Lecture 9
Hypertext Applications (HTAs)

Print page

In Lecture 8, we learnt how to write a script to create and send an email. In this lecture, we consider another application of scripting technologies. We learn how to write an HTML Application (HTA). An HTML application is a Windows-based program written in a language that combines HTML and script languages. Such a combination packs the power of Web browsers and scripting. Due to the use of HTML, an HTA requires a Web browser to run. However, it does not require hosting on a Web server to run the embedded scripts because an HTA uses the client machine to run (you have to download and run it locally).

Key words

HTAs (Hypertext Applications), HTML, DHTML (Dynamic HTML)

Reference to textbook chapters

In this lecture we use an HTA program (ClientCheck.hta) written by Jeff Fellinge to explain the basic components and scripting techniques of HTAs. This code is presented in Jeff Fellinge, *IT Administrator's Top 10 Introductory Scripts for Windows*.

Why use HTAs

Let's start with a code snippet (download from embedScript.html), which is to send a message taken from a web page to a specified address.

```
<HTML>
<HEAD>
<TITLE>Web Mail</TITLE>
<SCRIPT LANGUAGE="VBScript">
Sub cmdSendEmail_OnClick
  Dim objMsg, objConfiguration, objConfigFields
  Set objConfiguration = CreateObject("CDO.Configuration")
  Set objConfigFields = objConfiguration.Fields

  objConfigFields.Item("http://schemas.microsoft.com/cdo/" & _
"configuration/sendusing") = 2
  objConfigFields.Item("http://schemas.microsoft.com/cdo/" & _
"configuration/smtpserverport") = 25
  objConfigFields.Item("http://schemas.microsoft.com/cdo/" & _
"configuration/smtpserver") = "email.uws.edu.au"
  objConfigFields.Update

  Set objMsg = CreateObject("CDO.Message")
  objMsg.Configuration = objConfiguration
  objMsg.Subject = "subject of the message here"
  objMsg.From = "j.yang@uws.edu.au"
  objMsg.To = Document.webmail.EmailAddress.Value
  objMsg.TextBody = Document.webmail.Message.Value
  objMsg.Send
End Sub
</SCRIPT>
</HEAD>
<BODY>
<H1>Web Mail</H1>
```

```
<FORM NAME="webmail">
  <B>EmailAddress: </B>
      <INPUT TYPE="Text" NAME="EmailAddress" SIZE=30>
  <br>
<B>Message: </B><INPUT TYPE="Text" NAME="Message" SIZE=80>
  <BR>
  <INPUT TYPE="Button" NAME="cmdSendEmail" VALUE="Send">
</FORM>
</BODY>
</HTML>
```

The coding structure is obvious - simply a combination of HTML text and VBScript code, where VBScript statements are embedded into an HTML file to create an interactive webpage. It tries to collect a message from the Message textfield and use CDO to send the message. Unfortunately this code does not work. This is because the Web browser that runs the HTML file has no access to the `CDO` object on the local machine. The Web browser security (sandbox) prevents a browser from accessing local system information. In fact, if you take the VBScript code out from the HMTL and make it a standalone VBScript program, it works well. So this sample code is not what we mean by HTML application.

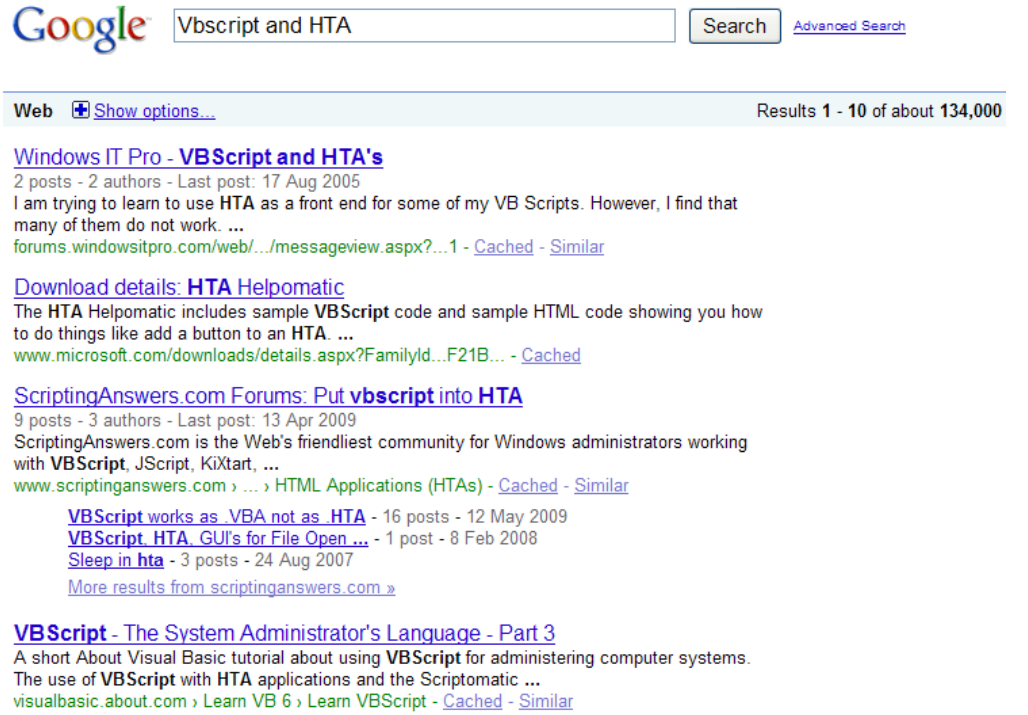
An HTA is an HTML-based script that runs locally in a web browser but is not served via HTTP from a web server. An HTA runs under the security context of the currently logged-in user who executes the script, which widens the script operational features beyond the limited to the browser. In fact, if you insert the following statement

```
<HTA:APPLICATION />
```

into the second line of the above HTML file and change the extension of the file name from .html to .hta, the program will work well. Download and run embedScript.hta (Note: save the code locally and change the email address and SMTP server name to fit into your case).

An HTA gets advantages from HTML and scripting languages. HTML provides an easy control of user interface design. Scripting technologies allows a full control on the client computer. Moreover, HTAs are not subject to the same security constraints as Web pages. Does this mean that HTAs are insecure? No. As with any executable file, the user is asked for permission of downloading an HTA. The user has to save and run the application in the local machine. In other words, the user is fully aware of the running of the program.

HTAs are suited to many uses, whether you are prototyping, making wizards, or building full-scale applications. Everything Dynamic HTML and script can deliver—forms, multimedia, Web applications, HTML editors, and browsers—HTAs can too. See Microsoft's HTA tutorial for more details: Introduction to HTML Applications (HTAs). You are also encouraged to search the Web.



HTA template

An HTA begins with the standard HTML framework but will likely include additional <script> tags to define the functionality of the application. The following is the general format of an HTA:

```
<html>
<HTA:APPLICATION options/>
<head>
<title>HTA test</title>
<style>Include Cascading Style Sheet information here</style>
<script>Include the program functionality here</script>
</head>
<body>
Include the HTML document defining the HTA format here and call functions or subroutines from this section.
</body>
</html>
```

It is easy to see that the ONLY visible difference of an HTA from a normal HTML file is the tag `<HTA:APPLICATION options/>` (besides the difference in file names). For instance, the following are the HTA options of the script ClientCheck.hta we are going to study in this lecture:

```
<HTA:APPLICATION
     ID="ClientCheck"
     APPLICATIONNAME="Client Diagnostic Tool"
     SCROLL="no"
     SINGLEINSTANCE="yes"
     WINDOWSTATE="normal"
     BORDER="thick"
     BORDERSTYLE="normal"
     MAXIMIZEBUTTON="yes"
     MINIMIZEBUTTON="yes"
     INNERBORDER="yes"
     SYSMENU="yes" />
```

Check out the options of HTA:APPLICATION for the meaning of the parameters and other options.

The second part of an HTA is the definition of layout style. You are allowed to use cascading style sheet (CCS) to define the look and feel of the application, including font style, size and color. The CSS styles are defined within the <style> tags. You can leave everything to be default if you do not seek for a fancy layout.

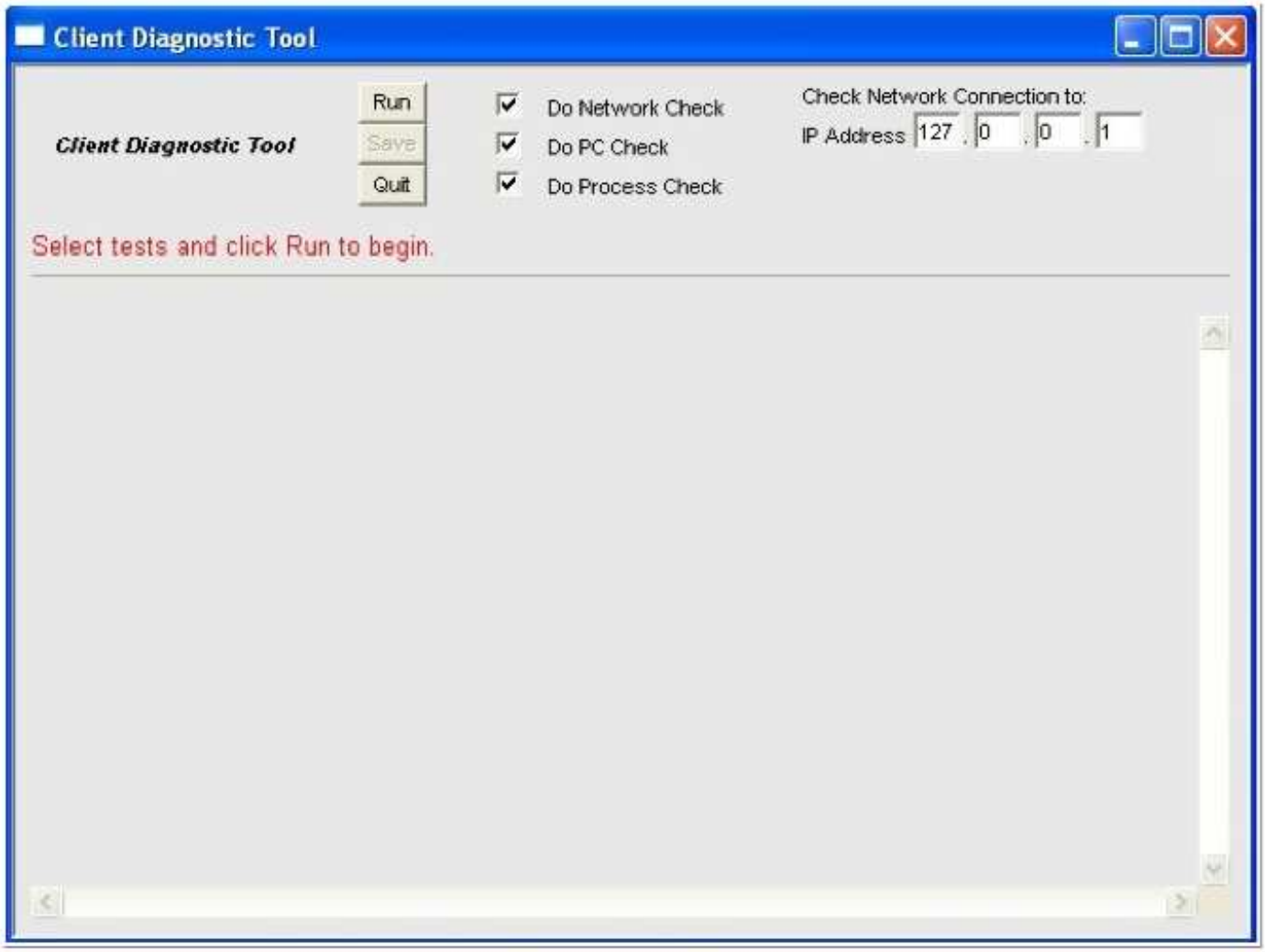
The third part, and possible the most tricky part, of an HTA is the script. You can write the script in either VBScript or JScript. The script is specified within the <script tags>. For instance,

<script language = "VBScript">

The final part is the body where you format a front-end. You can use HTML to define buttons, checkboxes, and text fields for collecting data and out put information.

Case study: Checking Client Machine

Jeff Fellinge introduced an HTA in his book: *IT Administrator's Top 10 Introductory Scripts for Windows*, which implements a tool that can be used to gather information for network connectivity, machine configuration and running processes from a client machine. The interface of the program ClientCheck.hta is the following:



The script accepts input from the user through HTML forms, including the buttons for controlling the process, the checkboxes for choosing different tasks and the text fields for specifying an IP address for testing.

Exactly following the template of HTAs, the file consists the following four components:

- HTA application declaration (within the tag <HTA:APPLICATION />)
- Defining document style (within the tag <style>)
- VBScript (within the tag <script>)
- HTML body (within the tag <body>)

Let's explain the script in more detail. The first subroutine `Window_Onload` is to configure the initial parameters of the HTA. Note that this subroutine is called automatically when the HTA is run. More specifically, it runs when the _onload event is fired.

```
Sub Window_Onload
```

```
Dim iReturn, objFSO, objShell, objLocator, strHTML
window.ResizeTo 640,480
cmdRun.disabled = False
cmdSave.disabled = True

On Error Resume Next
Set objLocator = CreateObject("WbemScripting.SWbemLocator")
If Err <> 0 Then
strHTML=Heading("The WMI Core Components do not " &_
"appear to be installed.","Red") & " As a result, both " &_
"the PC Check and Process Check have been disabled. <br><br>"
strHTML=strHTML & "<i>To run diagnostics, please download " &_
"and install the core WMI services for Windows.<br>" &_
"Download these files from Microsoft <a href=" & chr(34) &_
    "http://msdn.microsoft.com/library/default.asp?url" &_
    "=/downloads/list/wmi.asp" & chr(34) & ">here</a>." &_
    "</i><br><br>"
output.innerHTML= strHTML
chkPC.checked=False
chkPC.disabled=True
chkProcess.checked=False
chkProcess.disabled=True
End If
On Error Goto 0
Set objShell=CreateObject ("WScript.Shell")
g_strTemp=objShell.ExpandEnvironmentStrings("%temp%") &_
"/tempdiagnostics.txt"
End Sub
```

The subroutine first sets the size of the HTA widow, enables the Run button and disable the Save button. Next, it creates a `SWbemLocator` object. SWbemLocator is a scripting object that can get access to WMI on a particular host computer, much as the WMI moniker "winmgmts:" The next segment of code provides basic error handling.

The second subroutine `PerformDiagnostics` is used to launch the processes of PC check, Process check and Network check in response to the user's selection. The functions PCcheck, Processcheck and Networkcheck are designed to perform the functionality of checking information about the computer, current running processes, and network.

The PC Check subroutine uses WMI to demonstrate how to retrieve a variety of information from a local computer. The information is collected from five different WMI classes: `Win32_OperatingSystem`, `Win32_BIOS`, `Win32_ComputerSystem`, `Win32_Processor`, and `Win32_LogicalDisk`.

The Process Check subroutine also uses WMI in order to retrieve the information of the active processes. The WMI class that is used in the subroutine is `Win32_Process`, which contains the information about the active processes, such as the process name, the command line that was used to start it, how much memory the process has consumed, the number of page faults caused by the process, and others.

The Network Check subroutine uses WSH run methods and FSO methods to drive external applications such as ping, ipconfig, and tracert. Before running the test, the user can specify the IP address of a remote computer to measure the quality of the connection between the two computers. By default, this value is set to the 127.0.0.1, or localhost.

There are a few other subroutines or functions that are used for formating, save test results and quit the HTA. I would recommend all of you to read the code carefully for an in-depth understanding. Thanks for your time.