Scalable, Rich Functionality Created 12/30/2010

Last modified: 1/4/2011

VERIZON BUSINESS

SCALABLE, RICH FUNCTIONALITY

User Experience Design | Shaun McNamara

Table of Contents

Introduction

This is an analysis of various Pan & Zoom capabilities, but it should also be the springboard document that leads us to a richer discussion concerning the scalability of the Solution Designer, focusing on Usability and Information Architecture as a whole.

I did some initial research on the Google Maps zoom capability and came up with a few answers, as well as comparisons. The packages that are reviewed here include:

- **ICEfaces**
- **JavaFX**
- **Flex**
- **DeepZoom** (formerly Seadragon)

After detailed and careful analysis, I find that Flex is the best option for what Verizon wants to do with Solution Designer, including the keywords that I continue to hear as the ultimate desire for the project – scalable, rich functionality. At first glance, JavaFX may seem like the silver bullet solution. It runs on the Java platform and so it would be a logical association with ICEfaces, which is itself an extension of the Java environment. That though is partly the problem. For everything that existing Java applications lacks, Oracle always touts some sort of association that can solve the problem (e.g., Spring), and since JavaFX is still very new, why go with something that requires many add-ons to solve problems, as well as the fact that it runs on what some would consider a bear of an environment?

The issue with waiting to implement the necessary technology until we see the necessary scalability is that it very likely requires throwing out most of what's already been developed for something new instead of scaling existing functionality to the next level. It would really be a restart, and whether we have a small environment now and don't need much of the rich functionality that an alternate package would give us, it ends up being a waste of money and resources in the long run. It really comes down to a User Experience issue, with Verizon being the User. Redeveloping after market goals are not attained (i.e., a scalable, rich environment that stands shoulder to should with the completion) is very costly and often damaging to a company, large or small – especially in our current economic times. Performing quality research and analysis of needs before development yields greater returns on investment.

ICEfaces

ICEfaces is an open source Ajax framework that enables Java EE application developers to create and deploy server-based rich Internet application (RIA) using the Java language.

ICEfaces leverages the entire standards-based Java EE ecosystem of tools and execution environments. Rich enterprise application features are developed in pure Java, and in a pure thin-client model. There are no Applets or proprietary browser plug-ins required. ICEfaces applications are JavaServer Faces (JSF) applications, so Java EE application development skills apply directly and Java developers are isolated from doing any JavaScript related development.

See http://www.icefaces.org/main/home/ for more detailed information and access to the ICEfaces community.

ICEfaces Component Suite

The Component Suite provides all of the building blocks for the application UI. It includes both the standard JSF components, and a wide array of advanced components that enable the developer to assemble sophisticated application interfaces efficiently. All ICEfaces component renders leverage the server-based, direct-to-DOM rendering mechanism provided in the framework, and use their partial submit attribute to facilitate automated event generation over the Ajax Bridge based on user interaction with the component's presentation. Optionally, ICEfaces components can be enabled with a variety of script.aculo.us effects such as drag and drop. Again, ICEfaces components carry attributes that enable various effects, so the developer is never exposed to low-level JavaScript programming to get dynamic features from a component.

Custom components include the aforementioned drag & drop solution as well as a component called Google Maps. The code, definitions, and a demonstration of each can be reviewed at:

http://component-showcase.icefaces.org/component-showcase/showcase.iface?rvn=2

Specific to the Verizon Solution Designer for EaaS at this time, the definition of the Google Maps custom component is:

“ICEfaces uses the Google Map API to provide map services. Utilize the ice:gMap component to geocode an address (the process of converting an address to longitude and latitude coordinates), search for a location, and toggle the visibility of the map controls or markers.”

Whether this translates to further customization of zoom and pan for Verizon’s purposes was never truly found, but based on developer input, I believe the Google API is not customizable to the point of being useful or effective in the Solution Designer environment.

An alternative possibility, if we want to stick with a Java-oriented environment is JavaFX, discussed in the next section.

JavaFX

Oracle touts JavaFX as the best way to create expressive, feature-rich content. The current release enables building applications for desktop, browser, and mobile devices.

Still in its infancy at release 1.3.1, JavaFX 2.0 is planned for release in the second half of 2011. Much like Microsoft's Silverlight, Oracle's JavaFX is designed to compete with Adobe Flash Player, Adobe AIR, and Flex. The proposed feature list is available at:

http://javafx.com/roadmap/

A complete overview of current and future development can be found at:

http://javafx.com/

JavaFX does make zoom effects available. The details are documented at:

http://download.oracle.com/javafx/1.3/howto/Zoom-In-Tutorial.html

Flex

Flex is a highly productive, free, open source framework for building expressive web applications that deploy consistently on all major browsers, desktops, and operating systems by leveraging the Adobe Flash Player and Adobe AIR runtimes. While Flex applications can be built using only the free Flex SDK, Adobe Flash Builder (formerly Adobe Flex Builder) software can accelerate development through features like intelligent coding, interactive step-through debugging, and visual design of the user interface layout. Flex applications can be developed using any standard IDE, for example Eclipse.

Traditional application programmers found it challenging to adapt to the animation metaphor that the Flash Platform was originally designed upon. Flex seeks to minimize this problem by providing a workflow and programming model that is familiar to these developers. MXML, an XML-based markup language, offers a way to build and lay out graphic user interfaces. Interactivity is achieved through the use of ActionScript, the core language of Flash Player that is based on the ECMAScript standard. The Flex SDK comes with a set of user interface components including buttons, list boxes, trees, data grids, several text controls, and various layout containers. Charts and graphs are available as an add-on. Other features like web services, drag and drop, modal dialogs, animation effects, application states, form validation, and other interactions round out the application framework.

In a multi-tiered model, Flex applications serve as the presentation tier. Unlike page-based HTML applications, Flex applications provide a stateful client where significant changes to the view don't require loading a new page. Similarly, Flex and Flash Player provide many useful ways to send and load data to and from server-side components without requiring the client to reload the view. Though this functionality offered advantages over HTML and JavaScript development in the past, the increased support for XMLHttpRequest in major browsers has made asynchronous data loading a common practice in HTML-based development as well.

Technologies that are commonly compared to Flex include Curl, OpenLaszlo, Ajax, XUL, JavaFX, and Windows Presentation Foundation technologies such as Silverlight.

My recommendation is to pursue Flex. You can view the Pan/Zoom component at:

http://www.adobe.com/devnet/flex/samples/fig_panzoom.html

And for a tour of Flex, see:

http://www.adobe.com/devnet-archive/flex/tourdeflex/web/#

Flex 4 is the latest version of Flex. Adobe Flash Builder and Flex 4 are discussed in the next section.

Adobe Flash Builder and Flex 4

Adobe released Flex 4.0 (code named Gumbo) on March 22nd 2010.[2] The Flex 4 development environment is called Adobe Flash Builder,[3] formerly known as Adobe Flex Builder.

Some themes that have been mentioned by Adobe and have been incorporated into Flex 4 are as follows:

- Design in Mind: The framework has been designed for continuous collaboration between designers and developers.
- Accelerated Development: Be able to take application development from conception to reality quickly.
- Horizontal Platform Improvements: Compiler performance, language enhancements, Bidirectional components, enhanced text (Flex 4 includes the new Text Layout Framework [4]).
- Full Support for Adobe Flash Player 10 and above.
- Broadening Horizons: Finding ways to make a framework lighter, supporting more deployment runtimes, runtime MXML.

- Simpler skinning than the previous versions.
- Integration with Adobe Flash Catalyst.
- Custom templates

Flash Builder is available in two versions: Standard and Premium; the premium edition adds the following features:

- Testing tools
- Memory and performance profilers
- An automated testing harness to connect to all the leading testing tools
- FlexUnit support
- command-line build capability
- The new Network Monitor

DeepZoom (formerly Seadragon)

Currently a component of Silverlight 2, Deep Zoom is an implementation of the open source technology, provided by Microsoft, for use in image viewing applications. It allows users to pan around and zoom in a large, high resolution image or a large collection of images. It reduces the time required for initial load by downloading only the region being viewed and/or only at the resolution it is displayed at. Subsequent regions are downloaded as the user pans to (or zooms into) them; animations are used to hide any jerkiness in the transition. The libraries are also available in other platforms including java and flash.

Image Pyramid

Deep Zoom, or for that matter any other similar technology such as Zoomorama, Zoomify, Google Maps etc. uses something called an image pyramid. It allows the rendering engine to pick and choose only the data that is required for a particular view. Each level is significantly smaller than the next, and is loaded as soon as it becomes available. The smooth experience is created by progressively loading each new resolution level and seamlessly blending between them. The following illustration below shows what the image pyramid looks like conceptually.

See: http://bsix12.com/seadragon/ for more information.

And here is some comparison information, which includes a link to one of the premier Deep Zoom Web sites, Hard Rock Memorabilia. It's definitely worth a look. As with many blogs, it's extremely biased, but still presents some good analytics as well as potential legal ramifications for using the Google Maps API.

http://weblogs.asp.net/jgalloway/archive/2008/03/21/why-silverlight-2-deep-zoom-really-is-something-new.aspx