CHAPTER 19

Integrating Search

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.

This chapter shows how to use the Google, Yahoo!, and Live.com search APIs, as well as configuring searchable web sites for access as a search plug-in in Firefox 2.0 or Internet Explorer 7 using OpenSearch. This chapter will also examine briefly how to use the Google Desktop Search API.

Google Ajax Search

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

The Google Ajax Search API (`http://code.google.com/apis/ajaxsearch/`) gives you a search widget that you can embed in your web site. You can access functionality for searching the Web, doing local searches (tied to maps), and doing video searches. The widget displays a search box and takes care of displaying search results in an HTML element that you designate.

Like Google Maps, you have to sign up for a key that is tied to a specific directory; you can do that here:

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

Paste the "Hello, World" code into your page, and load it.[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]

1. http://examples.mashupguide.net/ch19/google.ajax.1.html
2. http://examples.mashupguide.net/ch19/google.ajax.2.html

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

    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
      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 initial 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. Here are some noteworthy extras:

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

Indeed, you can learn plenty of things for your specific 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 an HTML interface, take a look specifically at the following:

`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

The Yahoo! Search API (`http://developer.yahoo.com/search/`) is a RESTful one. I'll now show how to use the Yahoo! Search API.

You need an application ID, which you get from here:

`https://developer.yahoo.com/wsregapp/index.php`

You can see your registered apps here:

`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 can 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`), you can see a sample query:

`http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&~CCC`

```
query=madonna&results=2
```

If you substitute your own API key and search for `flower`, you'll come up with the following query:

```
http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=[YourAppID]&
~CCC
query=flower&results=1
```

An excerpt of the search results follows:

```
<Result>
  <Title>1-800-FLOWERS.COM - Official Site</Title>
  <Summary>1-800-Flowers delivers flowers and floral arrangements, gift baskets,
~CCC
    gourmet treats, or other presents for anniversaries, birthdays, and
special~CCC
    occasions. Order online, over the phone, or by visiting a store location.
  </Summary>
  <Url>http://www.1800flowers.com/</Url>
<ClickUrl>http://uk.wrs.yahoo.com/_ylt=~CCC
AOJe5VZ47HdGmOQAzhvdmMwF;_ylu=X3oDMTB2cXVjNTM5BGNvbG8DdwRsA1dTMQRwb3MDMQRzZWMDc3
IEdn~CCC
RpZAM-/SIG=19qu9j9dq/EXP=1182350840/**http%3A//rdrw1.yahoo.com/click%3
    Fu=http%3A//clickserve.cc-dt.com/link/click%253Flid%253D41000000011562437%26
    y=04765B7ED3D00A0BB4%26i=482%26c=37687%26q=02%255ESSHPM%255BL7ysphzm6%26
    e=utf-8%26r=0%26d=wow~WBSV-en-
us%26n=LP94K1LESHRKDFP3%26s=3%26t=%26m=4677EC78%26
    x=057E49A7F20A924F7B2C30A7101C217A96</ClickUrl>
  <DisplayUrl>www.1800flowers.com/</DisplayUrl>
  <ModificationDate>1181631600</ModificationDate>
  <MimeType>text/html</MimeType>
</Result>
```

The parameters for this RESTful interface are documented here:

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

I find it interesting that there is a published W3C XML Schema published for the response:

```
http://search.yahooapis.com/WebSearchService/V1/WebSearchResponse.xsd
```

There are also API Kits for Yahoo! Search; you may find one for your favorite language. They are BSD-licensed:

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

Yahoo! Images

The documentation for Yahoo!'s image search is at the following location:

```
http://developer.yahoo.com/search/image/V1/imageSearch.html
```

Note the sample search:

```
http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=YahooDemo&~
CCC
```

```
query=Corvette&results=2
```

You can substitute your own key and search term. For example, you can use this:

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

and receive an XML response similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:yahoo:srchmi"
  xsi:schemaLocation="urn:yahoo:srchmi http://api.search.yahoo.com/~CCC
ImageSearchService/V1/ImageSearchResponse.xsd"
  totalResultsAvailable="5446610" totalResultsReturned="2"
firstResultPosition="1">
  <Result>
    <Title>Flower.jpg</Title>
    <Summary>Flower.jpg</Summary>
    <Url>http://home.mchsi.com/~gentle501/images/Flower.jpg</Url>
    <ClickUrl>http://home.mchsi.com/~gentle501/images/Flower.jpg</ClickUrl>
    <RefererUrl>http://home.mchsi.com/~gentle501/pages/Flower.html</RefererUrl>
    <FileSize>104755</FileSize>
    <FileFormat>jpeg</FileFormat>
    <Height>800</Height>
    <Width>771</Width>
    <Thumbnail>
      <Url>http://sp1.mm-a7.yimg.com/image/3966820083</Url>
      <Height>155</Height>
      <Width>149</Width>
    </Thumbnail>
  </Result>
  <Result>
    <Title>dca_sunshine_flower.jpg</Title>
    <Summary>Sunshine Flower Sunday, 14 Nov 2004 | Disneyland , Flora A flower
taken~CCC
at Disney's California Adventure. Nikon D100 | 50mm f/1.4 D | 50mm | 1/250 sec
|~CCC
f/2.5 | ISO 200 | 26 Jun 2004</Summary>
    <Url>http://www.disneymike.com/photoblog/dca_sunshine_flower.jpg</Url>


<ClickUrl>http://www.disneymike.com/photoblog/dca_sunshine_flower.jpg</ClickUrl>


<RefererUrl>http://www.disneymike.com/photoblog/archives/2004/11/sunshine_flower
~CCC
.html</RefererUrl>
    <FileSize>311603</FileSize>
    <FileFormat>jpeg</FileFormat>
    <Height>635</Height>
    <Width>700</Width>
    <Thumbnail>
      <Url>http://sp1.mm-a4.yimg.com/image/2928630219</Url>
```

```
      <Height>136</Height>
      <Width>150</Width>
    </Thumbnail>
  </Result>
</ResultSet>
```

Yahoo! Local Search has a similar architecture:

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

Microsoft Live.com Search

Microsoft's Live Search APIs (http://msdn2.microsoft.com/en-us/library/bb251794.aspx) are SOAP-based. The WSDL for version 1.1 is as follows:

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

The Getting Started Guide is located here:

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

You need to set up an API ID (or get an existing one) to use the service; you can do this at the following location:

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

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

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

There are Express editions of Microsoft Visual Studio that are available for a free download:

`http://www.microsoft.com/express/`

Note In theory, because of the WSDL interface, you should be able to use 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 refer to 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 uses complex, nested types. As I described in Chapter 7, there are 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.

Note XML Spy and oXygen are not free, although you can try them for 30 days free of charge. I don't

know of any freeware (except perhaps Eclipse) that makes it quite so easy to work with WSDL and SOAP.

The search parameters are confusing, and it is not at all clear which parameters are mandatory without studying the WSDL directly; it's also not clear what the valid parameters would be. For instance, I needed to study the following:

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

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

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

* Connection endpoint: `http://soap.search.msn.com:80/webservices.asmx`
* SOAPaction HTTP header:
  `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. To figure out which parameters in the SOAP request are required and the range of possible values, start by reading this:

`http://msdn2.microsoft.com/en-us/library/bb266182.aspx`

which distinguishes between the following required parameters:

* `AppID`: Your application key
* `CultureInfo`: Language and regional information that must be chosen from a list of possible values[3] (for example, `en-US`)
* `Query`: Your search term
* `Requests`: A list of `SourceRequest` values drawn from a set of possible values[4] (for example, `Web`, `Ads`, `Image`)

and the following optional parameters:

* `Flags`: One of `None`, `DisableHostCollapsing`, `DisableSpellCheckForSpecialWords`, or `MarkQueryWord` (`None` is the default value)
* `Location`: The latitude, longitude, and optional search radius for the search
* `SafeSearch`: One of `Strict`, `Moderate`, or `Off` (`Moderate` is the default value)

Here's a sample SOAP request that searches the Web for `flower` in the American English context:

```
<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>
```

3. http://msdn2.microsoft.com/en-us/library/bb266177.aspx
4. http://msdn2.microsoft.com/en-us/library/bb266167.aspx

This shows how to do this with `curl`:

```
curl -H 'SOAPAction:
"http://schemas.microsoft.com/MSNSearch/2005/09/fex/Search"' ~CCC
-d '<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"~CCC
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"~CCC
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ~CCC
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><SOAP-ENV:Body> <m:Search~CCC
xmlns:m="http://schemas.microsoft.com/MSNSearch/2005/09/fex"><m:Request><m:AppID
>~CCC
[YOURKEY]</m:AppID> <m:Query>flower</m:Query><m:CultureInfo>en-
US</m:CultureInfo>~CCC
<m:SafeSearch>Moderate</m:SafeSearch> <m:Flags>None</m:Flags><m:Requests>~CCC
<m:SourceRequest> <m:Source>Web</m:Source> </m:SourceRequest> </m:Requests>~CCC
</m:Request></m:Search></SOAP-ENV:Body></SOAP-ENV:Envelope>'
http://soap.search.msn.com:80/webservices.asmx
```

This will return a SOAP message with search results:

```
<?xml version="1.0" encoding="utf-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <SearchResponse xmlns="http://schemas.microsoft.com/MSNSearch/2005/09/fex">
      <Response>
        <Responses>
          <SourceResponse>
            <Source>Web</Source>
            <Offset>0</Offset>
            <Total>192000000</Total>
            <Results>
              <Result>
                <Title>Flowers, Roses, Plants, Gift Baskets - 1-800-FLOWERS.COM
-
Your ... </Title>
                <Description>Florist and gift retailer and franchisor with more
than~CCC
100 stores nationwide offering online purchasing of arrangements, plants,
gift~CCC
baskets, confections and gourmet foods ... </Description>
                <Url>http://www.1800flowers.com/</Url>
              </Result>
              <Result>
                <Title>Flowers, plants, roses, &amp;amp; gifts. Flower delivery
with~CCC
fewer handlers ... </Title>
                <Description>Flowers, roses, plants and gift delivery. Order
flowers~CCC
from ProFlowers once, and you&amp;apos;ll never use flower delivery from
florists~CCC
again</Description>
                <Url>http://www.proflowers.com/</Url>
              </Result>
[...]
```

```
            </Results>
          </SourceResponse>
        </Responses>
      </Response>
    </SearchResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

OpenSearch

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 web sites have their own search boxes; many are also capable of creating RSS and Atom feeds. OpenSearch is a set of extensions that can wrap existing search functionality, leveraging the feeds to create lightweight search APIs. The most prominent clients for OpenSearch are the search plug-ins for Firefox 2 and Internet Explorer 7.

Let's get more concrete. One of the easiest ways to learn how to create a search plug-in is to use the search plug-in generator at the Mozilla Mycroft project (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 plug-in. I go to the blog to type in a term (for example, **Yahoo**) to search on and 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 plug-in generator:

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

You are given the option to register your search plug-in. One of the great features of the search plug-in wizard is its generation of OpenSearch documents. Here's the one for the Mashupguide.net plug-in (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:UpdateUrl>
  <moz:IconUpdateUrl>
    http://mycroft.mozdev.org/updateos.php/id0/orangeremix.png
  </moz:IconUpdateUrl>
  <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 plug-in. The relevant method is `window.external.AddSearchProvider()`, which you find documented here:

* `http://msdn2.microsoft.com/en-us/library/Aa744112.aspx` (for Internet Explorer 7)
* `http://developer.mozilla.org/en/docs/Adding_search_engines_from_web_pages` (for Firefox)

You can get a list of search engine plug-ins here:

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

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

> *While the implementation of Sherlock [the legacy Apple search tool] 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.*

You can use the following WordPress plug-in to generate a search plug-in:

`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 here:

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

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

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

and by 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 metasearch systems from them. In the specific case of WordPress search results, you can make WordPress into a full OpenSearch source using a WordPress plug-in, such as the following:

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

Google Desktop HTTP/XML Gateway

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

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

Note There is also a COM-based interface in Windows, located at http://desktop.google.com/dev/queryapi.html#registering. The XML gateway works on Mac OS X in Google Desktop Mac 1.0.3+. The API is currently unsupported for the Linux version of Google Desktop.

On Windows, you get the query URL from the registry key using this:

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

The query URL will be of the following 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 as follows:

```
http://127.0.0.1:4664/search&s={SECRETKEY}?q=bach
```

This query 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>128263024673430000</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%3~C
CC
Fevent%5Fid%3D277975%26schema%5Fid%3D2%26q%3Dbach%26s%3DuSIdPgul9xWiUyUybC6Ko3XA
2cI~CCC
&src=1&schema=2&s=uADtUWTU45Sf6jKTCjeexKOwxjY
    </cache_url>
  </result>
```

Summary

In this chapter, you learned the basics of using APIs for Google Ajax Search, Yahoo! Search, Yahoo! Image Search, and Microsoft Live.com for searching content on the Web. You looked at how you can use OpenSearch to wrap existing search functionality so that it can be accessed in search bars for web browsers. Finally, I presented an example of an API for desktop search by outlining the Google Desktop HTTP/XML gateway.