

Technical documentation

For professional programmers etc.

The Overview:

The benefits of using XML in this context

The specification's requirements are very demanding. The data must be processed and the solution must output XHTML and two different RSS feeds. It is a matter of fact that XHTML and RSS are actually both XML derivatives. This means that I am able to generate these files using XSLT (XML stylesheets). This makes fulfilling the requirements much easier.

The data is firstly inputted into a Visual Basic form located in a Word document. It is then stored as a XML file using the data structure specified below. Transformation of the XML then takes place and the contents of the XML file is turned into valid XHTML and RSS feeds. These newly created files are then automatically uploaded using FTP.

Storage & data modelling

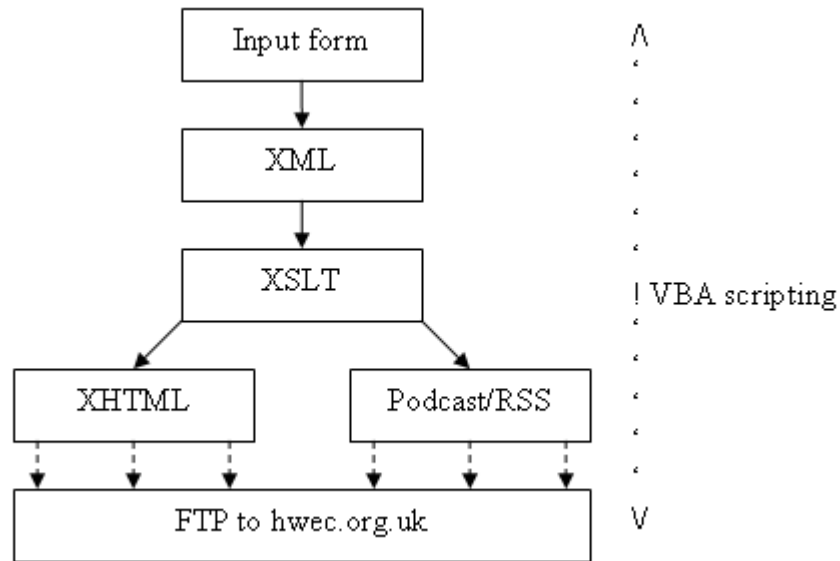
The custom solution must give full consideration of the need to structure the data appropriately (by using a clear naming convention and data modelling scheme) in order to control unwanted duplication and maximise efficiency. If I were using SQL in Access, I would normalise the attributes to the third normal form. However, I am using XML not SQL and normalising XML is a more complicated affair. Also, the problem does not require the solution to store a vast amount of data that's all interrelated.

I have come up with the following for the final scheme of storage:

```
<?xml version="1.0"?>
<archive>
  <sermon day="DD" month="MM" year="YYYY" time="AM/PM">
    <title>Title of sermon</title>
    <book>Book of the Bible</book>
    <chapter>CCC</chapter>
    <beginningverse>VVV</beginningverse>
    <endingverse>VVV</endingverse>
    <series>Title of series</series>
    <preacher>Preacher's name</preacher>
    <mp3>URL of audio</mp3>
    <ogg>URL of audio</ogg>
    <summary>Summary of sermon</summary>
    <durationmins>MM</durationmins>
    <durationsecs>SS</durationsecs>
    <filesize>File size in bytes</filesize>
    <datepub>DDD, MMM DD YYYY HH:MM:SS GMT</datepub>
  </sermon>
</archive>
```

The above hierarchial code is valid XML. The data of each sermon will be stored using this format.

Below is flow chart depicting the main concept behind solution design:



Function and routine list:

- Private Sub UserForm_Initialize()
- Function TimeTest() As String
- Sub EditLatestSermonXML()
- Public Function UpdateMe(NodeNum As Integer)
- Public Function RemoveMe(NodeNum As Integer)
- Public Function AddNewSermonNode(NodeName As String)
- Public Function TransformXML(ByRef XSLStyleSheet As String, ByRef InputXML As String, ByRef OutputXMLDeriv As String)
- Public Function QuerySermonNode(NodeNum As Integer)
- Function GenerateURL(FileFormat As String)
- Function FTPScr(FileName As String, Optional FileName2 As String, Optional FileName3 As String, Optional FileName4 As String, Optional FileName5 As String, Optional FileName6 As String, Optional FileName7 As String)
- Public Sub ClearForm()
- Private Sub btnSequentialSearch_Click()
- Private Sub btnAppendNode_Click()
- Private Sub btnClearForm_Click()
- Private Sub btnDeleteNode_Click()
- Private Sub btnGenURL_Click()
- Public Sub ExportFTP()
- Private Sub btnQueryNode_Click()
- Private Sub btnUpload_Click()
- Private Sub btnUpdateNode_Click()
- Private Sub btnNavigation_Change()

The Complete Code:

' 20070320 D.S Hollands

' Update sermons VBA form to valid XHTML via XML and XSLT - Version 3.0

Option Explicit

' Declaring global vars and constants '

' Begin/ ===== '

Const PATH As String = "E:\seq\"

Const SERMON_ARCHIVE As String = "sermons.xml"

Const LATEST_SERMON As String = "latest-sermon.xml"

Const NO_LOAD As String = "Cannot load: "

Const UPDATE_CURRENT As String = "update.xml"

Const CONTACT As String = ", contact David as soon as possible."

' ===== /end '

Private Sub UserForm_Initialize() 'Load title combo box thus enabling node search

Dim SermonArchiveXML As New XmlDocument50

Dim NodeList As IXMLDOMNodeList '<-- Assigns as a Node List (collection)

Dim Node As IXMLDOMNode

Dim i As Integer

SermonArchiveXML.async = False

If False = SermonArchiveXML.Load(PATH + SERMON_ARCHIVE) Then

MsgBox NO_LOAD & SERMON_ARCHIVE & CONTACT

Exit Sub

End If

Set NodeList = SermonArchiveXML.SelectNodes("//archive/*") '<-- makes use of XPath

For i = 0 To NodeList.Length - 1 '<-- pre-condition

Set Node = NodeList.NextNode

txtTitle.AddItem Node.ChildNodes(0).Text, i '<-- fill title combo box with current sermon node's title

Next 'loop until all sermon nodes titles are listed

End Sub

Function TimeTest() As String

' Tests the radio buttons and turns result into the appropriate string '

If radAM = True And radPM = False Then

 TimeTest = "AM"

Else

 TimeTest = "PM"

End If

End Function

Sub EditLatestSermonXML()

' Declaring local vars and constants '

' Begin/ ===== '

Dim LatestSermonXML As New MSXML2.DOMDocument

Dim strDatePub As String

Dim strTimePub As String

' ===== /end '

strDatePub = Format(Date, "ddd, mmm d yyyy")

strTimePub = Format(Time, "hh:mm:ss")

Call TimeTest

' Below dumps the text contained within the form's fields into a xml file in memory '

LatestSermonXML.loadXML _

"<sermon day=\"" & txtDay.Text & " month=\"" & txtMonth.Text & " year=\"" & txtYear.Text & " time=\"" & TimeTest & "\">" + vbNewLine + _

" <title>" & txtTitle.Text & "</title>" + vbNewLine + _

" <book>" & txtBook.Text & "</book>" + vbNewLine + _

" <chapter>" & txtChapter.Text & "</chapter>" + vbNewLine + _

" <beginningverse>" & txtBeginningVerse.Text & "</beginningverse>" + vbNewLine

+

" <endingverse>" & txtEndingVerse.Text & "</endingverse>" + vbNewLine + _

" <series>" & txtSeries.Text & "</series>" + vbNewLine + _

" <preacher>" & txtPreacher.Text & "</preacher>" + vbNewLine + _

" <mp3>" & txtMP3URL.Text & "</mp3>" + vbNewLine + _

" <ogg>" & txtOGGURL.Text & "</ogg>" + vbNewLine + _

" <summary>" & txtSummary.Text & "</summary>" + vbNewLine + _

" <durationmins>" & txtMins & "</durationmins>" + vbNewLine + _

" <durationsecs>" & txtSecs & "</durationsecs>" + vbNewLine + _

" <filesize>" & txtFilesize & "</filesize>" + vbNewLine + _

" <datepub>" & strDatePub & " " & strTimePub & " GMT</datepub>" + vbNewLine +

"</sermon>" + vbNewLine

' Saves the XML string located in memory to a file

LatestSermonXML.Save (PATH + LATEST_SERMON) '<-- saves the loaded XML

End Sub

Public Function UpdateMe(NodeNum As Integer) ' Update node using replace child

' Declaring local vars and constants '

' ===== '

Dim UpdateXML As New MSXML2.DOMDocument50

Dim UpdatingXML As FreeThreadedDOMDocument50

Dim SermonArchiveXML As DOMDocument50

Dim Node As IXMLDOMNode

Dim NodeClone As IXMLDOMNode

Dim msg As String

Dim strDatePub As String

Dim strTimePub As String

' ===== '

Call TimeTest

strDatePub = Format(Date, "ddd, mmm d yyyy")

strTimePub = Format(Time, "hh:mm:ss")

' Below dumps the text contained within the form's fields into a xml file in memory '

UpdateXML.loadXML _

"<sermon day=" & txtDay.Text & " month=" & txtMonth.Text & " year=" &
txtYear.Text & " time=" & TimeTest & ">" + vbNewLine + _
" <title>" & txtTitle.Text & "</title>" + vbNewLine + _
" <book>" & txtBook.Text & "</book>" + vbNewLine + _
" <chapter>" & txtChapter.Text & "</chapter>" + vbNewLine + _
" <beginningverse>" & txtBeginningVerse.Text & "</beginningverse>" + vbNewLine
+ _
" <endingverse>" & txtEndingVerse.Text & "</endingverse>" + vbNewLine + _
" <series>" & txtSeries.Text & "</series>" + vbNewLine + _
" <preacher>" & txtPreacher.Text & "</preacher>" + vbNewLine + _
" <mp3>" & txtMP3URL.Text & "</mp3>" + vbNewLine + _
" <ogg>" & txtOGGURL.Text & "</ogg>" + vbNewLine + _
" <summary>" & txtSummary.Text & "</summary>" + vbNewLine + _
" <durationmins>" & txtMins & "</durationmins>" + vbNewLine + _
" <durationsecs>" & txtSecs & "</durationsecs>" + vbNewLine + _
" <filesize>" & txtFilesize & "</filesize>" + vbNewLine + _
" <datepub>" & strDatePub & " " & strTimePub & " GMT</datepub>" + vbNewLine +
_

" </sermon>" + vbNewLine

'MsgBox (UpdateXML.XML)

UpdateXML.Save (PATH & UPDATE_CURRENT) '<-- saves current input as update
xml file

Set UpdatingXML = New FreeThreadedDOMDocument50

Set SermonArchiveXML = New DOMDocument50

SermonArchiveXML.async = False

If False = SermonArchiveXML.Load(PATH + SERMON_ARCHIVE) Then '<--
loading sermon archive xml

```
MsgBox NO_LOAD & SERMON_ARCHIVE & CONTACT
Exit Function
End If
```

```
UpdatingXML.async = False
If False = UpdatingXML.Load(PATH + UPDATE_CURRENT) Then '<-- loading
update xml ready for replacing
MsgBox NO_LOAD & UPDATE_CURRENT & CONTACT
Exit Function
End If
```

```
' Clone update node for replacing currently selected sermon node in
SermonArchiveXML '
' replaceChild method is used '
```

```
Set Node = UpdatingXML.SelectSingleNode("/sermon")
Set NodeClone = SermonArchiveXML.importNode(Node, True)
SermonArchiveXML.documentElement.replaceChild NodeClone,
SermonArchiveXML.documentElement.ChildNodes.Item(NodeNum)
SermonArchiveXML.documentElement.appendChild
SermonArchiveXML.createTextNode(vbNewLine) '<-- appends a new line under
updated node
```

```
Set Node = Nothing
Set NodeClone = Nothing
```

```
SermonArchiveXML.Save (PATH + SERMON_ARCHIVE) '<-- save updated sermon
xml
msg = "The sermon archive has been updated."
```

```
MsgBox msg
```

```
End Function
```

Public Function RemoveMe(NodeNum As Integer) ' nodenum as integer

Dim SermonArchiveXML As DOMDocument50

Dim Node As IXMLDOMNode

Dim root

Dim currNode

Dim oldChild

Set SermonArchiveXML = New DOMDocument50

SermonArchiveXML.async = False

If False = SermonArchiveXML.Load(PATH + SERMON_ARCHIVE) Then

MsgBox NO_LOAD & SERMON_ARCHIVE & CONTACT

Exit Function

End If

Set root = SermonArchiveXML.documentElement

Set oldChild = root.RemoveChild(root.ChildNodes.Item(NodeNum))

MsgBox oldChild.Text

SermonArchiveXML.Save (PATH & SERMON_ARCHIVE)

End Function

Public Function AddNewSermonNode(NodeName As String) ' NodeName must be "sermon" '

' Declaring local vars and constants '

' ===== '

Dim LatestSermonXML As FreeThreadedDOMDocument50

Dim SermonArchiveXML As DOMDocument50

Dim Node As IXMLDOMNode

Dim NodeClone As IXMLDOMNode

Dim msg As String

' ===== '

Set LatestSermonXML = New FreeThreadedDOMDocument50

Set SermonArchiveXML = New DOMDocument50

SermonArchiveXML.async = False

If False = SermonArchiveXML.Load(PATH + SERMON_ARCHIVE) Then

MsgBox NO_LOAD & SERMON_ARCHIVE & CONTACT

Exit Function

End If

LatestSermonXML.async = False

If False = LatestSermonXML.Load(PATH + LATEST_SERMON) Then

MsgBox NO_LOAD & LATEST_SERMON & CONTACT

Exit Function

End If

' Copy the sermon node from LatestSermonXML to SermonArchiveXML: '

' Fetch the "/sermon" (node) from LatestSermonXML (LATEST_SERMON) '

' Clone node for import to SermonArchiveXML '

' Append clone to SermonArchiveXML (SERMON_ARCHIVE) '

Set Node = LatestSermonXML.SelectSingleNode("/") & NodeName)

Set NodeClone = SermonArchiveXML.importNode(Node, True)

SermonArchiveXML.documentElement.appendChild NodeClone

SermonArchiveXML.documentElement.appendChild

SermonArchiveXML.createTextNode(vbNewLine)

Set Node = Nothing

Set NodeClone = Nothing

SermonArchiveXML.Save (PATH + SERMON_ARCHIVE)

msg = "The sermon archive has been updated."

MsgBox msg

End Function

Public Function TransformXML(ByRef XSLStyleSheet As String, ByRef InputXML As String, ByRef OutputXMLDeriv As String)

```
' Declaring local vars and constants '  
' ===== '  
Dim xslt As New MSXML2.XSLTemplate50  
Dim xslDoc As New MSXML2.FreeThreadedDOMDocument50  
Dim xmlDoc As New MSXML2.DOMDocument50  
Dim xslProc As MSXML2.IXSLProcessor '<-- only works with MSXML 3.0 or higher  
Dim OutputDoc As New MSXML2.DOMDocument50  
Dim Output As New MSXML2.DOMDocument50  
Dim Err '<-- declaring error var  
' ===== '
```

'MSXML2.30 (MSXML 3.0) or higher is used because lower versions do not have an IXSL Processor

```
xslDoc.Load (PATH & XSLStyleSheet) '<-- loads the xsl ready for transformation  
xslDoc.async = False
```

```
If (xslDoc.parseError.errorCode <> 0) Then '<-- checks the xslt is valid  
Set Err = xslDoc.parseError  
MsgBox ("Error: " & Err.reason) '<-- explains error  
Else  
Set xslt.stylesheet = xslDoc '<-- assigns xslDoc as the Stylesheet  
xmlDoc.Load (PATH & InputXML) '<-- loads the xml file  
End If
```

```
If (xmlDoc.parseError.errorCode <> 0) Then '<-- checks the xml is valid  
Set Err = xmlDoc.parseError  
MsgBox ("Error: " & Err.reason) '<-- explains error  
Else  
Set xslProc = xslt.createProcessor() '<-- initialises processor  
xslProc.input = xmlDoc '<-- inputs the valid xml file  
xslProc.transform '<-- transforms the xmlDoc using the xslDoc  
OutputDoc.loadXML (xslProc.Output) '<-- Loads the processors output  
OutputDoc.Save (PATH & OutputXMLDeriv) '<-- Saves the output  
End If
```

End Function

**Public Function QuerySermonNode(NodeNum As Integer)' Query sermon xml
function**

'Declaring vars and constants

```
=====
Dim oXMLDoc As New MSXML2.DOMDocument30 '<-- DOMDocument30 because
of functionality
Dim oNode As IXMLDOMNode
Dim oNodes As IXMLDOMNodeList '<-- Assigns oNodes as a Node List (collection)
Dim i As Integer
Dim detailName As String
Dim dName As String
Dim dText As String
=====
```

' Assign necessary truths/properties to XMLDocument

```
oXMLDoc.async = False
oXMLDoc.validateOnParse = False
oXMLDoc.resolveExternals = False
oXMLDoc.preserveWhiteSpace = True '<-- Keeps the white spaces inside the xml data
```

' Load XML data from latest-sermon.xml file

```
If oXMLDoc.Load(PATH + SERMON_ARCHIVE) = False Then
MsgBox "Failed to load latest sermon xml data from file."
Exit Function
End If
```

' Commence querying the sermon node-set

=====

' Select attributes of node and assign as appropriate

```
Dim SermonNode As IXMLDOMElement
Dim DayValue As String
Dim MonthValue As String
Dim YearValue As String
Dim TimeValue As String
```

If (oXMLDoc.parseError.errorCode <> 0) Then '<-- error checking

```
Dim myErr
Set myErr = oXMLDoc.parseError
MsgBox ("You have error " & myErr.reason)
Else
Set SermonNode = oXMLDoc.SelectSingleNode("//sermon[" & NodeNum & "]")
```

```
DayValue = SermonNode.getAttribute("day")
txtDay.Text = DayValue
```

```
MonthValue = SermonNode.getAttribute("month")
txtMonth.Text = MonthValue
```

```
YearValue = SermonNode.getAttribute("year")
txtYear.Text = YearValue
```

```
TimeValue = SermonNode.getAttribute("time")
If TimeValue = "AM" Then
    radAM = True
    radPM = False
Else
    radPM = True
End If
End If
```

```
' Navigate to and load contents of first sermon node as a node list (XPath)
' Integer NodeNum is inserted here...
Set oNodes = oXMLDoc.SelectNodes("//sermon[" & NodeNum & "]/*")
```

```
For i = 0 To oNodes.Length - 1 '<-- Loop until list ends
```

```
' Jumps the "sermon" node and goes straight to the "title" node
' then next time goes to the next node in the list until list ends
Set oNode = oNodes.NextNode
```

```
' Check that node name is not empty (error checking - maybe unnecessary??)
If Not (oNode Is Nothing) Then
```

```
'Makes coding less tedious - consider making function though
dName = oNode.NodeName '<-- dName = detail name = node name
dText = oNode.Text '<-- dText = detail text = text in node
```

```
'Check what node name is and assign node's text to correct text box
Select Case dName '<-- (nodeName)
```

```
Case Is = "title"
    txtTitle = dText '<-- Fills text box with node text
Case Is = "book"
    txtBook = dText
Case Is = "chapter"
    txtChapter = dText
Case Is = "beginningverse"
    txtBeginningVerse = dText
Case Is = "endingverse"
    txtEndingVerse = dText
Case Is = "series"
    txtSeries = dText
Case Is = "preacher"
    txtPreacher = dText
Case Is = "mp3"
    txtMP3URL = dText
Case Is = "ogg"
    txtOGGURL = dText
Case Is = "summary"
    txtSummary = dText
Case Is = "durationmins"
```

```
txtMins = dText
Case Is = "durationsecs"
txtSecs = dText
Case Is = "filesize"
txtFilesize = dText
End Select
End If
```

```
Next '<-- looping...
```

```
' function.... QuerySermonNode(x)
```

End Function

Function GenerateURL(FileFormat As String)

```
Dim URL As String
Const Dir As String = "http://hwec.org.uk/audio/"
```

```
Call TimeTest
```

```
URL = Dir & FileFormat & "/" & txtYear & "/" & txtYear & txtMonth & txtDay &
TimeTest & "." & FileFormat
```

```
If FileFormat = "mp3" Then
txtMP3URL = URL
Else
txtOGGURL = URL
End If
```

End Function

**Function FTPScr(FileName As String, Optional FileName2 As String, Optional
FileName3 As String, Optional FileName4 As String, Optional FileName5 As
String, Optional FileName6 As String, Optional FileName7 As String)**

' This function makes a .scr (script) file containing the FTP instructions for uploading
' either 2, 5 or 7 files.

```
Dim sSCR As String
Dim sEXPFile As String
Dim sPutFile As String
Dim sEXPFile2 As String
Dim sPutSecondFile As String
Dim sEXPFile3 As String
Dim sPutThirdFile As String
Dim sEXPFile4 As String
Dim sPutFourthFile As String
Dim sEXPFile5 As String
Dim sPutFifthFile As String
Dim sEXPFile6 As String
Dim sPutSixthFile As String
Dim sEXPFile7 As String
Dim sPutSeventhFile As String
Dim iFreeFile
Const strFTP_SERVE As String = "ftp.hwec.org.uk"
Const strFTP_LOGIN As String = "[removed for security reasons]"
Const strFTP_PASSWORD As String = "[removed for security reasons]"
Const strFTP_DIR As String = "aa/archive/"

sSCR = PATH & "FTP.scr"
sEXPFile = FileName
sEXPFile2 = FileName2
sEXPFile3 = FileName3
sEXPFile4 = FileName4
sEXPFile5 = FileName5
sEXPFile6 = FileName6
sEXPFile7 = FileName7
sPutFile = "Put " & sEXPFile
sPutSecondFile = "Put " & sEXPFile2
sPutThirdFile = "Put " & sEXPFile3
sPutFourthFile = "Put " & sEXPFile4
sPutFifthFile = "Put " & sEXPFile5
sPutSixthFile = "Put " & sEXPFile6
sPutSeventhFile = "Put " & sEXPFile7

iFreeFile = FreeFile

If Len(sEXPFile3) > 0 Then

If Len(sEXPFile5) > 0 And Len(sEXPFile7) = 0 Then ' 5 files
Open sSCR For Output As iFreeFile
Print #iFreeFile, "lcd " & PATH
```

```

Print #iFreeFile, "open " & strFTP_SERVE
Print #iFreeFile, strFTP_LOGIN
Print #iFreeFile, strFTP_PASSWORD
Print #iFreeFile, "cd " & strFTP_DIR
Print #iFreeFile, "binary"
Print #iFreeFile, sPutFile
Print #iFreeFile, sPutSecondFile
Print #iFreeFile, sPutThirdFile
Print #iFreeFile, sPutFourthFile
Print #iFreeFile, sPutFifthFile
Print #iFreeFile, "bye"
Close #iFreeFile
Else
Open sSCR For Output As iFreeFile ' 7 files
Print #iFreeFile, "lcd " & PATH
Print #iFreeFile, "open " & strFTP_SERVE
Print #iFreeFile, strFTP_LOGIN
Print #iFreeFile, strFTP_PASSWORD
Print #iFreeFile, "cd " & strFTP_DIR
Print #iFreeFile, "binary"
Print #iFreeFile, sPutFile
Print #iFreeFile, sPutSecondFile
Print #iFreeFile, sPutThirdFile
Print #iFreeFile, sPutFourthFile
Print #iFreeFile, sPutFifthFile
Print #iFreeFile, sPutSixthFile
Print #iFreeFile, sPutSeventhFile
Print #iFreeFile, "bye"
Close #iFreeFile
End If
Else
Open sSCR For Output As iFreeFile ' 2 files
Print #iFreeFile, "lcd " & PATH
Print #iFreeFile, "open " & strFTP_SERVE
Print #iFreeFile, strFTP_LOGIN
Print #iFreeFile, strFTP_PASSWORD
Print #iFreeFile, "cd " & strFTP_DIR
Print #iFreeFile, "binary"
Print #iFreeFile, sPutFile
Print #iFreeFile, sPutSecondFile
Print #iFreeFile, "bye"
Close #iFreeFile
End If

```

End Function

Public Sub ClearForm()

Const EmptyString As String = ""

txtDay = "DD"
txtMonth = "MM"
txtYear = "YYYY"
radAM = False
radPM = False

txtTitle = EmptyString
txtBook = EmptyString
txtChapter = "CCC"
txtBeginningVerse = "VVV"
txtEndingVerse = "VVV"
txtSeries = EmptyString
txtPreacher = EmptyString
txtMP3URL = EmptyString
txtOGGURL = EmptyString
txtSummary = EmptyString
txtFilesize = EmptyString
txtMins = "MM"
txtSecs = "SS"

End Sub

Private Sub btnSequentialSearch_Click() ' search for selected nodes

If txtTitle.ListIndex >= 0 Then
 QuerySermonNode (txtTitle.ListIndex) '<-- search xml for currently selected sermon
 title
 btnNavigation.Value = txtTitle.ListIndex '<-- change
Else
End If

End Sub

Private Sub btnAppendNode_Click()

Dim newi As Integer
Dim strTitle As String

Call EditLatestSermonXML

Call AddNewSermonNode("sermon")

newi = txtTitle.ListCount + 1
strTitle = txtTitle.Text

txtTitle.AddItem strTitle

btnNavigation.Value = txtTitle.ListCount + 1

End Sub**Private Sub btnClearForm_Click()**

Call ClearForm

End Sub**Private Sub btnDeleteNode_Click()**

If btnNavigation.Value >= 0 Then

If MsgBox("Are you sure you want to delete this sermon?", vbYesNo, "Delete sermon function") = vbYes Then

Call RemoveMe(btnNavigation.Value)

Call ClearForm

txtTitle.RemoveItem (btnNavigation.Value)

btnNavigation.Value = btnNavigation.Value - 1

Else
End If

Else
End If

End Sub**Private Sub btnGenURL_Click()**

Call GenerateURL("mp3")
Call GenerateURL("ogg")

End Sub

Public Sub ExportFTP()

Dim sSCR As String, sDir As String, sExe As String

sSCR = PATH & "FTP.scr" '<-- links to file created in FTPScr (ftp instructions)

sDir = Environ\$("COMSPEC")

sDir = Left\$(sDir, Len(sDir) - Len(Dir(sDir)))

sExe = sDir & "ftp.exe -s:" & sSCR '<-- sticks ftp instructions file on as a string

Shell sExe, vbNormalNoFocus '<-- shells out ftp.exe

End Sub**Private Sub btnQueryNode_Click()**

QuerySermonNode (0)

btnNavigation.Value = 0

lblNodeNum.Caption = btnNavigation.Value

End Sub**Private Sub btnUpload_Click() '<-- Transform selection, notify usr and upload files**

Dim msg As String

Const XSL_LS As String = "latest-sermon.xslt"

Const XSL_SA As String = "sermons.xslt"

Const XSL_SA_DATE As String = "byDate.xslt"

Const XSL_SA_TITLE As String = "byTitle.xslt"

Const XSL_SA_BOOK As String = "byBook.xslt"

Const XSL_SF As String = "feed.xslt"

Const XSL_PC As String = "podcast.xslt"

Const XML_LS As String = "latest-sermon.xml"

Const XML_SA As String = "sermons.xml"

Const XML_PC As String = "podcast.xml"

Const XML_SF As String = "sermon-feed.xml"

Const XHTML_LS As String = "latest-sermon.html"

Const XHTML_SA As String = "sermons.html"

Const XHTML_SA_DATE As String = "byDate.html"

Const XHTML_SA_TITLE As String = "byTitle.html"

Const XHTML_SA_BOOK As String = "byBook.html"

msg = "The following files have been created/updated:" + vbNewLine

If cboRSS = True And cboXHTML = True Then

' Transform all

Call TransformXML(XSL_SF, XML_SA, XML_SF) ' rss feed

msg = msg + " " + XML_SF + vbNewLine

Call TransformXML(XSL_PC, XML_SA, XML_PC) ' podcast

msg = msg + " " + XML_PC + vbNewLine

Call TransformXML(XSL_LS, XML_LS, XHTML_LS) ' latest-sermon.html

msg = msg + " " + XHTML_LS + vbNewLine

```

Call TransformXML(XSL_SA, XML_SA, XHTML_SA) ' sermons.html
msg = msg + " " + XHTML_SA + vbNewLine
Call TransformXML(XSL_SA_DATE, XML_SA, XHTML_SA_DATE) '
byDate.html
msg = msg + " " + XHTML_SA_DATE + vbNewLine
Call TransformXML(XSL_SA_TITLE, XML_SA, XHTML_SA_TITLE) '
byTitle.html
msg = msg + " " + XHTML_SA_TITLE + vbNewLine
Call TransformXML(XSL_SA_BOOK, XML_SA, XHTML_SA_BOOK) '
byBook.html
msg = msg + " " + XHTML_SA_BOOK + vbNewLine
' Notify user
MsgBox msg
' Upload all
Call FTPScr(PATH & XML_SF, PATH & XML_PC, PATH & XHTML_LS, PATH
& XHTML_SA, PATH & XHTML_SA_DATE, PATH & XHTML_SA_TITLE, PATH
& XHTML_SA_BOOK)
Call ExportFTP
Else
If cboRSS = True And cboXHTML = False Then
' Transform into RSS only
Call TransformXML(XSL_SF, XML_SA, XML_SF) ' rss feed
msg = msg + " " + XML_SF + vbNewLine
Call TransformXML(XSL_PC, XML_SA, XML_PC) ' podcast
msg = msg + " " + XML_PC + vbNewLine
' Notify user
MsgBox msg
' Upload RSS only
Call FTPScr(PATH & XML_SF, PATH & XML_PC)
Call ExportFTP
Else
If cboRSS = False And cboXHTML = True Then
' Transform into XHTML only
Call TransformXML(XSL_LS, XML_LS, XHTML_LS) ' latest-sermon.html
msg = msg + " " + XHTML_LS + vbNewLine
Call TransformXML(XSL_SA, XML_SA, XHTML_SA) ' sermons.html
msg = msg + " " + XHTML_SA + vbNewLine
Call TransformXML(XSL_SA_DATE, XML_SA, XHTML_SA_DATE) '
byDate.html
msg = msg + " " + XHTML_SA_DATE + vbNewLine
Call TransformXML(XSL_SA_TITLE, XML_SA, XHTML_SA_TITLE) '
byTitle.html
msg = msg + " " + XHTML_SA_TITLE + vbNewLine
Call TransformXML(XSL_SA_BOOK, XML_SA, XHTML_SA_BOOK) '
byBook.html
msg = msg + " " + XHTML_SA_BOOK + vbNewLine
' Notify user
MsgBox msg
' Upload XHTML only
Call FTPScr(PATH & XHTML_LS, PATH & XHTML_SA, PATH &
XHTML_SA_DATE, PATH & XHTML_SA_TITLE, PATH & XHTML_SA_BOOK)
Call ExportFTP

```

```
Else
  ' Do Nothing
End If
End If
End If
```

End Sub

Private Sub btnUpdateNode_Click()

```
Dim msg As String
```

```
'UpdateMe() - update sermon.xml
```

```
If btnNavigation.Value >= 0 Then
  Call UpdateMe(btnNavigation.Value) '<-- updates currently selected sermon
Else
End If
```

End Sub

Private Sub btnNavigation_Change() '<-- spin navigational button

```
Dim oNodes As IXMLDOMNodeList '<-- Assigns oNodes as a Node List (collection)
Dim oXMLDoc As New MSXML2.DOMDocument50
Dim Max As Integer
Dim TotalNodes As Integer
Dim Num As Integer
```

```
' Load XML data from latest-sermon.xml file
If oXMLDoc.Load(PATH + SERMON_ARCHIVE) = False Then
  MsgBox "Failed to load latest sermon xml data from file."
  Exit Sub
End If
```

```
Set oNodes = oXMLDoc.SelectNodes("//archive/*")
```

```
Max = oNodes.Length - 1
```

```
If btnNavigation.Value >= 0 Then
  If btnNavigation.Value <= Max Then
    QuerySermonNode (btnNavigation.Value)
  Else
    btnNavigation.Value = 0
  End If
Else
  QuerySermonNode (Max)
  btnNavigation.Value = Max
End If
```

```
lblNodeNum.Caption = btnNavigation.Value
```

End Sub