ActiveState

Dynamic Tools for Dynamic Languages

XulDb

as case study with

XUL, PHP and jQuery

Jeff Griffiths, ActiveState
phptek Chicago 2007

Part 1
the legend of George Nava

‘in the beginning, there was digg’
 - a digg post over a year ago about a demo site featuring a bunch of Xul example applications caused a lot of buzz at ActiveState. The site was georgenava.com (now defunct)

Aside: Who is this George Nava guy anyway?

???

What is XUL?

- XML User Interface Language
- cross platform Gui
- HTML-like mark-up, DOM & JavaScript
- allows HTML tags
- basis for Firefox development
- Komodo, Songbird, etc.
- remote XUL, but more on that later

Why is XUL cool?

- simple Gui design
- richer interface widgets as tags
- extensibility via XBL
- flexible
- Applications
- application extensions
- even 'web' apps using Remote Xul.

Hello world

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/"
type="text/css"?>
<window
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/
there.is.only.xul">
<script>function hello() { alert('hello!'); } </script>
<box>
  <label>Click:
    <button
      label='Hello!'
      oncommand='hello()'/></label>
</box>
</window>
```

Why XUL?

- XML language
- DOM Scriptable via JavaScript
- Rich widgets like tabs, decks, trees
- Things that are very hard in DHTML, are very easy in XUL
- open, standards-base technology

Honesty: why *not* XUL

 - Only works in Firefox
 - not a media-rich solution a la Flash or (heh) SilverLight
 - the next Facebook will not be written in XUL

...But

XUL can make sense for some use cases:

 - either all your users are running FF anyway (ie web developers)
 - *or* your app is so amazingly cool that people will want to download FF just to use it.
 - *or* you just want a MySQL query interface in your Firefox-based IDE

The case study: XulDb

I thought George’s example of a SQL-server-like database / query interface had potential, so I decided to ‘finish it off’.

The basic approach:

- write a PHP back-end
- clean up George’s JS code
- learn more about Xul along the way

Disclaimers

 - I am not a rock star army of coding monkeys
 - I had some fun and did the easy part, but in order make XulDb into a useful app, I had to actually learn DOM
 - I'm lazy, and dom scripting is a lot of typing =)
 - If I wrote this, you can probably write something even cooler.

XulDb in three parts

PHP back-end for running queries against MySQL, using PDO

REST-ish end-point that parses GET urls and emits JSON-encoded data

JS scripting in XUL that communicates with the back-end and displays data or error as necessary

Progress

XulDb’s initial implementation went well:

- most of the JS and Xul code were there
- added authentication
- implemented the back-end
- got stuck with the DOM-oriented JS code
- left it for a while

ActiveState
Dynamic Tools for Dynamic Languages

Enter jQuery!

The 3 stages of learning JS

- hating it because you don't understand it.
- hating it because you thought you understood it, and that bit you in the ass.
- hating it because you do understand it, and it makes you feel stupid.

jQuery is the 4th stage

I love JavaScript because I use jQuery!
 - created originally by John Ressig
 - allows extremely powerful, compact JS development
 - features great DOM capabilities that translate seamlessly to XUL scripting

Simple Example

Instead of:

```
document.getElemenyById('foo').innerHTML = '<p>bar</p>';
```

You can do:

```
$('#foo').html('<p>bar</p>');
```

Simple Example 2

Instead of:

```
var items = document.getElementsByTagName('treeitem');
for(i in items) {
  items[i].yadayadayada();
}
```

You can do:

```
$('treeitem').each(function(i) { this.yadayada() })
```

XPath works too

You can do:

```
$(‘treeitem[@class=primary]’).each(function(i)
{ this.yadayada() });

$(‘div/p/strong’).html(‘some bold text for ya’);
```

...it has a ‘$’.

And lots more!

JavaScript Patterns

This should be familiar to AJAX hackers. Each action in the XulDb ui has a pair of functions:

1. a function that calls $.getJSON with the proper parameters
2. a function that acts as a callback, parses the response and then manipulates the ui accordingly

functional pair

```
function loadTables() {
  $.getJSON(uri, params, showTables(json);
}

function showTables(json) {
  // brilliant code here
}
```

JavaScript Patterns pt.2

The main difference with XUL is that there is no innerHTML property, so simplifying things with the AHAH method doesn't work. Thus the need for JSON in the first place.

All manipulation of the ui is done through the DOM! AAIIEEE!

JavaScript Patterns pt.3

Simple Example:

- the query has returned column data for the selected table, and we want to update the label tag to show the name of the table.

JavaScript Patterns pt.4

Straight Dom code:

```
document.getElementById(
  ‘current-table-label’)
  .setAttribute(‘value’, "Table: "+table);
```

jQuery Code:

```
$("#current-table-label").val("Table: "+table);
```

the magic of JSON

- JSON *is* JavaScript
- jQuery makes getting data from the server even easier with json:

```
$.getJSON(url, params, callback(json));
```

JSON magic in PHP

PHP 5.2 includes the json extension by default:

```
echo json_encode($output);
```

PEAR has Services_JSON or

Steal my code that punishes PHP without json installed.

XulDb and JSON

- all data is transmitted from PHP using JSON
- if the json extension is available, I use that. Users without the json pecl extension are punished with a much slower pure PHP version.

The Gnarly bit

The Xul interface was already written, the PHP was pretty simple, and jQuery is awesome for handling the communication with the back-end. Sounds pretty easy, right?

Building XUL Trees

XulDb required two pieces of code that took me a lot longer to write than they should have.
1. I had to build the DB => Table browsing tree on the left on the fly
2. I had to be able to build entire XUL trees on the fly with arbitrary, re-sizable columns and rows

Problem 1

Request the Databases and Tables in the database, then build a hierarchical tree that displays them.

The tree element and columns are already defined in Xul, I just need to build the treeitems.

Stage 1

1. When you log in, I get the databases and inject those into the tree:

```
cell_attr = {'label': dbs[db], 'src':icon};
node = newTreeNode({
  'container':'true',
  'id':'database-'+dbs[db]},
  cell_attr);
$('#dbChildren').append(node);
```

Stage 2

1. Loop through the tables returned and append them to the selected Database.

```
for(i in tables) {
  node = newTreeNode({
    'id': 'table-'+tables[i]},
    {'src':icon, 'label':tables[i]});
  $('#'+id).append(node);
}

$('#'+id).parent().attr('open', 'true');
```

Note

I *could* load all db and table information in on login (or shortly after) but I found this was less than snappy over an internet connection when dealing with databases with lots (hundreds => thousands) of tables.

Problem 2

Creating an entire tree object in JS and then injecting it into the DOM, based on the JSON returned from a Query.

Instead of hacking this out, I decided to just write a new Class called XULTree

XULTree

In the Ui code, handling the data is as simple as:

```
var tree = new XULTree();
$('#'+container_id)
  .empty()
  .append(tree.newTable(dtable, 'data-tab'));
```

The other Gnarly bit

XUL Trees base column spacing on the value of the ‘flex’ attribute. Having a decent-looking table required that I calculate flex values based on the column label:

```
XULTree.prototype._getColWeights = function(cols) {
    var lengths = new Array();
    for(c in cols) {
        lengths.push(new Array(cols[c].length, cols[c]));
    }
    return lengths;
}
```

It works!

Next steps:
 - code cleanup
 - better error handling
 - use a cookie to preserve UI state between refreshes
 - save query history? Datamodel?

XUL needs a framework

Or a jQuery extension?

Luckily for you, you can grab the source for XulDb from my svn and use my code:

http://canuckistani.ca/svnrepo/xul/xuldb

I could use some time to clean it up though...

Tools

1. Firefox plus Firebug:
 - view the XUL DOM change as your JS creates new nodes
 - inspect the DOM tree or run JS code from the console

2. Komodo: Komodo Edit is free (as in beer) and includes autocomplete support for XUL and JS in XUL files, as well as good PHP support.

Resources

Mozilla is dedicated to improving XUL and providing resources for XUl developers:

http://xulplanet.com
http://developer.mozilla.org

ActiveState

Dynamic Tools for Dynamic Languages

Questions?

Email:
jeffg@activestate.com
Slides:
http://blogs.activestate.com/jeffg