Implementing JSR 168: Apache's Pluto and JA-SIG's uPortal

Overview

The Apache Pluto is the implementation of the portlet container that complies with JSR 168 requirements. The Pluto provides functionality that allows to run portlets, provide them with the required runtime environment and manages their life cycles. It also provides persistent storage mechanisms for the portlet preferences. Since the portal handles aggregation itself the portlet container is not responsible for aggregating the content produced by the portlets.
The Pluto implementation provides the following APIs:

1) Portal Container Invoker API – the interface allowing access from a portal server to a portlet container.
   Package: **org.apache.pluto.invoker**
2) Container Provider SPI (Service Provider Interface) – the callback interface of the portlet container providing access to portal-related information (container services).
   Package: **org.apache.pluto.services**
3) Portlet API – the portlet container's interface for invoking the portlets.
   Package: **javax.portlet**

Portlet API

The basic classes and interfaces for handling the portlet modes, window states, content rendering, portlet session/request/response, portlet preferences and locales can be found in this API.
The Pluto Portlet API implementation supports access to user profile information via the request attribute USER_INFO as a map. A portlet can define in the deployment descriptor that user information attributes it would like to access via this map in the request. The portal can then either map these attributes at deployment time to the attributes in its user datastore or ignore them if they are not supported.
The Portlet API provides portlet lifecycle management that consists of:

a) Initializing the portlet using using init() method
b) Request processing, methods processAction() and render()
c) Releasing the portlet from service using destroy() method

Portal context

To let portlets adapt to the portal that calls them, the Portlet API provides the PortalContext, which can be retrieved from the request. This portal context provides information such as the portal vendor, the portal version, and specific portal properties. The information allows the portlet to use specific vendor extensions when being called by a portal that supports these extensions and, when being called by other portals, to return to some simpler default behavior. As the portal context is attached to the request, it may change from request to request. This can be the case in the remote scenario, where one portlet (WSRP provider) may be called from different portals (WSRP consumers).

Localization

The Pluto framework provides localization on two levels: deployment descriptor and portlet. On the deployment descriptor level, all settings intended to be viewed or changed by the Web server administrator (portlet description, init parameters, display name, and so on) consist of an xml:lang attribute, like the Servlet 2.4 deployment descriptor. The xml:lang tag provides the same tag with descriptions in different languages (e.g., a display name in English, German, and Japanese).

Properties

Properties communicate vendor-specific settings between the portlet and portlet container, and between the portlet and the portal. These properties show up in several places in JSR 168. The portlet can read String properties with getProperty() from:

- **ActionRequest,** to receive properties that are action-request specific
- **RenderRequest,** to receive properties that are render-request specific
- **PortalContext,** to receive properties that are portal specific and do not change for different requests

The portlet can write String properties with setProperty() to:

- **ActionResponse,** to set properties in response to an action request
- **RenderResponse,** to set properties in response to a render request

Portlet deployment

A portlet application's resources, portlets, and deployment descriptors are packaged together in one Web application archive (war file). In contrast to servlet-only Web applications, portlet applications consist of two deployment descriptors: one to specify the Web application resources (web.xml) and one to specify the portlet resources (portlet.xml). All Web resources that are not portlets must be specified in the web.xml deployment descriptor. All portlets and portlet-related settings must be specified in an additional file called portlet.xml. There are three exceptions to this rule, which are all defined in the web.xml, as they are valid for the whole Web application:

1. The portlet application description
2. The portlet application name
3. The portlet application security role mapping

As result of the two deployment descriptor files, portlet application deployment is a two-step deployment that deploys the Web application

into the application server and the portlets into the portal server. A portlet.xml file always describes only one specific portlet application. To create a copy of a portlet application with slightly different settings, a new portlet application must be created.

Pluto's portal implementation

The Pluto's cvs repository contains some classes related to the portal implementation aggregated in **org.apache.pluto.portalImpl** package. Unfortunately most of them are under construction at the moment so it is rather hard to say anything detailed about their portal design. There are some classes responsible for portlets aggregation and navigation, servlet request/response processing. It also provides a few services for portlet entities, portlets and page fragments.

Pluto and uPortal

The Pluto framework can be effectively used in the future release of uPortal as a generic set of classes and interfaces that fully complies with JSR 168 specification.
The use of this framework will definitely simplify the design and development of the core code that will include uPortal specific functionality such as aggregated layouts, i18n, groups/permissions framework, etc. The container-related classes from the packages javax.portlet and org.apache.pluto can be used in any portal implementations. The portal-related classes from org.apache.pluto.portalImpl package can be re-written including the specific functionality of the current uPortal.