



Easy AJAX

Leon Fan

Senior Software Engineer

Sun Microsystems



Agenda

AJAX

Developpe AJAX in GlassFish

AJAX current Issue & Future

Summary

Agenda

AJAX

Developpe AJAX in GlassFish

AJAX current Issue & Future

Summary

AJAX

Why and What?

Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

AJAX

What and Why?

Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

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

Why AJAX?

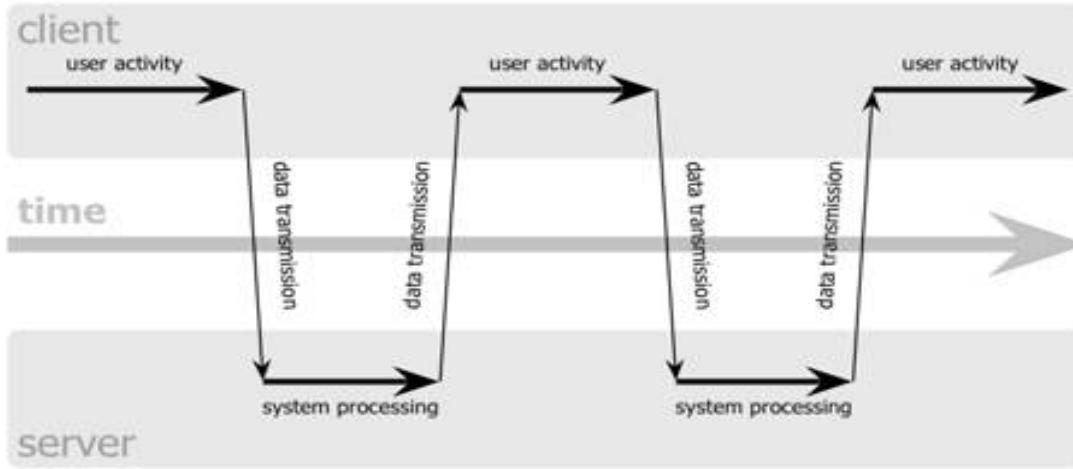
- Asynchronous communication replaces "synchronous request/response model."
 - > A user can continue to use the application while the client program requests information from the server **in the background**
 - > Separation of displaying from data fetching

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

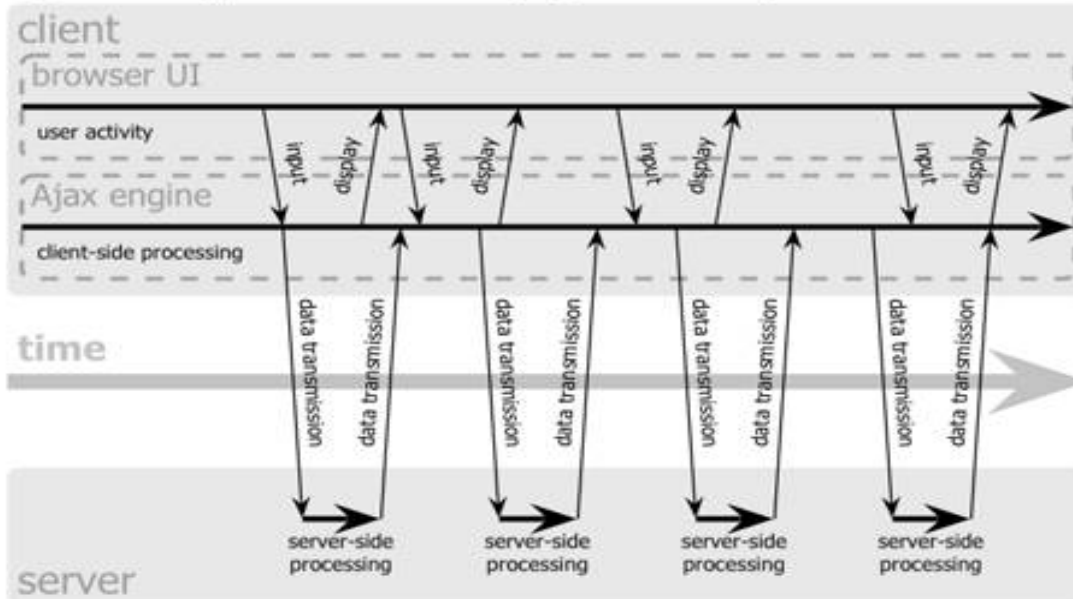
Model Comparison

classic web application model (synchronous)



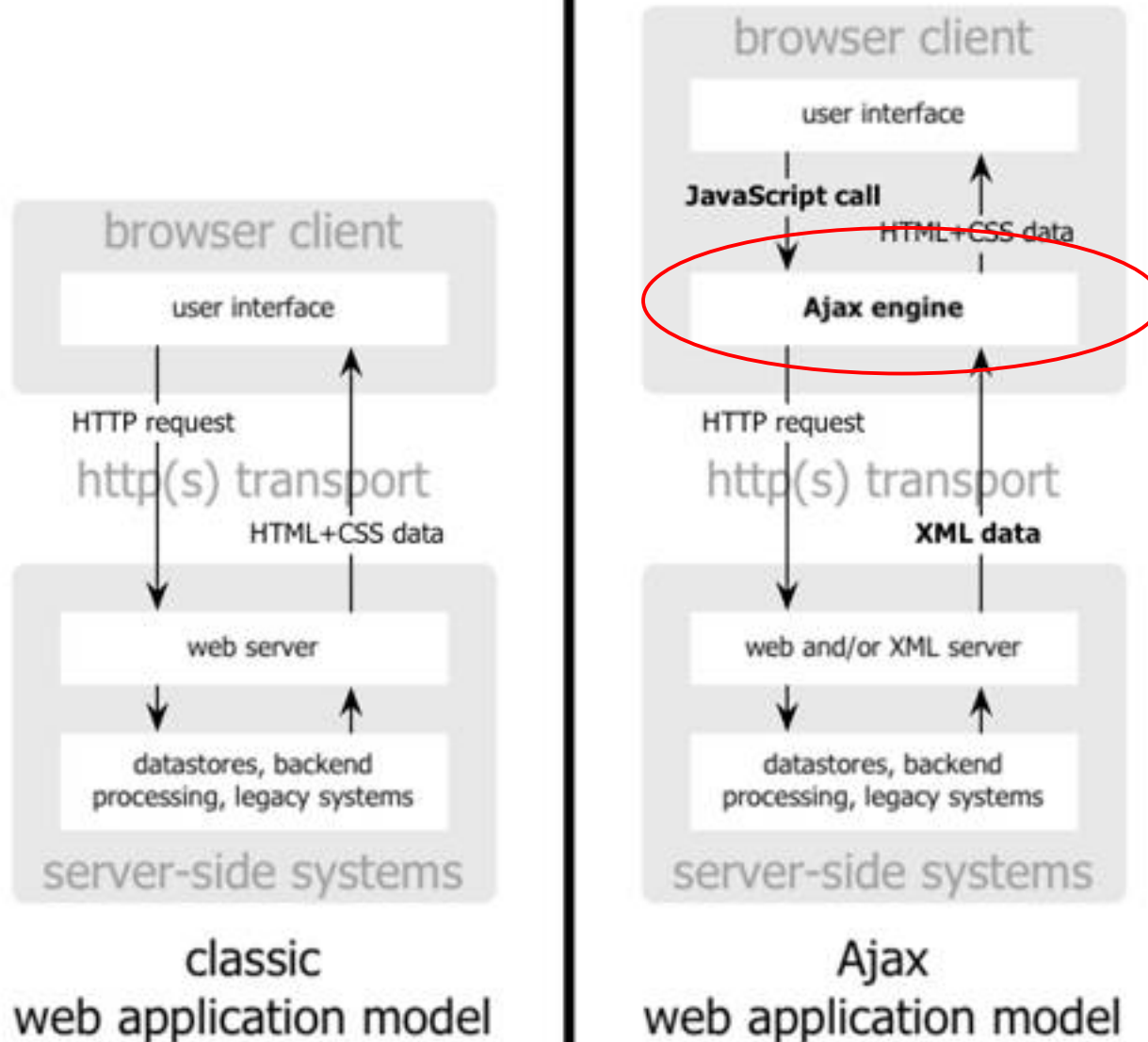
Interrupted user operation while the data is being fetched

Ajax web application model (asynchronous)



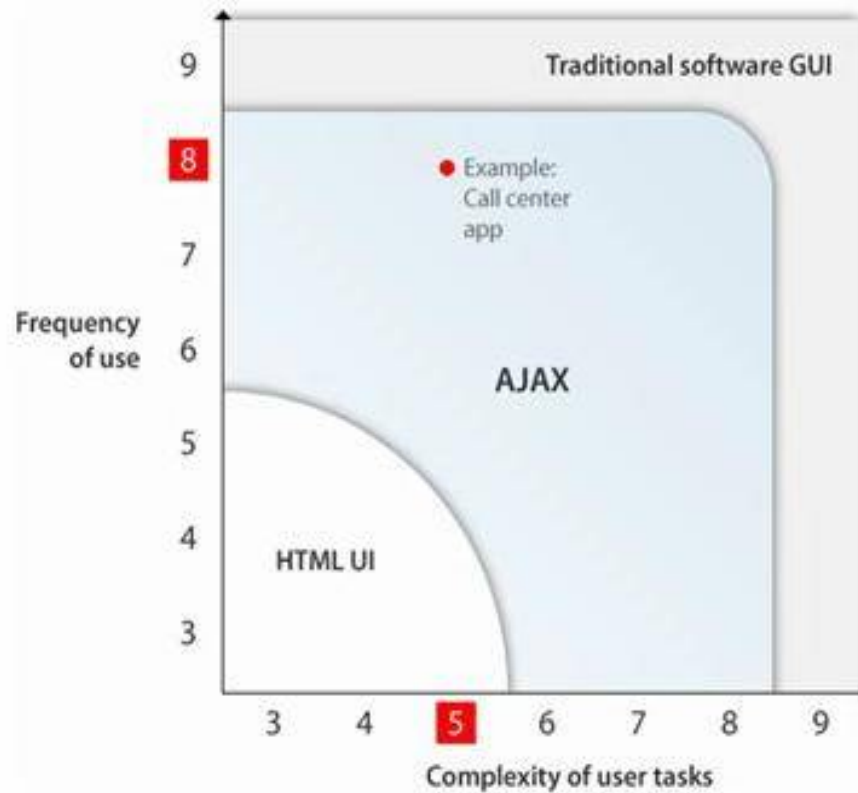
Uninterrupted user operation while data is being fetched

Model Comparison



Use AJAX or not

Determine High-Impact Apps



Source: Forrester 2006

AJAX

What and Why?

Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

Technologies Used In AJAX

- **Javascript**
 - > Loosely typed scripting language
 - > JavaScript function is called when an event in a page occurs
 - > Glue for the whole AJAX operation
- **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 style from the content and may be changed programmatically by JavaScript
- **XMLHttpRequest**
 - > JavaScript object that performs asynchronous interaction with the server

XMLHttpRequest

- JavaScript object
- Adopted by modern browsers
 - > Mozilla™, Firefox, Safari, and Opera
- Communicates with a server via standard HTTP GET/POST
- XMLHttpRequest object works in the background for performing asynchronous communication with the backend server
 - > Does not interrupt user operation

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 and finer-grained requests from client
 - > Response content type can be
 - > text/xml
 - > text/plain
 - > text/json
 - > text/javascript

AJAX

What and Why?

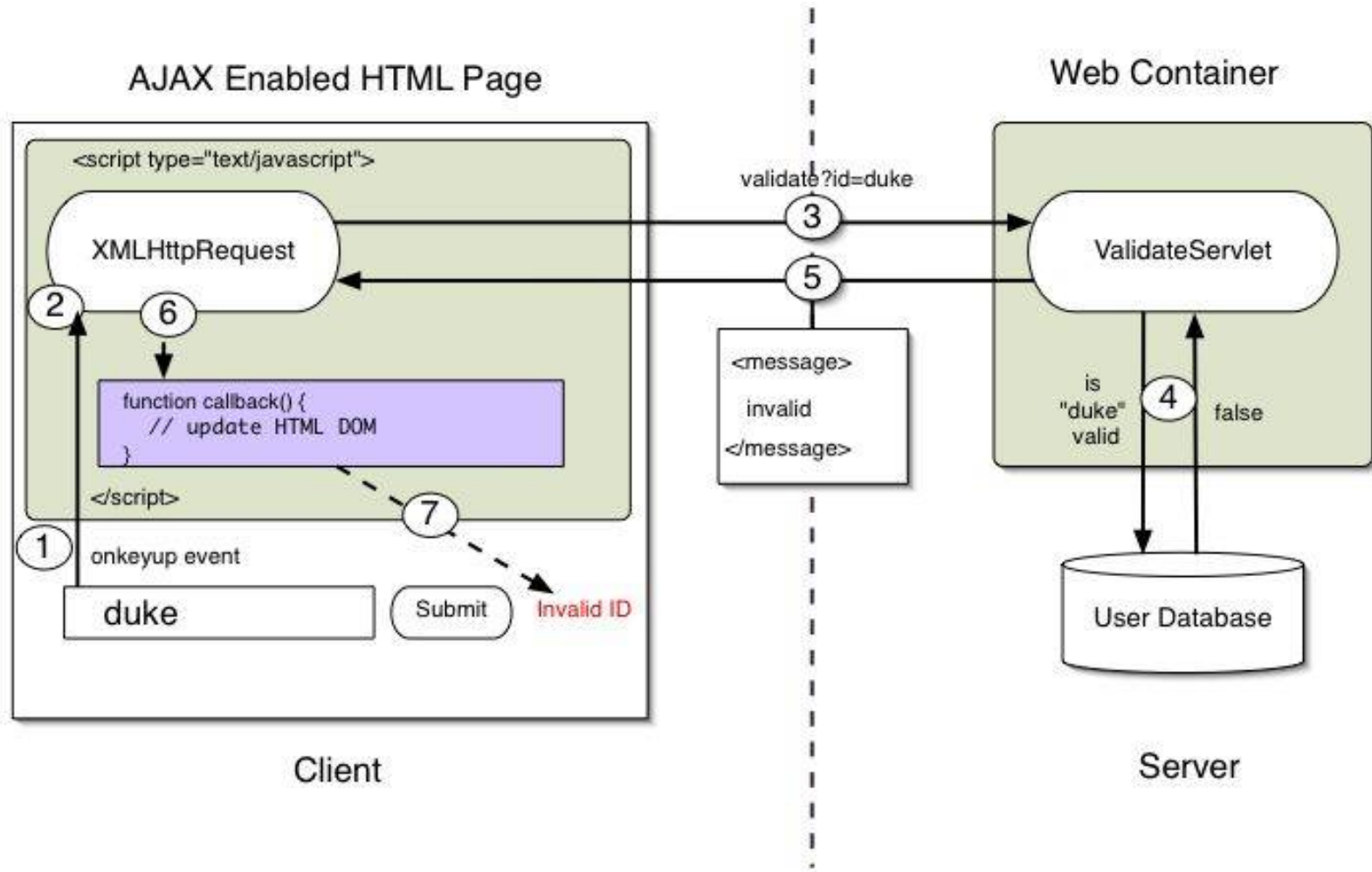
Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

Anatomy of an AJAX Interaction (Data Validation Example)



Steps of AJAX Operation

1. A client event occurs
2. An XMLHttpRequest object is created
3. The XMLHttpRequest object is configured
4. The XMLHttpRequest object makes an async. request
5. The ValidateServlet 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

1. A Client event occurs

- A JavaScript function is called as the result of an event
- Example: `validateUserId()` JavaScript function is mapped as a event handler to a `onkeyup` event on input form field whose id is set to “`userid`”

```
<input type="text"  
      size="20"  
      id="userid"  
      name="id"  
      onkeyup="validateUserId();">
```

2. An XMLHttpRequest object is created

```
var req;
function initRequest() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
}

function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest;
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    req.open("GET", url, true);
    req.send(null);
}
```

3. An XMLHttpRequest object is configured with a callback function

```
var req;
function initRequest() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

```
function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest; // callback function
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    req.open("GET", url, true);
    req.send(null);
}
```

4. XMLHttpRequest object makes an async. request

```
function initRequest() {  
    if (window.XMLHttpRequest) {  
        req = new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        req = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```

```
function validateUserId() {  
    initRequest();  
    req.onreadystatechange = processRequest;  
    if (!target) target = document.getElementById("userid");  
    var url = "validate?id=" + escape(target.value);  
    req.open("GET", url, true);  
    req.send(null);  
}
```

- URL is set to `validate?id=greg`

5. The `ValidateServlet` returns an XML document containing the results (Server)

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
```

```
    String targetId = request.getParameter("id");
```

```
    if ((targetId != null) && !accounts.containsKey(targetId.trim())) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
    } else {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>>false</valid>");
    }
}
```

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

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

```
function processRequest() {  
    if (req.readyState == 4) {  
        if (req.status == 200) {  
            var message = ...;  

```

...

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
 - > `document.getElementById("userIdMessage")`, where "userIdMessage" 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

7. The HTML DOM is update(Cont.)

```
1. <script type="text/javascript">
2. function setMessageUsingDOM(message) {
3.     var userMessageElement = document.getElementById("userIdMessage");
4.     var messageText;
5.     if (message == "false") {
6.         userMessageElement.style.color = "red";
7.         messageText = "Invalid User Id";
8.     } else {
9.         userMessageElement.style.color = "green";
10.        messageText = "Valid User Id";
11.    }
12.    var messageBody = document.createTextNode(messageText);
13.    // if the messageBody element has been created simple replace it otherwise
14.    // append the new element
15.    if (userMessageElement.childNodes[0]) {
16.        userMessageElement.replaceChild(messageBody,
17.                                         userMessageElement.childNodes[0]);
18.    } else {
19.        userMessageElement.appendChild(messageBody);
20.    }
21.}
22.</script>
23.<body>
24.  <div id="userIdMessage"></div>
25.</body>
```

AJAX

What and Why?

Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

Types of AJAX ToolKit and Framework Solutions of Today

- Clients-side JavaScript Libraries
- RMI-like remoting via proxy
- AJAX-enabled JSF components
- Wrapper(jMaki)
- Java to JavaScript/HTML translator(GWT)
- Web Application Frameworks with AJAX extension

Client-side JavaScript Libraries

- DOJO Toolkit
 - > Most prominent and comprehensive
 - > Gaining a leadership in this space
 - > Major industry support(Sun, IBM)
 - > <http://dojotoolkit.com/>
- Prototype
 - > Used by other toolkit libraries
 - > Wrapper(jMaki)
- Yahoo User Interface(YUI)

Clientside JavaScript Libraries(Cont.)

- 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/>

RMI-like remoting via proxy

- Direct Web Remoting(DWR)
 - > Designed specifically for Java application at the backend
 - > <http://getahead.ltd.uk/dwr>
- JSON-RPC
 - > Lightweight remote procedure call protocol similar to XML-RPC
 - > <http://json-rpc.org/>
 - > There are language-specific implementations
 - > <http://oss.metaparadigm.com/jsonrpc/>

AJAX-Enabled JSF Components

- DynaFaces
 - > Development on-going
 - > <https://jsf-extensions.dev.java.net/nonav/mvn/>
- Blueprint AJAX-enabled JSF components
 - > opensource
- Ajax4jsf
 - > opensource
 - > Can add AJAX capability to existing applications
 - > <https://ajax4jsf.dev.java.net/>

Wrapper(jMaki)

- JavaScript Wrapper framework for the Java platform
- Allows developers to take widgets from many popular AJAX toolkits and framework, and wrap them into a JSP or JSF tag
 - > Provides a common programming model to developers
 - > Leverages the widgets from popular frameworks
 - > JSP and JSF tags are familiar to Java EE application developers
- <https://ajax.dev.java.net/>

Java to JavaScript/HTML translator

- Develop and debug AJAX applications in the Java language using the Java development tools
- Translates Java application to browser-compliant JavaScript and HTML when deploy
- <http://code.google.com/webtoolkit/>

Web Application Frameworks with AJAX extension

- Wicket
 - > <http://wicket.sourceforge.net/>
- Echo2
 - > <http://www.nextapp.com/platform/echo2/echo/>
- Shale
 - > <http://struts.apache.org/struts-shale/>
- Ruby on Rails
 - > <http://www.rubyonrails.org/>

What Should I use?

- Features
 - > AJAX Engine
 - > Support Web Features
 - > Back-forward button
 - > Bookmarking
 - > Print
 - > UI Components & Behaviors
 - > Accessibility
 - > Server Integration
 - > Development Tools
- Security

What Should I use?

- Performance & Testing
- No-Product criteria
 - > Document & Examples
 - > Developer Community
 - > Licensing
- Your project own situation
 - > Existed application deployed
 - > Project's complexity
 - > Developer's skill

AJAX

What and Why?

Technology in AJAX

Data Validation Sample

AJAX Frameworks and Toolkits

Demo

Demo

Agenda

AJAX

Run AJAX in GlassFish

AJAX current Issue & Future

Summary

AJAX in GlassFish

What is GlassFish?

Why GlassFish?

Demo

What Is Project GlassFish?

- Java EE 5 Reference Implementation
 - > Included in Java EE 5 SDK
- Enterprise Quality Application Server
 - > SJS AS 9.0 & 9.1 PE / SE
- Open Source
- Community at Java.Net
 - > Sources, bug DBs, discussions at Java.Net
 - > Roadmaps, Architecture Documents

Timeline of Project GlassFish

Tomcat

Jasper

Catalina

JSTL

Struts

Crimson

XSLTC

Xalan

Xerces

JAXB

JAX-RPC

JSF



J1'04

June 2004

GlassFish

Launch



J1'05

June 2005

V1 final



J1'06

May 2006

V1UR1



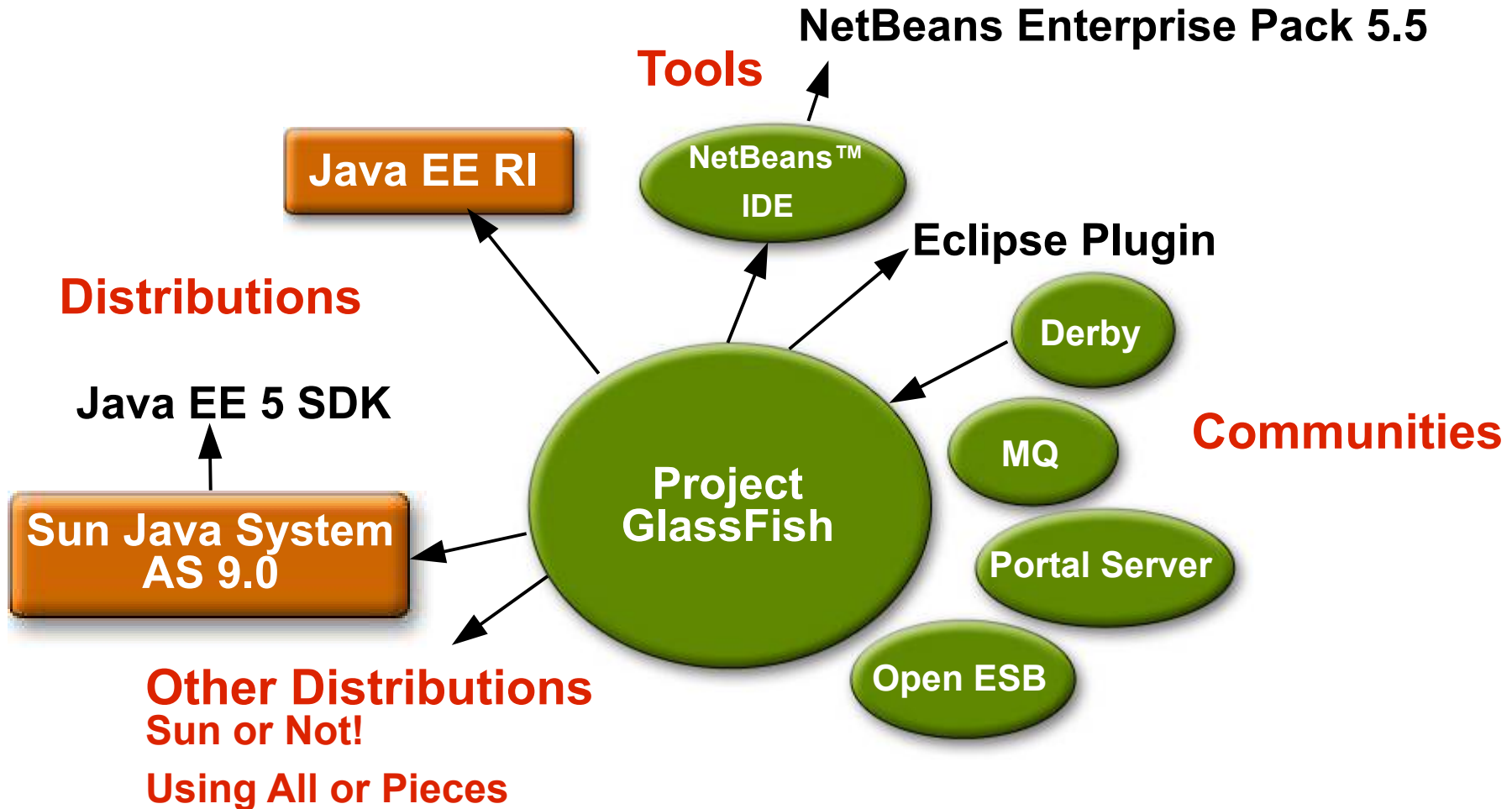
V2 (plan)



Apr 2007

tentative

Big Picture



Releases in Project GlassFish

- GlassFish v1
 - > Released! – Victory! Java EE 5 Compliance!
 - > UR1 - bug fixes
- GlassFish v2
 - > New WS stack, performance, startup time
 - > Load balancing, cluster management, failover
 - > Some scripting support
 - > Community, Transparency, Adoption
- GlassFish v3
 - > Larger architectural changes
 - > Better modularization, better scripting support

Frameworks and Applications

RIFE **blojsom** **OSWorkflow** **Lucene** **AtLeap**
JIVE SOFTWARE **RAILS** **AppFuse** **php** **OSCache** **iBATIS** **Project Tango**
Apache Httpd **CONFLUENCE** **JIRA** **CJUG-Classifieds** **Struts**
Open ESB **netBeans.ORG** **BIRT** **jBPM**
DOJO **Facelets** **eclipse** **MyFaces** **OpenSSO** **APACHE PLUTO**
Shale **Derby** **ADF** **SiteMesh** **WebDAV**
maven **Roller** **JSPwiki** **AJAX** **jasperreports**
Spring **Dalma** **PostgreSQL** **MC4J** **Tapestry**
StringBeans Portal **Hudson**
BlogTrader **WebSphere MQ** **Wicket** **Equinox**
Java WSDP

Community Contributors

- Code donations
 - > Sun Microsystems: SJS Application System 8.x+
 - > Oracle: Top Link Essentials
- Specification leadership
 - > SUN, Oracle, BEA
- Participation from corporations
 - > TmaxSoft (JEUS 6)
 - > BEA (WS Stack)
 - > JBoss (JAXB, JSF)
 - > Jetty (Grizzly)
- Individual contributors

AJAX in GlassFish

What is GlassFish?

Why GlassFish?

Demo

Great Application Server

- Opensource with Enterprise Quality
- High performance
- New Grizzly HTTP framework
- Java Persistence: TopLink Essentials
- Excellent Management Feature
- Lots of community contributor

Grizzly

What Is the Grizzly HTTP Framework?

- Grizzly is a HTTP framework
 - > Uses lower level Java NIO primitives
 - > Easy-to-use, high-performance APIs for socket communications
- Grizzly brings non-blocking sockets to the HTTP processing layer

Management Features

- Centralized, secure, remote access
 - > Accessible as GUI, CLI, IDEs, Java-based programs
- Off-line configuration
- Industry-standard infrastructure
 - > Java Management Extensions, JMX™ API
- Custom ANT tasks
- Management and Monitoring API
 - > Application Server Management eXtensions, AMX
- Can be monitored through *jConsole* and others

More than a Great AppServer - What You Need

- Tools, including an Eclipse Plug-In
- Samples, Documentation, How-To
- SOA - BPEL Engine, JBI Integration
- A DataBase - Derby
- Pragmatic Approach
 - > Improved Startup time
 - > No Security Manager by Default
- Focus on Popular Frameworks and Applications
 - > Running out of the box

AJAX and Scripting Activities

- jMaki - <http://ajax.dev.java.net>
 - > Encapsulates very easily AJAX widgets
- Phobos - <http://phobos.dev.java.net>
 - > Scripting on the Server
- Comet and Grizzly
 - > Long-term HTTP connections for push content
- DynaFaces - <http://jsf-extensions.dev.java.net>
 - > AJAX and JSF
- Blueprints - <http://bpcatalog.dev.java.net>
 - > Guidelines on many areas, including AJAX

AJAX in GlassFish

What is GlassFish?

Why GlassFish?

Demo

Demo

- Setup Environment
 - > Netbeans 5.5
 - > Netbeans JavaScript editor plug-in
 - > <http://www.liguorien.com/jseditor>
 - > Mozilla FireBug debugger
 - > GlassFish
 - > Add GlassFish in Netbeans
- Dojo-json demo

Agenda

AJAX

Developpe AJAX in GlassFish

AJAX current Issue & Future

Summary

Current Issues of AJAX

- Complexity is increased
 - > Server side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic
 - > 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
- Toolkits/Frameworks are not mature yet
 - > Most of them are in beta phase

Current Issues of AJAX

- No standardization of the [XMLHttpRequest](#) yet
 - > Future version of IE will address this
- No support of [XMLHttpRequest](#) in old browsers
 - > Iframe will help
- JavaScript technology 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

Browsers Which Support XMLHttpRequest

- Mozilla Firefox 1.0 and above
- Netscape version 7.1 and above
- Apple Safari 1.2 and above.
- Microsoft Internet Explorer 5 and above
- Konqueror
- Opera 7.6 and above

AJAX Futures

- AJAX-enabled JSF Component libraries
- Standardization of XMLHttpRequest
- Better browser support
- Better and Standardized Framework support
- More best practice guidelines in the programming model

Agenda

AJAX

Developpe AJAX in GlassFish

AJAX current Issue & Future

Summary

Summary

- Writing AJAX application is quite easy, it is also quite difficult
- Choose proper AJAX framework according to your situation
- Together with Netbeans, GlassFish is one of best choice to develop and deploy AJAX application
- Stay in Touch!
 - > <http://glassfish.dev.java.net>

Q & A ?



Easy AJAX

Leon Fan

Senior Software Engineer

Sun Microsystems

