WP. 11
ENGLISH ONLY

UNITED NATIONS STATISTICAL COMMISSION and
ECONOMIC COMMISSION FOR EUROPE
CONFERENCE OF EUROPEAN STATISTICIANS

UNECE Work Session on Statistical Dissemination and Communication
(12-14 September 2006, Washington D.C., United States of America)

Topic (i) Communicating effectively on the Web

RICH INTERNET APPLICATIONS

Supporting Paper

Submitted by the United States of America[1]

I. INTRODUCTION

1. While the Web revolutionized how information is accessed, it has been slow to move past a simple page-based model into truly interactive experiences. Many technologies have tried to make the Web experience more interactive—for example, JavaScript, DHTML, and ActiveX; however, inconsistent support across browsers has limited their success. In addition, many “hacks” have been developed to push browser-based technologies to their limits—often at the cost of standards-based development. A notable example is the use (or misuse) of tables in HTML to control the positioning of elements and to create sophisticated layouts with multiple columns. However, all that may soon change. Just as the Acrobat Reader has become a widely supported, cross-platform solution for delivering printed information, the Flash Player is emerging as the standard for delivering robust interactive experiences—“rich internet applications” (RIAs).[2]

2. Flash has traditionally been thought of as a designers’ tool, using “stages” and “timelines” in the development of sophisticated animations. With the MX release of Flash, applications delivered within the Flash Player could start exchanging data with servers; however, programming such applications was rather tedious. Then, in 2004 Macromedia (now Adobe) introduced the Flex framework, which greatly simplified application development. The Flex framework includes an XML-based language, MXML, and an implementation of ECMA Script called ActionScript. In addition, the framework also incorporates many standard Web technologies already in place, such as CSS for styling and GIF, JPG, PNG, and even SVG for graphics. While the traditional Flash tools will remain the standard for designing animations, innovations over the past two years have given developers new options for creating richer, more interactive applications.

3. Initially, the main drawback with the Flex framework was the need for a very expensive application server to deliver these new applications. With the release of Flex 2.0 in mid-2006, that will no longer be the case. While a server component will still be required for applications to implement advanced data services—such as messaging, data management services, and RPC services—it will no longer be needed for all applications.[3] Thus, making application delivery in the Flash Player a viable new alternative worth consideration.

---

[1] Prepared by Laurie Brown (Laurie.Brown@ssa.gov), U.S. Social Security Administration.

[2] While varying definitions of “rich internet applications” exist, this paper focuses on those that are delivered using the Flash Player.

[3] Adobe offers four products as part of its Flex family. A free SDK is available that includes the core components library, development languages, and a command-line compiler and debugger. For those needing a more robust editing environment, an Eclipse-based IDE, Flex Builder, is offered. Like many Web technologies, other editors are also available. For more advanced applications, Flex Charting offers extensible charting components, and Flex Data Services provides advanced data services, as discussed above.

II. BENEFITS OF RIAs

A. A New Model for Interaction

4. RIAs move beyond the page-based model associated with HTML browsers. Instead of a request/response paradigm with frequent page refreshes, RIAs deliver content seamlessly, providing an experience more like that of a desktop application. They can offer a better user experience in several respects:

- RIAs can execute logic on the client side; they do not need to call back to the server each time the user performs an action. HTML is developed as a series of files that get delivered individually to the users. While RIAs do consist of many pieces, it is a single, compiled application that is delivered to the user.
- RIAs can make asynchronous server calls, returning control of an application immediately to the user.
- RIAs can receive information broadcasts. For example, rather than making a call back to the server at a specified time interval to refresh a set of headlines or financial data, an RIA can simply register itself with a service and automatically be contacted when something changes.
- RIAs can progressively disclose information to the user and provide better feedback with state transitions.

5. Let’s compare how two simple applications—a glossary and a data query—could be implemented in HTML and as an RIA.

6. First consider a glossary application that allows a user to look up a word while reading a document. With HTML, this type of application could be implemented several ways:

- a link to a separate glossary,
- a simple form included on the document page, or
- frames.

With the first two options—a link or a form—the definition may be returned as a new page in the same window or may spawn a new window (either a new browser window or a popup box). In either case, it is difficult for the user to view the definition in the context of reading the document—either the document page is no longer displayed in the browser window or it may be obscured by the new window. While those problems can be avoided by the use of frames, frames take up a great deal of screen real estate—even when they are not in use.

7. With an RIA, a glossary tab could be situated unobtrusively off to the side of the document window—taking up very little space. When the user needed to use the glossary, the tab could be activated and both the glossary area and the document area dynamically resized without a page refresh. The user could then choose to leave the glossary sitting open as they continued to read the document or close it when it was no longer needed. This type of progressive disclosure means that the user is not distracted by information that may not be relevant to the initial task at hand and allows for a much more efficient allocation of screen real estate.

8. Now consider a simple data query application. With a traditional HTML model, users would likely fill out a form with their criteria—for example, calendar year data from 1990 to 2005, submit the form, and then receive a page back that lists the data concepts available that match their criteria. While this model lets users know what is available that matches their criteria, it fails to let them know what has been excluded as a result of their choices.

9. With an RIA, all the data concepts available could initially be listed and then progressively greyed out as the user incrementally narrows their criteria. With this approach, users are better able to understand the tradeoffs they are making with each new selection criteria they impose.

B. Extensibility

10. Both the MXML and ActionScript components of the Flex framework are extensible: Existing components may be customized, their functionality extended or overridden; existing components may be combined to create new functionality; or entirely new components may be built from scratch. Those familiar with Java will recognize many of the terms and techniques used in the technical discussions of extensibility.

C. Accessibility

11. Unlike HTML, MXML was designed with accessibility in mind from the start. HTML repurposed attributes that already existed in the specification for accessibility. Consider, for example, a graphic navigation button; the “alt” attribute of the given image tag may be used to specify a tool tip for a sighted user or an alternative text label to be read by a screen reader for a visually impaired user. However, the designer must choose between these two options—or, in some cases, the choice has been made for them by accessibility legislation. MXML, on the other hand, includes separate attributes for accessibility. Therefore, a graphic navigation button specified in MXML can have both a label and tool tip associated with it, both of which are visually presented to the sighted user and read to the visually impaired user.

III. EXAMPLE APPLICATIONS

12. A number of RIAs currently exist on the Web. While some are full-fledged working applications, many others have been developed as a proof-of-concept by developers starting to experiment with RIAs. (Note: The examples provided below are provided solely as technical demonstrations.)

- Audio Visualization Component – http://lab.benstucki.net/archives/visualizationexplorer/

  Displays sound data in a user-selected format. In addition, the user can customize several aspects of the presentation.

- Genworth Financial Retirement Income Gap Calculator – http://www.gefinancialassurance.net/calculators/incomeGap.mxml

  Notice how the graphics to the right immediately incorporate any changes made to the assumptions on the left. In addition, users can toggle between the graphics and a data table of the same information using the accordion panel control.

- Links to additional applications may be found at http://www.adobe.com/devnet/flex/?tab:samples=1 and http://www.adobe.com/devnet/flex/community_samples.html

IV. CONCLUSION

13. Although a relative newcomer on the Web technology scene, RIAs hold a great deal of potential for improving the user experience. Because RIAs draw on many standard technologies—XML, DOM, Web services, Java, HTTP, standard graphic formats, and CSS—the best practices and programming models already learned by developers can be employed. For example, the typical three-tier architecture of Internet applications still applies. Because there is extensive support for existing middleware solutions, in many cases, converting a current browser-based application to an RIA may be as simple as redeveloping the presentation tier. In summary, RIAs give developers the opportunity to greatly improve the user experience—moving away from the confines of the request/response model of the Web and into truly interactive experiences that better match users’ expectations and workflow.

V. RESOURCES

14. Additional information about RIAs and the Flex framework may be found at the following Web sites:

- Wikipedia - Rich Internet Application – http://en.wikipedia.org/wiki/Rich_Internet_Application
- Flex.org – http://www.flex.org/
- Adobe - Flex 2 – http://www.adobe.com/products/flex/
- W3C - Rich Web Clients – http://www.w3.org/2006/rwc/