AJAX and jMaki for Web 2.0 Development using Java

Inyoung Cho
Java Technology Evangelist
Sun Microsystems, Inc.
Agenda

• AJAX Basics
  > What is AJAX?
  > AJAX Interaction: Using “AutoComplete” Sample Application
  > JSON
  > AJAX Guidelines

• Jmaki

• AJAX Toolkits and Frameworks

• Summary and Resources
AJAX Basics: What is AJAX?
What is AJAX?

- AJAX is an acronym for Asynchronous Javascript And XML
  - AJAX uses JavaScript combined with xml to grab information from a server without refreshing the page
  - nothing new, the main requirement is the web browser has the support for XMLHttpRequest object
  - The term AJAX was coined Jesse James Garrett in February 2005

- Asynchronous communication replaces "synchronous request/response model."
  - A user can continue to use the application while the client program requests information from the server in the background
  - Separation of displaying from data fetching
Why AJAX?

- "Partial screen update" replaces the "click, wait, and refresh" user interaction model
  > Only user interface elements that contain new information are updated (fast response)
  > The rest of the user interface remains displayed without interruption (no loss of operational context)
- Intuitive and natural user interaction
  > No clicking required
  > Mouse movement is a sufficient event trigger
- Data-driven (as opposed to page-driven)
  > UI is handled in the client while the server provides data
User Interface: Traditional Web vs. AJAX

Interrupted user operation while the data is being fetched

Uninterrupted user operation while data is being fetched
Components: Traditional Web vs. AJAX
AJAX Basics:
AJAX Interaction:
Using “AutoComplete”
Sample Application
AutoComplete Using a Servlet

How can you provide a better means of navigating a large set of data based on a search query?
Steps of An AJAX Interaction

1. A client event occurs.
2. An XMLHttpRequest object is created and configured.
3. The XMLHttpRequest object makes a call.
4. The request is processed by the AutocompleteServlet.
5. The AutocompleteServlet returns an XML document containing the result.
6. The XMLHttpRequest object calls the callback() function and processes the result.
7. The HTML DOM is updated.
AutoComplete Update

Auto-Completion using Asynchronous JavaScript and XML (AJAX)

This example shows how you can do real time auto-completion using AJAX interactions.

In the form below enter a name. Possible names that will be completed are displayed beneath the form. Click on one of the selections to see the employee details. Try typing "Greg", "Murray", "Jones", or "Cindy".

Employee Name: [Field]

George Murphy
Greg Murphy
Cindy Murphy
George Murray
Greg Murray

Lookup Employee
AJAX Application Demo

• Use various real life apps
AJAX Basics: JSON (Java Script Object Notation)
AJAX vs Ajax

• Not always XMLHttpRequest
  > Comet, JSONP, IFRAME, Flash/Applet

• Not always XML
  > JSON, JavaScript, HTML, Plain text

• Not only JavaScript
  - JSON, JavaScript, HTML, Plain text
  - SVG / VML / dojo.gfx
  - Behaviour

• Fancy transitions / Effects / DnD
• Mashups
JSON (Java Script Object Notation)

object = { 'key' : value }

array = [ value1 , value2]

value =
    string,
    number,
    object,
    array,
    true,
    false,
    null
Example: XML and JSON

Category XML

```xml
<category id="0" name="Vegetables">
  <products>
    <product><name>Onion</name><price>.75</price></product>
    <product><name>Carrot</name><price>.50</price></product>
  </products>
</category>
```

Category JSON

```json
{"id": "0", "name":"Vegetables",
 "products": [
   {"name": "Onion", "price": .75},
   {"name": "Eggplant", "price":1.50}
 ]
}
```
var categories = [];

function processResults(responseXML) {
    var categoryElements = responseXML.getElementsByTagName("category");
    for (var l=0; l < categoryElements.length; l++) {
        var categoryElement = categoryElements[l];
        var catId = categoryElement.getAttribute("id");
        var catName = categoryElement.getAttribute("name");
        var cat = {id: catId, name: catName, products : []};
        var ps = categoryElement.getElementsByTagName("product");
        for (var pl=0; pl < ps.length; pl++) {
            var name = ps[pl].getElementsByTagName("name")[0].firstChild.nodeValue;
            var price = ps[pl].getElementsByTagName("price")[0].firstChild.nodeValue;
            cat.products.push({productName: name, productPrice: price});
        }
        categories.push(category);
    }
    var veggies = categories[0].products[0];
}
Parsing JSON Responses

```javascript
var jsonText = // get the JSON text listed above
var categories  = eval("("+jsonText")")
var veggies = categories[0].products[0];
```
AJAX Basics: AJAX Guidelines
AJAX Guidelines

- Usability
- JavaScript Libraries
- I18n
- State Management
- HTTP methods
- Return content-types
- Tools and Debuggers
Usability

- Back/Forward button meaningless
- Refresh button can kill your app
  > Save state in <body onload> method
- Bookmarking/URL sharing not working
- Printing dynamically rendered pages can be problematic
- JavaScript framework Dojo toolkit (http://dojotoolkit.org)
  > provides API's history manipulation and navigation control
  > provides client-side for bookmarking and URL manipulation
- Requires thinking about interface and interaction
  > Usability is the key
    > Do not break what the user is focusing on
    > Make sure to give the user feedback
  > Do not over-use it

Recommendation: Consider the meaning of each and weigh the benefits when designing your application.
JavaScript Libraries

- There are differences in javascript implementations
- Serializing complex data and mapping it to javascript isn't a trivial task
- Directly consuming xml in Javascript can be painful due to browser differences in technologies like xslt
- Dojo: a key focus on user experience
- Prototype: focuses more on AJAX interactions
  > JavaScript objects and utility methods
- DWR: both client-side and server-side framework
  > do RPC calls from client-side JavaScript to Java objects in a web container server side

Recommendation: Adopt a library and don't try to re-invent the wheel.
Internationalization (I18n)

• Page Content Type
  <meta http-equiv="Content-Type"
       content="text/html; charset=UTF-8">

• Use JavaScript `encodeURI()` when building URLs or
  sending localizable content.

• Call HttpServletRequest.setCharacterEncoding("UTF-8")
  before retrieving any parameters from Java EE

Recommendation: Use UTF-8 since it supports the
widest number of languages and browsers.
State Management

• On Client
  > In memory in JavaScript Objects
  > In hidden form variables
  > In cookies
• On Server
  > HttpSession
  > Persisted
• How do you handle state + sessions?
  > Can all be on the client so you can have truly stateless servers?
  > What happens if the server or network dies?

Recommendation: Consider keeping non-secure state related to the rendering of the client on the client. Keep secure persistent state on the server.
HTTP Methods

- **GET**
  > When the data will not change

- **POST**
  > When operation has “side-effects” and changes the state on the server.

Recommendation: Follow the HTTP idempotency rules.
Response Content Type

- XML
- Text
  - Post processing on client
  - Inject directly into the page
- JavaScript
  - Evaluated in JavaScript using eval()
  - JavaScript object representations of data (JSON)

Recommendation: Use XML for structured portable data. Use plain text for when injecting content into the HTML. Use JavaScript to return object representations data.
Development Tools for NetBeans IDE

- Building AJAX Applications over NetBeans is not that much different from building regular Web applications.
- NetBeans JavaScript editor plug-in
  > http://www.liguorien.com/jseditor/
Debuggers

- Mozilla FireBug debugger (add-on)
  > This is the most comprehensive and most useful JavaScript debugger
  > This tool does things all other tools do and more
- Mozilla JavaScript console
- Mozilla DOM inspector (comes with Firefox package)
- Mozilla Venkman JavaScript debugger (add-on)
- Mozilla LiveHTTPHeaders HTTP monitor (similar to NetBeans HTTP monitor)
- Microsoft Script Debugger (IE specific)
jMaki Toolkit
Motivations for jMaki

• You want to leverage widgets from existing and future AJAX toolkits and frameworks
  > Dojo, Scriptaculus, Yahoo UI Widgets and DHTML Goodies

• Today, there are multiple AJAX frameworks with their own widgets and with different syntax
  > There is a need for a common programming model to these various widgets

• Too much of JavaScript code required for Java developers
  > There is a need for Java Language view of JavaScript-based widgets
Why jMaki?

• Project jMaki makes it easier to use popular AJAX frameworks within the Java EE Platform

• It provides a base set of wrappers around some of the widgets from the more popular frameworks (such as Dojo, Prototype and Yahoo Widgets)

• Project jMaki is easily extensible, allowing developers to use the latest and most useful widgets as they appear
Why jMaki? (cont.)

- Enables Java developers to use JavaScript in their Java based applications as either a JSP tag library or a JavaServer Faces component
- Uses the best parts of Java and the best parts of JavaScript to deliver a rich AJAX style widgets
- Promotes a program methodology that cleanly separates behavior (JavaScript), from Styles (CSS), from template HTML
Usage Example of jMaki Widget (List) in a JSP page

```html
<%@ taglib prefix="a" uri="http://java.sun.com/jmaki" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>JSP Page</h1>
    <a:ajax name="list"  service="listService.jsp" />
  </body>
</html>
```
HTML that is Rendered by jMaki

```html
<html>
...
<body>
  <script type="text/javascript" src="http://localhost:8080/jMakiList/jmaki.js"></script>
  <script type="text/javascript">jmaki.webRoot='http://localhost:8080/jMakiList';</script>

  <link rel="stylesheet" type="text/css" href="http://localhost:8080/jMakiList/resources/list/component.css"></link>

  <div id="list0" class="listContainer">
    <form onsubmit="jmaki.attributes.get('list0').submitData(); return false;">
      <input id="list0_entryField" type="text" size="20" value="Enter new Value">
      <input type="button" onclick="jmaki.attributes.get('list0').submitData(); return false;" value="Add to List"></form>
    <div id="list0_list" class="listDiv"></div>
  </div>

  <script type="text/javascript">
    jmaki.addWidget({service:'listService.jsp',script:'http://localhost:8080/jMakiList/resources/list/component.js',uuid:'list0', name:'list'});
  </script>

</body>
</html>
```
How Does jMaki Know Which HTML Template to use for Rendering?

• jMaki (actually jMaki tag handler) takes the value from the name attribute - list in the example below
  > <a:ajax name="list" service="listService.jsp" />

• jMaki then finds the widget named as list

• list widget has the following 3 files associated with it - these files are parameterized
  > HTML template: list/component.html
  > CSS template: list/component.css
  > JavaScript code: list/component.js

• When rendered, the parameters are set with values
What Makes Up a jMaki Widget?

- HTML template
  - Defines the page layout
- JavaScript file
  - Defines behavior
- CSS file
  - Defines style
**jMaki: Pros and Cons**

**Pros**
- Provides unified programming model for using widgets over various AJAX toolkits and frameworks
- Allows Java developers to use familiar Java EE programming model (JSP or Faces tags) for using JavaScript widgets
- There is already a NetBeans Plug-in

**Cons**
- Event model is still not fully baked yet

**When to use**
- When you want to use widgets from different sources yet want to use uniform programming model
jMaki Demo

- Create jMaki apps in Netbeans
AJAX Toolkits and Frameworks
Types of AJAX Toolkit and Framework Solutions of Today

- AJAX-enabled JSF components
- Clients-side JavaScript Libraries
- Wrapper:jMaki
- Java to JavaScript/HTML translator:GWT
- Remoting via proxy
AJAX-enabled JSF Components

• AJAX-enabled JavaServer Faces components hides all the complexity of AJAX programming
  > Page author does not need to know JavaScript
  > The burden is shifted to component developers

• Leverages drag-and-drop Web application development model of Faces through an IDE
  > You can drag and drop AJAX-enabled Faces components within Sun Java Studio Creator 2 (and other Faces-aware IDE's) to build AJAX applications

• JavaServer Faces components are reusable
  > More AJAX-enabled Faces components are being built by the community
Implementations

- Blueprint AJAX-enabled JSF components (open-source)
  > [https://bpcatalog.dev.java.net/ajax/jsf-ajax/](https://bpcatalog.dev.java.net/ajax/jsf-ajax/)

- ajax4jsf (open-source)
  > Can add AJAX capability to existing applications
  > [https://ajax4jsf.dev.java.net/](https://ajax4jsf.dev.java.net/)

- ICEfaces (ICESoft) - commercial
Client Side JavaScript Libraries

- HTML Pages, JavaScript Event Handlers
- UI Widgets & Components
- Remoting Abstraction Layer
- XMLHttpRequest
- iFrame
- JavaScript Utilities
Architectural Layers (Client-side)

- **Remoting abstraction layer**
  > Hides handling of XMLHttpRequest and IFrame

- **Widgets and components**
  > Provides ready-to-use UI widgets such as calendar, button, etc

- **JavaScript event handlers**
  > Provides client-side logic
Characteristics of Client Side JavaScript Libraries

- Server side technology agnostic
  > The server side technology can be Java EE, .Net, PHP, Ruby on Rails, etc.

- Should be accessible during runtime – either locally or through a URL

- You can use various client-side libraries in a single application
  > Use widgets from multiple libraries
Client-side JavaScript Libraries

- **DOJO Toolkit (open-source)**
  > Key focus on user experience
  > Provides APIs for history manipulation and navigation control
  > Provides client-side for bookmarking and URL manipulation
  > Most prominent and comprehensive
  > Gaining a leadership in this space
  > Major industry support (Sun, IBM)
Other client-side JavaScript Libraries

- **Prototype**
  > Focuses more on AJAX interactions
  > JavaScript objects and utility methods
  > Used by other toolkit libraries
  > [http://prototype.conio.net/](http://prototype.conio.net/)

- **Script.aculo.us**
  > Built on Prototype
  > Nice set of visual effects and controls
  > [http://script.aculo.us/](http://script.aculo.us/)

- **Rico**
  > Built on Prototype
  > Rich AJAX components and effects

- **DHTML Goodies**
  > Various DHTML and AJAX scripts
What is Google Web Toolkit?

- Open source Java software development framework that makes writing AJAX applications easy
  - Java-to-JavaScript compiler and a special web browser that helps you debug your GWT applications
- You can develop and debug AJAX applications in the Java language using the Java development tools of your choice
- When you deploy your application to production, the compiler translates your Java application to browser-compliant JavaScript and HTML
GWT User Interface Classes

• Similar to those in existing UI frameworks such as Swing and SWT except that the widgets are rendered using dynamically-created HTML rather than pixel-oriented graphics

• While it is possible to manipulate the browser's DOM directly using the DOM interface, it is far easier to use classes from the Widget hierarchy

• Using widgets makes it much easier to quickly build interfaces that will work correctly on all browsers
Google Web Toolkit (Beta)

Widgets Gallery

The following are widgets and panels available in the GWT user-interface library.

- **Button**
  - Normal Button
  - Disabled Button

- **RadioButton**
  - Choice 1
  - Choice 2 (Disabled)

- **CheckBox**
  - Normal Check
  - Disabled Check

- **TextBox**
  - text box

- **PasswordTextBox**
  - ****

- **TextArea**
  - This is a big text area...

- **Hyperlink**
  - Info
  - Buttons
  - Menus
  - Images
  - Layouts

- **ListBox**
  - List0
  - List1
  - List2
  - List3
  - List4

- **MenuBar**
  - Style
  - Font
  - Size
  - More...
  - Code
  - Breakthrough

- **Tree**
  - foo@example.com
    - Inbox
    - Drafts
    - Templates
Remoting via Proxy (Client & Server)

HTML Pages, JavaScript Event Handlers

Proxy

Remote Abstraction Layer

XMLHttpRequest

iFrame

Java Classes

Skeleton

Framework runtime

HTTP Get/Post
Characteristics of “Remoting via Proxy” Framework

- Similar to general RPC communication schemes
- Allows RMI like syntax in the client side JavaScript code
- Framework generates client stub (Proxy), which is a JavaScript code
- Framework provides server side runtime
- Client stub (Proxy) handles marshaling of parameters and return value
Remoting via Proxy Implementations

- **Direct Web Remoting (DWR)**
  > Designed specifically for Java application at the back end
  > Do RPC calls from client-side JavaScript to Java objects in a web container server side
  > [http://getahead.ltd.uk/dwr](http://getahead.ltd.uk/dwr)

- **JSON-RPC**
  > Lightweight remote procedure call protocol similar to XML-RPC
  > There are language-specific implementations
  > JSON-RPC-Java
Summary & Resources

So... What Should I Use?
Summary

• AJAX helps make applications more interactive
• J2EE technology is a great platform for AJAX applications
• AJAX does not come for free
• Start small and don’t overdo it
• Choose the right Toolkits and Frameworks for your application
So What Should I Use? Assuming You are using Java EE...

• On the UI side
  > Use AJAX-enabled JavaServer Faces components whenever possible using an Faces-enabled IDE such as Sun Java Studio Creator 2
  > If you are not ready to commit yourself to Faces component solutions yet, use jMaki
  > If you want to have total control on the client side JavaScript coding, use Dojo toolkit
  > If you already have Swing apps that you want to expose as AJAX-fied Web apps or if you do not want to deal with JavaScript coding, use GWT
So What Should I Use? Assuming You are using Java EE...

• On the business logic side
  > If you already have Java EE business logic that you want to be exposed as RMI calls on the client with AJAX behavior, use DWR
  > If you are already using a particular Web application framework for building majority of your web application and the framework has AJAX extension, use it
For More Information

• The BluePrints Solutions catalog on AJAX:
  > https://bpcatalog.dev.java.net/nonav/solutions.html

• AJAX Q & A
  > https://blueprints.dev.java.net/ajax-faq.html

• Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform, Enterprise Edition

• AJAX Frameworks

• AJAX Library and Frameworks Comparison
  > http://wiki.osafoundation.org/bin/view/Projects/AjaxLibraries

• AJAX Developer Resource Center
  > http://developers.sun.com/ajax/

• JavaScript Developer Site
  > java.sun.com/javascript
Cross Browser Resources

• Cross Browser Discussion
  > http://www.quirksmode.org/index.html

• Cross Browser quirk

• JavaScript libraries such as Dojo that makes these differences less painful
  > http://dojotoolkit.org/
AJAX and jMaki for Web 2.0 Development using Java

Inyoung Cho
Java Technology Evangelist
Sun Microsystems, Inc.