Web Services for Libraries: Tips, Code Samples, Explanations, and Downloads

Jason A. Clark
Head of Digital Access and Web Services
Montana State University Libraries
jaclark@montana.edu
twitter.com/jaclark

Tips – Getting Started with Web Services

• Play in the sandbox – pick a service, study it
• Yahoo Query Language: http://developer.yahoo.com/yql/
• Yahoo Developer Central: http://developer.yahoo.com/
• Amazon Web Services Developer Connection: http://developer.amazonwebservices.com/connect/
• Google Code: http://code.google.com/

Tips – Consuming Web Services

• Pick a language or parsing tool
• Find a few data sources (APIs) worth learning about
• Make some requests and look at code in your browser
• Think about added value, more efficient workflows
• Browse around – many language libraries are already written

Tips – Building Web Services

• URIs are your friends – that's your interface
• Use simple CRUD (Create, Read, Update, Delete) functions over HTTP (Get, Delete, Put, Post)
• Keep verbs in API protocol intuitive and memorable
• Start small – simple, read-only requests
• Roll it out, beta version – once it's public you are restricted

Tips – Web Services Data Sources

• AllCDCovers.com http://www.allcdcovers.com/api
• ISBNdb.com http://isbndb.com/docs/api/index.html
• OpenDOAR http://www.opendoar.org/tools/api.html
• arXiv.org http://export.arxiv.org/api_help/
• Google Book Search APIs http://code.google.com/apis/books/
• LibraryThing APIs http://www.librarything.com/services/
• WorldCat Search API http://worldcat.org/devnet/wiki/SearchAPIDetails
• iTunes and App Store API http://www.apple.com/itunes/affiliates/resources/documentation/itunes-store-web-service-search-api.html
* See programmableweb http://www.programmableweb.com/apis/directory

Resources and Tools

• New York Times API Tool http://prototype.nytimes.com/gst/apitool/index.html
• YouTube Data API scratchpad http://stage.gdata.youtube.com/demo/index.html
• Google Code Playground http://code.google.com/apis/ajax/playground/
• Yahoo Pipes http://pipes.yahoo.com/pipes/
• Google Chart Wizard http://code.google.com/apis/chart/docs/chart_wizard.html
• Yahoo Query Language Console http://developer.yahoo.com/yql/console/

Web Services – Sample Applications

• Google Ajax Search API - Federate search of Google Data
• Flickr API - Display Photos (JSON)
* View samples and download code at http://www.lib.montana.edu/~jason/files.php

Web Services – Building Blocks

1. REQUEST – learn the protocol, ask for the data
2. RESPONSE – receive the data
3. PARSE – pick the pieces you need
4. DISPLAY – format those pieces for display

Code Sample #1: Google Ajax Search API – Javascript and CSS

xHTML source:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link href="http://www.google.com/uds/css/gsearch.css" type="text/css"
rel="stylesheet"/>
<style type="text/css">
body {background-color:white;color:black;font-family:Arial,sans-serif;font-
size:small;margin:15px;}
.gsc-control {width:400px;}
</style>
<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0" type="text/
javascript"></script>
...
```

Javascript source:

(http://www.google.com/uds/api?file=uds.js&v=1.0)

```
...
if (window['google'] != undefined && window['google']['loader'] != undefined) {
if (!window['google']['search']) {
window['google']['search'] = {};
google.search.CurrentLocale = 'en';
google.search.ShortDatePattern = 'MDY';
google.search.Version = '1.0';
google.search.NoOldNames = false;
google.search.JSHash = 'b2cf21b87d5348acb0a314b08588b757';
google.loader.ApiKey = 'notsupplied';
google.loader.KeyVerified = true;
google.loader.LoadFailure = false;
}
google.loader.writeLoadTag("script", google.loader.ServiceBase + "/api/search/1.0/en/
b2cf21b87d5348acb0a314b08588b757/default.I.js", false);
```

Code Sample #1: Google Ajax Search API - Explanation

- Javascript written by Google – heavy lifting
http://www.google.com/uds/api?file=uds.js&v=1.0
- "google.loader.writeLoadTag" – tells API to run, sets possibilities for search API
- CSS written by Google – formatting and display
http://www.google.com/uds/css/gsearch.css
- Understand these files, but you probably want to leave them as is – code library

Code Sample #2: Google Ajax Search API - Web page for user interface and display

```
…
<script type="text/javascript">
//<![CDATA[

function OnLoad() {
// Create a search control
var searchControl = new GSearchControl();

// Add in a full set of searchers
var localSearch = new GlocalSearch();
searchControl.addSearcher(localSearch);
searchControl.addSearcher(new GwebSearch());
searchControl.addSearcher(new GvideoSearch());
searchControl.addSearcher(new GblogSearch());
searchControl.addSearcher(new GnewsSearch());
searchControl.addSearcher(new GimageSearch());
searchControl.addSearcher(new GbookSearch());

// Set the Local Search center point
localSearch.setCenterPoint("Bozeman, MT");

// tell the searcher to draw itself and tell it where to attach
searchControl.draw(document.getElementById("searchcontrol"));

// execute an inital search
searchControl.execute("library books");
}
GSearch.setOnLoadCallback(OnLoad);
//]]
</script>
<div id="searchcontrol">Loading</div>
```

Code Sample #2: Google Ajax Search API - Explanation

• xHTML and javascript that gives action to our script
• Create the interface (GUI) controls with "var searchControl = new GSearchControl();"
• Set the local search parameter with "localSearch.setCenterPoint"
• Set the initial query with "searchControl.execute"
• Decide which pieces of Google data to federate with "searchControl.addSearcher"
• will be populated with script messages OR generated xHTML tags received via our Ajax requests
• Any customization begins with these parsing and display functions

Code Sample #1: Flickr API - Display Photos (JSON) – The URL Request

http://api.flickr.com/services/feeds/photos_public.gne?tags=cil2008&format=json

Code Sample #1: Flickr API - Display Photos (JSON) - Explanation

• HTTP Request to Flickr API
http://www.flickr.com/services/api/
• API provides data as XML feeds (RSS, ATOM)
• Requesting "/feeds/" with a "format" of JSON (Javascript Object Notation)
• Querying API for all public photos tagged "cil2008" with the "tags" parameter

Code Sample #2: Flickr API - Display Photos (JSON) – The URL Request in Javascript

```
<!-- use script tag to make request to flickr api, specify json format and tag to search -->
<script type="text/javascript" src="http://api.flickr.com/services/feeds/
photos_public.gne?tags=cil2008&format=json">
</script>
```

Code Sample #2: Flickr API - Display Photos (JSON) - Explanation

• JSON is actually javascript and to make JSON output available we must call it on the page via the <script> tag
• After <script> tag is run, JSON output exists as javascript object ready to be parsed

Code Sample #3: Flickr API - Display Photos (JSON) – JSON Response


```
jsonFlickrFeed({
"title": "Photos from everyone tagged cil2008",
"link": "http://www.flickr.com/photos/tags/cil2008/",
"description": "",
"modified": "2008-04-07T18:43:16Z",
"generator": "http://www.flickr.com/",
"items":
[
{
"title": "So many floors",
"link": "http://www.flickr.com/photos/nengard/2395908509/",
"media": {"m":"http://farm4.static.flickr.com/3182/2395908509_d6452e2d56_m.jpg"},
"date_taken": "2008-04-07T13:07:53-08:00",
"description": "So many floors",
"published": "2008-04-07T18:43:16Z",
"author": "nobody@flickr.com (nengard)",
"author_id": "10137764@N00",
"tags": "hyatt cil2008 cil08"
},
...

]
})
```


Code Sample #3: Flickr API - Display Photos (JSON) - Explanation

- More structured data ready to be parsed
- We'll extract the values and format for display using the second javascript

Code Sample #4: Flickr API - Display Photos (JSON) – Parse and display with Javascript

```
<script type="text/javascript">
//run function to parse json response, grab title, link, and media values - place in html tags
function jsonFlickrFeed(fr) {
for (var i = 0; i < fr.items.length;i++) {
document.write('<a title="' + fr.items[i].title + '" href="' + fr.items[i].link + '"><img src="'
+ fr.items[i].media.m + '" alt="' + fr.items[i].title + '"></a>');
}
}
</script>
```

Code Sample #4: Flickr API - Display Photos (JSON) - Explanation

• Create javascript function “jsonFlickrFeed” to parse JSON response returned from first javascript
• Loop statement: “for (var i = 0; i < fr.items.length;i++)” runs through all JSON data nodes
• ‘‘document.write” – native javascript function prints out values from JSON in xHTML markup

Final Thoughts

• Start with simpler data formats – RSS and ATOM are well-supported
• Keep experimenting and learning with a single web service, become a seasoned veteran
• Remember the primary actions for using web services: request, response, parse, display
* Translate these actions into your favorite tool or scripting language