Tyler Oppie

4/13/2011

VBA Project Submission: ProQuest Dissertation Search and Save

2.1 Executive Summary

An ISYS professor is gathering information on every ISYS related dissertation done by individuals, and what advisors advised these doctoral dissertations. My project's purpose was to facilitate the searching and storing of dissertation information through VBA. I started with a dataset from an Access database of Universities. I created an Excel form to view and search information from each school, then through a button click, Excel will query the online Proquest Database for any dissertations from a particular school that were advised by a given last name. Excel will navigate Internet Explorer to a proquest results page. From there the user can open a dissertation's abstract and then with a click on a button in Excel, save the dissertation title and link to a spreadsheet.

This project will save hours of time that would have been spent clicking through web pages, and copy/pasting links. The proquest website is a particularly slow website to use because it resets all your search criteria every time you conduct a new search or use the browser's back button. This is all avoided through the browser automation accomplished through VBA. The Excel form also facilitates queries on the existing university information. This project can easily be expanded to include database tables of authors, advisors, or subjects into the web queries.

2.2 Implementation documentation

The first thing I accomplished through VBA was importing a database table to an Excel sheet. This was done with a simple macro recording. After the data is in Excel, I created two Excel forms. The first form (figure 2) takes a username and password and an option for on campus or off campus. The form is accessible through a custom tab and button on the ribbon shown in figure 1. When submitted, an IE agent instance is opened and saved as a module level variable. The agent navigates to the proquest site where if off campus, the username/password will login to route y, otherwise the agent will go straight to proquest.

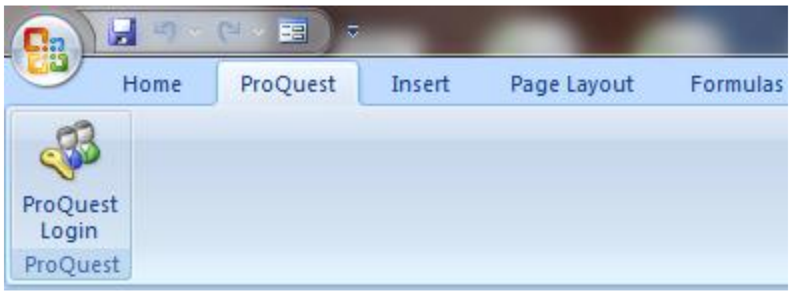


Fig 1, Custom Proquest tab and ProQuest login button

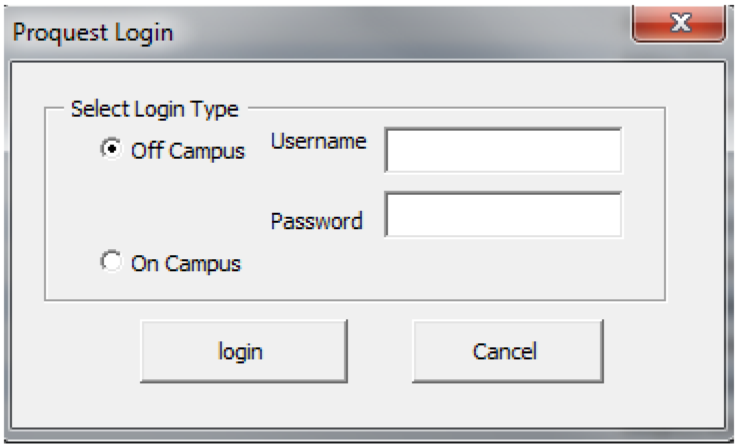
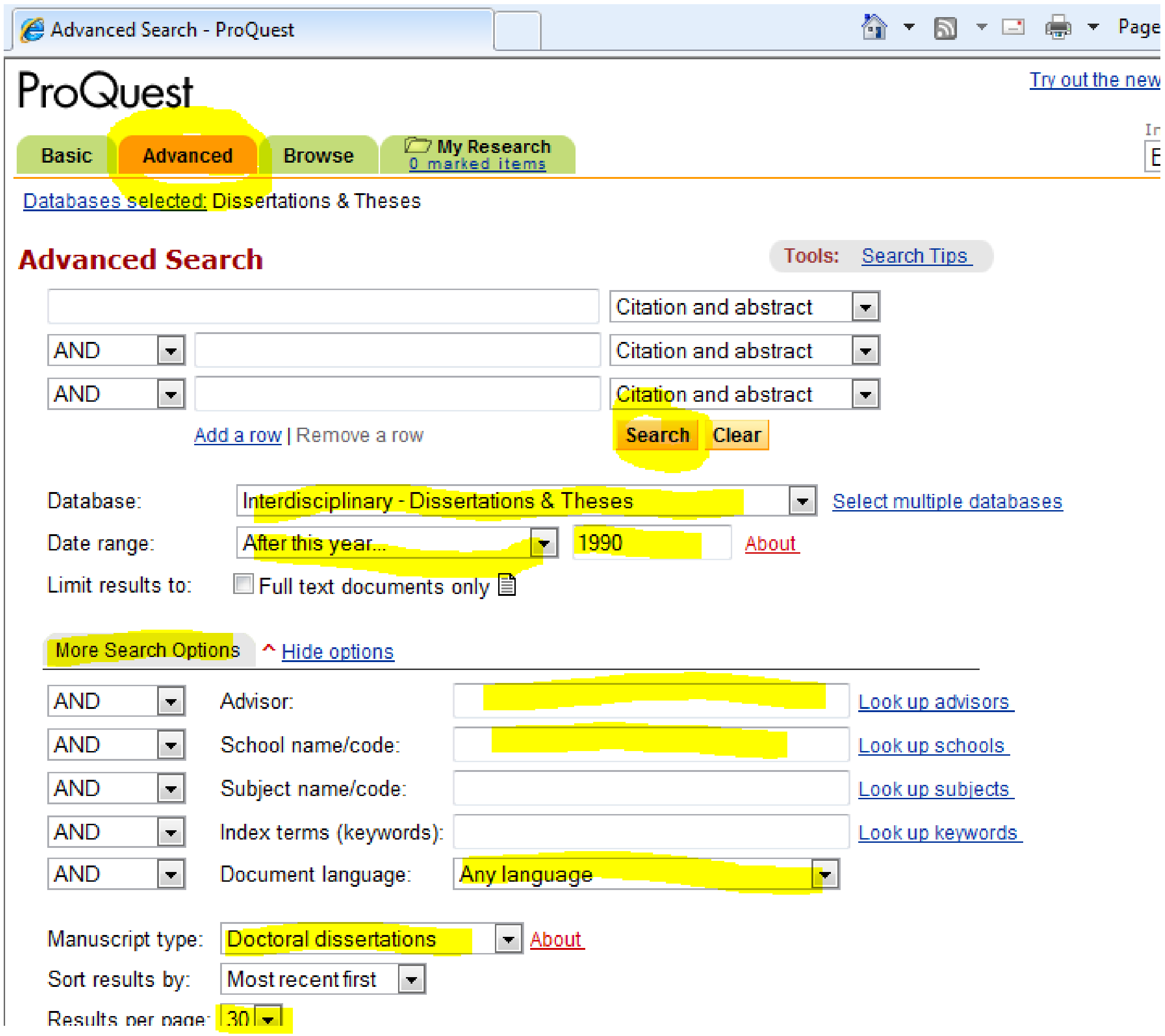


Fig 2, Proquest login form

Once at proquest, the agent clicks buttons and makes combo box selections to reach the page with the particular type of search to be done. The agent then saves this url as a module variable and will refer to it every time a search is initiated by the user. This is essential since using the browser back button to redo a search will wipe out all the button clicking and combo box selection the agent does at first. Figure 3 shows a proquest page and highlights all the fields on the page that the agent is manipulating to do the required type of search.



Databases selected: Dissertations & Theses

Advanced Search

		Citation and abstract
AND		Citation and abstract
AND		Citation and abstract

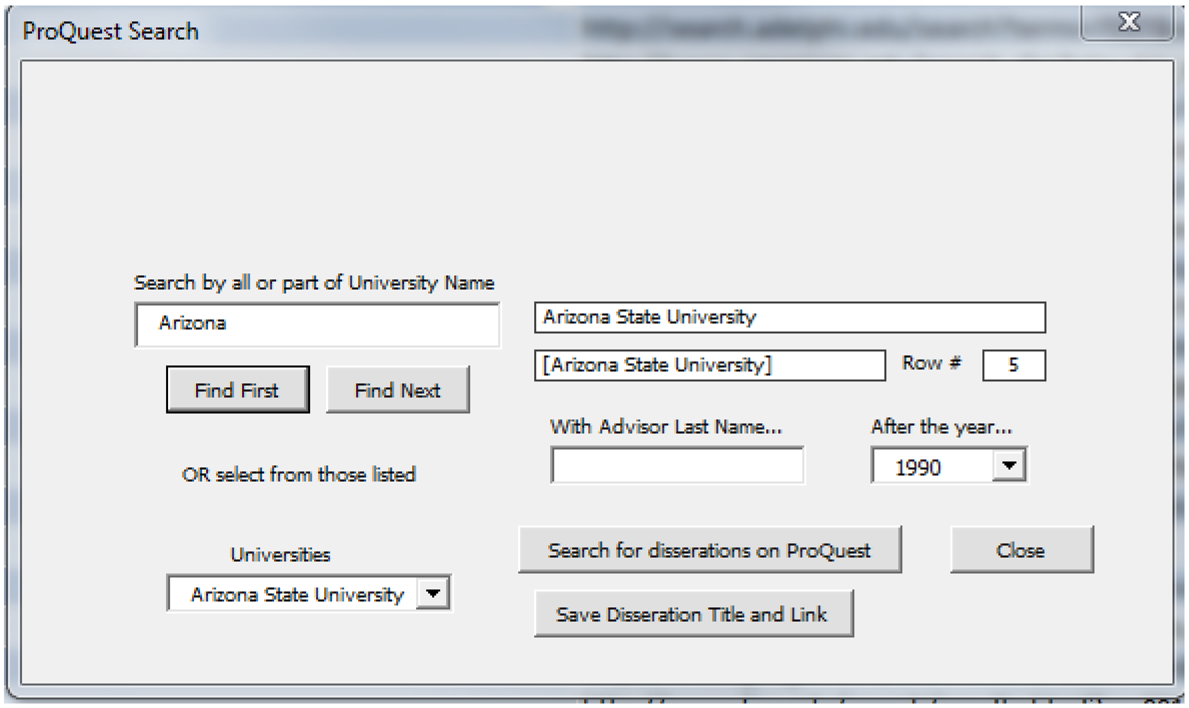
Database:	Interdisciplinary - Dissertations & Theses
Date range:	After this year... 1990
Limit results to:	Full text documents only

AND	Advisor:	
AND	School name/code:	
AND	Subject name/code:	
AND	Index terms (keywords):	
AND	Document language:	Any language

Manuscript type:	Doctoral dissertations
Sort results by:	Most recent first
Results per page:	30

Figure 3, Proquest search page highlighting all manipulated elements

Once the search URL is saved(with the session ID etc), the login form closes and the search form opens. The search form is based on the USDA form assignment from class with some additional functionality. Similar to the USDA form a user can search for a university by name and click first or next search. I added a combo box to allow the user to select a university from a list instead of searching by name. Figure 4 shows the search form.


ProQuest Search
Search by all or part of University Name
Find First
Find Next
OR select from those listed
Universities
With Advisor Last Name...
After the year...
Search for diserations on ProQuest
Close
Save Diseration Title and Link

Arizona

Arizona State University

[Arizona State University]

Row # 5

1990

Arizona State University

Figure 4, the Search form

After finding the University record they want to search for, the user inputs a last name of an advisor in a text box, selects a starting year to search from, then clicks "Search for dissertations on ProQuest". This button will initiate a method that will pass the universities' proquest search name, the last name of an advisor, and other appropriate parameters and feed them to the proquest site through the agent. The agent will then submit the search and the results will be displayed through Internet Explorer.

Once the results page is up, all the user has to do is read from the search results, which dissertations they want to capture information about. In the future, the results can be more specific on subject/ author etc, but for now, the main functionality that the customer wanted was the ability to see all dissertations at a school that were advised by a particular person.

When the user sees a dissertation he or she wants to save, she will click on the abstract link, go back to the excel form, and click “save dissertation title and link”. Save dissertation will parse through the open page on the Internet Explorer window and through the agent, save the title, author/date/etc, and a hyperlink directly to the dissertation on proquest. This information is inputted into the “dissertations” sheet of the excel file on the next available row.

With the excel forms and browser automation, the task of discovering which dissertations thousands of professors have advised can be accomplished in a fraction of the time. The system will also save time with the automatic scraping of titles and links to excel.

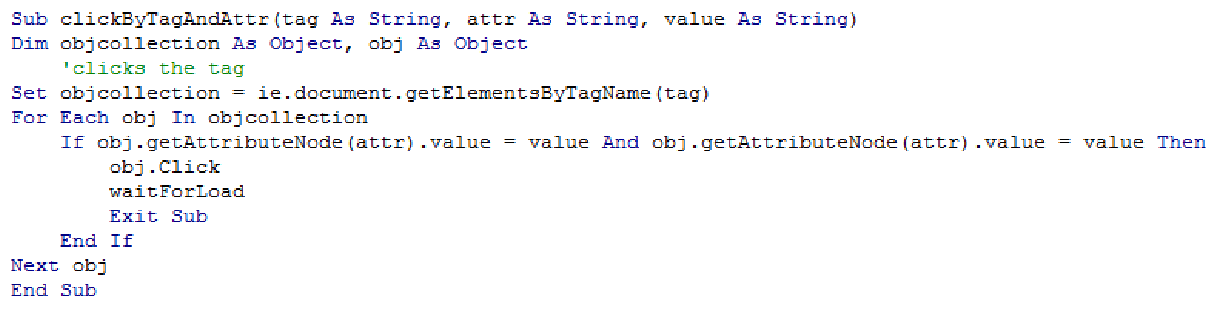
2.3 Learning and conceptual difficulties encountered

Most of the difficulties I encountered were with using the IE agent to automate the search on Proquest. The agent was initially setup to create a new agent every time it was run. What I needed was an agent that would stay as a live variable after a sub procedure is run. That way the user can navigate in IE and then go back to excel, click a button, and have excel use a handle on the IE instance to perform more automation.

Needing excel to keep a handle on the IE object made debugging very difficult because every time there is an error, Excel loses its module level variables and the process must be started again from the beginning. Through breakpoints and tedium I was able to preserve the agent object and continue its use between sub procedure calls.

Another issue I encountered was changing select boxes with javascript events and clicking tags without IDs. Some select boxes were loaded with a javascript “on change” event that changing their value failed to trigger. My work around ended up being manipulating the url parameters to have the same effect as the javascript event.

Some of the submit buttons had no IDs and the only way I found to identify them was though a tag attribute and value pair that was unique. So I added a sub routine to the agent class that searches and click a tag based on its type, a given attribute name and a given attribute value. Research into the IE object helped me to achieve this as shown in figure 5.



```
Sub clickByTagAndAttr(tag As String, attr As String, value As String)
Dim objcollection As Object, obj As Object
    'clicks the tag
Set objcollection = ie.document.getElementsByTagName(tag)
For Each obj In objcollection
    If obj.getAttributeNode(attr).value = value And obj.getAttributeNode(attr).value = value Then
        obj.Click
        waitForLoad
        Exit Sub
    End If
Next obj
End Sub
```

Fig 5, VBA code to click a tag based on a tag name, attribute, and attribute value