CHAPTER 19

Search

This chapter shows how to use the Google, Yahoo, and live.com search APIs, as well as configuring searchable websites for access as search plugin in Firefox 2.0 or IE 7 using OpenSearch. This chapter will also examine briefly how to use the Google Desktop Search API.

No one needs to be reminded that search engines are at the heart of the current Web infrastructure. Not surprisingly, it's useful to be able to integrate search functionality and search results into mashups. If a mashup is integrated with search engines via their APIs, users of the mashups can more easily find and reuse that digital content.

Google Search

Google was one of the first major search companies to provide an API: the Google SOAP API. Unfortunately, since December 2006, no new developer keys have been issued because Google is directing users to its newer Ajax Search, which we will study below.

Google SOAP Search

If you have a key and want to use the API, take a look at the documentation at

```
http://code.google.com/apis/soapsearch/
```

The WSDL for the service is at:

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

The combination of the WSDL and your key allows you to use the techniques in Chapter 7 to do a search. For example, in Python

```
>>> server.show_methods()
```

tells you that there are three methods: doGoogleSearch, doGetCachedPage, and doSpellingSuggestion. If you look specifically at doGoogleSearch, you can take a look at the parameters:

```
Method Name: doGoogleSearch

   In #0: key  ((u'http://www.w3.org/2001/XMLSchema', u'string'))
   In #1: q  ((u'http://www.w3.org/2001/XMLSchema', u'string'))
   In #2: start  ((u'http://www.w3.org/2001/XMLSchema', u'int'))
   In #3: maxResults  ((u'http://www.w3.org/2001/XMLSchema', u'int'))
   In #4: filter  ((u'http://www.w3.org/2001/XMLSchema', u'boolean'))
   In #5: restrict  ((u'http://www.w3.org/2001/XMLSchema', u'string'))
   In #6: safeSearch  ((u'http://www.w3.org/2001/XMLSchema', u'boolean'))
   In #7: lr  ((u'http://www.w3.org/2001/XMLSchema', u'string'))
   In #8: ie  ((u'http://www.w3.org/2001/XMLSchema', u'string'))
```

You can then invoke the method

```
server.doGoogleSearch('[YOURKEY]','flower',0,10,True,"",True,"","","")
```

to get some search results

You can invoke this method without using a SOAP library and work at the level of at HTTP library if you know the following parameters:

* SOAP Connection endpoint: http://api.google.com/search/beta2
* SOAPAction: urn:GoogleSearchAction
* the body of the SOAP request:

```
<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:Body>
    <m:doGoogleSearch xmlns:m="urn:GoogleSearch" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">[YOURKEY]</key>
      <q xsi:type="xsd:string">flower</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string"></restrict>
      <safeSearch xsi:type="xsd:boolean">true</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string"></ie>
      <oe xsi:type="xsd:string"></oe>
    </m:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You can use the following curl invocation to set two HTTP headers (for the SOAPAction and for Content-Type – since the Google SOAP search expects text/xml) and POST the SOAP body to the SOAP connection endpoint to get search results:

```
curl -H urn:GoogleSearchAction -H Content-Type:text/xml -d '<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:Body><m:doGoogleSearch
xmlns:m="urn:GoogleSearch" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><key
xsi:type="xsd:string">[YOURKEY]</key><q xsi:type="xsd:string">flower</q><start
xsi:type="xsd:int">0</start><maxResults xsi:type="xsd:int">1</maxResults><filter
xsi:type="xsd:boolean">true</filter><restrict
xsi:type="xsd:string"></restrict><safeSearch
xsi:type="xsd:boolean">true</safeSearch><lr xsi:type="xsd:string"></lr><ie
xsi:type="xsd:string"></ie><oe xsi:type="xsd:string"></oe></m:doGoogleSearch></SOAP-
ENV:Body></SOAP-ENV:Envelope>'   http://api.google.com/search/beta2
```

This is what you get back:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="ns1:GoogleSearchResult">
        <directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:Array" ns2:arrayType="ns1:DirectoryCategory[0]">
</directoryCategories>
        <documentFiltering xsi:type="xsd:boolean">false</documentFiltering>
        <endIndex xsi:type="xsd:int">1</endIndex>
        <estimateIsExact xsi:type="xsd:boolean">false</estimateIsExact>
        <estimatedTotalResultsCount
xsi:type="xsd:int">99100000</estimatedTotalResultsCount>
        <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[1]">
          <item xsi:type="ns1:ResultElement">
            <URL xsi:type="xsd:string">http://en.wikipedia.org/wiki/Flower</URL>
            <cachedSize xsi:type="xsd:string">77k</cachedSize>
            <directoryCategory xsi:type="ns1:DirectoryCategory">
              <fullViewableName xsi:type="xsd:string"/>
              <specialEncoding xsi:type="xsd:string"/>
            </directoryCategory>
            <directoryTitle xsi:type="xsd:string"/>
            <hostName xsi:type="xsd:string"/>
            <relatedInformationPresent
xsi:type="xsd:boolean">true</relatedInformationPresent>
            <snippet xsi:type="xsd:string">The &lt;b&gt;flower&amp;#39;s&lt;/b&gt;
structure contains the plant&amp;#39;s reproductive organs, and its function
&lt;br&gt;  is to produce seeds. After fertilization, portions of the
&lt;b&gt;flower&lt;/b&gt; develop &lt;b&gt;...&lt;/b&gt;</snippet>
            <summary xsi:type="xsd:string"/>
            <title xsi:type="xsd:string">&lt;b&gt;Flower&lt;/b&gt; - Wikipedia, the
free encyclopedia</title>
          </item>
        </resultElements>
        <searchComments xsi:type="xsd:string"/>
        <searchQuery xsi:type="xsd:string">flower</searchQuery>
        <searchTime xsi:type="xsd:double">0.129111</searchTime>
        <searchTips xsi:type="xsd:string"/>
        <startIndex xsi:type="xsd:int">1</startIndex>
      </return>
    </ns1:doGoogleSearchResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

At this point, the Google SOAP search is of interest primarily for historical interest because it has been deprecated by Google. Without this API, there is no longer any way

with the Google API to embed searching of Google within desktop applications. Even so, it remains a useful example of how to use a SOAP-based API.

Google Ajax Search

The Google Ajax search gives you a search widget that you can embed in your website. You can access functionality for searching the web, doing local searches (tied to maps), and video searches. The widget displays a search box and takes care of displaying search results in an HTML element that you designate.

The documentation for Google Ajax search is at:

`http://code.google.com/apis/ajaxsearch/`

Like Google Maps, you have to sign up for a key that is tied to a specific directory at

`http://code.google.com/apis/ajaxsearch/signup.html`

In my case, the directory is

`http://examples.mashupguide.net/ch19/`

and the returned key is

```
ABQIAAAAdjiS7YH6Pzk2Nrli02b5xxSx5C3cYPYuhZ4aEjydd6fwsFEUjhRR9rp9nxiVhhULjfmG3e9r7qoY
Tg
```

Paste the "hello world" type code into your page and load it up.[1] The Hello World code shows you how to create a basic search box and display the results.

Manipulating search results

Let's adapt the basic code to let a user search a particular search source (the web search) and save a result. This is done by creating a callback (`KeepHandler`) with the `setOnKeepCallback` method. You'll also see some code to access the attributes of the result.[2]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>google.ajax.2.html</title>
    <link href="http://www.google.com/uds/css/gsearch.css" type="text/css"
rel="stylesheet"/>
    <script
src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0&amp;key=ABQIAAAAdjiS7YH6Pzk
2Nrli02b5xxSx5C3cYPYuhZ4aEjydd6fwsFEUjhRR9rp9nxiVhhULjfmG3e9r7qoYTg"
type="text/javascript"></script>
    <script type="text/javascript">
    //<![CDATA[
```

---

[1] http://examples.mashupguide.net/ch19/google.ajax.1.html

[2] http://examples.mashupguide.net/ch19/google.ajax.2.html


```
    function KeepHandler(result) {
      // clone the result html node
      var node = result.html.cloneNode(true);

      // attach it
      var savedResults = document.getElementById("saved_results");
      savedResults.appendChild(node);

      // extract some info from the result to show to get at the individual
attributes.
      // see
http://code.google.com/apis/ajaxsearch/documentation/reference.html#_class_GwebResul
t
      var title = result.title;
      var unformattedtitle = result.titleNoFormatting;
      var content = result.content;
      var unescapedUrl = result.unescapedUrl;
      alert("Saving " + unformattedtitle + " " + unescapedUrl + " " + content);
    }


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

      // attach a handler for saving search results
      searchControl.setOnKeepCallback(this, KeepHandler);

      // expose the control to manipulation by the JavaScript shell and Firebug.
      window.searchControl = searchControl

      // Add in the web searcher
      searchControl.addSearcher(new GwebSearch());

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

      // Execute an inital search
      searchControl.execute("flower");
    }
    GSearch.setOnLoadCallback(OnLoad);

    //]]>
    </script>
  </head>
  <body>
    <div id="search_control"></div>
    <div id="saved_div"><span>Saved Search Results:</span><div
id="saved_results"></div></div>
  </body>
</html>
```

There's obviously more you can do with the Google Ajax search API such as styling the search widget. Consult the documentation to learn how. Noteworthy extras are:

* adding local search to a Google map: http://www.google.com/uds/solutions/localsearch/index.html
* searching outside of the widget context to do "raw searching":http://www.google.com/uds/samples/apidocs/raw-searchers.html

Indeed, there are pleanty of things to learn for your specifica applications from the sample code:

`http://code.google.com/apis/ajaxsearch/samples.html`

For those of you who are looking for a way of using Google search without creating a HTML interface, take a look specifically at

`http://www.google.com/uds/samples/apidocs/raw-searchers.html`

This sample gets the closest to giving you back the raw search functionality that the SOAP interface has – although you still need to use JavaScript and embed that search in a web page on the public web.

Yahoo! search

Tha Yahoo! search API is a RESTful one. Let's see how to use the Yahoo! search APIs:

`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`

Yahoo! has an authentication system called BBAuth:

`http://developer.yahoo.com/auth/`

In the authentication system, there is a `Single Sign-On` option. For this example, I signed up for the ability to do `Single Sign-On`, for which I needed to state an application endpoint:

`http://examples.mashupguide.net/ch07/yahoo.php`

Once you have registered your application, you get an application ID and a shared secret.

Now, let's do a web-search that doesn't require any authentication. Consulting the documentation

`http://developer.yahoo.com/search/web/`

and specifically the "classic" web search documentation

`http://developer.yahoo.com/search/web/V1/webSearch.html`

we see a sample query:

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

You can substitute your own key and search term. e.g.,

```
http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=[YourAppId]&query=flower&results=2
```

Yahoo! local search has a similar architecture:

```
http://developer.yahoo.com/search/local/V2/localSearch.html
```

live.com search

The documentation for Microsoft's live.com search APIs are at:

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

The interface is SOAP-based. The WSDL for version 1.1 is:

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

The Getting Started Guide is at

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

You need to set up an API ID (or get existing ones) to use the service at:

```
http://search.msn.com/developer
```

If you have access to Microsoft Visual Studio, I recommend trying out the code samples.

```
http://msdn2.microsoft.com/en-us/library/bb251815.aspx
```

In theory, because of the WSDL interface, you should be able to make use of the live.com in non-Microsoft environments. In practice, you will find it much easier to use Microsoft tools because the documentation and the samples are geared to those tools. To use other tools, I still used Microsoft tools to help me understand the important parameters.

The search parameters for the Live Search API are more complicated than those for the Google SOAP search because the former use complex, nested types. As I describe in Chapter 7, there is a variety of ways to invoke WSDL-described SOAP calls. Some generate language-specific bindings. The one I find the easiest to understand is the approach taken by such tools as the WSDL/SOAP tools in XML Spy and oXygen: feed them the WSDL and they determine the SOAP Connection endpoint, the SOAPaction, and a template for the body. That combination of parameters allows you to call the method without resorting directly to any SOAP libraries.

XML Spy and oXygen are not free – though you can try them out for 30 days free of charge. I don't know of any freeware that makes it quite so easy to work with WSDL and SOAP.

The parameters are confusing. It is not at all clear which parameters are mandatory without studying the WSDL directly – nor what would be valid parameters. For instance, I needed to study

`http://msdn2.microsoft.com/en-us/library/bb266177.aspx`

to get help with the "CultureInfo" field to figure out, for instance, that an acceptable value is, for instance, `en-US` for American English.

Feeding the live.com WSDL to XML Spy, you will get:

* Connection endpoint: `http://soap.search.msn.com:80/webservices.asmx`
* SOAPaction HTTP header is `http://schemas.microsoft.com/MSNSearch/2005/09/fex/Search`
* the following template for a SOAP request

```
<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:Body>
    <m:Search xmlns:m="http://schemas.microsoft.com/MSNSearch/2005/09/fex">
      <m:Request>
        <m:AppID>String</m:AppID>
        <m:Query>String</m:Query>
        <m:CultureInfo>String</m:CultureInfo>
        <m:SafeSearch>Moderate</m:SafeSearch>
        <m:Flags>None</m:Flags>
        <m:Location>
          <m:Latitude>3.14159265358979E0</m:Latitude>
          <m:Longitude>3.14159265358979E0</m:Longitude>
          <m:Radius>3.14159265358979E0</m:Radius>
        </m:Location>
        <m:Requests>
          <m:SourceRequest>
            <m:Source>Web</m:Source>
            <m:Offset>0</m:Offset>
            <m:Count>0</m:Count>
            <m:FileType>String</m:FileType>
            <m:SortBy>Default</m:SortBy>
            <m:ResultFields>All</m:ResultFields>
            <m:SearchTagFilters>
              <m:string>String</m:string>
            </m:SearchTagFilters>
          </m:SourceRequest>
        </m:Requests>
      </m:Request>
    </m:Search>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


If you just enter a key and a search term, no search results will come back. You need to read the documentation and experiment to figure out what's needed. The following SOAP request does work:

```
<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:Body>
    <m:Search xmlns:m="http://schemas.microsoft.com/MSNSearch/2005/09/fex">
      <m:Request>
        <m:AppID>[YOURKEY]</m:AppID>
        <m:Query>flower</m:Query>
        <m:CultureInfo>en-US</m:CultureInfo>
        <m:SafeSearch>Moderate</m:SafeSearch>
        <m:Flags>None</m:Flags>
        <m:Requests>
          <m:SourceRequest>
            <m:Source>Web</m:Source>
          </m:SourceRequest>
        </m:Requests>
      </m:Request>
    </m:Search>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Let's show how to do this with curl:

```
curl -H 'SOAPAction: "http://schemas.microsoft.com/MSNSearch/2005/09/fex/Search"' -d
'<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:Body> <m:Search
xmlns:m="http://schemas.microsoft.com/MSNSearch/2005/09/fex"><m:Request><m:AppID>[YO
URKEY]</m:AppID> <m:Query>flower</m:Query><m:CultureInfo>en-US</m:CultureInfo>
<m:SafeSearch>Moderate</m:SafeSearch> <m:Flags>None</m:Flags><m:Requests>
<m:SourceRequest> <m:Source>Web</m:Source> </m:SourceRequest> </m:Requests>
</m:Request></m:Search></SOAP-ENV:Body></SOAP-ENV:Envelope>'
http://soap.search.msn.com:80/webservices.asmx
```

will return a SOAP message with search results.

Opensearch

```
http://www.opensearch.org/Home
```

The A9 search engine (http://a9.com) created the OpenSearch protocol (http://www.opensearch.org/Home) as a "collection of simple formats for the sharing of search results."

Many websites have their own search boxes; many are also capable of creating RSS and Atom feeds. OpenSearch is a set of extensions that can wrap and existing search

functionality, leveraging the feeds to create lightweight search APIs. The most prominent clients for OpenSearch are the "search plugins" for Firefox 2 and IE 7.

Let's get more concrete. One of the easiest way to learn how to create a search plugin is to use search plugin generator at

```
http://mycroft.mozdev.org/submitos.html
```

Here I use

```
http://blog.mashupguide.net
```

as an example site for which I want to generate a search plugin. I go to the blog to type in a term (e.g., Yahoo) to search to see what URLs come back:

```
http://blog.mashupguide.net/?s=Yahoo&searchsubmit=Find
```

I can then replace `Yahoo` with `{searchTerms}` to generate the "Search URL" for the plugin generator

```
http://blog.mashupguide.net/?s={searchTerms}&searchsubmit=Find
```

You are given the option to register your search plugin. One of the great things about the search plugin wizard is its generation of OpenSearch documents. Here's the one for the mashupguide.net plugin

```
http://mycroft.mozdev.org/installos.php/17890/orangeremix.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/"
xmlns:moz="http://www.mozilla.org/2006/browser/search/">
  <!-- Created on Sun, 17 Jun 2007 17:08:21 GMT -->
  <ShortName>MashupGuide.net</ShortName>
  <Description>Search for info about mashups</Description>
  <Url type="text/html" method="get"
template="http://blog.mashupguide.net/?s={searchTerms}&amp;searchsubmit=Find"/>
  <Image width="16"
height="16">http://mycroft.mozdev.org/updateos.php/id0/orangeremix.png</Image>
  <Developer>Raymond Yee</Developer>
  <InputEncoding>UTF-8</InputEncoding>
  <moz:SearchForm>http://blog.mashupguide.net/</moz:SearchForm>

<moz:UpdateUrl>http://mycroft.mozdev.org/updateos.php/id0/orangeremix.xml</moz:Updat
eUrl>

<moz:IconUpdateUrl>http://mycroft.mozdev.org/updateos.php/id0/orangeremix.png</moz:I
conUpdateUrl>
  <moz:UpdateInterval>7</moz:UpdateInterval>
</OpenSearchDescription>
```

With the OpenSearch XML document in hand, you can then embed some JavaScript to let a user install the plugin. The relevant method is window.external.AddSearchProvider which you find documented at:

* http://msdn2.microsoft.com/en-us/library/Aa744112.aspx (for IE7)

* http://developer.mozilla.org/en/docs/Adding_search_engines_from_web_pages (for Firefox)

You can get a list of search engine plugins at:

* https://addons.mozilla.org/en-US/firefox/browse/type:4 (popular list linked to from within the Firefox Manage Search Engine List widget)
* http://mycroft.mozdev.org/dlstats.html (top downloads)

Note a caveat from `http://mycroft.mozdev.org/contribute.html`:

> *While the implementation of Sherlock in Mozilla-based browsers only supported GET requests, the introduction of OpenSearch has also allowed POST requests to be used but unfortunately this is not currently supported in IE7.*

If you want to get meta with WordPress, you use the following WP plugin to generate a search plugin:

`http://inner.geek.nz/projects/wordpress-plugins/mycroft-search-plugin-generator/`

There is another half to the OpenSearch specification. If the search results that come out of the search engine are in RSS 2.0 or Atom 1.0 format, wrapped with special elements documented at:

`http://www.opensearch.org/Specifications/OpenSearch/1.1#OpenSearch_response_elements`

the search results can then be consumed and presented by search clients that support the OpenSearch protocol:

`http://www.opensearch.org/Community/OpenSearch_search_clients`

and programming libraries that can use it:

`http://www.opensearch.org/Community/OpenSearch_software`

In other words, you can get lightweight APIs for these sources and build meta-search systems out of them. In the specific case of WordPress search results, you can make WP in to a full OpenSearch source using a WP plugin like

`http://williamsburger.com/wb/archives/opensearch-v-1-1`

Looking Beyond Google

It's useful to look at the world of search engines beyond the very top one. At the very least, I want to explore some "top search engines" – starting with the ones on Danny Sullivan's analysis:

`http://searchenginewatch.com/showPage.html?page=2156221#consider`

Then it would be fun to work through

`http://www.readwriteweb.com/archives/top_100_alternative_search_engines.php`

Google Desktop HTTP/XML gateway

If you find the Google Desktop useful, you might be glad to know that you can access results programatically via a HTTP/XML gateway, documented at:

`http://desktop.google.com/dev/queryapi.html#httpxml`

There is also a COM-based interface in Windows: http://desktop.google.com/dev/queryapi.html#registering The XML gateway works on Mac OS X in Google Desktop Mac 1.0.3 +.

On Windows, you get the query URL from the registry key

`HKEY_CURRENT_USER\Software\Google\Google Desktop\API\search_url`

which will be of the form

`http://127.0.0.1:4664/search&s=[SECRETKEY]?q=`

You can get XML out by tacking on `&format=xml`

A sample query is

`http://127.0.0.1:4664/search&s=[SECRETKEY]?q=bach`

which returns the following (excerpted here):

```
<results count="447">
–
       <result>
<category>web</category>
<doc_id>247278</doc_id>
<event_id>277975</event_id>
–
       <title>
Eventful - Mountain View Events - Mashup Camp IV at Computer History Museum
</title>
<url>http://eventful.com/events/E0-001-002642665-0</url>
<flags>259</flags>
<time>1282630246734300O0</time>
–
       <snippet>
Add to Reddit Add to calendar Eventful calendar Add to Calendar: <b>Bach</b> in San
Francisco metro area Berkeley, California, USA My Events Add to
</snippet>
–
       <thumbnail>
/thumbnail?id=6%5F76xk4cxwsgMBAAAA&s=KLp8LKWLzFxwQ25pvDi42EHVfTk
</thumbnail>
–
       <icon>
/icon?id=http%3A%2F%2Feventful%2Ecom%2F&s=YtdjKx9s9jRBxC11CW7vm377nNO
</icon>
```


_

```
      <cache_url>
http://127.0.0.1:4664/redir?url=http%3A%2F%2F127%2E0%2E0%2E1%3A4664%2Fcache%3Fevent%
5Fid%3D277975%26schema%5Fid%3D2%26q%3Dbach%26s%3DuSIdPgul9xWiUyUybC6Ko3XA2cI&src=1&s
chema=2&s=uADtUWTU45Sf6jKTCjeexKOwxjY
</cache_url>
</result>
```