Building Web service applications with the Google API

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Introduction

Should I take this tutorial?

The Google search engine can now be accessed via a SOAP-based Web service. This means that developers can now embed Google search results and other information into their own applications. Google also took this project one step further, creating an API and Java technology toolkit for accessing the data. This tutorial is for developers who want to use Google information from within their Java applications.

Developers need to be familiar with Java programming language, and will need an Internet connection to access the Google service. An understanding of Web services is not required to use the API.

What is this tutorial about?

Interest in Web services is booming, and now there is a practical application for them accessible to those outside large enterprise environments. Google has lately become the most popular search engine on the Web, and now they have made their index of 2 billion pages available to developers via a SOAP-based API in order to enable applications such as new content watchdogs, GUI search tools, and pattern analysis.

The Google API enables you to perform searches, retrieve cached pages from Google, and utilize Google's spell-checking capabilities. All of these capabilities can then be integrated into any application in an environment that supports Web services.

This tutorial demonstrates the creation of four separate applications:

- The first sends a search query to the Web service and retrieves and analyzes the individual and aggregated results using only Java skills.
- The second application retrieves a specific Web page from the Google cache.
- The third demonstrates Google's spelling suggestion capabilities.
- The last part of the tutorial discusses the different SOAP messages used by the Web service, and the process of sending them directly to the Google Web service. This fourth application uses JAXM to send and receive the messages, which are then transformed into useful results.

Tools

Make sure that the following tools are installed and tested before beginning the tutorial.

- Java ™ 2 SDK, Standard Edition version 1.3.1 or higher: The sample applications

demonstrate manipulation of the DOM through Java technology. You can download the Java SDK from *http://java.sun.com/j2se/*.

* The Google API, available at *http://www.google.com/apis/*.
* The last section of this tutorial deals with sending SOAP messages directly, rather than through the API. To run these samples, you'll need the Java API for XML Mess aging, or JAXM. It's available as part of the Java XML Pack, at *http://java.sun.com/xml/downloads/javaxmlpack.html*.

About the author

Nicholas Chase has been involved in Web site development for companies such as Lucent Technologies, Sun Microsystems, Oracle, and the Tampa Bay Buccaneers. Nick has been a high school physics teacher, a low-level radioactive waste facility manager, an online science fiction magazine editor, a multimedia engineer, and an Oracle instructor. More recently, he was the Chief Technology Officer of Site Dynamics Interactive Communications in Clearwater, Florida, and is the author of three books on Web development, including *Java and XML From Scratch* (Que) and the upcoming *Primer Plus XML Programming* (SAM). He loves to hear from readers and can be reached at *nicholas@nicholaschase.com*.

Section 2. What is the Google API?

What is Google?

Google is a search engine that indexes Web sites, Usenet news groups, and news sources. There are billions of pages on the Web, and the challenge in producing a useful search result is in figuring out which of them is truly relevant to the user.

Google focuses not on the number of responses, but on their relevance. The algorithm is proprietary, but it basically works on the theory that if a page is useful, other pages covering the same topic will link to it. In other words, if a search for German shepherds finds 1000 results, and one of those pages is linked to by 500 of them, it is probably a good source for information on the dogs; as a result, it will appear in a higher position than one that is not linked at all but perhaps contains a passing reference to someone's pet.

Google allows for fairly sophisticated searches, with required and forbidden words, and the ability to restrict results based on language or encoding, among other things.

Google and Web services

Google has a long tradition of allowing access to its service through other means, such as the Google Toolbar and wireless searches. But now the company has made its index available to other developers through a Web services interface. This means that developers can programmatically send a request to the Google server and get back a response.

These requests have all of the same parameters as a normal Google search that a user might enter from the Web interface, but as a programmer you can control them from within your application. What's more, the results come back in the form of a SOAP message, so you can manipulate them any way you please, rather than simply having to use the results as they are presented on the Google site.

You can also make use of other information less obvious than that produced by a traditional search. For example, you could determine how many results were present for a particular search term on a particular date, tracking trends over time. One of the services that Google has made available through the Web services interface is their spelling suggestion server. This means that you could write an application that checks users' spelling and makes suggestions using Google's engine

Requests and responses take the form of SOAP messages.

SOAP and the API

Ultimately, Google services are exposed via SOAP messages. For example, to request a spelling suggestion, you might send the message:

```
<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:doSpellingSuggestion xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <phrase xsi:type="xsd:string">pnewmonia</phrase>
    </ns1:doSpellingSuggestion>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

and get a response of

```
<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:doSpellingSuggestionResponse
xmlns:ns1="urn:GoogleSearch"SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">

      <return xsi:type="xsd:string">pneumonia</return>
    </ns1:doSpellingSuggestionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You would then have to manipulate the data manually.

Alternatively, you could use the Google API, creating an application something like:

```
GoogleSearch search = new GoogleSearch();

search.setKey("00000000000000000000000000000000");

String suggestion = search.doSpellingSuggestion(spellingRequest);
System.out.println(suggestion);
```

Google API limitations

At the time of this writing, the Google API is still in beta, and some fairly strict limits have been set for its use. The API Terms of Service basically states that it's available for personal, non-commercial use, so you can't mirror their results on your Web site in order to increase traffic. Make sure to read the full document before signing up for a key.

From a technical standpoint, keep the following points in mind:

First and foremost, all searches require a license key, which is free. This key enables up to 1000 searches per day. To obtain a key, visit

https://www.google.com/accounts/NewAccount?continue=
http://api.google.com/createkey&followup=http://api.google.com/createkey.

Searches will return up to 10 results at a time.

Only the first 1000 results are available through this service. (So a request can find results number 990 through 1000, but not 1001.)

Section 3. Creating a Search

Required classes

Using the Google API from Java involves making sure that the appropriate classes are available. Google has conveniently bundled them into the `googleapi.jar` file, which is included as part of the distribution. This file includes:

- The Java wrapper classes for the SOAP class, in the `package com.google.soap.search`.
- The Java Activation Framework 1.0.1, normally found in `activation.jar`.
- The JavaMail ™ API, normally found in `mailapi.jar`.
- The Apache SOAP 2.2 toolkit, normally found in `apache-soap-22.jar`.
- Apache Crimson 1.1.3, normally found in `crimson.jar`. (Note that this is Crimson, NOT Xerces. The two handle XML similarly, but not identically.)

Note that the Java API for XML Mess aging, JAXM, is not included in `googleapi.jar`, but it is not necessary for using the API.

Create the search

Here, you'll start with actually creating a simple search an executing it, then examining the results. The examples will show simple applications that output to the command line; what you do with the final data is up to you.

```
import com.google.soap.search.GoogleSearch;

public class GoogleSearchTutorial {

    public static void main (String[] args) {

        String searchTerm = args[0];

        GoogleSearch search = new GoogleSearch();

        search.setKey("000000000000000000000000000000000");
        search.setQueryString(searchTerm);

    }

}
```

The main object here is the `GoogleSearch`.To use it, you will set parameters and then execute the actual search to return the results.

Note that `setKey()` is how you send your personal license key to the application. Replace the zeroes with the key you were sent when you signed up for a Google account.

The only other parameter required for executing a search is the search term itself. I'll talk more about constructing searches in Constructing searches on page 10 , but for now understand that you can search for just about any word or phrase.

Once you've set the key and the search term, you're ready to actually execute the search.

Execute the search

Actually executing the search is fairly straightforward:

```
import com.google.soap.search.GoogleSearch;
import com.google.soap.search.GoogleSearchResult;
        import com.google.soap.search.GoogleSearchFault;

public class GoogleSearchTutorial {

   public static void main (String[] args) {

        String searchTerm = args[0];

         try {

           GoogleSearch search = new GoogleSearch();

           search.setKey("00000000000000000000000000000000");
           search.setQueryString(searchTerm);

            GoogleSearchResult result = search.doSearch();

           int numResults = result.getEstimatedTotalResultsCount();

           if ( result.getEstimateIsExact() ){
              System.out.println("Number of results: "+numResults);
           } else {
              System.out.println("Estimated number of results:"+numResults);

           }
                 } catch (GoogleSearchFault gsf) {
                     System.out.println("Google Search Fault: "+gsf.getMessage()
                 }

   }

}
```

The `doSearch()` method does the search based on the query set in the previous panel and returns a `GoogleSearchResult` object that carries methods used to access its data. One particularly useful bit of information is the number of results and whether that number is exact.

For example, if this application were compiled and run with a command-line argument of `"science fiction"` (including the quotes) it would return the text:

```
Estimated number of results: 1790000
```

Any number of things can go wrong in requesting information from a Web service, and the Google API includes a `GoogleSearchFault` class that acts as a wrapper for any exceptions that might come up.

Retrieving results

The results themselves are easy to retrieve. The `GoogleSearchResult` object provides an easy means for looking at the results themselves:

```
...
        search.setKey("00000000000000000000000000000000");
        search.setQueryString(searchTerm);

        GoogleSearchResult result = search.doSearch();
        System.out.println( result.toString() );

    } catch (GoogleSearchFault gsf) {
...
```

This overridden `toString()` method provides a convenient means for looking at the results. For example, the same search for `"science fiction"` returns results such as these:

```
 [
  URL  = "http://www.sfsite.com/home.htm"
  Title = "The SF Site: The Best in <b>Science</b>
<b>Fiction</b> and Fantasy"
  Snippet = " <b>...</b> Magazine of Fantasy &amp;
<b>Science</b> <b>Fiction</b>
, April 2002 The Magazine of Fantasy &amp; <b>Science</b
><br> <b>Fiction</b>, April 2002 reviewed by David Soyka Charles
        Coleman Finlay cover <b>...</b>   "
  Directory Category = {SE="", FVN=""}
  Directory Title = ""
  Summary = ""
  Cached Size = "36k"
  Related information present = true
  Host Name = ""
  ],

  [
  URL  = "http://www.asimovs.com/"
  Title = "Asimov's <b>Science</b> <b>Fiction</b>"
  Snippet = "Asimov's SF T-shirts Quantities are limited, so buy your own<br> As
imov's <b>Science</b> <b>Fiction</b> t-shirt while supplies last!
<b>...</b>    "
  Directory Category = {SE="", FVN="Top/Arts/Literature/Genres/
        Science_Fiction/Magazines_and_E-zines"}
  Directory Title = "Asimov's <b>Science</b> <b>Fiction</b> "
  Summary = "Essays by Isaac Asimov, stories, book and net reviews,
        readers' forum and chats, writers guidelines,... "
  Cached Size = "31k"
  Related information present = true
  Host Name = ""
  ],

  [
```

```
  URL  = "http://worldcon.org/"
  Title = "World <b>Science</b> <b>Fiction</b> Society /
Worldcon [Official]"
  Snippet = "World <b>Science</b> <b>Fiction</b> Society World
<b>Science</b> <b
>Fiction</b> Convention.<br> [updated 7 november 2001]. Upcoming Worldcons. <b>.
..</b>    "
  Directory Category = {SE="",
FVN="Top/Arts/Genres/Science_Fiction_and_Fantasy/
Fandom/Conventions_and_Organizations/WorldCon"}
  Directory Title = "World <b>Science</b> <b>Fiction</b> Society /
WorldCon "
  Summary = "Website with some basic information, schedule of upcoming WorldCons
, and links to the websites of... "
  Cached Size = "10k"
  Related information present = true
  Host Name = ""
  ],
...
```

This partial listing shows some of the information available for each result. I'll look at extracting (and controlling) some of this information starting with Retrieving the result set on page 12 the result set, but first take a look at some of the factors that affect a Google search itself.

Constructing searches

A big part of finding what you're looking for in any search involves knowing what to ask for. This holds true on Google as well. The API searches use the same algorithm as the Web searches, so it helps to understand it.

When searching for more than one word, Google returns only those pages that include all terms, though not necessarily in the order in which they were requested. For example, the search term

```
science fiction
```

returns any page that includes the words "science"and "fiction" anywhere in the body, title, or URL, whether or not they appear as a unit. To make them appear as a unit, the search term would have to be

```
"science fiction"
```

The quotes force Google to look at the term as a phrase. To search for pages that include one word or the other, use

```
science OR fiction
```

Note that the OR must be capitalized, or it will be considered a *stop word*.

Stop words and other search modifiers

Stop words are common words that are omitted from a query. They include `the`, `and`, `a`, `or`, and `how`, among others. To force a stop word to be included in a query, precede it with a plus sign (+), as in

```
"+the science +of fiction"
```

Note that simply enclosing the phrase in quotes is not always enough to force inclusion of common words.

Similarly, you can exclude terms using the minus sign (-), as in

```
science -fiction
```

which returns only results about science.

Google also looks for a number of different modifiers, which determine where it should look for a search term. These modifiers include:

```
intitle:
allintitle:
inurl:
allinurl:
allintext:
allinlinks:
filetype:
link:
daterange:
```

These modifiers can be combined with other search terms to restrict the result set. For example, the query

```
"science fiction" daterange:2452365-2452395
```

finds all pages that contain the phrase `"science fiction"` that were in the index between April 1, 2002 and April 30, 2002. (Note that it is possible to search the index for old information.)

Section 4. Search results

Retrieving the result set

Now you can start to look at individual results; the `GoogleSearchResult` object is actually an aggregation of all of the results returned by the query:

```
...
import com.google.soap.search.GoogleSearchResultElement;

public class GoogleSearchTutorial {
...
        GoogleSearchResult result = search.doSearch();

        GoogleSearchResultElement[] resultElements = result.getResultElements(
        System.out.println("Start index = "+ result.getStartIndex() );
        System.out.println("End index = "+ result.getEndIndex() );

    } catch (GoogleSearchFault gsf) {
...
```

The `getResultElements()` method returns an array of `GoogleSearchResultElement` objects, each of which has a place in the overall result set. This is a 1-based index, so a search that shows 1790000 results will return an array of `GoogleSearchResultElement` objects starting with number `1` and ending with number `10`. (Remember, each request returns a maximum of 10 results.)

Accessing individual results

Accessing each individual result is as simple as looping through the array:

```
import com.google.soap.search.GoogleSearch;
import com.google.soap.search.GoogleSearchResult;
import com.google.soap.search.GoogleSearchFault;
import com.google.soap.search.GoogleSearchResultElement;

public class GoogleSearchTutorial {

    public static void main (String[] args) {

        String searchTerm = args[0];

        try {

            GoogleSearch search = new GoogleSearch();

            search.setKey("00000000000000000000000000000000");
            search.setQueryString(searchTerm);

            GoogleSearchResult result = search.doSearch();

            GoogleSearchResultElement[] resultElements = result.getResultElements();
                int startIndex = result.getStartIndex() - 1;
```

```
                int endIndex = result.getEndIndex() - 1;

                for (int i = startIndex; i &lt;= endIndex; i++) {

                    GoogleSearchResultElement resultElement = resultElements[(i)];

                    String title = resultElement.getTitle();
                    String url = resultElement.getURL();

                    System.out.println(title);
                    System.out.println(url);
                    System.out.println("");
             }

        } catch (GoogleSearchFault gsf) {
            System.out.println("Google Search Fault: "+gsf.getMessage());
        }
    }
}
```

Because the array is zero-based, subtract one to get the actual start and end values.

Displaying individual results

A `GoogleSearchResultElement` object represents each result within the array, and
this object has methods for accessing information such as the title, and information, as
seen here. For the `"science fiction"` search, the results are something like:

```
<b>Science</b> <b>Fiction</b> Weekly
http://www.scifi.com/sfw/

SCIFI.COM
http://www.scifi.com/

<b>Science</b> <b>Fiction</b> and Fantasy Writers of America, Inc.
http://www.sfwa.org/

The Linköping <b>Science</b> <b>Fiction</b> &amp; Fantasy Archive
http://www2.lysator.liu.se/sf_archive/

Asimov's <b>Science</b> <b>Fiction</b>
http://www.asimovs.com/

The SF Site: The Best in <b>Science</b> <b>Fiction</b> and Fantasy
http://www.sfsite.com/home.htm

World <b>Science</b> <b>Fiction</b> Society / Worldcon [Official]
http://worldcon.org/

Analog <b>Science</b> <b>Fiction</b> &amp; Fact
http://www.analogsf.com/

ULTIMATE <b>SCIENCE</b> <b>FICTION</b> WEB GUIDE
http://www.magicdragon.com/UltimateSF/SF-Index.html

Feminist <b>Science</b> <b>Fiction</b>, Fantasy &amp; Utopia
http://www.feministsf.org/femsf/
```

Note that in a Web search, the search terms are highlighted in the results using the bold tag; this information carries through to the results returned from the Web service.

Directory categories

Google returns more than just site results. It also returns the Google category for each result:

```
...
import com.google.soap.search.GoogleSearchDirectoryCategory;

public class GoogleSearchTutorial {
...
        for (int i = startIndex; i <= endIndex; i++) {
            GoogleSearchResultElement resultElement = resultElements[(i-1)];
            String title = resultElement.getTitle();
            String url = resultElement.getURL();

        GoogleSearchDirectoryCategory category =
                        resultElement.getDirectoryCategory();
        String categoryViewable = category.getFullViewableName();
        String categoryString = category.toString();

            System.out.println(categoryViewable + " " + categoryString);
            System.out.println(title);
            System.out.println(url);
            System.out.println("");
        }

...
```

The GoogleSearchDirectoryCategory object includes information on the Google category name, represented as a formatted string, and on the Open Directory Project category name, returned by the getFullViewableName(). The Open Directory project is covered in Open Open Directory Information on page 16 .

Running the application returns the category information before each result:

```
Top/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines
  {SE="", FVN="To
p/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines"}
<b>Science</b> <b>Fiction</b> Weekly
http://www.scifi.com/sfw/

Top/Arts/Genres/Science_Fiction_and_Fantasy
  {SE="", FVN="Top/Arts/Genres/Science
_Fiction_and_Fantasy"}
SCIFI.COM
http://www.scifi.com/

Top/Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy
  {SE="", FVN="Top/Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy"}
<b>Science</b> <b>Fiction</b> and Fantasy Writers of America, Inc.
http://www.sfwa.org/
```

```
Top/Arts/Literature/Genres/Science_Fiction
   {SE="", FVN="Top/Arts/Literature/Genres/Science_Fiction"}
The Linköping <b>Science</b> <b>Fiction</b> &amp; Fantasy Archive
http://www2.lysator.liu.se/sf_archive/
...
```

Snippets and context

Part of determining whether a result is appropriate is seeing the search terms in the
context of the rest of the page. The API allows you to retrieve the "snippet", or portion
of the page that triggered the result's inclusion.

```
...
            String categoryViewable = category.getFullViewableName();

            String snippet = resultElement.getSnippet();

            System.out.println(categoryViewable + " " + categoryString);
            System.out.println(title);
            System.out.println(url);
            System.out.println("     " + snippet );
            System.out.println("");
        }
...
```

Note that the snippet doesn't always include the search terms. For example, your query
returns:

```
Top/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines
   {SE="", FVN="Top/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines"}
<b>Science</b> <b>Fiction</b> Weekly
http://www.scifi.com/sfw/
<b>...</b> Hollywood will choose to adapt better SF, and
more. <b>...</b>
 (c)<br> Copyright 2002, <b>Science</b> <b>Fiction</b> Weekly
(tm).

Top/Arts/Genres/Science_Fiction_and_Fantasy
   {SE="", FVN="Top/Arts/Genres/Science_Fiction_and_Fantasy"}
SCIFI.COM
http://www.scifi.com/
[ Farscape/Stargate SG-1 ]. Apr 28, 2002, Today's<br> News Hodder Talks
   More 'Daredevil' 'Sub-Mariner <b>...</b>

Top/Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy {SE="", FVN="Top/
Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy"}
<b>Science</b> <b>Fiction</b> and Fantasy Writers of America, Inc.
http://www.sfwa.org/
<b>...</b> not of SFWA.
Except where otherwise noted, content and design c    ]
copyright 1995-2002<br> by <b>Science</b> <b>Fiction</b> and Fantasy Writers of
 America, Inc. (&quot;SFWA&quot;). SFWA <b>...</b>
...
```

Again, notice that the search terms, if they do appear, are highlighted using the bold
tag. Note also that the snippet is HTML from the page.

Open Directory Information

The Open Directory Project is a volunteer-driven effort to catalog worthwhile sites on the Internet. Rather than an index generated by automated crawlers, the ODP is a human-edited list of sites. The ODP database is available free of charge, and Google makes use of its information within its results. The API can directly retrieve this information.

```java
import com.google.soap.search.GoogleSearch;
import com.google.soap.search.GoogleSearchResult;
import com.google.soap.search.GoogleSearchFault;
import com.google.soap.search.GoogleSearchResultElement;
import com.google.soap.search.GoogleSearchDirectoryCategory;

public class GoogleSearchTutorial {

    public static void main (String[] args) {

        String searchTerm = args[0];

        try {

            GoogleSearch search = new GoogleSearch();

            search.setKey("00000000000000000000000000000000");
            search.setQueryString(searchTerm);

            GoogleSearchResult result = search.doSearch();

            GoogleSearchResultElement[] resultElements = result.getResultElements();
            int startIndex = result.getStartIndex();
            int endIndex = result.getEndIndex();
            for (int i = startIndex; i <= endIndex; i++) {
                GoogleSearchResultElement resultElement = resultElements[(i-1)];
                String title = resultElement.getTitle();
                String url = resultElement.getURL();

                GoogleSearchDirectoryCategory category =
                          resultElement.getDirectoryCategory();
                String categoryString = category.toString();
                String categoryViewable = category.getFullViewableName();

                String snippet = resultElement.getSnippet();

                String directoryTitle = resultElement.getDirectoryTitle();
                String directorySummary = resultElement.getSummary();

                System.out.println(categoryViewable + " " + categoryString);
                System.out.println(title);
                System.out.println(url);
                System.out.println("    " + snippet);
                System.out.println("Open Directory Information:");
                System.out.println("    " + directoryTitle);
                System.out.println("    " + directorySummary);
                System.out.println("");
            }

        } catch (GoogleSearchFault gsf) {
```

```
                System.out.println("Google Search Fault: "+gsf.getMessage());
        }

    }

}
```

Summary information from ODP

Note that depending on length, the summary may be truncated in the results:

```
Top/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines {SE="",
FVN="Top/Arts/Literature/Genres/Science_Fiction/Magazines_and_E-zines"}
<b>Science</b> <b>Fiction</b> Weekly
http://www.scifi.com/sfw/
        <b>...</b> Hollywood will choose to adapt better SF, and more. <b>...</b>
(c)<br> Copyright 2002, <b>Science</b> <b>Fiction</b> Weekly(tm).
Open Directory Information:
        <b>Science</b> <b>Fiction</b> Weekly
        The leading electronic publication covering the world of <b>Science
            </b><b>
                    Fiction</b>, with news, reviews, original...

Top/Arts/Genres/Science_Fiction_and_Fantasy {SE="",
FVN="Top/Arts/Genres/Science_Fiction_and_Fantasy"}
SCIFI.COM
http://www.scifi.com/
    [ Farscape/Stargate SG-1 ]. Apr 29, 2002, Today's<br> News Hodder Talks
    More 'Daredevil' 'Sub-Mariner <b>...</b>
Open Directory Information:
                SCIFI.COM
                News, reviews, games and shopping from the SCI FI Channel.

Top/Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy {SE="",
FVN="Top/Arts/Writers_Resources/Fiction/Science_Fiction_and_Fantasy"}
<b>Science</b> <b>Fiction</b> and Fantasy Writers of America, Inc.
http://www.sfwa.org/
     <b>...</b> not of SFWA. Except where otherwise noted,
content and design copyright A 1995-2002
<br> by <b>Science</b> <b>Fiction</b> and Fantasy Writers of
America, Inc. (&quot;SFWA&quot;). SFWA <b>...</b>
Open Directory Information:
                <b>Science</b> <b>Fiction</b> & Fantasy
Writers of America, Inc.
                The official Website of the premier professional <b>science</b>
<b>fiction<
                /b> writer's association. The SFWA site...
                        ...
```

Choose a range of results

Up to now, the examples have shown searches that simply request the first 10 records

for a search. The API enables requests for any record within the first 1000 results, with a maximum of 10 at a time. By setting the starting position and maximum number of results, an application can work its way through the available results.

```
...
          search.setQueryString(searchTerm);
          int startResult = 100;
          search.setStartResult(startResult);

          int maxResult = 5;
          search.setMaxResults(maxResult);

        GoogleSearchResult result = search.doSearch();

        GoogleSearchResultElement[] resultElements = result.getResultElements();

        int startIndex = result.getStartIndex() - 1  - startResult ;
        int endIndex = result.getEndIndex() - 1  - startResult ;

        for (int i = startIndex; i <= endIndex; i++) {
            GoogleSearchResultElement resultElement = resultElements[i];
...
```

Note that the offset doesn't perform exactly as you think it might. The `startResult`, for example, actually represents the index of the last record NOT included. For example, the code above actually returns not listings 100 through 105, but 101 through 106.

For this reason, the `startIndex` and `endIndex` are adjusted not just by the `startResult`, but by 1 (as before) as well.

Section 5. Restricting results

Setting restrictions

Google searches can be restricted based on a number of factors. These factors include language, content, country, and topic. You can also control the encoding for input and output.

For example, Google lists country codes, allowing you to restrict data based on country of origin. The complete list is shown on the distribution file `APIs_Reference.html`, but to limit the search to pages containing the phrase `"science fiction"` that have the United Kingdom as their source of origin, set the `restrict` parameter:

```
...
        search.setMaxResults(maxResult);

        search.setRestrict("countryUK");

        GoogleSearchResult result = search.doSearch();
...
```

Google also specifies four "special collections"of information:

- U.S. Government (unclesam)
- Linux (linux)
- Macintosh (mac)
- FreeBSD (bsd)

So to search for `"science fiction"` pages within the `linux` collection set the restriction as:

```
...
        search.setQueryString(searchTerm);

        search.setRestrict("linux");

        GoogleSearchResult result = search.doSearch();
...
```

Note that topic areas do not always restrict results the way that you might think. For example, there are actually 8710 results for `"science fiction"` in the `linux` collection.

Results for restricted search

Here is a sample of the 8710 results from the search for `"science fiction"` pages within the `linux` collection:

```
...
Bills Pointers to <b>Science</b> <b>Fiction</b>
http://starship.python.net/crew/gandalf/rtlinux/luz.cs.nmt.edu/%257Etobor/jwz/ba
bylon.shtml
     Bills Pointers to <b>Science</b> <b>Fiction</b>. Here is what
happens<br> w

hen barney meets star trek. Warner <b>...</b>

General and <b>Science</b> <b>Fiction</b> books
http://www.bowling.uklinux.net/bookshop/sfbooks.php3
      <b>...</b> Striking Home <b>Science</b>
<b>Fiction</b> Bookshop In Associa

tion with Amazon.com<br> and Amazon.co.uk [Christian Books] [Bowling Books] <b>.
..</b>

<b>Science</b> <b>Fiction</b>
http://nico-s.homelinux.org/ScienceFiction/livres/categories.php?FInfos=sciencef
iction
     <b>Science</b> <b>Fiction</b>.
<b>...</b> Definition de la &quot;<b>Science

</b> <b>Fiction</b>&quot; par les editions Pocket. Barrington<br> J. Bayley. Al'
unite. Le rayon Zen (auxditions le livre de poche). <b>...</b>
...
```

You can also combine restrictions, as in:

```
unclesam.countryUS
linux.(countryUS|countryUK)
max.(-countryUS)
```

Choosing languages

The API also provides a specific parameter for limiting the language for results.

```
...
        search.setQueryString(searchTerm);

        search.setLanguageRestricts("lang_fr");

        GoogleSearchResult result = search.doSearch();
  ...
```

Google recognizes the following language codes:

```
Arabic:  lang_ar
Chinese (S):  lang_zh-CN
Chinese (T):  lang_zh-TW
Czech:  lang_cs
Danish:  lang_da
Dutch:  lang_nl
English:  lang_en
Estonian:  lang_et
Finnish:  lang_fi
French:  lang_fr
German:  lang_de
```

```
Greek:  lang_el
Hebrew:  lang_iw
Hungarian:  lang_hu
Icelandic:  lang_is
Italian;  lang_it
Japanese:  lang_ja
Korean;  lang_ko
Latvian:  lang_lv
Lithuanian:  lang_lt
Norwegian:  lang_no
Portuguese:  lang_pt
Polish:  lang_pl
Romanian:  lang_ro
Russian:  lang_ru
Spanish:  lang_es
Swedish:  lang_sv
Turkish:  lang_tr
```

Specifying encoding

For situations where information is coming from or going to environments where a different encoding is in use, the API enables you to specifically set the input and output language encoding. For example:

```
...
        search.setQueryString(searchTerm);
         search.setInputEncoding("greek");
         search.setOutputEncoding("hebrew");

        GoogleSearchResult result = search.doSearch();
...
```

In this case, the query string `searchTerm` will be interpreted as though it were encoded using `greek`, and the results will be sent back encoded in `hebrew`. The default encoding is `latin1`.

Eliminating adult results

It's hard to surf for any significant period of time without noticing the large amount of adult material on the Web. To eliminate adult results from the search set, selected a safe search:

```
...
        search.setQueryString(searchTerm);

        search.setSafeSearch(true);

        GoogleSearchResult result = search.doSearch();
...
```

If `setSafeSearch` is set to `true`, Google uses a filter that checks keywords and

phrases, URLs and Open Directory categories.

Eliminating related results

In situations where many results are similar, the API allows you to return only representative pages. For example, if filtering is turned on, Google will return only the first result of a set with substantially similar titles and snippets, and only the first two results from the same Web host. This prevents the user from being swamped by results from a single site, but still provides a look at what's available from that site.

The API allows you to turn filtering off using the `setFilter()` method, but see the note below the code sample.

```
...
        search.setQueryString(searchTerm);

        search.setFilter(true);

        GoogleSearchResult result = search.doSearch();
...
```

Note that according to the documentation, filtering is turned on when this value is set to `true` and off when it's set to `false`. It appears, however, that this situation is reversed, and that the related results are returned when the value is set to `true` and kept out of the result set when it is set to `false`.

Section 6. Spell-checker and cached pages

Retrieving cached results

As part of its indexing process, Google retrieves a copy of the page that it has indexed, and provides users with a link to that cached page within the search results. The primary disadvantage of this is that the user doesn't get an updated look at content, but several advantages do exist.

Retrieving the cached page is often faster than the real page, due to Google's bandwidth and processing power, and is available even if the real server is not, for whatever reason. Most of all, however, search terms are highlighted within the cached page, making it easier to see the relevance of the page.

The Google API makes it possible to retrieve the text of a cached page:

```
import com.google.soap.search.GoogleSearch;
import com.google.soap.search.GoogleSearchFault;

public class GoogleCacheTutorial {

   public static void main (String[] args) {

      try {

         GoogleSearch search = new GoogleSearch();

         search.setKey("00000000000000000000000000000000");

        byte[] pageText = search.doGetCachedPage(
        "http://www-106.ibm.com/developerworks/xml/library/x-tiphdln.html");

        System.out.println(new String(pageText));

      } catch (GoogleSearchFault gsf) {
         System.out.println("Google Search Fault: "+gsf.getMessage());
      }

   }

}
```

Here, rather than using `doSearch()`, use `doGetCachedPage()`. The result is a `byte` array that you can use like any other Java `byte` array.

Spelling suggestions

Understanding that users often don't know how to spell the topic that they're looking for, the Google search engine has a spell-checker built in that analyzes queries and makes suggestions. These spelling suggestions are available independently of any searches.

```
import com.google.soap.search.GoogleSearch;
import com.google.soap.search.GoogleSearchFault;

public class GoogleSpellingTutorial {

    public static void main (String[] args) {

        String spellingRequest = args0];

        try {

            GoogleSearch search = new GoogleSearch();
            search.setKey("00000000000000000000000000000000");

             String suggestion = search.doSpellingSuggestion(spellingRequest);

            if (suggestion == null) {
               System.out.println("There is no suggestion in the database.");
            } else {
               System.out.println(suggestion);
            }

        } catch (GoogleSearchFault gsf) {
           System.out.println("Google Search Fault: "+gsf.getMessage());
        }

    }

}
```

Any word or phrase can be checked. If there are no spelling errors, or if the engine
doesn't recognize the word at all, it returns `null`. Otherwise, a suggested word or
phrase is returned.

Section 7. Using the Web service directly

The Google Web service

So far this tutorial has discussed using the Google Web service through Java applications using the API, but this is just a convenience. The service itself is available directly through SOAP messages at `http://api.google.com/search/beta2`.

A SOAP message carries information on the server side procedure to execute, with parameters specified within the body of the message. In response, the server sends a SOAP message with information encoded in its body.

The application can then take this response and deal with it, either as individual pieces of information, or by transforming the data using traditional XML methods.

This section looks at the request and response SOAP messages and a JAXM application to send and receive them. (For more information on sending and receiving SOAP messages using JAXM, see the Building Java applications with the Google API resources on page 32 .)

Sending a SOAP request

First, take a look at a JAXM application that sends a pre-determined SOAP message and outputs the results.

```
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPElement;
import java.io.FileInputStream;
import javax.xml.transform.stream.StreamSource;
import javax.xml.messaging.URLEndpoint;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.Source;

import javax.xml.transform.stream.StreamResult;

public class GoogleSOAP {

    public static void main(String args[]) {

        try {

          //First create the connection
           SOAPConnectionFactory soapConnFactory =
                         SOAPConnectionFactory.newInstance();
```

```
        SOAPConnection connection =
                        soapConnFactory.createConnection();

        //Next, create the actual message
        MessageFactory messageFactory = MessageFactory.newInstance();
        SOAPMessage message = messageFactory.createMessage();

        //Create objects for the message parts

        SOAPPart soapPart =      message.getSOAPPart();

        //Populate the Message
        StreamSource preppedMsgSrc = new StreamSource(
                new FileInputStream("search.msg"));
        soapPart.setContent(preppedMsgSrc);

        //Save the message
        message.saveChanges();

        //SEND THE MESSAGE AND GET A REPLY

        //Set the destination
        URLEndpoint destination =
            new URLEndpoint("http://api.google.com/search/beta2");
        //Send the message
        SOAPMessage reply = connection.call(message, destination);

        //SAVE THE OUTPUT

        //Create the transformer
        TransformerFactory transformerFactory =
                        TransformerFactory.newInstance();
        Transformer transformer =
                    transformerFactory.newTransformer();
        //Extract the content of the reply
        Source sourceContent = reply.getSOAPPart().getContent();

        //Set the output for the transformation
        StreamResult result=new StreamResult("results.out");
        transformer.transform(sourceContent, result);

        //Close the connection
        connection.close();

    } catch(Exception e) {
        System.out.println(e.getMessage());
    }
  }
}
```

This application first creates a connection, then a SOAPMessage object. The file
search.msg populates the SOAPPart of the message, which is then sent to the
Google Web service.

The reply comes back in the form of another SOAP message, with the relevant
information in the SOAPPart. In this case, you're not actually transforming the
information, but rather using the Transformer as a serializer to send the result to the
results.out file.

In the next panels, I'll examine the actual messages.

The search request

The search request SOAP message contains all of the same parameters that the API set previously, but this time they are XML elements:

```
<?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: doGoogleSearch  xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <q xsi:type="xsd:string">"science fiction"</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">false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1: doGoogleSearch >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Notice that the method name, `doGoogleSearch`,is specified in the message, and that each parameter is listed, even if it has no value. This is the same as executing a method of:

```
doGoogleSearch("", "\"science fiction\"", 0, 10, true, "", false, "", "latin1",
"latin1")
```

Search results

The result is an XML file that carries the same information the API extracted. The partial result shown here has had spacing added to make it easier to read.

```
<?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">
      <documentFiltering xsi:type="xsd:boolean">false</documentFiltering>
      <estimatedTotalResultsCount xsi:type="xsd:int">
      1500000</estimatedTotalResultsCount>
      <directoryCategories
```

```
                xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
                xsi:type="ns2:Array"
                ns2:arrayType="ns1:DirectoryCategory[0]">
        </directoryCategories>
        <searchTime xsi:type="xsd:double">0.060022</searchTime>
        <resultElements
                xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
                xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[10]">
           <item xsi:type="ns1:ResultElement">
              <cachedSize xsi:type="xsd:string">29k</cachedSize>
              <hostName xsi:type="xsd:string"/>
              <snippet xsi:type="xsd:string">
                  &lt;b&gt;...&lt;/b&gt; After a long career creating superior
                  &lt;b&gt;science&lt;/b&gt;-&lt;b&gt;fiction&lt;/b&gt; TV,
                  writer/producer David Kemper&lt;br&gt; reflects on the finale of
                  Farscape's current season and his startling &lt;b&gt;...&lt;/b&gt;
              </snippet>
              <directoryCategory xsi:type="ns1:DirectoryCategory">
                 <specialEncoding xsi:type="xsd:string"/>
                 <fullViewableName
                     xsi:type="xsd:string">Top/Arts/Literature/Genres/Science_Fiction
                     /Magazines_and_E-zines</fullViewableName>
              </directoryCategory>
              <relatedInformationPresent xsi:type="xsd:boolean">
                  true
              </relatedInformationPresent>
              <directoryTitle xsi:type="xsd:string">
                  &lt;b&gt;Science&lt;/b&gt; &lt;b&gt;Fiction&lt;/b&gt; Weekly
              </directoryTitle>
              <summary xsi:type="xsd:string">
                  The leading electronic publication covering the world of
                  &lt;b&gt;Science&lt;/b&gt; &lt;b&gt;Fiction&lt;/b&gt;, with news,
                  reviews, original...
              </summary>
              <URL xsi:type="xsd:string">http://www.scifi.com/sfw/</URL>
              <title xsi:type="xsd:string">
                  &lt;b&gt;Science&lt;/b&gt; &lt;b&gt;Fiction&lt;/b&gt; Weekly
              </title>
           </item>
...
        </resultElements>
        <endIndex xsi:type="xsd:int">10</endIndex>
        <searchTips xsi:type="xsd:string"/>
        <searchComments xsi:type="xsd:string"/>
        <startIndex xsi:type="xsd:int">1</startIndex>
        <estimateIsExact xsi:type="xsd:boolean">false</estimateIsExact>
        <searchQuery xsi:type="xsd:string">
            &amp;quot;science fiction&amp;quot;
        </searchQuery>
     </return>
   </ns1:doGoogleSearchResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Notice that each information item retrieved by the API is available in an XML element.

Transforming the result

Making use of the results can be as simple as creating a style sheet to transform them into the desired format. For example, a simple style sheet can extract site titles and URLs:

```xml
<?xml version="1.0"?>

<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text"/>

<xsl:template match="/">
<xsl:apply-templates select="//item"/>
</xsl:template>

<xsl:template match="item">
  <xsl:value-of select="title"disable-output-escaping="yes"/><xsl:text>
  </xsl:text><xsl:value-of select="URL"/><xsl:text>

</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

The application can then call this style sheet...

```java
...
        //Create the transformer
        TransformerFactory transformerFactory =
                      TransformerFactory.newInstance();
          Source styleSheet = new StreamSource("translate.xsl");
        Transformer transformer =
                      transformerFactory.newTransformer( styleSheet );
        //Extract the content of the reply
        Source sourceContent = reply.getSOAPPart().getContent();

        //Set the output for the transformation
        StreamResult result=new StreamResult("results.out");
        transformer.transform(sourceContent, result);
...
```

... leading to nicely formatted results:

```
<b>Science</b> <b>Fiction</b> Weekly
   http://www.scifi.com/sfw/

SCIFI.COM
   http://www.scifi.com/

<b>Science</b> <b>Fiction</b> and Fantasy Writers of America, Inc.
   http://www.sfwa.org/

The Link ping <b>Science</b> <b>Fiction</b> &amp; Fantasy Archive
   http://www2.lysator.liu.se/sf_archive/
...
```

Note that this is just a simple example. You can use XSLT to extract and even process data for inclusion in databases and other systems, as well as for use by other applications.

Spelling requests

The SOAP messages for spelling suggestions are fairly straightforward.

The request:

```
<?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:doSpellingSuggestion
        xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <phrase xsi:type="xsd:string">scince</phrase>
    </ns1:doSpellingSuggestion>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The result:

```
<?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:doSpellingSuggestionResponse xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:string">science</return>
</ns1:doSpellingSuggestionResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Cache requests

Cache requests are a bit more complex because responses are returned in `base64`.
The request:

```
<?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:doGetCachedPage xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">181gQtaOdlLD+83qvh6PnssQR5WlaB9d</key>
      <url xsi:type="xsd:string">
```

```
                   http://www-106.ibm.com/developerworks/xml/library/x-tiphdln.html
        </url>
    </ns1:doGetCachedPage>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

generates the result:

```
<?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:doGetCachedPageResponse xmlns:ns1="urn:GoogleSearch"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:base64">PG1ldGEgaHR0cC1lcXVpdj0iQ29udGVudC1UeXBlIiBjb250ZW50
PSJ0ZXh0L2h0bWw7IGNoYXJzZXQ9SVNPLTg4NTktMSI+CjxCQVNFIEhSUY9Imh0dHA6Ly93d3ctMTA
2LmlibS5jb20vZGV2ZWxvcGVyd29ya3MveG1sL2xpYnJhcnkveC10aXBoZGxuLmh0bWwiPjx0YWJsZSBib
3JkZXI9MSB3aWR0aD0xMDAlPjx0cj48dGQ+PHRhYmxlIGJvcmRlcj0xIGJnY29sb3I9I2ZmZmZmZiBjZW
xscGFkZGluZz0xMCBjZWxsc3BhY2luZz0wIHdpZHpH...
ZT4NCjwhLS0gRU5EIElCTSBGT09URVIgLS0+PC9ib2R5Pgo8L2h0bWw+Cg==</return>
</ns1:doGetCachedPageResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Be sure to adjust your application accordingly, or simply use the API to do the
translation automatically.

Section 8. Summary

Building Java applications with the Google API summary

The Google search engine has made its index, spelling suggestions, and cached pages available via a Web services interface, allowing programmers of all languages to access their information for personal use. They have also provided a Java API, which makes accessing the data more straightforward.

Using the API, developers can execute searches using the same complex queries that can be used on the Google Web site itself, receiving in response an array of results that carry information as to the page, title, URL, category, and other information. This information can then be used by other applications.

Building Java applications with the Google API resources

developerWorks tutorials and tips:
- *TIP: Send and receive SOAP messages with JAXM*.
- *Web services -- the Web's next revolution*
- *Creating a complete Web service*
- *Transforming XML documents*
- *Manipulating data with XSL*
- *Using JDBC to insert data from XML into a database*
- *Introduction to XML messaging*
- *XML messaging with SOAP*

Google information
- Get information about the Google search engine at *http://www.google.com/about.html*.
- Get more information about the Google APIs at *http://www.google.com/apis/*.
- Request a Google license key at *https://www.google.com/accounts/NewAccount?continue= http://api.google.com/createkey&followup=http://api.google.com/createkey*.
- Read the Google API Terms of Service at *https://www.google.com/accounts/TermsOfService*.

Download software
- Download the Java ™ 2 SDK, Standard Edition version 1.3.1or higher from *http://java.sun.com/j2se/*.

- The Google API, is available at *http://www.google.com/apis/*.
- The Java API for XML Mess aging, or JAXM is available as part of the Java XML Pack, at *http://java.sun.com/xml/downloads/javaxmlpack.html*.

Offline classes
- IBM offers multi-day classroom courses to help developers get up to speed on various topics. For a list of XML-related coursework, visit *http://www-3.ibm.com/services/learning/spotlight/ebusiness/all.html#XML* .
- In partnership with iSeries, IBM also offers internet-based classes covering various topics. For information on Java technology and XML-related coursework, see *http://www.iseries.ibm.com/developer/education/ibo/view.html?jva*.

Feedback

Please send us your feedback on this tutorial. We look forward to hearing from you!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial Building tutorials with the Toot-O-Matic demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11. We'd love to know what you think about the tool.