

Extending Zotero Translators and Plug-Ins

Simon Kornblith
February 24, 2008
code4lib

Sequence of Translation

- Check to see whether URL matches a regular expression
- If regular expression matches, execute a JavaScript function to determine whether translator can translate page
- Upon request, execute another JavaScript function to translate page

Introduction to Scaffold

Translator Detection



- Regular expression specified in Scaffold “Target” field
- `detectWeb(doc, url)` in “Detect Code” called
 - `doc` provides (mostly) complete access to page DOM
 - `url` is page’s URL (including proxy translation)
 - Return status indicates type of page to be translated

XPath

- Convenient way of addressing fragments of an XML (or HTML) document
- Not necessary to write a translator, but very useful
- MIT SIMILE project's Solvent greatly simplifies generation of XPaths
- More documentation at MDC

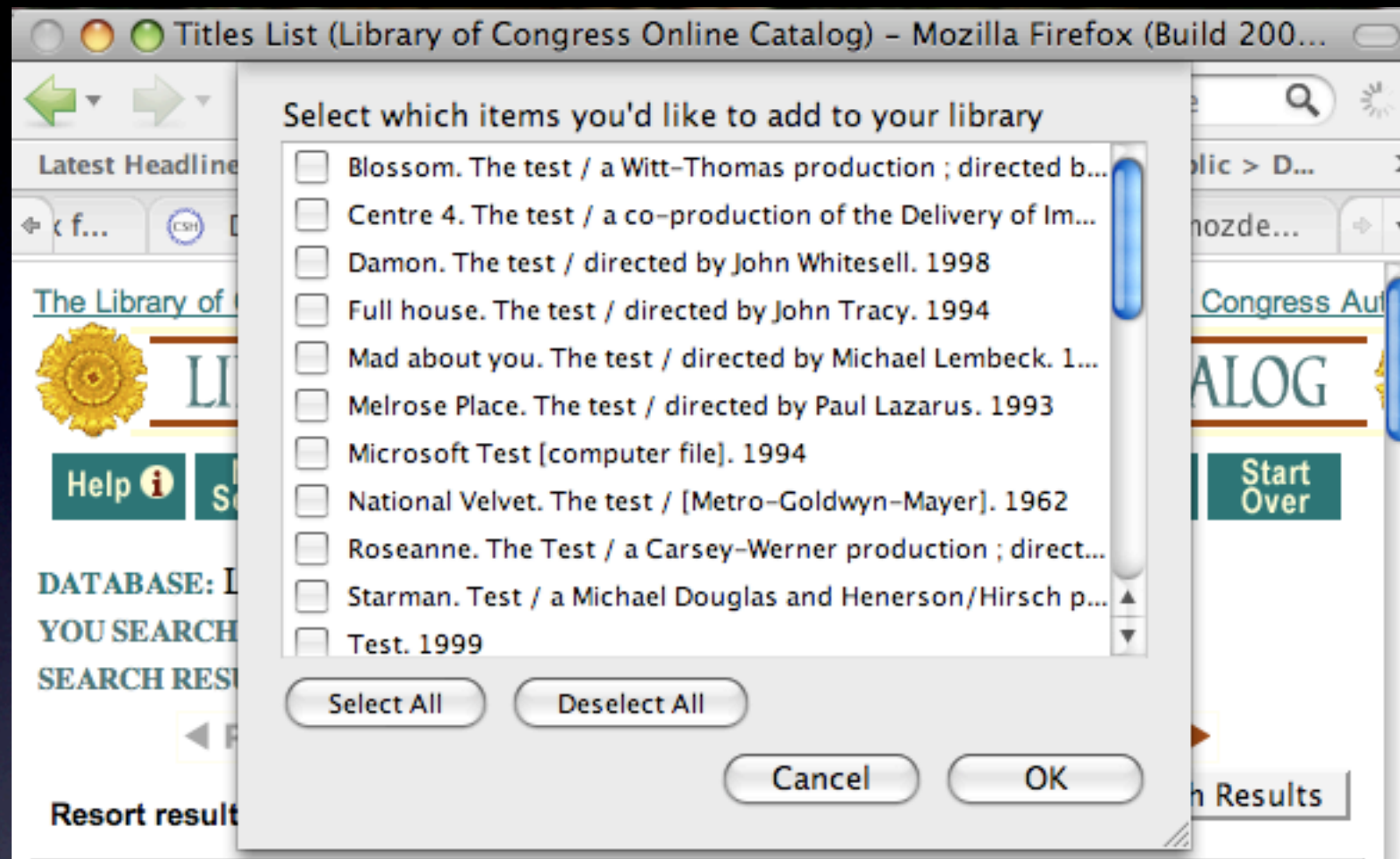
```
var nsResolver = doc.createNSResolver(doc.documentElement);
var IDs = doc.evaluate(' //input[@name="id"] ',
    doc, nsResolver, XPathResult.ANY_TYPE, null);
var firstID = IDs.iterateNext();    // a regular DOM node
```


Detect Code Demo

Translation

- `doWeb(doc, url)` in “Code” called when user clicks button to save item to library
- Numerous Zotero library functions for cleaning data (see source for `zotero/chrome/content/xpcom/utilities.js`)
- Can call other translators and use them to process data retrieved from a site
- Can load and extract data from other pages

Selecting Multiple Items



- `zotero.selectItems(items)` presents a dialog with a list of items that can be selected for saving
- `items` is a JS object of value \Rightarrow label pairs
- Returns items user selected, or `false` if user cancelled

Loading Other Pages

- `Zotero.Utilities.HTTP.doGet(urls, loadFunction, doneFunction)`
 - Does an asynchronous GET request using XMLHttpRequest
 - Returns text retrieved
- `Zotero.Utilities.HTTP.doPost(url, postString, doneFunction)`
 - Does an asynchronous POST request
- `Zotero.Utilities.processDocuments(urls, loadFunction, doneFunction)`
 - Loads full page DOM
 - Slower and cannot access pages across domains due to limitations of Firefox sandbox

Asynchronous Operation

- `processDocuments`, `doGet`, and `doPost` calls are asynchronous
- Call `zotero.wait()` before end of synchronous operation
- Call `zotero.done()` when asynchronous operation is complete
- `Zotero.Utilities.processDocuments(urls, loadFunction, function() { Zotero.done(); })`

Calling Other Translators

- Zotero includes import translators for MARC, RIS, BibTeX, Dublin Core RDF, and several other formats
- `zotero.loadTranslator("import")` generates new instance of translation interface
- `translator.setTranslator()` sets the translator to use
 - Specify translator GUID (use Scaffold or scrapers.sql)
- `translator.setString()` sets the data to be translated
- `translator.translate()` translates items and saves to database
- Can use `translator.setHandler()` to intercept items before they are saved to Zotero database

Saving Items Directly

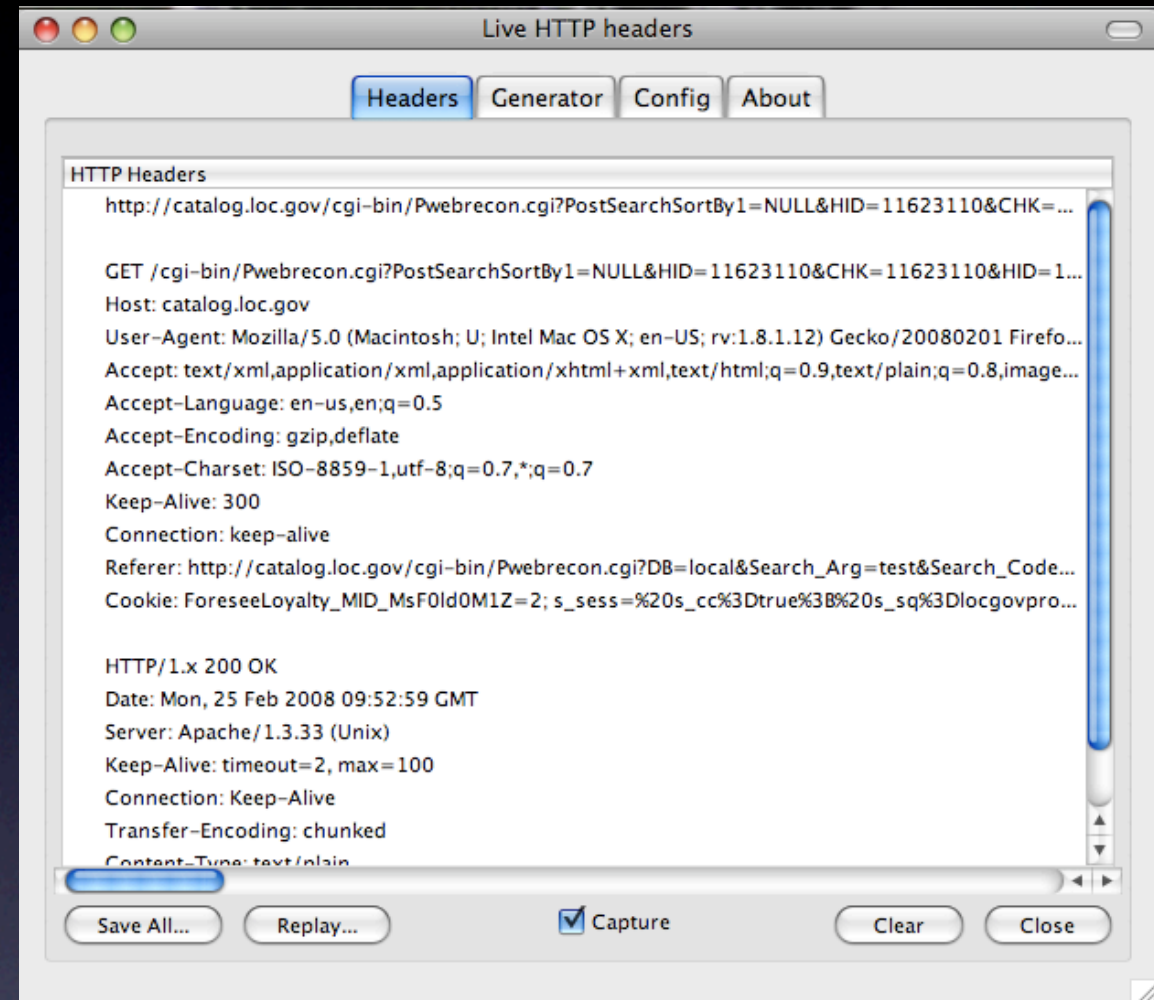
- `Zotero.Item()` object
- Object properties set item properties
 - `title`, `date`, `publisher`, `item`, `volume`, `ISBN`, `pages` (documented online)
- `creators` property holds item creators
 - `item.creators.push({firstName:"First", lastName:"last", creatorType:"editor"});`
- `attachments` property holds attached links and snapshots
 - `item.attachments.push({title:"Page", snapshot:false, url:url});`
- `item.complete()` saves item to Zotero database
- Can also alter items returned from import translators using `setHandler("itemDone", function(obj, item) {})` to intercept items before they are saved

Translator Demo

Additional Considerations

Reverse Engineering

- Often, writing a translator requires determining the HTTP GET/POST requests required to retrieve a document containing citation information
- Two good ways
 - Look at page source (<form> and <input> tags)
 - LiveHTTPHeaders Firefox extension



```
.R>  
INPUT TYPE=HIDDEN NAME=HID VALUE=11690416>  
INPUT TYPE=CHECKBOX NAME=CHK VALUE=11690416  
</input>
```


Common Translator Styles

- Library of Congress is easy because from search page, we can get all of the citations with one request
- Other sites require that we load each page individually (often using `Zotero.Utilities.processDocuments`)
- Some sites don't provide citations in a format for which Zotero has an import translator
- Need to add items directly to Zotero database

Debugging

- Can't use Venkman (Mozilla JavaScript debugger) on translators
- Doesn't support code in `eval()`
- `zotero.debug()` function can pretty-print any variable (including nested objects) to Scaffold output or system console

Proxy Servers

- VPN and non-URL-rewriting proxies present no issues
- Zotero can recognize when a user accesses an EZProxy system through a library website
 - Maps proxy domain to site domain and passes to detectWeb as url
 - Cannot recognize when a user travels from one EZProxy-based database to another
 - Does not work with proxies from other vendors
 - Improved proxy support in Zotero 1.5
- Regular expressions should be designed to maximize accessibility through proxies
 - `https?://[^\.]*example\.com[^\.]*/xyz` versus
`^https://example\.com/xyz`

Plug-Ins

Zotero Plug-Ins

- Firefox extensions can access Zotero object and all its methods
- This workshop will provide a basic introduction to the methods most useful to plug-in developers
 - Adding new items
 - Registering handlers to respond upon item creation, modification, and deletion

Firefox Extensions

- Zotero plug-ins are packaged as Firefox extensions
- Need 2 files
 - chrome.manifest
 - Tells Firefox where included files are
 - Registers “overlays” to patch existing XUL
 - install.rdf
 - Package metadata in RDF format
- Need chrome directory with content, locale, skin subdirectories

XUL Overlays

- XML User Interface Language
- All of Firefox's user interface is written with XUL
- A plug-in may patch any piece of Firefox or Zotero's interface with a XUL overlay to add additional menu items, toolbar items, etc.
- Most commonly, a plug-in will want an overlay to alter Zotero actions menu (cog)
- Might also want to add a pane to Zotero preferences
- Add `<script src="chrome://zotero/content/include.js"/>` to overlay to access Zotero object and its methods

XUL Overlay Demo

Plug-In Data Storage

- Mozilla preferences
 - Pros: easy
 - Cons: can't store much data
- Mozilla RDF data source
 - Pros: powerful, uses RDF
 - Cons: slow, interface can be confusing, doesn't directly mesh with Zotero DB paradigms, Mozilla-generated RDF/XML looks like crap
- Zotero SQLite interface
 - Pros: easy (if you know SQL), very powerful, uses SQL
 - Cons: some SQLite corruption issues with large transactions in Firefox 2.0 (fixed in 3.0)

Harnessing SQL

- Zotero abstracts Firefox's built-in SQLite database layer to allow easy access to SQL
- `var db = new Zotero.DBConnection('helloworld');`
 - Loads or creates a SQLite database file in the Zotero directory and returns a database connection object
- `db.tableExists("table")`
 - Checks existence of a table
 - Allows creation of DB structure on first run
- `db.query("SELECT * FROM table")`
 - Runs an SQL query (in this case, a `SELECT` query)
 - Returns results as an array of objects indexed by column name
 - `[{name:"Bob", occupation:"Builder"}, {name:"Fred", occupation:"Fireman"}]`

SQL Demo

Handling Notifier Events

- Registering a function to receive notifier events is easy
 - `var notifierID = Zotero.Notifier.registerObserver(this.notifierCallback, ['item']);`
- Don't forget to unregister notifier
 - `Zotero.Notifier.unregisterObserver(notifierID);`
- Notifier passes four arguments to handler function
 - `event`
 - `"add"` (any)
 - `"modify"` (collection, search, item, tag)
 - `"delete"` (collection, search, item, tag)
 - `"remove"` (collection-item, item-tag)
 - `"move"` (collection)
 - `type: "collection", "search", "item", "collection-item", "item-tag", "tag"`
 - `ids`: IDs of changed items, collections, tags, etc. (item-tag and collection-item give both separated by a hyphen)
 - `extraData`: `"old"` field on item contains old item

Getting Existing Items

- Use `Zotero.Items.get()` to get item or items by their IDs (returned from the notifier)
- Methods for returned items documented in `chrome/content/zotero/xpcom/data_access.js`
 - `getField("fieldName")` method returns the contents of an item field
- Can also use `toArray()` method to get item in the format used in translators (but this is slightly inefficient)

Notifier Demo

Adding New Items

- Unfortunately, a completely different interface than translators use
- Easy way
 - ```
var data = {
 title: "Shakespeare: The Invention of the Human",
 publisher: "Riverhead Hardcover",
 date: '1998-10-26',
 ISBN: 1573221201,
 pages: 745,
 creators: [
 ['Harold', 'Bloom', 'author']
]
};
var item = Zotero.Items.add('book', data);
```
- Can also use `setField('field', 'value')` and other methods (described in `data_access.js`)



# Adding Items Demo



# Other Possibilities

- Can access any Zotero interface, not just interfaces for adding, removing, and modifying items
- `Zotero.Translate()` allows plug-ins to access import and export functions (see `translate.js`)
- `Zotero.Cite()` allows access to citation interface (see `cite.js`)