Using AJAX to Build a New CICS User Interface

From the HostBridge Archive

When the first web-to-host gateways came onto the market in the mid-1990s, they provided a mechanism to access legacy applications through web browsers using static HTML documents. This approach made applications available to a wider group of users, but it did little to alter the user interface to make applications more intuitive. As these users became more accustomed to using Web-based applications, they grew frustrated with the constant page refreshes that occurred each time they submitted data to the mainframe applications.

The earliest attempts to address user frustrations included the development of thick-client Java programs or using browser-specific tricks to allow dynamic data refreshes on a web page. Unfortunately, these solutions were difficult to maintain since each Java client application would need to be updated and each browser-specific solution led to cross-browser support issues. Over the last few years, the use of asynchronous JavaScript and XML (or AJAX) has become the preferred method for creating dynamic content within browsers.

AJAX Components

Applications using AJAX are a collection of technologies that combine to allow direct manipulation of the data received by a web browser.

#AJAX Components

Here is a list of components with descriptions:

- **JavaScript (aka ECMAScript)**. Provides the programmatic interactions between the browser and the server. JavaScript can provide greater efficiency, simplicity and flexibility since most web developers do not have to learn an unfamiliar language or new development tools.
- **XML**. Data format used to transfer information between the browser and the web server.
- **DOM (Document Object Model)**. Represents the structure of an XML document and allows JavaScript to directly manipulate that structure as well as insert and remove content.
- **XMLHttp Request Object**. Interface available in all major browsers that allows asynchronous calls between the browser and web servers.

Other technologies such as XHTML and Cascading Style Sheets (CSS) provide the presentation layer within the browsers. With the exception of the XMLHttpRequest object, all these components are standards-based. (A standard for the XMLHttpRequest object is currently in the working draft stage under the World Wide Web consortium.)

Usability and the AJAX Model

The driving force behind the development of AJAX applications is usability. One high-profile example includes Google Maps, which expands the scope of the information displayed by loading map information outside the visual area of the browser. If the user moves the map in any direction the images are immediately available, which creates an improved user experience since the user does not have to wait to see the maps load as they scroll.

Another usability feature provided by the AJAX model is continuous feedback for actions. For example, when loading large amounts of data using the standard web model the user must simply wait to see if the action of clicking a link or submitting a form has had any effect. AJAX allows developers to include feedback mechanisms that alert the user that a request has been received and the response is coming. This assures users that the wait is not in vain.

Yet another usability feature is the fact that calls using AJAX maintain the context of the browser interface. Moving from one page to another using the standard browser model reloads pages and forces the user to re-orient themselves to any changes that might appear. By loading content into the existing page users keep the same frame of reference.

CICS and AJAX Uses

There are several cases where AJAX provides business benefits for CICS integration. First, AJAX enables true two-tier access to CICS. Instead of using a middle-tier server to process output from CICS and return a static page to the browser, AJAX allows the browser to contain the application logic and make calls directly to the mainframe.

AJAX applications also allow you to retrieve data from CICS, maintain the data in memory, and repurpose the data as needed. This means a single call to the mainframe can provide the basis for multiple views within the browser. For example, you can retrieve customer contact information from CICS applications or databases and use the return data to view individual customers, groups of customers, or sort and associate data in as many ways as you need. This greatly reduces the network traffic and all the calls to the loaded XML document are local to the browser for immediate responses.

Recommendations for AJAX Interface Design

Successful implementations of AJAX solve usability problems and take advantage of existing AJAX development projects to make programming easier. Here are some simple recommendations to get started.

1. Use AJAX frameworks. AJAX toolkits are usually open source and exist to make development easier. These frameworks include low-level interactions with the XMLHttpRequest Object (e.g., Prototype), higher-level tools to develop AJAX interfaces (e.g., script.aculo.us), and even full-blown AJAX development environments (e.g., TIBCO General Interface).
2. Add server-side processing. Combined client-side/server-side interactions (e.g., Ruby on Rails) provide support for service-oriented architecture (SOA), repositories for data and metadata, and support for team development and application management.
3. Design based on user patterns. A successful project from the technical side might be a failure in terms of adoption by users due to usability issues that do not improve the user experience and diminish the expected return on investment (ROI).

Tech Notes

This section answers common technical questions.

Q. Can we aggregate data from multiple domains within our AJAX application?

A. The security model surrounding AJAX applications limits requests to the domain from which a user loaded the page. So, if a user loads a page from the URL http://my.domain.com the AJAX calls cannot go to http://other.domain.com. However, you can write a simple proxy server to handle the interactions so all calls go to a single domain. In the examples given above, we wrote a proxy server using PHP to receive the browser request, call our mainframe, retrieve the XML document from HostBridge, and return the document to the browser.