Chapter VII

Web Service Design Issues

Web Services are gaining in popularity because of the benefits they provide. One of the major benefits is their support for interoperability in a heterogeneous environment, which leads to the capability to add systems and solutions that require different platforms. As long as the various systems are enabled for Web Services, the services can be used to facilitate interoperability. Web Services let enterprise application developers reuse and customize existing information assets. Web Services provide developers with standard ways to access middle-tier and back-end services, such as database management systems and transaction monitors, and to integrate them with other applications.

A Web service implementation (service endpoint), a Web service client, and a server’s interfaces to the service form a cooperating structure. Because all of the service interfaces have a consistent implementation, developers do not need to acquire new skill or learn new styles to work with various services. It also means that many components can be reused or modified and then reused. Building these components requires consideration of some general rules that both Web service client and service endpoint developers must understand. The following sections discuss design issues that are relevant to building service endpoints, server interfaces, and Web service clients.
Service Endpoint Design

To enable interaction with clients, Web Services must expose their interfaces. Web Services can either be built from scratch or can be added to an existing application, thus making it look like a Web service. There are two separate issues in the design:

- Design of Web service capabilities, their interoperability, and efficiency of logic flow; and
- Design of business logic to support business processes.

Business logic is typically designed with no regard to whether the application is used as a Web service and with no consideration given to the interoperable interface structure.

Interoperability is the main focus for the design of Web service capabilities. This design should start at a higher level of abstraction (architecture stage), and it should involve assigning the responsibilities to layers. We start with splitting the responsibilities by assuming (similar to XML-based Web service endpoint) that each Web service endpoint has a service interaction layer (service interface for communication with its clients) and a service processing layer, which is the actual implementation of business logic processing (Figure 70).

From this perspective, a typical Web service developer must resolve several design issues:

Figure 70. High level view of Web service