Web UI tool comparison

Introduction

There are many opportunities surrounding Fedora to create user interface experiences. Many times it is preferred that these interfaces be deployed via a web browser. Some examples of potential interfaces to build using these tools are a replacement Fedora Administrative GUI, a Fedora Share application, object builder applications, and server monitoring applictaions. This comparison considers the capabilities and benefits of several tools for building rich internet applications. The goal of this comparison is to aid developers in determining the appropriate tool(s) to use when building UIs for Fedora. Since many of these tools will need to work directly against the existing Fedora REST and SOAP APIs, special attention will be given to each tool's ability to connect to these services.

Disclaimer

These write ups are generally completed after conducting research about the tool and experimenting with using it for a day or two. If you have implementation experience with a particular tool feel free to add notes or comments.

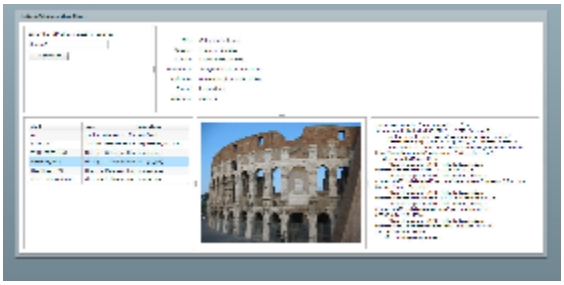
Adobe Flex

Overview

Flex is a framework for building rich internet applications which are run using Adobe's Flash player in the browser, and Adobe's AIR on the desktop

Sample application

Created a test application which allows a user to view an object's Dublin Core metadata, FOXML, and datastream list after entering an object's PID. The user can then click on any of the datastreams that are image files and view the image. This application required about a day to create (including technology ramp time) and the completed code (an XML file) is 77 lines, including comments and whitespace.



APIs

Rich selection of UI components and widgets. Component placement is organized using layout containers, which provide guidelines and reasonable default values that can be changed to produce a desired look.

REST requests can be made very easily in a single line. SOAP requests are also fairly simple to accomplish. The BlazeDS data service can be set up to allow calls directly to POJOs on the server, with transfers in serialized binary. The Flex Builder tool integrates with Eclipse and makes development and deployment convenient.

Documentation

Documentation seems very good. Answers to development questions are available through examples and API reference information. Complete reference API similar to javadoc is available. Significant documentation is available from Adobe, and there are several recently published books available on Amazon for developing with Flex.

Development Languages

A flavor of XML called MXML is used to lay out and define components while ActionScript is used to handle events and perform more complex behaviors. Didn't have to use ActionScript at all to create the test application, but it appears similar syntax-wise to JavaScript. Learning the tags, attributes, and possible values for MXML takes time but is straight-forward in most cases.

Deployment schemes

Compiles down to an swf (Flash) file which can be deployed by simply serving the file on a web server. Flex apps can also be deployed on the Adobe AIR platform, allowing them to run on the desktop.

Open source?

The language, the SDK, and the (MXML and ActionScript) compilers are open source. LifeCycle Data Services product BlazeDS, along with the AMF (Action Message Format - used to transfer data in compressed binary) spec, are also open source. The Flex Builder application (an Eclipse plugin) is not open source or free, however free licenses are available to staff of educational institutions. Flex applications can be developed and deployed without the Flex Builder.

Client requirements

Adobe Flash Player 9, the most current Flash player, is required.

Market penetration

Flex has a wide industry adoption, the flex showcase alone lists around 500 sites built using Flex. The open sourcing of most parts of Flex seems to have encouraged use. The SOURCE Fedora interface will be using Flex. The RepoMMan project used Flex.

General Impressions

I was impressed by how quickly and easily my test application came together. I appreciated being able to run the end product (a .swf file) directly in a browser (with the latest Flash plugin) without even needing to deploy it to a server. I came away with the impression that, once you get used to MXML and ActionScript you could do a lot in a short amount of time. It's nice that, in general, the application looks pretty decent even if you're not a graphic artist. There tends to be at lot of focus on getting, displaying, and working with data from a server, particularly XML data, which is very useful. A primary drawback is the requirement for the client to have the latest Flash player.

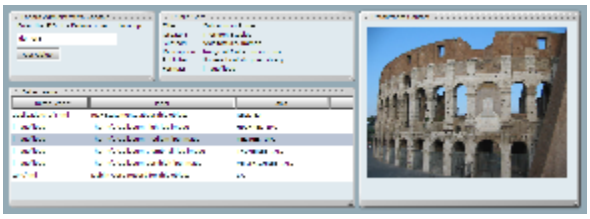
Open Laszlo

Overview

Open Laszlo is a framework for building rich internet applications which can be deployed either to Adobe's Flash player or as DHTML in the browser.

Sample application

Created a test application which allows a user to view an object's Dublin Core metadata and datastream list after entering an object's PID. The user can then click on any of the datastreams that are image files and view the image. This application required about two days to create (including technology ramp time) and the completed code (an XML file) is 124 lines, including comments and whitespace.



API

All the basic page widgets appear to be available, including trees tabs, sliders, etc. Layouts within a window are organized using views, which are similar to tags in html. This allows specific placement of components, but also requires attention to layout and design in order to produce even a decent looking interface.

The API is focused primarily on REST-style retrieval of XML, though SOAP and Java RPC are also supported. Scripting uses Javascript, which is included in tags within the XML.

Documentation

Documentation seems good. A listing of components with attributes and methods similar to javadoc is available which also includes editable examples. I did find that some components did not operate as presented in the examples (specifically gridcolumn and gridtext) and that some component attribute value options were not clear (specifially the layout attribute of the grid component.) Beyond the component reference there are several other manuals provided by Laszlo Systems. There is one recently published book on Amazon - Laszlo in Action.

Development Languages

Language is LZX (laszlo XML), which combines scripting syntax (from javascript) into XML syntax to allow for binding and callbacks, etc. Uses XPath for selection of content in XML. Scripting (in Javascript) is necessary for even the most basic interactions.

Deployment schemes

There are two deployment options, SOLO and proxied. SOLO deployment compiles the application to either an swf (for Flash deployment) or a javascript file (for DHTML deployment) which can be served by any web server. There are limitations in the data access options of SOLO deployments.

Proxied deployment serves your application files from within the OpenLaszlo server, which can either be a stand-alone server (which uses Tomcat internally) or a WAR running in any java servlet container. The content of the WAR file can be stripped down (by hand) to limit the unnecessary files, then repackaged prior to deployment.

Open source?

Completely open source.

Client requirements

For Flash support, requires Adobe Flash Player version 7 or higher. For DHTML support, enabled JavaScript is required.

Market penetration

There are a few notable sites built using Laszlo, including H&R Block, Behr, Pandora Radio, and bits of Walmart and Sears sites, however the Laszlo showcase only includes 10 sites.

General Impressions

Laszlo seems to focus more on design flexibility and less on simple/quick application developement. There is lots of flexibility to put whatever you'd like wherever you want it. Design feels like a richer version of html with css, which is very flexible, but also very time consuming, and would require graphic design skills to do it right.

Laszlo seems more focused on interface design than on data integration. The use of XPath to expose XML content is admirable because it is a standard, but the syntax is not always obvious or the most convenient.

In comparison with Flex, an obvious benefit is the ability to use DHTML rather than Flash. It took twice as long and significantly more code to create essentially the same application with Laszlo as with Flex. The documentation for Laszlo is not as comprehensive as what is available for Flex.

Google Web Toolkit (GWT)

Overview

The Google Web Toolkit is a development toolkit for creating AJAX-based web UIs which are written in Java and compiled to javascript for deployment.

Sample application

I spent about a day and a half attempting to create a sample application. In that time I was able to achieve a textbox which accepted a PID and queried for the object's xml, datastream list, and dublin core. The data all came back in xml, as long as the app was running on the same host/port as the Fedora server, but unfortunately either the xml parsing or widget display components did not cooperate well enough to actually display anything. What did display was very ugly, which is why there's no screenshot. This all could have been fixed if I'd had the time to spend.

API

UI code is written in Java in a style similar to Swing. Basic widgets and layout options are available with extension libraries providing a richer set.

Communication is limited to the server and port from which the UI is served, so allowing the server to perform any external communication to REST/SOAP /etc interfaces is the expected technique.

Debugging is performed using "hosted" mode where the UI is constructed on-the-fly in the vm and pushed to an embedded tomcat server, allowing you to debug directly in the java code. This is somewhat useful while in development, except that the UI output is not quite the same in hosted mode as it is when it is compiled to javascript. Debugging a deployed application would be very difficult as the javascript code is optimized to reduce transfer time.

Documentation

Documentation is good, at least for getting started. There is Javadoc for all classes provided by Google.

Development Languages

Development is in Java, which is familiar, but tends to require a lot of code to do simple things like implement action handlers and provide callbacks. Since Java provides many capabilities that javascript does not, only parts of the standard Java libraries are supported, meaning that care must be taken to not use classes/methods/types/etc which are not supported. The ability to use external libraries is limited.

Styling is done via a combination of external CSS and embedded component styles.

Deployment schemes

Java code is compiled to Javascript using XML config files. Along with the Javascript are HTML and CSS files (which can be edited directly). All of these together comprise the application which can be served from any standard web server.

Open source?

Completely open source.

Client requirements

Enabled JavaScript is required.

Market penetration

Based on the Who's Using GWT page, user groups, and other websites, it appears that while GWT has generated significant buzz, there's not much happening in the "real" application space. There are several recently published books about GWT.

General Impressions

Being able to work in Java in a manner resembling Swing is nice, but it also reminds me how long it takes to create even a simple UI. Learning how to use all of the GWT-specific classes (and learning which standard Java classes and libraries to not use) would take time similar to that of learning how to use components in an XML syntax. Not being able to access REST or SOAP interfaces directly without being on the same host/port is a major drawback.

Others

Echo2/3

Similar to GWT except that rather than compiling to Javascript, the Java code runs on the server, producing the UI and storing state on the server side. See: Comparing the Google Web Toolkit to Echo2 . Creating an application which runs only on the client is not possible with Echo.

Other opinions

- http://20agogo.wordpress.com/2007/09/28/ria-platforms-compared/
- http://java.sys-con.com/node/540504