Localizing and Using Services in Web-Enabled Environments

Ying Zou and Kostas Kontogiannis

University of Waterloo
Dept. of Electrical & Computer Engineering
Waterloo, ON, N2L 3G1
Canada

Abstract

The Web browser technology has revolutionized in the 1980's the way Internet is explored and utilized for gathering and presenting information. With the emergence of distributed object technologies and new programming languages Internet is now moving from a world-wide information pool towards a service providing facility. This position paper examines the origin and architecture of the "services-based" Web and, discusses techniques for searching and integrating content and services with the Web as the primary medium.

1 Introduction

The Web browser technology, the so-called first-wave of the Internet [1], brings the explosive usage of the Internet in terms of world-wide information shared resources. The convergence of the Internet and distributed-object technologies, extend this "information-based" Internet to the "services-based" Web. This evolution is referred to, as the Internet's second-wave [1], where software services and content are distributed over the Internet, Intranet, or Extranets. In this context, services are denoted by the behavior of software that is accessible as a component in the Web domain. As more data and software components are made available on the Web, these software services have unique addresses and relate to processes that can be dynamically executed on a server upon the request of arbitrary Web clients.

As an example, consider a scenario of the "services-based" Web, whereby a global infrastructure enables software components that have been developed independently, be integrated with each other in order to facilitate complex programming tasks. In this way, content (data) located virtually everywhere in the world and, software components can be combined. This integration can happen dynamically by allowing requirements, functionality descriptions and, signatures of components be published on the Internet so that client processes invoke them without necessarily downloading all the components to the local client machine. In this context, a software agent can search for available software services on the Internet, select them to compose new applications, invoke the services remotely, and obtain the results utilizing the Web. Such an infrastructure offers the opportunities for the Web-based enterprise integration and e-commence to function.

2 Distributed Object Technologies

Essentially, the "services-based" Web relies on the emerging concept of "object Web" [12] which assumes universal connectivity, broadly distributed environments, and cross-platform interoperation of both infrastructure and applications [2]. The "services-based" Web adopts the distributed object technologies, and can be constructed from the interoperability among the language and platform-independent applications. Therefore, it moves the distributed applications beyond the boundaries of the simple client-server communication and message passing, towards a larger scale collaboration infrastructure.

Definition of Distributed Object

In [11] a distributed computing system can be defined as a system of multiple autonomous processes that do not share the primary memory, but that cooperate by sending messages over a communication network. This definition denotes the behavior of physically separated components and logically autonomous

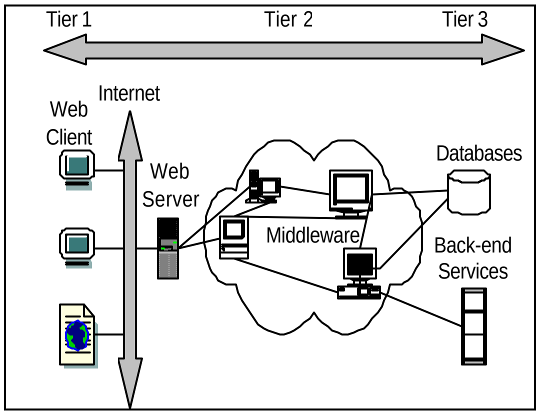

Figure 1: Three-tier Architecture

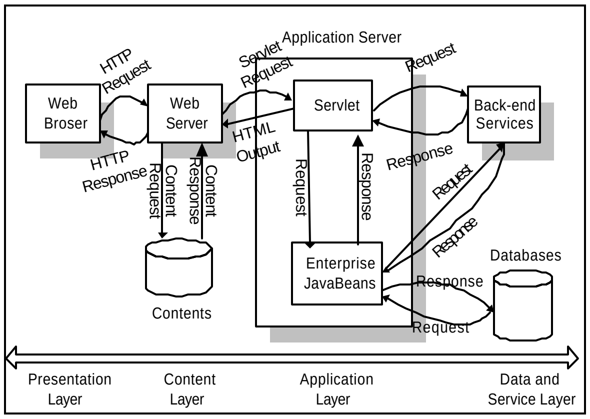

Figure 2: Control Flows in Three-Tier Architecture

modules communicating via message passing. On the other hand, a distributed system can also be defined as a system whose services and functionality are encapsulated in objects so that only specific interfaces allow to mediate message passing between processes. These objects are referred to, as distributed objects.

The distributed objects encapsulate software components at different level of granularity and detail. For example, the coarse-grained components can be executable legacy applications and the fine-grained component can be specific software components, which provide specific functionality. Although the fine-grained components are intended to be reused arbitrarily are often too small for practical use and, they may require considerable work from programmers for being customized to a given application context [7]. For the sake of reusability, it is more practical that legacy software systems and be wrapped as distributed objects and be easily integrated with other applications.

In real life, the distributed objects can be Java servlets, Enterprise JavaBeans and, CORBA objects, which communicate by Java RMI or, IIOP protocols.

Three-Tier Architecture

The three-tier architecture is fundamental to the deployment of the distributed objects on a Web-based environment. A Web server acts as a front-end between client processes (Tier 1) and to the middleware (Tier2). The middleware in its turn facilitates the interoperability of many disparate software applications or distributed objects (Tier3). The Web clients (Tier1) send the requests to the Web server for the software services to be invoked over the HTTP protocol. The Web server invokes the requested service on behalf of the clients (shown in Figure 1). Currently, the major middleware technologies include Java RMI from SUN, CORBA from OMG, and DCOM from Microsoft.

3 An Infrastructure for Locating Web Services

Control Flows in Three-Tier Architecture

From another point of view, the generic three-tier architecture (Figure 1) is further separated into four layers [1]: *a)* presentation layer (Web browser); *b)* content layer (Web sever); *c)* application layer (application server) and; *d)* back-end data and services layer.

The Web server and the application server can be located in the same machine or on different machines. The Web server is responsible to accept HTTP requests from Web clients, and delivers them to the application server, while the application server is in charge of locating the services and returning responses back to the Web server. For example, IBM's HTTP server, which is a Web server, uses IBM WebSphere application server as its front-end.

On the other hand a thin client in the presentation layer has little or no application logic. It sends the request via HTTP to the web server for an interactive view of the static and dynamic information.

To provide the static information, the Web server can maintain a content repository, or a file system, where the information-based resources are stored and serve the static HTML pages. Upon receiving a request from the client, the Web server can retrieve the requested document from the content repository and send it to the client. In this case, the client entirely relies on the Web server. Programming languages, such as Java, and scripting languages, like JavaScripts and CGI, can be used to access databases and other on-line resources.

To provide the dynamic information, generated by software services, the Web server needs to constantly interact with the application server. A servlet, which is a Java program, provides the dynamic HTML content to clients. When the Web server receives the request for a servlet, it re-directs the client's request along with the parameters to the application server, which loads the servlet and runs it. The servlet, an actual Java class, can make calls to back-end services, other servlets or, to the Enterprise JavaBeans. In the end of the process, servlets construct the response in the HTML format and passes the results to the Web server for transmission to the client. [13]

In particular, the EJBs (Enterprise JavaBeans) are server-side components which act as building blocks for the other EJBs and servlets inside the application server, or for other Java applications and Java applets available in the Web via IIOP.

Categories of Software Services

According to the client processes, the software services via the Web can be divided into two categories. The first category focuses on clients for which the servlets provide the end services via Web browsers. As stated before, the servlets rely on the Web server to present parameters and other information required for a service to be invoked. Consequently, the Web server transmits the input to the servlet on the client's behalf. After execution, the servlets generate results that can be presented or rendered in HTML or any other form.

The second category focuses on Intranet or Extranets where, software services can be provided as distributed objects in the form of servlets, Enterprise JavaBeans, JavaBeans or back-end services. These distributed objects are reused to construct another applications without the aid of the Web server.

Specifications of Software Services

To facilitate the "service-based" Web, a meta-level description for the software services should be published at the same time as the distributed objects are deployed onto the application servers. The specification of the software services includes the following information:

- General compile-time and runtime properties, including keywords for describing its functionality, execution environment.
- HTTP URL links for the servlets Web page, the IIOP hostname for the name server along with the port number, in which the name server listens to the naming request.
- Interfaces of the distributed objects, specifying the available behaviors of the objects, the expected parameters and the types of the return value.
- Inputs for servlets are simply the HTML forms. It is necessary to describe the range of the parameters.

Such specifications supply rich information for the available software services, and ease the way to understand and locate them.

Search Engine for Software Services

A search engine is needed to locate distributed software services much like a search engine locates data in Web pages. In its current infrastructure, the description of Web documents is not stored in any structured way. Therefore, it is very common to get low precision and low recall results using the current Web search engines, especially when the semantic content of the queries is not precise. The main reason for this problem is that the search engines scan the flat abstract information from the Web servers to locate keywords without taking into account the particular query context.

In a similar way, searching and dynamically linking content and services using the Web as the medium, require that description information for components be represented in a structured but flexible way. Signatures of components, pre- and post-conditions as well as, triggering mechanisms need be specified. XML provides a natural way to represent the specification of software services using the hierarchy relationship inherent in its tags. Service localization engines should be modified accordingly. With the DOM

(Document Object Model) API, the localization engine can easily navigate the hierarchy structure of the XML document tags, and extract the different parts of such information. It is noticeable that the description keywords for the functionality can be structured according to the context. It allows the clients to search the services by functionality not just static descriptions. Such an environment is under development at the University of Waterloo in collaboration with IBM Canada, Center for Advanced Studies. The application area deals with the integration of services in e-commerce and e-business to business environments. The invocation of services is based on scripts encoded in XML and is enacted using the Event-Condition-Action paradigm.

4 Conclusion

With the aid of service description and service localization mechanism, the "services-based" Web transforms the traditional way of pulling static information from repositories, to pushing dynamic information to clients upon request. As a result, the new infrastructure makes the reuse of software components available in a much larger scale, shortens the time to architect new applications, and eases the enterprise integration and e-commence operations. Meanwhile, it provides a new market for e-commence and e-business by allowing user configurable and customized content and services be delivered using existing World-Wide-Web infrastructure.

References

[1] Paul Dreyfus, "The Second Wave: Netscape on Usability in the Services-Based Internet", IEEE Internet Computing, March/April 1998.

[2] Cynthia McFall, "An Object Infrastructure for Internet Middleware: IBM on Component Broker", IEEE Internet Computing, March/April 1998.

[3] Gary R. Voth, Charles Kindel, and Jon Fujioka, "Distributed Application Development for Three-Tier Architectures: Microsoft on Windows DNA", IEEE Internet Computing, March/April 1998.

[4] Ellis Horowitz, "Migrating Software to the World Wide Web", IEEE Software, May/June 1998.

[5] Vivekanand P. Kochikar, "The Object-Powered Web", IEEE Software, May/June 1998.

[6] Hans-W. Gellersen and Martin Gaedke, "Object-Oriented Web Application Development", IEEE Internet Computing, January/February 1999.

[7] Israel Ben-Shaul, James W. Gish, and William Robinson, "An Integrated Network Component Architecutre", IEEE Software, September/October 1998.

[8] Israel Ben-Shaul, Gail Kaiser, "Coordinating Distributed Components Over The Internet", IEEE Internet Computing, March/April, 1998.

[9] Writing Enterprise Applications, http://developer.java.sun.com/developer/onlineTraining/J2EE/Intro/session.html#setup.

[10] Israel Ben-Shaul, James W. Gish and William Robinson, "An Integrated Network Component Architecture", IEEE Software, September/October 1998.

[11] Reaz Hoque, "CORBA 3", P573, IDG Books Worldwide Inc., 1998.

[12] Robert Orfali, Dan Harkey, and Jeri Edwars, "Instant CORBA", John Wiley & Sons, 1997.

[13] Joqquin Picon, et al, "Enterprise JavaBeans Development Using VisualAge for Java", http://www.redbooks.ibm.com.