Web 2.0 and Rich Client Architectures: AJAX Integration Tools for Programmers

INTRODUCTION

With years of experience in delivery of Rich Client/Web 2.0 applications, ProKarma is well-versed in the programming languages and tools that are currently being used to deliver a rich client-side experience. While the toolkit and trends are always changing, it's worth examining the different options available for developing a rich interactive client-side system for the user that integrates with the server-side effectively. Typically, characteristics of Web 2.0 applications include the following:

- Delivery through a Web browser using approaches that support Web standards.
- User-friendly, often in a similar way to typical desktop applications.
- Use of rich application interfaces, often including pre-built components and widgets.
- Allow the user to interact and contribute content.

These characteristics are quite different from the qualities of Web1.0 applications. Web1.0 applications are static, used for delivering information, and don't allow user interaction. The dramatic shift from Web1.0 applications to Web2.0 comes as no surprise when considering the way that interaction with the Web has evolved.

AJAX

The client-side or web browser technologies used in Web 2.0 development include Ajax and JavaScript frameworks, such as Dojo Toolkit and Prototype JavaScript Framework. Ajax is an acronym for asynchrous Java scripts. Ajax programming uses JavaScript to upload and download new data from the web server without undergoing a full page reload. This allows the user to interact with the page without requiring the entire page to change, making things like drag and drop, comments, and tagging possible. JavaScript is a simple, versatile language used to extend functionality in websites. Though their names are similar, JavaScript and Java are very different languages. JavaScript is a language used within HTML pages, while Java is a programming language. Typically, Java is used on the server-side and JavaScript is on the client-side.

AJAX Toolkit and Framework Solutions

- Client-side JavaScript libraries: Dojo ToolKit or Prototype
- Remote Procedure Call library: Direct Web Remoting (DWR) enables Java on server and JavaScript on browser to communicate.
- Lightweight remote procedure call protocol: JSON-RPC is a lightweight remote procedure call protocol similar to XML-RPC. JSON-RPC-Java is the language-specific implementation for Java.
- AJAX-enabled JavaServer Faces components: AjICEfaces
- Wrapping tools: jMaki wraps AJAX widgets with Java coding.
- Java to JavaScript/HTML translator: GWT
- Model view controller (MVC) server-side scripting: Phobos
- Web application frameworks with AJAX extensions: Shale or Echo2

ProKarma

Client-Side JavaScript Libraries

Client-side JavaScript libraries are server-side technology agnostic, meaning the libraries are interoperable among various systems and can be used in combination in a single application. In other words, one can use widgets and JavaScript utilities from multiple sources. However, they are somewhat difficult to interface with server-side technologies such as JavaEE, .Net, PHP, and Ruby on Rails. Additional tools, such DWR, can be used to simplify communications between the client-side and the backend Java applications.

Technical Benefits of Client-Side JavaScript Libraries

- Enables remote asynchrous communication and hides low-level XML Http request operation
- Deals with browser incompatibilities: no need to clutter your code with if/else's
- Handles graceful degradation, using IFrame if the browser does not support XML Http request
- Provides page navigation hooks over Ajax (back and forward buttons) and bookmarking
- Provides advanced UI features, including animation, drag and drop, and fade in-fade out.

There are a number of popular client-side JavaScript libraries that ProKarma's development team use in their development processes and recommend for Web2.0 development.

DOJO TOOLKIT

- Widely used and comprehensive
- Major industry support (Sun, IBM)

PROTOTYPE

- Used by other toolkit libraries

SCRIPT.ACULO.US

- Built on Prototype framework
- Nice set of visual effects and controls

RICO

- Built on Prototype framework
- Rich AJAX components and effects

DHTML GOODIES

- Various DHTML and AJAX scripts

Dojo Toolkit versus Prototype

A powerful event toolkit, Dojo Toolkit makes every function call an event, which is the first use of Aspect-Oriented Programming. Also, the widget framework is well-organized and modular, which is useful for huge JavaScript projects. Prototype has good shortcuts for functions and many other good features, such as element Helper functions and Ruby-like features like Enumerable, Array, and Hash. Dojo is a fairly good approximation of the above, but adds some other heavy-duty features. Generally, Prototype and Dojo deserve high marks; however, Dojo may be somewhat over-engineered.

Remoting

DWR is a Java library that enables Java on the server and JavaScript in a browser to interact with each other as simply as possible. DWR simplifies the server-side code and increases security. JSON-RPC is a lightweight remote procedure call protocol similar to XML-RPC. Multiple input parameters can be called and returned at one time as objects or arrays. JSON-RPC-Java is the language-specific implementation for Java.

Wrapper Technology

Several existing AJAX widget toolkits such as Dojo, Scrip.taculo.us, Yahoo UI Widgets, and DHTML Goodies can be leveraged to create wrapped Java components. Wrapped Java components address the need for a common programming model when using widgets from multiple AJAX toolkits and frameworks, and aids JavaEE developers who are less familiar with JavaScript coding. JavaScript widgets can be wrapped with JavaEE syntax and JSP custom tags to ensure compatibility with the Java platform. jMaki is a popular wrapper framework for JavaScript and the Java platform. The name, jMaki, was derived from “j,” for JavaScript, and “Maki,” a Japanese word for wrap. It allows developers to take widgets from many popular AJAX toolkits and frameworks and wrap them into a JSP or JSF tag for integration with JavaEE.

JAVASERVER FACES

JavaServer Faces (JSF) is a standard JavaEE Web2.0 development framework. It is a component-based framework and provides a simple and cohesive programming model. JSF architecture itself leads to Ajax development. It defines three things: component architecture, a standard set of UI widgets, and an application infrastructure. JSF component architecture defines a common way to build UI widgets. The framework consists of components, events, validators and converters, navigation, and back-end-date integration.

The main advantages of JSF include:

- MVC for web applications
- Clean separation of roles
- Ease of use
- Extendable component and rendering architecture
- Support for client device independence
- Standardization
- Huge vendor and industry support

Using Struts as a framework with JavaServer Pages (JSP) and Servlets is a popular method, but is not considered a standard. JSP/Servlet does not include a built-in UI component model. Using Struts as a framework is also problematic because it doesn't include built-in UI component models, including event models, state management, or built-in support of multiple renderers.

Combining JSF and AJAX

Ajax development with JSF does not require JavaScript. The JSF architecture makes it easy to add Ajax support because components and renderers are separate and phase listeners can be invoked during request processing. JSF architecture enables transparent Ajax support because it is the same programming model with or without AJAX. There are many ways to wrap AJAX functionality using JSF. Using the phase listener is a common approach that handles many common AJAX-enabled JSF component creation cases.

AJAX-Enabled JSF Components

Ajax-enabled JSF components hide all the complexity of AJAX programming. The UI developer does not need to know JavaScript. The burden is shifted to component developers and components are reusable. They can leverage the drag-and-drop Web application development model of JSF through an Integrated Development Environment (IDE). You can drag and drop AJAX-enabled JSF components within the IDE to build AJAX applications.

Three types of AJAX integration with JSF are possible:

1. Add Ajax support to JSF view: AjaxAnywhere and DynaFaces
2. Use JSF Components with integrated AJAX support: ICEsoft's ICEfaces and RichFaces Visual Component Platform.
3. Wrap existing Ajax widgets: jMaki and DojoFaces

DECIDING YOUR INTEGRATION STRATEGY

- Use AJAX-enabled JSF components whenever possible, using a JSF-enabled IDE. If you are not ready to commit to JSF component solutions, use jMaki. If you want to have total control on the client-side JavaScript coding, use Dojo Toolkit.
- If the applications need AJAX push, then RichFaces or ICEfaces may be a better choice.
- Use jMaki if there is a need for a lot of Web2.0 components, as it can wrap all the current Ajax JavaScripts available in the market with a uniform JSF interface.
- Use GWT if you already have Swing apps that you want to expose as AJAX Web apps, or if you do not want to deal with JavaScript coding.
- On the business logic side, if there is already JavaEE business logic that needs to be exposed as RMI calls on the client-side with AJAX behavior, use DWR.
- If you are already using a particular Web application framework for building the majority of your Web applications and the framework has AJAX extensions, use those.

CONCLUSION

Many different component suites and solutions currently exist and are constantly being updated and improved. JSF 1.2 makes Ajax integration easier for component developers. JSF 2.0 provides a single mechanism for Ajax integration for component vendors. At ProKarma, most Web2.0 developers make use of RichFaces. However, developers have used Dojo components in combination with Apache MyFaces and sandbox components, or by writing custom components using a phase listener approach. Various scenarios require 'div layer popups' and for this Jenia4Faces has been used. MyFaces components also have very good accordion components that encapsulate Rico Ajax components. RichFaces has very advanced skinnability features for UI styles. From a best practices perspective, ProKarma recommends against using JavaScript directly in a JSF page. Instead, use a more object-oriented approach by using Ajax-JSF components or jMaki to encapsulate the JavaScript as a phase listener approach.

ProKarma

Web 2.0 and Rich Client Architectures:
AJAX Integration Tools for Programmers

REFERENCES

Official JSF site
http://java.sun.com/javaee/javaserverfaces

Dojo Toolkit
http://dojotoolkit.org/

JSF Central product Directory
http://www.jsfcentral.com/products

AJAX JSF Component Feature Matrix
http://www.jsfmatrix.net/

Ajax Component Suite Comparison
http://www.thserverside.com/tt/articles/article.tss?l=JSFComparison

GWT
http://code.google.com/webtoolkit/gettingstarted.html

Ajax Frame works and Toolkits
By - Sang Shin
http://www.javapassion.com

Using Phase Listener Approach for Java Server Faces Technology with AJAX
https://blueprints.dev.java.net/bpcatalog/ee5/ajax/phaselistener.html

Emerging Technologies for the enterprise: AJAX development with JSFBy - K, Mann
http://www.chariotsolutions.com/

Ajax tutorial
http://java.sun.com/javaee/javaserverfaces/ajax/tutorial.jsp

Ajax4JSFDemos
https://ajax4jsf.dev.java.net/nonav/ajax/ajax-jsf/download.html#examples

Jenia Faces
http://sourceforge.net/projects/jenia4faces

Rich Faces
http://www.exadel.com/web/portal/products/VisualComponentPlatformReferences

ABOUT PROKARMA