Web-Tier Programming with Javascript and AJAX

Abhishek Mahanty
Architecture Solutions Team
Sun Microsystems

UNLOCK
OPPORTUNITY
What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference

Agenda

- Web 2.0
- AJAX
  - Overview
  - Demo: AutoComplete
- Web 2.0 & AJAX Toolkits/Frameworks
  - DoJo
  - jMaki
  - DWR

Web 2.0 Phenomenon

- Web as a Platform
  - User's computing activities occur on the web
  - Email, chatting, photo sharing, etc.
  - More and more apps are moving to the web
- Collective intelligence
  - Folksonomy: Collaborative categorization
- Data is key and should be shared
  - Mashups: Seamlessly combines content from more than one source into an integrated experience
- Rich user experience
  - AJAX

Web 1.0

- Personal Websites
- Email/News Groups
- Popups
- Web Directories
- Web Classifieds
- Terraserver
- mp3

Web 1.5

- Wikis
- Discussion Forums
- Popunders
- Yahoo
- Cragislist
- MapQuest
- Napster

Web 2.0

- Blogging - Roller
- RSS / Syndication
- Google Ad
- Delici.io.us
- HousingMaps.com
- Google Maps
- iTunes

Conventional Web Application issues

- “Click, wait, and refresh” user interaction
  - No instant feedback's to user activities
  - Loss of operational context
- Page refreshes from the server needed for all events, data submissions, and navigation
  - Excessive server load and bandwidth consumption
- Synchronous “request/response” communication model
  - The user has to wait for the response
- Page-driven
  - Workflow is defined in the server

Rich User Experience

- Take a look at a typical desktop application (Spreadsheet, email app, etc.)
- The program responses intuitively and quickly
- The program gives a user meaningful feedback's instantly
  - The cursor changes shape
  - Icons light up as mouse hovers them
  - Selected text changes color
- Things happen naturally
  - No need to click a button or a link

Rich Internet Application (RIA) Technologies

- Macromedia Flash
- Applet
- Java WebStart
- DHTML (JavaScript + DOM + CSS)
- AJAX

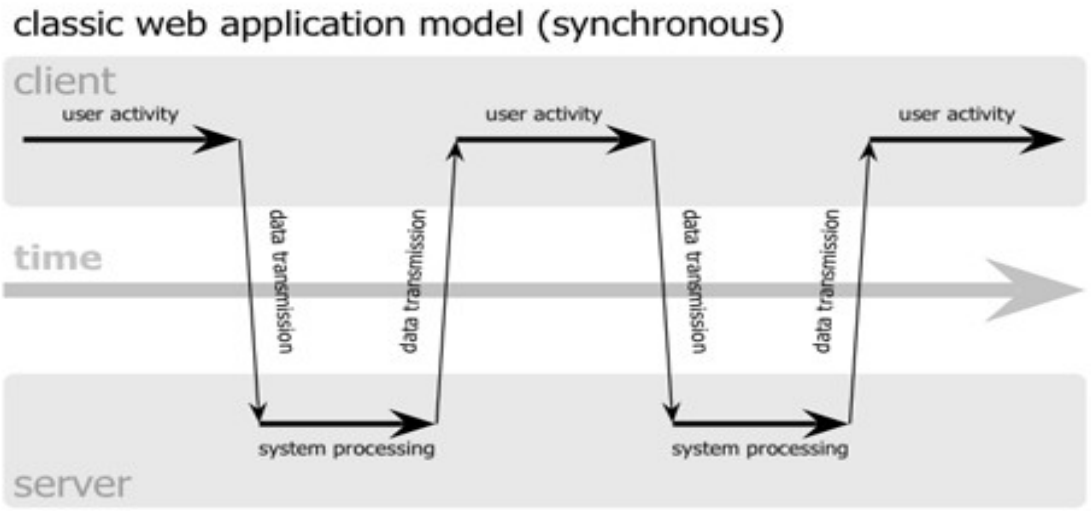
Agenda

- Web 2.0
- AJAX
  - Overview
  - Demo: AutoComplete
- Web 2.0 & AJAX Toolkits/Frameworks
  - DoJo
  - DWR
  - Jmaki

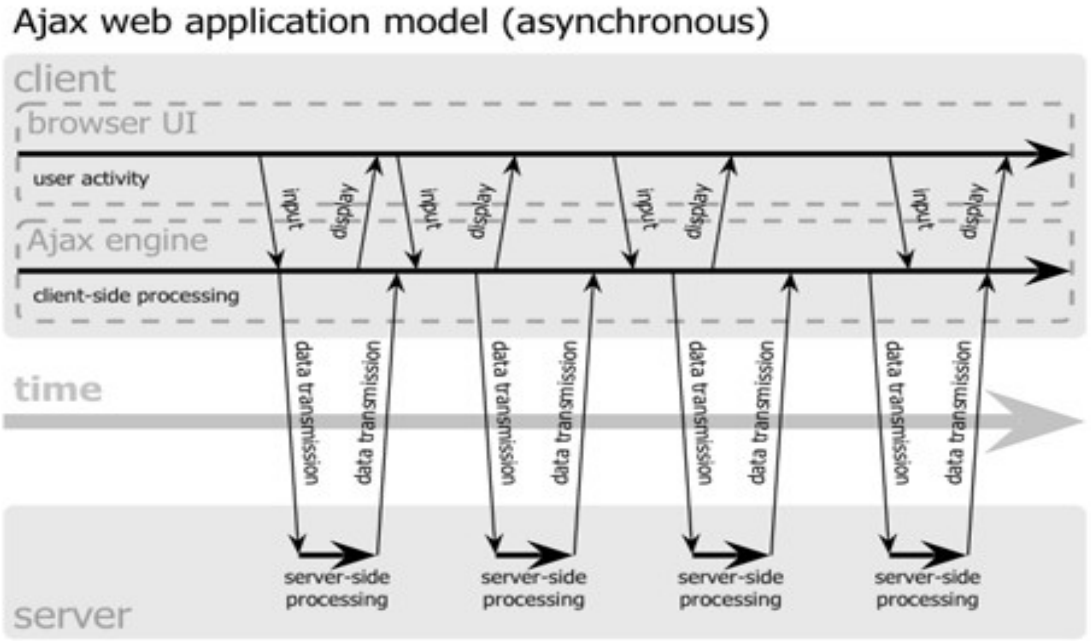
What is AJAX?

- 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 by 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

Traditional Web vs AJAX: User Interface

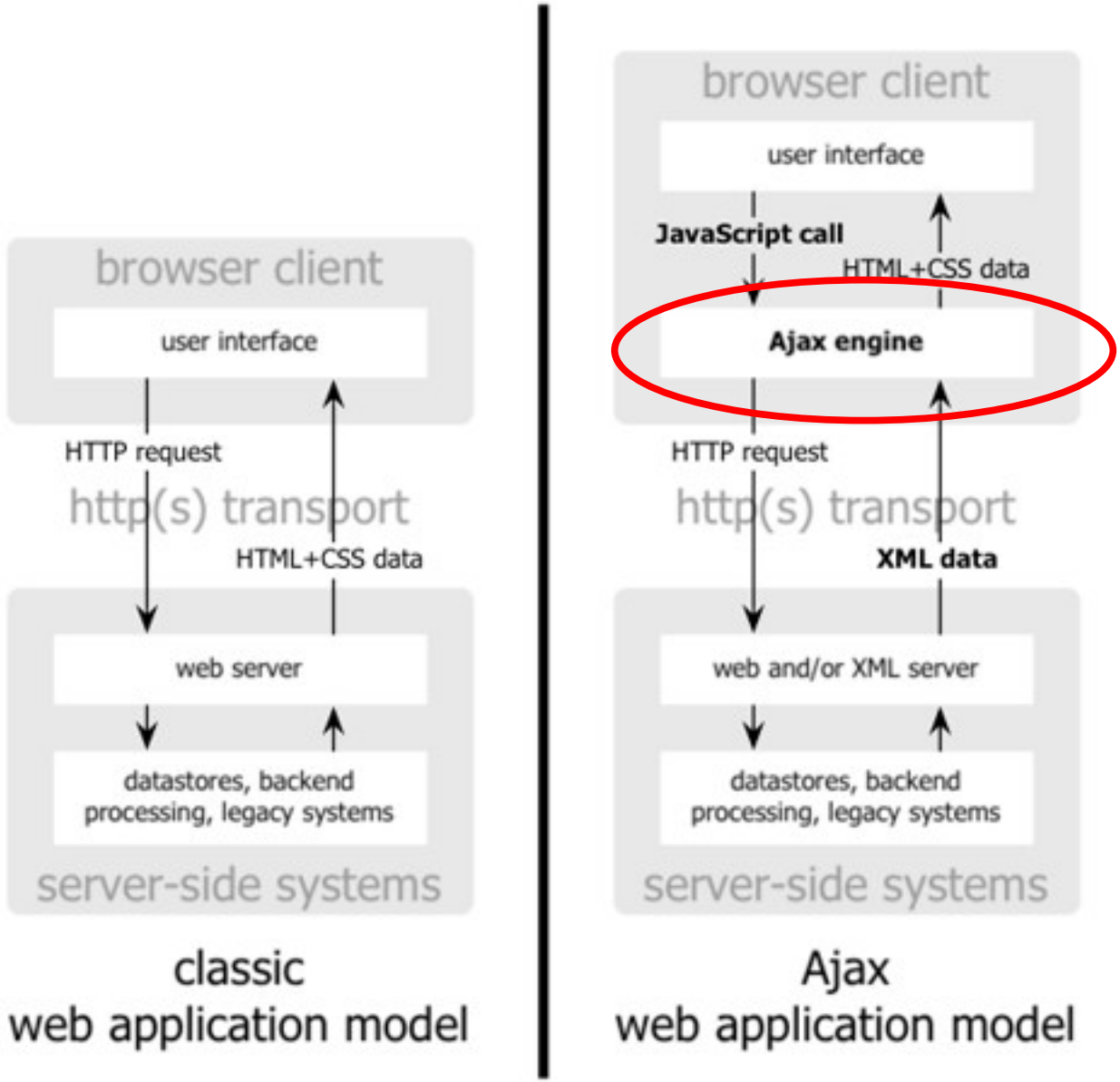


Interrupted user operation while the data is being fetched



Uninterrupted user operation while data is being fetched

Traditional Web vs AJAX: Components



Real-Life Examples of AJAX Apps

- Google maps
  - http://maps.google.com/
- Google Suggest
  - http://www.google.com/webhp?complete=1&hl=en
- Gmail
  - http://gmail.com/
- A9.com - Amazon.com search
  - http://a9.com/
- Many more are popping everywhere

Demo: AJAX

- AJAX Interaction
  - > Using “AutoComplete” Sample Application
- Examining under the hood.

NetBeans.ORG

Why AJAX?

- Intuitive and natural user interaction
  - No clicking required, mouse movement is a sufficient event trigger.
- "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)
- Data-driven instead of page-driven
  - Separation of displaying from data fetching

AJAX: Some usage scenarios

- Real-time server-side input form data validation
  - User IDs, serial numbers, postal codes, or even special coupon codes that require server-side validation can be validated in a form before the user submits a form
- Auto-completion
  - Email address, name, or city name may be auto-completed as the user types
- Master detail operation
  - Based on a client event, an HTML page can fetch more detailed information on data such as a product listing
- Advanced GUI widgets and controls
  - Controls such as tree controls, menus, and progress bars may be provided that do not require page refreshes

Technologies Used in AJAX

- JavaScript
  - Loosely typed scripting language
  - Allows programmatic interaction with the browser's capabilities
  - JavaScript function is called when an event in a page occurs
- DOM
  - API for accessing and manipulating structured documents
  - Represents the structure of XML and HTML documents
- CSS
  - Allows for a clear separation of the presentation from the content and may be changed programmatically by JavaScript
- HTTP
  - XMLHttpRequest

XMLHttpRequest

- JavaScript object
  - Created within a JavaScript function
- Adopted by modern browsers
  - Mozilla™, Firefox, Safari, IE 5, and Opera
- Communicates with a server via standard HTTP GET/POST
- XMLHttpRequest object works in the background
  - Does not interrupt user operation

XMLHttpRequest Methods

Method Name	Description
`open("method", "URL", asyncFlag)`	Setup the request (note async)
`send(content)`	Send the request (content = post data)
`abort()`	Stop the request
`getAllResponseHeaders()`	Return a hash of the headers
`getResponseHeader("header")`	Return the header value
`setRequestHeader("label", "value")`	Set a header

XMLHttpRequest Properties

Property Name	Description
`onreadystatechange`	Setup the callback event handler
`readyState`	Object status integer: 0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
`responseText`	Text value of response
`responseXML`	XML DOM of response
`status`	Status code (numeric)
`statusText`	Status text

Server-Side AJAX Request Processing

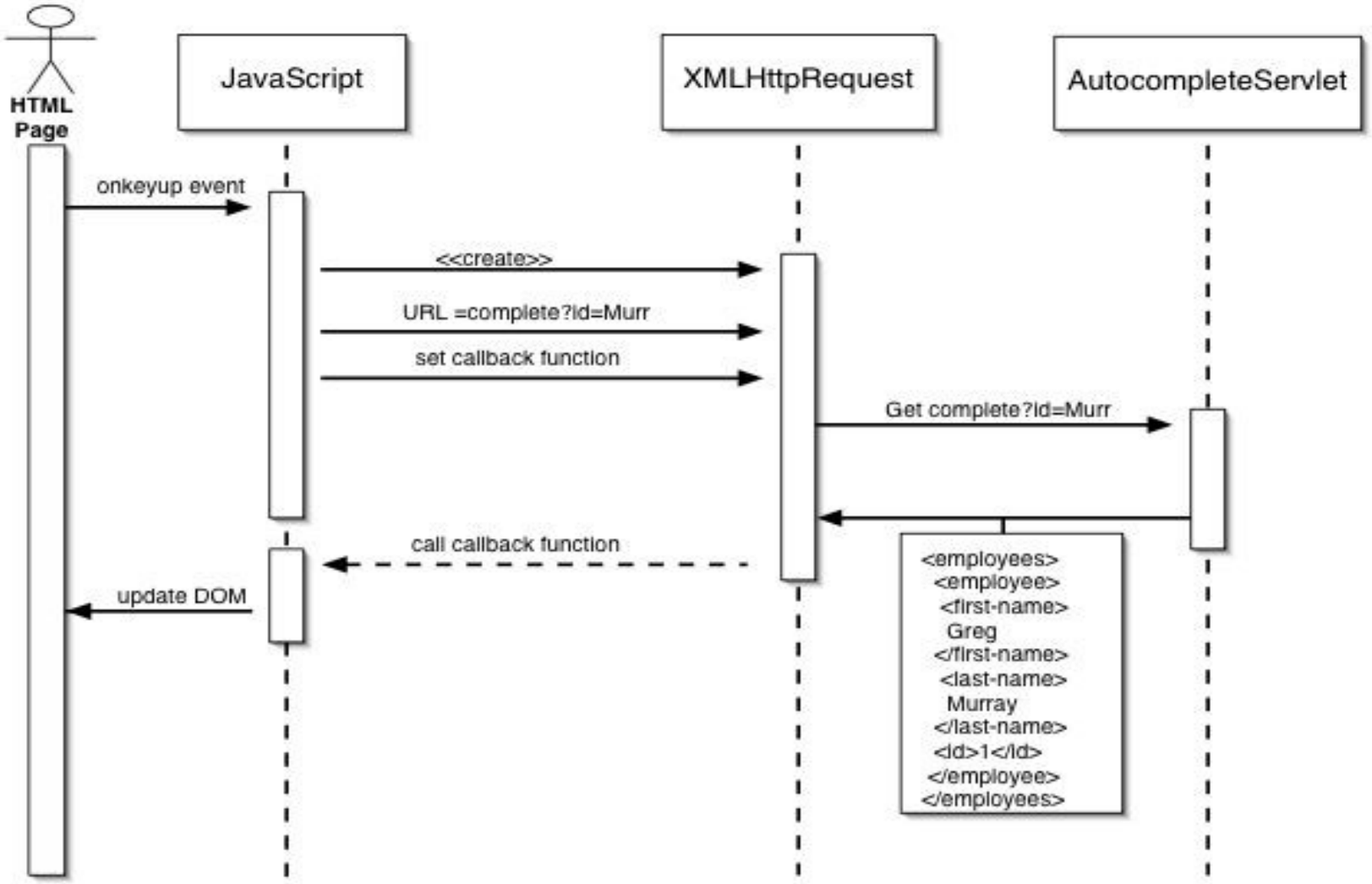
- Server programming model remains the same
  - It receives standard HTTP GETs/POSTs
  - Can use Servlet, JSP, JSF, ...
- With minor constraints
  - More frequent granular requests from client
  - Response content type needs to be text/xml

Demo: AJAX

- AJAX Interaction
  - > Using “AutoComplete” Sample Application
- Examining under the hood.

NetBeans.ORG

AutoComplete: Sequence Diagram



1. A Client event occurs

- A JavaScript function is called as the result of an event
  - Example: doCompletion() JavaScript function is mapped to a onkeyup event on a link or form component

```
1. <form name="autofillform" action="autocomplete" method="get">
2.     <input type="text"
3.             size="20"
4.             autocomplete="off"
5.             id="complete-field"
6.          name="id"
7.           onkeyup="doCompletion();">
8.   <input id="submit_btn" type="Submit" value="Lookup Employee">
9. </form>
```

2. An XMLHttpRequest object is created and configured ... contd.

```
1. function doCompletion() {
2.     if (completeField.value == "") {
3.         clearTable();
4.     } else {
5.         var url = "autocomplete?action=complete&id=" + escape
                                                    (completeField.value);
6.         var req = initRequest(url);
7.         req.onreadystatechange = function() {
8.             if (req.readyState == 4) {
9.                 if (req.status == 200) {
10.                    parseMessages(req.responseXML);
11.                } else if (req.status == 204){
12.                    clearTable();
13.                }}};
14.        req.open("GET", url, true);
15.        req.send(null);
16.    }}
```

2. An XMLHttpRequest object is created and configured

```
1. function initRequest(url) {
2.     if (window.XMLHttpRequest) {
3.         return new XMLHttpRequest();
4.     } else if (window.ActiveXObject) {
5.         isIE = true;
6.         return new ActiveXObject  ("Microsoft.XMLHTTP");
7.     }
8. }
```

3. XMLHttpRequest object makes an async. request

```
1. function doCompletion() {
2.    if (completeField.value == "") {
3.        clearTable();
4.    } else {
5.        var url = "autocomplete?action=complete&id=" + escape
                                                    (completeField.value);
6.        var req = initRequest(url);
7.        req.onreadystatechange = function() {
8.            if (req.readyState == 4) {
9.                if (req.status == 200) {
10.                   parseMessages(req.responseXML);
11.               } else if (req.status == 204){
12.                   clearTable();
13.               }}};
14.       req.open("GET", url, true);
15.       req.send(null);
16.   }}
```

4. The request is processed by the AutocompleteServlet at the Server

```
1. public void doGet(HttpServletRequest request, HttpServletResponse
   response)  throws IOException, ServletException {  ...
2.    String targetId = request.getParameter("id");
3.    Iterator it = employees.keySet().iterator();
4.    while (it.hasNext()) {
5.        EmployeeBean e = (EmployeeBean)employees.get((String)it.next());
6.        if ((e.getFirstName()... || e.getLastName()...) {
7.          sb.append("<employee>");    sb.append("<id>" + e.getId() + "</id>");
8.          sb.append("<firstName>" + e.getFirstName() + "</firstName>");
9.          sb.append("<lastName>" + e.getLastName() + "</lastName>");
10.         sb.append("</employee>");    namesAdded = true;
11.         } }
12.   if (namesAdded) {
13.     ...
14.   }
15.}
```

5. The AutocompleteServlet returns an XML document containing the result

```
1. public void doGet(HttpServletRequest request, HttpServletResponse
response)  throws IOException, ServletException {  ...
2.    ...
3.    while (it.hasNext()) {
4.        EmployeeBean e = (EmployeeBean)employees.get((String)it.next());
5.        ...
6.    if (namesAdded) {
7.        response.setContentType("text/xml");
8.        response.setHeader("Cache-Control", "no-cache");
9.        response.getWriter().write("<employees>" + sb.toString() +
"</employees>");
10.    } else {
11.        response.setStatus(HttpServletResponse.SC_NO_CONTENT);
12.  }
13.}
```

6. XMLHttpRequest object calls the callback function and processes the result

- The XMLHttpRequest object was configured to call the function when there is a state change to the readyState of the XMLHttpRequest object

```
1.  req.onreadystatechange = function() {
2.  if (req.readyState == 4) {
3.      if (req.status == 200) {
4.          parseMessages(req.responseXML);
5.      } else if (req.status == 304){
6.          clearTable();
7.      }}};
```

7. The HTML DOM is updated

- JavaScript technology gets a reference to any element in a page using DOM API
- The recommended way to gain a reference to an element is to call
  - responseXML.getElementsByTagName("employees")[0], where "employees" is the ID attribute of an element appearing in the HTML document
- JavaScript technology may now be used to modify the element's attributes; modify the element's style properties; or add, remove, or modify child elements

Tools

- Development
  - 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 LiveHTTPHeaders HTTP monitor
  - Microsoft Script Debugger (IE specific)

Current Issues with AJAX

- No standardization of XMLHttpRequest yet
  - W3C Working Draft on XMLHttpRequest
- No support of XMLHttpRequest in old browsers
  - IFrame+DOM+CSS is the solution here
- Using AJAX requires cross browser testing
- JavaScript dependency & incompatibility
  - Must be enabled for applications to function
  - Still some browser incompatibilities
- JavaScript code is visible to a hacker
  - Poorly designed JavaScript code can invite security problem

Current Issues with AJAX

- Complexity is increased
  - Presentation logic dispersed across HTML client pages and the server-side (Servlets/JSP, JSF)
  - Page developers must have JavaScript technology skills
- AJAX-based applications can be difficult to debug, test, and maintain
  - JavaScript is hard to test - automatic testing is hard
  - Weak modularity in JavaScript
  - Lack of design patterns or best practice guidelines yet

Agenda

- Web 2.0
- AJAX
  - Overview
  - Demo: AutoComplete
- Web 2.0 & AJAX Toolkits/Frameworks
  - DoJo
  - DWR
  - Jmaki

Types of AJAX Toolkit and Framework Solutions of Today

- Clients-side JavaScript Libraries
  - DoJo
- Wrapper
  - jMaki
- Remoting via proxy
  - DWR
- AJAX-enabled JSF components

Client-side JavaScript Libraries: Architecture

- 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

Client-side JavaScript Libraries: Characteristics

- 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: Implementations

- 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)
  - http://dojotoolkit.com/

Demo: DoJo

- DoJo

NetBeans.ORG

Client-side JavaScript Libraries: Other Implementations

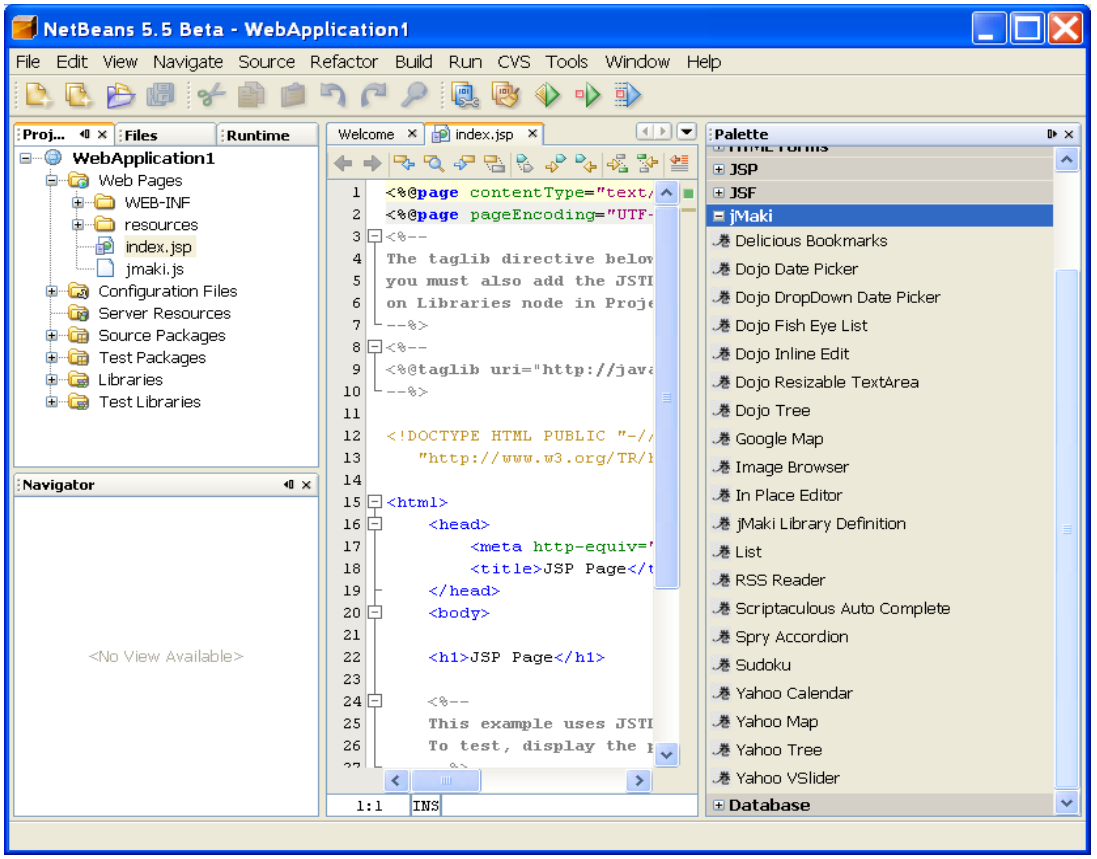
- Prototype
  - Focuses more on AJAX interactions
  - JavaScript objects and utility methods
  - Used by other toolkit libaries
  - http://prototype.conio.net/
- Script.aculo.us
  - Built on Prototype
  - Nice set of visual effects and controls
  - http://script.aculo.us/
- Rico
  - Built on Prototype
  - Rich AJAX components and effects
  - http://openrico.org/
- DHTML Goodies
  - Various DHTML and AJAX scripts
  - http://www.dhtmlgoodies.com/

Wrapper Framework: jMaki

- Project jMaki is a wrapper framework that allows developers to take widgets from popular AJAX frameworks, and wrap them into a JSP or JSF tag
  - Provides a common programming model to developers
  - Leverages the widgets from popular frameworks
  - Familiar to Java EE application developers
- JavaScript Wrapper framework for the Java platform
  - The name, jMaki, was derived from "j," for Java, and "Maki," a Japanese word for wrap

Demo: jMaki

- jMaki
  - > Samples
  - > Netbeans Plugin



Direct Web Remoting (DWR)

- When you want to expose existing Java backend business logic to AJAX client with minimum effort
  - Do RPC calls from client-side JavaScript to Java objects in a web container server side
  - Allows RMI like syntax in the client side JavaScript code
  - Framework generates client stub (Proxy), which is a JavaScript code
    - Client stub (Proxy) handles marshaling of parameters and return value
  - http://getahead.ltd.uk/dwr

AJAX-enabled JSF Components

- AJAX-enabled JSF components hide 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 NetBeans (with Visual Web Pack) 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)
  - http://developers.sun.com/ajax/componentscatalog.jsp
  - https://bpcatalog.dev.java.net/ajax/jsf-ajax/
- ajax4jsf (open-source)
  - Can add AJAX capability to existing applications
  - https://ajax4jsf.dev.java.net/
- ICEfaces (ICESoft) - commercial
  - http://www.icesoft.com/products/icefaces.html

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

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
    - Wicket, Shale etc.

Summary

- Web 2.0 & Rich Internet Application technologies
- AJAX helps make applications more interactive
- J2EE technology is a great platform for AJAX applications
- AJAX does not come for free
  - Issues involved
- Choose the right Toolkits and Frameworks for your application

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
  - http://java.sun.com/developer/technicalArticles/J2EE/AJAX/index.html
- AJAX Frameworks
  - http://ajaxpatterns.org/wiki/index.php?title=AJAXFrameworks
- 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

UNLOCK
OPPORTUNITY
What will you open?

Thank You!

Abhishek Mahanty
Architecture Solutions Team
AST@Sun.COM

solaris Java

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference