Imports System.IO

Imports System.Web

Imports System.Data

Imports System.Diagnostics

Imports System.Web.Services

Imports System.Web.Script.Services

<System.Web.Script.Services.ScriptService()> \_

<WebService(Namespace:="pdfHTMLDOC")> \_

<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1\_1)> \_

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> \_

Public Class PDFAjaxWebService

Inherits System.Web.Services.WebService

<WebMethod(EnableSession:=False)> \_

Public Function ajaxPDFGenerator(ByVal httpURL As String, ByVal httpURLVirtual As String, ByVal rawTxt As String) As XMLPDFDataBlock

' Our custom-defined "XML/SOAP" class

Dim xmlResponse As New XMLPDFDataBlock

' An array which will hold chunks of the StandardOutput stream

Dim generationResultsBin(-1) As String

' The location and filename where the temporary PDF will be created (in the case of generating PDF from submitted HTML code)

Dim localTempLocation As String = "\\webserverA\siteB\folder\temppdf.txt"

' The location where the physical PDF will be created

Dim localSaveLocation As String = "\\webserverA\siteB\folder\files"

' The results of operation message to send back

Dim generationResultsMsg As String = "An error has occurred with the web service."

' The URL to point a user to, in order to download the physical PDF

Dim generationResultsURL As String = "http://www.somesite.com/folder"

'

' Perform requested action

'

If Len(httpURL) > 0 Then

'

' Option #1: Call htmldoc and have it create a physical PDF from the URL supplied.

' NOTE: htmldoc cannot pull data from a URL, convert that into PDF, and then send that

' PDF data to StdOut; there has to be an intermediary save step.

'

localSaveLocation = localSaveLocation & "\temppdf.pdf"

Dim externalProcessPage As ProcessStartInfo = New ProcessStartInfo("htmldoc", "--webpage -f " & localSaveLocation & " " & httpURL)

externalProcessPage.RedirectStandardOutput = True ' Grab StdOut

externalProcessPage.RedirectStandardError = True ' Grab any errors (StdErr) that may come up if needed although not handled in this example

externalProcessPage.CreateNoWindow = True ' No need to create a window for this operation

externalProcessPage.UseShellExecute = False ' We want the raw output

Dim runProcessPage As Process = Process.Start(externalProcessPage)

runProcessPage.Start()

' Grab StdOut (null if successful)

Dim stdoutProcessPage As String = runProcessPage.StandardOutput.ReadToEnd()

If stdoutProcessPage <> "" Then

generationResultsMsg = Replace(stdoutProcessPage, vbCrLf, "\n")

generationResultsURL = ""

Else

generationResultsMsg = "A PDF has been created from the web page URL specified."

generationResultsURL = generationResultsURL & "/temppdf.pdf"

End If

ElseIf Len(httpURLVirtual) > 0 Then

'

' Option #2: Create a physical PDF from a static file on server containing html code

' (one that does not require parsing from server in order to render correctly).

'

localSaveLocation = localSaveLocation & "\temppdf.pdf"

Dim externalProcessPage As ProcessStartInfo = New ProcessStartInfo("htmldoc", "--webpage -f " & localSaveLocation & " " & httpURLVirtual)

externalProcessPage.RedirectStandardOutput = True ' Grab StdOut

externalProcessPage.RedirectStandardError = True ' Grab any errors (StdErr) that may come up if needed although not handled in this example

externalProcessPage.CreateNoWindow = True ' No need to create a window for this operation

externalProcessPage.UseShellExecute = False ' We want the raw output

Dim runProcessPage As Process = Process.Start(externalProcessPage)

runProcessPage.Start()

' Grab StdOut (null if successful)

Dim stdoutProcessPage As String = runProcessPage.StandardOutput.ReadToEnd()

If stdoutProcessPage <> "" Then

generationResultsMsg = Replace(stdoutProcessPage, vbCrLf, "\n")

generationResultsURL = ""

Else

generationResultsMsg = "A PDF has been created from the file containing static html code."

generationResultsURL = generationResultsURL & "/temppdf.pdf"

End If

ElseIf Len(rawTxt) > 0 Then

'

' Option #3: Create a physical file and return the data as stream blocks in XML/SOAP response

' (by default, htmldoc cannot pull html code from StdIn).

' NOTE: StdIn is limited by the amount of text it can contain anyway.

'

' Save the html code to a temporary file

Dim objFileSystemObject As StreamWriter

objFileSystemObject = File.CreateText(localTempLocation)

objFileSystemObject.Write(rawTxt)

objFileSystemObject.Close()

' Call htmldoc and have it return PDF stream blocks translated from the temporary file containing html code

Dim externalProcessPageVirtual As ProcessStartInfo = New ProcessStartInfo("htmldoc", "--quiet --webpage -t pdf " & localTempLocation)

externalProcessPageVirtual.RedirectStandardOutput = True ' Grab StdOut

externalProcessPageVirtual.RedirectStandardError = True ' Grab any errors (StdErr) that may come up if needed although not handled in this example

externalProcessPageVirtual.CreateNoWindow = True ' No need to create a window for this operation

externalProcessPageVirtual.UseShellExecute = False ' We want the raw output

Dim runProcessPageVirtual As Process = Process.Start(externalProcessPageVirtual)

runProcessPageVirtual.Start()

Dim generationResultsBinTemp As String = ""

generationResultsBinTemp = runProcessPageVirtual.StandardOutput.ReadToEnd

generationResultsMsg = "The text representation of the binary data generated at StandardOutput will be shown in a series of alert boxes."

Try

' Create a few child nodes for <generationResultsBin></generationResultsBin>

Dim parseFinished As Integer = 0

Dim lenBoundary As Integer = 2500 ' Maximum size of a child node before a new child node is created which holds a fragment of StandardOutput

Dim maximumNodeSize As Integer = 30000 ' Maximum size of all child nodes to be transported. If you want the to be able to transport a SOAP message much larger, you will need to make changes on the server, MaxRequestLength, etc.

If generationResultsBinTemp.Length < maximumNodeSize Then

maximumNodeSize = generationResultsBinTemp.Length

End If

Do Until parseFinished >= maximumNodeSize

If parseFinished + lenBoundary >= maximumNodeSize Then

ReDim Preserve generationResultsBin(UBound(generationResultsBin) + 1)

generationResultsBin(UBound(generationResultsBin)) = Mid(generationResultsBinTemp, parseFinished + 1, maximumNodeSize)

Else

ReDim Preserve generationResultsBin(UBound(generationResultsBin) + 1)

generationResultsBin(UBound(generationResultsBin)) = Mid(generationResultsBinTemp, parseFinished + 1, lenBoundary)

End If

parseFinished = parseFinished + lenBoundary

Loop

Catch ex As Exception

generationResultsMsg = Replace(ex.ToString, vbCrLf, "\n")

End Try

generationResultsURL = ""

Else

'

' Default message when incoming data is blank

'

generationResultsMsg = "Neither a URL, harddrive path or HTML code was submitted. No PDF document was created."

generationResultsURL = ""

End If

'

' Remove temp file if one was created

'

Dim objFile As FileInfo

objFile = New FileInfo(localTempLocation)

If objFile.Exists Then

objFile.Delete()

End If

'

' Build XML/SOAP stream going back to the user by letting .Net do it for us

'

With xmlResponse

.generationResultsMsg = generationResultsMsg

.generationResultsURL = generationResultsURL

.generationResultsBin = generationResultsBin

End With

Return xmlResponse

End Function

End Class

' "XML/SOAP" Class. Does not look like XML/SOAP here, but is described as such so you can see the parallels

' with what is created by using it.

' NOTE: .Net does all the work of creating and maintaining the XML, WSDL, ecetera, based on what we define

' here in our class. No need to handle writing our own definitions/files for XML/SOAP.

' HOW TO GET .Net TO TELL YOU WHAT THE XML/SOAP LOOKS LIKE:

' After performing a build of the project, if you wanted to find out what the XML/SOAP would look like

' that the web service received or sent, you could do the following:

' 1. In browser, type in www.yoursite.com/somefolder/PDFAjaxWebService.asmx?op=ajaxPDFGenerator

' NOTE: You will not be able to test a transaction using the URL and your browser on the server.

' 2. Under Solution Explorer, right-click on 'PDFAjaxWebService.asmx' and choose 'View In Browser'. A

' local session will be created on your local computer (where you are working on the project from)

' and you will have the option to simulate submitting test data to the web service.

' XML/SOAP CREATED BY THIS CLASS WILL LOOK LIKE THE FOLLOWING:

' <ajaxPDFGeneratorResponse xmlns="pdfHTMLDOC">

' <ajaxPDFGeneratorResult>

' <generationResultsBin>

' <string>string</string>

' <string>string</string>

' </generationResultsBin>

' <generationResultsMsg>string</generationResultsMsg>

' <generationResultsURL>string</generationResultsURL>

' </ajaxPDFGeneratorResult>

' </ajaxPDFGeneratorResponse>

Public Class XMLPDFDataBlock

' The following basically allows you to have an unlimited number of "child nodes" inside what may be

' considered the parent node "generationResultsBin".

Private \_generationResultsBin() As String

Public Property generationResultsBin() As String()

Get

Return \_generationResultsBin

End Get

Set(ByVal value() As String)

\_generationResultsBin = value

End Set

End Property

' This is the equivalent of a single XML node, which holds one value which is of the type "string".

Private \_generationResultsMsg As String

Public Property generationResultsMsg() As String

Get

Return \_generationResultsMsg

End Get

Set(ByVal value As String)

\_generationResultsMsg = value

End Set

End Property

' This is the equivalent of a single XML node, which holds one value which is of the type "string".

Private \_generationResultsURL As String

Public Property generationResultsURL() As String

Get

Return \_generationResultsURL

End Get

Set(ByVal value As String)

\_generationResultsURL = value

End Set

End Property

End Class