White Paper:

Consuming Web Services with the Surge™ Runtime Environment

Table of Contents

Abstract

The next evolution of the Internet is rapidly approaching. A global network once used for simple file transfer and email communication, transformed by the World Wide Web into a medium for transmitting content and conducting commerce, will now become a dynamic environment for distributed computing, enabling networked applications built on shared components, or Web services. Software-as-a-service will become a reality, and innovative ways of sharing data and content will become standard. This is being made possible by a new generation of industry standards like XML, SOAP, and UDDI, and will require a new generation of server and client-side enterprise software. Curl Corporation is on the forefront of delivering rich client software, enabling organizations to create robust client applications for consuming Web services.

Introduction

Background

Web services are already a reality. The sheer market influence of the companies involved ensures that distributed software architectures will take hold and change the way companies do business. Sun ONE, Microsoft's .NET strategy, and initiatives from HP and IBM have shown that the software industry is relying on Web services to drive the next evolution of Internet and intranet technologies. Although the specific focus and implementation of these initiatives vary considerably, software as a service and distributed applications are core tenets of each.

Emerging Standards for Web Service Deployment

Although there is not universal agreement among industry players about which standards to adopt, there are several specifications which are being adopted quickly by a majority of the Web services market. At the core of universal data exchange is XML, the W3C Recommendation which has been hailed as the future of data description, presentation, transport, and much more. The Curl™ API already provides integrated support for an XML parser, which allows for dynamic data retrieval and presentation on the client. XML is now being extended even further as the basis for many of the protocols which are being considered for widespread industry adoption to enable Web services, in particular SOAP (a communications protocol), WSDL (a description language), and UDDI (a registry of available services).

See Appendix A for a more thorough discussion of these standards.

Where Do We Go From Here?

Having a common communications protocol, a universal way to describe these services, and a registry to find them are all important steps along the way to effective deployment of distributed network applications. With this foundation, the challenge for enterprise software vendors is how to tie these Web services together to create value for their customers. There is still one critical piece missing, however: the necessary software to deliver the power of Web services to the client in a rich, interactive way. Current client-side technologies such as HTML and JavaScript do not provide the end user experience that customers are demanding. These static, point and click Web applications cannot match the rich, interactive interface that end users are accustomed to with desktop applications. Until now, there hasn't been an alternative technology platform available to tie Web services together with a rich client interface, providing greater interactivity and scalability.

The Curl™ Solution

The Curl™ platform includes the Curl language for rich client development, the Surge™ Runtime Environment, and the Surge Lab™ Integrated Development Environment. The platform was designed as a powerful, extensible platform for Web applications, offering the ability to integrate the text formatting and content presentation capabilities of HTML/JavaScript with an object-oriented programming language. This presents a true enabling technology for Web service developers. In addition, the Surge™ runtime environment has an integrated XML parser, allowing universal data exchange and connectivity right out of the box. And with native support of Web services standards, the Curl platform enables even more rapid development and deployment of Web services and distributed applications.

The Curl platform offers a unique opportunity for the Web services application developer: a unified client-side environment for aggregating Web services, offering the power and responsiveness of a desktop application delivered directly to the Web browser over any network.

Improved User Productivity

Applications written for the Surge runtime environment offer superior responsiveness and an enhanced user experience through vastly reduced download sizes and a powerful client-side engine. Traditional problems associated with Web-delivered application interfaces stem largely from the limitations of the presentation layer; the Curl platform provides an innovative approach to solving this problem by moving most formatting and computationally intensive tasks to the client.

In a traditional Web interface, all data and presentation logic are generated by the server on the fly, with the client acting simply as an interpreter. For each action or data request by the end-user, a new request is made to the server, which generates a response. This response is usually a combination of markup, text, data, and scripting. Delivery of this response involves a time delay which varies by the size of the page to be displayed. For end-users accustomed to desktop applications running locally, which respond instantaneously to user input, these delays make the application frustrating and unresponsive.

For example, let's say a Web services provider has developed a simple service providing online financial information. A Web application consuming this service simply needs to make SOAP calls to query this service. There could be other application developers utilizing this same service in exactly the same way. Since the underlying backend technology is the same, with similar responsiveness and reliability, this front-end developer has only one way to differentiate himself from any other application developer: by providing an intuitive, flexible, and responsive user interface (UI). Limited by server-side technologies to make SOAP requests and interpret responses, the developer can dynamically create HTML and JavaScript to act as the presentation layer. For the end user, this creates a significant delay when trying to update or view the information, as each change requires another round-trip to the server and more server-side processing. Meanwhile, the client simply sits idle waiting for its next data set to display the required information.

Alternatively, one could create a client-side interface using Java™. In practice, however, the development of flexible, extensible, and customizable user interfaces in Java has proven difficult. With large download sizes and platform-specific implementations of the Java™ Virtual Machine (JVM), it becomes impossible to guarantee the experience that end users demand. In addition, XML/SOAP support isn't native to all JVMs; to use these protocols in a Java™ applet, the developer must rely on all clients to have the proper VM extensions and configuration.

Writing a user interface for the Surge runtime environment enables the dynamic, real-time data delivery that developers are seeking without sacrificing the end user experience that customers expect. When a user browses an application built on the Curl platform, the code is downloaded to the client along with any supporting packages on an as-needed basis. These packages and data may be reused, allowing further requests to and from the server to be used exclusively for data or additional component retrieval. The formatting, presentation, and manipulation of the data is accomplished exclusively on the client.

In our previous example, the client-side power of the Curl platform offers the flexibility to create a rich, interactive UI for the application without the performance or bandwidth limitations of current Web solutions. This can be accomplished in two different ways, depending on developer preference. The first option is using established server-side technologies to make SOAP requests and parse SOAP responses. In doing so one can create a custom back-end which acts as middleware between the client and the Web service. Thus, an application built on the Curl platform can be downloaded by the end-user, and connect back to the middleware to retrieve any necessary data. Upon receiving a request from the user for additional data, the application creates an XML request to the middleware running on the backend server, parses the response, and displays the data in any desired format.

The second option involves utilizing the Surge™ runtime environment's native SOAP support. An application built on the Curl platform can make SOAP requests and parse SOAP responses directly from the client. These networking operations previously

limited to the server can now be performed by the Surge runtime. Each response will vary in size based on the data requested but for the most part will be limited to a few kilobytes, which to the end user seems nearly instantaneous, even over the slowest connection. Data can now be manipulated dynamically on the client in a variety of ways; the only limitation is the developer's imagination.

In short, end users feel as if they are interacting with an application, not a series of slow, static Web pages. By enabling this enhanced usability and user productivity, the Curl platform provides a true competitive differentiator for Web service developers providing distributed applications.

Extensibility

Another severe limitation of current Web technologies is that developers are forced to use the controls, formatting, and layout tools supported by whichever technology they choose. For example, to create a button in HTML, developers have two options: use the pre-defined HTML button object or use an image (.gif or .jpg format) to replace the button. To layout a page, most developers still rely on using tables, adjusting widths and alignment as necessary. While there have been attempts to allow the Web developer more flexibility for layout and formatting (DHTML, layers, CSS), these attempts have met with limited success due to divergent implementations across browsers and platforms. Even if the implementation were more uniform, the developer would still be limited by the HTML controls and layout tools.

The Curl platform was designed with extensibility in mind. The Curl API offers the ability to programmatically alter the appearance, position, and functionality of graphical objects. The developer now has the flexibility to alter properties of controls and other graphical elements based on design goals, programming logic, or user input. Using the example above, if one wanted to place a button on a page using Curl, the developer would have several options. Using the Surge™ GUI system, there are objects that are ready for immediate use, and you can alter many of their properties programmatically. In addition, you could create your own button objects, either derived from the ones supplied or written completely from scratch. The Curl API's flexibility puts the power in the hands of the developer, bringing a whole new dimension to Web application UI design.

Ease of Development/Maintenance

Bringing a fully featured programming language to the Web is not a revolutionary concept in and of itself. If one were willing to develop and maintain code for multiple implementations of JavaScript/DHTML across different browsers/platforms, there would be many programming features and logical constructs they could include in a Web application. In practice, this has proven very difficult to do, because every change to the code requires testing on every variation of each browser. Limiting development and testing to one browser limits one's audience, which is not an acceptable trade-off for most organizations. Java provides an API for user interface development as well as

network connectivity. However, development costs and fragmented JVM implementations have hampered its adoption on the client.

The Curl API was created with ease of development and maintenance in mind. Our customers find that advanced development tasks are completed much more quickly using the Curl platform. One reason for this is that, being JIT-compiled directly from source code rather than an intermediate or statically compiled form, Curl offers real-time testing and debugging capabilities. When editing Curl code, the developer can immediately view the effects of changes by running the application in the browser, debugging in real-time. This is in stark contrast to the hours of development that are lost waiting for applications written in other languages to compile before testing can begin.

Development is also accelerated because much of the Curl runtime's powerful capabilities are encapsulated in easy to use, high-level APIs. This allows very readable code, similar to HTML source code but with greater simplicity, when editing text or altering the layout of text and controls. The top-level code for a user interface, for example, might consist of a few graphical containers with simple text integrated into the layout, with procedure calls to create any necessary controls or buttons. For more in-depth control of the UI look, feel, and functionality, the developer can create packages of their own objects and procedures. Maintenance and alteration of these packages can then be handled on a case by case basis, without having to rework the entire application. This greatly reduces the cumulative UI development time, allowing enterprise software companies to focus on their core development tasks.

Conclusion

Although the potential of distributed applications built upon Web services is nearly unbounded, there has been no client-side software infrastructure to enable their success. Curl Corporation is eliminating the roadblocks which have hindered development of next generation Web applications over the past 5 years.

Built from the ground up for Web application development, Curl offers high-level APIs to all common text formatting, UI controls, 2D and 3D graphics, and data/networking connectivity. The Surge runtime environment delivers all of these capabilities in a secure, stable environment which will perform similarly across multiple platforms and browser configurations. In addition, the Curl platform allows one to harness client-side power and move processing tasks off of the server, utilizing hardware and network infrastructure far more efficiently, and enhancing usability, performance, and ultimately user productivity. Furthermore, the elegant design of the language allows a vast increase in developer productivity.

For Web application vendors looking to aggregate best-of-breed Web services with a breakthrough user interface, desktop application-level functionality, and the portability and accessibility of an Internet application, no other solution compares to the Curl platform.

Appendix A – Web Services Standards

With support from some of the biggest names promoting Web services, the Simple Object Access Protocol (SOAP) has been identified as a leading candidate for the next generation Remote Procedure Call (RPC) mechanism. RPC allows one to send a message to a remote application, such as a Web service, and request that the service perform some action. This message is then received by the Web service, which determines what operations it can and will perform, and responds with either the requested data, or a message informing the user of the status of the request.

For example, there may be a Web service you want to utilize for user authentication, rather than building your own authentication mechanism. When a user attempts to log in, your application will send a SOAP request to the authentication service containing two major parts: an “envelope” describing what is in the message and how to process it, and the remote procedure calls which ask the authentication service to log in the user with a supplied username and password. After processing the request, the service will send your application a SOAP response containing two parts: an envelope indicating where the response is from and how it was processed, and the results of the procedure calls requested, which in this case would be a confirmation or rejection of the username and password supplied.

This is a relatively simple case of a remote procedure call. However, one can imagine very elaborate request and response pairs when Web services begin handling more complex business logic.

Once the world of Web services becomes a reality, how will one go about locating the proper Web service or eBusiness partner for their software or business needs? This question prompted the creation of the Universal Description, Discovery and Integration (UDDI) specification. UDDI is the foundation for establishing registries of companies and the services they provide. Two of these registries are currently hosted by IBM and Microsoft. Any company can submit their information to the UDDI registry, and then list the services, Web based or not, that they provide. Companies can now locate possible vendors or eBusiness partners by accessing this registry. For Web services, this directory provides the ability to describe functionality the service provides.

An integral part of making the UDDI registry successful is coming up with a common language to describe the interface, protocol bindings and the deployment details of network services. The XML based specification developed to solve this problem is the Web Services Description Language (WSDL). In doing so, the writers of this standard have created a universal way to describe the capabilities and functionality of any Web service. By agreeing on a common language, Web service providers have enabled a new level of interoperability which allows one to quickly make effective product decisions.