Beyond Cookies:

Persistent Storage (and Offline Access) for AJAX Applications Using Dojo.Storage

Brad Neuberg
bkn3@columbia.edu
New Kind of Web

- Persistent client-side storage
  - Saving large amounts of data securely
- Offline access
- Works right now, with installed base
  - Cross browser and cross platform
New Kind of Web

What could you build?

- Web-based collaborative word processors
- Offline AJAX RSS readers
- Offline web book reader using Open Library
New Kind of Web

- Working on vision for years
- Many false starts and dead ends
- Thought it might not be possible
Vision is Real

- Exists right now
- Demo
Moxie

- Example web-based word processor
- Open source in Dojo repository
- Persistent client-side storage
  - Needs no server
- Offline access
- Works across big three
  - IE, Firefox, Safari
Demo

Moxie
Moxie

Put together in one day using Dojo:
- Dojo Widgets
- Dojo Events
- Dojo Storage
Agenda

- Dojo Storage
- Storage Providers
- Flash Storage Provider
- Building Sample Application - Moxie
- Dojo.flash
- How to do Offline
- Status
- Future
Acknowledgements

- Julien Couvreur
- The Dojo Team
What is Dojo Storage?

- Unified API to provide JavaScript applications with storage
- Open source
- Dojo Toolkit
- Automagic detection of available storage and environment
Relationship to AMASS

- Ajax MAssive Storage System
- Released in October, 2005
- Proof-of-concept prototype of Flash based storage
- Only worked on Firefox and IE
- Not well-tested
- Not integrated into Dojo or generic
Dojo Storage Architecture

JavaScript App/Web Page

Dojo Storage Manager

Storage Provider
Storage Provider API

- Generic API
- Gives hash table abstraction to underlying storage type
Storage Provider API

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize</td>
</tr>
<tr>
<td>isAvailable</td>
</tr>
<tr>
<td>isPermanent</td>
</tr>
<tr>
<td>getMaximumSize</td>
</tr>
<tr>
<td>hasSettingsUI</td>
</tr>
<tr>
<td>getType</td>
</tr>
<tr>
<td>put</td>
</tr>
<tr>
<td>get</td>
</tr>
<tr>
<td>hasKey</td>
</tr>
<tr>
<td>getKeys</td>
</tr>
<tr>
<td>clear</td>
</tr>
<tr>
<td>remove</td>
</tr>
<tr>
<td>showSettingsUI</td>
</tr>
<tr>
<td>hideSettingsUI</td>
</tr>
<tr>
<td>onHideSettingsUI</td>
</tr>
</tbody>
</table>
Possible Providers

- Cookie Storage Provider
- Flash Storage Provider
- ActiveX Storage Provider
- Form Save Storage Provider
- XPCOM Storage Provider
- More
Flash Storage Provider

- Currently only storage provider available
- Uses Flash 6+ features
- SharedObjects
- dojo.flash
Why Flash?

- Has greater installed base than Internet Explorer
  - Flash 6+ = 97.1%
  - Internet 5, 6, 7 = 64.7%
- Cross-browser and cross-platform
- Use as hidden runtime to extend browser
- This is what Flash Storage Provider does
### Storage Manager API

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
</tr>
<tr>
<td>setProvider</td>
</tr>
<tr>
<td>autodetect</td>
</tr>
<tr>
<td>isAvailable</td>
</tr>
<tr>
<td>isInitialized</td>
</tr>
<tr>
<td>supportsProvider</td>
</tr>
<tr>
<td>getProvider</td>
</tr>
</tbody>
</table>
Building Sample Application - Moxie

Don't have to know about FlashStorageProvider
Moxie's Architecture

- Two files - editor.html, editor.js
- Singleton JS object

```
Moxie

initialize
directoryChange
save
configure

save
_printAvailableKeys
_handlerLoad
_printStatus
```
Moxie's HTML (editor.html)

- **Dojo Editor Widget**
- **Dojo Widgets**
  - Easy, reusable DHTML components

```html
<div id="storageValue" dojoType="Editor"
  items="textGroup;|;blockGroup;|;
         justifyGroup;|;colorGroup;|;
         listGroup;|;indentGroup;|;
         linkGroup;">
  Click Here to Begin Editing
</div>
```
Import Dojo packages

```javascript
dojo.require("dojo.dom");
dojo.require("dojo.event.*");
dojo.require("dojo.html");
dojo.require("dojo.fx.*");
dojo.require("dojo.widget.Editor");
dojo.require("dojo.storage.*");
```
if(dojo.storage.manager.isInitialized() == false){
    dojo.event.connect(dojo.storage.manager, "loaded", Moxie, Moxie.initialize);
} else{
    dojo.event.connect(dojo, "loaded", Moxie, Moxie.initialize);
}
Loading Data

var results = dojo.storage.get(key);
try{
    dojo.storage.put(key, value, saveHandler);
}
catch(exp){
    alert(exp);
}

- Value can be string or JS object
  - Internal JSONing of all objects
- User might decline save request
Results Handler

Callback function with two arguments:

- status
  - dojo.storage.SUCCESS
  - dojo.storage.FAILED
  - dojo.storage.PENDING
- keyName
Results Handler

Moxie

*Web Editor With Persistent Client-Side Storage and Offline Access.*

Drag this link, [Run Moxie](#), to your favorites toolbar above for offline access. To work offline, on Internet Explorer and Firefox select File > Work Offline; Safari does not need this. Next, click the bookmark you made for this page.

**File Name:** testFile

**Load File:**

**Font:** Normal

**Size:** 1 (8 pt)

---

![Madromedia Flash Player Settings](#)

Madromedia Flash Player Settings

Local Storage

[local.com](#) is requesting permission to store information on your computer.

Requested up to 100 KB
Currently Used: 0 KB

[Allow](#)  [Deny](#)
Results Handler

Macromedia Flash Player Settings

Local Storage

brad.com is requesting permission to store information on your computer.

Requested: up to 100 KB
Currently Used: 0 KB

[Allow] [Deny]
Results Handler

```
var saveHandler = function(status, keyName){
  if(status == dojo.storage.PENDING)
      // ...
  else if(status == dojo.storage.FAILED)
      // ...
  else if(status == dojo.storage.SUCCESS){
      // ...
  }
```
Print Available Keys

var directory = dojo.byId("directory");

// clear out any old keys
directory.innerHTML = ""

// add new ones
var availableKeys = dojo.storage.getKeys();
for (var i = 0; i < availableKeys.length; i++) {
    var optionNode = document.createElement("option");
    optionNode.appendChild(document.createTextNode(availableKeys[i]));
    optionNode.value = availableKeys[i];
    directory.appendChild(optionNode);
}
Configure

Configuration button to control storage

Macromedia Flash Player Settings

Local Storage

How much information can brad.com store on your computer?

100 KB

Never Ask Again
Currently used: 62 KB

Close
if(dojo.storage.hasSettingsUI()){  
    var self = this;
    dojo.storage.onHideSettingsUI = function(){
        self._printAvailableKeys();
    }

    // show the dialog
    dojo.storage.showSettingsUI();
}
Flash Dialog + Rich Edit Control

- Firefox

- Flash dialog on top of rich text area
  - z-index issues

- When dialog is showing, hide rich edit area
if(status == dojo.storage.PENDING){
    if(dojo.render.html.moz){
        var storageValue = dojo.byId("storageValue");
        storageValue.style.display = "none";
    }

    return;
}
else{
    if(dojo.render.html.moz){
        var storageValue = dojo.byId("storageValue");
        storageValue.style.display = "block";
    }
}
Moxie - That's It

- A bit more code for some fancy status displaying
- Responding to mouse and keyboard events
Dojo Flash

- Cross-browser, fast, reliable JS+Flash communication is hard and ugly
- Encapsulates these details
Dojo.flash

Provides several major services:

- dojo.flash.Info
  - Is Flash available + what version of Flash?
- dojo.flash.Embed
  - Embeds Flash into page for Flash+JS communication
Dojo.flash

- Provides several major services:
  - `dojo.flash.Communicator`
    - Provides uniform, fast, reliable, JS + Flash communication
  - `dojo.flash.Install`
    - Uniform installation and upgrading of Flash
Dojo.flash.Communicator

- Very hard to create
- Where magic happens
Dojo.flash.Communicator

- Provides method abstraction between Flash + JS

  **JavaScript:**
  - `sayHello()` is Flash function
  - `dojo.flash.comm.sayHello();`

  **Flash:**
  - `DojoExternalInterface.call("dojo.storage.save", resultsHandler)`
DojoExternalInterface

- Backport of Flash 8 External Interface to Flash 6
- Callbacks are registered
DojoExternalInterface.initialize();
DojoExternalInterface.addCallback("put", this, put);
DojoExternalInterface.addCallback("get", this, get);
DojoExternalInterface.addCallback("remove", this, remove);
DojoExternalInterface.loaded();
Flash + JS Communication

Three ways:

1) LiveConnect/ActiveX + fscommands - Flash 6
   - Pro: Extremely fast, can send very large data, mature
   - Con: Only works on IE and Firefox
Flash + JS Communication

2) `ExternalInterface` - Flash 8
- **Pro**: Easy to use, Works on Safari
- **Con**: Unbelievably slow, performance degrades
  - $O(n^2)$, serious serialization bugs
Flash + JS Communication

3) `getURL/LocalConnection/New Flash Applets` - Flash 7

- **Pro:** Very cross platform
- **Cons:** Destroys history, serious data size limitations and performance issues
Flash + JS Communication

- Only mechanisms 1 and 2 are acceptable
- Use LiveConnect/ActiveX + fscommands for IE/Firefox - Flash 6 communication
- Use ExternalInterface for Safari - Flash 8 communication
  - Find workarounds to fix performance and serialization bugs
  - Performance workarounds only work in Safari
Flash 8 Communication

- Needed for Safari
- 3 months to finish
- Safari Support = 30% more time to any project
Flash 8 Communication

- Performance/Serializing issues
- Workarounds:
  - Chunk data into many different small calls through ExternalInterface
    - Makes performance linear
Flash 8 Communication

- Used debugger to find hidden JS serialization methods used by Flash plugin
  - Internal XML serialization - doesn't escape characters
  - Uses eval()
  - Found way to bypass and do it all manually
Example bypass code:

```javascript
plugin.CallFunction(
    '<invoke name="chunkArgumentData" ' 
    + 'returntype="javascript">' 
    + '<arguments>' 
        + '<string>' 
            + piece 
        + '</string>' 
        + '<number>' 
            + argIndex 
        + '</number>' 
    + '</arguments>' 
    + '</invoke>');</
```
Flash 8 Communication

Workarounds:

- Double encode and decode all XML characters on both sides:
  - &amp; --> &amp;&amp;
- Very important for persisting XML
  - Lots of testing
Flash 8 Communication

- With workarounds, performance and reliability are great
- Without them, simply not realistic to use ExternalInterface
- Only works on Safari
Flash 8 Communication Demo

- Saving book used to take minutes - now takes seconds
- XML impossible before
- Demo of saving XML and large book with Safari
Flash 6 Communication

- Little more straightforward

JS -> Flash:
- JS uses plugin.SetVariable to build up method values
- Calls plugin.Play() to execute
- Flash reads values off _root

Flash -> JS:
- fscommand
Flash 6 Communication

Things that were hard to figure out:

- Sensitive to way you embed Flash into page
- Robustly handle different timing issues

For example, on some IE versions, Flash can start running before fscommands will work
Other Areas

- Showing Flash settings dialog and knowing when closed
- Centering across different HTML DOCTYPEs and browsers
Other Areas

- Serializing and deserializing JavaScript objects as JSON strings
- Short-circuiting JSON eval() if dealing with large strings
  - Drastically better performance
Other Areas

- Build system to easily create Flash 6 and Flash 8 versions of ActionScript files
  - Ant task - buildDojoFlash
- Reliably determining Flash version installed
- Automated Flash installation and upgrading
Dojo Flash

- Lots of QA testing
- Poor mans QA/distributed QA:
  - Go to copy shops like Kinkos
  - Spend lots of money to use rental machines
  - Beg people at coffee shops
Dojo Flash

- I experienced pain so you don't have to
- Dojo.flash externally is easy to use
- Internally was hell to create
How to do Offline

Julien Couvreur discovered offline mechanism

- Figured out HTTP response headers
- TiwyWiki - Take It With You Wiki
- Blog
  - http://blog.monstuff.com
How to do Offline

- HTTP Caching
- Magic happens on server-side

Send over HTTP response headers:
  - Etag
  - Last-Modified
  - Expires
  - Cache-Control
How to do Offline

- Etag and Last-Modified on by default in Apache 2
How to do Offline

Expires and Cache-Control must be turned on in httpd.conf:

- mod_expires

LoadModule expires_module modules/mod_expires.so

<Directory "c:/dev/dojo/">
  ExpiresActive On
  ExpiresDefault "access plus 1 month"
</Directory>
How to do Offline

- The page is now in browser cache after first access
- In IE and Firefox, go to File > Work Offline
  - Just navigate to URL
- In Safari, just go to URL
- Provide link to drag to toolbar
How to do Offline

- Depends on persistent storage
- Even if offline, still have access to data
- When network appears, just sync using persistent cache
How to do Offline

- Simple

- Issue:
  - If user clears cache, UI is gone
  - If user has not visited site recently, not in cache

- Can live with issues

- In practice works well for commonly used web apps
Status

- Dojo.storage is in beta and in Dojo repository
- Will be bundled with next release of Dojo
  – http://dojotoolkit.org
- Has had lots of QA testing and is stable
Status

- Download Dojo 0.3 (hot off the presses!)
  - http://blog.dojotoolkit.org

- Moxie:
  - http://codinginparadise.org/e

- Test Storage UI:
  - http://codinginparadise.org/x
Status

Offline mechanisms have had less QA
– Need more QA testing
Future

- Bring dojo.storage to full release after wide stress testing by community
- Create zoo of storage providers
Future

- Dojo.offline and dojo.sync
  - Truly advanced offline abilities
    - client-server syncing
  - More offline QA
Advanced collaborative tools that simply work, right in the browser

Use dojo.flash to get access to "Flash runtime"

- audio and video conferencing with webcams
  - GTalk right in the browser
  - Wikis + webcams, no downloads
- Streaming sockets and multiplexing
  - Browser-Based SubEthaEdit
Future

Paper Airplane - 2003

- Research mockup and prototypes of deeply collaborative web browser
- Pieces can be done in AJAX/DHTML
  - Don't need new browser
- http://codinginparadise.org/paperairplane
Beyond Cookies:
Persistent Storage for AJAX Applications Using Dojo.Storage

Brad Neuberg
bkn3@columbia.edu