Java™ EE 5 BluePrints for AJAX-Enabled Web 2.0 Applications

Sean Brydon, Greg Murray, Inderjeet Singh, Mark Basler

Java BluePrints
Sun Microsystems Inc.
http://blueprints.dev.java.net/
TS-1615
Goal of Our Talk

Learn how to architect and build web applications using Java™ Platform, Enterprise Edition (Java EE) 5 technology
Agenda

Definitions: Web 2.0, Rich Web Applications, AJAX
Demo
Guidelines
JavaServer™ Faces Technology Approach
Status and Futures
Q & A
Web 2.0

- Web as a Platform
- Collection Intelligence
  - Folksonomy—Collaborative Categorization
- Data is key and should be shared
- Software is in constantly evolving
  - Software release cycles dead?
- Lightweight Programming Models
  - SOAP/REST
- The Network is the computer
- Rich User Experience
Conventional Rich Web Applications

- Plugins/ Applets
- Frames/ iframes
- Dumb browser
- Server Centric
- Page to Page navigation based
Conventional Interaction Model

Client

User

Index.jsp URL

Show Products Event

Show Product Event

Add to Cart Event

Add to Cart Event

Java EE Container

Controller

Cart

HTTP GET index.jsp

HTTP GET products.jsp

HTTP GET products.jsp

Add to Cart - HTTP POST

Update Cart Quantity - HTTP POST

Add Product

Cart Details

Update Cart Quantity

Cart Details

Browser UI

index.jsp

products.jsp

product.jsp

cart.jsp
High Level AJAX Interaction Model
AJAX
Asynchronous JavaScript™ programming language + XML

- AJAX is using JavaScript, namely the XMLHttpRequest object, to communicate asynchronously with a server-side component and dynamically update the source of an HTML page based on the resulting XML/Text response.
Anatomy of an AJAX Interaction

HTML Page

XMLHttpRequest

function callBack() {
  //Update HTML Source
  onkeyup event

  <script type="text/javascript">
  <employees>
  </employees>

  <employee>
    <id>422</id>
    <name>Greg Murray</name>
  </employee>

  <employee>
    <id>232</id>
    <name>Gregg Murphy</name>
  </employee>

  <employee>
    <id>222</id>
    <name>George Klugy</name>
  </employee>

  <employee>
    <id>422</id>
    <name>Gregg Murphy</name>
  </employee>

  <employee>
    <id>232</id>
    <name>Gregg Murphy</name>
  </employee>

  <employee>
    <id>222</id>
    <name>George Klugy</name>
  </employee>

    Database

Java EE Container

EmployeeServlet

complete?id=G

{422,Greg Murray},
{232,Gregg Murphy},
{222,George Klugy}
HTML Page Event

```html
<form name="autofillform" action="autocomplete" method="get">
  <table border="0" cellpadding="5" cellspacing="0">
    <tr><td><b>Employee Name:</b></td><td>
      <input type="text" id="complete-field" size="20"
             autocomplete="off"
             onkeyup="doCompletion();">
    </td><td align="left">
      <input id="submit_btn" type="Submit" value="Lookup Employee">
    </td></tr>
    <tr><td id="auto-row" colspan="2">&nbsp;</td></tr>
  </table>
</form>
<div style="position: absolute; top:170px;left:140px" id="menu-popup">
  <table id="completeTable" border="1" bordercolor="black"
cellpadding="0" cellspacing="0" />
</div>
```
JavaScript Based Event Handler

```javascript
function getXHR() {
    if (window.XMLHttpRequest) {
        return new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        return new ActiveXObject("Microsoft.XMLHTTP");
    }
}

function doCompletion() {
    var url = "autocomplete?action=complete&id=" + encodeURI(target.value);
    var req = getXHR();
    req.onreadystatechange = processRequest;
    req.open("GET", url, true);
    req.send(null);
}
```
Servlet

```java
public void doGet(...) {
    String targetId = request.getParameter("id");
    Iterator it = emps.keySet().iterator();
    while (it.hasNext()) {
        EmployeeBean e = (EmployeeBean)emps.get((String)it.next());
        if ((e.getFirstName().toLowerCase().startsWith(targetId) ||
             e.getLastName().toLowerCase().startsWith(targetId))
             && !targetId.equals("")) {
            sb.append("<employee>");
            sb.append("<id>" + e.getId() + "</id>");
            sb.append("<firstName>");
            ...  
            sb.append("</employee>" Alla)
        }
    }
    response.setContentType("text/xml");
    response.getWriter().write("<employees>" +
    sb.toString() + "</employees>");
}
```
function postProcess(responseXML) {
   clearTable();
   var emps = responseXML.getElementsByTagName("employees")[0];
   if (emps.childNodes.length > 0) {
       completeTable.setAttribute("bordercolor", "black");
       completeTable.setAttribute("border", "1");
   } else {
       clearTable();
   }
   for (loop = 0; loop < emps.childNodes.length; loop++) {
       var e = emps.childNodes[loop];
       var firstName = e.getElementsByTagName("firstName")[0];
       var empId = e.getElementsByTagName("id")[0];
       appendEmployee(firstName.childNodes[0].nodeValue,
                       lastName.childNodes[0].nodeValue,
                       employeeId.childNodes[0].nodeValue,
                       empId.childNodes[0].nodeValue,
                       employeeId.childNodes[0].nodeValue);
   }
}
DEMO

AJAX Demo
Agenda

Definitions: Web 2.0, Rich Web Applications, AJAX
Demo

Guidelines
JavaServer™ Faces Technology Approach
Status and Futures
Q & A
AJAX Guidelines

- JavaScript Programming Language Libraries
- Usability
- I18n
- AJAX Design
- Architectures
- HTTP methods
- Return content-types
- Usecases/ Patterns
- Security
JavaScript Programming
Language Libraries

- Prototype
- RICO
- Script.aculo.us
- Dojo
- Zimbra

**Recommendation:** Adopt a library and don't try to re-invent the wheel
Usability

- Back/ Forward/ Refresh Buttons
- Bookmarking
- URL sharing
- Printing
- 508 Compliance

**Recommendation:** Consider the meaning of each and weigh the benefits when designing your application
XMLHttpRequest (XHR)

- **HTTP Method**
  - GET—When the result of \( N > 0 \) requests is the same
  - POST—When operation has “side-effects” and changes the state on the server

- **Concurrent Requests**
  - Max is 2 (IE) Consider—Pooling
  - JavaScript Technology Closures—Inline functions

**Recommendation:** Take care using the XHR. Use Closures to track the requests/callbacks. Consider using a library
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 `request.setCharacterEncoding("UTF-8")` before retrieving any parameters from Java EE
- Call `response.setContentType("text/xml; charset=UTF-8")`

**Recommendation:** Use UTF-8 since it supports the widest number of languages and browsers
AJAX Design

- Add Around the Edges
  - Small components (autocomplete, tree, partial submit)
- Page is the Application
  - Client and Server split MVC responsibilities

Recommendation: Consider designing initial AJAX applications around the edges as you gain experience. Don't go overboard
Architectures

- **Server Centric**
  - Server Renders Everything
- **Client Centric**
  - More JavaScript technology
  - Page as the Application
  - Client Controller

**Recommendation:** If you already have a server centric architecture consider adding some client centric components. When using a client centric architecture consider using an existing library.
Response Content Type

- XML
- HTML
- Text
  - Post processing on client
  - Inject directly into the page
- JavaScript Technology
  - Evaluated in JavaScript programming language using `eval()`
  - JavaScript based 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.
UsCases/Patterns

- Advanced Controls
- Autocomplete
- Observer
- Master Details
- Partial Submit
- Server Side Validation
- Value List Handler

Recommendation: Recommendation: Use AJAX to Enhance the user experience
Security

- Sandboxed
  - Cross Domain XMLHttpRequest restricted
  - Access to file system restricted
- HTTPS – Requires a page refresh
- JavaScript Programming Language Libraries for Encryption Exist
- JavaScript code visible to the world

**Recommendation:** Use HTTPS when you want to secure AJAX communication. Don't put compromising code in your JavaScript
Performance


“We peaked at approximately 103,000 simultaneous web visitors and 6,000 IRC viewers during the Keynote speech and transmitted over 32 GB of data in a three hour period. If not for the efficiency of the MacRumorsLive AJAX update system, the same webcast would have required approximately twice as many servers and would have had to transfer almost 6 times as much data (196 GB).”

- Patterns—Value List Handler/ Master Details
- JavaScript Technology—Compression

**Recommendation:** AJAX Performs! Use patterns and try to reduce the size of your JavaScript files. Consider dynamic loading of script
Agenda

Definitions: Web 2.0, Rich Web Applications, AJAX

Demo

Guidelines

JavaServer™ Faces Technology Approach

Status and Futures

Q & A
JavaServer Faces Technology
Component Approach

Benefits include:

- Control Content Rendering
- Control of Server Side Logic
- All in one component
- Reusable
- Usable in a tool
- Hide AJAX complexity from page developers
Anatomy of an AJAX enabled JavaServer Faces Based Component

Client

<script type="text/javascript" src="faces/ajax-textfield.js"

JavaScript

XMLHttpRequest

XMLHttpRequest Callback

onkeypress() event

Update HTML Source

Name: G

Greg Murray
Gregg Murphy
George Kluugy

Search

Java EE

AutoComplete JSF Component

FacesServlet

GET faces/ajax-textfield.js

RenderPhaseListener

GET faces/ajax-autocomplete.js

AutoCompleteTextField

AutoCompleteTextFieldTag

AutoCompleteTextFieldRenderer

items[]

doCompletion()

items[]

SessionBean

ApplicationBean

Form POST to Search URL

GET faces/ajax-textfield.js

GET faces/ajax-autocomplete.js

Search

G

Greg Murray
Gregg Murphy
George Kluugy

onkeypress() event

Update HTML Source
Page Developer’s View of JavaServer Faces Based Component

<ajaxTags:completionField
    size="40" id="cityField"
    completionMethod="
        #{ApplicationBean.completeName}"
/>

public String[] completeName() {
    ArrayList results = new ArrayList();
    Iterator it = employees.keySet().iterator();
    while (it.hasNext()) {
        EmployeeBean e = (EmployeeBean)employees.get((String)it.next());
        if ((e.getFirstName().toLowerCase().startsWith(targetId) || e.getLastName().toLowerCase().startsWith(targetId)) && !
            targetId.equals("") ) {
            results.add(e.getId() + "  " + e.getFirstName() +
                        e.getLastName());
        }
    }
    return (String[])results.toArray();
}
Agenda

Definitions: Web 2.0, Rich Web Applications, AJAX
Demo
Guidelines
JavaServer™ Faces Technology Approach
Status and Futures
Q & A
AJAX BluePrints

- Java BluePrints Solutions Catalog Entries on AJAX
  - NetBeans™ Technology
  - Command Line
  - Written for Project GlassFish
    http://glassfish.dev.java.net
- Java BluePrints AJAX Components
- Java Petstore Demo
Futures

- AJAX Pet Store
- More Java BluePrints Solutions Catalog entries
- More Java BluePrints AJAX components
- Better tool support
Summary

- Java technology provides the ideal platform for AJAX and Web 2.0 style applications
- Use AJAX where it makes sense
- Follow the guidelines
- The BluePrints team and BluePrints Solutions Catalog is a great AJAX resources
For More Information

- AJAX Components http://blueprints.dev.java.net/ajaxcomponents.html
- Java BluePrints Projects on Java.net http://blueprints.dev.java.net/
Q&A

Sean Brydon, Greg Murray, Inderjeet Singh, Mark Basler