jQuery Datatables
Part I: The Phantom Menace

Widget of the Week (WoW) Series
Program sponsored by …

- org.apache.commons.lang.StringUtils
- org.apache.commons.beanutils.BeanUtils
Things to know …

- jQuery
- jQuery UI
- AJAX
- JSON
- Datatables.net
jQuery

- Arguably the most popular JavaScript library in use today
- Excels at DOM manipulation, event handling, client-server interaction
- Good cross-browser support
- Readily extensible through plug-ins and libraries …
jQueryUI

- Library of widgets, effects, events
- Extension to core jQuery
AJAX

- “Asynchronous JavaScript and X(HT)ML”
- Technique (not a framework) often implemented with jQuery
- Information retrieved from server with JavaScript XMLHttpRequest
- Displayed on page by manipulating DOM/CSS with JavaScript
- Major restriction: “same origin policy”
JSON

- “JavaScript Object Notation”
- Lightweight data interchange format
- Arguably human-readable
- Based on object and array literal syntax
Practical exercise

- Demonstrate use of jQuery, datatables API, AJAX, and Spring.
Formal requirements
Practical exercise …

- Retrieve data from data store
- Display data in enhanced table
- Allow end user to edit existing records
- Allow end user to add new records
- All on one page
- Graceful error handling
Client-server interaction
jQuery.ajax() method

- Takes care of the messy details of XMLHttpRequest
- Choice of GET or POST method
- Pass arbitrary parameters
- Set callback method
- Automatically detects return data type: XML, JSON, JavaScript, or HTML
Handling response data in AJAX

- Can return raw data (XML or JSON) and write to the underlying page with JavaScript
- Can return HTML fragment and just stick it somewhere on the page
Use the 5Ps approach:

- Prepare
- Plan
- Prototype
- Plumb
- Proselytize
- Secure
Prepare

- Which widget(s) do I want/need to use?
  - Datatables.net plugin, jQuery UI dialog

- What docs are available?
  - jQuery online docs, uxt.ucsd.edu, datatables.net

- What client-side resources are required?
  - jQuery core, jQuery UI, datatables plugin, HTML table

- What server-side resources are required?
  - Data source, SpringMVC forms, controllers, validators, etc.
Plan

- Build table data dynamically
- Use expandable row to view/edit record details
- Use jQuery UI dialog to add new record
- Use SpringMVC with AJAX to handle interactions
Prototype

- Create a basic HTML table, add expandable row enhancement and test implementation of AJAX.
<table>
<thead>
<tr>
<th></th>
<th>Trinity Lake</th>
<th>USBR</th>
<th>Trinity</th>
<th>2447700</th>
<th>1693520</th>
<th>1721143</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prototype (cont’d)

```javascript
this.src = "http://uxt.ucsd.edu/common/act/1/img/icon_arrow_down.gif";
var guid = nTr.id;
$.ajax({
    url: "fakeittilyoumakeit.html",
    context: nTr,
    data: { id: nTr.id },
    success: function(data) {
        wowTable.fnOpen(this, data);
    }
});
```
<table>
<thead>
<tr>
<th>Reservoir</th>
<th>Agency</th>
<th>Watershed</th>
<th>Capacity</th>
<th>Average</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don Pedro</td>
<td>USACE</td>
<td>Tucson</td>
<td>2030000</td>
<td>1344279</td>
<td>1692542</td>
</tr>
</tbody>
</table>

**Static Datatable**

- Name: Don Pedro
- Agency: USACE
- Watershed: Tucson
- Capacity: 2030000
- Average: 1344279
- Current: 1692542

---

Datatables-static.htm
Prototype (cont’d)

"Excellent. Everything is going as planned."
Plumb

- Create the add and update forms as if you were building a regular SpringMVC app ...
  - Form taglib
  - Validators
  - Controllers
  - Views
Plumb (cont’d)

- With one major exception … you can’t use a regular form submission!
- Write AJAX handler using jQuery
- Return results as JSON and update table data
Plumb (cont’d)

- In case of validation error, return form view with appropriate messages
- If callback return type is JavaScript object, assume success, update table
- If callback return type is string, assume failure, display as HTML
Proselytize

Tell your friends!
Secure

- Use of GET vs. POST method
  - GET method: primarily for retrieving data
  - POST method: anything that modifies data
Secure

- Use of GET vs. POST method
  - GET method: primarily for retrieving data
  - POST method: anything that modifies data
Alternatives

- jEditable
  - [http://www.datatables.net/examples/api/editable.html](http://www.datatables.net/examples/api/editable.html)
- Server-side data model
Optional exercise

### Distributions for ID: 000123456

<table>
<thead>
<tr>
<th>P/O</th>
<th>Name</th>
<th>Emp ID</th>
<th>Title</th>
<th>Appt</th>
<th>Dist</th>
<th>Index</th>
<th>Start</th>
<th>End</th>
<th>Expiration</th>
<th>Amount ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Joe Smith</td>
<td>000123456</td>
<td>7777</td>
<td>10</td>
<td>11</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P/O</th>
<th>Type</th>
<th>Description</th>
<th>Sub</th>
<th>% Time</th>
<th>Rate</th>
<th>Index</th>
<th>Start</th>
<th>End</th>
<th>Expiration</th>
<th>Amount ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>PPS</td>
<td>PPS</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
<tr>
<td>P</td>
<td>PR</td>
<td>Payroll</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
<tr>
<td>P</td>
<td>TF</td>
<td>Tuition Rem</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
<tr>
<td>P</td>
<td>GL</td>
<td>General Liability</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
<tr>
<td>P</td>
<td>BN</td>
<td>Benefits</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
<tr>
<td>P</td>
<td>VC</td>
<td>Vacation</td>
<td>1</td>
<td>1.0000</td>
<td>100.00</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>25.00</td>
</tr>
</tbody>
</table>

Projected SUBTOTAL $150.00

Notes

<table>
<thead>
<tr>
<th>Name</th>
<th>Emp ID</th>
<th>Title</th>
<th>Appt</th>
<th>Dist</th>
<th>Index</th>
<th>Start</th>
<th>End</th>
<th>Expiration</th>
<th>Amount ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe Smith</td>
<td>000123456</td>
<td>7777</td>
<td>20</td>
<td>21</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>100.00</td>
</tr>
<tr>
<td>Joe Smith</td>
<td>000123456</td>
<td>7777</td>
<td>30</td>
<td>31</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>100.00</td>
</tr>
<tr>
<td>Joe Smith</td>
<td>000123456</td>
<td>7777</td>
<td>40</td>
<td>41</td>
<td>ADC0011</td>
<td>08/16/2010</td>
<td>06/30/2011</td>
<td>06/30/2011</td>
<td>100.00</td>
</tr>
</tbody>
</table>

TOTAL $400.00
Middleware, UI, and database
Coming soon