ポートレットを使ってみよう

株式会社エヌツーエスエム
菅谷信介

自己紹介

- 名前: 菅谷信介
- 所属: N2SM, Inc. (オープンソース推進室長)
- オープンソース活動:
  - Apache Portals(Jetspeed, Portals Bridges) PMC
  - Seasarプロジェクトコミッタ(S2Container, Teeda, SAStruts..)
  - Portal Application Laboratory(PAL)プロジェクト運営
  - SSO Proxy プロジェクト運営

  などなど・・・
- ブログ: http://d.hatena.ne.jp/shinsuke_sugaya/

Agenda

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

本セッションでのゴール

ポータルを使って
ポートレットを作りたくなる

ポータル・ポートレットの概要

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

ポータルとポートレット

- ポータル：別々のシステムの情報を統合して、一元的に表示することで、システム全体の入り口になるシステム（EIPとか）
- ポートレット：ポータル上で一部のコンテンツを生成するアプリケーション（ガジェットとか）

歴史

- 各ベンダーの独自ポータルが散在（互換性なし）
- 2003年に JSR 168 の登場（ポートレットAPI1.0）
- 多くのポータルが JSR 168 に対応
- 2008年に JSR 286 の登場（ポートレットAPI2.0）

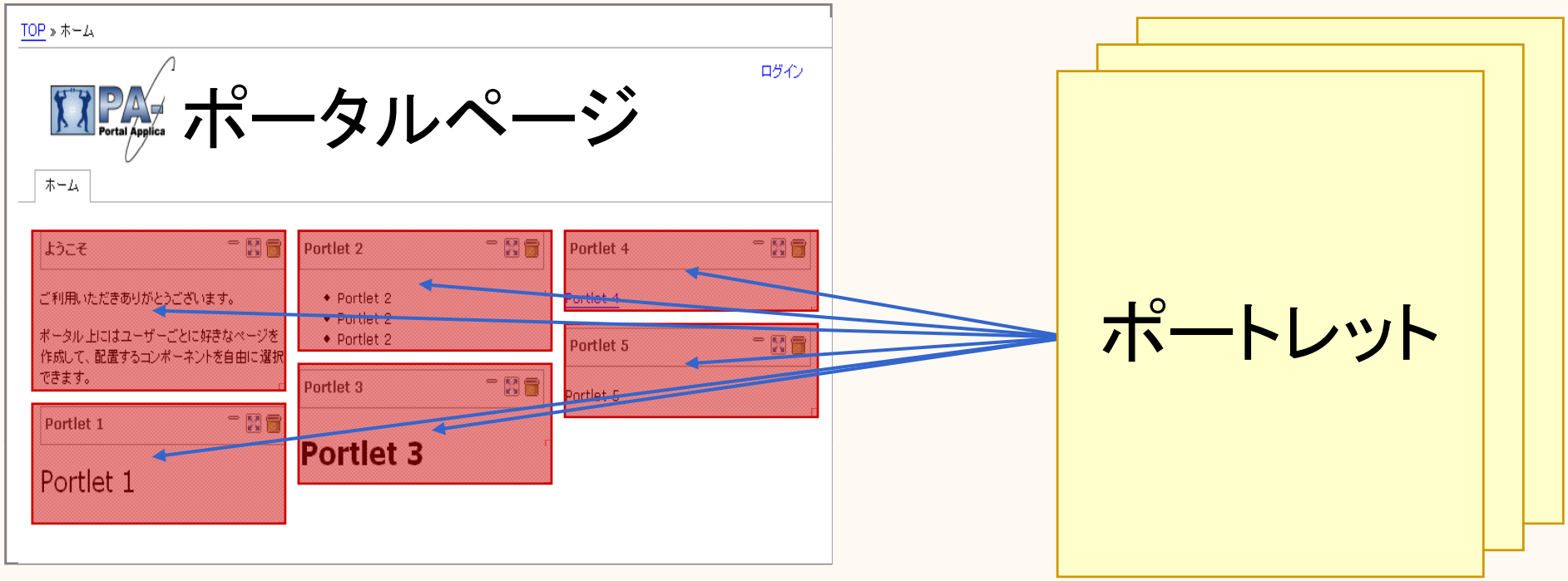
ポータル

- 商用製品
  - WebSphere Portal, Oracle AS Portal, WebLogic Portal, Sun Java System Portal, Interstage Portalworks, N2 Portal, …
- オープンソース製品
  - Jetspeed2, Pluto, PALポータル, JBoss Portal, Liferay, OpenPortal, iPoint Portal, uPortal, Gridsphere, eXo Portal, …

ポータル市場

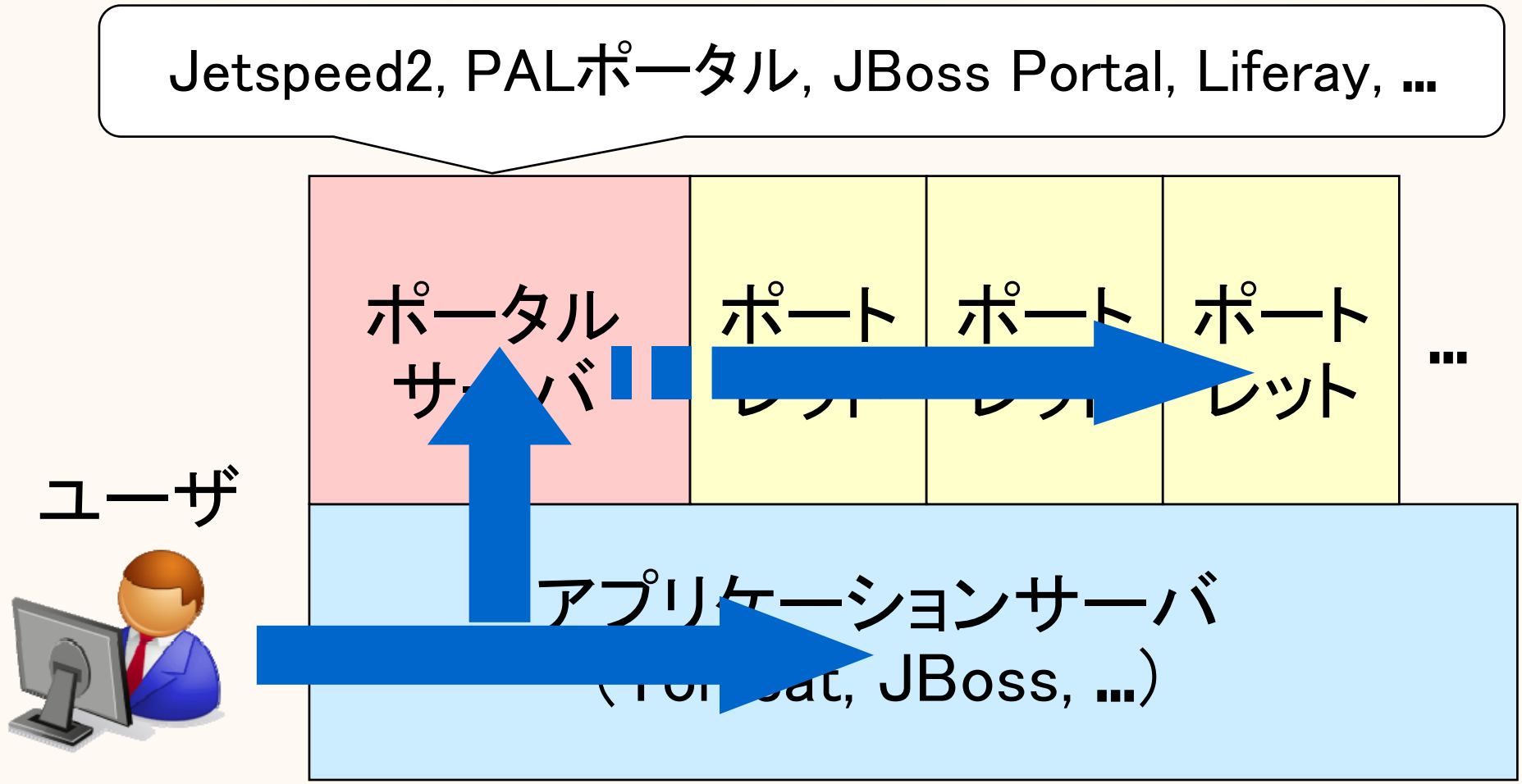
- 2007年の時点で、企業ポータル・ソフトウェア市場は12億ドルのライセンスと保守収益で、年率18.3%を占めるまで成長（IDC）
- 2012年までに企業ポータル・ソフトウェア市場が、20億ドルに成長するだろう（IDC）
- ポータル市場は、民間企業から官公庁へ、また大企業から中堅中小企業へと導入の裾野が拡大しており、前年比23.6%の高成長（IDC Japan）
- ～2011年の年間平均成長率19.8%と、最も成長が期待される市場（IDC Japan）

ポータルの使用イメージ



ページ内にポートレットを自由に配置できる

アーキテクチャ



ポータルを適用すると良い場面

- ユーザ認証があるウェブアプリ開発
  - ポータルの機能を利用できる（様々な認証に対応）
  - コンテンツの開発に集中できる
- 将来での拡張が求められる場合
  - 必要な拡張をポートレットとして追加できる
  - スモールスタートが可能
- 乱立するウェブアプリを統合したい場合
  - 入り口として導入する

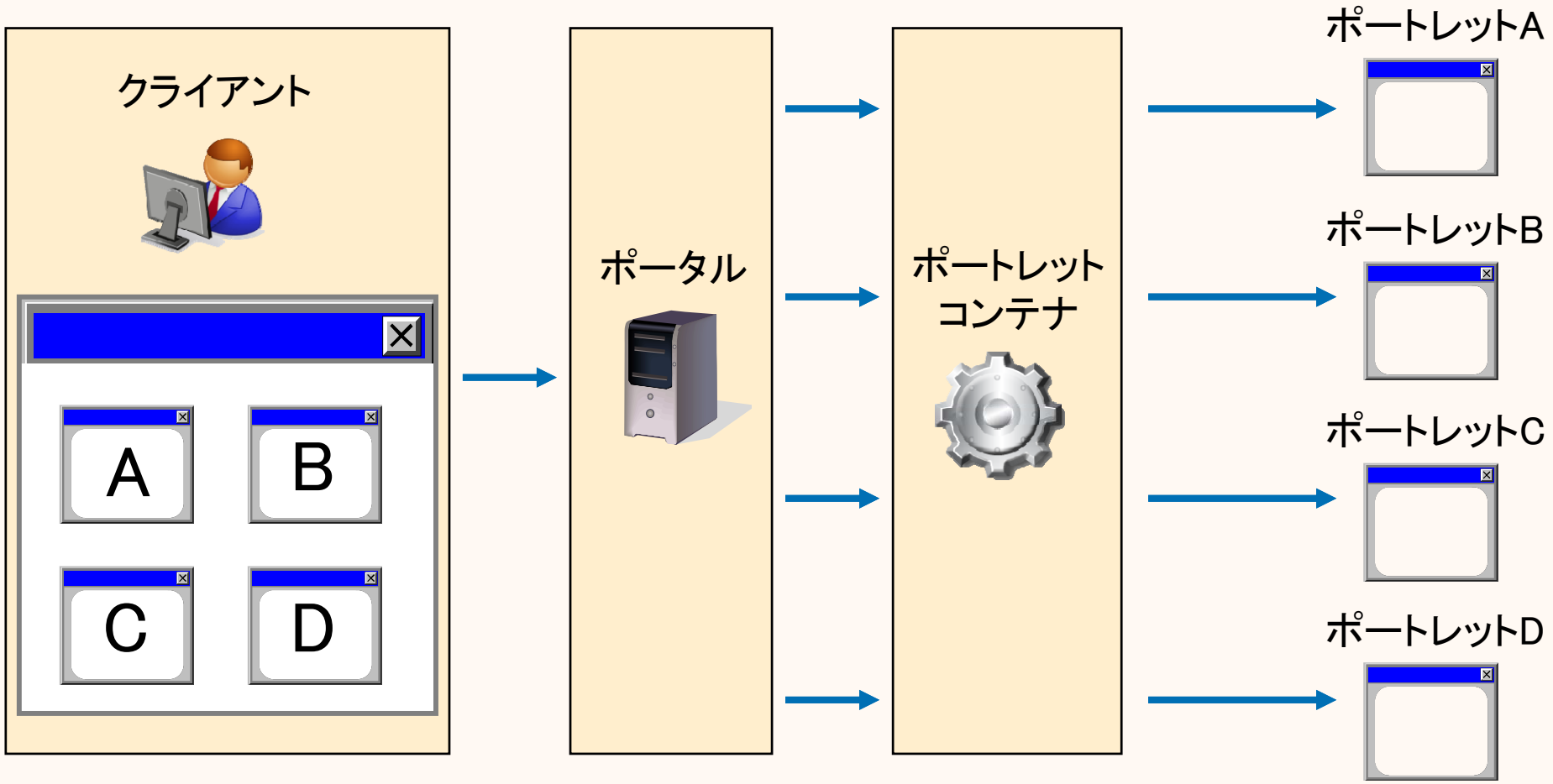
ポートレットAPI (JSR 168)

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

JSR 168 とは

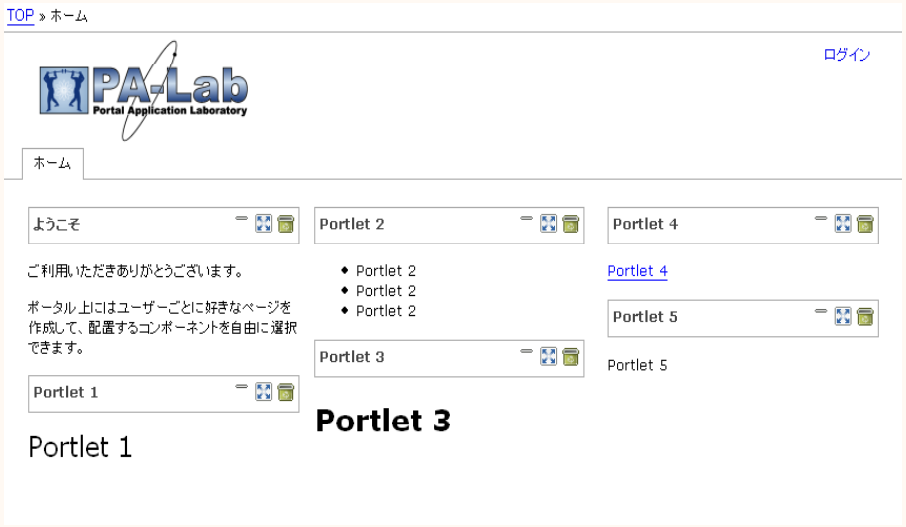
- JCPにより提供されるJavaポートレット仕様
- ポートレットの実行環境を定義
- コンテナとポートレット間のAPIを定義
- ポートレットのデータを保存する機能を提供
- サーブレットやJSPを呼び出し可能にする
- パッケージング方法を定義（配備を簡単に）
- ベンダー非依存

ポータル・ポートレットの概略図



ポータルの定義

- ポータルは1つのWebアプリケーション
- 情報システムのプレゼンテーション層を提供
- シングルサインオン(SSO)
- ユーザごとにパーソナライズ可能
- ページ内に異なるリソースからコンテンツ集約



ポートレットの定義

- JavaベースのWebコンポーネント
- 着脱可能な UI コンポーネント
- ページの一部(Fragment)を生成する
- ポートレットコンテナにより管理される
- JSR 168 ポートレットは JSR 168 準拠のポータルで稼動可能

ポートレットコンテナの定義

- ポートレットを実行し、必要な実行環境を提供する
- ポータルからのリクエストを受け取り、ポートレットを実行する
- ポートレットのライフサイクルを管理する
- 永続的なストレージを提供する(PortletPreference)
- ポートレットが生成したコンテンツを集約まではしない
- ポータルと組になって1つとしてパッケージされる場合もある

アーキテクチャ概要

- サーブレットの拡張仕様
- リクエストとレスポンスを渡される
- アクションと描画の分離
- ウィンドウ状態、モード、プリファレンス、ユーザ属性
- ポートレットはページの一部を生成
- 固定の URL に結びつかない

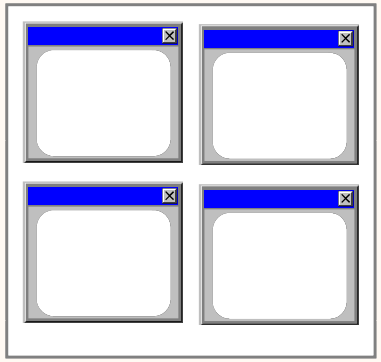
ライフサイクル

- サーブレットに似ている
- init()
- destroy()
- render() と processAction() ←ここが違う
- ポートレットコンテナにより管理
- VMごとにポートレット定義で1つのインスタンスを生成

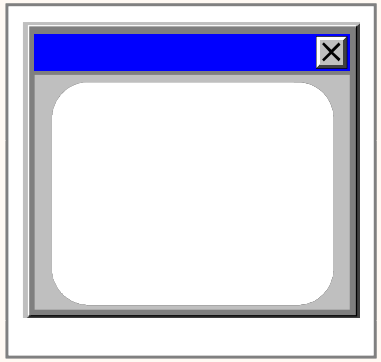
ウィンドウ状態

- 通常(Normal)
- 最大化(Maximized)
- 最小化(Minimized)
- カスタム状態 (ポータルによる)

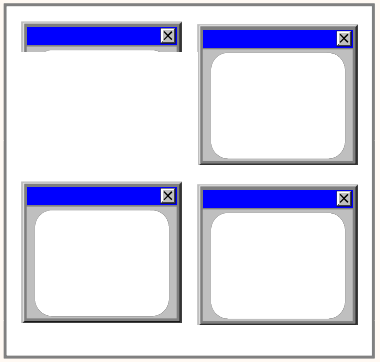
通常



最大化

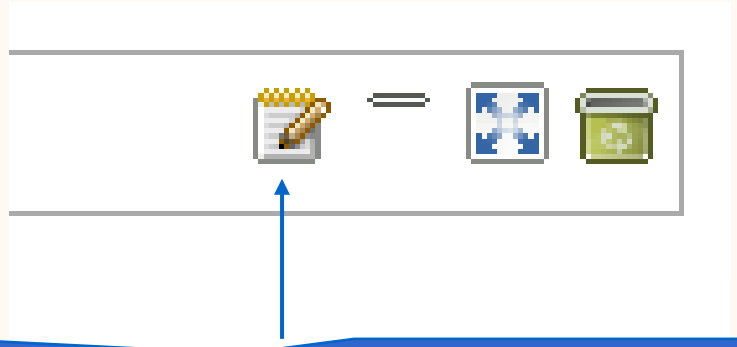


最小化



ポートレットモード

- 表示(View)
- 編集(Edit)
- ヘルプ(Help)
- カスタムモード(ポートレットによる)

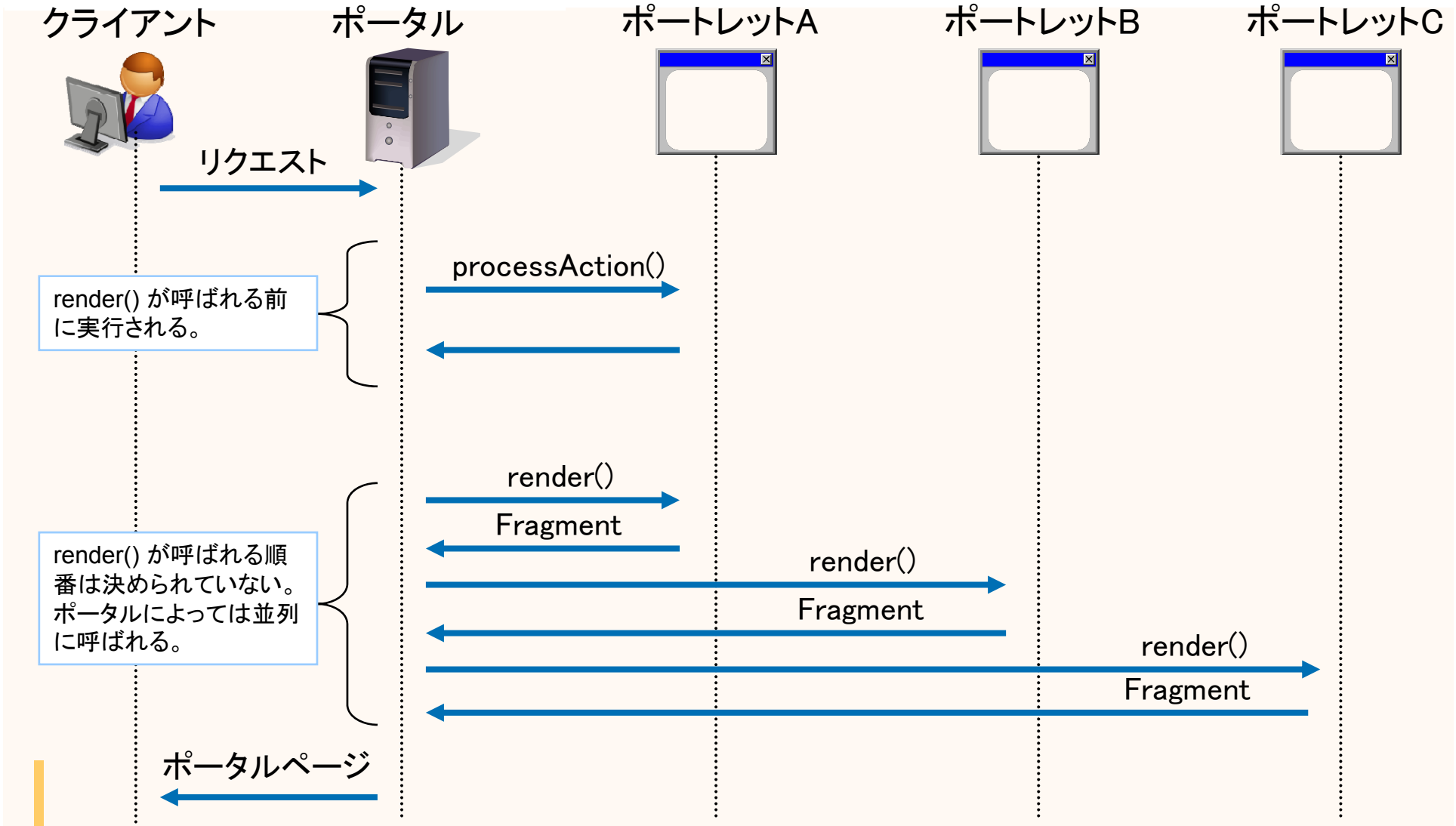


ポートレットのタイトルバーにボタンが表示される

ポートレットの処理

- アクションと描画で処理が分かれる
  - processAction() と render()
- リクエストのフローは2種類
  - アクションなし: 各 render()
  - アクションあり: processAction()→各 render()
- アクション用と描画用のリクエストとレスポンス
  - ActionRequest と ActionResponse
  - RenderRequest と RenderResponse

アクションありのフロー



portlet.xml

- ポートレットの情報を記述
- portlet.xml に複数のポートレットを定義
- WEB-INF に置く

WARファイル

- 基本的にはサーブレットと同じ
- /WEB-INF/portlet.xml が必要
- WARをポータルに配備する
- 複数のポートレットを含む

パフォーマンス

- 1つのポートレットでアクションがあると、ページ上のすべてのポートレットで render() が呼ばれる
  - render() を非同期にする
  - キャッシュを有効にする
  - Ajax で対象のポートレットだけ処理

ネームスペース問題

- JavaScript を使った複数のポートレットを配置するとよく遭遇する
- ポートレットの固有IDを付加する
  - <portlet:namespace/>
  - renderResponse.getNamespace()

ポータル関連プロダクトの紹介

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

Jetspeed 2

- Apache Portals が提供するポータル実装
- ポータルコンテナには Apache Pluto を利用
- Spring を利用しているので、コンポーネントの差し替えが可能
- JSR 168 に合格している
- 現在 2.1.3 (2.2 に向けて活動中)

PALポータル

- SourceForge.jpのPALプロジェクトから提供
- Jetspeed 2 をベースにしたポータル
- 管理ツールや UI を拡張
- 現在 1.0.5 (1.1 に向けて活動中)

フレームワーク

- ポートレットでもいろいろなフレームワークを利用することができる
  - JSF
  - Struts
  - PHP
  - Perl
  - Groovy
  - Velocity
  - …

ポートレットフィルタ

- JSR 168 では未定義
- Apache Portals Bridge から提供
- 基本的にはサーブレットフィルタと同様
- processAction() と render() の前後で処理

Seasar2

- S2Container でポートレットをサポート
- S2Portlet でポートレットに必要なものを提供
  - S2GenericPortlet
  - S2PortletFilter
  - HotdeployPortletFilter
- Teeda でポートレットをサポート
- SAStruts でポートレットをサポート

ポートレットAPIでポートレット作成

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

HelloWorldポートレット

- 入力されたものを表示するだけのポートレット
- 処理の概要
  - 表示にはJSPを使う
  - GenericPortletを継承したポートレット
  - メッセージをリソースバンドルから取得
- PALプロジェクトのhelloworld 1.3
  - http://sourceforge.jp/projects/pal

必要な環境

- Java 1.4 以上
- ビルド環境 (Ant や Maven2 とか)
- JSR 168 準拠のポータル (PALポータルとか)

（普通のウェブアプリ開発と変わらない・・・）

ファイル一覧

- portlet.xml（ポートレット配備記述子）
- web.xml
- HelloWorldPortlet.java（ポートレットクラス）
- helloworld.jsp
- HelloWorldResources.properties
- HelloWorldResources_ja.properties

portlet.xml（一部省略）

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://…">
  <portlet>
    <description>HelloWorld is a portlet</description>
    <portlet-name>HelloWorld</portlet-name>
    <display-name>Hello World</display-name>
    <portlet-class>jp.sf.pal.helloworld.HelloWorldPortlet</portlet-class>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
    </supports>
    <resource-bundle>jp.sf.pal.helloworld.resources.HelloWorldResources</resource-bundle>
    <portlet-info>
      <title>Hello World</title>
    </portlet-info>
  </portlet>
</portlet-app>
```

portlet-name: ユニークなID
display-name: ポータル上の表示名など
portlet-class: ポートレットのクラス名

web.xml

```
<?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>HelloWorld Portlet</display-name>
  <description>HelloWorld Portlet</description>
</web-app>
```

helloworld.jsp

```
<%@ taglib uri="http...let" %>
<%@ taglib uri="http://java.sun.com/jsti/fmt" prefix="fmt" %>
<fmt:setBundle basename="jp.sf.pal.helloworld.resources.HelloWorldResources" />
<portlet:defineObjects/>
<form action="<portlet:actionURL />" method="POST">
  <table border="0">
    <tr>
      <td align="center"><fmt:message key="helloworld.lable.Hello"/></td>
      <td align="center"><%= request.getAttribute("yourName") %></td>
    </tr>
    <tr>
      <td align="right"><fmt:message key="helloworld.lable.YourName"/></td>
      <td align="left"><input type="text" name="yourName"/></td>
    </tr>
    <tr>
      <td align="center" colspan="2"><input type="submit" value="<fmt:message
      key="helloworld.lable.Submit"/>"/></td>
    </tr>
  </table>
</form>
```

portletタグの初期化

フォームの送信先はポータルへ

HelloWorldPortlet.java

```
public class HelloWorldPortlet extends GenericPortlet {

    public static final String YOUR_NAME_KEY = "yourName";

    public void init() throws PortletException {
    }

    public void destroy() {
    }
… doViewと processAction() …
}
```

HelloWorldPortletのdoView()

```
protected void doView(RenderRequest request, RenderResponse
    response) throws PortletException, IOException {
        response.setContentType("text/html");

        String yourName = request.getParameter(YOUR_NAME_KEY);
        if (yourName == null) {
            yourName = "";
        }
        request.setAttribute(YOUR_NAME_KEY, yourName);

        PortletContext context = getPortletContext();
        PortletRequestDispatcher rd = context
                .getRequestDispatcher("/helloworld.jsp");
        rd.include(request, response);
    }
```

processActionからの値を取得

jspに値を渡す

jspを呼ぶ

HelloWorldPortletの processAction()

```
public void processAction(ActionRequest request, ActionResponse
   response)
        throws PortletException, IOException {
    String yourName =
      request.getParameter(YOUR_NAME_KEY);
    if (yourName != null) {
        response.setRenderParameter(YOUR_NAME_KEY,
          yourName);
    }
    return;
  }
```

jspファイルから送信された値を取得

renderに値を渡す

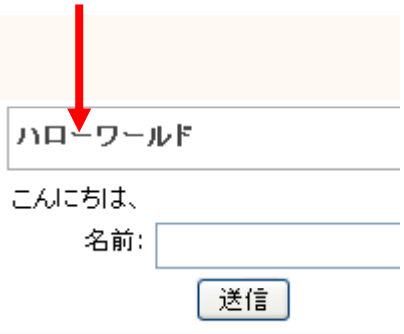
HelloWorldResources.properties

```
helloworld.lable.Hello=Hello!
helloworld.lable.YourName=Your Name:
helloworld.lable.Submit=Submit
```

HelloWorldResources_ja.properties

```
javax.portlet.title=ハローワールド
javax.portlet.short-title=ハロー
javax.portlet.keywords=テスト,ハロー

helloworld.lable.Hello=こんにちは、
helloworld.lable.YourName=名前:
helloworld.lable.Submit=送信
```



ビルドと配備

- Maven2でビルド
- PALポータルを起動
- webapps/palportal/WEB-INF/deploy に helloworld.war をコピー
- PALポータルにログインして、コンテンツを追加

フレームワークでポートレット作成（Teeda編）

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

Teedaを使う

- ちょっとした設定でポートレットとしても動作する
- 修正のポイント
  - web.xmlを変更
  - portlet.xmlを追加
  - ポートレットフィルタとS2Portletのjarを追加
  - htmlファイルからhtml、head、bodyタグを削除する

web.xmlを変更

- TeedaServlet であれば TeedaPortletExtendedServlet に書き換える

or

- TeedaConfigureListener であれば TeedaPortletExtendedConfigureListener に書き換える

portlet.xmlを追加

```
<portlet-app version="1.0">
  <portlet>
    <portlet-name>TeedaHtmlExample</portlet-name>
    <display-name>Teeda HTML Example</display-name>
    <description>This is an example portlet for Teeda.</description>
    <portlet-class>org.apache.portals.bridges.portletfilter.FilterPortlet</portlet-class>
    <init-param>
      <name>portlet-class</name>
      <value>org.seasar.teeda.core.portlet.FacesPortlet</value>
    </init-param>
    <init-param>
      <name>portlet-filters</name>
      <value>org.seasar.portlet.filter.S2PortletFilter,
        org.seasar.portlet.filter.HotdeployPortletFilter</value>
    </init-param>
    <init-param>
      <name>view-page</name>
      <value>/view/start/index.html</value>
    </init-param>  …
```

ポートレットフィルタとS2Portletのjarを追加

- 下記の jar ファイルを WEB-INF/lib に置く
- S2Portlet: http://s2portlet.sandbox.seasar.org/
- Portlet Filter: http://portals.apache.org/bridges/

htmlファイルからタグを削除する

- htmlファイルからhtml、head、bodyタグを削除

or

- PALプロジェクトのフィルタを使う
  - jp.sf.pal.facesresponse.FacesResponseFilter
  - jp.sf.pal.pooptimizer.OptimizerFilter
- Jarファイルを追加
  - faces-response-filter-0.2.jar
  - portlet-output-optimizer-0.2.jar

ビルドと配備

- いつも通りビルド
- PALポータルを起動
- webapps/palportal/WEB-INF/deploy に war ファイルをコピー
- PALポータルにログインして、コンテンツを追加

フレームワークでポートレット作成（SAStruts編）

- ポータル・ポートレットの概要
- ポートレットAPI (JSR 168)
- ポータル関連プロダクトの紹介
- ポートレットAPIでポートレット作成
- フレームワークでポートレット作成（Teeda編）
- フレームワークでポートレット作成（SAStruts編）

SAStrutsを使う

- ちょっとした設定でポートレットとしても動作する
- 修正のポイント
  - web.xmlを変更
  - struts-config.xmlを変更
  - common.jspを変更
  - portlet.xmlを追加
  - sa-struts-portlet-1.0-*.jar を追加

web.xmlを変更

- org.seasar.struts.portlet.filter.PortletRequestFilter をencodingFilter の後にINCLUDEで追加
- S2ContainerFilter を org.seasar.struts.portlet.filter.S2ContainerFilter に変更
- HotdeployFilter を org.seasar.framework.container.hotdeploy.HotdeployFilter に変更
- org.seasar.struts.portlet.filter.PortletRoutingFilter を routingfilter の後にINCLUDEで追加

struts-config.xmlを変更

- S2RequestProcessor を org.seasar.struts.portlet.action.S2RequestProcessor に変更

common.jspを変更

- <%@taglib prefix="s" uri="http://sastruts.seasar.org"%>

を以下に変更

<%@taglib prefix="s" uri="http://sastruts.seasar.org/portlet"%>

portlet.xmlを追加

```
<portlet-app>
  <portlet>
    <description>SAStruts Tutorial</description>
    <portlet-name>SAStrutsTutorialPortlet</portlet-name>
    <display-name>SAStruts Tutorial Portlet</display-name>
    <portlet-class>org.seasar.struts.portlet.SAStrutsPortlet</portlet-class>
    <init-param>
      <name>viewPage</name>
      <value>/</value>
    </init-param>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>configFactory</param-name>
      <param-value>org.seasar.struts.config.S2ModuleConfigFactory</param-value>
    </init-param>
...
```

sa-struts-portletのjarを追加

- SAStruts: http://sastruts.seasar.org/
  （上記から配布予定）
- 上記の jar ファイルを WEB-INF/lib に置く

注意点

- aタグのリンクは必要に応じて s:linkタグに書き換える
- デフォルトでbodyタグ内のみが表示される
- JavaScript の名前空間問題に注意する

ビルドと配備

- いつも通りビルド
- PALポータルを起動
- webapps/palportal/WEB-INF/deploy に war ファイルをコピー
- PALポータルにログインして、コンテンツを追加

ご清聴ありがとうございました。