Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2000 Proceedings

International Conference on Information Systems (ICIS)

December 2000

Developing Internet Agents: A Tutorial Using Visual Basic 6.0

Gove Allen
University of Minnesota

Salvatore March
Vanderbilt University

DEVELOPING INTERNET AGENTS: A TUTORIAL USING VISUAL BASIC 6.0

Gove N. Allen
Carlson School of Management
University of Minnesota
U.S.A.

Salvatore T. March
David K. Wilson Professor of Management
Owen Graduate School of Management
Vanderbilt University
U.S.A.

1. INTRODUCTION

An agent is someone or something authorized to "act on behalf of" another person. In professional sports, for example, an athlete's agent may be authorized to negotiate the athlete's contract, but may or may not be authorized to accept the terms of a contract. Similarly, an Internet agent acts on behalf of a person who wishes to conduct some activity utilizing the Internet. The capabilities and authority invested in such an agent are at the discretion of the person it represents. Typically Internet agents perform search and data collection activities. They may or may not have authority to negotiate or conduct purchase or sale transactions.

Internet agents have varying levels of sophistication including lifespan, error detection and recovery, data validation, and embedded intelligence (Kauffman et al. 1999). A simple Internet agent, for example, may contact a single Web site (e.g., Amazon.com), extract a single fact (e.g., the price of a specified book) and report that fact to the user. A more sophisticated Internet agent may contact multiple Web sites (e.g., Amazon.com and BarnesAndNoble.com), track facts for several days or weeks (e.g., prices of a basket of books), record those facts for later analysis (e.g., in a database), and conduct transactions (e.g., purchase a subset of the basket of books when prices and availability meet given criteria).

Today's component-based, rapid application development environments allow individuals with very limited programming experience to build relatively sophisticated Internet agents without lengthy courses in Internet protocols or advanced programming techniques. Using development environments such as Visual Basic 6.0, simple but non-trivial Internet agents can be specified using a handful of components and a few dozen lines of code.

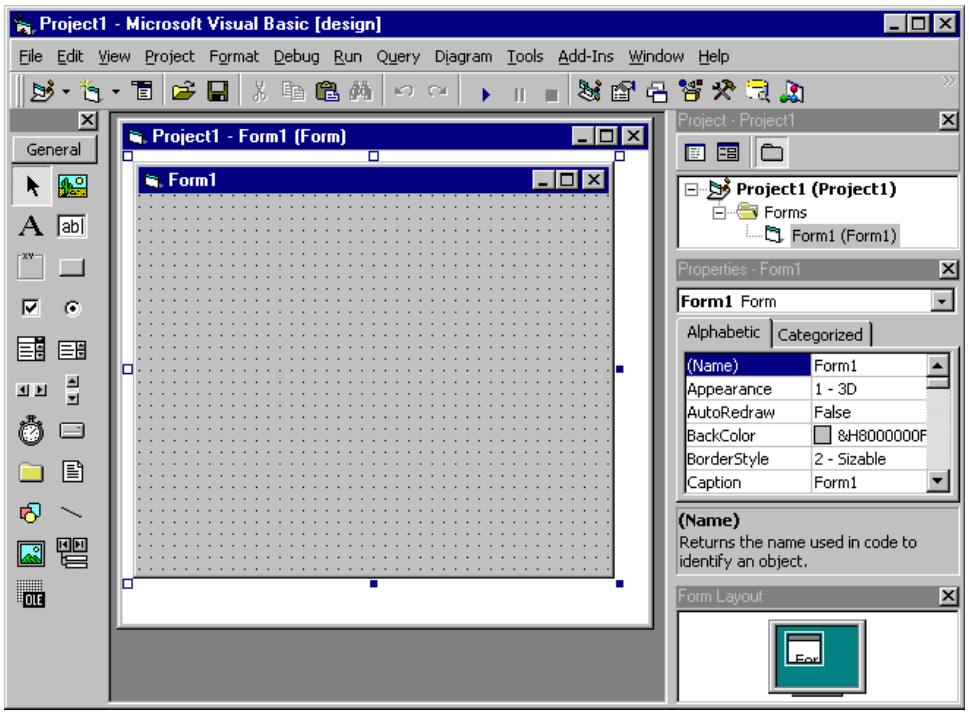
The following sections present a single example illustrating the most rudimentary capabilities needed to create an Internet agent. This agent merely retrieves the raw HTML from a specified URL. A more complete tutorial, available at http://www.internet-technology.org/tutorials/agents/visualbasic/march includes examples of more sophisticated agents having more useful capabilities. These include following links, extracting and interpreting the data, and storing that data in a database for later analysis.

2. SETTING UP NECESSARY VISUAL BASIC 6.0 COMPONENTS

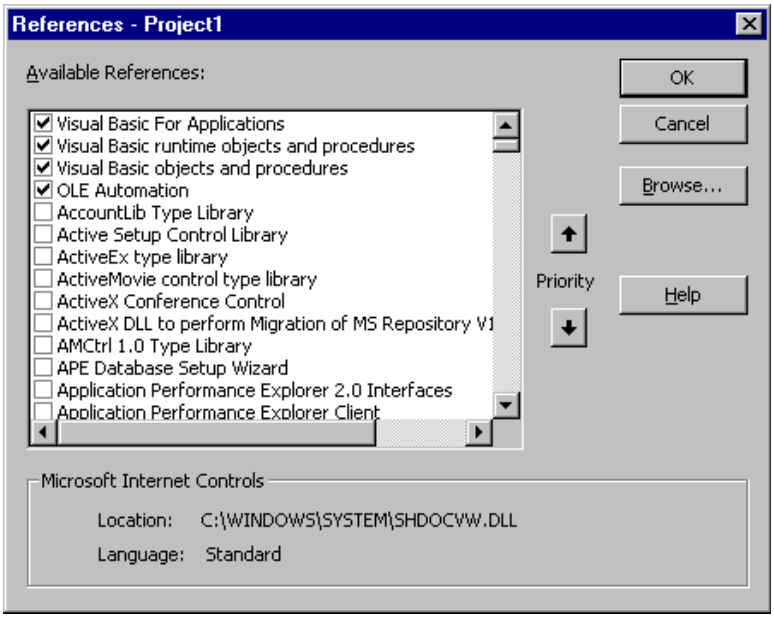
Aside from the standard Visual Basic 6.0 components, several custom components are extremely useful, if not essential for developing Internet agents. These are *Inet* (enables Internet connections), *Adodc* (enables simple database connections), and *DataGrid* (enables the simple display of database tables). These must be installed into the Visual Basic 6.0 library as follows.

1. Open Visual Basic and create a new project (use Standard.EXE). The following screen will be displayed. The toolbar is displayed at the left. The form window is displayed in the middle, and the Project, Properties, and Form Layout windows

are displayed at the right. The Properties window displays the properties of the selected component. Currently, by default, the form (Form1) is selected. Its properties are displayed in the Properties window. For example, its (Name) property is Form1. Its Caption property is also Form1.



2. Display the available library references. Select **Project -> References** from the main menu. The following window will be displayed.



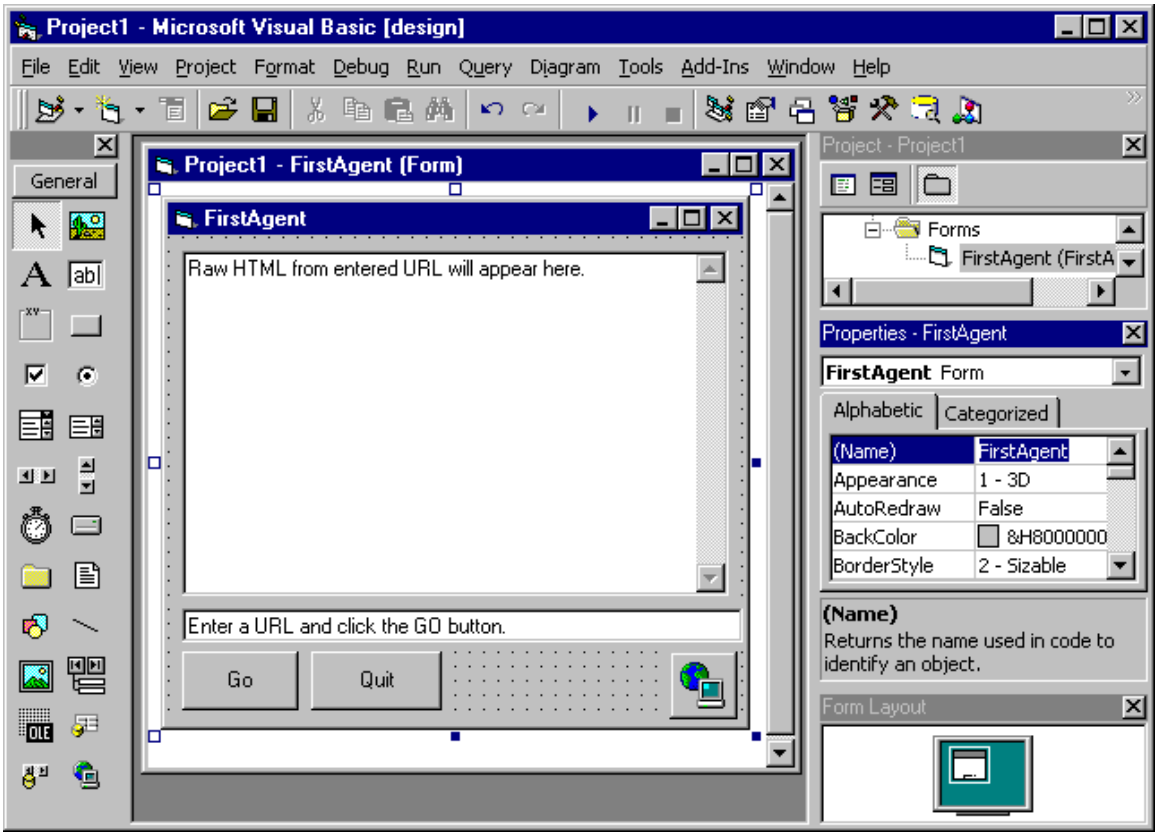
3. Add the following Microsoft libraries by scrolling down and clicking the check box preceding the named library:
   Microsoft ActiveX Data Objects 2.x Library
   Microsoft DAO 3.x Object Library
   Microsoft Data Binding Collection
   Then click the OK button.

4. Add the Inet, Adodb, and DataGrid components to the toolbox as follows. Select **Project -> Components** from the main menu. The Components window will open. With the Controls tab selected, scroll down and click the check box preceding the components:
   Microsoft ADO Data Control 6.x (OLEDB)
   Microsoft DataGrid Control 6.x (OLEDB)
   Microsoft Internet Transfer Control 6.x
   Then click the OK button. The Inet, Adodb, and DataGrid controls should now appear at the bottom of the toolbar.

3. FIRSTAGENT: CONTACTING A SITE BY URL

Length considerations prevent a complete discussion of helpful techniques for agent development; however, the most fundamental task that Internet agents perform is to request data from a server on the Internet and retrieve that data in a format which can be parsed for links or desired data. This tutorial demonstrates this basic task.

Contacting a Web site by its URL is the most basic capability an Internet agent must possess. FirstAgent will do so and display its raw HTML code. When completed, the form for FirstAgent should look as follows.



To use the agent, a URL must be entered into the lower text box and the Go button clicked. The raw HTML will then be displayed in the upper text box. This may be repeated any number of times. When the Quit button is pressed, the agent will terminate. Proceed to develop the agent as follows.

Text boxes, buttons, labels, and so forth are referred to as *Controls* in Visual Basic. The toolbox icons represent the different controls available. Pointing to a control will cause the "tool tip" giving the name of the control to be displayed. Point to the A in the toolbox. The tool tip, Label, should appear, indicating that this specifies a Label control.

1. Name the form FirstAgent by entering this text in the (Name) property (it defaults to Form1 when the form is created). Similarly, change its Caption property to FirstAgent. To change a property, click on the current value and replace the existing value.

2. Save the project. Select **File -> Save Project** and create a subfolder named FirstAgent in an appropriate folder. Then click the Save button to save the FirstAgent form (FirstAgent.frm). Change the project file name from Project1.vbp to FirstAgent and again click the Save button. You may add the project to Visual Source Save if you wish, but it is not necessary to do so.

3. Add an Inet control. Click the Inet control icon (the computer in front of the world icon) in the toolbox to select it. Then point to the form, press the left mouse button, and drag a spot for the control. The control should appear on the form and its properties should be displayed. Its default (Name) property will be Inet1.

4. Add the TextBox controls as illustrated above. Click the TextBox control icon (the shadowed ab| icon). Point to the form and drag a large textbox to display the raw HTML. Name it rawHTML and enter the text "Raw HTML from entered URL will appear here." in its Text property (without the quotes). Set its ScrollBars property to 2 - Vertical and its MultiLine property to True. Similarly add a TextBox for the URL. Name it searchURL and enter the appropriate text in its Text property, "Enter URL and click the Go button." It does not need a scroll bar nor will it have multiple lines, so leave these properties at their defaults.

5. Similarly add Go and Quit buttons. Name them goButton and quitButton, respectively, and set their Caption properties to Go and Quit, respectively.

6. Enter the code for the Go button. Double click the Go button to open its code window. Enter the code:

   ```
   rawHTML = Inet1.OpenURL(searchURL.Text)
   ```

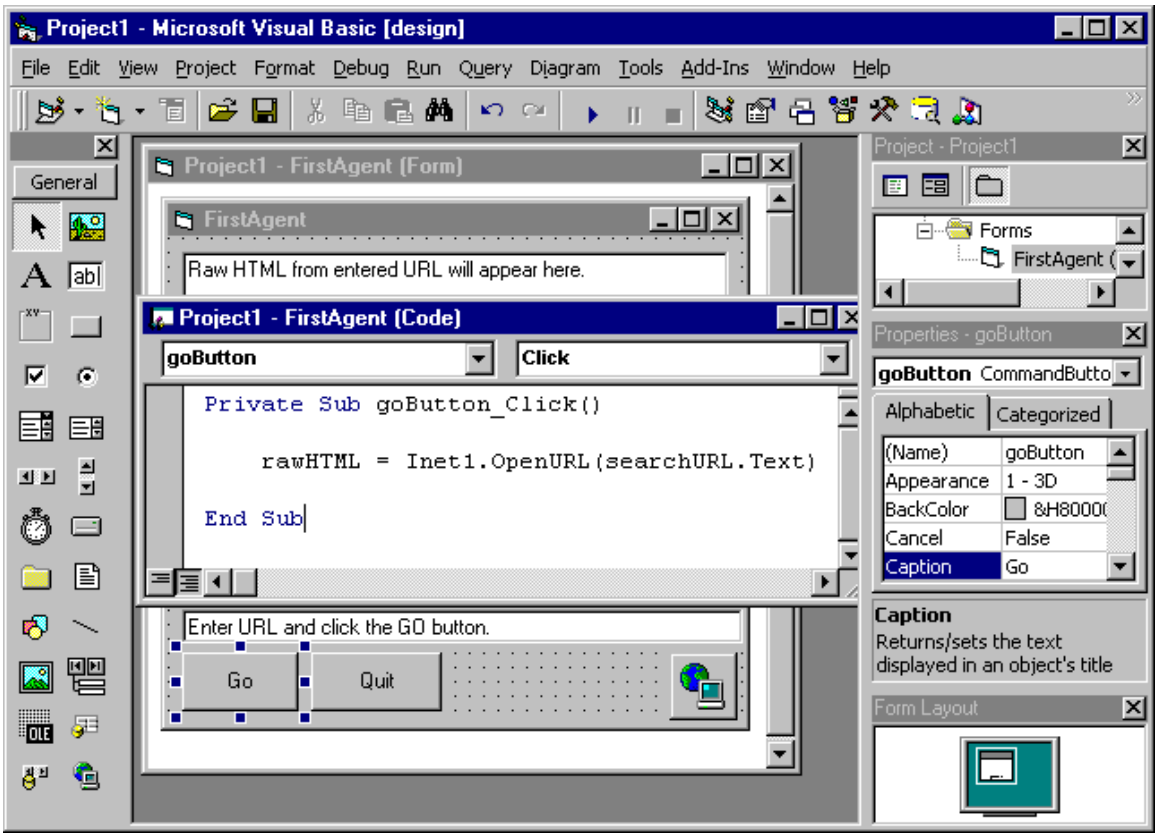
   as illustrated on the next page. This is interpreted as follows. The message OpenURL is sent to the Inet1 object. The parameter of this message is the text contained in the text box searchURL (this text is obtained by sending the message Text to the searchURL object, a TextBox). When the Inet1 object receives this message it contacts the Web site identified by the URL and obtains the named file. The contents of this file are assigned to the TextBox rawHTML where they are displayed. Close the code window.

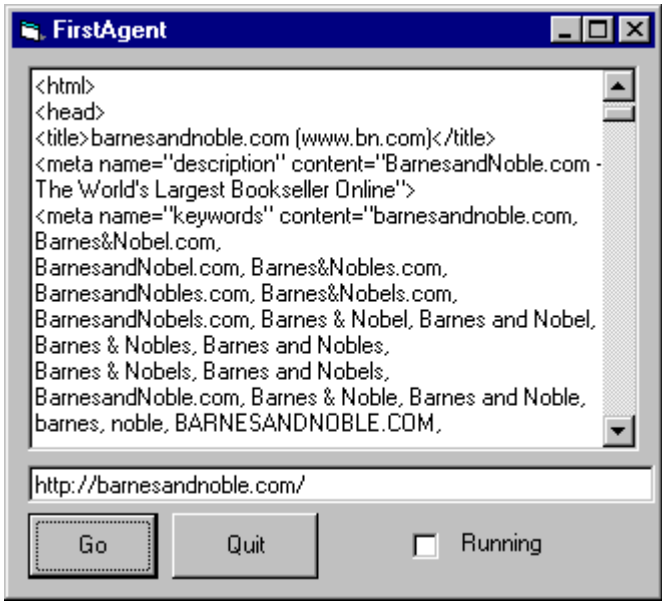
7. Similarly enter the code for the Quit button by double clicking it. Its code should be:

   ```
   Private Sub quitButton_Click()
       Unload Me
   End Sub
   ```

   The line Unload Me simply closes the form.

8. Save the project. Then try it out by selecting **Run -> Start** (or **Start With Full Compile** to check for errors) from the main menu. Then enter a URL. Enter http://BarnesAndNoble.com and click the Go button (actually the http:// is not required, BarnesAndNoble.com is sufficient). It may take a minute to make the contact. Do not click the Go button more than once or you will get an error message (we will worry about that below). After scrolling down a few lines, the result will be as illustrated in the second figure on the following page. Enter a different URL and click the Go button. You have just completed an agent that is capable of contacting any URL in existence and displaying its contents. While this is not overly exciting at the moment, recognize that the HTML is now available for analysis. It can be "mined" for content. Links can be identified and followed. Parameters can be added to those links to simulate user interaction. We indeed have the beginnings of a very powerful Internet Agent.





9. Clean up the error messages. When the Go button is clicked, the Subroutine GoButton_Click() is executed and the Inet1 object is asked to open the URL in the TextBox, searchURL. However, Inet1 cannot respond to such a request if it is already actively processing an earlier request, hence pressing the Go button before the request is completed generates an error. To

prevent this situation, add a CheckBox control to the form. Name it, runStatus. Double click the Go button to display the goButton_Click() code. Modify it as follows (modifications in bold):

```
Private Sub goButton_Click()
    If runStatus.Value = 0 Then
        runStatus.Value = 1
        rawHTML = Inet1.OpenURL(searchURL.Text)
        runStatus.Value = 0
    Else
        MsgBox ("Busy processing prior request.")
    End If
End Sub
```

The If statement checks the status of the runStatus CheckBox object. It has a default value of 0 (unchecked). Hence, when the Go button is first clicked, it has a value of 0 and the If part is executed. That is, the value of the runStatus CheckBox is set to 1 (checked) and the Inet1 object is asked to open the URL in the searchURL TextBox. If the user clicks the Go button again before that request has completed, the runStatus has a value of 1 (checked) and the message box is displayed with the message, "Busy processing prior request." After the request is completed, the value of the runStatus CheckBox is set back to 0 (unchecked).

This is the first of several agents that accomplish different tasks. Space limitations prevent the remaining five agents from being presented in this publication. The complete tutorial can be accessed at http://www.internet-technology.org/tutorials/agents/visualbasic/march.

Reference

Kauffman, R. J., March, S. T., and Wood, C. A. "Agent Sophistication: Design Aspects For Data-Collecting Agents" in *Proceedings of the Workshop on Information Technology Systems,* Charlotte, NC, December 11-12, 1999.