Using the Dojo Toolkit to Develop AJAX-Enabled Java™ EE Web Applications

Alex Russell
The Dojo Foundation

Greg Murray
Sun MicroSystems

TS-3577

Web UIs Do Not Have to Be Painful

…either to build or to use

Agenda

JavaScript™ Technology and Rich UIs

What does Dojo Solve?

Demo

Dojo and Java™ Technologies

Java Technology Approaches

Demo

AJAX Is The Web, Evolved

- AJAX is about improved experience
  - Improves common interactions
  - Drastically improved experiences
- Responsive UIs are now approachable
  - Browsers now universally capable
  - Toolkits like Dojo reduce cost and complexity

Why Ajax Now?

- We’ve tapped out the expressive capacity of static HTML and CSS
- Interface idioms have settled down
- New kinds of applications require better UI primitives
- Tools have wrangled server-side complexity down
- REST and SOAP expose discrete services that browsers can now use

JavaScript Technology Is Powerful

- Completely dynamic
- Closures

```
var ctr = 0;
var foo = (function(){
      var bar = ++ctr;
      return function(){
            return bar;
      }
})();
alert(foo()); // alerts "1"
```

- But JavaScript technology is also missing a lot of things

Dojo: A Better Developer Experience

- Dojo Provides much of what JavaScript technology lacks:
  - Package system
  - Build system
  - Aspect-oriented event system
  - Rapid widget prototyping
  - Declarative markup for building UIs
  - Powerful I/O abstraction
- Dojo focuses on making things good, then fast

About Dojo

- Open Source
  - BSD or AFL 2.1 (you choose!)
  - All code owned by independent Foundation
- Vibrant community
  - Commercial support available
  - Shipping products based on Dojo
  - Daily contributions from independent contributors
- Written by experts, for real world apps
  - We know where the bodies are buried
  - We round off the sharp edges for you

Fixing the Event Mess

- Multiple event models from different spec versions
- Buggy browser implementations
- DOM Events are not treated the same way as other function calls
- Solution:
  - Aspect-oriented event system
  - Automatic memory leak prevention for IE
  - Topic-based “event cloud” for anonymous components

The Old (DHTML) Way

```
<!-- markup -->
<a onclick="doFoo();" id="foo" />

// from script
var fooNode = document.getElementById("foo");

// direct function assignment
fooNode.onclick = doFoo;

// DOM 2-style handlers
if(fooNode.attachEvent){
        fooNode.attachEvent("onclick", doFoo);
}else{
        fooNode.addEventListener("click", doFoo,
                false);
}
```

The Dojo Way

```
// from script
var fooNode = document.getElementById(”foo”);
dojo.event.connect(fooNode, “onclick”, doFoo);

// normal objects
var bar = { baz: function(){ alert(”baz!”); } };
var xyzzy = { quux: function(){ alert(”quux!”);} };

dojo.event.connect(bar, “baz”, xyzzy, “quux”);
// now, bar.baz() will also call xyzzy.quux();

// normal objects, anonymously
dojo.event.topic.subscribe(”/foo”, bar, “baz”);
dojo.event.topic.subscribe(”/foo”, xyzzy, “quux”);

// next line now calls bar.baz() and xyzzy.quux()
dojo.event.topic.publish(”/foo”, “arg1”, “arg2”);
```

DEMO

The Dojo Event System

Building Widgets

- Raw HTML only goes so far
- Widgets encapsulate UI behaviours
- Traditionally require hacks or onerous constraints to manage the state and event handling
- Solution:
    - HTML+CSS templates
    - Markup-driven widget creation
    - “Unobtrusive”
    - Built-in degradability

Widget Classes

```
// src/widget/FooWidget.js

dojo.widget.defineWidget({
"dojo.widget.FooWidget", // the widget we're defining
dojo.widget.HtmlWidget,  // what we inherit from
{                        // widget properties
templatePath: "/template/path.html",
templateCssPath:/template/path.css",

randomNode: null,
labelValue: "some label which you can change",
clickHandler: function(){
dojo.debug("clickHandler:",
this.randomNode.innerHTML);
}
}
);
```

Template Customization

```
<div>
  <span dojoAttachPoint=”randomNode”>
    ${labelValue}
  </span>
  <button dojoOnClick=”clickHandler”>click me</button>
</div>
```

- We can change the template any way we like

```
<div dojoAttachPoint=”randomNode”>
  <button dojoOnClick=”clickHandler”>click me</button>
  <p>${labelValue}</p>
</div>
```

Using the Widget

```
<!-- some_page.html -->

<html>
        <head>
                <script src="path/to/dojo.js"></script>
                <script>
                        dojo.require("dojo.widget.FooWidget");
                </script>
        </head>
        <body>
                <div dojoType="FooWidget"
              labelValue="foo bar baz">
           this text will be *replaced* when the widget
           is created you can therefore put content for
           non-DHTML browsers here
                </div>
        </body>
</html>
```

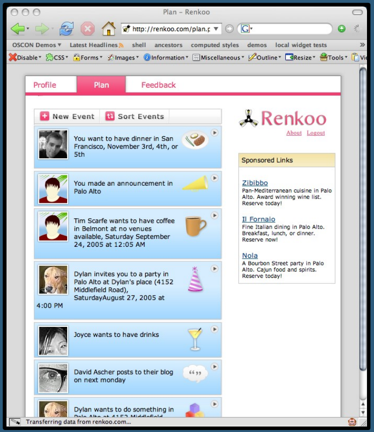
Talking to the Server

- dojo.io.bind() provides a powerful AJAX Abstraction

```
dojo.io.bind({
    method: “get”,
    url: “someEndpoint.jsp”,
    type: “text/plain”,
    content: {
        param1: “value1” // ...
    },
    error: function(type, event){ /* ... */ },
    load: function(type, data, event){ /* ... */ }
});
```

DEMO

Dojo Demos



Renkoo

Profile | Plan | Feedback

New Event | Sort Events

You want to have dinner in San Francisco, November 3rd, 4th, or 5th

You made an announcement in Palo Alto

Tim Scarfe wants to have coffee in Belmont at no venues available, Saturday September 24, 2005 at 12:05 AM

Dylan invites you to a party in Palo Alto at Dylan's place (4152 Middlefield Road), SaturdayAugust 27, 2005 at 4:00 PM

Joyce wants to have drinks

David Ascher posts to their blog on next monday

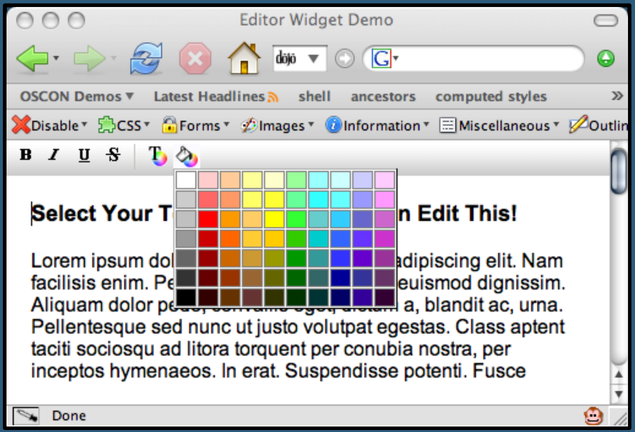
Sponsored Links

Zibibbo
Pan-Mediterranean cuisine in Palo Alto. Award winning wine list. Reserve today!

Il Fornaio
Fine Italian dining in Palo Alto. Breakfast, lunch, or dinner. Reserve now!

Nola
A Bourbon Street party in Palo Alto. Cajun food and spirits. Reserve today!

Transferring data from renkoo.com...

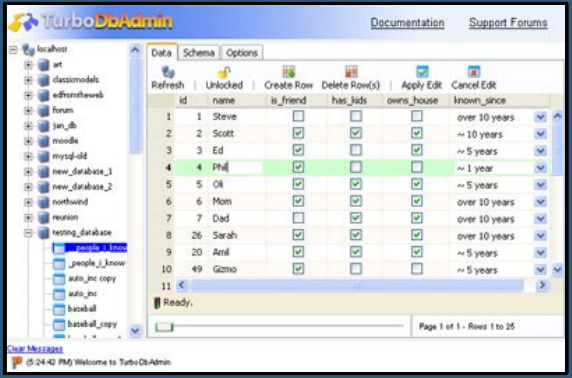


Editor Widget Demo

Select Your T… n Edit This!

Lorem ipsum dol… adipiscing elit. Nam facilisis enim. Pe… euismod dignissim. Aliquam dolor pe… a, blandit ac, urna. Pellentesque sed nunc ut justo volutpat egestas. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. In erat. Suspendisse potenti. Fusce

Done



TurboDbAdmin — Documentation | Support Forums

Data | Schema | Options

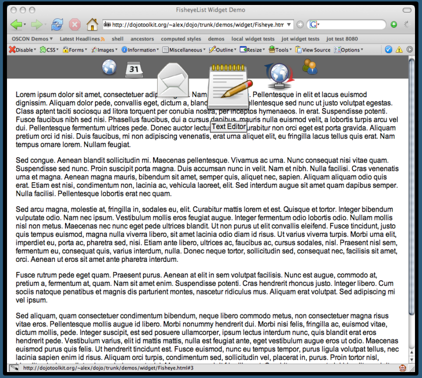
Refresh | Unlocked | Create Row | Delete Row(s) | Apply Edit | Cancel Edit

	id	name	is_friend	has_kids	owns_house	known_since
1	1	Steve	☐	☐	☐	over 10 years
2	2	Scott	☑	☑	☑	~ 10 years
3	3	Ed	☑	☐	☑	~ 5 years
4	4	Phil	☑	☐	☐	~ 1 year
5	5	Oli	☑	☑	☑	~ 5 years
6	6	Mom	☑	☑	☑	over 10 years
7	7	Dad	☐	☑	☑	over 10 years
8	26	Sarah	☑	☑	☑	over 10 years
9	20	Amil	☑	☑	☑	~ 5 years
10	49	Gizmo	☑	☐	☐	~ 5 years

Ready.

Page 1 of 1 - Rows 1 to 25

(5:24:42 PM) Welcome to TurboDbAdmin



FisheyeList Widget Demo

Dojo and Java Technologies

Dojo Provides:

- Widget Model
- Event handling
- I/O to Java
- Client-side rendering
- Cross browser support

Java Technology Provides:

- Services (JDBC™ software, Web Services, Data Access)
- Model data (Java Persistence API)
- Session support
- Business logic
- Persistence layer
- Content

Java Technology Approaches

- **Serlvet/JavaServer Pages™** technology provide server-side logic
- **JSP™/JavaServer™** Faces based Components specific for each Dojo component
- Generic **JSP Tag/JavaServer Faces** based component wrapper

Servlet Approach

- Servlets are ideal for Dojo services
  - Close to HTTP
  - Generating XML
  - Generating JSON
  - Linking to back end logic

- Servlet issues:
  - Nuts and bolts approach
  - Lots of out.println()
  - Mapping services (web.xml)

Servlet Based Service

```
 public  void doGet(HttpServletRequest request, HttpServletResponse  response)
        throws IOException, ServletException {
        String targetId = request.getParameter("id");
            Iterator it = employees.keySet().iterator();
            while (it.hasNext()) {
                String id = (String)it.next();
                EmployeeBean e = (EmployeeBean)employees.get(id);
                // simple matching only for start of first or last name
                if ((e.getFirstName().toLowerCase().startsWith(targetId) ||
                    e.getLastName().toLowerCase().startsWith(targetId)) &&
                    !targetId.equals(""))  {
                    sb.append("<employee><id>" + e.getId() + "</id>");
                    sb.append("<firstName>" + e.getFirstName() + "</firstName>");
                    sb.append("<lastName>" + e.getLastName() + "</lastName>");
                    sb.append("</employee>");
                    namesAdded = true;
                }
            }
             if (namesAdded) {
                 response.setContentType("text/xml");
                response.setHeader("Cache-Control", "no-cache");
                            response.getWriter().write("<employees>" + sb.toString() +
"</employees>");
            } else {
                response.setStatus(HttpServletResponse.SC_NO_CONTENT);
            }
          }
}
```

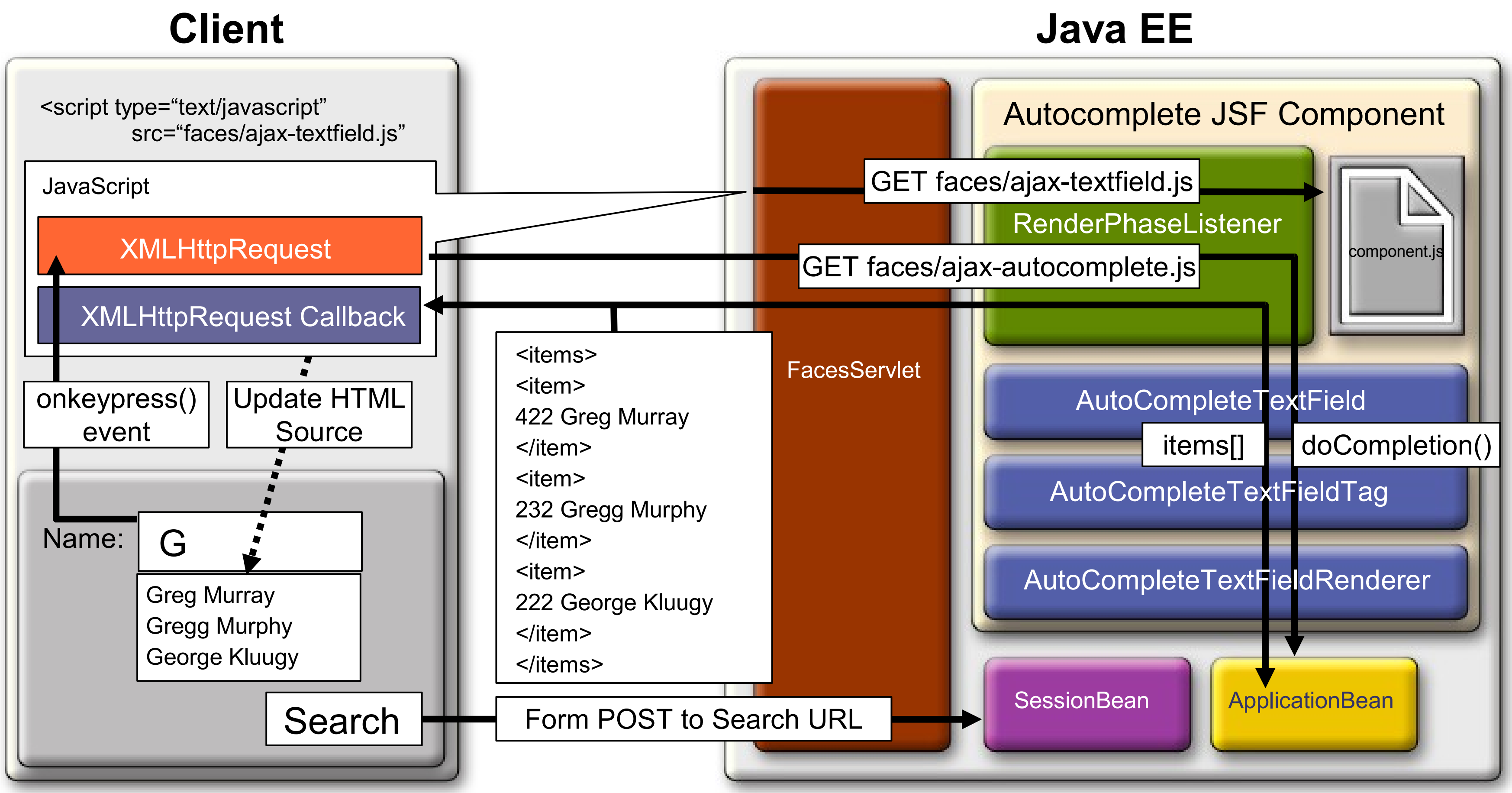
JavaServer Faces Technology Approach

JavaServer Faces based Components provide are ideal:

- Reusable
- Easy to bind to services using method bindings
- Toolable
- Hide the JavaScript technology/CSS

- JavaServer Faces technology issues:
    - Developing components correctly can take time
    - JavaServer Faces based component model tends to be more geared towards server-side rendering and view-state management

JavaServer Faces Technology Approach



```
<items>
<item>
422 Greg Murray
</item>
<item>
232 Gregg Murphy
</item>
<item>
222 George Kluugy
</item>
</items>
```

Wrapper Approach

- Generic JSF Component/JSP Tag
- Component binds template HTML + CSS + JavaScript
  - CSS links and script tags are only rendered once per page
- Generic renderer for JSF
- Generic PhaseListener to provide access to component resources
- Generic access to services using method binding expressions

- Issues:
  - Wrapper has many options
  - Loosely coupled to JavaScript centric component

The Dojo Wrapper

```
<f:view>

<a:ajax type="dojo" name="ComboBox">

<form action="#" method="GET">

   <input dojoType="combobox" value="this should be replaced!"

      dataUrl="AutocompleteBean-completeCountry.ajax"
style="width: 300px;" name="foo.bar">

   <input type="submit">

</form>

</a:ajax>

</f:view>
```

The Wrapper Java Logic:

```
private String[] countryCodes =
       new String[] {
           "CA", "FR", "UG", "UR", "USA", "UK", "JP", "KR", "JA", "TH"
       };

 public void completeCountry(FacesContext context, String[] args, AjaxResult result) {
       result.setResponseType(AjaxResult.JSON);
       result.append("[");
       for (int loop=0; loop < countries.length; loop++){
            result.append("[\"" + countries[loop] + "\",\"" + countryCodes[loop] + "\" ]");
            if (loop < countries.length -1) result.append(",");
       }
       result.append("];");
   }
```

Wrapper Approach

- Generic JavaServer Faces Component/JSP Tag
- Component binds template HTML + CSS + JavaScript
  - CSS links and script tags are only rendered once per page
- Generic renderer for JavaServer Faces technology
- Generic PhaseListener to provide access to component resources
- Generic access to services using method binding expressions
- Issues:
  - Wrapper has many options
  - Loosely coupled to JavaScript technology centric component

The Wrapper Decration

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

<h2>Dojo Autocomplete Test</h2>
<hr>

<f:view>
<a:ajax type="dojo" name="ComboBox"
     service="AutocompleteBean-completeCountry.ajax"
    template="autocomplete.htmf" />
</f:view>
```

Renderend HTML

```
<h2>Dojo Autocomplete Test</h2>
<hr>

<script type="text/javascript" src="dojo.js"></script>
<link rel="stylesheet" type="text/css"
href="ComboBox.css"></link>
<script type="text/javascript" src="ComboBox.js"></script>
<script type="text/javascript">
 dojo.require("dojo.widget.ComboBox");
dojo.hostenv.writeIncludes();
</script>

<form action="#" method="GET">
       <input dojoType="combobox" value="this should be
replaced!"
              dataUrl="AutocompleteBean-completeCountry.ajax"
style="width: 300px;" name="foo.bar">

       <input type="submit">
</form>
```

The Wrapper Java Technology Logic:

```
 private String[] countryCodes =
        new String[] {
            "CA", "FR", "UG", "UR", "USA", "UK", "JP", "KR",
"JA", "TH"
        };

  public void completeCountry(FacesContext context, String[]
args, AjaxResult result) {
        result.setResponseType(AjaxResult.JSON);
        result.append("[");
        for (int loop=0; loop < countries.length; loop++){
             result.append("[\"" + countries[loop] + "\",\""
+ countryCodes[loop] + "\" ]");
             if (loop < countries.length -1)
result.append(",");
        }
        result.append("];");
    }
```

Why Wrap?

- You don’t need to worry about writing the script or link tags
- You don’t need to worry about script tags getting called multiple times in a page
- You don’t need to write component bootstrap code
- You can utilize JavaScript technology centric components with JavaServer Faces technology—as is
- Leverage Java technology to do the boot-strapping and provide the services

When Writing Components That Use Dojo Keep in Mind:

- Don’t compress Jar files containing JavaScript based resources
- Use Dojo event system for cross component communication rather than home grown events
- Don't over-ride the onload of a page. Register onload handlers with dojo.addOnLoad()
- Dojo will modify the rendered HTML after the page is loaded and bind events
- Don't redefine dojo.js script tag more than once
- Use styleClass attribute name

DEMO

Servlet, JavaServer Faces Based Component, and Wrapper Approach

What’s Beyond Ajax?

- JSON-RPC
- Cross-domain request/response (xdAjax)
  - JSON-P mashups (dojo.rpc.YahooService)
- Comet: low-latency push from the server
- Local storage on clients (dojo.storage.*)
- Native vector rendering

Summary

- JavaScript technology can be used to create rich UIs
- Dojo fills in where JavaScript technology leaves out
- Dojo provides a great way of writing widgets
- Java technology is ideal for providing Dojo services
- Developers of Java based applications can use servlets/JSP technology to service Dojo requests
- A Java technology view of Dojo may be provided by a tightly bound component or a generic wrapper
- Together Dojo and Java technology provide a great solution

For More Information

List

- http://dojotoolkit.org/
- http://java.sun.com/blueprints/ajax.html

Q&A

Using the Dojo Toolkit to Develop AJAX-Enabled Java™ EE Web Applications

Alex Russell
The Dojo Foundation

Greg Murray
Sun MicroSystems

TS-3577