)	2	50%,50%	ポートレットの表示エリアに左から 50%、50% 幅を割り当てる 2 列表示。
VelocityTwoColumns3366	2 列レイアウト(33/66)	2	33%,66%	ポートレットの表示エリアに左から 33%、66% 幅を割り当てる 2 列表示。
VelocityTwoColumns2575	2 列レイアウト(25/75)	2	25%,75%	ポートレットの表示エリアに左から 25%、75% 幅を割り当てる 2 列表示。
VelocityTwoColumns2080	2 列レイアウト(20/80)	2	20%,80%	ポートレットの表示エリアに左から 20%、80% 幅を割り当てる 2 列表示。
VelocityTwoColumns1585	2 列レイアウト(15/85)	2	15%,85%	ポートレットの表示エリアに左から 15%、85% 幅を割り当てる 2 列表示。
VelocityTwoColumns1090	2 列レイアウト(10/90)	2	10%,90%	ポートレットの表示エリアに左から 10%、90% 幅を割り当てる 2 列表示。
VelocityTwoColumns595	2 列レイアウト(5/95)	2	5%,95%	ポートレットの表示エリアに左から 5%、95% 幅を割り当てる 2 列表示。
VelocityTwoColumns6633	2 列レイアウト(66/33)	2	66%,33%	ポートレットの表示エリアに左から 66%、33% 幅を割り当てる 2 列表示。
VelocityTwoColumns7525	2 列レイアウト(75/25)	2	75%,25%	ポートレットの表示エリアに左から 75%、25% 幅を割り当てる 2 列表示。
VelocityTwoColumns8020	2 列レイアウト(80/20)	2	80%,20%	ポートレットの表示エリアに左から 80%、20% 幅を割り当てる 2 列表示。

ポートレット名	レイアウト名	カラム数	サイズ	説明
VelocityTwoColumns8515	2 列レイアウト(85/15)	2	85%,15%	ポートレットの表示エリアに左から 85%、15% 幅を割り当てる 2 列表示。
VelocityTwoColumns9010	2 列レイアウト(90/10)	2	90%,10%	ポートレットの表示エリアに左から 90%、10% 幅を割り当てる 2 列表示。
VelocityTwoColumns955	2 列レイアウト(95/5)	2	95%,5%	ポートレットの表示エリアに左から 95%、5% 幅を割り当てる 2 列表示。
VelocityThreeColumns	3 列レイアウト (33/33/33)	3	33%,33%,33%	ポートレットの表示エリアに左から 33%、33%、33% 幅を割り当てる 3 列表示。
VelocityThreeColumns206020	3 列レイアウト (20/60/20)	3	20%,60%,20%	ポートレットの表示エリアに左から 20%、60%、20% 幅を割り当てる 3 列表示。
VelocityThreeColumns157015	3 列レイアウト (15/70/15)	3	15%,70%,15%	ポートレットの表示エリアに左から 15%、70%、15% 幅を割り当てる 3 列表示。
VelocityFourColumns	4 列レイアウト	4	20%,30%,30%,20%	ポートレットの表示エリアに左から 20%、30%、30%、20% 幅を割り当てる 4 列表示。

ページへのレイアウトの適用方法については、「管理ガイド」を参照してください。

レイアウトの追加方法

新しい列数のレイアウトを新規に追加する場合は、そのポートレットの定義をレイアウトポートレットに追加する必要があります。 ポートレットクラスには、org.apache.jetspeed.portlets.layout.ActionLayoutPortlet を利用して定義を追加することができます。 また、javax.portlet.GenericPortlet を継承して、独自のレイアウトを作成することもできます。

たとえば、ActionLayoutPortlet を利用して新規にレイアウトを追加する場合は以下のような定義を webapps/palportal/WEB-INF/apps/jetspeed-layouts/WEB-INF/portlet.xml に追加します。

```
<portlet>
  <portlet-name>MyLayout</portlet-name>
  <display-name>My Layout</display-name>
```

```
    <display-name xml:lang="ja">私のレイアウト</display-name>
    <init-param>
      <name>ViewPage</name>
      <value>columns</value>
    </init-param>
    <init-param>
      <name>MaxPage</name>
      <value>maximized</value>
    </init-param>
    <init-param>
      <name>columns</name>
      <value>2</value>
    </init-param>
    <init-param>
      <name>sizes</name>
      <value>50%,50%</value>
    </init-param>
    <init-param>
      <name>layoutType</name>
      <value>TwoColumns</value>
    </init-param>
    <portlet-class>org.apache.jetspeed.portlets.layout.ActionLayoutPortlet</portlet-class>
    <resource-bundle>org.apache.jetspeed.portlets.layout.resources.LayoutResource</resource-bundle>
    <expiration-cache>0</expiration-cache>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>view</portlet-mode>
      <portlet-mode>edit</portlet-mode>
      <portlet-mode>help</portlet-mode>
    </supports>
    <portlet-info>
      <title>My Layout</title>
      <short-title>Layout</short-title>
    </portlet-info>
  </portlet>
```

portlet-name には追加する任意のレイアウトの ID を設定してください。 display-name には、ポータル上で表示されるレイアウト名を記述します。 display-name では、xml:lang 属性を利用して、複数の言語のレイアウト名を追加することができます。 init-param の columns の値には列数を記述して、sizes の値に表示される左から順番に幅をパーセントで指定します。 portlet-info の title 要素にはレイアウト名を記述します。

portlet.xml に上記の portlet 要素を追加後、ポータルを再起動すると、追加したレイアウトが有効になります。

レイアウトテンプレートファイル

レイアウトテンプレートファイルは、PALポータル内に配置されています。 それらのレイアウトテンプレートファイルがレイアウトポートレットから呼び出されて処理されます。 レイアウトテンプレートファイルは、webapps/palportal/WEB-INF/templates/layout に配置されています。 それらのファイルは、Velocity により記述されます。 PALポータルでは、レイアウトテンプレートファイルは汎用的に実装されており、テンプレート言語として JSP も利用可能ですが、標準では Velocity によるテンプレートファイルをサポートしています。

レイアウトテンプレートファイルは、レイアウトポートレットの設定情報に基づいて呼び出されます。 レイアウトテンプレートファイルの指定方法は、webapps/palportal/WEB-INF/templates/layout をルートディレクトリとして、<メディアタイプ>/<ViewPage>/layout.vm のテンプレートになります。 メディアタイプはポータルがクライアントの情報を元に決定され、ViewPage はレイアウトポートレットの portlet.xml で指定した情報を利用します。 ポートレットの表示モードにより、layout-<表示モード>.vm のテンプレートを利用できます。

列を記述している HTML を編集したい場合は、対象の layout.vm を編集することで変更することができます。

7.3 レイアウトデコレータ

レイアウトデコレータについて

レイアウトデコレータは、ポータルページのデザインを構成するファイル群です。 レイアウトデコレータは、1 つの PSML ファイルを表現するポータルページのヘッダーとフッターを記述するテンプレートで処理されます。 テンプレート言語には Velocity を用います。 PALポータルはテンプレート部分を汎用的に実装されているので、テンプレート言語に JSP も利用可能ですが、Velocity を標準でサポートします。

構成ファイル

レイアウトデコレータは、webapps/palportal/decorations/layout に配置されます。 このディレクトリをルートディレクトリとして、レイアウトデコレータの名前でディレクトリを作成して、以下のファイルを配置します。

- decorator.properties
- header.vm
- footer.vm
- css/styles.css

たとえば、mydesign というレイアウトデコレータを利用したい場合は、webapps/palportal/decorations/layout/mydesign ディレクトリを作成して、上記のファイルを配置します。

基本設定(decorator.properties)

decorator.properties ファイルは、レイアウトデコレータの基本情報を保持します。 実際のところ、このファイルは空でも構いませんが、他の API が使用可能なデコレータを探すために使用されるので、ファイルは存在する必要があります。 ですので、下記の定義にあるプロパティは 全てオプションであると仮定しても問題ありません。

プロパティ名	説明	デフォルト
base.css.class	この値は、ヘッダーのテンプレートファイルの一番上の要素タグの中に配置されます。	デコレータ名
stylesheet	デコレータのスタイルシートへの相対パス	css/styles.css

プロパティ名	説明	デフォルト
header	デコレーションのヘッダーテンプレートへの相対パス	header.vm
footer	デコレーションのフッターテンプレートへの相対パス	footer.vm

テンプレートファイル

テンプレートファイルは、ヘッダーテンプレートとフッターテンプレート二より構成されます。

ヘッダーテンプレート

ヘッダーテンプレートファイル header.vm は、ポータルページのヘッダー部分を生成します。 注意: 前提事項として、HTML と CSS についての十分な知識があると仮定してあります。 デコレータを作成するにあたり、基本的な Velocity の知識はあれば良いですが、必須の知識ではありません。

通常、header.vm では、上部に下記のようにレイアウトデコレータで利用するパラメータの設定等を行います。

```
#set($portalMode = "standard")##
## Add the current layouts configuration values to the context
#defineLayoutObjects()##
## Loads our custom macros
#parse($layoutDecoration.getResource("decorator-macros.vm"))##
## Username
#set($username = $JS2RequestContext.request.remoteUser)##
## Check edit
#set($_actions = $layoutDecoration.actions)##
#foreach ($_action in $_actions)##
  #if($_action.actionName == "edit")##
    #set ($editable = true)##
  #end##
#end##
#set($layoutId = $jetspeed.currentFragment.id)##
#set($servletName = $JS2RequestContext.request.servletPath)##
#initMessageResourceBundles()##
```

#defineLayoutObjects()は、Velocity でマクロと言われるものです。 マクロは、事前に定義された Velocity の小さなコードで、Velocity テンプレートの中で再利用されます。

これから使用する全てのグローバルマクロ（このマクロも含む）は、webapps/palportal/WEB-INF/jetspeed_macros.vm で定義されています。 もし、カスタムマクロを作成したい場合は、レイアウトデコレータ内に decorator-macros.vm を作成して、その中にカスタムマクロを記述します。

#defineLayoutObjects()は、様々な値をレイアウトデコレータ内で利用可能にします。 設定された値は、header.vm や footer.vm などの様々なテンプレートからアクセス可能になります。 #defineLayoutObjects() では、webapps/palportal/WEB-INF/jetspeed_macros.vm で以下のように記述されています。

```
#macro (defineLayoutObjects)
  #set($preferedLocale = $JS2RequestContext.locale)
  #set($rootFragment = $jetspeed.currentFragment)
  #set($site = $request.getAttribute("org.apache.jetspeed.portalsite.PortalSiteRequestContext"))
  #set($theme = $request.getAttribute("org.apache.jetspeed.theme"))
  #set($layoutDecoration = $theme.getDecoration($rootFragment))
#end
```

#set() は、Velocity の変数に値を設定する関数です。 これは Velocityで指示子と呼ばれるものです。 指示子はオリジナルの関数であり、マクロではありません。 #set()は、イコール記号の右辺にある値を受け取り左辺へ割り当てます。 Velocity の変数は $ で始まる文字列になります。

全てのVelocityテンプレートで利用可能なオブジェクトは、$someObjectという書き方で参照することができます。 メソッドの実行は、$someObject.getFoo() の様に書くことで実行できます。 getFoo()を実行する場合、引数を取らないgetterメソッドは、$someObject.fooという記述も可能です。 この $JS2RequestContext に関しては、org.apache.jetspeed.RequestContext のインスタンスで Velocity で使用できるように、PALポータルによってインスタンス化されています。 org.apache.jetspeed.RequestContext の JavaDoc を見るとわかりますが、$JS2RequestContext.locale は、現在のポートレットのユーザーのロケールとなる、java.util.Locale のインスタンスを渡しています。

次に、#set($rootFragment = $jetspeed.currentFragment）という set の行は、$rootFragmentと呼ばれるインスタンスを生成しています。 これは、org.apache.jetspeed.om.page.ContentFragment のインスタンスで、ページ内のルートフラグメントです。 通常ルートフラグメントはレイアウトポートレットになります。

$request は、javax.servlet.http.HttpServletRequestのインスタンスで、PALポータルからVelocity に渡されたオブジェクトを取り出します。$site と $theme は、それぞれorg.apache.jetspeed.portalsite.PortalSiteRequestContext とorg.apache.jetspeed.decoration.Theme です。

パラメータの設定後、header.vm では、〈html〉から始まるタグを記述していきます。

```
<html>
```

```
  <head>
    #includeHeaderResource()
    <base href="#BaseHref()">
    <meta http-equiv="Content-type" content="#ContentType()" />
    <meta http-equiv="Content-style-type" content="text/css" />
    #IncludeStylesheets()
    <title>#PageTitle()</title>
    <meta name="version" content="#SiteVersionTag()">
    <meta name="keywords" content="#PageKeywords()" />
    <meta name="description" content="#PageDescription()" />
    <script type="text/javascript" src="#GetPageResource('js/jquery-1.2.6.min.js')"></script>
    <script type="text/javascript" src="#GetPageResource('js/ui-1.5.2/ui.core.packed.js')"></script>
    <script type="text/javascript"
src="#GetPageResource('js/ui-1.5.2/ui.sortable.packed.js')"></script>
    <script type="text/javascript" src="#GetPageResource('js/main.js')"></script>
  </head>
<body class="#PageBaseCSSClass()">
  <input type="hidden" id="ajaxUrl" value="#BaseHref()ajaxapi$JS2RequestContext.request.pathInfo"/>
  <div class="#PageBaseCSSClass()">
...
<html>
```

最初に、<base href="#BaseHref()"> は、ウェブのリソースの場所の解決をするためのベースを定義する際に使用します。 <base> の議論については、 W3C Schools Reference を参照してください。 もしすでにあなたが PALポータルで遊んだ経験があるのであれば、一貫した html やスタイルシートへのパスを取得するのを妨げる、幾通りもの非常にややこしい URL リライトの問題に気づくでしょう。 BASEタグを定義することにより、一貫した html やスタイルシートへのパスを取得するのを妨げる、URL リライトの問題が解決されます。

#BaseHref() マクロは、単にウェブアプリケーションのルートディレクトリへの絶対パスを作成します。 このコードは、サーブレットAPIを使って記述した場合、以下の通りです。

```
HttpServletRequest request;
StingBuffer baseHref = new StringBuffer(request.getScheme())
     .append("://").append(request.getServerName())
         .append(":").append(request.getServerPort())
         .append(request.getContextPath()).append("/");
return baseHref.toString();
```

Velocity マクロのコードは簡潔になり、以下の通りです。

```
```

```
${request.scheme}://${request.serverName}:${request.serverPort}${request.contextPath}
```

#ContentType() は text/htmlと、UTF-8 のような適切なエンコードの種類を返します。

#PageBaseCSSClass() は base.css.class プロパティの値が利用され、スタイルシートの適用範囲を指定するために body タグ直下のタグでスタイルクラスとして適用するためなどに用いられます。

フッターテンプレート

footer.vm についても header.vm と同様に記述することができます。

スタイルシート

通常のCSSと同様に記述することができます。

レイアウトデコレータの追加

新規にレイアウトデコレータを追加したい場合は、上記のファイル群を作成し、webapps/palportal/decorations/layout/〈mydesign〉以下に配置することで〈mydesign〉が有効になります。デコレータの追加には、ポータルの再起動は必要ありません。

追加したレイアウトデコレータはサイトエディターから選択することができます。 設定方法の詳細については「管理ガイド」を参照してください。

7.4 ポートレットデコレータ

ポートレットデコレータについて

ポートレットデコレータは、ポートレットの枠の部分やポートレット内のコンテンツ部分のスタイルなどを定義するファイル群です。 ポートレットデコレータは、ポータルページ内の各ポートレットに適用可能なデザインです。 テンプレート言語には Velocity を用います。 PALポータルはテンプレート部分を汎用的に実装されているので、テンプレート言語に JSP も利用可能ですが、Velocity を標準でサポートします。

構成ファイル

ポートレットデコレータは、webapps/palportal/decorations/portlet に配置されます。 このディレクトリをルートディレクトリとして、ポートレットデコレータの名前でディレクトリを作成して、以下のファイルを配置します。

- decorator.properties
- css/styles.css
- decorator.vm

ただし、decorator.vmについては、定義されていない場合には webapps/palportal/decorations/portlet/decorator.vm がデフォルトとして使用されます。

基本的な定義方法については、レイアウトデコレータと同様です。

テンプレートフィル

テンプレートファイルは、decorator.vm で処理されます。 デザインに関する記述方法は、レイアウトデコレータと同様です。

decorator.vm では、対応するポートレットのコンテンツを出力する必要があります。 ポートレットのコンテンツの出力には、以下のコードで出力します。

```
$f.renderedContent
```

ポートレットデコレータの追加

新規にポートレットデコレータを追加したい場合は、上記のファイル群を作成し、webapps/palportal/decorations/portlet/〈mydesign〉以下に配置することで〈mydesign〉が有効になります。デコレータの追加には、ポータルの再起動は必要ありません。

追加したポートレットデコレータはサイトエディターから選択することができます。 設定方法の詳細については「管理ガイド」を参照してください。

7.5 デコレータユーティリティ

ユーティリティについて

デコレータでの記述ツールとして、Jetspeed Power Tool（JPT）が提供されています。JPT は、動的なコンテンツを生成するためのレイアウトやデコレータで使われる Velocity のツールです。 JPT は、リクエストレベルの Velocity のツールです。そして、レイアウトデコレータおよびポートレットデコレータで利用可能です。 JPT は、デコレータもしくはレイアウトで、

```
$jetspeed
```

として参照されます。 全ての公開の JPT API は、

- Java のメソッドのドット記法
- プロパティへの JavaBean ショートカット（getter/setter）

でアクセス可能です。 たとえば、メソッドの呼び出しは、

```
# 2 つのパラメータをもつメソッド 'getTitle' の呼び出し
$jetspeed.getTitle($myPE, $myF)
```

のようになります。 プロパティの取得は、

```
# page ビーンの呼び出し（$jetspeed.getPage に等しい）
$jetspeed.page
```

のようになります。

JPT Velocity API

以下の表は Velocity 用の PALポータルPower Tool の定義です。

API	decorateAndInclude($fragment)
説明	Velocity にインクルードするための、与えられたフラグメントパラメータ（$fragment）に対する、デコレータテンプレートへの相対パスを取得します。フラグメントは Velocity の #parse 関数にパラメータとして渡されます（#parse は他の Velocity テンプレートをインクルードします）。
パラメータ	$fragment - インクルードとデコレートを行うフラグメント。
返り値	String - デコレータテンプレートのアプリケーションでの相対パス。
例	#parse($jetspeed.decorateAndInclude($fragment)) 戻り値 /WEB-INF/decorations/layout/html/tigris/decorator-top.vm

API	getAbsoluteUrl(appRelativePath)
説明	ポートレットアプリケーションリソースへの相対パスを与えると、絶対 URL を返します。 この API はテンプレートの URL を参照するのに使うべきではありません。なぜなら、テンプレートの URL は、しばしば CMS 内に位置したり、保護された WEB-INF ディレクトリに位置したりするために、通常は絶対 URL としてはアクセスできないからです。
パラメータ	relativePath - デコレータパッケージ内のリソースへの相対パス。
返り値	String - ウェブリソースへの絶対パス
例	$jetspeed.getAbsoluteUrl("/images/test.gif") 戻り値 http://localhost:8080/jetspeed/portal/images/test.gif

API	columns
説明	レイアウトの集約の間に現在のフラグメントのための列のリストを返す。
パラメータ	-
返り値	List - サブフラグメントもしくは Java の標準の List
例	#set ($table = $jetspeed.columns) #foreach($entry in $table)

7.6 ログインデザイン

ログインのデザインについて

ログイン画面の表示には webapps/palportal/WEB-INF/templates/login に配置される JSP ファイルにより HTML が生成されます。 ですので、ログイン画面のデザインを変更したい場合は、JSP ファイルを編集することで自由に変更することができます。

ログイン JSP ファイル

ログイン画面の表示に利用される JSP ファイル名は login.jsp になります。 login.jsp ファイルは、指定されたテーマやメディアタイプなどのクライアント環境によって、以下の順で検索されます。

- <theme>/<mediatype>/<language>/<country>/login.jsp
- <theme>/<mediatype>/<language>/login.jsp
- <theme>/<mediatype>/login.jsp
- <theme>/login.jsp
- <mediatype>/<language>/<country>/login.jsp
- <mediatype>/<language>/login.jsp
- <mediatype>/login.jsp
- login.jsp

上記の順に login.jsp を探していき、存在したものを利用します。〈theme〉は、http://<hostname>/palportal/login/proxy へのアクセスする際にリクエストパラメータとして org.apache.jetspeed.login.theme の値として渡された文字列が利用されます。<mediatype>、<language>、<country> については、アクセスしたクライアント情報から取得した値を利用します。

デザインの追加と変更

通常の JSP ファイルをデザインする方法で作成および変更が可能です。

ログイン情報のフォームの送信方法については、J2EE の認証方法に従い、j_username と j_password にユーザー名とパスワードを設定して、j_security_check に送信します。 ログインが成功するとログインリダイレクターによりログイン後のポータルページへリダイレクトされ、失敗した場合は再度ログインページが表示されます。

8.1 ポートレットの概要

ポートレットの概要

PALポータルは、ポートレットAPI 1.0（JSR 168）に準拠しているので、JSR 168 のポートレットを配備することが可能です。 多くの J2EE ポータルが JSR 168 に対応しており、それらの上で動作する JSR 168 のポートレットを PALポータルに配備して動作させることができます。 現在、ポートレット API 2.0（JSR 286）が公開されており、PALポータルでは将来のバージョンで対応する予定です。

8.2 配備

配備について

ポートレットの配備は、ウェブアプリケーションの配備と同様に、ポートレットの war ファイルをポータルに対して配備することによって処理されます。 PALポータルでは、GUI によるポートレットの配備と CLI によるポートレットの配備をサポートしています。 どちらの操作も、PALポータルの再起動など必要なく、ポートレットを配備することができます。

GUI による配備

ポートレット管理ツールを利用することで、ブラウザ上からポートレットを配備することができます。 配備手順は、「管理ガイド」を参照してください。

CLI による配備

コマンドラインでは、ポートレットの war ファイルを webapps/palportal/WEB-INF/deploy にコピーすることで、配備することができます。 たとえば、helloworld.war を配備するには、

```
$ cp <somewhere>/helloworld.war webapps/palportal/WEB-INF/deploy
```

とすることで、helloworld ポートレットが配備されます。 定期的に webapps/palportal/WEB-INF/deploy は確認され、ポートレットが存在している場合は自動で配備プロセスが実行されます。

8.3 ポータル機能の利用

ポータル機能の利用について

ポートレット開発において、ポートレットからポータル固有の機能を呼び出したい場合もあるかと思います。 PALポータルでは、ポートレット内でポータル固有の機能を呼び出す仕組みを提供しています。

RequestContext

RequestContext は、クライアントからのリクエストを処理するために PALポータルが利用するリクエストのインスタンスです。 RequestContext はパイプラインに渡されて、パイプライン内の各バルブで処理がされ、様々な設定情報を保持しています。 ポートレットから RequestContext を呼び出すことで、ポータルの様々な情報にアクセスすることができます。

RequestContext は、ポートレットの処理に渡される前に、PortletRequest のリクエスト属性として格納されます。 ですので、ポートレット内では PortalReservedParameters.REQUEST_CONTEXT_ATTRIBUTE をキーにして、PortletRequest から取得することができます。

たとえば、呼び出し方法は以下のようになります。

```
RequestContext requestContext = (RequestContext) request
    .getAttribute(PortalReservedParameters.REQUEST_CONTEXT_ATTRIBUTE);
```

上記の方法を利用するためには、ポートレット開発において、ビルド時には jetspeed-api の jar ファイルが必要になります。 jetspeed-api はアプリケーションサーバーに保存されるので、実行時にはポートレットの war ファイル内に含めないように注意してください。

コンポーネント

PALポータルは、様々なコンポーネントにより構成されています。 ユーザー認証を行う機能などもコンポーネント化されているので、ポートレットからもそのコンポーネントを呼び出したい場合もあるかと思います。 そのような呼び出したいコンポーネントは、設定ファイルを追加することでポートレット内からアクセスすることができます。

PALポータルのコンポーネントを利用したい場合は、ポータル拡張設定ファイル jetspeed-portlet.xml をポートレットの WEB-INF に配置して、利用したいコンポーネントを記述します。 jetspeed-portlet.xml の XSD ファイルについては、Appendix を参照してください。

たとえば、PageManager などのコンポーネントを呼び出したい場合は、jetspeed-portlet.xml は以下のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app
    xmlns="http://portals.apache.org/jetspeed"
    xmlns:js="http://portals.apache.org/jetspeed"
    xmlns:dc="http://www.purl.org/dc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="pal-admin" version="1.0"
    xsi:schemaLocation="http://portals.apache.org/jetspeed
        http://portals.apache.org/jetspeed-2/2.1/schemas/jetspeed-portlet.xsd">
  <dc:title>PAL Portal Administration Portlets</dc:title>
  <dc:creator>PAL Team</dc:creator>
  <js:services>
    <js:service name="UserManager"/>
    <js:service name="PageManager"/>
    <js:service name="PermissionManager"/>
  </js:services>
</portlet-app>
```

呼び出したいコンポーネントは、js:service で記述します。 これらのコンポーネントは PortalServices により管理され、利用可能なコンポーネント一覧は webapps/palportal/WEB-INF/assembly/jetspeed-services.xml で定義されています。 jetspeed-services.xml は以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>
  <!-- Portlet Services  -->
  <bean id="PortalServices"
    class="org.apache.jetspeed.services.JetspeedPortletServices" >
    <constructor-arg>
      <map>
        <entry key="SecurityProvider">
          <ref bean="org.apache.jetspeed.security.SecurityProvider"/>
        </entry>
        <entry key="PortletRegistryComponent">
          <ref bean="org.apache.jetspeed.components.portletregistry.PortletRegistry" />
        </entry>
```

```
<entry key="SearchComponent">
  <ref bean="org.apache.jetspeed.search.SearchEngine"/>
</entry>
<entry key="PAM">
  <ref bean="PAM" />
</entry>
<entry key="UserManager">
  <ref bean="org.apache.jetspeed.security.UserManager"/>
</entry>
<entry key="PageManager">
  <ref bean="org.apache.jetspeed.page.PageManager"/>
</entry>
<entry key="RoleManager">
  <ref bean="org.apache.jetspeed.security.RoleManager"/>
</entry>
<entry key="GroupManager">
  <ref bean="org.apache.jetspeed.security.GroupManager"/>
</entry>
<entry key="Profiler">
  <ref bean="org.apache.jetspeed.profiler.Profiler"/>
</entry>
<entry key="SSO">
  <ref bean="org.apache.jetspeed.sso.SSOProvider"/>
</entry>
<entry key="EntityAccessor">
   <ref bean='org.apache.jetspeed.components.portletentity.PortletEntityAccessComponent'/>
</entry>
<entry key="WindowAccessor">
   <ref bean='org.apache.jetspeed.container.window.PortletWindowAccessor'/>
</entry>
<entry key="ApplicationServerManager">
  <ref bean="org.apache.jetspeed.tools.pamanager.servletcontainer.ApplicationServerManager"/>
</entry>
<entry key="PortletFactory">
  <ref bean="portletFactory"/>
</entry>
<entry key="DeploymentManager">
  <ref bean="deploymentManager"/>
</entry>
<entry key='IdGenerator'>
  <ref bean='IdGenerator'/>
</entry>
<entry key='Powertools'>
  <ref bean='Powertools'/>
</entry>
<entry key="HeaderResource">
  <ref bean="org.apache.jetspeed.headerresource.HeaderResourceFactory"/>
```

```
    </entry>
    <entry key="TemplateLocator">
      <ref bean="TemplateLocator"/>
    </entry>
    <entry key="DecorationLocator">
      <ref bean="DecorationLocator"/>
    </entry>
    <entry key="DecorationFactory">
      <ref bean="DecorationFactory"/>
    </entry>
    <entry key="PermissionManager">
      <ref bean="org.apache.jetspeed.security.PermissionManager"/>
    </entry>
    <entry key="PortalStatistics">
      <ref bean="PortalStatistics"/>
    </entry>
    <entry key="PortalAdministration">
      <ref bean="PortalAdministration"/>
    </entry>
    <entry key="PreferencesProvider">
      <ref bean="org.apache.jetspeed.prefs.PreferencesProvider"/>
    </entry>
    <entry key="org.apache.jetspeed.container.session.PortalSessionsManager">
      <ref bean="org.apache.jetspeed.container.session.PortalSessionsManager"/>
    </entry>
    <entry key="SecurityAccessController">
      <ref bean="org.apache.jetspeed.security.SecurityAccessController"/>
    </entry>
    <entry key="PortletTrackingManager">
      <ref bean="org.apache.jetspeed.aggregator.PortletTrackingManager"/>
    </entry>
    <entry key ="PortalAuthenticationConfiguration">
      <ref bean="org.apache.jetspeed.administration.PortalAuthenticationConfiguration"/>
    </entry>
    <entry key ="PortalConfiguration">
      <ref bean="PortalConfiguration"/>
    </entry>
    <entry key="ImporterManager">
      <ref bean="importerCastorPageManager"/>
    </entry>
    <entry key="decorationContentCache">
      <ref bean="decorationContentCache"/>
    </entry>
    <entry key="portletContentCache">
      <ref bean="portletContentCache"/>
    </entry>
    <entry key="AuditActivity">
```

```
          <ref bean="org.apache.jetspeed.audit.AuditActivity"/>
        </entry>
        <entry key="JetspeedSerializerFactory">
          <ref bean="org.apache.jetspeed.serializer.JetspeedSerializerFactory"/>
        </entry>
      </map>
    </constructor-arg>
  </bean>
</beans>
```

ポートレット内での呼び出し方法は、PortletContext から取得します。 たとえば、PageManager は以下のように取得します。

```
PageManager pageManager = (PageManager) getPortletContext()
    .getAttribute(CommonPortletServices.CPS_PAGE_MANAGER_COMPONENT);
```

8.4 JSR 168 ポートレット

簡単なポートレットの作成

この節は、簡単なポートレットを作成するためのチュートリアルです。 ポートレットの開発者は以下のような順序で開発ができます。

ポートレットクラス

simplest/WEB-INF/classes ディレクトリに Simplest.java というファイルを以下のように作成してください。

```
public class Simplest extends javax.portlet.GenericPortlet
{
    public void doView(javax.portlet.RenderRequest request, javax.portlet.RenderResponse response)
            throws javax.portlet.PortletException, java.io.IOException
    {
        response.setContentType("text/html");
        response.getWriter().println("A very simple portlet.");
    }
}
```

以下のようにコマンドを入力してクラスをコンパイルしてください。

```
$ javac -cp ~/.maven/repository/org.apache.portals.jetspeed-2/jars/portlet-api-1.0.jar Simplest.java
```

portlet.xml

simplest/WEB-INF ディレクトリに portlet.xml ファイルを作成してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app id="simplest" version="1.0">
  <portlet id="Simplest">
    <portlet-name>Simplest</portlet-name>
    <display-name>Simple Display Name</display-name>
```

```
    <portlet-class>Simplest</portlet-class>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Simple Title</title>
      <short-title>The world's simplest portlet</short-title>
    </portlet-info>
  </portlet>
</portlet-app>
```

web.xml

simplest/WEB-INF ディレクトリに web.xml ファイルを作成してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
                         "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Simplest</display-name>
  <description>The world's simplest portlet</description>
</web-app>
```

WAR ファイル

simplest ディレクトリで、以下のようなコマンドを実行して、以上のファイルを war ファイルにまとめてください。

```
$ jar cvf ../simplest.war .
```

WAR ファイルの配備

war ファイルを webapps/palportal/WEB-INF/deploy にコピーしてください。 PALポータルは webapp を配備します。

ページに配置

作成したポートレットをページ上に配置してください。 配置する方法については、「管理ガイド」を参照してください。

9.1 PSML の概要

PSML の概要

PSML は Portal Structure Markup Language の頭文字を取ったものです。 PSML は PALポータル内のコンテンツの構造化と抽象化を行うために作成されます。 PSML はポータルページ上でポートレットがどのように集約され、レイアウトされ、装飾されるかを定義します。 ページレイアウトに関する仕様は、ポートレット API の守備範囲ではないことに注意してください。 ですので、PSML は PALポータル特有の実装です。 そして、PALポータルの PSML は Apache Portals Jetspeed 2 の PSML の仕様に従います。

PSML ファイルは、ページ、フォルダ、リンクや大域的なセキュリティ制限と関係する情報も保持します。 これらの主要な PSML の要素のそれぞれは、ファイルシステム上のディレクトリ構造内に別々のドキュメントとして保存されるか、ポータルのデータベース内に保存することもできます。 保存先は PALポータルのインストール時に選択することができます。

以下にポータルサイトのページ用の PSML ファイル（*.psml）の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<page>
  <!-- ページの情報 -->
  <title>Welcome to PALポータル</title>
  <metadata name="title" xml:lang="fr">Ma Premiere Page de PSML</metadata>
  <metadata name="title" xml:lang="es">iBienvenido a PALポータル!</metadata>
  <metadata name="title" xml:lang="hu">Köszönti a PALポータル!</metadata>

  <!-- ページの装飾 -->
  <defaults skin="orange" layout-decorator="tigris" portlet-decorator="tigris"/>

  <!-- ページフラグメント -->
  <fragment id="100393" type="layout" name="jetspeed-layouts::VelocityOneColumn">
    <fragment id="100939" type="portlet" name="j2-admin::LocaleSelector">
      <property layout="OneColumn" name="row" value="0"/>
    </fragment>
    <fragment id="100345" type="layout" name="jetspeed-layouts::VelocityTwoColumns">
      <property layout="OneColumn" name="row" value="1"/>
      <property layout="TwoColumns" name="sizes" value="33%,66%"/>
      <fragment id="100121" type="portlet" name="j2-admin::LoginPortlet">
        <property layout="TwoColumns" name="row" value="0"/>
```

```
        <property layout="TwoColumns" name="column" value="0"/>
      </fragment>
      <fragment id="100171" type="portlet" name="demo::UserInfoTest">
        <property layout="TwoColumns" name="row" value="0"/>
        <property layout="TwoColumns" name="column" value="1"/>
      </fragment>
    </fragment>
  </fragment>

  <!-- セキュリティ制限 -->
  <security-constraints>
    <security-constraints-ref>public-view</security-constraints-ref>
  </security-constraints>
</page>
```

以下に、ポータルサイトのフォルダ（folder.metadata）用の定義を行う PSML ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<folder>
  <!-- フォルダの説明 -->
  <title>Root Folder</title>
  <metadata name="title" xml:lang="fr">Répertoire racine</metadata>
  <metadata name="title" xml:lang="es">Carpeta raiz</metadata>

  <!-- フォルダ内のドキュメントの順序 -->
  <document-order>Jetspeed2.link</document-order>
  <document-order>Jetspeed2Wiki.link</document-order>
  <document-order>apache_portals.link</document-order>
  <document-order>apache.link</document-order>

  <!-- ポータルサイトのメニュー -->
  <menu name="page-navigations">
    <separator>
      <text>Top Pages</text>
      <metadata name="text" xml:lang="fr">Page haut</metadata>
      <metadata name="text" xml:lang="es">Páginas más populares</metadata>
    </separator>
    <options>/Administrative</options>
    <separator>
      <text>Profiled Pages</text>
      <metadata name="text" xml:lang="es">Páginas del Perfil</metadata>
    </separator>
    <options regexp="true">/p[0-9][0-9][0-9].psml</options>
    <separator>
```

```
      <text>Non Java Pages</text>
      <metadata name="text" xml:lang="es">Ejemplos sin java</metadata>
    </separator>
    <options>/non-java</options>
  </menu>

  <!-- セキュリティ制限 -->
  <security-constraints>
    <security-constraints-ref>public-view</security-constraints-ref>
  </security-constraints>
</folder>
```

以下に、ポータルサイトのリンク（*.link）用の PSML ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<link target="top">
  <!-- リンクの説明 -->
  <title>PALポータルHome Page</title>
  <url>http://portals.apache.org/jetspeed-2/</url>
  <metadata name="title" xml:lang="es">PALポータル</metadata>
</link>
```

以下に、ポータルサイトのページのセキュリティ（page.security）用の PSML ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<page-security>
  <!-- 大域的な管理者の制限 -->
  <security-constraints-def name="admin">
    <security-constraint>
      <roles>admin</roles>
      <permissions>view, edit</permissions>
    </security-constraint>
  </security-constraints-def>
  <global-security-constraints-ref>admin</global-security-constraints-ref>

  <!-- パブリックな制限 -->
  <security-constraints-def name="public-view">
    <security-constraint>
      <users>*</users>
      <permissions>view</permissions>
    </security-constraint>
```

```
  </security-constraints-def>
  <security-constraints-def name="public-edit">
    <security-constraint>
      <users>*</users>
      <permissions>view, edit</permissions>
    </security-constraint>
  </security-constraints-def>
</page-security>
```

ページ

<page> 要素は、ポータルサイトのページに関連する他の PSML 要素を保持するための単純な入れ物です。 この要素は、親であるフォルダを構成する適切なファイルシステムディレクトリ内に、'.psml' という拡張子を持つファイルとして存在します(PSML をファイルシステム保存した場合)。 2 つの有効な属性がページエレメントの属性として存在します。

属性	説明
hidden	ページがポータルサイトのサイトメニューまたは他のナビゲーション用の要素に表れるかどうかを指示するための真偽値の属性。
version	一般的な目的のバージョントラッキングのための属性。PALポータルでは現在使われていません。

<page> 要素は多くの他の PSML 要素を含みます。

要素	説明
title?	デフォルトのページタイトルを表すテキストを含む単純な要素。ページのタイトルは、その長い説明とみなされます。もしメニューテキスト用の短いタイトルが利用可能であれば、一部のデコレータでロールオーバー用のテキストとして使われます。もし指定されていなければ、PALポータルは page 要素を含むファイルのファイル名からタイトルを決めようと試みます。
short-title?	ページのデフォルトの短いタイトルのテキストを含む単純で省略可能な要素。短いタイトルは、もし利用可能であれば、一部のデコレータ内のメニューテキストとして使われます。もし指定されない場合は title のテキストが使われます。
defaults	ページとそのフラグメントのための装飾を指定します。defaults はページ毎に必要です。
fragment	フラグメントの階層構造のルート。全てのページにルートのフラグメントが必要です。
metadata*	省略可能の指定である、ページ用のロケール特有のタイトルと短いタイトル。

要素	説明
menu*	省略可能の指定である、ページ用の追加または上書きされた継承したメニュー定義の指定。
security-constraints?	省略可能の定義である、ページ用のインラインのセキュリティ制限の定義。もし指定されなければ、ページは親フォルダ内の有効なセキュリティ制限を継承します。

例: 「概要」で示した例を参照してください。

デフォルト

ページ内の〈defaults〉要素は、デフォルトのレイアウトデコレータと、デフォルトのポートレットデコレータを定義します。 デフォルトのレイアウトデコレータは、デコレータ属性を持たない全てのレイアウトフラグメントに対して適用されます。 デフォルトのポートレットデコレータは、デコレータ属性を持たない全てのポートレットフラグメントに対して適用されます。 defaults 要素には 3 つの有効な属性があります。

属性	説明
layout-decorator	ページをレンダリングするときに使うレイアウトデコレータの名前。この属性は必須です。
portlet-decorator	ページフラグメントをレンダリングするときに使うデフォルトのポートレットデコレータの名前。この属性は省略可能ですが、ほとんどいつも設定します。
skin	ページやフラグメントの表示を制御するためにデコレータ内で参照される一般的な目的のデコレータ属性。現時点では PALポータルでは使用していません。

例:

```
<page>
  ...
  <defaults skin="orange" layout-decorator="tigris" portlet-decorator="tigris"/>
  ...
</page>
```

レイアウトフラグメント

ページのレイアウト〈fragment〉要素は階層構造のコンテナです。 これはポートレットフラグメントと、階層化されたレイアウトフラグメントを保持するのに使われます。 ページのルートのフラグメントは、ページに 1 つしかポートレットがないときでも、レイアウト

フラグメントでなければなりません。 レイアウトフラグメントは親となるレイアウトフラグメントのレイアウトに従属します。 ですので、 "row" と "column" のレイアウトプロパティをサポートします。 加えて、レイアウトフラグメントは 'sizes' レイアウトプロパティもサポートします。 これは、複数列のレイアウトの列の割合をコントロールするのに使われます。 この要素には 3 つの必須の属性があります。

属性	説明
id	必須である id は、フラグメントを特定するのに使います。そして、サイト内で定義される全てのフラグメントに渡ってユニークでなければなりません。この値は PALポータルに対して透過的でなく、一意性を保証するためのどのような仕様にも従うことが可能です。フラグメントは PALポータルで内部的にキャッシュされる可能性があるので、既存のページに対する任意の編集が、PALポータルが行う変更を保証するために新しい id を必要とするかも知れません。
type	この必須の属性はレイアウトフラグメント全てで 'layout' と指定しなければなりません。
name	必須であるポートレット名は、フラグメントレイアウトを実装するのに使われます。ポートレット名は一般的に、portlet.xml ファイルで指定された 'portlet-app-id::portlet-id' の形を取ります。

レイアウト〈fragment〉要素はたくさんの他の PSML 要素を含みます。

要素	説明
fragment*	指定された fragment 要素は、レイアウトフラグメントによってレイアウトされるポートレットフラグメント、または階層化されたレイアウトフラグメントのどちらにも成り得ます。全ての子フラグメントにはこのレイアウトの位置を指定するための property 要素が必須です。レイアウトフラグメント内に指定されるフラグメントがない場合は結果としてコンテンツは生成されません。
property*	レイアウトフラグメントは省略可能で 'sizes' プロパティを持つことが可能です。これは複数列の形状の明細を指定するのに使うことが可能です。階層化されたレイアウトフラグメントは、親レイアウト内の位置を特定するために 'row' または 'column' またはその両方のプロパティも持つかもしれません。
security-constraints?	フラグメントとそのフラグメントの子フラグメントに対して、省略可能なインラインのセキュリティ制限を定義します。ページ、リンクの制約と違って、'view' パーミッションのみが制限されます。もし指定されなければ、フラグメントはページ内で有効なセキュリティ制限を継承します。

例:

```
〈page〉
  ...
  〈fragment id="100393" type="layout" name="jetspeed-layouts::VelocityOneColumn"〉
    〈fragment id="100939" type="portlet" name="j2-admin::LocaleSelector"〉
```

```
      <property layout="OneColumn" name="row" value="0"/>
    </fragment>
    <fragment id="100345" type="layout" name="jetspeed-layouts::VelocityTwoColumns">
      <property layout="OneColumn" name="row" value="1"/>
      <property layout="TwoColumns" name="sizes" value="33%,66%"/>
      <fragment id="100121" type="portlet" name="j2-admin::LoginPortlet">
        <property layout="TwoColumns" name="row" value="0"/>
        <property layout="TwoColumns" name="column" value="0"/>
      </fragment>
      <fragment id="100171" type="portlet" name="demo::UserInfoTest">
        <property layout="TwoColumns" name="row" value="0"/>
        <property layout="TwoColumns" name="column" value="1"/>
      </fragment>
    </fragment>
  </fragment>
  ...
</page>
```

ポートレットフラグメント

ポートレットの〈fragment〉要素は、ページのポートレットを識別するために使われます。 ポートレットフラグメントは、親レイアウトフラグメントのレイアウトの影響を受けます。 ですので、レイアウトで定められている 'row' と 'column' のレイアウトプロパティをサポートします。 この要素に対しては多数の有効な属性が存在します。

属性	説明
id	必須である id は、フラグメントを特定するのに使います。そして、サイト内で定義される全てのフラグメントに渡って、ユニークでなければなりません。この値は PALポータルに対して透過的でなく、一意性を保証するためのどのような仕様にも従うことが可能です。フラグメントは PALポータルで内部的にキャッシュされる可能性があるので、既存のページに対する任意の編集が、PALポータルが行う変更を保証するために新しい id を必要とするかも知れません。
type	この必須の属性は全てのポートレットフラグメントで 'portlet' と指定しなければなりません。
name	必須であるポートレット名は、フラグメントコンテンツを流し込むのに使われます。ポートレット名は一般的に portlet.xml ファイルで指定された 'portlet-app-id::portlet-id' の形を取ります。
skin	フラグメントの表示のコントロールのための、ポートレットデコレータ内で参照する可能性のある一般的な目的のデコレータ属性。現時点では PALポータルでは使っていません。

属性	説明
decorator	デフォルトのポートレットデコレータ名は、フラグメントのレンダリングに使われます。この属性は省略可能です。しかし、この属性が指定されない場合は、ページの defaults がデフォルトのポートレットデコレータを指定します。
state	フラグメントポートレットの初期状態; 現時点ではこの属性に対しては、唯一 'hidden' が有効な値です。

ポートレットの〈fragment〉要素は他の PSML 要素を含みます。

要素	説明
property*	親フラグメントは、親レイアウト内の位置を特定するために、'row' または 'column' またはその両方のプロパティを持てます。
title?	フラグメントポートレットのタイトルを含む、省略可能で単純なテキストの要素。portlet.xml でセットされたタイトルを上書きします。
preference*	フラグメントポートレットにたいして、設定される初期のユーザプリファレンスを指定します。portlet.xml 内で指定された、どのようなポートレットプリファレンスも上書きされます。
security-constraints?	フラグメントに対するインラインのセキュリティ制限を省略可能で定義します。ページ、フォルダの制約と違って、'view' パーミッションだけを制限できます。もし指定されない場合は、そのフラグメントは page 内で有効なセキュリティ制限を継承します。

例:

```
<fragment id="100393" type="layout" name="jetspeed-layouts::VelocityOneColumn">
  ...
  <fragment id="100939" type="portlet" name="j2-admin::LocaleSelector">
    <property layout="OneColumn" name="row" value="0"/>
  </fragment>
  ...
</fragment>
```

フラグメントのプロパティ

フラグメントの〈property〉要素はフラグメントのプロパティの名前を指定するのに使われます。 これらのプロパティは一般にレイアウトポートレットのパラメータとページ内のポートレットの位置を指定するために使われます。 プロパティ要素は 3 つの属性をサポートします。

属性	説明
layout（廃止予定）	プロパティが紐付いているフラグメントポートレットの識別名。この属性は互換性のために用意されています。レイアウトとポートレットフラグメントの厳密な階層構造は、どのような単一のフラグメントとそのプロパティも、 1 つのレイアウトポートレットの影響下にある可能性があることを意味します。
name	フラグメントプロパティの名前。PALポータルのレイアウトポートレットは 'row'、'column'、'sizes' プロパティをサポートします。
value	フラグメントプロパティの値。'row' と 'column' プロパティはゼロベースの指標値を受け取ります。'sizes' プロパティは HTML frameset タグの 'rows' と 'cols' 属性の書式（例えば '25%,75%' 等）を受け取ります。

例:

```
<fragment id="100876" type="portlet" name="j2-admin::LoginPortlet">
  ...
  <property layout="TwoColumns" name="row" value="2"/>
  <property layout="TwoColumns" name="column" value="1"/>
  ...
</fragment>
```

```
<fragment id="103456" type="layout" name="jetspeed-layouts::VelocityTwoColumns">
  ...
  <property layout="TwoColumns" name="sizes" value="33%,66%"/>
  ...
</fragment>
```

プリファレンス

ポートレットフラグメントの〈preference〉要素は、ポートレットプリファレンスを定義します。 これは、ポートレットアプリケーションの portlet.xml 内に、ポートレットの定義を何度も行う必要なく、ページ上のポートレットインスタンス用に、デフォルトのポートレットプリファレンスの設定を、簡単に行う手段を提供します。

プリファレンスの優先度: ユーザ定義 〉 フラグメント定義 〉 portlet.xml での定義

〈preference〉要素の属性は以下のようなものです。

属性	説明
name	プリファレンスの名前。
readOnly	ユーザがこのプリファレンス値を変えることができるかどうかを示す真偽値。

〈preference〉要素は以下の要素を含みます。

要素	説明
value+	指定したプリファレンスと関係する値を含む単純なテキスト要素。

例:

```
<fragment id="uhtemp-1012" type="portlet" name="demo::BookmarkPortlet">
  ...
  <preference name="Google" readOnly="false">
    <value>http://www.google.com</value>
  </preference>
  ...
</fragment>
```

フォルダ

〈folder〉要素は、ポータルサイトのフォルダと関連する他の PSML 要素を保持する単純な入れ物です。この要素は、関連するファイルシステムディレクトリ内の folder.metadata ファイルとして存在します。フォルダの要素には 2 つの有効な属性が存在します。

属性	説明
hidden	フォルダが、ポータルサイトのメニューや他のナビゲーション用の要素に、表示されるかどうかを示す真偽値の属性。
version	一般的な用途のバージョントラッキングの属性。現時点では PALポータルでは使われていません。

〈folder〉要素は多くの他の PSML 要素を含みます。

要素	説明
title?	デフォルトのフォルダのタイトルを含む単純な要素。フォルダのタイトルは、自身の長い説明とみなされます。もし短いタイトル名が利用可能であれば、一部のデコレータでロールオーバー用のテキストとして使われます。もし指定されていなければ、PALポータルはフォルダ要素を含むファイル名からタイトルを決めようと試みます。
short-title?	フォルダのデフォルトの短いタイトルのテキストを含む省略可能な単純要素。もし短いタイトルが利用可能であれば、一部のデコレータでメニューテキストとして使われます。もし指定されない場合は、title のテキストが使われます。
default-page?	フォルダのデフォルトページまたはサブフォルダのテキストを含む単純で省略可能な要素。デフォルトページは、フォルダが直接ポータルから参照されるときに使われます。このフォルダ内のどのようなページやフォルダを（親フォルダとして使われる「..」も含む）指定することも可能です。もしデフォルトページが設定されない場合は、「default-page.psml」が使われます。もし 「default-page.psml」が存在しない場合は、フォルダ内の最初のページがデフォルトになります。
document-order*	ページ、サブフォルダ、リンクといった構成要素のソートの順番を定義するのに使われる、構成要素のテキスト名を含む単純で省略可能な要素。名前にマッチする構成要素は、その要素が定義された順番に配置されます。他の構成要素は名前でソートされます。そして、メニューや他のリストでは、一致した構成要素の後に表示されます。正規表現による一致はサポートしません。
metadata*	省略可能の指定である、ロケール固有のフォルダのタイトルや短いタイトル。
menu*	省略可能の指定である、追加の、または継承したものを上書きするフォルダのメニュー定義。
security-constraints?	省略可能の定義である、フォルダ用のインラインのセキュリティ制限の定義。もし指定されなければ、フォルダは親フォルダ内の有効なセキュリティ制限を継承します。

例: 「概要」の例を参照してください。

リンク

〈link〉 要素は、ポータルサイトの外部のコンテンツを参照するのに使われるポータルリンクに関連づけられている、他の PSML 要素を保持するシンプルな入れ物です。この要素は、親フォルダと関連づけられている、適切なファイルシステムのディレクトリ内に存在する「.link」という拡張子のファイルとして存在します。この要素には 2 つの有効な属性があります。

属性	説明
target	省略可能の指定である、外部コンテンツを開くための対象となるフレームの名前。もし指定されなければ、リンクされたコンテンツはブラウザで開いているポータルと置き換わります。
version	一般的な用途のバージョントラッキングの属性。現時点では PALポータルでは使われていません。

<link> 要素は多数の他の PSML 要素を含みます。

要素	説明
title?	デフォルトのリンクタイトル用のテキストを含むシンプルな要素。リンクのタイトルは、その長い説明とみなされます。もしメニューテキスト用の短いタイトルが利用可能であれば、一部のデコレータでロールオーバー用のテキストとして使われます。もし指定されていなければ、PALポータルは link 要素を含むファイルのファイル名からタイトルを決めようと試みます。
short-title?	リンクのデフォルトの短いタイトルのテキストを含む単純で省略可能な要素。もし短いタイトルが利用可能であれば、一部のデコレータでメニューテキストとして使われます。
url	コンテンツの URL のための必須の要素。この要素のテキストは、指定されたブラウザの対象フレームをナビゲートするのに使われます。
metadata*	省略可能の指定である、ロケール固有のリンクのタイトルや短いタイトル。
security-constraints?	省略可能の指定である、リンクのためのセキュリティ制限。もし指定されなければ、リンクは親フォルダ内の有効なセキュリティ制限を継承します。

例: 「概要」の例を参照してください。

グローバルなページのセキュリティ

<page-security> 要素はグローバルなセキュリティ制限と、その定義を宣言するために使われる、他の PSML 要素を保持するためのシンプルな入れ物です。この要素は page.security ファイルとして存在し、常に PSML ファイルのシステムディレクトリのルートディレクトリに存在します。page-security 要素の有効な属性は 1 つだけ存在します。

属性	説明
version	一般的な用途のバージョントラッキングの属性。現時点では PALポータルでは使われていません。

<page-security> 要素は PSML 要素に関連する 2 つのセキュリティ制限を含みます。

要素	説明
security-constraints-def*	省略可能の定義である、セキュリティ制限のコレクションの名前。これらのセキュリティ制限はページ、フラグメント、フォルダ、リンク内で、そして再利用とセキュリティのメンテナンスを容易にするために、この要素内で参照されます。
global-security-constraints-ref*	サイト内の全てのセキュリティ制限を適用するための、省略可能の定義であるセキュリティ制限参照。それぞれのシンプルな要素のテキストは、このエレメント内のここで指定された名前のセキュリティ制限の定義を参照します。

例: 「概要」の例を参照してください。

PSML タイトルとメタデータ

ページ, フォルダ, リンクの〈metadata〉要素はロケール特有のタイトルと短いタイトルのテキストを定義するのに使われます。これらの要素は何度でも、含まれる PSML 要素内で表れるかもしれません。しかし、複数の値が 1 つのロケールにたいして指定してはいけません。PSML XML 文書は、普通は広範囲のキャラクタセットをサポートするために、UTF-8 エンコーディングで宣言されます。

name	メタデータテキストの名前。この名前は、ロケール特有のタイトルテキストを指定するための、'title' と 'short-title' のどちらかを指定してください。
xml:lang	メタデータテキストのためのロケール言語または言語／国セレクタ。標準の Java ロケール名が期待されます (ISO-639 と ISO-3166)。有効な値として 'en' や 'en_US' を含みます。

例:

```
<page>
  ...
  <metadata name="title" xml:lang="fr">Ma Premiere Page de PSML</metadata>
  <metadata name="title" xml:lang="es">¡Bienvenido a PALポータル!</metadata>
  <metadata name="title" xml:lang="hu">Köszönti a PALポータル!</metadata>
  ...
</page>
```

PSML セキュリティ制限

page、fragment、folder、link、page-security 要素内で現れる〈security-constraints〉、〈security-constraints-def〉、〈global-security-constraints-ref〉要素については、

本章の「セキュリティ」の節で記述されています。

PSML メニュー

page、folder 要素内で現れる〈menu〉要素は、本章の「メニュー」の節で記述されています。

9.2 セキュリティ

セキュリティについて

セキュリティ制約は、ページとフォルダに適用されます。セキュリティ制約は、ページとフォルダに対するアクセスを許可したり拒否したりします。制約は、以下の 4 つの場所の 1 つまたは全てで定義されます。

- グローバル: PSML ツリーのルートに存在する page.security ファイル内の宣言として定義
- フォルダ: 各ディレクトリにオプショナルに存在する folder.metadata ファイル内で定義
- ページ: 特定のページへのアクセスを制限するために PSML ファイル内で定義
- フラグメント: ページ内の特定のフラグメントに対するアクセスを制限するために PSML ファイルの中で定義

許可

許可は、ページまたはフォルダへのアクセスに対するパーミッション、承認、権限の授与、主体のリストのどれかに関係します。セキュリティ制約を与えるということは、 1 つ以上のパーミッションと組み合せた、 1 つ以上のセキュリティ主体のリストの関連付けを行うということです。制約を与えると、関連づけられたパーミッションのリストの通りになるようページまたはフォルダへのアクセスが許可されます。

否認

否認のセキュリティ制約は、 1 つ以上のセキュリティ主体と共に宣言されます。制約の否認は、与えられた主体のリストの通りになるよう、ページやフォルダに対するアクセスを禁止します。制約の否認は、制約の承認の前にリストアップされる必要があることに注意してください。

宣言型と参照型の制約

ページとフォルダのリソース制約が適用されるとき、制約は 宣言型 または 参照型 の制約のどちらかである可能性があります。宣言型の制約は、特定のページまたはフォルダのリソースが、適切に使われるために宣言されます。参照型の制約は、中央集権的なセキュリティ制約リソースである page.security ファイル内で宣言された制約を参照します。サイト毎かサブサイト毎に、任意のページやフォルダ内で参照される制約を宣言するために、 page.security が 1 つあります。

セキュリティ制約

セキュリティ制約は、PSML ファイル内、もしくはフォルダのメタデータファイル内、もしくはグローバルなセキュリティの宣言中にある XML 要素です。セキュリティ制限には name という属性が 1 つ存在します。セキュリティ制約は、以下の要素を持ちます。

- roles - カンマで区切られた 1 つ以上のロール主体のリスト、もしくは全てのロールを表す
- groups - カンマで区切られた 1 つ以上のグループ主体のリスト、もしくは全てのグループを表す
- users - カンマで区切られた 1 つ以上のユーザー主体、もしくは全てのユーザーを表す
- owner - 単一のユーザー主体
- permissions - カンマで区切られた 1 つ以上のパーミッション（view, edit, help）のリスト

最初の 4 つの要素（roles, groups, users, owner）は全て、承認されるもしくは拒否されるパーミッションを持つ主体を定義します。

パーミッション

パーミッションは、セキュリティ制限によって許可が与えられるポータルのモードです。パーミッションは許可を行うだけで、否認はしません。 view パーミッションは、オペレーティングシステムにおける read パーミッションと同様のものです。 edit パーミッションは、オペレーティングシステムにおける write パーミッションと同様のものです。help パーミッションは、他のポータルで info パーミッションとなっているものと同様のものです。

ロール

制約は、与えられたリソースへのパーミッションのセットを、 1 つ以上のロール主体に与えることができます。ロールは、承認されたロール主体（つまりそのユーザーがメンバーであるということ）のユーザーリストから得られます。もし承認されたユーザーが、リストされたロールのどれかのメンバーであるのなら、リソースに対するパーミッションが与えられます。

```
<security-constraint>
  <roles>adminstrator, manager</roles>
  <permissions>view, edit</permissions>
</security-constraint>
```

制約は、ロール主体のリソース全体に対するアクセスを拒否することも可能です。もし承

認されたユーザーが、リストされたロールのどれかのメンバーであるのなら、リソースに対する全てのアクセスが拒否されます。

```
<security-constraint>
  <roles>adminstrator, manager</roles>
</security-constraint>
```

グループ

制約は、与えられたリソースへのパーミッションのセットを、 1 つ以上のグループ主体に与えることができます。 グループは、承認されたグループ主体（つまりユーザーがメンバーであるグループ）のユーザーのリストから得られます。 もし承認されたユーザーが、リストされたグループのどれかのメンバーであるのなら、リソースに対するパーミッションが与えられます。

```
<security-constraint>
  <groups>accounting, development</groups>
  <permissions>view</permissions>
</security-constraint>
```

制約は、グループ主体のリソース全体に対するアクセスを拒否することもできます。もし承認されたユーザーが、リストされたグループのいずれかのメンバーであるのなら、リソースに対する全てのアクセスが拒否されます。

```
<security-constraint>
  <groups>accounting, development</groups>
</security-constraint>
```

ユーザー

制約は、与えられたリソースへのパーミッションのセットを、 1 つ以上のユーザー主体に与えることができます。現在のユーザーは、リソースに対するパーミッションを与えるためにカンマで区切られたリスト中にリストされる、主体の 1 つでなければなりません。

```
<security-constraint>
  <users>joey, deedee, johnny</users>
  <permissions>view, edit, help</permissions>
</security-constraint>
```

制約は、ユーザー主体の全てのリソースに対するアクセスを拒否することも可能です。もし承認されたユーザーがリスト内にあれば、全てのアクセスは拒否されます。

```
<security-constraint>
  <users>fred</users>
</security-constraint>
```

組み合せ

1 つ以上の種類の主体の集合に対してパーミッションを与えたり、拒否したりできることに注意してください。例えば、ここではロール（manager, developer）とグループ（QA と Research）と特定のユーザー（dilbert）に対して、view と edit のパーミッションを与えています。もし承認されたユーザーが、ここにリストされたロール、ユーザー、グループのどれかのメンバーであるのなら、このリソースに対するパーミッションが与えられます。

```
<security-constraint>
  <roles>hacker, coder, guru</roles>
  <groups>unix, linux, freebsd</groups>
  <users>betty, fred, barney, wilma</users>
  <permissions>view, edit</permissions>
</security-constraint>
```

制約は、主体の組み合せが、全てのリソースに対するアクセスを拒否することも可能です。もし承認されたユーザーが、リストされたグループやロールやユーザーのメンバーであれば、リソースに対する全てのアクセスは拒否されます。

```
<security-constraint>
  <roles>hacker, coder, guru</roles>
  <groups>unix, linux, freebsd</groups>
  <users>betty, fred, barney, wilma</users>
</security-constraint>
```

すべて *

全てを意味する *（アスタリスク）は、ロール、グループ、ユーザー、パーミッションの全てに適用可能です。

```
```

```
<security-constraint>
  <users>*</users>
  <permissions>*</permissions>
</security-constraint>
```

宣言型の制約とグローバルの制約

宣言型の制約は、サイトのルートにある page.security ファイル内で宣言されます。宣言型の制約は、 security-constraints-ref タグを使って、ページやフォルダ内で参照されます。グローバルな制約も宣言型の制約です。これらは、ルート PSML リポジトリ内の page.security ファイル内で定義され、見付かります。グローバルな制約との違いは、page.security ファイルのスコープ内（すなわちサイト）の全てのフォルダとページに、暗黙のうちに適用されることです。PALポータルをインストールすると、 1 つしか page.security ファイルは存在できないことに注意してください。

```
<security-constraints-def name="admin">
  <security-constraint>
    <roles>admin</roles>
    <permissions>view, edit</permissions>
  </security-constraint>
</security-constraints-def>
<global-security-constraints-ref>admin</global-security-constraints-ref>
```

デフォルトの制約

セキュリティ制約の宣言には、PALポータルのデフォルトの配備で作成されるものがあります。

制約名	与えられる対象	パーミッション	グローバルかどうか
admin	roles: admin	view, edit	yes
manager	roles: manager	view	no
users	roles: user, manager	view	no
public-view	users: *	view	no
public-edit	users: *	view, edit	no

フォルダの制約

フォルダのセキュリティ制約は、サイト内の各フォルダにオプショナルで存在する folder.metadata ファイル内の security-constraints リスト 内に記述されます。folder.metadata ファイルがない場合、もしくはそのファイル内にセキュリティ制約の記述がない場合は、フォルダは、サイトかサブサイトのルートフォルダまでディレクトリをたどって、親フォルダの制約を継承することに注意してください。以下に 2 つの例を示します。 1 つ目は参照型の制約であり、 2 つ目は宣言型の制約です。

```
<security-constraints>
  <security-constraints-ref>public-view</security-constraints-ref>
  <security-constraint>
    <groups>engineering</groups>
    <permissions>view</permissions>
  </security-constraint>
</security-constraints>
```

全てのセキュリティ制約は、 security-constraints 内に記述しなければなりません。

ページの制約

ページのセキュリティ制約は、 PSML ファイル内の security-constraints list に記述されます。これはオプショナルです。このファイルにセキュリティ制約の記述がない場合は、フォルダは、自身が存在するフォルダの制約を継承することに注意してください。ページのセキュリティ制約は、宣言型のセキュリティ制約と参照型のセキュリティ制約から作成されます。以下に、 2 つの例を示します。 1 つ目は参照型の制約、 2 つ目は宣言型の制約です。

```
<security-constraints>
  <security-constraints-ref>global-view</security-constraints-ref>
  <security-constraint>
    <groups>accounting</groups>
    <permissions>view, edit</permissions>
  </security-constraint>
</security-constraints>
```

全てのセキュリティ制約は、 security-constraints 内に記述しなければなりません。

フラグメントの制約

ページのセキュリティ制約と同様に、フラグメントのセキュリティ制約は、 PSML ファイル内の security-constraints リスト に記述されます。この記述は省略可能です。期待通り、セキュリティの制約のリストがない場合は、フラグメントは、自身が属するページの

制約を継承します。view パーミッションだけがフラグメントの制約に対してチェックされることに注意してください。他のパーミッションは含まれるページに対してのみテストされます。

Spring の設定

宣言型のセキュリティ制約は、デフォルトでページマネージャコンポーネントの Spring の設定で有効になります。以下に、 page-manager.xml という Spring の部品設定ファイルのデフォルトのページマネージャ bean の設定を示します。

```
<bean id="org.apache.jetspeed.page.PageManager"
      name="pageManager"
      class="org.apache.jetspeed.page.psml.CastorXmlPageManager">
      <constructor-arg index="0"><ref bean="IdGenerator"/></constructor-arg>
      <constructor-arg index="1"><ref bean="DocumentHandlerFactory"/></constructor-arg>
      <constructor-arg index="2"><ref bean="FolderHandler"/></constructor-arg>
      <constructor-arg index="3"><ref bean="PageFileCache"/></constructor-arg>
      <!-- permissions security enabled flag, default=false -->
      <constructor-arg index="4"><value>false</value></constructor-arg>
      <!-- 制約セキュリティモデルの有効フラグ、デフォルト true  -->
      <constructor-arg index="5"><value>true</value></constructor-arg>
</bean>
```

この例の 6 番目（index="5"）の真偽値のコンストラクタ引数が、"制約セキュリティ" モデルを有効にするかどうかの指定を行うものです。もし、宣言型のセキュリティ制約が有効でないのなら、全てのインライン、参照型、グローバルのセキュリティ制約は無視されます。

9.3 メニュー

メニューの宣言

メニューの宣言は、表示されているポータルページに新しいナビゲーション要素を追加したり、デフォルトのものをカスタマイズしたりするために使われます。これらの PSML 宣言は、page 要素と folder 要素の一部分です。他の PSML 要素と同様に、メニュー宣言は、親フォルダで定義された全てに加えて、ページ内で明示的に定義されるものが有効となります。このため、グローバルなサイトのメニューは、サイトのルートフォルダに関連づけられている PSML ファイル内で定義されます。特定のページやフォルダのメニューの定義は、親フォルダで定義されている同じ名前のメニューを上書きします。

ポータルのレイアウトデコレータは、ポータルのコンテンツを描画しているときに、名前でページメニューの宣言にアクセスします。メニューは、動的なナビゲーションペインや、ポータルページのタブや、パンくずリストや、プルダウンメニューを作成するのに使われます。ポータルサイトコンポーネントは、サイトのページ固有の PSML エレメントを持つ動的なメニュー定義の選択肢を実際に形成することを担当しています。例えば、グローバルなメニュー定義は、一般的に、ポータルページが構成されている間、次のような順序でイベントに関係します。

1. ポータルレイアウトデコレータは、名前でメニュー定義の要求を行います。
2. 定義は、現在のページに継承され、ルートフォルダの PSML に設定されます。
3. ポータルサイトコンポーネントは、現在のページのコンテキスト内のメニュー定義を解釈し、ページやフォルダやメニューの選択肢のリンクを設置します。
4. デコレータは、メニュー定義自身と算出されたメニューの選択肢を元に、ポータルのナビゲーションコンテンツにタイトルとテキストを描画し、メニューを表示します。

ポータルサイトコンポーネントによって、内部的にサポートされた共通のメニュー定義が存在し、サイトのページやフォルダの PSML 要素に明示的には定義の必要がありません。

- pages: ポータル上のページタブを定義するのに使われる、相対的なページメニュー。
- bredcrumbs: ページタブ以下にリンクの履歴を提供するのに使われる、ページへのパス。
- navigations: ポータルの横のナビゲーションパネルを定義するのに使われる、相対サブフォルダとルートレベルのリンクメニュー
- back: ポータルページタブ上の、以前のページへ 1 つ戻るためのリンクを定義するのに使われる、親フォルダのメニュー。

メニュー定義内で構築されるこれらは、期待どおり、サイトの PSML 内で同じ名前で宣言されるメニューに上書きされます。

メニューの定義

〈menu〉要素は、レイアウトデコレータもしくは他のメニュー内の階層化されたメニューによって使われるメニューを定義します。menu 要素には、たくさんの有効な属性が存在します。

属性	説明
name	テンプレートコードおよび(または)メニュー参照からの検索のためのメニュー名を指定します。この属性は、トップレベルのノードで必須で、階層化されたメニューでは無視されます。
options	メニューの指定に、ドキュメントとフォルダの深い階層が存在する場合、メニューのためのルートドキュメントのパスを指定します。この属性は、ページ、フォルダ、メニューの選択肢のリンクの指定を行うドキュメントパスも定義します。カンマで区切ったパスおよび（または）正規表現のパターンを、複数のオプションパスとして指定可能です。相対パスは、現在のページからの相対パスとして解釈されます。「~」や「@」といった特別なパターンを、現在のページを参照するために使うことができます。
depth	オプションフォルダからのドキュメントの深さを指定します（0 以下の深さの指定は無限大を表す）。メニューの選択肢は、それぞれの可視ページを表示するか、サイトへのリンクを行うために作成されます。フォルダはこの設定によって階層化したメニューに変換されます。
paths	ルートフォルダから指定された選択肢への順序付きのパスオプションを含むかどうかを指定する真偽値。この設定は「履歴」もしくは「パンくずリスト」メニューを生成するのに使います。
regexp	選択肢の値としてワイルドカードや正規表現が使われるかどうかを指定する真偽属性。ファイルシステムのコマンドラインの正規表現をサポートします。
profile	オプションの値を評価する際に使われるプロファイルロケータの名前を指定します。この属性は、オプションと階層化したメニューのためのデフォルトのプロファイルの値も設定します。「*」の指定は、全てのプロファイルロケータの名前を受け入れます。これは、親メニューが選択したプロファイル名を上書きするのに使うことができます。
order	マッチする選択肢の順序を判断するための、カンマで区切られたマッチするパターンの正規表現のリスト、もしくは正規表現のドキュメントパスの値。この属性は、どの選択肢の要素に対してもデフォルトの選択肢の値として適用されます。しかし、複数の選択肢の子供へのマッチを再度順序づけることはありません。もし指定されない場合、複数の選択肢は、含まれるフォルダドキュメントのものとして返される順序に含まれます。この属性にマッチしない選択肢のパスは、順序づけられたものの後に追加されます。
skin	メニューのためのレイアウトのヒントを定義する省略可能なデコレータ。この属性は、選択肢や階層化したメニューのデフォルトスキンの値としても使われます。このヒントは、現時点では PALポータルのデコレータでは使われていません。

〈menu〉要素は、多数の他のメニュー定義の PSML 要素を含みます。タイトルとメタデータ要素を除いて、これらの要素の相対的な順序は、レイアウトデコレータがそれらを提示す

る順序を決定します。

要素	説明
title?	メニュー用のデフォルトのロケール非依存なタイトルを指定する単純な要素。メニューのタイトルは、その長い記述とみなされます。そして、もしメニュー用の短いタイトルが利用可能である場合、デコレータによってはロールオーバー用のテキストとして使う場合があります。もし指定されない場合、メニューの名前が使われます。
short-title?	メニュー用のデフォルトのロケール非依存な短いタイトル。短いタイトルは、もし利用可能である場合、デコレータによってはメニューのテキストとして使う場合があります。もし指定されない場合、タイトルのテキストが使われます。
metadata*	追加のロケール特有のタイトルと短いタイトルを指定します(省略可)。
options*	この順序付きのメニュー要素は、このメニュー定義のためのコンテンツ要素を指定します。
separator*	このメニュー定義内で、インラインで含まれるテキスト行を指定するために使われる、順序付きのメニュー要素。
menu*	このメニュー定義に含まれる、階層化されたメニューを定義するのに使われる、他の順序付きメニュー要素。
include*	このメニュー定義に含める、他のメニューの階層化したメニュー、もしくは選択肢を指定します。含めるメニューの名前はこの順序付きのメニュー要素のテキストです。
exclude*	このメニュー定義から除外する、他のメニューの選択肢と階層化したメニューを指定します。除外するための要素に含まれるメニューの名前は、この順序付きのメニュー要素のテキストです。

例:

```
<menu name="simple">
    <options>/some-top-page.psml,/custom/some-other-page.psml</options>
</menu>
```

```
<menu name="top-2-levels" options="/" depth="2" skin="dhtml-pull-down"/>
```

```
<menu name="top-role-pages" regexp="true" options="/*" profile="roles"/>
```

```

```

```
<menu name="top-custom">
    <title>Top Menu</title>
    <metadata name="title" xml:lang="fr">Haut</metadata>
    <options regexp="true" profile="groups">/group-pages/*</options>
    <menu options="/" profile="page">
        <separator>
            <text>-- Top Pages --</text>
            <title>Top Pages</title>
        </separator>
        <options regexp="true">/*</options>
        <separator>
            <title>Custom Pages</title>
        </separator>
        <options depth="2">/custom-folder/</options>
    </menu>
    <exclude>top-role-pages</exclude>
    <separator>More Top Pages</separator>
    <include nest="true">top-role-pages</include>
</menu>
```

デフォルトの共通メニュー宣言のための定義は、内部的にポータルサイトコンポーネントによってサポートされます。

```
<menu name="pages" regexp="true" options="*.psml"/>
```

```
<menu name="breadcrumbs" options="~" paths="true"/>
```

```
<menu name="navigations">
  <separator>Folders</separator>
  <options regexp="true">./*/</options>
  <include>page-navigations</include>
  <separator>Additional Links</separator>
  <options regexp="true">/*.link</options>
</menu>
```

```
<menu name="back" options="../"/>
```

これらの定義のセパレータのテキストは、内部的に変換されることに注意してください。

メニューの選択肢の定義

〈options〉要素は、メニュー内に単一または複数の選択肢を定義します。この単純な要素の文字列は、ページ、もしくはフォルダ、もしくはリンクのメニューの選択肢を作るドキュメントのパスを指定します。複数の選択肢のパスを、パスおよび（もしくは）正規表現パターンをカンマで区切ったリストとして指定することも可能です。相対パスは、現在のページからの相対パスとして解釈されます。「~」や「@」といった特別なパターンを、現在のページを参照するために使うことができます。この要素は、メニュー要素とたくさんの属性を共有します。

属性	説明
depth	選択肢のフォルダからのドキュメントの深さを指定します（0 以下の深さは無限大を表します）。メニューの選択肢は、サイトへのそれぞれの可視ページまたはリンクを表示するように作成されます。フォルダは、この設定によって階層化したメニューに変換されます。
paths	ルートフォルダから指定された選択肢への順序付きのパスの選択肢を含むかどうかを指定する真偽値。この設定は「履歴」もしくは「パンくずリスト」メニューを生成するのに使います。
regexp	選択肢の値としてワイルドカードや正規表現が使われるかどうかを指定する真偽属性。ファイルシステムのコマンドラインの正規表現をサポートします。
profile	選択肢の値を評価する際に使われるプロファイルロケータの名前を指定します。この属性は、選択肢と階層化したメニューのためのデフォルトのプロファイルの値も設定します。「*」の指定は、全てのプロファイルロケータの名前を受け入れます。これは、親メニューが選択したプロファイル名を上書きするのに使うことができます。
order	マッチする選択肢の順序を判断するための、カンマで区切られたマッチするパターンの正規表現のリスト、もしくは正規表現のドキュメントパスの値。この属性は、どの選択肢の要素に対してもデフォルトの選択肢の値として適用されます。しかし、複数の選択肢の子供へのマッチを再度順序づけることはありません。もし指定されない場合、複数の選択肢は、含まれるフォルダドキュメントの順番で返される順序に含まれます。この属性にマッチしない選択肢のパスは、順序づけられたものの後に追加されます。
skin	メニューの選択肢のレイアウトのヒントを定義するデコレータ(省略可)。このヒントは、現時点では PALポータルのデコレータでは使われていません。

例:

```
〈menu〉
    ...
    〈options regexp="true" order="*.psml,*.link"〉
        /some-top-page.psml,/custom/some-other-page.psml,/*.link
```

```
    </options>
    ...
</menu>
```

メニューのセパレータの定義

<separator> 要素は、メニューに含めるセパレータを定義します。セパレータは、選択肢もしくは階層化したメニューが、メニューの定義内のこの要素以下に現れた場合のみ含まれます。セパレータのテキストは、この要素内のテキスト、もしくはテキストメニュー定義の要素内で指定することが可能です。セパレータ要素に記述可能な属性は 1 つしかありません。

属性	説明
skin	メニューセパレータのレイアウトのヒントを定義する省略可能なデコレータ。このヒントは、現時点では PALポータルのデコレータでは使われていません。

<separator> 要素は、たくさんの他のメニュー定義の PSML 要素を含みます。

要素	説明
title?	メニュー用のデフォルトのロケール非依存なタイトルを指定する単純な要素。セパレータのタイトルは、その長い記述とみなされます。そして、もしセパレータのテキストが利用可能である場合、デコレータによってはロールオーバー用のテキストとして使う場合があります。
text?	メニュー用のデフォルトのロケール非依存なテキストを指定する単純な要素。セパレータのテキストは必須で、この属性によって指定されようと、セパレータ要素内のテキストとして指定されようと、レイアウトデコレータがメニュー内に挿入するためのテキストとなります。
metadata*	ロケール特有の追加のタイトルとセパレータテキスト(省略可)。

例:

```
<menu>
    ...
    <separator>----------</separator>
    ...
    <separator>
        <text>-- Top 10 Pages --</text>
        <metadata name="text" xml:lang="fr">Haut Pages</metadata>
        <title>Top 10 pages as voted by the PALポータルusers!</title>
    </separator>
```

```
    ...
</menu>
```

メニューをインクルードするための定義

〈include〉要素は、選択肢または、他のメニューの階層化した メニューを含みます。含めるメニューの名前は、このエレメントのテキスト値として指定します。この要素に含めることのできる有効な属性は 1 つしかありません。

属性	説明
nest	指定したメニューが、階層化するためのものかどうかを制御する真偽値。もしこの属性に 'true' が指定された場合、含めるメニューはサブメニューとして階層化されます。そうでないのなら、全ての選択肢と指定したメニューの階層化したメニューは、このメニュー内にインラインで挿入されます。

例:

```
<menu>
    ...
    <include nest="true">navigations</include>
    ...
</menu>
```

メニューの除外を行うための定義

〈exclude〉要素は、選択肢もしくは、他のメニューの階層化した メニューを除外します。このオプションは主に、このメニューから、ポータルページ内の、他のメニューに既に表示されているメニューの選択肢を消去するために使われます。一致する選択肢またはメニューがメニュー定義内のこの要素より上に表示されている場合は、除外されます。除外するメニューの名前は、この単純な要素のテキスト値として指定します。

例:

```
<menu>
    ...
    <exclude>pages</exclude>
    ...
</menu>
```

ページレイアウトデコレータ

ページレイアウトデコレータは、ページが PALポータルパイプラインで描画されるとき、ポータルサイトコンポーネントからメニューを要求します。デコレータは、メニューが PSML で定義されることを期待します。カスタムメニューがポータルに表示される前に、デコレータは、カスタムメニューの描画を変更する必要があります。以下に、Velocity レイアウトデコレータの header.vm ファイルから抜き出した例を示します。

```
...
#set($pagesStandardMenu = $site.getMenu("pages"))
#if(!$pagesStandardMenu.empty)
  <div class="tabs">
#includeTabsNavigation($pagesStandardMenu $LEFT_TO_RIGHT)
  </div>
#end
...
```

このコードは、ポータルページ内の標準的なページのタブメニューを描画します。ポータルサイトコンポーネントは Velocity のコンテキスト変数 $site として表れます。空のメニュー定義のための判断は、空のメニューがページコンテンツ内に描画されないことを保証するために設計されています。最後に、デコレータマクロが、メニューコンテンツを展開するために実行されます。一般に、再帰的なマクロの起動で階層化したメニューを描画することをサポートするために、このアプローチを取ります。この場合、デコレータはページオプションだけを期待します。以下に、ページのタブメニューを描画するのに使われるマクロ実装を示します。

```
#macro (includeTabsNavigation $_menu $_orientation)
  <table border="0" cellpadding="0" cellspacing="0">
  <tr>
    #foreach($element in $_menu.elements.iterator())
      #if($element.elementType == "option")
        #set($tabTitle = $element.getTitle($preferedLocale))
        #set($tabName = $element.getShortTitle($preferedLocale))
        #if($_orientation == $LEFT_TO_RIGHT)
          #if($element.isSelected($site))
            <td class="LTabLeft" nowrap="true"> </td>
            <td class="LTab" align="center" valign="middle" nowrap="true"
title="$!tabTitle">${tabName}</td>
            <td class="LTabRight"  nowrap="true"> </td>
          #else
            #set($tabUrl = $jetspeed.getAbsoluteUrl($element.url))
            <td class="LTabLeftLow"  nowrap="true"> </td>
```

```
            <td class="LTabLow" align="center" valign="middle" nowrap="true" title="$!tabTitle"><a
href="$tabUrl">${tabName}</a></td>
            <td class="LTabRightLow"  nowrap="true"> </td>
          #end
        #end
      #end
    #end
  </tr>
  </table>
#end
```

このマクロは、ポータルサイトコンポーネントによって提供される、処理されたページメニューの選択肢全てに渡って反復されます。これは、表示されるタブそれぞれを作るための HTML 要素のテーブルデータを描画します。メニュー全体は、単一の行の HTML テーブルとして描画されます。

以下に、ポータルサイトコンポーネントオブジェクトがレイアウトデコーレータに表示する、宣言型のメニューのインタフェースを作るためのプライマリのインタフェースのリストを以下に挙げます。

- org.apache.jetspeed.portalsite.Menu
- org.apache.jetspeed.portalsite.MenuElement
- org.apache.jetspeed.portalsite.MenuOption
- org.apache.jetspeed.portalsite.MenuSeparator
- org.apache.jetspeed.portalsite.PortalSiteRequestContext>

ポータルサイトコンポーネント

PALポータルのポータルサイトコンポーネントには、 2 つの密接に関係した役割があります。

1. ユーザからのポータルのリクエスト URL を、サイト内の PSML ページもしくはフォルダ要素にマップします。
2. サイトでユーザが利用可能なものを反映した、サイトのナビゲーションメニューと、メニューの選択肢を構築します。

一見、これらの機能は、PSML サイト定義を検索し、走査するだけの、比較的シンプルなものであるように見えます。しかし、一度プロファイラーの集約とセキュリティ制約のフィルタリングの複雑さが採り入れられると、マッピングはもはや単純ではなくなります。

メニューの選択肢は、PSML サイトのポータルサイトコンポーネントがマッチさせるビューと一致していることに注意することが重要です。メニュー要素の指定するドキュメントのパスは、PSML フォルダやページを指し示すために使われる物理パスではありません。代わりに、ポータルリクエストを反映するパスは、メニューの選択肢として含める PSML 要素を選択するのに使われます。このように、プロファイリングとセキュリティは、あたかも

実行中のリクエスト URL を逆にマップするかのように、メニューの選択肢の集団に自動的に含まれます。

10.1 プロファイラーの概要

プロファイラーの概要

PALポータルのプロファイラーは、ポータルのリソースの場所に関するルールベースのエンジンです。 PALポータルでは、プロファイラーは以下のような種類のポータルリソースの場所を決定します。

- PSML ページ
- フォルダ
- メニュー

リクエストがポータルで受信されたとき、プロファイラーは実行時パラメータや状態（リクエストパラメータ、HTTP ヘッダー、セッションの属性など）の正規化された値に基づき、リクエストをリソースと結びつけます。 プロファイラーはプロファイラーバルブ内のPALポータルのリクエスト処理パイプラインの中で呼び出されます。 このバルブは、リクエストコンテキストが既にポータルのリクエストやレスポンス、機能、言語、ユーザ情報と関連づけられている必要があります。 実行時のパラメータは、プロファイラーがポータルのリソースを位置を決定するのに使われる判断規準を作ります。

10.2 プロファイリングルール

プロファイリングルール

概要

プロファイリングルールは、特定のリソースの場所を解決するリクエストを評価するときに使われる判断規準のリストを定義します。 プロファイリングルールはプロファイラーが使用します。 これは、既知のポートレットリクエストデータ用の、細分化した判断規準によるポータルリソースの位置を決定するためです。 ルールは与えられた順序に適用される判断規準の順序つきのリストから構成されます。 このルールの順序に従って、プロファイリングエンジンは最少のリソースの判断規準とみなせるまで、少ない記述の設定によるアルゴリズムで、それぞれのルールの判断規準を適用します。 全ての条件が適用されつくした場合、ルールは失敗となりフォールバックリソースが要求されます。

ルール規準

ルール規準はプロファイルのプロパティの場所を示すためのテンプレートです。 デフォルトの実装では、リソースを特定する URL、MIME タイプ、言語の設定に基づく案で定義されたプロファイリングポリシーを持っています。 もっと複雑な実装を行うには、Cookie や IP アドレスの範囲や統計的なリソースの利用解析やサーブレットや EJB 内部のビジネスのルールのような、リソースとマッピングされる他の入力が必要となります。

ルール規準のリゾルバ

リゾルバ	説明
request	リクエストパラメータ名とのマッチングによる解決
session	セッションの属性名とのマッチングによる解決
request.session	最初にリクエストの属性名とのマッチングを行ってから、セッションの属性名とのマッチングを行って解決する
hard.coded	ハードコードされた値による解決
group.role.user	フォールバックコントローラ: 現在のユーザーグループ、次にロール、次にユーザーの所有するフォルダ以下のリソースを探す
user	現在のユーザーのディレクトリ内のリソースとマッチさせる

リゾルバ	説明
group	現在のグループのディレクトリ内のリソースとマッチさせる
role	現在のロールのディレクトリ内のリソースとマッチさせる
mediatype	リクエストコンテキスト内のメディアタイプとマッチさせる
country	リクエストコンテキスト内の国とマッチさせる
user.agent	リクエストコンテキスト内のユーザーエージェントとマッチさせる
language	リクエストコンテキスト内の言語とマッチさせる
roles	ロールのフォールバック
path	指定されたパスとマッチする
page	指定されたページとマッチする
path.session	指定されたパスまたはセッションの値とマッチする
user.attribute	指定されたリクエスト属性の値とマッチする
navigation	現在のナビゲーションパスを変える

11.1 ユーザー属性の概要

ユーザー属性の定義

ポートレット仕様はどのようにしてポートレットアプリケーションがユーザー属性を使用できるのかについて定義をしています。 たとえば、JSR 168 の PLT.17.1 では、属性に関して portlet.xml で以下のように定義されます。

```
<portlet-app version="1.0" xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
   <user-attribute>
      <description>User Given Name</description>
      <name>user.name.given</name>
   </user-attribute>
   <user-attribute>
      <description>User Last Name</description>
      <name>user.name.family</name>
   </user-attribute>
   <user-attribute>
      <description>User eMail</description>
      <name>user.home-info.online.email</name>
   </user-attribute>
   ...
</portlet-app>
```

このように属性が定義されると、ポートレットは PortletRequest インタフェースで定義された USER_INFO 定数を使って PortletRequest から変更できない Map としてログインしているユーザーの属性値にアクセスすることが可能になります。

```
Map userInfo = (Map)request.getAttribute(PortletRequest.USER_INFO);
String givenName = (userInfo!=null) ? (String)userInfo.get("user.name.given") : "";
String lastName = (userInfo!=null) ? (String)userInfo.get("user.name.family") : "";
String email = (userInfo!=null) ? (String)userInfo.get("user.home-info.online.email") : "";
```

ポートレット仕様で定義されないのは、ポータルが定義されたユーザー属性をユーザーの具体的な属性にどのようにマッピングするかです。

ユーザー属性のマッピング

PALポータルは具体的なユーザー属性を定義したり、そのユーザー属性へのアクセスを定義したりするためにとても柔軟な方法を提供しています。

ユーザー属性は PALポータルが情報を格納しておくためのデータベースとしている User Preferences を使って格納されます（これは PALポータルの殆んどのコンポーネントと同じ方法でカスタマイズできます)。 ユーザー属性の実装は PALポータルの java.util.prefs.Preferences の実装を利用しています。 ユーザー属性は User preferences 内の特定のノード下に格納されます。 そして任意の名前の属性を自由に含むことができます。 これらの具体的なユーザー属性は portlet.xml 内で以下のような 2 つの方法で定義されたユーザー属性にマップされます。

1.属性名との完全一致の使用

2.jetspeed-portlet.xml 内で定義されるカスタムマッピングの使用

カスタムのユーザー属性マッピング

もし対象とするポータルとして PALポータルを利用して新しいポートレットアプリケーションを書いたのなら、ポータル内でユーザー属性と一致する User Attributes を定義することは簡単でしょう。
しかし、もし既に存在するポートレットアプリケーションを PALポータル上に配備したいのなら、ポートレットアプリケーションに必要な属性名と PALポータルの User Preferences に格納される具体的な属性名の間のミスマッチがあるかもしれません。

jetspeed-portlet.xml に PALポータル特有の設定やカスタマイズを記述することができます。 このファイルの配備は PALポータルで必須ではありません。 しかし、もしポートレットアプリケーションの war ファイル内の WEB-INF フォルダにこのドキュメントが見つかれば、処理が行われます。 PALポータル特有の設定は "http://portals.apache.org/jetspeed" 名前空間を使って定義します。

ユーザー属性のマッピングはカスタムのユーザー属性名を定義する "name" エレメントと、その属性名にマップされる具体的な属性名を定義する"name-link" エレメントを含む "user-attribute-ref" エレメントによって定義されます。

```
<portlet-app version="1.0"
   xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
   xmlns:js="http://portals.apache.org/jetspeed">
   <js:user-attribute-ref>
      <js:name>user-name-given</js:name>
      <js:name-link>user.name.given</js:name-link>
   </js:user-attribute>
   <js:user-attribute-ref>
      <js:name>user-name-family</js:name>
      <js:name-link>user.name.family</js:name-link>
```

```
   </js:user-attribute>
   ...
</portlet-app>
```

先の例のようなカスタムのマッピングを使って、ポートレットは次のようにユーザー属性にアクセスできます。

```
Map userInfo = (Map)request.getAttribute(PortletRequest.USER_INFO);
String givenName = (userInfo!=null) ? (String)userInfo.get("user-name-given") : "";
String lastName = (userInfo!=null) ? (String)userInfo.get("user-name-family") : "";
String email = (userInfo!=null) ? (String)userInfo.get("user.home-info.online.email") : "";
```

カスタムのマッピングが定義されていない email 属性も完全一致でのアクセスが可能であることに注意してください。

12.1 Ajax の概要

Ajax の概要

PALポータルの XML Ajax API は、Ajax クライアントが、非同期のリクエストを PALポータルサービスに送ることができるようにするための XML ベースの API です。

典型的な使用例としては、

- ページのカスタマイズとポートレットの配置 - ページ上でのポートレットの移動、コピー、追加、削除。
- レイアウトの選択 - ページのレイアウトの変更（行や列の数の変更、列のサイズの変更）。
- テーマとデコレータの選択 - ページのテーマやページ上のポートレットのデコレータの変更。
- ポートレットのセレクタ - エンドユーザへのポートレットの選択リストを提供。
- セキュリティの設定 - リソース（ページ、ポートレット、フォルダ、リンク、フラグメント）もしくはポータル全体でのセキュリティ制約、またはポリシーの設定。
- メニューの設定 - PALポータルサイトのメニューの生成や編集。
- 全体的な管理 - 全体的な管理を行う全てのユースケースはまだ検討されていません。

保証されたアクセス

Ajax XML API リクエストの全ては、Ajax 用に登録されているのパイプラインを通して実行されます。 これは、通常の PALポータルコンポーネントのバルブ配列を用いて、Ajax リクエストを構成可能であることを意味します。 デフォルトの Ajax パイプラインは、全てのリクエストへのアクセスを保証します。 Ajax アクションのそれぞれは、自身のセキュリティ制約を持ちます。 ページを生成する全てのリクエストは、本来のアクションに依存する、編集もしくは表示モードの元で実行されます。

12.2 API

API

Ajax XML API は、単純な HTTP リクエストベースの API で、REST（Representational State Transfer）プロトコル上での通信を行います。 API へは、以下のようなポータル URL パスの "ajaxapi" サーブレット経由で HTTP を使ってアクセスします。

```
http://hostname/contextname/ajaxapi
```

リクエストパラメータとページ

リクエストパラメータは、リクエストした API の動作と、追加の API パラメータを指定します。 リクエストが参照しているページは、HTTP URL 内に含まれます。 ですので、もしページを変更するリクエストを作成したい場合は、ページは以下のような HTTP URL 内で指定します。

```
http://localhost:8080/jetspeed/ajaxapi/Public/db-browser.psml
```

標準の PALポータルプロファイリングを使ったページ検索アルゴリズムが、ページを決定します。 PALポータルのプロファイリングルールにより、ユーザーのためのページを検索するため、URL に含まれていなくても動作します。 たとえば、

```
http://localhost:8080/jetspeed/ajaxapi
```

という URL は、現在のユーザのデフォルトページを表示します。

リクエストパラメータは、API それぞれで固有です。 多くの場合、"action" パラメータというリクエストパラメータは必要になります（デフォルトの場合を除く）。 デフォルトのアクションは "action=getpage" です。 これは、プロファイリングルールにより検索された PSML ページの XML 表現を返します（PSML は XML 形式です）。 リクエストパラメータのそれぞれの例は以下の表を参照してください。

現時点で利用可能な API を示します。

ページの取得

コンポーネント: AjaxGetPage

説明: getpage は、ページマネージャから、PSML に保存されたページを取得します。

パラメータ:

API の例:

```
http://localhost:8080/jetspeed/ajaxapi/Public/content.psml
```

XML レスポンス:

```
<js>
<status>success</status>
<action>getpage</action>
  <page hidden="false">
  <defaults layout-decorator="tigris" portlet-decorator="tigris"/>
  <name>public.psml</name>
  <path>/Public/public.psml</path>
  <title>Public Share</title>
  <short-title>Public Share</short-title>
  <metadata name="title" xml:lang="es">Carpeta compartida</metadata>
  <fragment id="ps-1000" type="layout" name="jetspeed-layouts::VelocityTwoColumns" decorator="">
    <fragment id="ps-1001" type="portlet" name="rss::RSS" decorator="">
      <property name="row" value="0"/>
      <property name="column" value="0"/>
    </fragment>
       <fragment id="ps-1002" type="portlet" name="demo::BookmarkPortlet" decorator="">
      <property name="row" value="1"/>
      <property name="column" value="1"/>
    </fragment>
       <fragment id="ps-1003" type="portlet" name="jsf-demo::CalendarPortlet" decorator="">
      <property name="row" value="0"/>
      <property name="column" value="1"/>
    </fragment>
       <fragment id="P-1080bff9b03-10000" type="portlet" name="jsf-demo::CalendarPortlet"
decorator="">
      <property name="row" value="1"/>
      <property name="column" value="0"/>
    </fragment>
  </fragment>
</page>
</js>
```

絶対位置で移動

コンポーネント: AjaxMovePortletAbsolute

説明: ページ上のポートレットの絶対位置をリクエストパラメータで指定した row と col に従って移動する。

パラメータ:

API の例:

```
http://localhost:8080/jetspeed/ajaxapi/Public/public.psml?action=moveabs&id=ps-1003&row=0&col=1
```

XML レスポンス:

```
<js>
   <status>success</status>
    <action>moveabs</action>
    <id>ps-1003</id>
        <old_position>
      <col>1</col>
      <row>1</row>
    </old_position>
        <new_position>
      <col>1</col>
      <row>0</row>
    </new_position>
</js>
```

移動

コンポーネント: AjaxMovePortletLeft, AjaxMovePortletRight, AjaxMovePortletUp, AjaxMoveDown

説明: アクションを指定して、ページ上のポートレットを相対的なある位置から移動する。

パラメータ:

API の例:

```
http://localhost:8080/jetspeed/ajaxapi/Public/public.psml?action=movedown&id=ps-1003
```

XML レスポンス:

```
<js>
   <status>success</status>
    <action>movedown</action>
    <id>ps-1003</id>
        <old_position>
      <col>1</col>
      <row>0</row>
    </old_position>
        <new_position>
      <col>1</col>
      <row>1</row>
    </new_position>
</js>
```

ポートレットの追加

コンポーネント:	AjaxAddPortlet
説明:	現在のページに新しいポートレットを追加します。ポートレットは指定する行と列の場所に追加することが可能です。もし、行も列も指定されない場合は、デフォルトでそれぞれ 0 に設定されます。
パラメータ:	

API の例:

```
http://localhost:8080/jetspeed/ajaxapi/Public/public.psml?action=add&id=jsf-demo::CalendarPortlet
```

XML レスポンス:

```
<js>
  <status>success</status>
  <action>add</action>
  <id>jsf-demo::CalendarPortlet</id>
  <new_position>
    <col>0</col>
    <row>0</row>
  </new_position>
</js>
```

ポートレットの削除

コンポーネント:	AjaxRemovePortlet
説明:	現在のページから新たにポートレットを削除する。
パラメータ:	

API の例:

```
http://localhost:8080/jetspeed/ajaxapi/Public/public.psml?action=remove&id=ps-1003
```

XML レスポンス:

```
<js>
  <status>success</status>
  <action>remove</action>
  <id>jsf-demo::CalendarPortlet</id>
  <new_position>
    <col>0</col>
    <row>0</row>
  </new_position>
</js>
```

ポートレットの取得

コンポーネント:	AjaxGetPortlets
説明:	ポートレットの取得は、現在のサブジェクトで利用可能な（ソートされた）ポートレットのリストを返します。ポートレットリストはフィルタリングされ、現在のサブジェクトで表示可能なポートレットが返されます。PALポータルセキュリティポリシー（JAAS）は、このフィルタリングを実行します。ポートレット（のリスト）は、それぞれのポートレットの名前、表示名、説明の記述された XML フォーマットで返されます。
パラメータ:	

API の例:

```
http://localhost:8080/jetspeed/ajaxapi?action=getportlets
```

XML レスポンス:

```
<js>
<status>success</status>
<action>getportlets</action>
-
        <portlets>
<portlet name="demo::AttributeScopePortlet" displayName="Attribute Scope Demo"
description="$portlet.Description">
        </portlet>
<portlet name="demo::BookmarkPortlet" displayName="Bookmark Portlet" description="Bookmark
Portlet">
        </portlet>
<portlet name="demo::BookmarkPortletForXHTMLBasic" displayName="Bookmark Portlet for XHTML Basic"
description="Bookmark Portlet for XHTML Basic">
        </portlet>
<portlet name="demo::CSSDemoPortlet" displayName="CSS Demo Portlet"
description="$portlet.Description">
        </portlet>
....
<portlet name="rss::RSS" displayName="RSS Portlet" description="RSS Portlet">
        </portlet>
<portlet name="rss::RomeRSS" displayName="Rome RSS Portlet" description="Rome RSS Portlet">
        </portlet>
</portlets>
</js>
```

Spring アセンブリ

AjaxRequestService は、Ajax リクエストを扱う Spring のコンポーネントです。これは、Ajax リクエストの特別な処理のために、Ajax パイプライン内でフックされます。以下が Spring アセンブリです。それぞれの API は Ajax サービス内で設定されます。

```
<bean id="AjaxRequestService" class="org.apache.jetspeed.ajax.AjaxRequestServiceImpl">
    <constructor-arg index="0">
        <map>
            <entry key="moveabs">
                <ref bean="AjaxMovePortletAbsolute"/>
            </entry>
            <entry key="moveleft">
                <ref bean="AjaxMovePortletLeft"/>
            </entry>
            <entry key="moveright">
                <ref bean="AjaxMovePortletRight"/>
            </entry>
            <entry key="moveup">
```

```
                <ref bean="AjaxMovePortletUp"/>
            </entry>
            <entry key="movedown">
                <ref bean="AjaxMovePortletDown"/>
            </entry>
            <entry key="add">
                <ref bean="AjaxAddPortlet"/>
            </entry>
            <entry key="remove">
                <ref bean="AjaxRemovePortlet"/>
            </entry>
            <entry key="getportlets">
                <ref bean="AjaxGetPortlets"/>
            </entry>
            <entry key="getpage">
                <ref bean="AjaxGetPage"/>
            </entry>
        </map>
    </constructor-arg>
    <constructor-arg index="1">
        <ref bean="AjaxVelocityEngine"/>
    </constructor-arg>
</bean>
```

13.1 バッチ処理の概要

バッチ処理の概要

ポータルシステムにおいて、ユーザー情報などを他のシステムから入出力しなければならない要件はよく起こります。 PALポータルでは、それを実現するため、大量のユーザー情報をバッチで処理可能な API を実装しています。 これにより、複雑なデータベーススキーマを操作など必要なく、一括で更新などが可能です。

13.2 ユーザー情報のバッチ処理

ユーザー情報のバッチ処理

ユーザー情報のバッチ更新サーブレットを導入することで、ウェブ経由のアクセスでユーザー情報の更新を行うことができます。 アクセスするだけで情報を更新できるので、シェルスクリプトや別なプログラムからアクセスすることで一括の更新等を実現できます。

利用方法

webapps/palportal/WEB-INF/web.xml に以下のサーブレット定義を加えます。

```
...
  <servlet>
    <servlet-name>UserManagerServlet</servlet-name>
    <servlet-class>jp.sf.pal.portal.servlet.UserManagerServlet</servlet-class>
  </servlet>
...
  <servlet-mapping>
    <servlet-name>UserManagerServlet</servlet-name>
    <url-pattern>/userManager</url-pattern>
  </servlet-mapping>
...
```

必要に応じて、セキュリティの設定をする必要があります。

アクセス方法

http://localhost:8080/palportal/userManager?name1=value1&... にアクセスすると対象の操作が実行されます。 以下の操作を実行できます。

- ユーザーの認証確認
- ユーザーの追加
- パスワードの更新
- ユーザーの削除
- ユーザー属性の更新

- ユーザー属性の削除

出力結果

以下の XML 形式で結果が出力されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<results>
  <status>文字列</status>
  <message>文字列</message>
  <result>
    <name>文字列</name>
    <value>文字列</value>
  </result>
</results>
```

各処理の説明

ユーザーの認証確認

指定されたユーザーが認証できるかを確認します。

URLで渡すパラメータ

リクエストパラメータ	値
action	authenticate
username	ユーザー名
password	パスワード

出力結果のstatus

値	説明
success	正常終了(ユーザーが存在して、パスワードも正しい場合)
authentication-failed	ユーザー認証に失敗した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った

ユーザーの追加

指定したユーザー名でユーザーを作成します。

URLで渡すパラメータ

リクエストパラメータ	値
action	create
username	ユーザー名
password	パスワード

出力結果のstatus

値	説明
success	正常終了(ユーザーが正常に作成された場合)
user-already-exists	ユーザーが既に存在していて、作成できない
server-error	ユーザー作成時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った

パスワードの更新

指定されたユーザーのパスワードを変更します。

URLで渡すパラメータ

リクエストパラメータ	値
action	update
username	ユーザー名
password	パスワード

出力結果のstatus

値	説明
success	正常終了(ユーザー情報が正常に更新された場合)
user-not-found	対象ユーザーが存在しない
server-error	ユーザー更新時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った

ユーザーの削除

指定されたユーザー名のユーザーを削除します。

URLで渡すパラメータ

リクエストパラメータ	値
action	update
username	ユーザー名
password	パスワード

出力結果のstatus

値	説明
success	正常終了(ユーザーが正常に削除された場合)
user-not-found	対象ユーザーが存在しない
server-error	ユーザー削除時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った

ユーザー属性の取得

指定されたユーザーのユーザー属性を取得します。

URLで渡すパラメータ

リクエストパラメータ	値
action	get-user-attribute
username	ユーザー名
key	キー

出力結果のstatus

値	説明
success	正常終了(ユーザー属性が正常に取得された場合)
user-not-found	対象ユーザーが存在しない
server-error	ユーザー削除時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った
value-is-null	キーに対応する値が null の場合

出力結果のresult

ユーザー属性の更新

指定されたユーザー名のユーザー属性を更新します。

URLで渡すパラメータ

リクエストパラメータ	値
action	update-user-attribute
username	ユーザー名
key	キー
value	値

出力結果のstatus

値	説明
success	正常終了(ユーザー属性が正常に登録または更新された場合)
user-not-found	対象ユーザーが存在しない
server-error	ユーザー削除時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った

ユーザー属性の削除

指定されたユーザー名のユーザー属性を削除します。

URLで渡すパラメータ

リクエストパラメータ	値
action	get-user-attribute
username	ユーザー名
key	キー

出力結果のstatus

値	説明
success	正常終了(ユーザー属性が正常に削除された場合)
user-not-found	対象ユーザーが存在しない
server-error	ユーザー削除時にサーバー側でエラーが発生した
invalid-parameter	リクエストパラメータが正しくない
excluded-user	操作の対象外ユーザーへの処理を行った
value-is-null	キーに対応する値が null の場合

14.1 jetspeed-portlet.xsd

jetspeed-portlet.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    targetNamespace="http://portals.apache.org/jetspeed"
    xmlns:tns="http://portals.apache.org/jetspeed"
    xmlns:dc="http://www.purl.org/dc"
    xmlns:js="http://portals.apache.org/jetspeed"
    xmlns:p="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">

  <xs:import namespace="http://www.purl.org/dc" schemaLocation="dublin-core.xsd"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xs:element name="portlet-app">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element name="security-constraint-ref" minOccurs="0" type="xs:string"/>
        <xs:group ref="tns:metadataGroup" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:custom-portlet-mode" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:custom-window-state" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:portlet" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="tns:services" minOccurs="0"/>
        <xs:element ref="tns:user-attribute-ref" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID"/>
      <xs:attribute name="version" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="portlet">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="portlet-name" type="xs:string"/>
        <xs:element name="security-constraint-ref" minOccurs="0" type="xs:string"/>
        <xs:group ref="tns:metadataGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
```

```
  </xs:complexType>
</xs:element>

<xs:element name="custom-portlet-mode">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="mapped-name" type="xs:string"/>
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="custom-window-state">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="mapped-name" type="xs:string"/>
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="user-attribute-ref">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="name-link" type="xs:string"/>
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="metadata">
  <xs:complexType mixed="true">
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute ref="xml:lang"/>
  </xs:complexType>
</xs:element>
<xs:element name="services">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="js:service"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="service">
```

```
    <xs:complexType>
      <xs:attribute name="name" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:group name="metadataGroup">
    <xs:choice>
      <xs:element ref="dc:creator"/>
      <xs:element ref="dc:description"/>
      <xs:element ref="dc:title"/>
      <xs:element ref="dc:type"/>
      <xs:element ref="dc:source"/>
      <xs:element ref="dc:right"/>

      <xs:element ref="dc:relation"/>
      <xs:element ref="dc:publisher"/>
      <xs:element ref="dc:language"/>
      <xs:element ref="dc:identifer"/>
      <xs:element ref="dc:format"/>
      <xs:element ref="dc:contributor"/>
      <xs:element ref="dc:coverage"/>
      <xs:element ref="dc:subject"/>
      <xs:element ref="js:metadata"/>
    </xs:choice>
  </xs:group>
</xs:schema>
```