White Paper:

Delivering Enterprise Web Applications on the Curl™ Platform

Table of Contents

Executive Summary

The Internet has revolutionized the way people access and interact with data, and every major enterprise software vendor has attempted to capitalize on this paradigm shift by creating Web-enabled interfaces for their products. The vision of being able to access enterprise data remotely, on any platform, from anywhere in the world, is a compelling one. Unfortunately, there is a fundamental problem with delivering Web-enabled enterprise software: the current suite of Web technologies is ill-suited to provide the end-user experience enterprise customers demand.

Curl Corporation has developed an ideal solution to this problem. By implementing applications with the Curl™ platform, enterprise software vendors can eliminate three flaws inherent in current Web-based implementations- poor user productivity, limited extensibility, and high costs of application development & maintenance- and in the process enable true rich client capability.

Introduction

Background

User interface (UI) design and development is one of the most critical business activities for any software company; without an intuitive and powerful UI the capabilities of the underlying software cannot be fully exploited. Common practices around usability testing involve extensive analysis of many competing priorities: simplicity vs. completeness, customizability vs. standardization, etc. The explosion of network connectivity has compounded these issues by adding additional parameters such as accessibility and portability. The expectation that customers will only need to access their data from a PC at their desks is no longer valid; end-users are demanding a level of flexibility which was previously not a consideration.

As networked computing became more and more pervasive throughout the 90's, software companies began leveraging the Internet revolution to better serve their customers and differentiate their product offerings. Originally, stand-alone client/server applications utilized the local area network to distribute tasks such as database administration and server configuration. The adoption of Internet protocols allowed this trend to move a step further, as Web applications could be written to run inside a Web browser, allowing access to critical enterprise functions and data from anywhere in the world.

Challenges

As companies began to migrate or adapt their client interfaces for the Web, they realized that while the Internet as a network may be well suited to their needs, the languages and tools available to them were woefully inadequate to deal with anything more than simple text markup and content delivery. Server-side technologies such as application servers and portal platforms were developed to provide a level of dynamic interaction which was previously impossible with HTML alone, but they did little to address the bandwidth limitations resulting in sluggish responsiveness and poor end-user productivity.

In addition, the capabilities of these technologies were limited to a small subset of useful graphics display and text markup. Dynamic, interactive user interfaces were difficult if not impossible to implement. Data presentation was limited for the most part to static, table-driven formatting with limited utility.

Additional point solutions were developed to try to address these issues. The Java™ programming language (designed for application programming) was retrofitted to support Web-based miniature applications, running on a browser plug-in. However, problems with user interface development, download sizes, and security revealed that it was not designed for the Web. These shortcomings have made Java much more appropriate for server-side applications, and in practice it has been mostly abandoned

as a client side technology. Other multimedia formats have been introduced to provide enhanced animation and UI capabilities (Flash, DHTML), but they lack the rich programming capabilities offered by a true object-oriented language.

At the core of these issues there is one common element: none of these technologies were designed specifically for delivering dynamic, rich applications over the Web.

The Curl Solution

The Curl™ platform was designed from its inception as a powerful, extensible solution for Web-delivered applications. The platform includes the Surge Lab Integrated Development environment for rapid application development as well as the Surge Runtime Environment for deployment of scalable rich client applications across the enterprise. The platform also features the Curl language, which provides the ability to integrate all of the text formatting and content presentation capabilities of HTML/JavaScript with a powerful object-oriented programming language and which presents a unique enabling technology for developers writing Web applications. Together these three pieces of Curl technology combine to create a completely new, powerful architecture for Web applications.

The benefits of this design approach show that the Curl platform is ideally suited to solve the problems which confront application vendors attempting to create Web-enabled interfaces for their enterprise products.

Improved End User Productivity

Applications built on the Curl platform eliminate latency and enhance user productivity through superior user interface capabilities and vastly reduced download sizes. The traditional problems associated with Web-delivered application interfaces stem largely from the limitations of the presentation layer. Curl Corporation provides an innovative approach to solving this problem by providing a software platform that can move presentation logic and computationally-intensive tasks to the client.

In a traditional Web interface, all data and presentation logic is 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 determined 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 can make the application virtually unusable.

For example, let's say an ERP software vendor has developed a simple interface to pull data about raw materials in the supply chain and graph their levels over time. The initial page download is probably in the range of 60KB, including graphics, data, and markup. A user selects the desired material and submits a request. The Web server then pulls the requested data out of the ERP system and formats it in a simple table. The size of

this download is probably around 50KB. If the user would like to see additional data points, or change the way the data is sorted and displayed, the process must be repeated, requiring a 50KB download each time. If the user would like the data graphed, this requires another round trip, with potentially larger download sizes for these graphing images. Depending on the quality of data presentation desired and the load on the back-end servers, the graph generation alone may take quite a long time. Meanwhile, the client simply sits idle waiting for its next data set to display the required information.

Writing a user interface with the Curl platform enables the dynamic, real-time data delivery that developers are seeking without sacrificing the end user productivity enterprise customers demand. When a user browses a Curl application, there is an initial download to the client along with any supporting packages and data on an as-needed basis. These packages and data are cached locally and may be reused, allowing all 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.

Using our previous example, the initial download of the ERP interface would probably be smaller, around 30KB, because of the way the Curl platform compresses applications that have been deployed. Upon receiving a subsequent request from the user for data on a particular raw material, the applet would create a request for data from the server, parse the response, and display the data in any format designated by the programmer. Each request will vary in size based on the data requested but for the most part will be limited to a few kilobytes.

Regardless of the initial size, the Curl platform enables you to compress and expand these data streams on the fly. To the end user this 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. Computationally intensive tasks such as graph generation which could previously occur only on the server are now performed by the Surge™ runtime environment. In short, end users feel like they are interacting with a desktop application, not a series of slow, static Web pages.

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. It 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, the developer would have several options. With the Curl platform, there are objects that are ready for immediate use, and one can alter many of their properties programmatically. In addition, one could create his/her own button objects, either derived from the ones supplied or written completely from scratch. This flexibility puts the power in the hands of the developer, bringing a whole new dimension to Web application UI design.

Ease of Development/Maintenance

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 he/she 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, fragmented JVM implementations, and security issues have hampered its adoption on the client.

The Curl platform was created with ease of development and maintenance in mind. Our customers find that advanced development tasks are completed much more quickly using our software. One reason for this is that, being JIT-compiled directly from source code rather than from an intermediate or statically compiled form, the Curl platform offers real-time testing and debugging capabilities. When editing 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 platform'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, developers 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 and maintenance time, allowing enterprise software companies to focus on their core development tasks.

Conclusion

Curl Corporation is committed to eliminating the roadblocks which have hindered development of next-generation Web-enabled rich client applications over the past 7 years. Built from the ground up as a Web application development solution, the Curl platform offers high-level APIs to all common text formatting, user interface 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. For an enterprise software vendor looking to differentiate itself with a breakthrough user interface, only Curl offers a solution built from the ground up to deliver the interactivity, usability, and productivity of a rich client application with the portability and accessibility of an Web application.