HELLO WORLD

Programmatic Access to Wikipedia

by Dmitry Batenkov

Wikipedia is often regarded as today's best-known source of collaborative intelligence. As such, it can also be an excellent subject for research which comes into the domain of "distributed cognition." In this tutorial we will learn how to programmatically access the data behind Wikipedia by using its Web API.

Web API

Web application programming interfaces (APIs) are the standard way of communication in the Web 2.0 environment. There may be several variants, but the most basic one is as follows:

1. The client (for example, the web browser) requests data by sending an HTTP request to the server, optionally passing parameters in the query string.
2. The server returns the result in a well-defined format, usually XML or JSON.

The description of the API methods should somehow be made available to the client.

Accessing MediaWiki

MediaWiki, the wiki engine behind Wikipedia and many other collaborative projects, exposes a public Web API whose entry point varies but in general looks like this: *http://SITE/.../api.php*.

For English Wikipedia, it is *http://en.wikipedia.org/w/api.php*, while for the Polymath project it is *http://michaelnielsen.org/polymath1/api.php*. Pointing your browser to this address will give you a complete documentation of the API. To get a feeling of how it works, let's consider some examples.

Example 1

http://en.wikipedia.org/w/api.php?action=query&list=random&rnnamespace=0

- **action=query**—this is what you will use most of the time unless you want to edit data (when developing bots for example)
- **list=random**—instructs to choose a random page
- **rnnamespace=0**—instructs to select a page in namespace with id=0. More on that later.

The return value from the above call is a data structure which looks like this:

```
<?xml version="1.0"?>
<api>
 <query>
  <random>
   <page id="3997844" ns="0"
title="European medieval architecture in North America" />
  </random>
 </query>
</api>
```

The format of the data can be controlled by passing "format" parameter to each API call. Since in this example we didn't supply this parameter, the default value "xmlfm" was used, which means "XML pretty-printed in HTML." You would mostly use this for debugging. In real applications, you will probably specify "xml" or "json."

JSON stands for JavaScript object notation. It is a data interchange format which is both human-readable and can be easily parsed, analogous to XML.

Example 2

Each resource in MediaWiki belongs to a single namespace, for example:

- id=0: normal pages (i.e., content pages)
- id=1: talk pages
- id=14: category pages

To see a complete list of namespaces, issue the following call:

http://en.wikipedia.org/w/api.php?action=query&meta=siteinfo&siprop=namespaces

Using Python API

There are wrappers around the Web API for many scripting languages. In **Listings 1-3**, we demonstrate how to use the Python API. You should install the following prerequisites:

- Python 2.5+
- **python-wikitools** package, which is available at *http://code.google.com/p/python-wikitools/*.

RESOURCES & FURTHER READING

MediaWiki API Main Documentation
http://www.mediawiki.org/wiki/API

Wikipedia Bots Development
http://en.wikipedia.org/wiki/Wikipedia:Creating_a_bot

MediaWiki Client tools
http://www.mediawiki.org/wiki/API:Client_Code

WikiXRay
http://meta.wikimedia.org/wiki/WikiXRay

Listing 1: Requests to the WebAPI are sent using api.APIRequest objects. Individual pages can be obtained with pagelist.listFromQuery() method which returns a list of page objects.

```
from wikitools import *  # wiki, page, pagelist, api, category, ...

site = wiki.Wiki("http://en.wikipedia.org/w/api.php")
params = {'action':'query',
      'list':'random',
      'rnnamespace':0 }
request = api.APIRequest(site, params)

lst = pagelist.listFromQuery(
  site, request.query()['query']['random'])
p = lst[0]
p_url = ''.join( (site.domain, '/wiki/',p.urltitle)]

print ''.join(['Title: ',p.title])
print ''.join(['URL: ', p_url])
cats = p.getCategories()
```

Listing 2: A page can be queried for the categories it belongs to.

```
if cats:
  print 'Categories:'
  for ct in pagelist.listFromTitles(site,cats):
    print ''.join(['\t* ',ct.unprefixedtitle])
else:
  print 'No categories'
```

Listing 3: You can easily access the links and backlinks for any page.

```
request2 = api.APIRequest(site, {'action':'query',
                'titles':p.title,
                'prop':'extlinks'})
pageinfo = request2.query()['query']['pages'][p.pageid]
if 'extlinks' in pageinfo:
  exts = pageinfo['extlinks']
  print 'External links:'
  for e in exts:
    url = e['*']
    print ''.join(['\t* ', url])
else:
  print 'No external links'

request3 = api.APIRequest(site, {'action':'query',
                'list':'backlinks',
                'bltitle':p.title,
                'blnamespace':0})
blinks = request3.query()['query']['backlinks']
if blinks:
  print 'Backlinks:'
  for bl in blinks:
    print ''.join(['\t* ',bl['title']])
else:
  print 'No backlinks'
```