Server-Side JavaScript with jQuery and AOLserver

jQueryCamp07
October 27, 2007

Dossy Shiobara
dossy@panoptic.com

Why Use JavaScript on the Server

- Leverage web developer's mandatory skill-set
  - HTML, JavaScript
- Increase code reuse on client and server
- Lots of good documentation and examples
- Stable and mature implementations available

How to Use jQuery on the Server

- JavaScript interpreter
  - SpiderMonkey (in C), Rhino (in Java)
- John Resig's **env.js**
  - http://ejohn.org/blog/bringing-the-browser-to-the-server/
- Integrate into web server request processor

Our Example

- jQuery & **env.js**
- SpiderMonkey
- AOLserver
  - Tcl
  - nsjsapi module
  - tDOM

How AOLserver integrates SpiderMonkey

- nsjsapi module
  - Wraps SpiderMonkey with a “js” Tcl command
  - Exposes Tcl interp. as “tcl” JS function

```
nscp 1> set js [js create]
js0xad19ad8
nscp 2> js eval $js "new Date(0)"
Wed Dec 31 1969 19:00:00 GMT-0500 (EST)
nscp 3> js eval $js "tcl('info patchlevel')"
8.4.12
nscp 4> js destroy $js
```

How nsjsapi Loads env.js and jQuery

- **nsjsapi.js** provides JavaScript implementation of java.*, javax.* and org.w3c.dom.* APIs
- Uses Tcl's tDOM module for XML parsing
- Calls back into AOLserver/Tcl using the JS "tcl" function

nsjsapi.js Code Snippet

```
Packages.org.w3c.dom.Document = function(xml) {
  this._xml = xml;
  tcl('package require tdom');
  /* NB: Need a JS-to-Tcl stringifier!  This is unsafe: */
  this._dom = tcl('dom parse {' + xml + '}')[0];
};
Packages.org.w3c.dom.Document.prototype = {
  getElementsByTagName: function(name) {
    var nodes = [];
    var result = tcl(this._dom + ' getElementsByTagName {' + name + '}');
    if (result.length) {
      var domNodes = result[0].split(" ");
      for (var i = 0; i < domNodes.length; i++) {
        nodes.push(new Packages.org.w3c.dom.Node(domNodes[i], this));
      }
    }
    return new Packages.org.w3c.dom.NodeList(nodes);
  },
  getDocumentElement: function() {
    var result = tcl(this._dom + ' documentElement');
    if (result.length)
      return new Packages.org.w3c.dom.Node(result[0], this);
    else
```

Example

```
set js [js create]
js0x841d298
```

```
js load $js [ns_url2file nsjsapi/js/nsjsapi.js]
undefined
```

```
js load $js [ns_url2file nsjsapi/js/env.js]
undefined
```

```
js load $js [ns_url2file nsjsapi/js/jquery-1.2.1.js]
undefined
```

```
js eval $js {jQuery.fn.toString = DOMNodeList.prototype.toString; true;}
true
```

```
js eval $js {window.location = 'test/index.html';}
undefined
```

```
js eval $js {$('body').html()}
<div><span id='foo'>This is a test.</span></div>
```

```
js eval $js {$('div').append('Hello world!')}
[ <div> ]
```

```
js eval $js {$('body').html()}
<div><span id='foo'>This is a test.</span><span>Hello world!</span></div>
```

Challenges

- nsjsapi module still very incomplete
- No clear vision for or good examples of a server-side JavaScript web framework
- Hard to debug – lots of moving parts
  - AOLserver, nsjsapi, SpiderMonkey, nsjsapi.js, env.js, jQuery

TODO

- Automated test suite
- Simple web development framework
- Better documentation, with examples
- More evangelism

Questions?