Wildfire Content Generator 1.0

Mike Karlsven
ISYS 540
April 13, 2011

Executive Summary

My company has a need to easily integrate existing content on the web into new websites. This could be product data, directory listings, or even blog postings. The data is used to generate content for the purpose of search engine optimization (SEO) and will be crawled by the search engines to improve page rank for our customers. The data being gathered is either public knowledge or product data which our customers either own or have access to legally to sell the products.

The Excel tool has two parts: the data collector and the database connector. The data collector is operated automatically (statically set for a single site, but able to be altered for additional sites by just changing a few variables) by using a button on the user form (see Appendix Exhibit 1). The program is currently configured to hit a website and gather dentist information for each city in every state in the US and compile the information in the workbook organized by state. When the collection is complete the user will receive an email notifying them that it is complete (this is optional if the user chooses to enter their email address). The database connector portion of the program allows the user to select a state (sheet) of the workbook to be uploaded to the database. The database information is entered by the user including username and password, database server, and database name. The program connects to the database and loops through each entry on the sheet chosen to upload and inserts the information into the database. The database is set up to have categories which are city names. Whenever a new city is encountered in the data, a new category is created in the database and the entries that belong to that particular city/category are labeled as such when added to the database.

Implementation Documentation

User Form (UserForm1)

The user form (Appendix Exhibit 1) was created using Excel's design mode. The form is initiated automatically when the workbook is opened by adding code to the *ThisWorkbook* object. It can also be opened via a button on Sheet1 of the workbook. The user form is the primary way for the user to interact with the program. It allows the user to enter the following information:

- Database settings (Database Uploader)
    - Database Server
    - Database Name
    - Username
    - Password
    - Select sheet to upload
- Options Settings (Collector)
    - Email address (optional)
    - # of Cities to gather
    - Hide Browser Option

The form allows the user to customize their experience with the program. The *Hide Browser* checkbox allows the user to decide whether or not to display the browser during data collection, allowing them to either monitor or not monitor the progress as it is running. The *# of Cities* option allows the user to decide how many cities per state should be looked at. This was added primarily for troubleshooting purposes and is a quick way to see the program in action by limiting the number of cities to look at per state. When the program was run allowing all cities to be gathered, it took well over 10 hours to complete the job, but did so flawlessly. This leads to the next functionality of the data gathering which is the email address. Because of the potential delay in completion of the task of gathering over 60,000 records, the email address is used to notify the user when the job is complete.

The database settings are fairly straight forward. In order to get the program to connect to the database the MySQL ODBC 5.1.8 driver must be installed and configured correctly on the user's machine. The program can connect to any MySQL database it can reach with the proper credentials, but the tables and fields are hard coded into the program so it is fairly limited in functionality to work with a specific database table structure that was defined for the project. For future revisions of the program a more flexible database structure will need to be implemented (perhaps by utilizing the header row of the worksheets to identify table/field names).

The form also contains buttons to Start, Stop, and Resume data collection as well as reset the data existing in the workbook. The Start button begins the data collection process by calling the data_collection sub procedure. When the data_collection sub procedure is running the start button becomes the Stop button. The stop button will interrupt the data collection process at any point. If data already exists (the stop button was pressed) in the workbook, then the Resume button is displayed and if pressed the data collection will pick up where it left off with the last city worked for the last state worked.

If required data is missing in the form then the user is notified via messages displayed on the form as well as highlighting the fields that are in violation of the requirements (See Appendix Exhibit 2). The user cannot continue until the violations are resolved.

Data Collector (Module1)

The data collector is implemented using the agent class provided by the course instructor to manipulate the Internet Explorer browser. The agent is used to start an Internet Explorer session and navigate to a given website address. Once there, the agent searches for all of the states on the page and stores the links to each state in an array. This array is then traversed one state at a time and the agent navigates to the state URL stored in the array. The state pages contain a list of cities that are in that state. Another array is used to store each city name and URL. The program loops through each city and the agent navigates to the city. Once on the city page, a list of records for that city is displayed. The agent gathers each record and copies the information onto the worksheet one row per record. Once all of the records have been recorded on the worksheet, then the agent navigates to the next city. Once all of the cities have been hit for a state, the loop then starts on the next state and the process starts over collecting all of the cities and each record in each city. The iterations are controlled via nested While and for loops.
In order to accommodate the resume functionality, the current city being worked on is stored in a cell on the state's worksheet.

Database Uploader (Module2)

The database uploader portion is very custom coded for a particular set of tables in my company database. The user selects which worksheet to upload to the database via a combo box on the user form. The user also inputs the database connection settings including username and password. The first thing that happens is the connection to the database using the information provided by the user via the user form. Upon successful connection to the database, the records are inserted into the database tables. The way the database is configured, each city is considered a category. Therefore, when each record is checked, if the city is a new one in the list, then a new category must be created and the category id must be retrieved from the database to properly categorize the records. This is done by comparing the city name to the previous record's city name. If it is different, then a new category is created using the current record's city name. The ID is retrieved using a MySQL command [last_insert_id()]. The records are uploaded one cell at a time into the proper locations in the database. In total, there are 5 queries to the database per record in the spreadsheet. To upload the 1160 records for that state of Utah, it took roughly 5-7 minutes.

Difficulties & Learning Experiences

There were many moments during this project that caused me to pause in my coding and really dig for answers. Most of these experiences came while building the database uploader module. The first roadblock was in the database connection string. I found a tutorial online that helped me with the connection string, but I was still having an issue

getting connected correctly. It turned out that I had my script correct on the Excel end, but my server was not allowing connections from my IP address.

Another one of the great issues that I dealt with during data collection was that of getting all of the data to be gathered up correctly. I ran into an issue while gathering the info from multiple columns of data. I was able to gather info from the left column, but not the right column. I found that the text I was searching on was different for the right hand column. I had to overcome this by adding a second set of data parsers and looped this by row of data. This corrected my column issue, but it also introduced a bug that I was only retrieving the info for every other row of records. I found that my DO loop was incrementing the counter as well as I was manually incrementing the counter.

The third issue that I encountered was during testing my code, I found that I wanted to stop the program early because I had seen what I needed to see, but in doing so, I was never able to see the program progress past the first few cities in the state. I didn't want to wait around forever to see it finish. To overcome this, I added the *# of Cities* field and the Start/Stop/Resume button features. This allowed me to do much better testing as well as added functionality to the user experience to not have to wait for a complete record set in one sitting. The resume feature added a lot of flexibility to the user in case of a data interruption, the user no longer had to start over from scratch on the data gathering.

Appendix

Exhibit 1: User Form

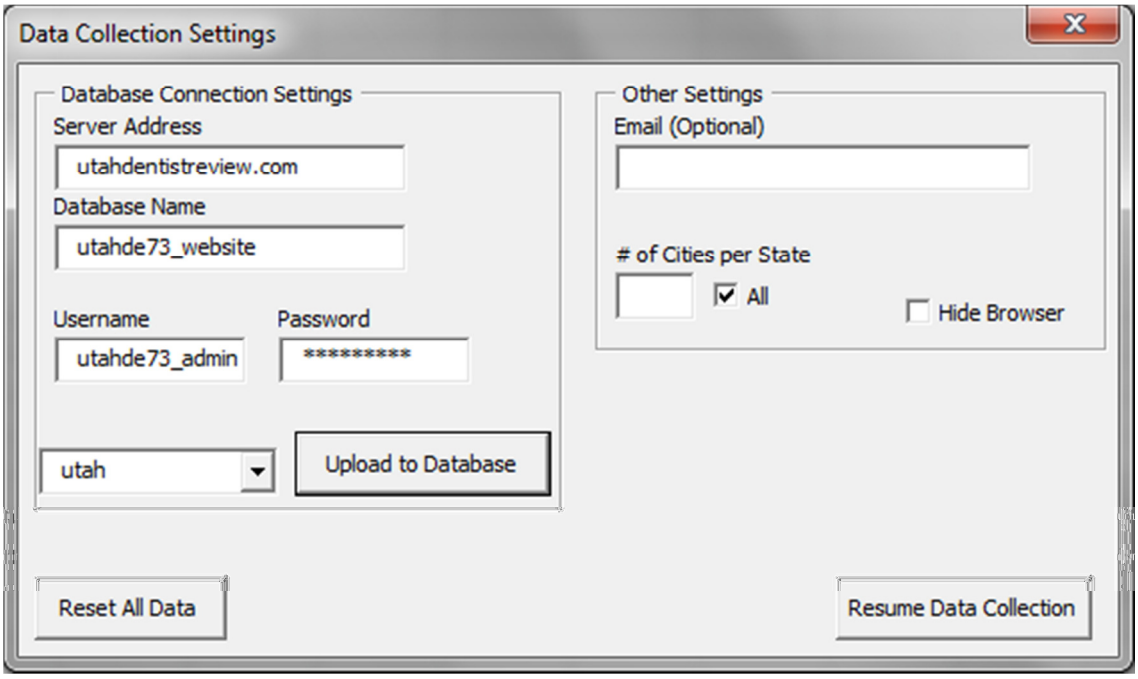

Exhibit 2: User Form Errors