

Spago Portlet

Redatto da

Daniela Butano

Indice

INFORMAZIONI SULLA VERSIONE	3
SCOPO DEL DOCUMENTO.....	3
RIFERIMENTI.....	3
1 INTRODUZIONE.....	4
SPAGO FRAMEWORK MULTICANALE	4
CANALE HTTP	4
ADAPTERPORTLET.....	4
2 CREAZIONE DI UNA PORTLET	5
AMBIENTE DI SVILUPPO.....	5
CONFIGURAZIONE DI UNA PORTLET.....	5
2.1.1 Creazione di una portlet application	6
2.1.2 Librerie	6
2.1.3 File di configurazione web.xml.....	6
2.1.4 File di configurazione portlet.xml.....	6
PUBLISHER	7
PAGINA JSP.....	7
3 CONFIGURAZIONE NEL PORTALE	9

Informazioni sulla versione

Versione/Release n° :	Bozza	Data Versione/Release :	28/02/2005
Descrizione modifiche:	Versione iniziale		

Scopo del documento

Il documento ha l'obiettivo di fornire alcune linee guide per la creazione di una portlet scritta con Spago.

La lettura del documento presuppone una conoscenza base di Spago

Si assume che il lettore abbia già noti i concetti di portale, portlet, portlet container etc.

Il progetto Spago-portlet, scaricabile dal sito <http://sourceforge.net/projects/spago> nel package *Spago Pre-release Components*, è ancora in fase di test. Questo significa che non sono ancora state testate/implementate tutte le funzionalità previste per il canale http tra cui la modalità "freeze" (modalità per cui la action si prende in carico la produzione della risposta), la modalità di pubblicazione di tipo LOOP, la generazione automatica delle liste.

Riferimenti

Per ulteriori informazioni sul framework Spago si rinvia ai seguenti documenti disponibili in http://spago.eng.it/docs_it/documentation/index.html :

- [1] *Spago Overview*
- [2] *Spago User Guide*
- [3] *Spago Tutorial*

Per informazioni sulle portlet si rinvia ai seguenti documenti:

- [1] *JavaTM Portlet Specification*

1 Introduzione

Breve introduzione alla multicanalità di Spago e nello specifico al canale http.

SPAGO FRAMEWORK MULTICANALE

Spago è un J2EE framework che consente di erogare, in modo semplice, servizi indipendentemente dal canale: HTTP, WAP, SOAP, EJB.

Questo è reso possibile dal fatto che sia stato progettato in modo da rendere indipendenti i layer sottostanti alla pubblicazione da ogni componente specifico del canale, utilizzando l'XML quale supporto alla comunicazione (ad esempio, la logica applicativa viene sviluppata in modo indipendente dagli oggetti `HttpRequest` ed `HttpSession`, specifici del canale HTTP) .

Tramite gli adapter, i dati della richiesta provenienti da uno specifico canale, vengono trasformati nel formato interno del framework (ovvero XML) e viene eseguito il binding del contesto conversazionale sulle risorse del container specifico.

CANALE HTTP

Per quanto riguarda il canale HTTP l'ultima versione consolidata di Spago prevede la pubblicazione solo sul web container grazie all' **AdapterHTTP**, servlet che gestisce le richieste provenienti da un client http (browser web o dispositivo wap) e che vive quindi nel web container.

Il progetto Spago-portlet, permette la pubblicazione sul portlet container grazie all' **AdapterPortlet**.

ADAPTERPORTLET

L'**AdapterPortlet** è una portlet ovvero un componente web che vive in un portlet container.

Come tutti gli adapter pre-esistenti, l'**AdapterPortlet** è un oggetto in ascolto delle richieste di servizi che traduce le richieste nel formato XML ed attiva il servizio con le due modalità di logica di business: action o moduli.

La classe *it.eng.spago.dispatching.httpchannel.AdapterPortlet* estende la classe astratta *javax.portlet.GenericPortlet*.

Attualmente implementa i metodi:

- *processAction*
- *doView*

Viene rilasciata in *spago-portlet.jar*

2 Creazione di una portlet

Alcune linee guida per la creazione di una portlet utilizzando Spago.

AMBIENTE DI SVILUPPO

Strumenti di sviluppo per la creazione di un'applicazione web:

- ❑ **Eclipse 3.0** scaricabile dal sito <http://www.eclipse.org/>

Ambiente di test per le portlet.

- ❑ **Pluto Apache**

Non è un portale ma solo l'implementazione di un portlet container. Ambiente leggero, utile nella fase iniziale di sviluppo. Esiste una versione alfa del plugin per eclipse scaricabile da <http://plutoeclipse.sourceforge.net/>. Installato il plugin, creare un Pluto Portal Application che installa su Tomcat le librerie di pluto e il progetto pluto. A questo punto è possibile creare una portlet. Consultare il tutorial per i dettagli.

Controindicazioni: il plugin funziona con versione Tomcat 4.1.x (testato con 5.0.x dà problemi), in web.xml non si può configurare più di un tag-lib (si può ovviare al problema impostando il path del file *.tld direttamente nella JSP), manca la gestione degli user attribute, non implementato il sistema di caching. Non effettua il deploy.

Per il deploy dell'applicazione il plugin **Lomboz 3.0.1** scaricabile dal sito www.objectlearn.com può andare bene.

- ❑ **Exo Platform**

Portale Open Source. Decisamente meno leggero di Pluto, può andare bene nella fase avanzata di sviluppo. Esiste il plugin per eclipse che permette di creare una portlet e fare il deploy direttamente sul portale.

Problemi per effettuare il debugging su librerie esterne.

CONFIGURAZIONE DI UNA PORTLET

Vediamo i passi principali per configurare la portlet utilizzando Spago.

Si suppone che il lettore sappia già configurare un'applicazione web fatta con Spago.

2.1.1 CREAZIONE DI UNA PORTLET APPLICATION

Dopo aver avviato *Eclipse*, creare il progetto per l'applicazione d'esempio selezionando la voce *New->Project->Portlet Application*. Si consideri che a seconda del plugin che si sta utilizzando le voci di menu potrebbero variare.

Impostare i parametri relativi al nuovo progetto.

2.1.2 LIBRERIE

Impostare

- le librerie di Spago: spago-corex.x.x.jar, spago-webx.x.x.jar, spago-portletx.x.x.jar
- le librerie utilizzate da Spago rilasciate assieme a Spago
- le librerie relative al portlet container ovvero *portlet-api.jar* e *portal-container-impl.jar* (quest'ultima potrebbe avere un nome diverso a seconda del plugin che si sta utilizzando)

Queste ultime probabilmente saranno già state configurate dal plugin per eclipse che state utilizzando nel momento in cui avete creato la Portlet Application.

2.1.3 FILE DI CONFIGURAZIONE WEB.XML

Il file di configurazione *web.xml* è legato all'ambiente in cui si sta lavorando, ovvero Pluto e Exo-tomcat hanno dei file di configurazione diversi. Per i dettagli consultare le rispettivi documentazioni.

In entrambi i casi va comunque configurata la *ConfigServlet* all'avvio del server ovvero:

```
<servlet>
  <servlet-name>ConfigServlet</servlet-name>
  <servlet-class>it.eng.spago.configuration.ConfigServlet</servlet-class>
  <init-param>
    <param-name>AF_CONFIG_FILE</param-name>
    <param-value>\WEB-INF\conf\spago\master.xml</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
```

e le tag libraries implementate dal portale che state utilizzando:

```
<taglib>
  <taglib-uri>portal_tags</taglib-uri>
  <taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>
```

2.1.4 FILE DI CONFIGURAZIONE PORTLET.XML

Le portlet e le relative configurazioni devono essere censite nel file di configurazione *portlet.xml*. Il file di configurazione contiene il riferimento all'AdapterPortlet:

```
<portlet-class>it.eng.spago.dispatching.httpchannel.AdapterPortlet</portlet-class>
```

Si possono definire più portlet all'interno dell'applicazione web utilizzando il tag `<portlet>` all'interno di `portlet.xml`.

Per ogni portlet definita bisogna impostare l'attributo `pageStart` che contiene la pagina che deve essere invocata alla prima chiamata della portlet, ovvero quando ancora non è stata eseguita alcuna logica di business.

```
<init-param>
  <name>pageStart</name>
  <value>/jsp/employee.jsp</value>
</init-param>
```

PUBLISHER

La prima richiesta effettuata alla portlet è una *render request*, ovvero il portlet container invoca il metodo `doView()` dell'AdapterPortlet. L'AdapterPortlet sapendo di essere stato invocato per la prima volta, non avendo ancora eseguito alcuna logica di business, ovvero non avendo eseguito alcun action o modulo (e quindi non avendo alcuna informazione sulla pagina da pubblicare) recupera dal file di configurazione `portlet.xml` l'attributo `pageStart`, che contiene la prima pagina da pubblicare, e ne demanda la sua pubblicazione al sottosistema di pubblicazione.

Le richieste successive alla portlet devono essere invece di tipo *action request*, ovvero il portlet container invoca nell'ordine i metodi `processAction()` e `doView()` (nel caso di modalità VIEW). Nel metodo `processAction()` avviene l'esecuzione della logica di business.

Nel metodo `doView()` avviene la pubblicazione del risultato secondo le solite regole. Il sottosistema di presentazione associa l'action, o la pagina, al corrispondente oggetto di pubblicazione secondo i parametri configurati in `presentation.xml` e `publisher.xml`.

Per le portlet, il pubblicatore deve essere definito nel seguente modo:

```
<PUBLISHER name="INSERT_EMPLOYEE">
  <RENDERING channel="PORTLET" type="JSP" mode="">
    <RESOURCES>
      <ITEM mode="VIEW" resource="/jsp/employee.jsp" />
      <ITEM mode="EDIT" resource="/jsp/edit.jsp" />
    </RESOURCES>
  </RENDERING>
</PUBLISHER>
```

PAGINA JSP

Vediamo un esempio di pagina JSP che può essere pubblicata in un portale all'interno di un'applicazione che utilizzi Spago.

```
<%@ page session="false" %>
```

```

<%@ page
extends="it.eng.spago.dispatching.httpchannel.AbstractHttpJspPagePortlet"%>

<%@ page import="it.eng.spago.base.*"%>

<%@ taglib uri='portlet_tags' prefix='portlet'%>

<portlet:defineObjects/>

Current Portlet Mode: <%=renderRequest.getPortletMode()%><br>
Current Window State: <%=renderRequest.getWindowState()%><br>

<%

SourceBean serviceResponse = getServiceResponse(request);
String name = null;
if (serviceResponse != null)
    name = (String)serviceResponse.getAttribute("name");

RequestContainer requestContainer = getRequestContainer(request);
String test = null;
if (requestContainer != null)
    test = (String)requestContainer.getSessionContainer().getAttribute("sessionTest");

%>

Risultato del test: <%=test%>

<form name="employeeForm" action="<portlet:actionURL><portlet:param
name="ACTION_NAME" value="INSERT_EMPLOYEE"/><portlet:param name="NEW_SESSION"
value="TRUE"/></portlet:actionURL>" method="post">

    <table>

    <tr>

        <td>Nome:</td>

        <td><input class="text" type="text" size="10" name="name" value="<%=name%>"
></td>

    </tr>

    <tr>

        <td>Cognome:</td>

        <td><input type="text" size="5" name="surname" value=""></td>

    </tr>

    <tr>

        <td>Ufficio:</td>

        <td><input type="text" size="10" name="department" value=""></td>

    </tr>

</table>

<table>

<tr>

        <td><INPUT TYPE="submit" name="save" value="Salva"></td>

```



```
</tr>
</table>
</form>
```

Alcune note importanti:

- ❑ La pagina JSP prodotta da una portlet non deve contenere i seguenti tag: base, body, iframe, frame, frameset, head, html and title.
- ❑ La pagina estende la classe `it.eng.spago.dispatching.httpchannel.AbstractHttpJspPagePortlet`: questo le permette di accedere alla `serviceResponse`, al `requestContainer`, etc con dei metodi analoghi al web container ovvero `getServiceResponse(request)`, `getRequestContainer(request)`, etc.

Nel caso non si estenda `AbstractHttpJspPagePortlet` è possibile accedere agli stessi oggetti tramite le classi `it.eng.spago.dispatching.httpchannel.RequestContainerPortletAccess` e `it.eng.spago.dispatching.httpchannel.ResponseContainerPortletAccess` che espongono tutti i metodi statici.

- ❑ Tramite la riga `<%@ taglib uri="portlet_tags" prefix="portlet"%>` si possono utilizzare i tag implementati dal portlet-container che permettono di accedere agli oggetti specifici delle portlet (`RenderRequest` e `RenderResponse`) e alle funzionalità quali la creazione di un `PortletURL`.
- ❑ Al submit del form di login viene invocata l'action `INSERT_EMPLOYEE` utilizzando il tag `ActionURL` (per la creazione di un URL che chiama direttamente la portlet corrente tramite una `actionRequest`) e il tag `param` per l'invio dei parametri.
- ❑ Utilizzare sempre la modalità POST per l'invio dei parametri al server.
- ❑ E' consigliato non utilizzare path relativi per accedere alle risorse all'interno del portale ma:

```
renderResponse.encodeURL(renderRequest.getContextPath()+path_risorsa);
```

3 Configurazione nel portale

Come primo passo bisogna effettuare il deploy dell'applicazione su Tomcat.

A questo punto è necessario importare la portlet all'interno del portale.


Nel caso si stia utilizzando Pluto bisogna configurare i file di configurazione `pluto\WEB-INF\data\pageregistry.xml` per la configurazione della pagina e `pluto\WEB-INF\data\portletentityregistry.xml` per la configurazione della portlet appena creata. Per i dettagli consultare la documentazione.

Nel caso di Exo Tomcat, è necessario autenticarsi come *admin*, selezionare dalle voci di menu *Portal/Portlet Registry* e accedere al link *Import Portlets*. Una volta importata la portlet aggiungerla ad una pagina preesistente o ad una nuova seguendo la documentazione.

Vediamo il risultato di una semplice portlet di inserimento creata con Spago e esportata rispettivamente sugli ambienti Pluto ed Exo-Tomcat.

Pluto - Reference Implementation

Sample Portlet	Insert	
Users List		
Insert	Simple Employee Insert	Simple Departement Insert
BPM Search	Current Portlet Mode: view	Current Portlet Mode: view
Spago BI - Foglio	Current Window State: normal	Current Window State: normal
Spago BI - Report Handler	Risultato del test: TEST	Verifico scope della sessione. Test: null
	Nome: <input type="text" value="Daniela"/>	Codice: <input type="text" value="code1"/>
	Cognome: <input type="text" value="Butano"/>	Descrizione: <input type="text"/>
	Ufficio: <input type="text" value="Ricerca e Inn"/>	<input type="button" value="Salva"/>
	<input type="button" value="Salva"/>	

 **try it, adopt it...**

[Home](#) [Organization](#) **Portal** [Monitoring](#) [Portlets](#) [Site Map](#)

[Home > Portal](#)

Home

- [+ Organization](#)
- [- Portal](#)
 - [Portlet Registry](#)
 - [Internationalization](#)
 - [Import/Export](#)
 - [Indexing](#)
 - [WSRP Admin](#)
- [+ Monitoring](#)
- [+ Portlets](#)
- [Site Map](#)

[Logout](#)


[Visited Pages](#)


InsertEmployeePortlet


Current Portlet Mode: view
Current Window State: normal
Risultato del test: TEST


Nome:
Cognome:
Ufficio:

Administration summary


[Register a user](#)


[Organisation management](#)


[Portals management](#)


[Portlet regist](#)