Hello and welcome to this presentation on Portal managed client integration for Lotus® Expeditor 6.1.

Agenda

- Introduction to Composite Applications
- Developing Composite applications
- Sample Applications

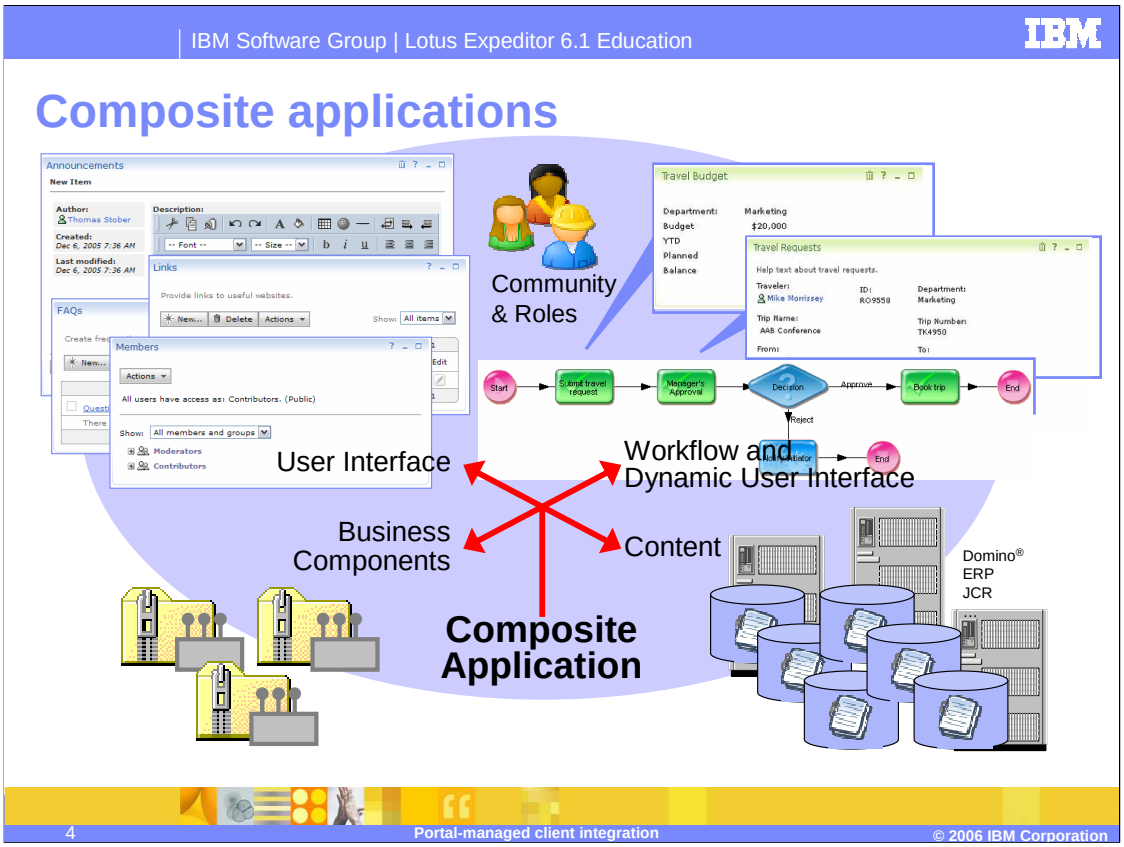
In this presentation, we will learn how to develop composite applications in portal and then deploy and view these applications in the Expeditor 6.1 client.

Section

Composite applications

What are composite applications?

Composite applications are groups of applications that were aggregated together from various places and with various components.

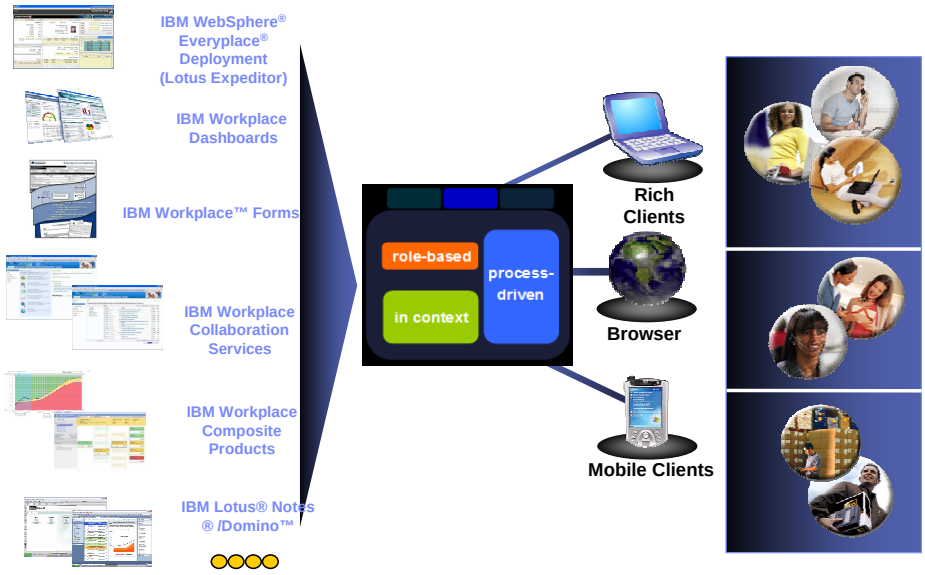


First, we start with some business components and content.

Next, we add the user interface, which could be Eclipse views, portlets, or projected C/C++ apps (like Hannover).

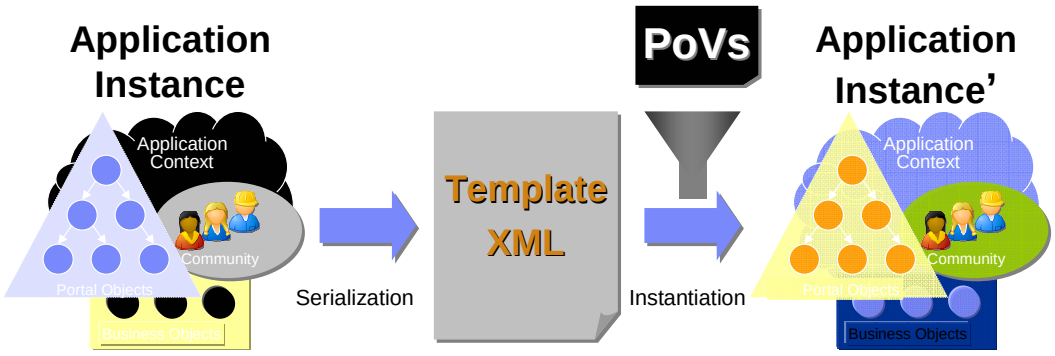
Then we add the people and workflow aspect to create what is referred to as a composite application.

Portal family: The framework for flexible SOA based interaction services



IBM's mobile and enterprise access products extend Lotus' Service Oriented Architecture value proposition by taking these applications to the edge of the network, where the "edge" may be a desktop, laptop, kiosk, PDA, smartphone or feature phone. These mobile and enterprise access products also support non-browser or occasionally connected applications at the edge on rich-clients and mobile clients.

Templates and applications

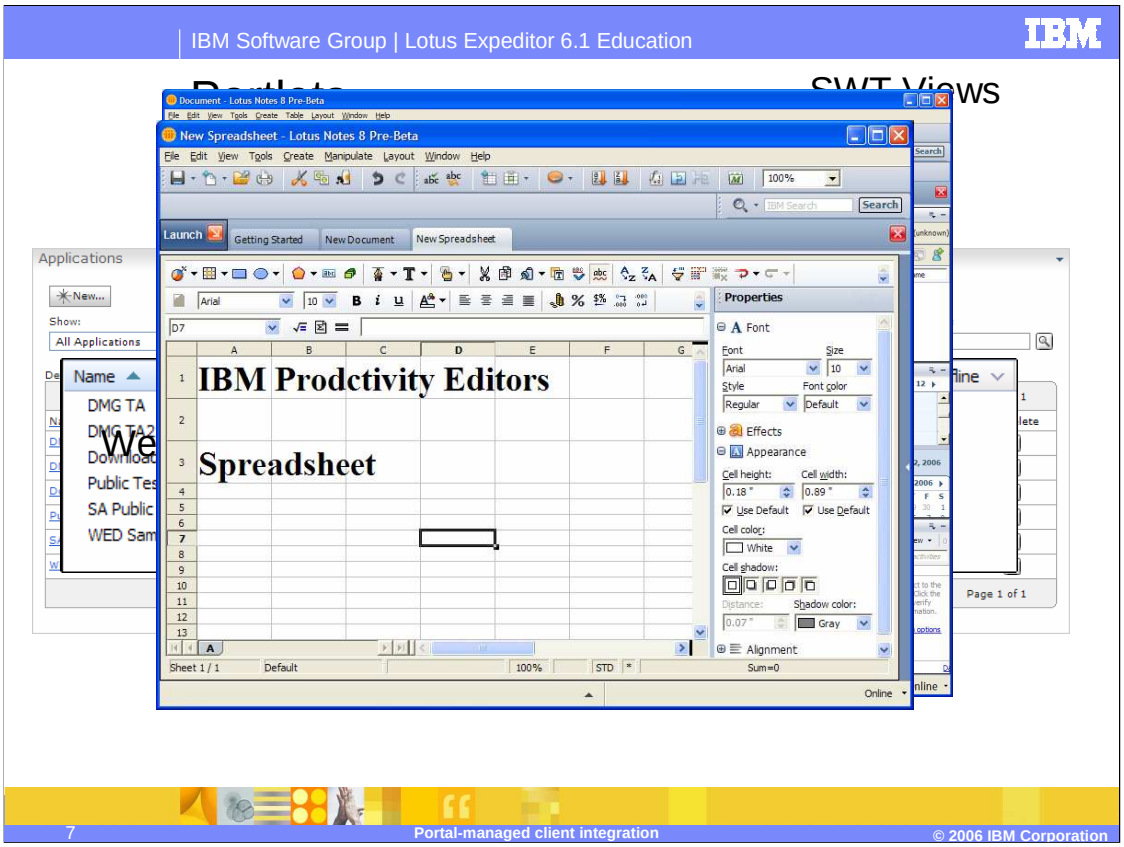


Template XML

- contains the blue prints to easily create another instance of that application
- allows for points of variability to be filled out during instantiation

Templates and Applications in the composite application space with Portal is very similar to Lotus Notes templates and databases. You can think of a template as a quick head-start in creating an application. A template can be defined as a series of pages, portlets, and points of variability.

When an application is created in Portal 6.0 you can now serialize that application to XML template. This template can then be imported on another server as a starting point for new applications. When the application is created, there are points in the template that can be overridden to give the application a specific context – these are called the points of variability.

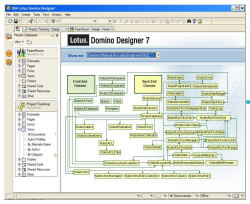


Composite applications in Lotus Expeditor can come from many sources.

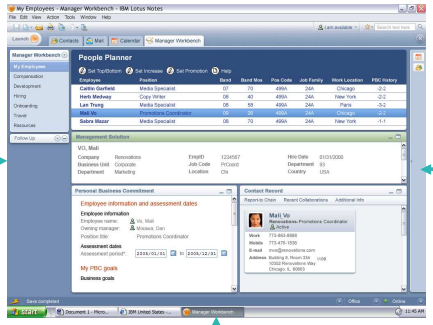
You can have components that are ActiveX based by encapsulating the ActiveX content in a viewpart - similar to what the Lotus Notes and Productivity Editors do today.

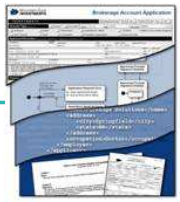
Using the same model as the Lotus Notes you can take your existing Windows based applications and project them through the Eclipse SWT framework.

Components can be built with a variety of tools, and developer skills. Components can be built independently from assembly

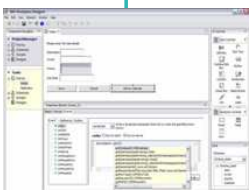


Notes applications built with Domino Designer

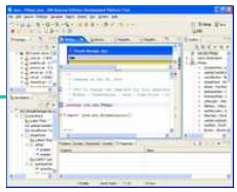




E-form built with Workplace Forms Designer



Collaborative components built with Workplace Designer



Eclipse component built with Rational RAD

Components can be built by many different tools – many of which are freely available in the open source community and obtainable through IBM.

Section

Developing composite applications

This section will give an overview of developing composite applications.

Using the Portal administration and programming model

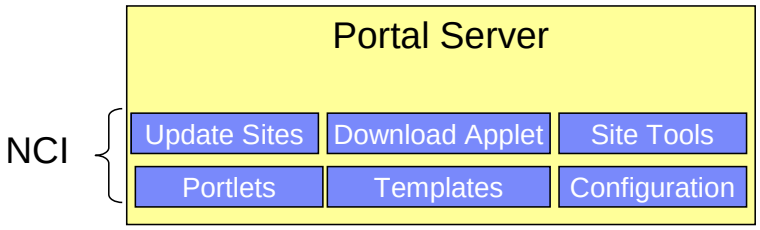
- Use existing Portal Administration tools
  - Manage Pages/Portlets
  - Page Layout
  - Portlet Wiring
  - Page Inheritance
  - User access/roles
- Extend Portal applications with additional Properties
  - Custom Business Components
  - Pages
  - Portlets – both definition and entity

In order to get a consistent runtime environment, the programming models between the client and server need to be the same. The concept is to leverage the same portal tools to administer and develop the composite applications using existing Portal tools and technology. Things like pages, portlets, wires, page inheritance and user access are all used to create a composite application on the Expeditor client.

You can also extend the portal applications with additional properties for the business components, pages and portlets.

Installing the network client installer on Portal

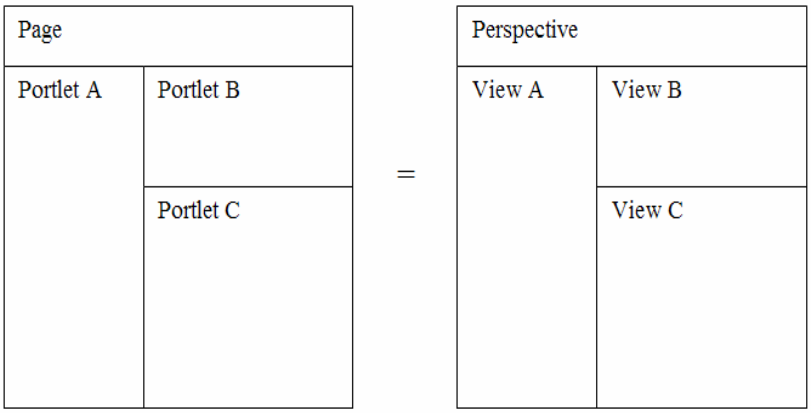
- Installs all necessary components onto the portal server for client side installation.



The Network Client installer installs all of the extra artifacts onto Portal that are needed to support the Expeditor Client. Portal, however, can be used out of the box. The elements shown on this slide will give the portal administrator value added tools when deploying applications to the Expeditor client.

Portal to rich client comparison

- Page = Perspective
- Portlet = SWT View



This slide explains the terminology comparison of Eclipse based terms and Portal based terms.

Lotus Expeditor technologies

- All components are ultimately SWT views
- Open Business Component framework
- Expeditor ships with the following view support
  - All SWT views
  - Portlet Viewer
    - JSR 168 Viewer
    - WSRP Viewer
  - WEB container – JSP's
  - Embedded Browser View

One important factor to mention is that ALL components in an Eclipse based product are Views. Expeditor uses the SWT view as its container for all components (portlets, SWT components, C/C++ containers, etc).

The client is essentially an open business component framework driven by XML (for example, the output from Portal).

Lotus expeditor supports the following views:

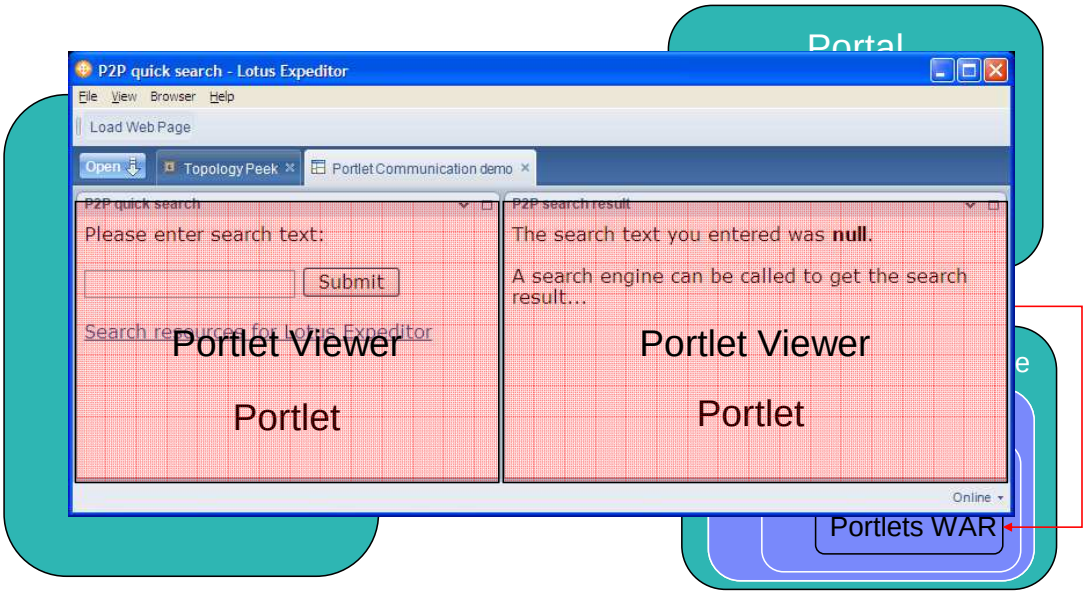
SWT Views

Portlet viewer

Web container

Embedded browser view.

Portlets and Expeditor



This slide shows how to deploy a portlet to an Eclipse update site using the WAB tool or the Expeditor Toolkit export option. The point of the slide is to show that the portlet needs to be deployed, not only as a traditional WAR file into the Portal server, but also exported as an Eclipse feature and plug-in.

When the expeditor client attempts to “show” the portlet it is actually using, the local plug-in (which contains the portlet) and projects the portlet inside of the already installed Portlet viewer (which ships with Expeditor).

Composite applications in the Lotus Expeditor

- A catalog of applications available locally
- Download of application description (CA XML)
- Persisting of CA XML for off-line use
  - Portal pages, navigation, meta-data
- Download of client specific code – Features/plug-ins
- Data synchronization
- User interface – (for example, Navigating between pages)

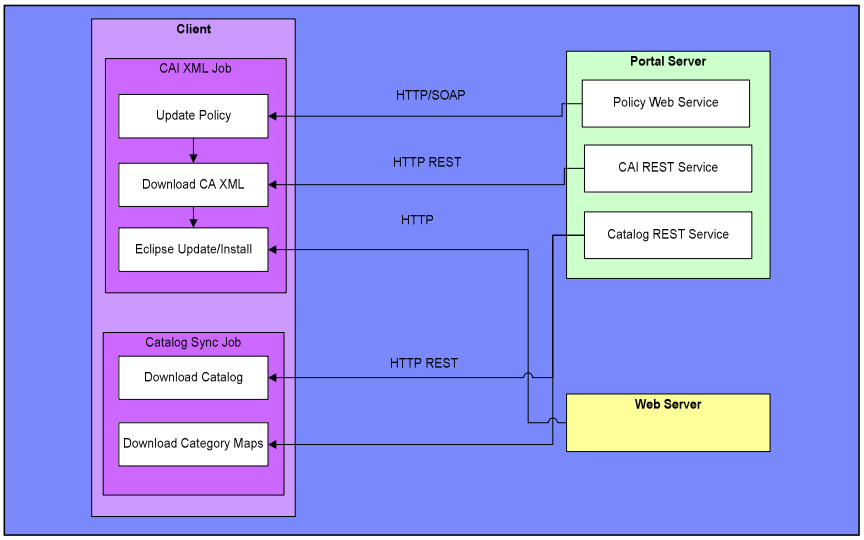
The next series of slides explain how composite applications from portal are accessed, downloaded and run in the Expeditor client.

It all starts with the Portal Catalog in the client. The catalog is the primary entry point for initially accessing a composite app from Portal. When you click on the application in the catalog, it will prompt you to install it and the application definition will be downloaded and cached – this is the composite app xml or CA XML.

The client then installs any Eclipse based components identified in the XML. The normal Eclipse provisioning process is used to install these new features.

Then the client does any specified data synchronization of the application through the sync framework.

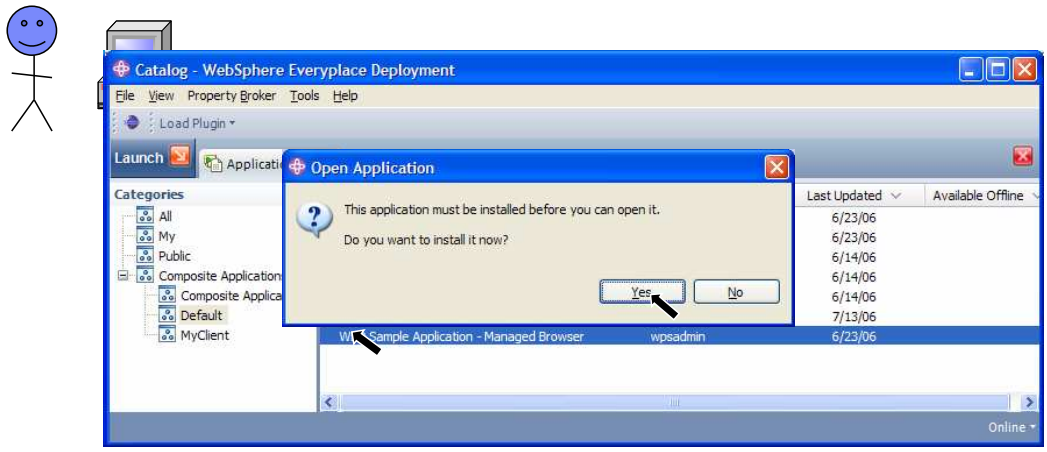
Connecting to Portal



Sample URL: http://localhost:10038/wps/catalogHandler?uri=urn:portal.app.catalog:application

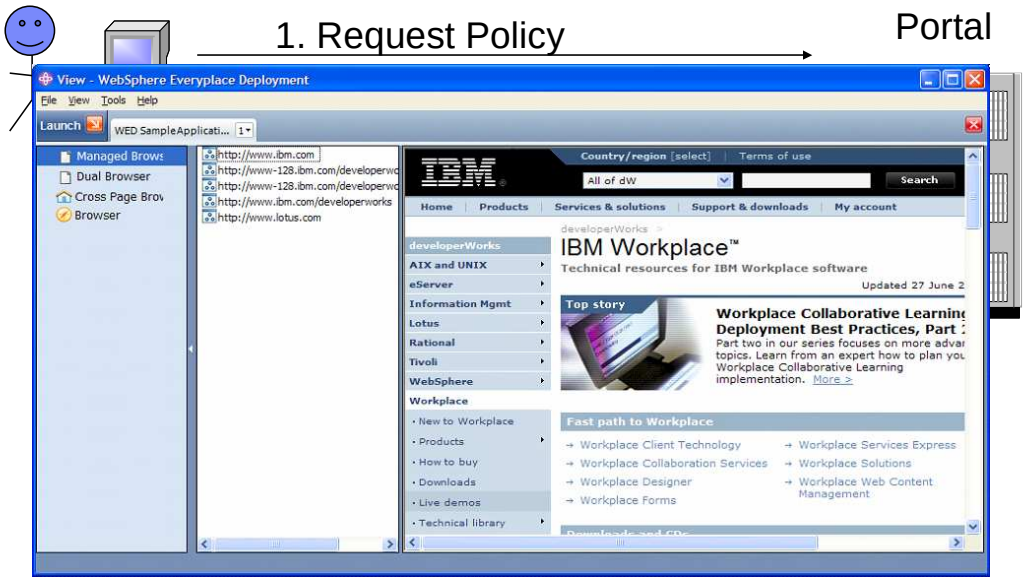
This example above shows how the expeditor client connects to portal and under what protocol. All of the connections are over HTTP.

Installing a composite application from portal



Here we will walk through opening an application from the client catalog, which will then walk through the steps of installing the remote application form the portal server to the client.

Installing a composite application from Portal



This slide provides a “running” example of what happens when a user clicks on an application in the portal catalog. Follow the animation on this slide after the initial screen disappears and the steps are presented.

Section

Developing composite applications

Now we are going to walk through the development of a composite application.

Many different options for applications

- Normal composite application with no off-line
- Locally running composite application – Portlets
  - Portlets run locally
  - Synchronize data
  - Offline supported
- SWT Application aggregated by portal
  - Synchronize data
  - Offline supported
- Custom business components (serialize/deserialize)
- All Portlets/Views can communicate together!

1. There is no browser caching; all components are running locally.
2. Having locally running applications like Portlets, data synchronization and offline is fully supported.
3. Basic SWT applications (Eclipse applications) that are deployed from portal and aggregated together using the portal tools.

You can use Wiring and the Property Broker to wire any components together.

What is a business component?

- Custom code that can serialize its data on Portal and deserialize the data on the client
- Metadata that is specific to the user of the application
- The Portal Topology is a shipping example
  - Contains all of the pages, portlets, points of variability in UI
- A simple extension point on the client

•Business components are used to have their data serialized inside of the CA xml with all of the other components. The XML is sent to the client and the client side version of the component gets the XML and processes it.

•Additional data – points of variability – are also sent in the CA xml and are specialized for the user accessing the application.

•The Portal Topology is an example of a business component. It serializes the page hierarchy, portlets, and Points Of Variability inside the XML and is then consumed by the topology handler on the client.

•The business component is an extension point with a basic interface on the Expeditor client.

Business component extension

Extension:

Plugin ID: com.ibm.portal.app

Point ID: BusinessComponents

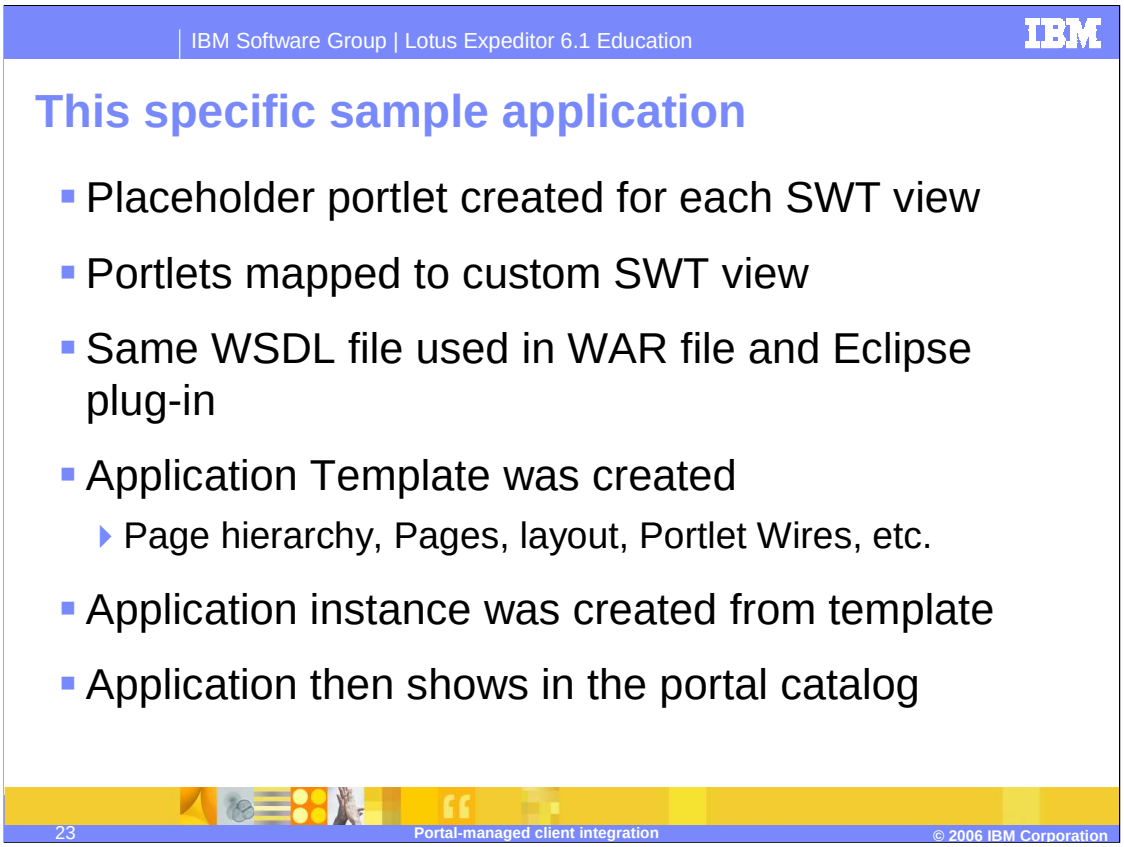
Package: com.ibm.portal.app.component

Interface: Deserializable

Method:

public void deserializeApplicationInstance(String objectdata, Map context);

This is a reference of the business component extension point.



Here we explain the sample application we are building.

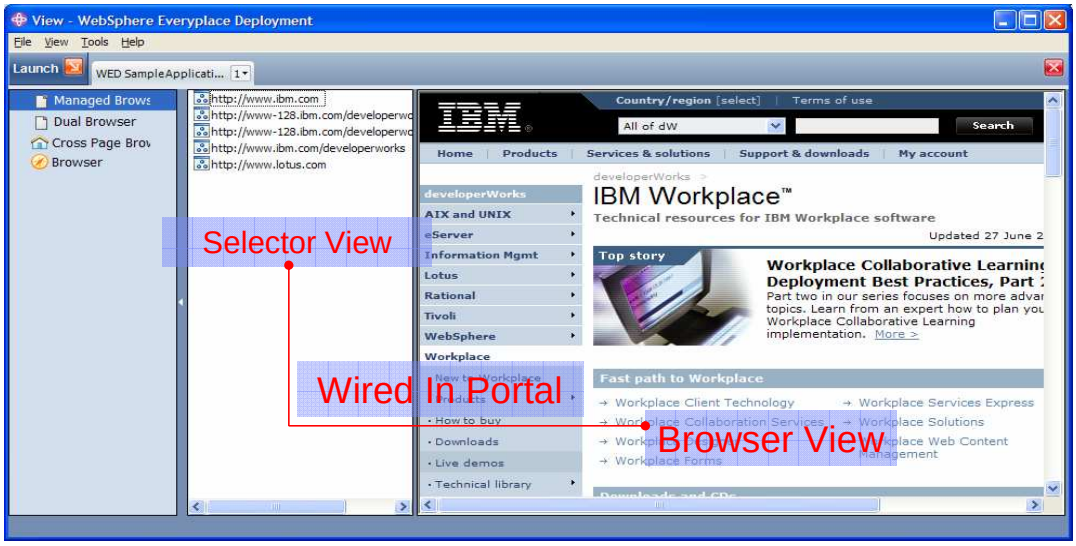
1. The first step is to create a placeholder portlet for each of our SWT views. The reason this is called a placeholder is because the portlet will not “do” anything when displayed on the portal. We are simply using the portlet to register our WSDL for wiring and then for defining where our view will show on the screen.
2. Each of the portlets will map to its respective SWT view. We will use the Rich Client tab supplied by the NCI to do this association.
3. Since we are just wanting to register our properties with the portal broker we can use the same WSDL as our SWT component.
4. Next, we will create our application template and then the application instance using the portal tools

Key technologies exploited:

- The Composite Application Infrastructure to deploy an SWT based application
- Declarative wiring through Property Broker on the portal and client
- Dynamic provisioning with no restart - Open the application from the CAI Portal Catalog and it will be downloaded/provisioned and then launched.
- Reading Portlet preferences from the SWT view - 4 additional URL's are set by the portal admin and shown in the selector.
- Drag and Drop with Property Broker Property Values - Drag and Drop URL's between the selector views
- Click to Action - Right click C2A menu shows compatible Actions with selected Property.
- Cross Page wiring - clicking the lower URL selector launches a different page.

The following is a list of key technologies exploited by this sample application. Take a moment to read each line on this slide.

Sample screen capture



This is the sample application. From left to right, we have the Application Navigation view – which is part of the CAI architecture and shows up as a page switcher for the different pages in your application. Next we have our URL selector view where we have a list of URL's. Lastly, we have the embedded browser view on the right.

We will wire the URL Selector view with the embedded browser view. The wiring is possible by having each of the views register the actions and properties with the broker on both the portal and the Expeditor client.

Screen capture – Portlet to SWT view

The Portlet to SWT sample is architecturally the same construct as the previous screen. The primary difference is the view on the left is an actual portlet where you enter a URL, click Submit and the URL is sent over to the browser on the right.

Once again, each view (the portlet and the browser) registered their properties and actions with the browser and then were wired together using the Portal wiring tool.

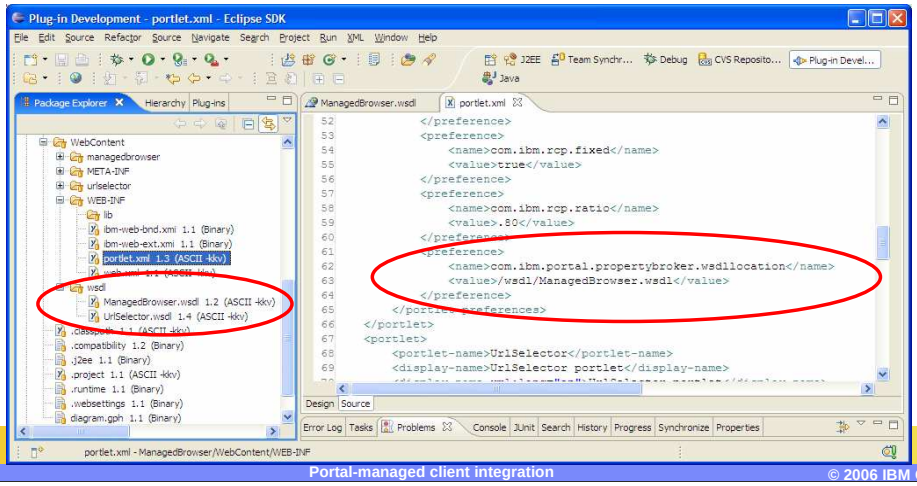
How do you develop this kind of sample?

1. Create your portlet in Rational® Application Developer
   - You can optionally use a place holder portlet
   - Creating a specific portlet gives you a direct mapping between the client and server components
   - All portlet metadata can be declared inside of the portlet
     - Feature ID, view ID, etc.
2. Create your Property Broker WSDL file
3. Bundle your portlets into a single WAR file.
4. Deploy your WAR file to the portal server

The instructions on this slide describe how to develop this sample.

Creating your portlets

- This sample has no portal code – using the sample RAD portlets from the portlet wizard.
- Create the WSDL files for the Actions and Properties.
- Update your portlet.xml with the WSDL location in the portlet preferences.

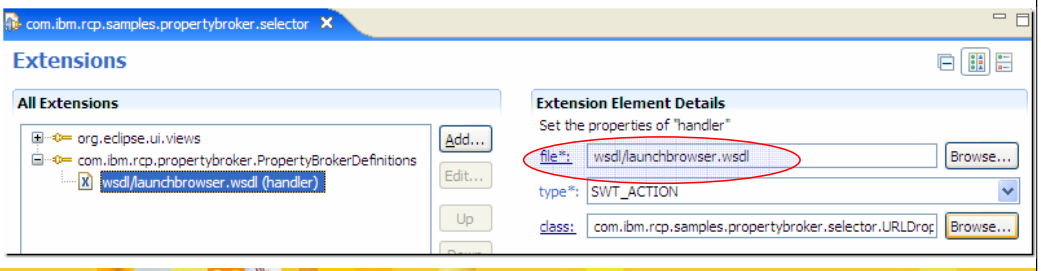


When you create your portlet you can simply use the samples from RAD. Create a Portlet using the RAD wizard. You will then supply your WSDL file and edit the portlet.xml to identify the location of the newly inserted WSDL file.

This is the file that tells Portal what properties and actions your portlet exposes for others to wire against.

How do you develop this kind of sample?

1. Create your Eclipse based view.
2. Use the same WSDL file in portal and create the client side extension.
3. Code your view to publish properties.
4. Code your action to update your view on property changes.
5. Deploy your Features and plug-ins to an HTTP site.



Next, we will create our Eclipse based view and also identify the WSDL in the PropertyBrokerDefinitions extension point. The procedure outlined above outline shows what steps were taken in developing and deploying this plug-in.

Selector view publishes selected URL

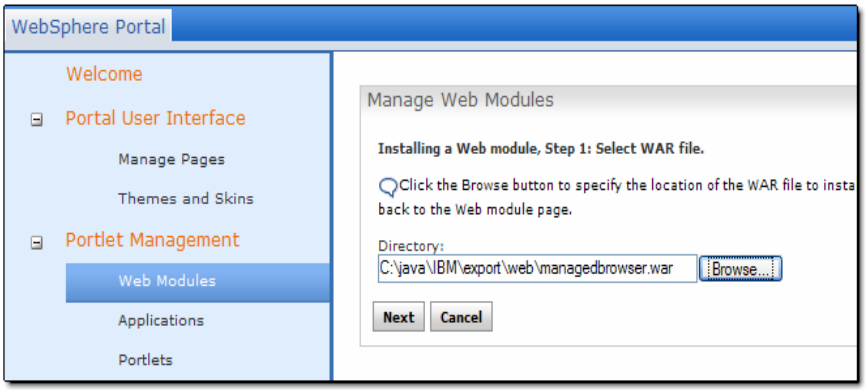

```
PropertyValue value = PropertyFactory.createPropertyValue(prop, selection);

SWTHelper.changedProperties(new PropertyValue[]{value}, this);
```


Here we show the actual API used to publish the property from the URL Selector view.

Deploying your sample to the Portal Server

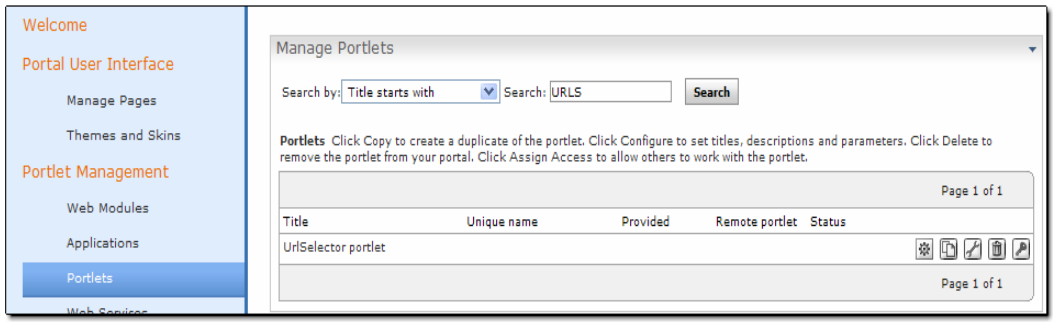
- Install your WAR file as a WEB Module.



After your Portlet, Plug-in and Feature are created, it is time to put the artifacts onto the portal server. This screen shows where you install the WAR file which contains the placeholder portlet.

Deploying your sample to the Portal Server

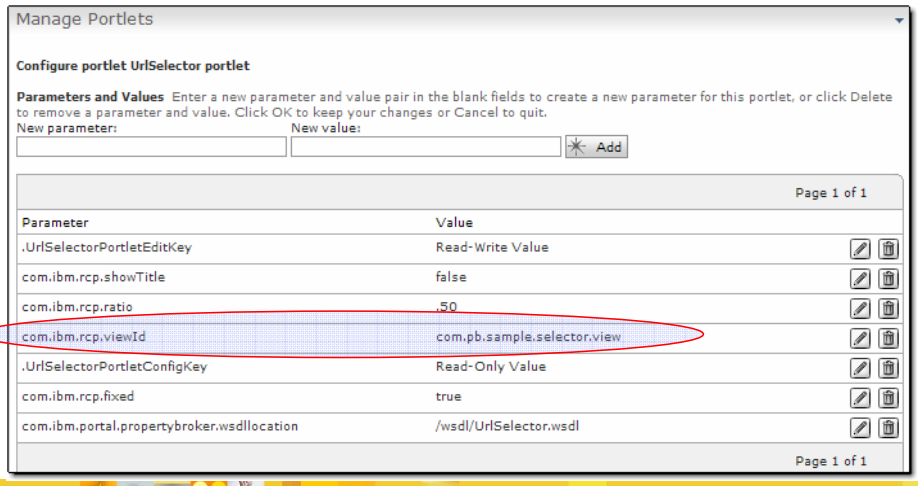
- Verify the portlets were installed under Portlets.



You can verify the portlets are installed correctly by verifying that you have not received an error on import and by checking that the portlet is now listed under Portlet Management > Portlets. Here I search for “URLS” to find my portlet.

Deploying your sample to the Portal Server

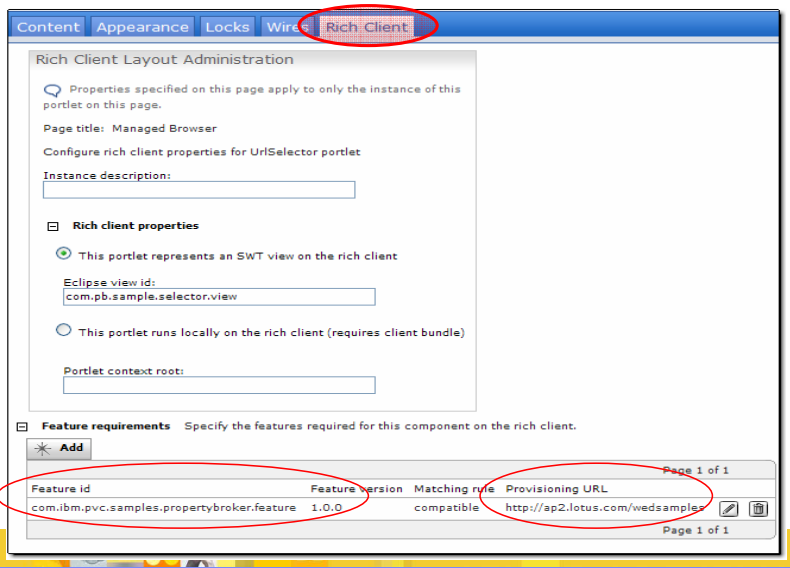
- Concrete parameters were set in Rational Application Developer



By examining the portlet preferences, you can see the preferences which we set in RAD. This is a good way to set the concrete preferences that should be shared across all instances of this portlet.

Deploying your sample to the Portal Server

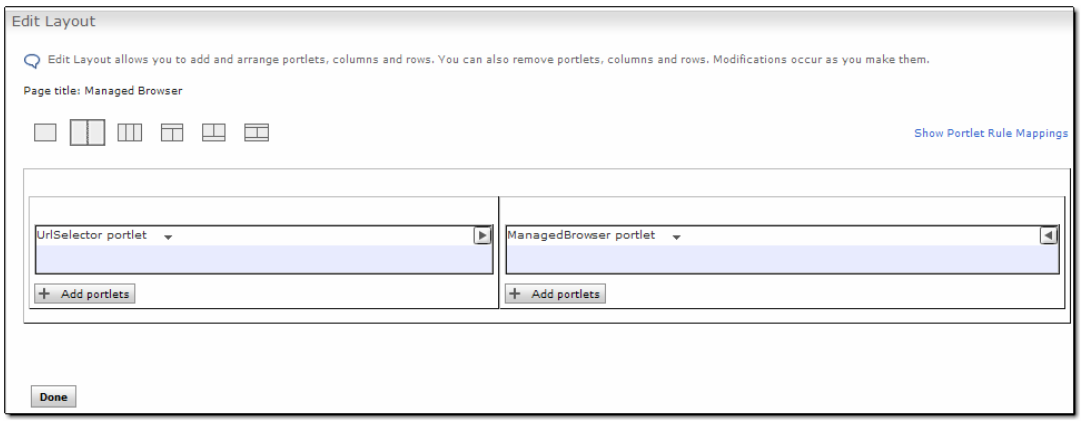
- Use the Rich Client Layout Admin portlet.
- Set the icon, features, and so on.



You can now use the Rich Client tab to associate this portlet with the SWT view and the Eclipse feature. By specifying the Eclipse feature and an update site to install the feature you are telling the Expeditor client to go and retrieve that feature if the version rule was not matched. The provisioning of the feature will happen when the application is opened from the Expeditor Applications Catalog.

Deploying your sample to the Portal Server

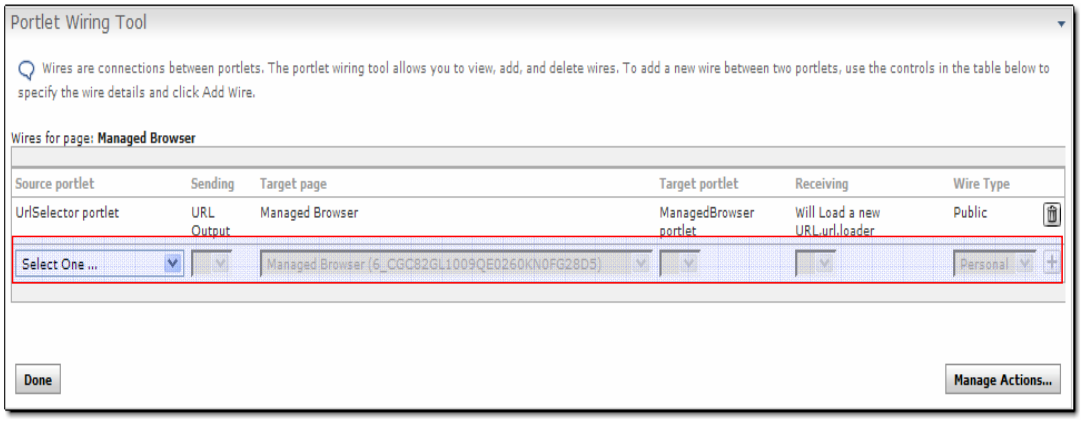
- Create an Application under Templates.
- Create your page and layout your portlets.



You can create your application from the Blank Portal template and place your new portlets on a screen. Here we see the screen layout looks similar to our running application.

Deploying your sample to the Portal Server

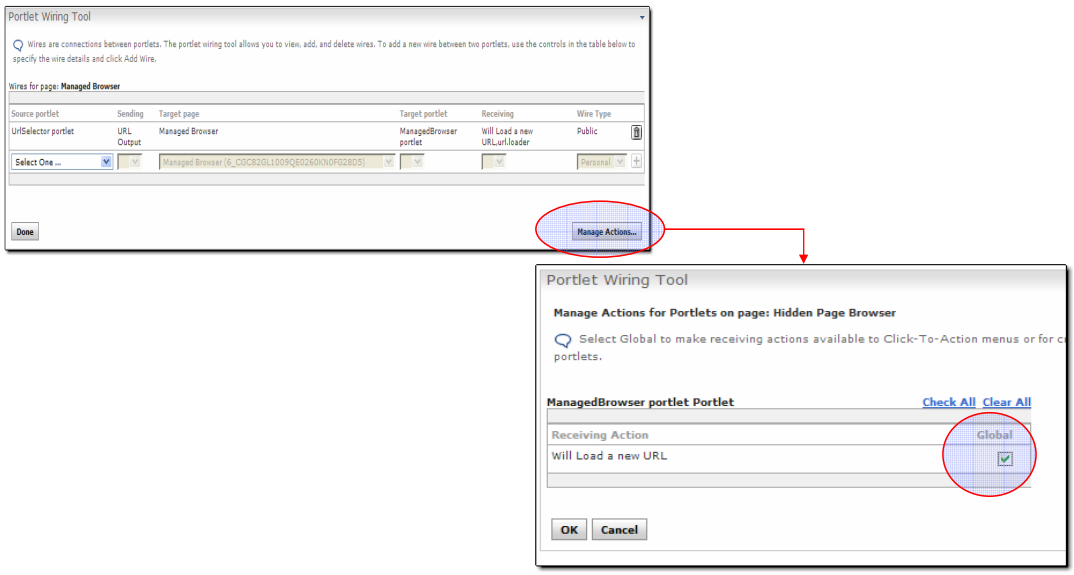
- Create wires between your portlets.
- These wires will be used on the client.



Once the portlets are on the screen we can associate the properties and actions by declaring a wire. The wire will map the output from one portlet to a compatible input parameter on the receiving portlet.

Deploying your sample to the Portal Server

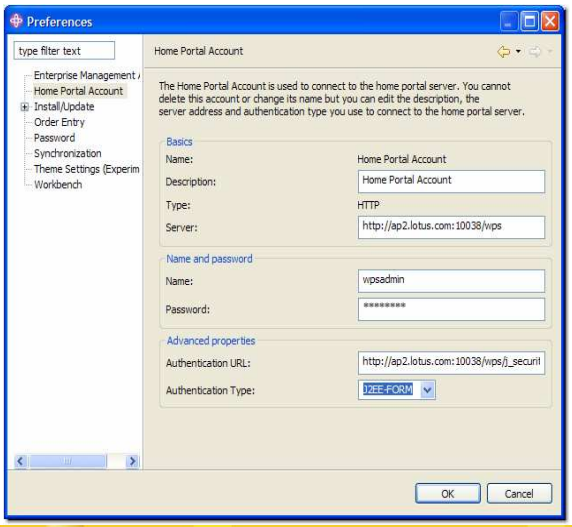
- For cross page wires, you must make your actions Global.



If you ever want to create a cross page wire, then you must remember to click the Manage Actions button in portal and select that the action is global. This will make the page and portlet selectable in the wiring tool for a cross page wire.

Configuring the WebSphere Everyplace Deployment client

- Use the preference screen to set the Portal Account connection information.



Finally, you are ready to connect the Expeditor client to the portal server. You can do this by configuring the Home Portal Account preference screen to point to the portal server.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Domino Everyplace IBM Lotus Notes Rational WebSphere Workplace

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

We hope you found this information helpful. This concludes this presentation.