CHAPTER 7

Other XML Web Services APIs

In Chapter 6, we examined the Flickr API in great detail. Now we turn to other APIs that are based on XML web services. I will show you how these APIs are both similar and different from Flickr's as a way of jumpstarting your use of these various APIs. We want to have a broad view and dive into some specific APIs. Broadly speaking, there are the following (not-strictly orthogonal) types of APIs.

* REST
* SOAP
* Javascript
* desktop APIs
* plugin frameworks (e.g., browser plugins, Office programming, COM, Applescript)

We focus on the first two in this chapter and hint at the others. Even with REST and SOAP, we can't possibly explain all the APIs out there. I will outline a general approach for getting up to speed with REST and SOAP-based APIs that will get you started with most APIs. I will use popular APIs and/or ones that come from specific fields or ones that have a special feature that illustrates stuff you should know about.

We survey the types of services available and how to think about the sheer range of APIs. (New APIs will crop up as time goes on but I can recommend the categories at http://programmableweb.com/apilist/bycat as a way to think about the types of APIs out there.)

This chapter looks at sites such as programmableweb.com that documents these various APIs, the challenges faced in doing so. What framework is there to understand their commonalities and differences? WSDL for SOAP but what's the equivalent for other XML web services?

This covers the public internet -- not web services that are not accessible widely. Also, we won't have coverage of lots of field specific stuff -- the APIs here are very much about the popular Web 2.0 space...although I'll hint at other stuff.

What's out there in terms of APIs

This section outlines the broad spectrum of APIs out there.

Overall approach to learning Web services APIs.

Say you want to program/extend/customize an application -- what are some general approaches to take? Recalling the theme from the first section of the book:

* you need to learning about the app as an end-user
* study how people have adapted, mashed up, or remixed the application already to get a sense of the mashability of the app.
* figure how do the URLs work for the application
* look for RSS feeds that the application provides
* look for ways to export or import data with the application

Finding information about an API

Some general approaches:

* Type into Google or your favorite search engine, the name of the app + "API" to see whether there is an API and the documentation for it. Also consult directories such as http://programmableweb.com
* When you are reading the documentation, see how much can you can leverage of what we've learned so far (which is about looking at apps as end-user, looking for mashup opportunities, and now using the Flickr APIs (at least the REST interface)

Using programmableweb.com to learn about APIs

Programmableweb.com, run by Jon Musser, is an excellent resource for learning about what APIs are available, the basic parameters around the APIs, and the mashups that use any given API. Noteworthy parts:

* `http://www.programmableweb.com/apis` is the "API Dashboard" lists the newest APIs to be registered and the most popular APIs for mashups over the last 14 days.
* `http://www.programmableweb.com/apilist/bycat` lists APIs by categories. Understanding the various categories that have emerged is helpful to understanding what fields of endeavours for which people are making APIs).
* I like looking at `http://www.programmableweb.com/apilist/bymashups` to see which is the most popular APIs by the number of mashups that use the API. Surveying the top 10 or 20 and getting a good handle on the field.

In his database, John Musser is trying to make regular what is not regular. WADL would help in this situation, if everybody who provides rest service would write a corresponding WADL file.

SCREEN-SCRAPING PROGRAMMABLEWEB.COM

There is a tremendous weath of data on programmableweb.com for getting oriented with specific APIs as well as the mashups that use the APIs. Since programmableweb.com does not itself currently support a public API., one can set out to mine the site's data by screenscraping the site. Some qestions that can be answered are:

* How many APIs support REST vs SOAP?

* Which APIs use WSDL?
* What's the distribution of the number of mashups that combine a given number of APIs?

End Sidebar

How to think about APIs: Flickr Redux

There are no standard taxonomy for APIs. I'd say programmableweb.com is the closest to the directory of public web services. We can use Flickr as a guide -- and as we work through specific examples, other categories that emerge to help classify APIs

* encoding
* user authentication
* request format: rest vs soap vs xml-rpc
* response format: XML, json,....
* the use of API keys
* terms of service
* URLs -- what's the little language of that application?
* community -- what mailing lists and forums exist around the API.
* special calls (e.g. uploading)
* API kits
* documentation of methods
* Is there an equivalent to the Flickr API Explorer?
* error handling

REST-like/RESTful/POX over HTTP

You will hear the term REST to describe the most common form of XML-based APIs. When we have talked about REST, we mean "RESTful" or REST-like or POX over HTTP -- not precisely Roy Fielding's technical definition. By RESTful, I mean the many APIs out there for which where you can build a URL -- mostly with parameters-- , do mostly HTTP GET (and sometime HTTP POST) and get back XML (and maybe JSON and other formats) Technically RPC-style SOAP follows this pattern and XML-RPC also -- but the usual way to get at those APIs is to think of them as SOAP and XML-RPC and treat them accordingly. (See the sidebar to figure out the intricacies of REST vs POX over HTTP).

The good news is that while people are trying to get to the bottom of this debate, you can do a lot with RESTful APIs if you know a few technical tricks -- specifically a bit about how encode your URLs properly and how to use curl to invoke URLs.

MAKING PRECISE DISTINCTIONS AMONG REST, POX OVER HTTP

Arguably, the Flickr API is not exactly REST[1] although it's not clear how important that distinction really is.[2]

Making the fine distinctions is a complicated business. To get a feel of the issues, take a look at:

* http://www.crummy.com/2006/11/05/2
* http://www.intertwingly.net/blog/2006/11/05/POX-and-SOAP
* http://www.trachtenberg.com/blog/2006/11/06/rest-vs-httppox-vs-soap/
* http://www.1060.org/blogxter/entry?publicid=9FFBC835E3D9904A288982A638A8F45B

End Sidebar

URL encoding

At first glance, you can make requests by simply forming the following URL:

```
[baseURL]?p1=v1&p2=v2 ....
```

where px,vx are parameter/value pairs.

However, when you use characters which are not part of the allowed set, you need to encode the characters. The canonical(?) discussion of URL encoding issues is RFC 3986.[3] (though there is important history in RFC 1738.[4]

Language specific techniques for doing this encoding:

* PHP: urlencode()[5] and urldecode -- specifically urlencode(utf8_decode('enquête') to handle unicode issues.

Using cURL

cURL is an incredibly useful utility for invoking XML web services:[6]

> *curl is a command line tool for transferring files with URL syntax, supporting FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, FILE and LDAP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading,*

---

[1] http://gonze.com/weblog/index.cgi/2004/08/24#8-24-4: " I, REST accolyte, must point out that Flickr's REST interface is not REST, because it specifies the method in the arguments rather than using the HTTP method, and because it specifies the return status within the response body rather than using the HTTP status code. REST design is about transferring documents, not invoking procedure calls."

[2] http://korrespondence.blogspot.com/2004/08/flickr-rest-service-not-so-restful.html -- Stewart Butterfield, the co-founder of Flickr doesn't seem to.

[3] http://tools.ietf.org/html/rfc3986

[4] http://tools.ietf.org/html/rfc1738

[5] http://www.php.net/urlencode

[6] http://curl.haxx.se/

> *HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and a busload of other useful tricks.*

Documentation:

* http://curl.haxx.se/docs/manpage.html is the man page
* http://curl.haxx.se/docs/httpscripting.html is the most helpful page in many ways because it gives concrete examples.

A few sample invocations of curl

Search for flower-tagged photos in Flickr:

Flickr REST call:

```
http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8102a5160154ef4662adcc9046b&tags=flower
```

You do the same in curl:

curl [url]

```
curl "http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=95cd140d37a9fd0dd414cdbe589e32ee&tags=flower"
```

Note the quotes around the URL -- crucial to keep the special characters in the URL from being incorrectly interpreted by the shell.

Let's do a POST -- using the Yahoo term extraction service. It happens that the GET works:

```
http://search.yahooapis.com/ContentAnalysisService/V1/termExtraction?appid=YahooDemo&context=Italian+sculptors+and+painters+of+the+renaissance+favored+the+Virgin+Mary+for+inspiration.&query=madonna
```

```
curl "http://search.yahooapis.com/ContentAnalysisService/V1/termExtraction?appid=YahooDemo&context=Italian+sculptors+and+painters+of+the+renaissance+favored+the+Virgin+Mary+for+inspiration.&query=madonna"
```

Let's do the POST:

```
curl -d "appid=YahooDemo&context=Italian+sculptors+and+painters+of+the+renaissance+favored+the+Virgin+Mary+for+inspiration.&query=madonna" http://search.yahooapis.com/ContentAnalysisService/V1/termExtraction
```

Examples of RESTful services

Let's generate a list of the top 20 APIs on ProgrammableWeb by mashupup count and also list the type of protocols supported by the API.

Top 20 APIs by mashup count (as March 11, 2007)[7]

API Name	Number of Mashups	Protocols Support
Google Maps	857	REST, Javascript
Flickr	186	REST, SOAP, XML-RPC
Amazon E-commerce	140	REST , SOAP
YouTube	93	REST , XML-RPC
Yahoo Maps	75	REST
411Sync	74	RSS input over HTTP
del.icio.us	73	REST
eBay	71	SOAP , REST
Yahoo Search	63	REST
Microsoft Virtual Earth	58	**Javascript**
Yahoo Geocoding	57	REST
Google Search	50	SOAP
Technorati	33	REST
Yahoo Image Search	25	REST
Yahoo Local Search	24	REST
Trynt	20	REST
Windows Live Search	18	SOAP
Yahoo Term Extraction	18	REST
Upcoming.org	18	REST
geocoder	17	REST, XML-RPC, SOAP

What to do with this chart? I'd say that we should go down this list of the 20 most popular, gather them by theme and document what's it's like to get started with as many of these APIs. We can draw general lessons. (For instance, we see a predominance of REST-type services -- even though we won't want to confine ourselves to REST.)

First of all, the Flickr API is the #2 used API in mashups and is a main subject throughout this book. The Javascript-based maps (first and foremost Google maps, but also

[7] http://www.programmableweb.com/apilist/bymashups

Yahoo maps and Virtual Earth, will be covered first in Chapter 8 (the next chapter) and in depth in Chapter 13. The geocoding APIs, which are REST, are covered extensively also in Chapter 13.

In the following sections, we cover the following RESTful APIs:

* Yahoo search (including the regular search, image search, and local search)
* term extraction services (including Yahoo and maybe Trynt)
* del.icio.us
* Amazon E-commerce
* YouTube
* eBay
* Technorati
* Upcoming.org
* 411Sync

Later in this chapter, we will cover a bunch of SOAP services:

* Google Search
* Windows Live Search

If I have time, look at ones I think are of note that aren't among the most popular.

* google calendar -- a google GData
* google widgets

Yahoo search

`http://www.programmableweb.com/api/Yahoo` points to `http://developer.yahoo.com/search/`

You need an "application ID", which you get from
`https://developer.yahoo.com/wsregapp/index.php`

You can see your registered apps:
https://developer.yahoo.com/wsregapp/index.php?view

Authentication system: BBAuth (http://developer.yahoo.com/auth/)

I signed up for the ability to do Singlesignon -- need to set up an application endpoint (http://examples.mashupguide.net/ch07/yahoo.php). You get an application ID and a shared secret.

Let's do a web-search (w/o authentication)

http://developer.yahoo.com/search/web/ is the documentation -- has different parts.
Classic web search docs -- http://developer.yahoo.com/search/web/V1/webSearch.html

The docs give a sample query:

http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=madonna&results=2

Substitute your own API key (`ShJ.7GrIkY3X..DqmjFAX4cYIejaFg--`) and search for `flower`:

http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=ShJ.7GrIkY3X..DqmjFAX4cYIejaFg--&query=flower&results=2

Parameters are documented at:
http://developer.yahoo.com/search/web/V1/webSearch.html

Interesting that there is a XSD for the response:
http://search.yahooapis.com/WebSearchService/V1/WebSearchResponse.xsd

API Kit: http://developer.yahoo.com/download/ BSD license.

Yahoo Images

use pw.com:

http://www.programmableweb.com/api/YahooImages

sample search:

```
http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=YahooDemo&query=Corvette&results=2
```

Substitute your own key and search term:

```
http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=ShJ.7GrIkY3X..DqmjFAX4cYIejaFg--&query=flower&results=2
```

Yahoo Local Search

more of the same pattern....

Term Extraction

Yahoo Term Extraction

http://developer.yahoo.com/search/content/V1/termExtraction.html

should do POST not GET -- especially context is large

```
http://search.yahooapis.com/ContentAnalysisService/V1/termExtraction
```

parameters:

* appid
* context
* query (optional)

```
http://search.yahooapis.com/ContentAnalysisService/V1/termExtraction?appid=YahooDemo&context=Italian+sculptors+and+painters+of+the+renaissance+favored+the+Virgin+Mary+for+inspiration.&query=madonna
```

del.icio.us

http://www.programmableweb.com/api/del.icio.us

http://del.icio.us/help/api/:

* " All del.icio.us APIs are done over https and require HTTP-Auth."
* need to worry about rate
* HTTP error codes
* user-agents (notice no keys)

Implication: you can work on your own stuff but not others unless they give you their credentials.

```
curl -u USER:PASSWORD   https://api.del.icio.us/v1/posts/all?tag=architecture
```

API kits: python: http://code.google.com/p/pydelicious/

PHP:

* http://dietrich.ganx4.com/delicious/
* http://www.ejeliot.com/pages/5

Amazon E-commerce

http://www.programmableweb.com/api/Amazon

`http://aws.amazon.com`

`http://www.amazon.com/gp/browse.html?node=12738641` -- specifically for the E-commerce services

Need to join to get keys: http://www.amazon.com/gp/aws/registration/registration-form.html

To get your keys if you are already a member: http://aws-portal.amazon.com/gp/aws/developer/account/index.html/?ie=UTF8&action=access-key

You get an Access Key ID and a secret access key -- you can also use a X.509 certificate.

REST or SOAP available.

concise documentation of REST requests: http://docs.amazonwebservices.com/AWSEcommerceService/2005-10-05/PgRestRequestsArticle.html

different baseURLs depending on which Amazon service you use.

US: http://webservices.amazon.com/onca/xml?Service=AWSECommerceService

More example docs:

http://docs.amazonwebservices.com/AWSEcommerceService/2007-01-17/PgRestRequestsArticle.html

Example given in the docs:

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&AWSAccessKeyId=[your Access Key ID here]&Operation=ItemSearch&SearchIndex=Books&Keywords=flower

Substituting my own key:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&AWSAccessKeyId=0Z8Z8FYGP01Q00KF5802&Operation=ItemSearch&SearchIndex=Books&Keywords=flower
```

To get info about a specific book:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&AWSAccessKeyId=0Z8Z8FYGP01Q00KF5802&Operation=ItemLookup&ItemId=0596527462&ResponseGroup=ItemAttributes
```

YouTube

YouTube is probably the most famous video-sharing site on the Web -- and it also uses tagging as one way of organizing content.

I will add more context. Interesting see how tags might work for video.

My account: `http://www.youtube.com/user/rdhyee`

Tagging in YouTube

Basic documentation:
`http://www.google.com/support/youtube/bin/answer.py?answer=55769`

I uploaded a video to test out tagging:
`http://www.youtube.com/watch?v=XHnE4umovw4`

(I can't seem to get a different type of query for full-text vs tags.)

Other Gathering mechanisms in YouTube

Features of YouTube that might not be directly related to tagging but are related to gather/create/share:

* you can save a video to your "Favorites", specifically into Playlists
* you can save a video to a group, including videos that don't belong to you.

Can you deeplink to a YouTube video?

Groups in YouTube

I bring up groups as a way of looking at ways to aggregate in YouTube.

I joined the JSB group: `http://youtube.com/group/JSB`

You can get the list of videos in the group: `http://youtube.com/groups_videos?name=JSB`

Categories and tagging

Categories are formal hierarchy in which you can classify videos. So YouTube uses a mix between a fixed, though broad, hierarchical classification and tags.

YouTube API

YouTube API has REST and XML-RPC interface. Examples here are using the REST interface.

1. You first set up your own development profile:

```
http://www.youtube.com/my_profile_dev
```

   Very interesting that you enter your own secret! When you have created your profile, you get a "developer ID" (e.g., `U6bAdaYYLtc`)

2. Read the documentation: `http://www.youtube.com/dev_docs`

3. To get the user profile for rdhyee:

```
http://www.youtube.com/api2_rest?method=youtube.users.get_profile&dev_id=U6bAdaYYLtc
&user=rdhyee
```

4. To get the list of rdhyee's favorite videos:

```
http://www.youtube.com/api2_rest?method=youtube.users.list_favorite_videos&dev_id=U6
bAdaYYLtc&user=rdhyee
```

5. To get details of a video:

```
http://www.youtube.com/api2_rest?method=youtube.videos.get_details&dev_id=U6bAdaYYLt
c&video_id=XHnE4umovw4
```

6. To get videos by tag:

```
http://www.youtube.com/api2_rest?method=youtube.videos.list_by_tag&dev_id=U6bAdaYYLt
c&tag=HolidayWeekend&page=1&per_page=100
```

7. To get videos by category and tags

```
http://www.youtube.com/api2_rest?method=youtube.videos.list_by_category_and_tag&dev_
id=U6bAdaYYLtc&category_id=10&tag=Bach&page=1&per_page=10
```

There's more to the API, but we can see how this works.

I wonder how many API kits have been written:

* Ruby[8]
* Python?[9]

eBay

http://www.programmableweb.com/api/eBay

http://developer.ebay.com/

Nice quick guide start on http://developer.ebay.com/

1. Need to join: http://developer.ebay.com/join
2. Wait to get an email to confirm your email address. Go to the link given in the email and type in the confirmation code.
3. Once you have done that, you will be given 3 keys: a DevID, AppID, and a CertID
4. Need to register a sandbox test account: http://developer.ebay.com/DevZone/sandboxuser.asp
5. for REST, you then need to generate a token: http://developer.ebay.com/tokentool/ -- to use for " by applications which make API calls on behalf of a single or few eBay users." Enter your 3 keys and then you'll be sent to login as your sandbox test user. At the end, you'll get a token and a REST key -- both of which have expiration times.
6. consult REST docs (http://developer.ebay.com/developercenter/rest) -- you need to download a PDF
7. Try the query: http://rest.api.ebay.com/restapi?CallName=GetSearchResults&RequestToken=UserToken&RequestUserId=UserName&Query=toy%20boat&Version=491&UnifiedInput=1

[8] http://shanesbrain.net/articles/2006/09/28/a-ruby-interface-to-the-youtube-api

[9] http://thinkhole.org/wp/2006/01/09/the-youtube-api-and-python/

I've not been able to get this to work yet.

Technorati

http://www.programmableweb.com/api/Technorati points to
http://developers.technorati.com/wiki -- which is confusing
http://developers.technorati.com/wiki/TechnoratiApi says you need to signup first:
http://technorati.com/signup/

" You may use this API key to make use of the Technorati API for **personal, noncommercial use** only."

Better place to get docs: http://technorati.com/developers/

Let's do a search on the cosmos query -- which returns blogs pointing to a specific URL

REST-ful -- GET or POST

baseURL: http://api.technorati.com/cosmos

mandatory parameters: key and url

http://api.technorati.com/cosmos?key=[YOUR KEY]&url=http://programmableweb.com

Way slow!!

Upcoming.org

See Chapter 15 for a detailed discussion of this API.

411Sync

Work this out.

SOAP and XML-RPC Services

Using SOAP from PHP

Libraries in PHP

SOAP libraries for PHP:

* http://sourceforge.net/projects/nusoap/

pear SOAP package[10]

[10] http://pear.php.net/package/SOAP

```
pear install -of http://download.pear.php.net/package/SOAP-0.10.1.tgz
http://pear.php.net/package/SOAP/
```

or

```
pear install  channel://pear.php.net/SOAP-0.10.1
```

Let's write something for the PEAR SOAP client.[11]

There is a built-in SOAP client in PHP 5 -- if the installation of PHP was configured with *--enable-soap* -- which is not the case with dreamhost:

http://us2.php.net/soap

If the library were installed, a test script would work.[12]

We will try out

http://www.xmethods.net/ve2/ViewListing.po;jsessionid=pDZ_eCQdAHgHauXN2jElh5sf(2MgVnSRM)?key=uuid:889A05A5-5C03-AD9B-D456-0E54A527EDEE

Good soapclient tutorial: http://devzone.zend.com/node/view/id/689

Using WSDLfrom PHP

Good docs: http://www.phppatterns.com/docs/develop/pear_soap_client_fast_start

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
    <title>GoogleSearch.php</title>
  </head>
  <body>
  <?php
  ini_set(
    'include_path',
      ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . '/usr/local/lib/php'
      );
  require 'SOAP/Client.php';

  $wsdl=new SOAP_WSDL(
      'http://api.google.com/GoogleSearch.wsdl');

// Look at the generated code...
  echo ( $wsdl->generateProxyCode() );
```

[11] http://proquest.safaribooksonline.com/1565926811/phpckbk-CHP-12-SECT-9

[12] http://examples.mashupguide.net/ch07/soaptest1.php

```
  ?>
  </body>
</html>
```

generates:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
    <title>GoogleSearch.php</title>
  </head>
  <body>
  class WebService_GoogleSearchService_GoogleSearchPort extends SOAP_Client
{
    function WebService_GoogleSearchService_GoogleSearchPort($path =
'http://api.google.com/search/beta2')
    {
        $this->SOAP_Client($path, 0);
    }
    function &doGetCachedPage($key, $url)
    {
        $result = $this->call('doGetCachedPage',
                              $v = array('key' => $key, 'url' => $url),
                              array('namespace' => 'urn:GoogleSearch',
                                    'soapaction' => 'urn:GoogleSearchAction',
                                    'style' => 'rpc',
                                    'use' => 'encoded'));
        return $result;
    }
    function &doSpellingSuggestion($key, $phrase)
    {
        $result = $this->call('doSpellingSuggestion',
                              $v = array('key' => $key, 'phrase' => $phrase),
                              array('namespace' => 'urn:GoogleSearch',
                                    'soapaction' => 'urn:GoogleSearchAction',
                                    'style' => 'rpc',
                                    'use' => 'encoded'));
        return $result;
    }
    function &doGoogleSearch($key, $q, $start, $maxResults, $filter, $restrict,
$safeSearch, $lr, $ie, $oe)
    {
        $result = $this->call('doGoogleSearch',
                              $v = array('key' => $key, 'q' => $q, 'start' =>
$start, 'maxResults' => $maxResults, 'filter' => $filter, 'restrict' => $restrict,
'safeSearch' => $safeSearch, 'lr' => $lr, 'ie' => $ie, 'oe' => $oe),
                              array('namespace' => 'urn:GoogleSearch',
                                    'soapaction' => 'urn:GoogleSearchAction',
                                    'style' => 'rpc',
                                    'use' => 'encoded'));
```

```
        return $result;
    }
}
  </body>

</html>
```

Now I'm trying to invoke the call:[13]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
    <title>GoogleSearch.php</title>
  </head>
  <body>
  <?php
  ini_set(
    'include_path',
      ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . '/usr/local/lib/php'
      );
  require 'SOAP/Client.php';

  $wsdl=new SOAP_WSDL(
      'http://api.google.com/GoogleSearch.wsdl');

  $google = $wsdl->generateProxyCode();

  // Create an object directly from the proxy code
  $key= "YOUR KEY";
  $q = "Bach";
  $start = 0;
  $maxResults=10;
  $filter = "1";
  $restrict = "String";
  $safeSearch = "1";
  $lr = "String";
  $ie = "String";
  $oe = "String";
  print_r($google);
  $results =${$google->doGoogleSearch()}($key, $q, $start, $maxResults, $filter,
$restrict, $safeSearch, $lr, $ie, $oe);
  #$results = $google->doGoogleSearch($key, $q, $start, $maxResults);

  print_r ( $results);
  ?>
  </body>
```

[13] http://examples.mashupguide.net/ch07/GoogleSeach.php

```
</html>
```

I don't have this code quite working.

XML-RPC from PHP

Two possible libraries:

* http://phpxmlrpc.sourceforge.net/
* http://pear.php.net/package/XML_RPC/

For completeness, I might want to write up an exact how-to.

Example SOAP Services

Flickr API via SOAP

How can I use Python to invoke flickr.test.echo?

If I had a WSDL file, I could do the following[14]:

```
from SOAPpy import WSDL
server = WSDL.Proxy('https://calendar-
ws.berkeley.edu:8004/calendaraccess/services/CalendarAccessImpl.wsdl')
server.(...)
```

Maybe I will try a new soaplib:[15] Won't work unless it was installed with the php 5 installation.

We can also try to invoke it using curl

We use the following SOAP message:

```
<?xml version='1.0' encoding='UTF-8'?>
<s:Envelope xmlns:s='http://www.w3.org/2003/05/soap-envelope'
xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/1999/XMLSchema'>
  <s:Body>
    <x:FlickrRequest xmlns:x='urn:flickr'>
      <method>flickr.test.echo</method>
      <api_key>e81ef8102a5160154ef4662adcc9046b</api_key>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

---

[14] http://pywebsvcs.sourceforge.net/apidocs/SOAPpy/

[15] http://trac.optio.webfactional.com/

which we can send to the Flickr SOAP endpoint `http://api.flickr.com/services/soap/` using curl:[16]

```
curl -d "<?xml version='1.0' encoding='UTF-8'?><s:Envelope
xmlns:s='http://www.w3.org/2003/05/soap-envelope'
xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/1999/XMLSchema'><s:Body><x:FlickrRequest
xmlns:x='urn:flickr'><method>flickr.test.echo</method>
<api_key>e81ef8102a5160154ef4662adcc9046b</api_key></x:FlickrRequest></s:Body></s:En
velope>"  http://api.flickr.com/services/soap/
```

to which we get:

```
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <s:Body>
    <x:FlickrResponse xmlns:x="urn:flickr">
&lt;method&gt;flickr.test.echo&lt;/method&gt;
&lt;api_key&gt;e81ef8102a5160154ef4662adcc9046b&lt;/api_key&gt;
                </x:FlickrResponse>
  </s:Body>
</s:Envelope>
```

We can use PHP + the PEAR SOAP client:

```
<?php
ini_set(
  'include_path',
    ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . '/usr/local/lib/php'
    );
require 'SOAP/Client.php';

$soap = new SOAP_Client('http://api.flickr.com/services/soap/');

$params = array(
            new SOAP_Value('method','string', 'flickr.test.echo'),
            new SOAP_Value('api_key','string', 'e81ef8102a5160154ef4662adcc9046b'),
          );

$echo_value = $soap->call('FlickrRequest', $params, 'urn:flickr');

print_r($echo_value);
?>
```

Let's do a search for flowers and display the total number of pictures:[17]

---

[16] http://www.flickr.com/services/api/request.soap.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
    <title>pearsoap_flickr2.php</title>
  </head>
  <body>
  <?php
  ini_set(
    'include_path',
      ini_get( 'include_path' ) . PATH_SEPARATOR . "/home/rdhyee/pear/lib/php" .
PATH_SEPARATOR . '/usr/local/lib/php'
      );
  require 'SOAP/Client.php';

  $soap = new SOAP_Client('http://api.flickr.com/services/soap/');
  $tags = "flower";

  $params = array(
              new SOAP_Value('method','string', 'flickr.photos.search'),
              new SOAP_Value('api_key','string',
'e81ef8102a5160154ef4662adcc9046b'),
              new SOAP_Value('tags','string', $tags),
              new SOAP_Value('per_page','string', '5')
            );

  $results = $soap->call('FlickrRequest', $params, 'urn:flickr');
  $xml = simplexml_load_string($results);
  echo "total number of pictures:", $xml['total'];
  ?>
  </body>
<./html>
```

Flickr and WSDL

I've been interested in generating WSDL from the Flickr reflection methods to generate a library that could better keep up with the changes in the Flickr APIs. However, I've run into a problem that stems from what I believe to be either unorthodox SOAP syntax in Flickr -- or just the limitations in my own understanding of WSDL and SOAP.

Here's what I did. I first set out to invoke the flickr.test.echo method[18] by generating the right WSDL to call the flickr.test.echo SOAP method. The documentation for SOAP invocation is found at http://www.flickr.com/services/api/request.soap.html

---

[17] http://examples.mashupguide.net/ch07/pearsoap_flickr2.php

[18] http://www.flickr.com/services/api/flickr.test.echo.html

The following SOAP message works with the Flickr SOAP endpoint

http://api.flickr.com/services/soap/

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.test.echo</method>
      <api_key>e81ef8102a5160154ef4662adcc9046b</api_key>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

You can try this out with curl:

```
curl -d "<?xml version='1.0' encoding='UTF-8'?><s:Envelope
xmlns:s='http://www.w3.org/2003/05/soap-envelope'
xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/1999/XMLSchema'><s:Body><x:FlickrRequest
xmlns:x='urn:flickr'><method>flickr.test.echo</method>
<api_key>e81ef8102a5160154ef4662adcc9046b</api_key></x:FlickrRequest></s:Body></s:En
velope>" http://api.flickr.com/services/soap/
```

The problem I'm running into now is to how to get this SOAP call

expressed in WSDL. My first shot is:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://mashupguide.net" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mashupguide.net" name="FlickrService">
  <message name="flickr.test.echo_Request">
    <part name="method" type="xsd:string"/>
    <part name="api_key" type="xsd:string"/>
  </message>
  <message name="flickr.test.echo_Response">
    <part name="response" type="xsd:string"/>
  </message>
  <portType name="flickr.test.echo_PortType">
    <operation name="FlickrRequest">
      <input message="tns:flickr.test.echo_Request"/>
      <output message="tns:flickr.test.echo_Response"/>
    </operation>
  </portType>
  <binding name="Flickr_Binding" type="tns:flickr.test.echo_PortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="FlickrRequest">
```

```
      <soap:operation soapAction="FlickrRequest"/>
      <input>
        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:flickr"/>
      </input>
      <output>
        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:flickr"/>
      </output>
    </operation>
  </binding>
  <service name="Flickr">
    <documentation>WSDL File for Flickr</documentation>
    <port name="Flickr_Port" binding="tns:Flickr_Binding">
      <soap:address location="http://api.flickr.com/services/soap/"/>
    </port>
  </service>
</definitions>

which, using XML-Spy gets me a SOAP body of

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:FlickrRequest xmlns:m="urn:flickr">
      <method xsi:type="xsd:string">String</method>
      <api_key xsi:type="xsd:string">String</api_key>
    </m:FlickrRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

I then have to fill out the method -> flickr.test.echo and the api_key -> e81ef8102a5160154ef4662adcc9046b -- and that gets me what I want -- sorta.

The problem is that 1) I shouldn't have to fill out the method name -- that's what I want the WSDL to have to generate and 2) I can't see a way to add other Flickr methods to the same WSDL file because they all invoke a FlickrRequest method -- and pass the real method name as a method parameter. I'm getting the sense that if Flickr used the following SOAP body instead;

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <s:Body>
    <x:flickr.test.echo xmlns:x="urn:flickr">
      <api_key>e81ef8102a5160154ef4662adcc9046b</api_key>
```



```
    </x:flickr.test.echo>
  </s:Body>
</s:Envelope>
```

I'd be able to generate the WSDL w/o any problem.

Do you any way around this problem? I could generate one WSDL file for each Flickr method -- but that doesn't seem quite right either.

Google Search

Deprecation of the Google SOAP search API[19] -- but included here for historic interest -- and for those who happen to have old keys.

http://www.programmableweb.com/api/Google

http://api.google.com/GoogleSearch.wsdl

doGoogleSearch -- you need key, q, and maxResults <= 10

Windows Live Search

http://msdn2.microsoft.com/en-us/library/bb251794.aspx

Getting started: http://dev.live.com/blogs/livesearch/archive/2006/03/23/27.aspx

WSDL: http://soap.search.msn.com/webservices.asmx?wsdl

Need to set up an API id: http://search.msn.com/developer

Will need http://msdn2.microsoft.com/en-us/library/bb266177.aspx to help with the "CultureInfo" field -- en-US for American Englis

The parameters are confusing -- which ones are mandatory. Would be nice if WSDL provided a canned search that you just have to plug in your key....

eBay SOAP + WSDL

http://www.programmableweb.com/api/eBay

http://developer.ebay.com/

Amazon E-commerce

WSDL:
http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl?

Using XML Spy, the SOAP request that comes back has tons of parameters and it wasn't obvious to me what the minimalist request is.

---

[19] http://radar.oreilly.com/archives/2006/12/google_depreciates_SOAP_API.html

Amazon S3 -- using WSDL

Refer to the ProgrammableWeb.com page on the AWS:
`http://www.programmableweb.com/api/Amazon`

which leads to
`http://www.amazon.com/gp/browse.html/ref=sc_fe_l_2/103-1644811-9832630?%5Fencoding=UTF8&node=12738641&no=3435361`

Keys needed:

* Access Key ID: 0Z8Z8FYGP01Q00KF5802
* Secret Access Key:

Can use X.509 certificates

list buckets for Amazon s3

S3 code:

```
AWSAccessKeyId='0Z8Z8FYGP01Q00KF5802'
AWSSecret = '[secret kye]'

from SOAPpy import WSDL

import sha

def calcSig(key,text):
    import hmac, base64
    sig = base64.encodestring(hmac.new(key, text, sha).digest()).strip()
    return sig

def ListMyBuckets(s):
    from time import gmtime,strftime
    method = 'ListAllMyBuckets'
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecret,text)
    print "ListMyBuckets: ts,text,sig->", ts, text, sig
    return s.ListAllMyBuckets(AWSAccessKeyId=AWSAccessKeyId,
Timestamp=ts,Signature=sig)

def CreateBucket(s, bucketName):
    from time import gmtime,strftime
    method = 'CreateBucket'
    print 'method: ', method
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecret,text)
    print "CreateBuckets: ts,text,sig->", ts, text, sig
```



```
    return s.CreateBucket(Bucket=bucketName, AWSAccessKeyId=AWSAccessKeyId,
Timestamp=ts,Signature=sig)


s = WSDL.Proxy("http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl")
print ListMyBuckets(s)
#print CreateBucket(s,'RaymondYee')
#print ListMyBuckets(s)
```

Using an API kit

A better example might be a PHP one....but this is an example of using Python to use the Basecamp API.

The following is an example of using an API kit to use an API -- specifically the Basecamp API I've done some work in Python, using the Basecamp Python wrapper.[20] Here's some code that works for converting Basecamp milestones to XML that can be fed into the Simile AJAX timeline widget[21]:

```
def ds_milestones():

    import basecamp
    import datetime, time, pytz
    from dateutil import tz
    from elementtree import ElementTree as ET


    url = 'https://ist-dataservices.grouphub.com/'
    user = 'Raymond'
    pw = 'MYPASSWORD'

    bc = basecamp.Basecamp(url,user, pw)


    ds_allstaff_id = ID_FOR_SITE
    s = bc.list_milestones(ds_allstaff_id) # generates XML for milestones
    et = ET.fromstring(s)

    Pacific_tz = pytz.timezone('US/Pacific') # hardcode to Pacific (option 1)

    # create output XML
    data_root = ET.Element("data")

    milestones = et.findall('milestone')
```

[20] http://homework.nwsnet.de/products/79/

[21] http://simile.mit.edu/timeline/

```
    for m in milestones:
        title = m.find('title').text.strip()
        deadline = m.find('deadline').text.strip()
        deadline_tuple = time.strptime(deadline,"%Y-%m-%d")
        deadline_dt =
apply(datetime.datetime,deadline_tuple[0:3],{'tzinfo':Pacific_tz})
        # turn date from "2006-12-08" into "Aug 02 2006 00:00:00 GMT" (iso-format) -
- do I have to?
        # I think we need to use the Pacific timezone in the code.
        event = ET.SubElement(data_root, "event")
        event.attrib['title'] = title
        event.attrib['start'] = deadline_dt.astimezone(tz.tzutc()).strftime("%b %d
%Y %H:%M:%S GMT")

    print ET.tostring(data_root)
```

Now, I've been trying to come up with code to create milestones in a site:

```
import basecamp
bc = basecamp.Basecamp('https://ist-dataservices.grouphub.com/',"Raymond",
"MYPASSWORD")
bc.create_milestone(530019,"learn the Basecamp API","2007-03-10",746005,True)
```

is supposed to work -- but doesn't.

I can get a curl invocation to work[22]:

```
curl  -H "Accept: application/xml" -H "Content-Type: application/xml" -u
Raymond:MYPASSWORD  https://ist-dataservices.grouphub.com/contacts/person/746005
```

and

```
curl  -H "Accept: application/xml" -H "Content-Type: application/xml" -u
Raymond:MYPASSWORD  -d  "<request> <milestone><title>Learn the Basecamp
API</title><deadline type='date'>2007-03-10</deadline> <responsible-
party>746005</responsible-party>     <notify>true</notify> </milestone> </request>"
https://ist-dataservices.grouphub.com/projects/530019/milestones/create
```

[22] http://curl.haxx.se/docs/manual.html

what needs to happen is to debug the Python code -- maybe there is a problem with the library.

REST vs SOAP

This section might be too esoteric for the readers.

In some ways, the SOAP vs REST argument is a reflection of a typical pattern of argumentation in the tech field. One wants to reduce the combinatorics of the problem. But any given solution isn't optimal for all cases. So do you go for one solution and make it work for a wide range of cases or do you go for a spectrum of solutions, each optimal for various subcases?

If you spend a lot of time in this field, you'll hear some people argue that one is better than the other. Pragmatically speaking, I would say you should learn how to consume either REST or SOAP (or XML-RPC) because doing so gives you as a remixer many more options than if you stick to one or the other exclusively. As a service provider, I would try to emulate Flickr, which offers REST and SOAP (as well as XML-RPC).

A useful link is[http://blogs.msdn.com/xmlteam/archive/2005/02/15/372982.aspx Microsoft XML Team's WebLog : MS Ignoring developer demand for REST tools?], a very useful blog post for a number of reasons. There's recognition of cases in which REST shines -- and there's a discussion of conditions for which a more complex set of technologies, as represented by SOAP and the WS-* stack, are warranted:

> *Second, flickr and Amazon are definitely "semantically rich" platforms, but they don't have the pain for which WS-* offers an analgesic. Plain ol' XML over HTTP works just fine, and WS-* is overkill, in situations where:*
>
> ** information is public and encryption/authentication are unnecessary;*
>
> **all communication visible to the service consumer is done over one protocol, HTTP;*
>
> **nothing terribly bad happens if a message is lost or duplicated;*
>
> ** and there are few demands for multi-part transaction management beyond what can be implemented with HTTP sessions or cookies.*
>
> *Users of browser-based applications come to expect the quality of service one associates with the Web. Web services toolkits such as Indigo on the other hand, are oriented toward use cases that demand high-security, support for existing enterprise messaging systems, message failure is not an option, and industrial-strength transaction processing is assumed. The REST approach (as far as I can tell after years of discussion) assumes that all these features are provided at the application level rather than being provided by the infrastructure. I am convinced that developers with demanding requirements for security, reliability, transactions, etc. generally DO want what WS-* and*

> *Indigo promise. Perhaps there is some way to offer these services within a RESTful toolkit so that application developers don't have to wrestle with them. The obvious answer, however, is that few people seem to have much of a clue how to do this. Those who do, such as the developers at Amazon, have invested fortunes in making it happen.*

The article points to some good references:

* [http://www.w3.org/TR/ws-arch/ Web Services Architecture]. This a major W3C article that is worth my careful study when I get to this section of the book.
* [http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp Web Services and Other Distributed Technologies Developer Center: Advanced Web Services An Introduction to the Web Services Architecture and Its Specifications] is a good exposition of the WS-* side of the house.
* [http://en.wikipedia.org/wiki/Representational_State_Transfer Representational State Transfer - Wikipedia, the free encyclopedia] is a good argument for the REST side of the house.

In addition, I will keep an eye out for [http://www.crummy.com/writing/REST-Web-Services/ REST Web Services] by Sam Ruby and Leonard Richardson.

Conclusions

In this chapter, you have learned about the wide range of public APIs that are based on XML Web services. You've also learn how to make use of any given Web service whether it is RESTful or based on SOAP.

In the next chapter we will look at JavaScript-based APIs.

