Chapter X

Accessing Remote Locations

Most of today’s portal implementations provide a model that facilitates plugging various components (portlets) into the portal infrastructure. Portlets run locally on the portal server, process input data, and render output. A local portlet and a good caching strategy for the content improves the response times, performance, and scalability of portal systems. However, very often we need to access remote Web services. One solution is to use a local portlet to access a remote Web service via its interface, obtain the required results as a raw data stream, and locally render the results in a fragment. This approach is relevant for data-oriented Web services. An alternative solution is to equip the Web service with an additional interface in the form of a portlet. When the Web service is called, it returns the entire portlet instead of raw data. This approach is suitable for presentation-oriented Web services.

The second solution appears to be more convenient. The remote Web Services appear as remote portlets, including presentation and application logic. Remote portlets produce portlet markup, which is sent to the requesting (local) portal server and is accessed by the enterprise portal through portlet proxies. Instead of just providing raw data or single business functions that still require special rendering on the portal side, remote portlets are visual representations of (remote) Web services. They behave like Web applications that can be invoked through a standard interface using generic portlet proxies on the local portal side (Figure 92).
The vision for the portals of the future is that they must be able to run local portlets, include remote portlets, and share local portlets with other portals by making them available as remote portlets for Web services (the concept is becoming known as Web Services for Remote Portlets, or WSRP).

Jetspeed, WebSphere Portal, and the Sun ONE Portal allow enterprises to plug the portal server into a UDDI registry and make portlets available as presentation oriented Web services. Portals are frequently deployed as a means to make enterprise applications more accessible to end users. Therefore, publishing an application as a Web service further expands the boundaries of application usage.

Portal clients access portals via the HTTP protocol, either directly or through appropriate proxies or gateways like WAP or voice gateways. To accommodate different devices, portals need to support different markup languages. During the aggregation phase, the portal invokes all portlets that belong to a user’s page through the portlet APIs. Two different kinds of portlets are available:

- **Local portlets** run on the portal server itself. They are deployed by installing *portlet archive* files on portal servers, and they are invoked by the portal server directly through local method calls.
- **Remote portlets** run as Web services on remote servers. They are published as Web services in a UDDI directory so that they are easy to find and bind to. A remote portlet Web service is bound to a portlet proxy and registered in the portal’s portlet registry. Portlet proxies are used to