Chapter IX

Portal Development Framework

To understand portal applications and the principles of their development, we need to distinguish between a portal and a portlet. A portal is a framework that lets you plug in new features or extensions called portlets. In the same way that a servlet is an application component within a Web server, a portlet is an application component within a portal. A portal is a Web application. Developing portlets is the most important task in making portals function as the user’s window to information and tasks. Portlets are an encapsulation of content and functionality. They are reusable components that combine Web-based content, application functionality, and access to resources.

Portlets must be considered as complete applications, having multiple states and view modes. They have messaging and event recognition capabilities. Portlets are assembled in portal pages that make up a portal implementation. An important issue is the design of a portal page as the aggregate of several different portlets.

Portlets are similar to Windows applications in that they present their content in a window-like display on a portal page. The portlet window has a title bar that contains basic controls to maximize, minimize, edit, configure, and provides help to the application. From the portal user’s perspective, a portlet is a window on a Web-like page that provides access to a specific service or content.
The portal engine (also called portal server) provides the runtime environment - the portlet container - for the portlets that make up the portal implementation. The portlet container concept is fully compliant with J2EE in the sense of container functionality. The portlet container is responsible for providing the environment for the portlet’s lifecycle methods such as instantiating, invoking, and destroying portlets. The user interacts with portlets through Action-URIs. By clicking on a link inside of the portlet, the user can perform certain actions, for example changing the display view.

When a user interacts with the portlet, for example by pressing a link, the action results in a new URL (page). However, the user’s experience is such that the page did not change; only the portlet content may have changed. If the user presses the back button, the previous URL is displayed, but nothing has changed on the screen. This is because the portlet maintains its state. It is necessary to use so called Action-URIs for navigating forward and backward, rather than using the browser’s built in navigation (the back button).

The behavior described previously is often a source of frustration for both the user and the designer. It underlines the differences between a Web application using a single window associated with single component (servlet) and a portal application, which is an aggregation of multiple windows.

The portal infrastructure supports the core sets of services for the portlets. These services include:

- Access to user profile information,
- A framework for portlets to participate in events,
- A framework to communicate with other portlets,
- Access to remote content,
- Access to user credentials, and
- A framework for storing persistent data.

Part of the conceptual knowledge about portals is the requirement to understand how the portal stack works. In Figure 71, the users connect to the Internet or Intranet using their PCs equipped with a browser. The server has a portal server installed on top of the HTTP server (the HTTP server is designed to handle static Web content) and the Application server (the Application server deals with Web Services and generic Web applications).