

Rational Application Developer: Portal Development

1a <0:00>

Rational Application Developer is a comprehensive integrated development environment built on the Eclipse open source platform. Application Developer accelerates the creation of portal themes, skins and applications by providing visual portal development tools and a built-in WebSphere Portal test environment.

1b <0:23>

In this demonstration we will create and test a contact list portlet.

1c <0:30>

The new portlet project wizard allows developers to create portlets based on the IBM portlet API, or the industry standard API, JSR 168.

1d<0:44>

The wizard offers several portlet types including Java Server Faces. Faces technology allows developers to build JSP pages using pre-built user interface components.

1e<0:56>

Here, the portlet wizard prompts the developer for general portlet setting information, and then automatically generates a completely functional portlet, in a project structure conforming to the J2EE standard. Portlet projects include java source code and skeleton JSPs.

1f<1:19>

Once the wizard is finished, the portlet view mode JSP file is opened in Page Designer, where developers can drag and drop user interface components onto the page, and add portal specific components such as Click-to-Action properties and person links.

2a<1:35>

In Page Designer simple text changes can be made quickly without editing HTML source. Attributes, like font and color, can be modified in the Properties tab on the bottom of the screen.

2b<1:48>

In our sample portlet, we will use some previously developed Java Beans to display a Contact List. These beans are available in a jar file on the local file system, and can be imported to the portlet by dragging and dropping, or by using the Import wizard.

2c<2:11>

The Project Explorer displays a custom view of the Portlet project, which includes deployment descriptors, Java Resources and Web Content files, like JSPs. Here we see the Contact and ContactList java files we just imported.

2d<2:29>

The Page Data view shows data sources defined to the JSP, and can be used to add general data components, such as Java beans and Web Services, to the page.

3a<2:40>

Here we use the Page Data view to define one of the JavaBeans we just imported. When the JSF ManagedBean checkbox is selected, Application Developer automatically updates the faces-config file for the project, making the bean accessible to the JSP.

3b<2:58>

The faces-config file defines Java Beans and navigation rules for Java Server Faces web applications.

3c<3:07>

Beans in the Page Data view can be expanded to display details. Note that the contactList bean contains an ArrayList, but the object type of the list is not available until runtime. Application Developer provides the option to specify the object type during development so object properties can be available to visual tools.

3d <3:32>

Objects in the Page Data view can be dragged and dropped to the page, quickly defining the output layout.

3e <3:42>

When objects are dragged onto the Page Designer, Application Developer displays a data controls configuration page, allowing developers to define the layout and order of output on the screen.

3f <3:56>

Based on these configuration choices, output controls for the selected objects will be created in a table layout. Table attributes can be easily modified using the properties tab.

4a <4:08>

Application Developer automatically generates a java pagecode file for each Faces JSP in the portlet. The pagecode file contains the underlying Java code that controls the objects behind the JSP.

4b<4:17>

In this example, we will modify the generated code to initialize the ContactList bean in the JSP.

4c<4:34>

The Java Editor provides specialized features for accelerating Java development, including code formatting, syntax highlighting, integrated debugging and Code Assist. The Code Assist feature displays a scrollable list of possible code completions.

5a<4:58>

Application Developer includes a WebSphere Portal unit test environment which supports testing and debugging portlets locally or remotely on IBM WebSphere Portal. Developers can create and configure server instances, step through portlets and set breakpoints, and even modify code while debugging without restarting the unit test server.

5b<5:19>

During test, the developer may define a new server, or select a previously configured server. Application Developer publishes the portlet application and displays the portlet on the test page of the selected WebSphere Portal server. If the Portal server is not already started, Application Developer starts it automatically.

5c <5:40>

Server messages are displayed in the Console tab. The Server tab can be used to monitor status or stop and start servers.

5d<5:50>

In the Web browser tab we see our new portlet running under test.

An on demand business environment requires developers to respond quickly to rapidly changing requirements. Rational Application Developer increases productivity, minimizes learning curves and shortens the development and test cycle allowing businesses to deploy high quality portal applications.