Paper CC-03-2012

Search Engine Using SAS

Pramod.R,
Target Corporation, Minneapolis, Minnesota

ABSTRACT

It is a world of search engines. We have Google, Yahoo and Bing as the major giants in the search engines. This paper attempts to have a SAS Enterprise Guide application which mimics the functionality of a search engine. The SAS procedure – proc http enables you to connect to the internet and access the web pages, APIs that are available on the internet servers. Google offers a free API for accessing its Custom Search engine which is accessed using the proc http.

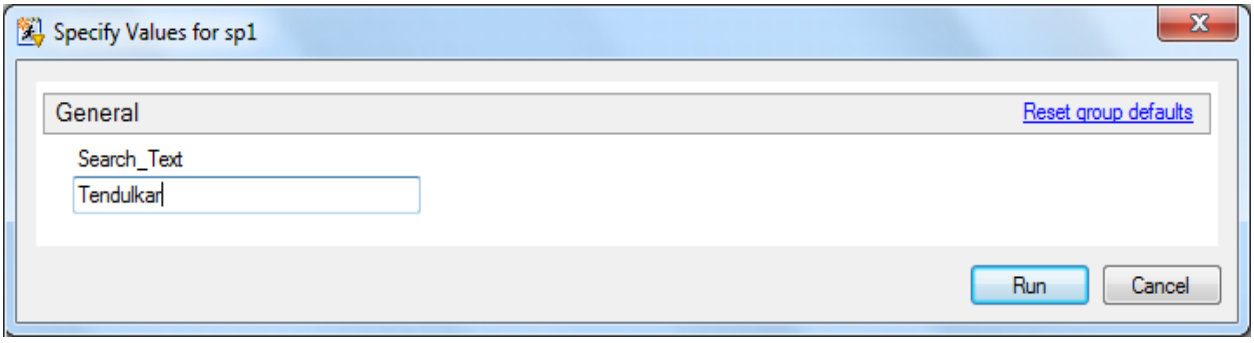
I first create a stored process in SAS Enterprise Guide 5.1 which has a text box to take in the search key words from the user. This is then captured in the prompt macro variable and passed on to the Google Custom Search API as a parameter along with the API key and other variables using the PROC HTTP or cURL/wget tool in UNIX. The Google webservice throws out the result after processing the API which is then displayed in the SAS result in the Enterprise Guide output window.

INTRODUCTION

I cannot imagine a day without Google or Bing. People are so much used to the concept of having a search engine for everything. And it is convenient that we have a variety of browsers to support the search functionality. This prompted me to develop search engine functionality by having SAS Enterprise Guide as a makeshift browser.

SAS Enterprise Guide Stored Process introduces a new dimension in the SAS – Interact-ability. The ability to have the user dynamically input the data adds in a lot more value than a static method of performing an ETL, reporting or some analysis and data manipulation.

The front end that I designed was using the Stored Process which had a simple UI, containing a text box as shown below. This has a simple text box which takes in the search key words. The word that is keyed in is passed to the back end code via prompt macro variable. This can also be opened using the Stored Process for Web application.



THE GOOGLE CUSTOM SEARCH API

The Google Custom Search enables you to search the Google web using their API, retrieve the results of the custom search and embed them in the desired format. With this API, you can use RESTful requests to get either web search or image search results in JSON or Atom format.

In order to use this API, we would first need to create an API key. This API key would be one of the parameters that we would be passing while accessing the Google Custom Search API, alongside with the other parameters like the query string and the output format that is desired.

Once the API key is created, we would be invoking the Google search API using the REST (Representational State Transfer) directly using the HTTP GET Method, with the parameters as mentioned in the API documentation.

The basic most set of parameters would include the API key, which would actually be sent inside the URL, the query key word that was keyed in (the word for which the search is triggered) and optionally a custom search name that we have created while creating the Custom Search for our profile. However, the API documentation explains the detailed list of other parameters that can be used along side.

The URL is constructed as mentioned above, using the SAS data step which takes in the prompt macro variable from the Stored Process variable that is keyed in by the user. This is then passed on to the Google API using proc http as shown below:

```
FILENAME in "~/playground/google_in";

FILENAME out "~/playground/google_out";

DATA _NULL_;
var='key=xxxxxxxxxxx&cx=xxxxxxxx&q='||"&prompt"||'alt=atom';
 FILE in;
 PUT var $;
RUN;

PROC HTTP IN=IN OUT=OUT URL="https://www.googleapis.com/customsearch/v1?"
METHOD="get" CT="application/x-www-form-urlencoded";
RUN;
```

The output can be retrieved in two formats – JSON and ATOM. This paper would use ATOM as an output which is essentially an XML-based language used for web feeds.

This is then parsed to fetch the results and their corresponding URLs. These URLs are then put into the output html, displayed as a hyperlink. The html is then shown as an output to the Stored Process.

USING WGET/CURL TO CALL THE API

Alternatively, one can even use the wget or the curl to invoke the Google Custom Search API using the GET method. After constructing the URL along with the parameters, we can use the curl through call system command or the x command in SAS to invoke the API as shown below:

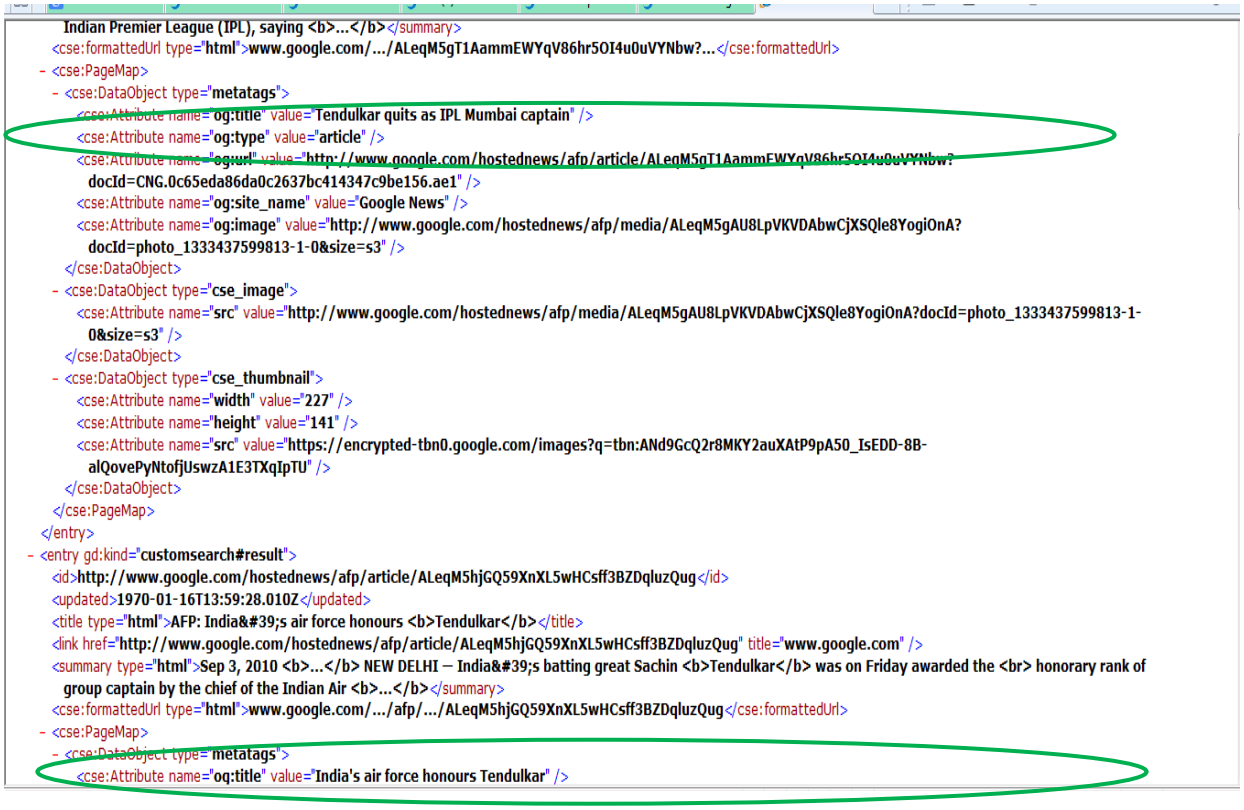
```
DATA _NULL_;
X "wget -o ~/google.log -O ~/google.xml
https://www.googleapis.com/customsearch/v1?key=xxx&q=Tendulkar&alt=atom";
RUN;
```

The result ATOM can be stored in a file in the directory which can later be read into the SAS using the filename xml tool.

Both the above mentioned methods of invoking the webservice would require us to have the API key being passed along with the url query string.

PARSING THE ATOM FILE

The XML file (ATOM) is now parsed and read into the SAS datasteps using simple infile input statements. The sample ATOM file looks like this (for the keyword – Google)



As we can see, there are tags having the urls and the text labels. We can now parse only these fields and construct the html. Below I have used index function to find a specific tag (like title and url in this case) and fetch them into SAS datasets. Alternatively, we can even use prxparse or SAS xmlmapper to parse the xml. Index was used for the simplicity sake.

```
FILENAME XML1 '~/playground/wget_google.html';

DATA T;
INFILE XML1 LRECL=10000 TRUNCOVER;
INPUT TAGS $16000.;
IF INDEX(TAGS,'name="og:title"')>0 THEN OUTPUT;
IF INDEX(TAGS,'name="og:url"')>0 THEN OUTPUT;
RUN;
```

Once we have parsed the required tags containing the description and the url, we construct the html that requires to be displayed in the output and write that out to the _webout. In the below code, I separate out the url tags from the title tags into two separate datasets and then merge them in order to have them appearing side by side. All these transformations are done in order to better align the data that can fit into the html.

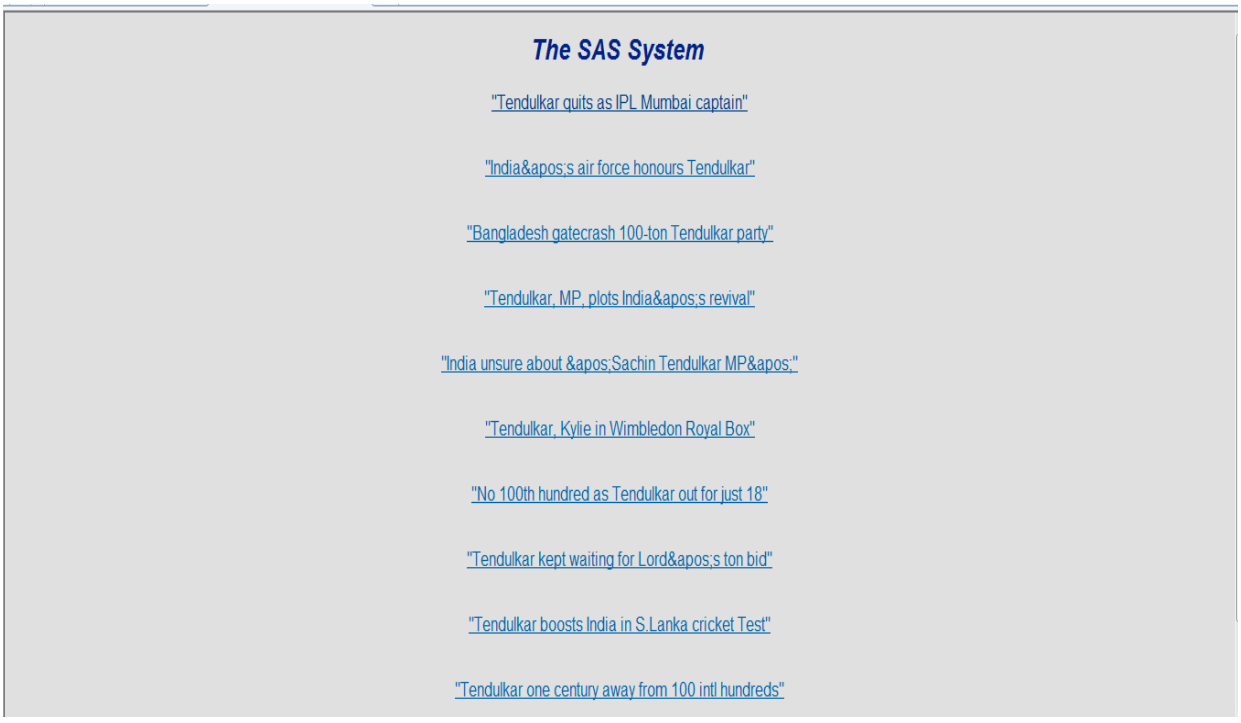
```
DATA TITLE(DROP=TAGS) LINK(DROP=TAGS);
SET T;
N= INDEX(TAGS,'value=');
TXT=SUBSTR(TAGS,N+6,LENGTH(TAGS)-N -7);
IF MOD(_N_,2)=1 THEN OUTPUT TITLE;
ELSE OUTPUT LINK;
RUN;

DATA FINL(DROP=TITLE LINK);
MERGE TITLE(RENAME=(TXT=TITLE)) LINK(RENAME=(TXT=LINK) DROP=TAGS);
TAG='<a href='||STRIP(LINK)||'>'||STRIP(TITLE)||'</a> <br /><br /><br />';
RUN;
```

The next step is to write the output into the web browser using the _webout. Below code creates a simple output which has the hyperlinks and the text. However, one can construct even more jazzy html for including the images, etc.

```
DATA _NULL_;
FILE _WEBOUT;
IF _N_=1 THEN PUT '<HTML> <BODY>';
SET WORK.FINL END=LAST;
PUT TAG;
IF LAST THEN PUT '</BODY> </HTML>';
```

The output is as shown below:



CONCLUSION

With the advent of SAS Stored Process and SAS Web Reporting suites, added with the web interacting tools like PROC HTTP and PROC SOAP, SAS can be used in more than one ways to achieve various solutions. This paper has attempted to illustrate one such hypothetical example of emulating a search engine.

REFERENCES

Google API documentation: https://developers.google.com/custom-search/v1/getting_started

Aanderud, Tricia, and Angela Hall. 2012. *Building Business Intelligence Using SAS®: Content Development Examples.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The Author wishes to thank Jeelani Basha, Shalini MR and Jeyvinth Rayan for willing to proofread my papers and providing valuable comments and suggestions. I would like to thank Jared Moore for all his help, right from getting information regarding the conference till helping me print this paper. Lastly, I would like to thank my wife Bhargavi, who has been a constant motivator and support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Pramod R
Enterprise: Target Corporation Pvt. Ltd
Address: Apt No. 2405, 1117, Marquette Avenue
City, State ZIP: Minneapolis, Minnesota, 55403
Work Phone: 612-272-1958
E-mail: getpramod.r@gmail.com
Web: www.pramod-r.blogspot.com