

MỤC LỤC

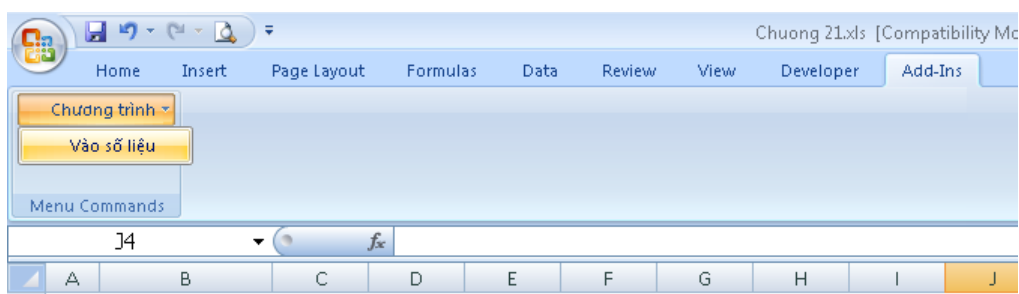
Contents

MỤC LỤC.....	1
1. Mô hình quản lý với Ribbon	3
2. Xây dựng một tab mới cùng với group	6
3. Cấu trúc mã XML và tìm hiểu đối tượng Ribbon	9
3.1. Cấu trúc mã XML:	9
3.2. Các thuộc tính của điều khiển Ribbon:	11
3.3. Tính năng callback của điều khiển Ribbon:	13
3.4. Hình ảnh của điều khiển Ribbon:	15
4. Sử dụng XML Notepad để xây dựng Ribbon	17
4.1. Hỗ trợ Clipboard:	22
4.2. Hỗ trợ kéo/thả:	22
4.3. Giảm đồ hiệu lực XML:	23
4.4. Tạo các phần tử và thuộc tính của chúng:	23
4.5. Xây dựng hình ảnh của điều khiển Ribbon:	25
5. Xây dựng các điều khiển Ribbon	26
5.1. Xây dựng điều khiển button, checkBox, editBox:	26
5.2. Xây dựng điều khiển label, menu, combobox, gallery:	28
5.3. Xây dựng điều khiển dropDown:	32
Tài liệu tham khảo.....	36

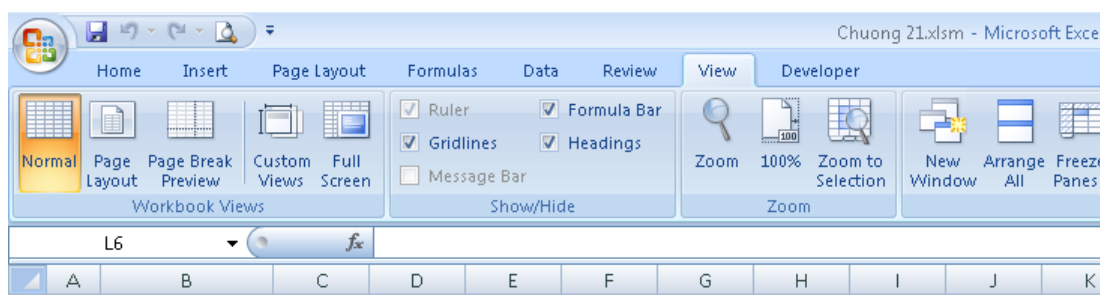
Đây là một nội dung trong sách **Lập trình VBA trong Excel phiên bản 2012** do Nhà xuất bản thống kê xuất bản. Tôi gửi nội dung này để bổ sung cho bạn đọc đã có sách ở phiên bản trước không có nội dung này.

Trong phần này, chúng ta cùng khám phá, tìm hiểu về Ribbon. Cho dù am hiểu về VBA nhưng làm việc với Ribbon không hề đơn giản vì môi trường khác hoàn toàn. Ribbon được xây dựng bằng ngôn ngữ XML (eXtensible Markup Language - ngôn ngữ đánh dấu mở rộng). Tuy nhiên, chúng ta không cần tìm hiểu nhiều về ngôn ngữ XML vì đã có một số công cụ hỗ trợ cho riêng điều khiển Ribbon. Chỉ cần nắm một số kiến thức XML liên quan đến một số thao tác với Ribbon là đủ. Trong nội dung này, chỉ cần copy các ví dụ mã XML có sẵn và chỉnh sửa theo ý muốn (có trong các tập tin ví dụ kèm theo).

Còn một vấn đề chúng ta quan tâm là thực đơn (menu) tự tạo sẽ như thế nào trong Excel 2007? Khi đó, toàn bộ thực đơn tự tạo được quản lý trong tab Add-Ins, group “Menu Commands” (hình 21-30) và chúng hoạt động bình thường.



Hình 21-30: Menu tự tạo được quản lý trong tab Add-Ins khi mở bằng Excel 2007

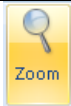
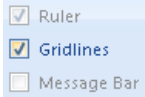


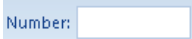

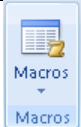
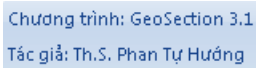

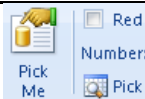
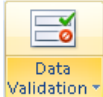
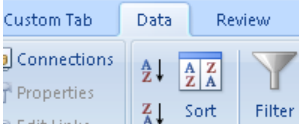


Hình 21-31: Ribbon thay thế menu từ Office 2007 trở đi

1. Mô hình quản lý với Ribbon

Theo mặc định, Excel 2007 gồm các tab Home, Insert, Page Layout, Formulas, Data,... (hình 21-31). Tab quản lý các group, ví dụ tab View quản lý group Workbook Views, Show/Hide, Zoom, Window, Macro,... Mỗi group quản lý các đối tượng Ribbon khác nhau. Các dạng điều khiển Ribbon chính được liệt kê trong bảng 21-6, cột thứ ba là ví dụ kiểu đối tượng tương ứng trong Excel 2007. Để tìm hiểu chi tiết các điều khiển Ribbon, chúng ta xem trong tập tin ExcelRibbonControls.xlsx như hình 21-32 (nguồn từ www.microsoft.com).

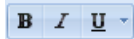
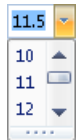
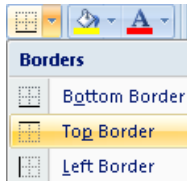
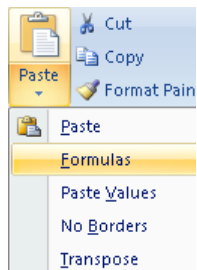
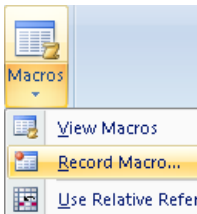
Bảng 21-6: Các điều khiển Ribbon trong Excel 2007

Tên điều khiển	Tên lớp	Ví dụ trong Excel 2007
Button	RibbonButton	 View/Zoom/Zoom
CheckBox	RibbonCheckBox	 View/“Show/Hide”/Gridline
ComboBox	RibbonComboBox	 Home/Font/Font
DropDown	RibbonDropDown	 Home/Font/Border
EditBox	RibbonEditBox	 Number: <input type="text"/>
Gallery	RibbonGallery	 Insert/Charts/Column
Group	RibbonGroup	 View/Macros
Label	RibbonLabel	
Menu	RibbonMenu	 Home/Alignment/Orientation
Separator	RibbonSeparator	 Gạch ngăn trong group
SplitButton	RibbonSplitButton	 Data/Data Tools/Data Validation
Tab	RibbonTab	 Data

ToggleButton	RibbonToggleButton		View/Workbook Views/Normal
--------------	--------------------	---	----------------------------

Có một số điều khiển Ribbon có chứa các phần tử bên trong (item) giúp lựa chọn chi tiết hơn, chúng ta có thể tùy biến các phần tử đó. Danh sách điều khiển Ribbon chứa các phần tử tại bảng 21-7.

Bảng 21-7: Các điều khiển chứa mục con trong Excel 2007

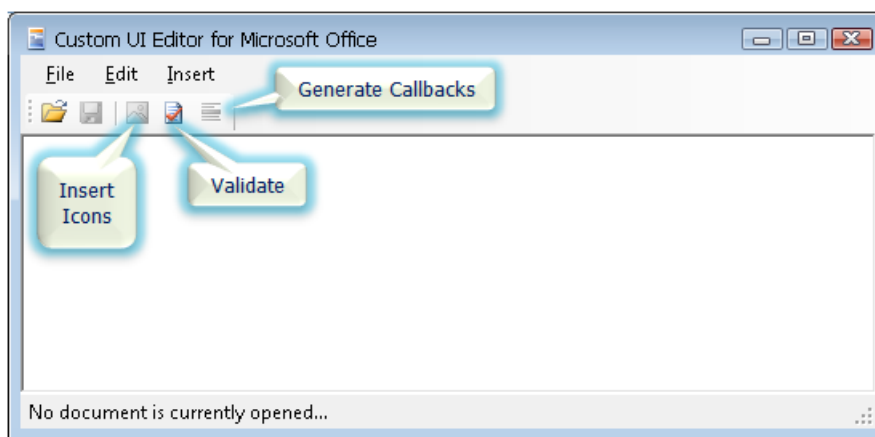
Tên điều khiển	Mô tả	Hình ví dụ
<code><buttonGroup...></code> contents <code></buttonGroup></code>	Chúng chứa các điều khiển khác nhau như <code><button></code> , <code><toggleButton></code> , <code><gallery></code> , <code><menu></code> , <code><dynamicMenu></code> , <code><splitButton></code> . Chúng quản lý một nhóm điều khiển có liên quan với đường bao xung quanh, giữa các điều khiển có sự phân cách.	
<code><comboBox...></code> contents <code></comboBox></code>	Chúng chỉ chứa được các <code><item></code> . Có thể chọn giá trị bằng cách gõ trực tiếp (có thể tùy biến, không có trong danh sách) hay chọn từ danh sách thả xuống.	
<code><dropDown...></code> contents <code></dropDown></code>	Chúng có thể chứa các <code><item></code> hoặc <code><button></code> . Chúng được cung cấp một danh sách mục con thả xuống để lựa chọn. Chúng ta có thể thiết lập được chiều rộng của danh sách mục con để thể hiện hết nội dung mục con.	
<code><menu...></code> contents <code></menu></code>	Chúng chứa các điều khiển khác nhau như <code><button></code> , <code><toggleButton></code> , <code><checkBox></code> , <code><gallery></code> , <code><splitButton></code> , <code><Separator></code> , <code><dynamicMenu></code> , <code><menu></code> . Một menu popup, đã được xác định trong tập tin RibbonX. Menu có thể chứa các nút hoặc trình đơn khác, cho phép chúng ta tạo ra các cấu trúc trình đơn phân cấp.	
<code><splitButton...></code> <code><button.../></code> <code><menu...></code> menu contents <code></menu></code> <code></splitButton></code>	Chúng có thể chứa <code><button></code> hoặc <code><toggleButton></code> trong danh sách mục con. Nhấp vào mũi tên thả xuống cho thấy một danh sách các lựa chọn liên quan. Danh sách này có thể chứa các mục con khác nhỏ hơn. Ví dụ: <code><splitButton></code> Format có menu con "Hide/Unhide" chứa các menu con chi tiết để lựa chọn.	

Mỗi điều khiển đều có thuộc tính và sự kiện riêng, chúng ta đề cập chi tiết hơn ở mục sau. Khi nắm được nguyên tắc quản lý trên, chúng ta có thể xây dựng các điều

khởi Ribbon phục vụ mục đích riêng. Có nhiều cách xây dựng điều khiển Ribbon nhưng trước hết, hãy bắt đầu từ cách đơn giản nhất! Cần chú ý khi khai báo tên đối tượng Ribbon cho đúng vì XML phân biệt chữ hoa với chữ thường. Ví dụ: “TabHome” là khai báo đúng, còn “tabHome” là khai báo sai.

	Control Name	Control Type	Tab Set Name	Tab Name	Group Name
4	FileOpen	button	None (Quick Access Toolbar)	Quick Access Toolbar	
5	FileSave	button	None (Quick Access Toolbar)	Quick Access Toolbar	
6	FileSendAsAttachment	button	None (Quick Access Toolbar)	Quick Access Toolbar	
7	FilePrintQuick	button	None (Quick Access Toolbar)	Quick Access Toolbar	
8	FilePrintPreview	button	None (Quick Access Toolbar)	Quick Access Toolbar	
9	Spelling	button	None (Quick Access Toolbar)	Quick Access Toolbar	
10	Undo	gallery	None (Quick Access Toolbar)	Quick Access Toolbar	
11	Redo	gallery	None (Quick Access Toolbar)	Quick Access Toolbar	
12	SortAscendingExcel	button	None (Quick Access Toolbar)	Quick Access Toolbar	
13	SortDescendingExcel	button	None (Quick Access Toolbar)	Quick Access Toolbar	
14	FileNew	button	None (Office Menu)	Office Menu	
15	FileOpen	button	None (Office Menu)	Office Menu	
16	UpgradeWorkbook	button	None (Office Menu)	Office Menu	
17	FileSave	button	None (Office Menu)	Office Menu	
18	FileSaveAsMenu	splitButton	None (Office Menu)	Office Menu	
19	FileSaveAs	button	None (Office Menu)	Office Menu	FileSaveAsMenu
20	FileSaveAsExcelXlsx	button	None (Office Menu)	Office Menu	FileSaveAsMenu
21	FileSaveAsExcelXlsxMacro	button	None (Office Menu)	Office Menu	FileSaveAsMenu
22	FileSaveAsExcelXlsb	button	None (Office Menu)	Office Menu	FileSaveAsMenu
23	FileSaveAsExcel97_2003	button	None (Office Menu)	Office Menu	FileSaveAsMenu
24	AdvertisePublishAs	button	None (Office Menu)	Office Menu	FileSaveAsMenu
25	FileSaveAsPdfOrXps	button	None (Office Menu)	Office Menu	FileSaveAsMenu
26	FileSaveAsOtherFormats	button	None (Office Menu)	Office Menu	FileSaveAsMenu
27	FilePrintMenu	splitButton	None (Office Menu)	Office Menu	
28	FilePrint	button	None (Office Menu)	Office Menu	FilePrintMenu

Hình 21-32: Danh sách các điều khiển Ribbon có trong Excel 2007



Hình 21-33: Công cụ hỗ trợ cho xây dựng Ribbon trong Microsoft Office

Để tiếp cận nhanh nhất với Ribbon, hãy sử dụng công cụ “Custom UI Editor for Microsoft Office” (gọi tắt là CUE). Công cụ này hỗ trợ cho cả Office 2007 và 2010, được tải từ trang web <http://openxmldeveloper.org>. Khi đã cài đặt và chạy, giao diện công cụ như hình 21-33. Chúng ta chú ý các chức năng sau:

- Insert Icons: Chèn biểu tượng, hình ảnh cho điều khiển (từ bên ngoài).

- Validate: Kiểm tra, xác định tính hợp lệ của mã XML.
- Generate Callbacks: Tạo thủ tục VBA thực thi cho điều khiển, chức năng này khai báo chi tiết tham số của đối tượng Ribbon làm việc tới. Chỉ cần copy đoạn mã này và dán vào trong module VBA và bổ sung mã.

2. Xây dựng một tab mới cùng với group

Chúng ta xây dựng một tab mới có tên “Custom Tab” đứng sau tab Formulas. Tab mới chứa group Font (lấy từ tab Home), Macros (lấy từ tab View) và group mới “New Group”. Group “New Group” chứa button “Hello the Ribbon”. Trình tự thực hiện theo các bước sau:

- Bước 1: Tạo mới tập tin có tên CustomTab.xlsm trong thư mục nào đó. Nếu xây dựng trong Word thì chúng ta tạo tập tin *.docm.
- Bước 2: Chạy ứng dụng CUE, sau đó mở tập tin CustomTab.xlsm. File được chọn cùng biểu tượng (Excel) hiện ra ngay bên dưới ở cột bên trái, tên tập tin cùng đường dẫn hiện phía dưới cửa sổ (hình 21-34).
- Bước 3: Vào menu Insert, nếu sử dụng Office 2007 thì chọn “Office 2007 Custom UI Part”, mục customUI.xml hiện ra ngay dưới File được chọn (hình 21-34). Sau đó copy nội dung mã XML sau vào cột bên phải (hoặc có thể gõ trực tiếp):

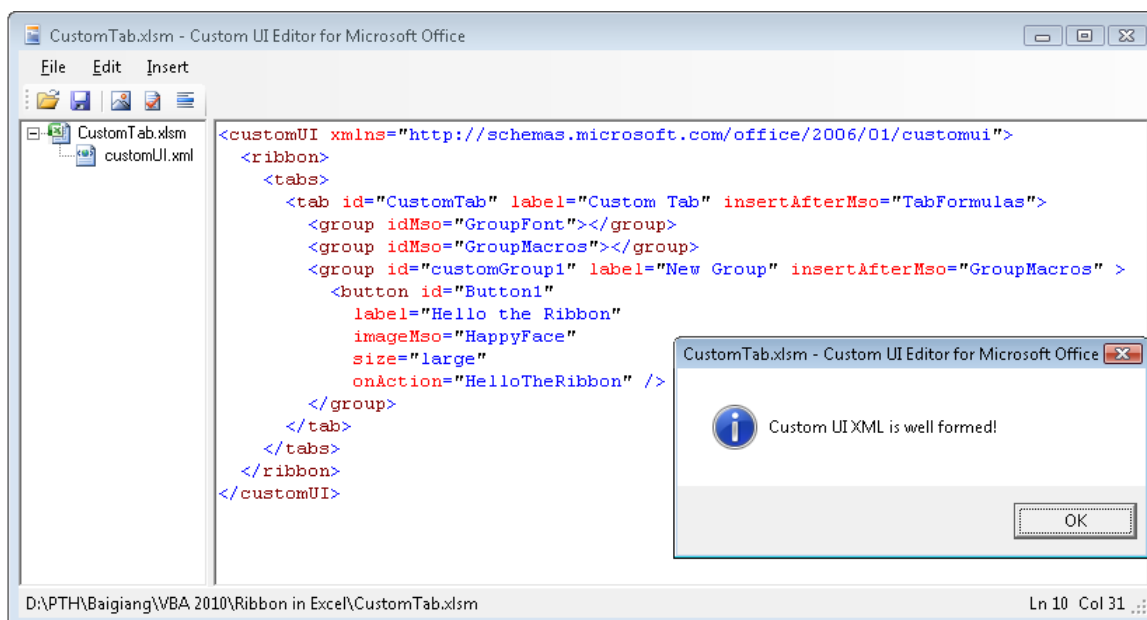
```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="CustomTab" label="Custom Tab" insertAfterMso="TabFormulas">
        <group idMso="GroupFont"></group>
        <group idMso="GroupMacros"></group>
        <group id="customGroup1" label="New Group" insertAfterMso="GroupMacros" >
          <button id="Button1"
            label="Hello the Ribbon"
            imageMso="HappyFace"
            size="large"
            onAction="HelloTheRibbon" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Nếu sử dụng Office 2010, chúng ta chọn “Office 2010 Custom UI Part” và thay thế dòng mã đầu tiên bằng:

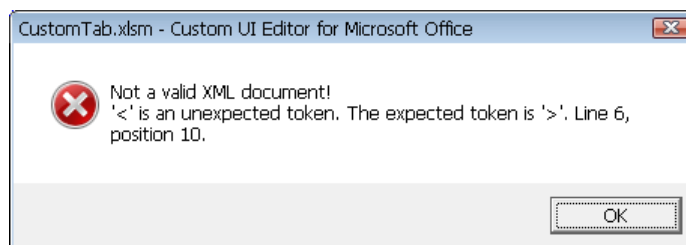
```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
```

Trong đoạn mã trên, nếu bỏ qua mục insertAfterMso, “Custom Tab” mặc định nằm ở vị trí cuối cùng.

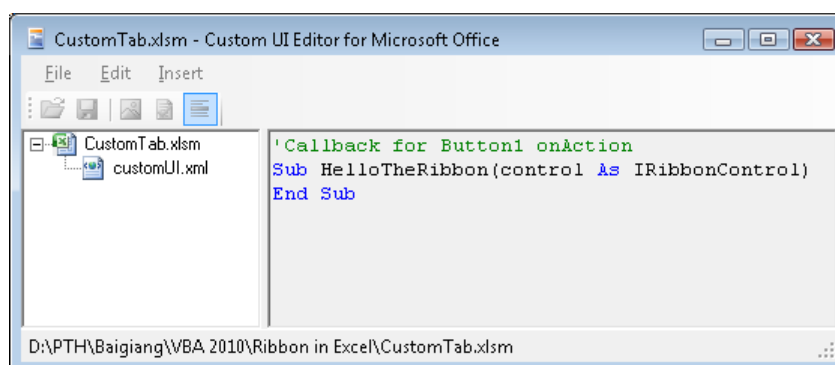
- Bước 4: Kiểm tra mã bằng cách bấm nút Validate (hình 21-34). Trong trường hợp mã XML bị lỗi, thông báo lỗi xuất hiện cùng với địa chỉ hàng, vị trí của lỗi để giúp chúng ta sửa lỗi (hình 21-35). Nếu mã XML làm việc tốt, sẽ xuất hiện thông báo “Custom UI XML is well formed!”. Khi đó hãy lưu lại.



Hình 21-34: Mã XML của CustomTab.xlsm đã được xác định hợp lệ



Hình 21-35: Thông báo lỗi khi mã XML sai với vị trí cụ thể



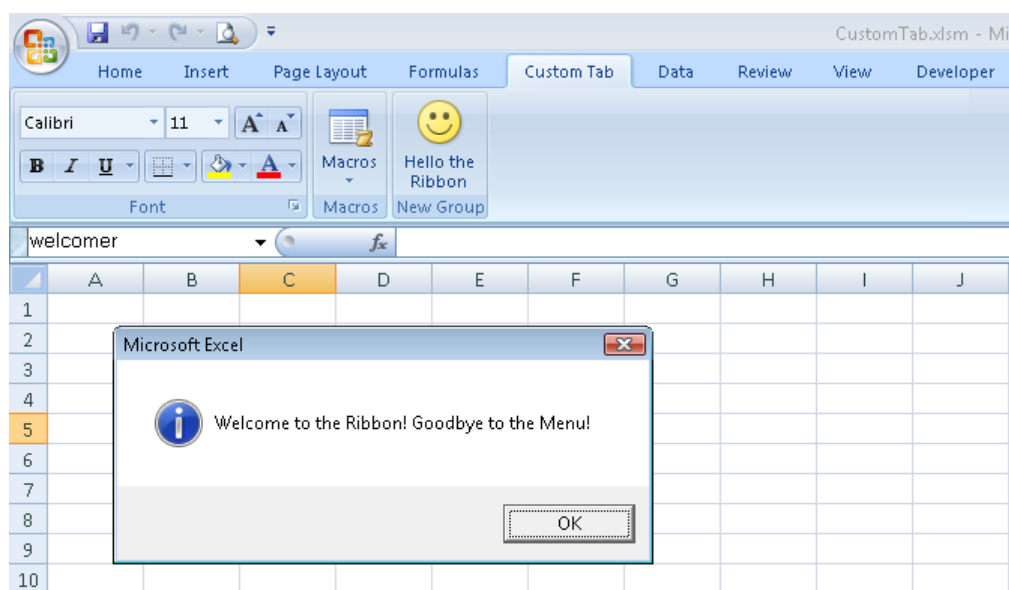
Hình 21-36: Nội dung thủ tục ban đầu trong VBA được tạo ra từ Generate Callbacks

- Bước 5: Mở CustomTab.xlsm trong Excel, xây dựng thủ tục HelloTheRibbon cho điều khiển “Hello the Ribbon”. Trước hết trong cửa sổ CUE, bấm Generate Callbacks, chúng ta nhận được thủ tục ban đầu được tạo ra (hình 21-36) và dán vào module VBA, thay đổi nội dung như ở dưới.

```
'Đối tượng control là button “Hello the Ribbon”
Sub HelloTheRibbon(control As IRibbonControl)
    MsgBox "Welcome to the Ribbon! Goodbye to the Menu!", vbInformation
End Sub
```

Bây giờ hãy bấm điều khiển “Hello the Ribbon” xem kết quả thế nào? Chúng ta sẽ thấy kết quả thực hiện như hình 21-37.

Đây mới là bước ban đầu tạo tab cùng với các đối tượng bên trong, chúng ta cần tìm hiểu thêm các dạng điều khiển khác nhau cùng với thuộc tính và thủ tục thi hành của chúng.



Hình 21-37: Tab "Custom Tab" được tạo mới và các điều khiển bên trong

Như vậy, chúng ta đã tạo được điều khiển Ribbon đầu tiên và bây giờ hãy khám phá cách xây dựng Ribbon bằng ngôn ngữ XML như thế nào?!

3. Cấu trúc mã XML và tìm hiểu đối tượng Ribbon

3.1. Cấu trúc mã XML:

Đầu tiên, chúng ta trở lại các dòng lệnh ở hình 21-34. XML làm việc thông qua từng dòng và thể hiện kết quả ở Ribbon. Chúng ta thấy sự phân cấp quản lý rõ rệt từ lớn đến bé: <customUI...> → <ribbon> → <tabs> → <tab> → <group> → <control>.

- Đối tượng <customUI...>: được hiểu là giao diện người dùng (custom User Interface), là cơ sở của XML. <customUI...> được xác định như tư liệu của RibbonX. Đối tượng được kết thúc bằng </customUI> ở cuối chương trình.
- Đối tượng <ribbon>: quản lý tất cả những thay đổi liên quan đến giao diện Ribbon. Ribbon quản lý thanh tiêu đề, Office Button (<officeMenu>), Quick Access Toolbar (<qat>) và các Tabs. Trong đó, đối tượng <tabs> là quan trọng nhất mà chúng ta đang tìm hiểu. Đối tượng được kết thúc bằng </ribbon> ở phía cuối chương trình.

- Đối tượng <tabs>: quản lý tất cả các <tab> hiện tại hoặc tạo mới. Đối tượng được kết thúc bằng </tabs> ở phía cuối chương trình.

- Đối tượng <tab>: quản lý tất cả những thay đổi liên quan đến các <group> hiện tại hoặc tạo mới. Đối tượng được kết thúc bằng </tab> ở phía cuối chương trình.

Dòng `<tab id="CustomTab" label="Custom Tab" insertAfterMso="TabFormulas">` tạo ra tab tùy chỉnh. Trong CUE, chữ màu đỏ là thuộc tính, màu xanh là giá trị (nằm trong ngoặc kép) của đối tượng làm việc. Một đối tượng Ribbon (như tab, group,...) phải có thuộc tính chỉ danh (id), tên hiển thị (label) và vị trí hiển thị (insertAfterMso, insertBeforeMso,...).

Có ba thuộc tính id tùy theo mục đích sử dụng là id, idMso và idQ. Trong trường hợp này, chúng ta tạo ra một <tab> tùy chỉnh, do vậy sử dụng thuộc tính id và tạo cho nó một cái tên theo ý muốn. Giá trị id không được trùng lặp với id của các điều khiển khác. Nếu giá trị id trùng nội dung với điều khiển khác, Ribbon sẽ không hoạt động (nhưng không báo lỗi khi sử dụng chức năng Validate).

Tiếp theo là nhãn label, cách thể hiện này không hiển thị được tiếng Việt vì CUE không hỗ trợ Unicode. Chúng ta tìm hiểu phương pháp hiển thị nhãn bằng tiếng Việt trong mục tiếp theo.

Cuối cùng là vị trí của tab mới, đứng sau (insertAfterMso) hoặc trước (insertBeforeMso) tab hiện hành nào đó, là tab Formulas trong trường hợp này.

- Đối tượng <group>: quản lý các điều khiển Ribbon hiện tại hoặc tạo mới. Đối tượng được kết thúc bằng </group>. Bên trong group là nơi xây dựng các điều khiển Ribbon khác nhau, tùy thuộc yêu cầu công việc.

+ Dòng `<group idMso="GroupFont"></group>` sẽ tạo group Font có sẵn trong Microsoft. Thuộc tính idMso cho phép tải các điều khiển được xây dựng sẵn trong Microsoft Office. Ví dụ: tab Home có chứa group idMso như Clipboard, Font, Alignment, Number, Styles, Cells, Editing.

+ Dòng `<group idMso="GroupMacros"></group>` sẽ tạo group Macros có sẵn trong Microsoft.

+ Dòng `<group id="customGroup1" label="New Group" insertAfterMso="GroupMacros">` tạo một group mới. Đây là nơi quản lý và

xây dựng các điều khiển Ribbon (như button “Hello the Ribbon”). Khi xây dựng xong, cần đóng group bằng </group>.

- Đối tượng <button>: là điều khiển dạng nút (chúng ta có thể chọn điều khiển khác), thuộc tính của nó gồm có id, label, imageMso, size, onAction. Chúng ta đã tìm hiểu thuộc tính id và label. Thuộc tính imageMso hiển thị biểu tượng của điều khiển được xây dựng sẵn trong Excel (hình 21-38). Thuộc tính size điều khiển kích cỡ điều khiển, gồm có normal (bình thường) và large (lớn). Thuộc tính onAction gán tên thủ tục trong VBA để thi hành lệnh khi bấm vào điều khiển đó. Ngoài ra, với điều khiển dạng khác thì thuộc tính cũng khác. Chúng ta tìm hiểu chi tiết hơn ở mục tiếp theo.

Như vậy, chúng ta đã tìm hiểu sơ bộ về cách tạo Ribbon và trình tự xây dựng chúng trong CUE bằng ngôn ngữ XML.

3.2. Các thuộc tính của điều khiển Ribbon:

Mỗi điều khiển Ribbon đều có những thuộc tính nhất định. Những thuộc tính này điều khiển sự xuất hiện, diện mạo cũng như sự làm việc của chúng. Bảng 21-8 liệt kê danh sách thuộc tính cho toàn bộ điều khiển Ribbon.

Bảng 21-8: Các thuộc tính của điều khiển Ribbon

Thuộc tính	Mô tả	Giá trị	Điều khiển áp dụng
boxStyle	Sắp xếp các biểu tượng theo chiều ngang (mặc định) hoặc dọc	horizontal, vertical	Box
columns	Số cột xuất hiện khi trưng bày	1 ÷ 1024 ký tự	Gallery
Enable	Bật hoặc tắt điều khiển	True, False	Tất cả
Id	Tên riêng của điều khiển	1 ÷ 1024 ký tự	Tất cả
IdMso	Tên điều khiển có sẵn trong Microsoft Office	1 ÷ 1024 ký tự	Tất cả
Image	Biểu tượng của điều khiển, được tải từ bên ngoài	1 ÷ 1024 ký tự	Được hỗ trợ biểu tượng
ImageMso	Biểu tượng của điều khiển, có sẵn trong Microsoft Office	1 ÷ 1024 ký tự	Được hỗ trợ biểu tượng
itemHeight	Chiều cao phần tử của Gallery,	1 ÷ 4096	Gallery

	tính bằng pixel		
itemSize	Kích cỡ các phần tử trong thực đơn	normal, large	menu
itemWidth	Chiều rộng phần tử của Gallery, tính bằng pixel	1 ÷ 4096	Gallery
keytip	Phím tắt kết hợp được sử dụng để truy cập vào đối tượng	1 ÷ 3 ký tự	Tất cả điều khiển, group, tab
label	Nhãn hiển thị của điều khiển xuất hiện trong Ribbon	1 ÷ 1024 ký tự	Tất cả điều khiển, group, tab
maxLength	Chiều dài lớn nhất của ký tự nhập vào	1 ÷ 1024 ký tự	editBox, comboBox
rows	Số hàng thả xuống của Gallery	1 ÷ 1024	Gallery
screentip	Chỉ dẫn nhỏ xuất hiện khi rê chuột qua điều khiển	1 ÷ 1024 ký tự	Tất cả
showImage	Bật hay tắt hiển thị biểu tượng của điều khiển	True, False	Được hỗ trợ biểu tượng
showItemImage	Bật hay tắt biểu tượng phần tử khi được thả xuống	True, False	comboBox, dropDown, gallery
showItemLabel	Bật hay tắt nhãn hiển thị phần tử khi được thả xuống	True, False	comboBox, dropDown, gallery
showLabel	Bật hay tắt nhãn hiển thị của điều khiển	True, False	Tất cả
size	Kích cỡ điều khiển. Normal chiếm 1 dòng, Large chiếm 3 dòng	normal, large	Tất cả
sizeString	Chuỗi đại diện lớn nhất được sử dụng để thiết lập chiều rộng của điều khiển	1 ÷ 1024 ký tự	editBox, comboBox, dropDown
supertip	Chỉ dẫn lớn xuất hiện khi rê chuột qua điều khiển	1 ÷ 1024 ký tự	Tất cả
tag	Gắn thẻ cho điều khiển, là chuỗi tùy biến	1 ÷ 1024 ký tự	Tất cả
title	Tiêu đề của menu	1 ÷ 1024 ký tự	menu, menuSeparator

3.3. Tính năng callback của điều khiển Ribbon:

Microsoft cho phép tạo sự móc nối giữa XML với thủ tục hoặc hàm VBA. Công việc này được thực hiện bằng cách sử dụng cơ chế có tên là callback (gọi lại). Chúng ta khai báo tên của một thủ tục trong XML theo quy định với từ khoá get..., sau đó xây dựng thủ tục đó trong VBA (phải trùng tên với khai báo trong XML). Khi điều khiển Ribbon được bấm hay bị thay đổi,... thì thủ tục đó sẽ chạy và có thể can thiệp ngược lại XML. Công việc đó giống như sử dụng Application.OnKey, hay phương thức OnAction của đối tượng CommandBarButton,... Callbacks cũng được sử dụng khi cần thay đổi thuộc tính của một điều khiển khi chạy. Đúng hơn là xác định một giá trị cụ thể cho thuộc tính trong XML, cung cấp tên của nó cho VBA và có thể xác định được giá trị của thuộc tính khi cần.

Ví dụ như với CustomTab.xlsm, chúng ta khai báo thủ tục thi hành “HelloTheRibbon” trong XML. Trong module VBA của CustomTab.xlsm, chúng ta xây dựng thủ tục “HelloTheRibbon” và khi bấm “Hello the Ribbon”, thủ tục được thi hành (hình 21-37). Thay vì xuất hiện tên label="Custom Tab" của tab ở CustomTab.xlsm (hình 21-38), chúng ta có thể thay tên nhãn từ VBA nhờ getLabel (trước khi thực hiện hãy lưu sang tên mới CustomTab1.xlsm):

```
<tab id="CustomTab" getLabel="CustomLabelRun" insertAfterMso="TabFormulas">
```

Thủ tục CustomTabRun trong VBA như sau:

```
'Thủ tục này sẽ callbacks lại XML
Sub CustomTabRun(control As IRibbonControl, ByRef returnedVal)
    returnedVal = "Lựa chọn riêng"
End Sub
```

Khi hiển thị tab CustomTab lần đầu, Excel sẽ gọi thủ tục CustomTabRun trong module VBA. Khi đó, tab có tên “Lựa chọn riêng”. Sử dụng getLabel cũng giúp chúng ta Việt hoá thuộc tính Label.

Đối tượng control hết sức đơn giản, chỉ gồm 3 thuộc tính:

- Id: Thuộc tính chỉ tên đối tượng, sử dụng để phân biệt các điều khiển.

- Tag: Thuộc tính gắn thẻ của đối tượng (tag) khi sử dụng trong XML.
- Context: Ngữ cảnh, không sử dụng trong Excel.

Bảng 21-9 liệt kê các callback chỉ có giá trị vào thời điểm chạy.

Bảng 21-9: Danh sách Callbacks

Callback	Đối tượng sử dụng	Mô tả
getContent	dynamicMenu	Cung cấp XML cho toàn bộ menu.
getPressed	toggleButton, checkBox	Chỉ định điều khiển được bấm (đánh dấu) hay không?
getItemCount	comboBox, dropdown, gallery	Chỉ định bao nhiêu mục con có trong danh sách vào thời gian chạy.
getItemID, getItemLabel, getItemImage, getItemScreentip, getItemSupertip	comboBox, dropdown, gallery	Gọi mỗi mục con để cung cấp các thuộc tính cho chúng.
getSelectedItemID	dropdown, gallery	Xác định mục nào được chọn bằng cách cung cấp ID của chúng.
getSelectedItemIndex	dropdown, gallery	Xác định mục nào được chọn bằng cách cung cấp chỉ mục của chúng trong danh sách.
getText	comboBox, editBox	Cung cấp chuỗi văn bản được hiển thị trong điều khiển

Bảng 21-10 liệt kê các hàm sử dụng cho tất cả các điều khiển. Khi sử dụng CUE, nhờ chức năng General Callbacks, chúng ta chỉ cần sao chép vào dán vào các dự án VBA của mình.

Bảng 21-10: Danh sách Callbacks

Callback	Đối tượng VBA Callback
getContent, getDescription, getEnabled, getImage, getItemCount, getTitle, getItemHeight, getItemWidth, getKeytip, getLabel, getPressed, getSize, getScreentip, getSelectedItemID, getVisible, getSelectedItemIndex, getShowImage,	Sub CallbackName(ByRef Control As IRibbonControl, ByRef ReturnValue As Variant)

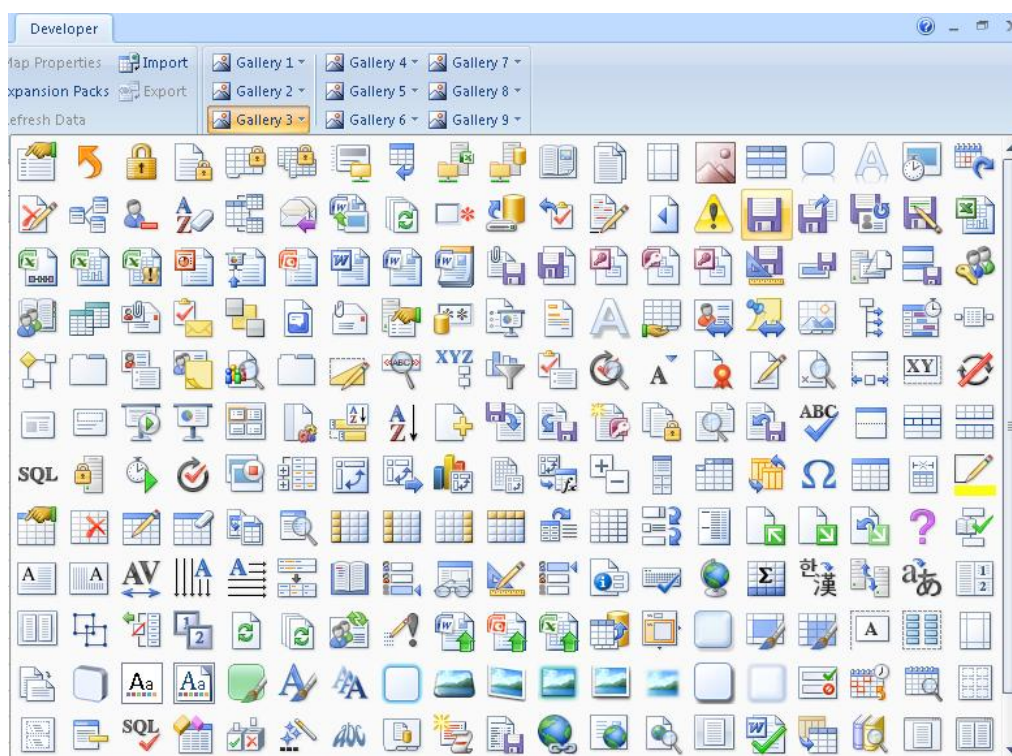
getShowLabel, getSupertip, getText	
getItemID, getItemImage, getItemLabel, getItemScreentip, getItemSupertip	Sub CallbackName(ByRef Control As IRibbonControl, ByRef ReturnValue As Variant)
onAction cho điều khiển button	Sub CallbackName(ByRef Control As IRibbonControl)
onAction cho điều khiển checkBox, toggleButton	Sub CallbackName(ByRef Control As IRibbonControl, ByRef Pressed As Boolean)
onAction cho điều khiển dropDown, gallery	Sub CallbackName(ByRef Control As IRibbonControl, ByRef SelectedID As String, ByRef SelectedIndex As Integer)
onChange cho điều khiển editBox, hoặc comboBox	Sub CallbackName(ByRef Control As IRibbonControl, ByRef Text As String)

Trong bảng 21-10, nội dung một số đối tượng khai báo như sau:

- Control: Tên điều khiển được khai báo trong XML.
- ReturnValue: Giá trị trả về của Control. ReturnValue là String với getContent, getDescription, getLabel, getText, getTitle, getKeytip,... ReturnValue là True (False) với getEnabled, getVisible, getPressed, getSelectedItemID, getSelectedItemIndex,... ReturnValue là Double với getItemCount, getItemHeight, getItemWidth,...
- Pressed: Trả về True (False) nếu bấm phím (không bấm) vào điều khiển.
- SelectedID (SelectedIndex): Trả về giá trị id (chỉ mục) của phần tử trong dropDown và gallery.
- Text: Trả về chuỗi dữ liệu nhập.

3.4. Hình ảnh của điều khiển Ribbon:

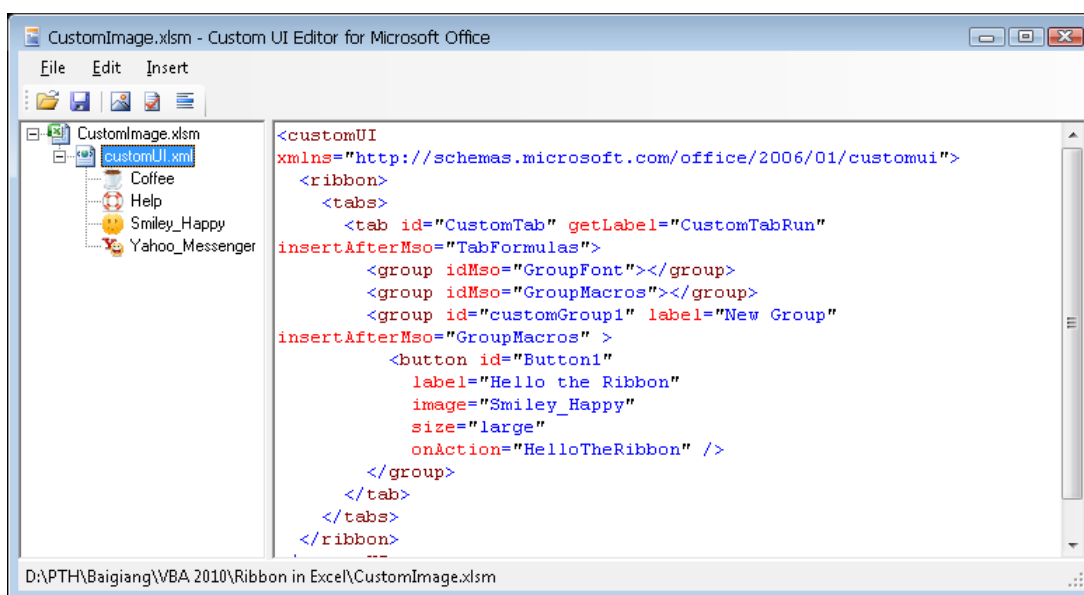
Hầu hết các điều khiển có thể gắn hình ảnh liên quan. Sự lựa chọn của hình ảnh và phong cách hiển thị được điều khiển bởi các imageMso, Image, getImage, showImage, getShowImage, showItemImage, getShowItemImage, kích thước, và các thuộc tính getSize.



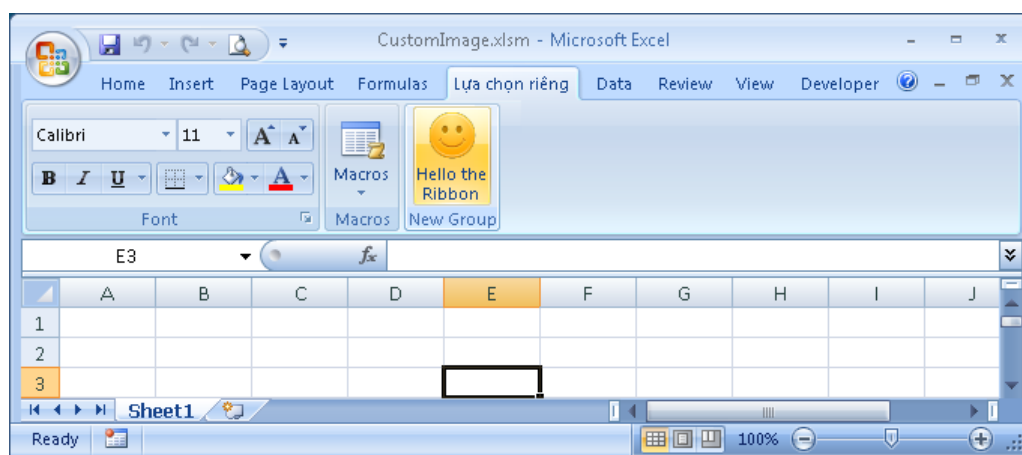
Hình 21-38: Danh sách hình ảnh được xây dựng trong imageMso

Thuộc tính `imageMso` sử dụng khi khai thác các hình ảnh được xây dựng sẵn trong Microsoft Office. Giá trị của thuộc tính phải là tên được định nghĩa sẵn, có thể được tìm thấy bằng cách tải về tập tin `Office2007IconsGallery.xlsm` từ trang web MSDN. Tập tin `Office2007IconsGallery.xlsm` trưng bày 2.586 hình ảnh trong 9 mục, từ đó giúp chúng ta xây dựng điều khiển Ribbon (hình 21-38). Ví dụ: Để hiển thị hình khuôn mặt cười, sử dụng `imageMso="HappyFace"`.

Ngoài ra, chúng ta còn có thể sử dụng thuộc tính `image` khi sử dụng hình ảnh bên ngoài. Trước khi thực hiện công việc này, cần tải hình ảnh đó bằng cách bấm “Insert Icons” (hình 21-33). Sau đó khai báo `image` là tên hình ảnh (không cần đuôi). Hình ảnh hợp lệ có đuôi *.ico, bmp, png, jpg, tif. Kích thước chuẩn của hình ảnh là 16x16 hoặc 32x32 với 96dpi. Những yếu tố này rất quan trọng để hiển thị hình ảnh rõ ràng và sắc nét trong giao diện người dùng. Những hình ảnh đó được chứa bên trong tập tin bảng tính (tìm hiểu thêm ở mục 21.3.4). Ví dụ: Xây dựng tập tin `CustomImage.xlsm` (trên cơ sở tập tin `CustomTab1.xlsm`) và tải các hình ảnh bên ngoài vào (hình 21-39). Khi tải xong các hình ảnh, bấm `customUI.xml` sẽ hiện danh sách các hình ảnh. Sau đó, chúng ta chỉ cần dán tên hình ảnh đó vào thuộc tính `image`. Kết quả mở `CustomImage.xlsm` như hình 21-40.



Hình 21-39: Các hình ảnh bên ngoài được tải vào trong CustomImage.xlsm

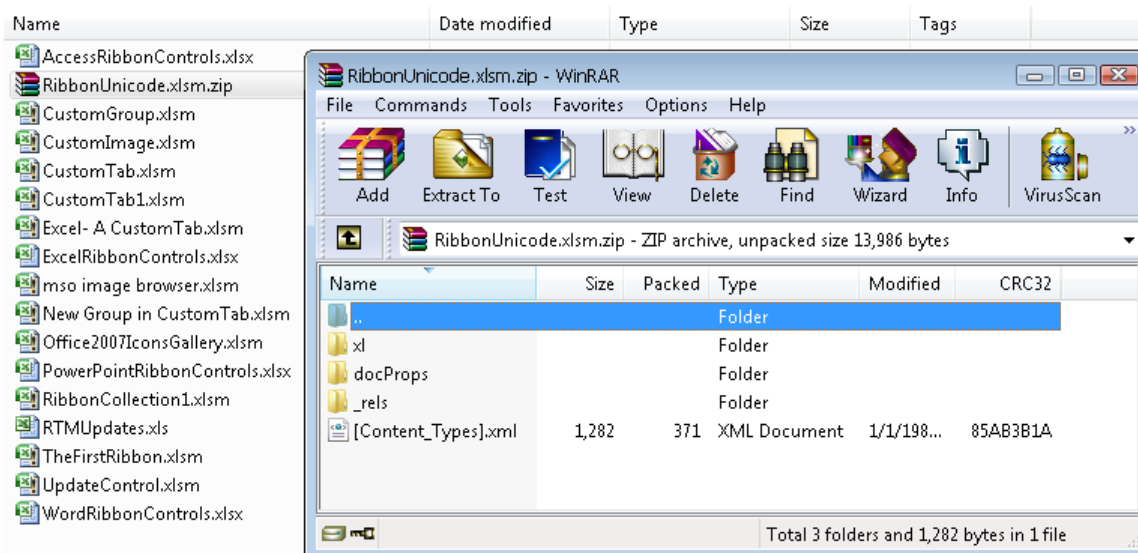


Hình 21-40: CustomImage.xlsm hiển thị hình ảnh được xây dựng từ bên ngoài (image)

4. Sử dụng XML Notepad để xây dựng Ribbon

Chúng ta đã biết phương pháp xây dựng Ribbon nhờ tiện ích CUE. Nhìn chung, sử dụng CUE tiện lợi cho việc xây dựng Ribbon. Tuy nhiên, CUE có hạn chế là không hỗ trợ Unicode, tức là không thể Việt hoá Ribbon trong CUE. Để Việt hoá Ribbon, bắt buộc sử dụng callback `getLabel` kết hợp với VBA nên khá phức tạp. Do vậy, chúng ta cần sử dụng phương pháp khác để Việt hoá Ribbon. Đó là sử dụng Notepad (có sẵn trong Window) và “XML Notepad 2007” (gọi tắt XN2007). Chúng ta hãy tải XN2007 từ trang web của Microsoft và tiến hành cài đặt. Để sử dụng các ứng dụng này, chúng ta phải thiết lập một chút.

- Bước 1: Đầu tiên, tạo tập tin RibbonUnicode.xlsm. Sau đó, đổi đuôi thành RibbonUnicode.xlsm.zip. Sau đó bấm đúp chuột vào tập tin, chúng ta sẽ thấy cấu trúc tập tin của Excel 2007 (2010) như hình 21-41. Các hình ảnh Image tải bên ngoài cũng được lưu trong tập tin này. Các bước thực hiện tiếp theo.



Hình 21-41: Cấu trúc bên trong tập tin Excel 2007 (2010)

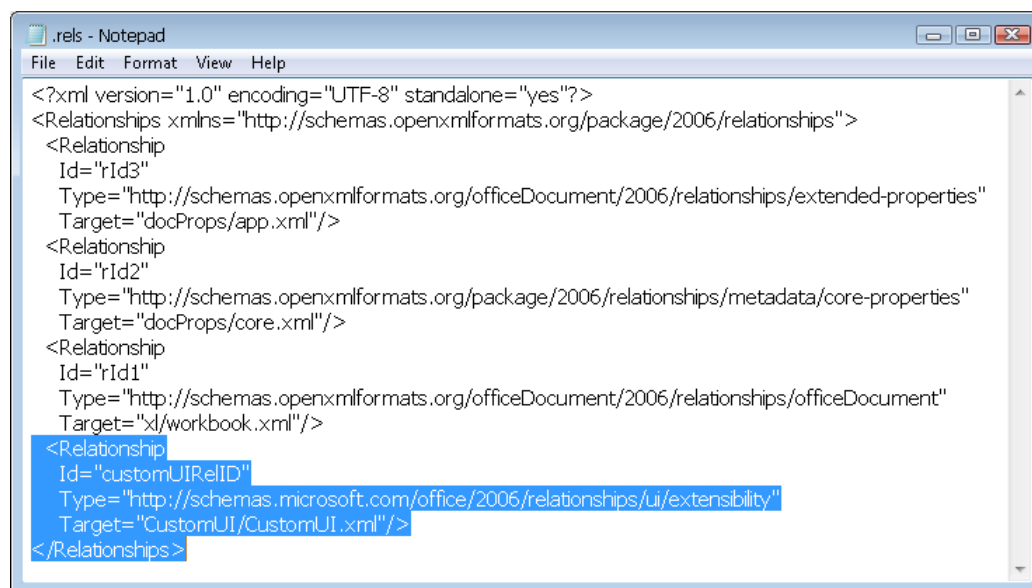


Hình 21-42: Cấu trúc tập tin .rels ban đầu trong thư mục _rels

- Bước 2: Di chuyển thư mục _rels ra Desktop. Tiếp theo, chúng ta tạo thư mục customUI trong Desktop, sau đó tạo tập tin CustomUI.xml (trong thư mục customUI) trong Notepad, ban đầu tập tin rỗng. Chúng ta tạo liên kết giữa tập tin .rels với CustomUI.xml bằng cách mở tập tin .rels trong Notepad (hình 21-42) và bổ sung đoạn mã XML phía dưới cùng như sau:

```
<Relationship
  Id="customUIRelID"
  Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility"
  Target="CustomUI/CustomUI.xml"/>
</Relationships>
```

Kết quả chỉnh sửa như hình 21-43, sau đó hãy đóng tập tin và sao chép thư mục _rels vào RibbonUnicode.xlsm.zip.



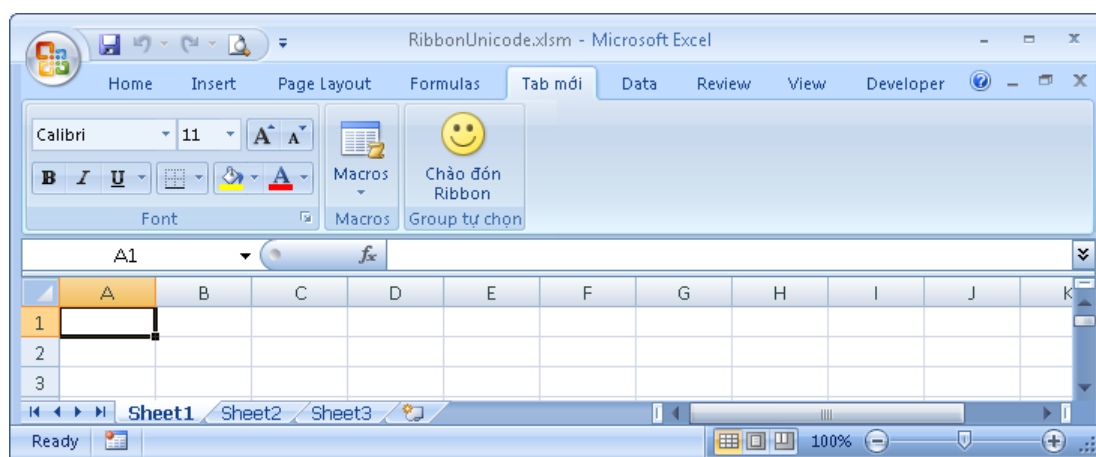
Hình 21-43: Tập tin .rels đã được bổ sung (phần bôi đen), chỉnh sửa

- Bước 3: Xây dựng mã XML trong CustomUI.xml (tạo điều khiển Ribbon) bằng cách dùng CUE mở tập tin CustomTab.xlsm. Sau đó, sao chép toàn bộ đoạn mã trên và dán vào CustomUI.xml. Gõ tên nhãn bằng tiếng Việt (Unicode), lưu lại và thoát khỏi Notepad. Sao chép thư mục CustomUI vào RibbonUnicode.xlsm.zip.

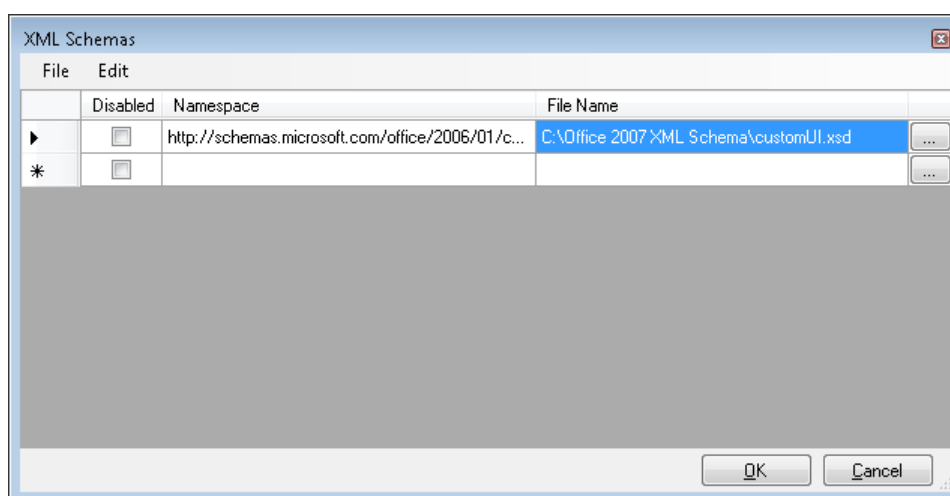


Hình 21-44: Mã XML đã được chỉnh sửa tiếng Việt

- Bước 4: Bỏ đuôi .zip của tập tin RibbonUnicode.xlsm.zip. Sau đó mở RibbonUnicode.xlsm, kết quả thể hiện ở hình 21-45.



Hình 21-45: Mã XML của đã được chỉnh sửa tiếng Việt

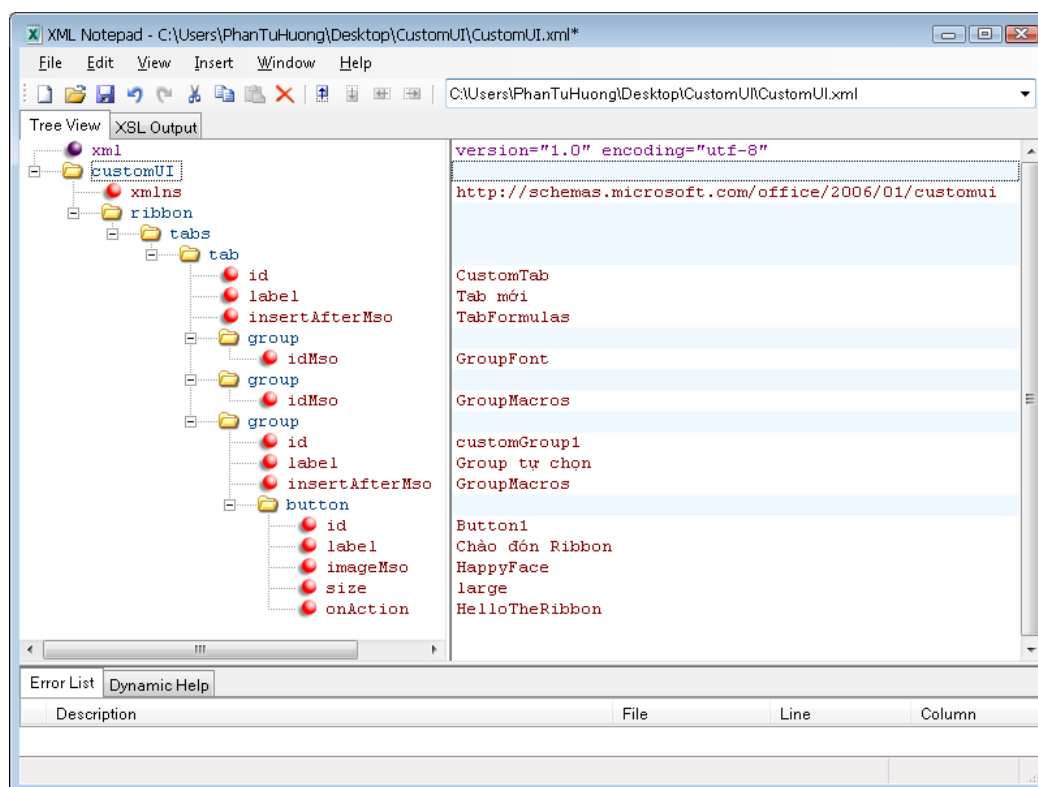


Hình 21-46: Tạo đường dẫn tới XML Schemas trong XN2007

Việc soạn thảo và quản lý mã XML trong Nopated (hay WordPad) khó khăn vì không phân biệt được đối tượng cùng với thuộc tính của chúng. Ngoài ra, chúng không phát hiện được sai sót trong quá trình làm việc. Để dễ dàng Việt hoá Ribbon cũng như xây dựng các điều khiển thuận tiện, chúng ta sử dụng XN2007. Bình thường, XN2007 không hỗ trợ chức năng tự động nhận danh sách đối tượng cùng với thuộc tính của chúng. Do đó, Microsoft cung cấp thêm công cụ hỗ trợ “Office 2007 XML Schema” để giúp chúng ta nhận biết đối tượng và thuộc tính của chúng. Khi cài đặt “Office 2007 XML Schema”, chúng ta tạo đường dẫn từ XN2007 bằng cách vào menu View/Schemas... Cửa sổ “XML Schemas” hiện ra như hình 21-46, khi đó chúng ta tìm đường dẫn tới tập tin customUI.xsd vừa được tạo ra. Như vậy, công việc thiết lập ban đầu đã hoàn thành và chúng ta bắt đầu tìm hiểu XN2007 làm việc như thế nào?!

Trong cửa sổ XN2007, chúng ta mở tập tin CustomUI.xml trong thư mục CustomUI tại Desktop, XN2007 tự động nhận mã XML đã có (hình 21-47). Cửa sổ XN2007 chứa hai phần là Tree View và XSL Output.

TreeView bên trái cho thấy hệ thống phân cấp quản lý XML từ cao xuống thấp, cột bên phải hiển thị giá trị các thuộc tính. Các đối tượng có hình thư mục và chữ màu xanh da trời (như tab, group, button,...), còn các thuộc tính bên trong đối tượng có nút tròn bóng đỏ và chữ màu tím, chỉ dẫn (comment) có nút tròn bóng xanh lá cây. Chúng ta có thể thay đổi màu sắc, font, kiểu định dạng trong thiết lập Option.

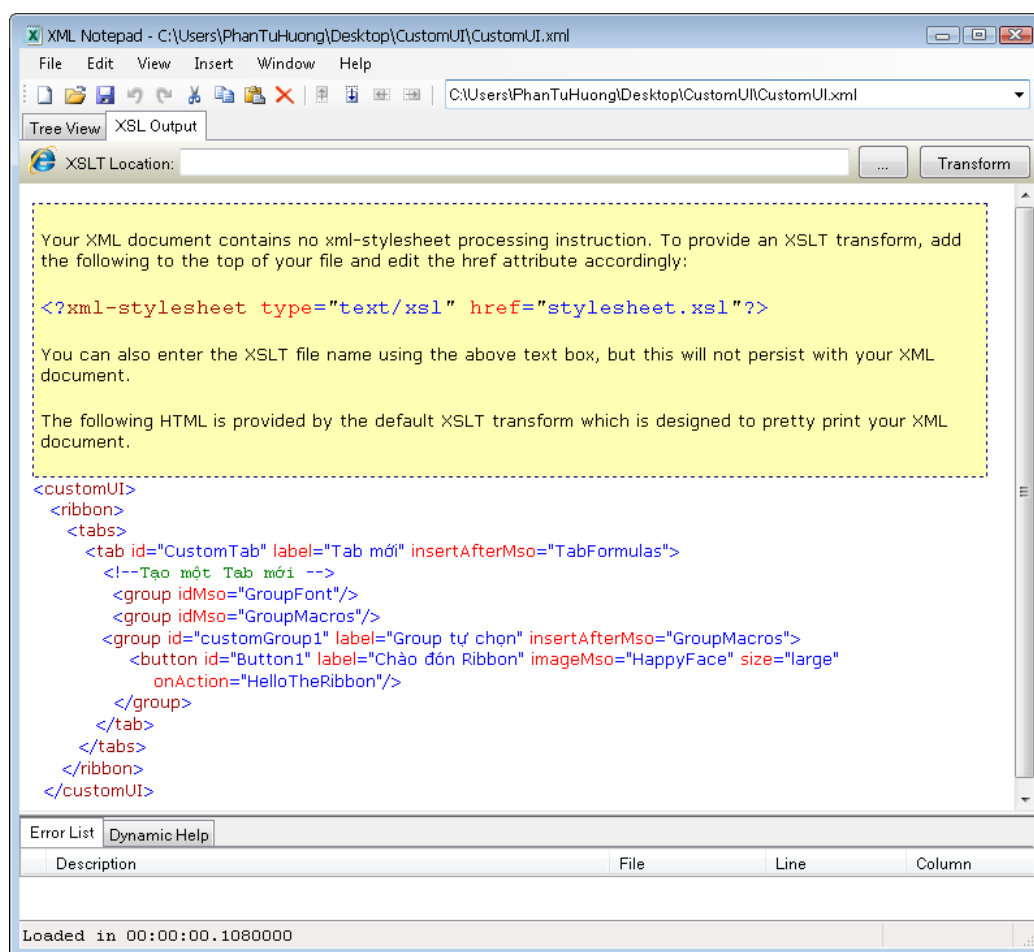


Hình 21-47: Cấu trúc XML của CustomTab.xlsm trong XN2007

XSL Output chứa trình duyệt web của đầu ra HTML từ một chuyển đổi XSLT liên quan. Chuyển đổi XSLT được xác định theo kiểu xử lý tài liệu XML của bạn:

```
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>
```

Với XN2007, chúng ta dễ dàng tạo đối tượng Ribbon và các thuộc tính của chúng. Khi di chuyển chuột đến đối tượng hoặc thuộc tính của chúng, chúng ta sẽ nhận được thông tin hỗ trợ. Dưới đây là một số tính năng của XN2007:



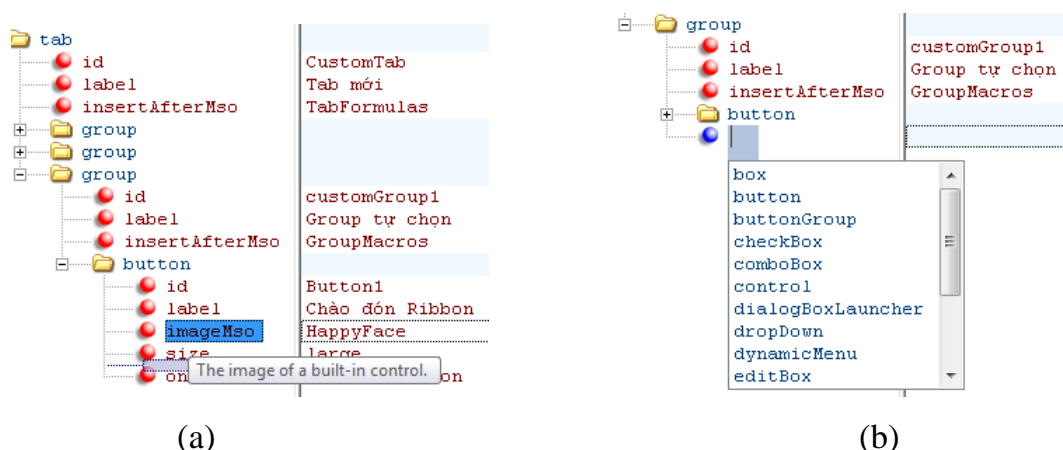
Hình 21-48: Cửa sổ XSL Output chứa mã XML trong XN2007

4.1. Hỗ trợ Clipboard:

Chúng ta có thể sử dụng chức năng cut/copy/paste và kéo/thả được dựa trên cùng một định dạng clipboard của XML. Ví dụ: có thể sao chép mã XN2007 đến bất kỳ trình soạn thảo có hỗ trợ một định dạng clipboard như Notepad, Wordpad,... Ngoài ra, XN2007 đọc được các tập tin mã XML và quản lý chúng một cách dễ dàng.

4.2. Hỗ trợ kéo/thả:

Chúng ta có thể sử dụng chức năng kéo/thả các nút di chuyển trong Tree View. Khi di chuyển một nút nào đó (ví dụ: imageMso trong hình 21-49a) trong Tree View, chọn đối tượng giữ trái chuột, sẽ xuất hiện bóng mờ theo vị trí di chuyển của chuột để đặt đối tượng vào. Trong trường hợp muốn copy đối tượng thì bấm thêm phím Ctrl.



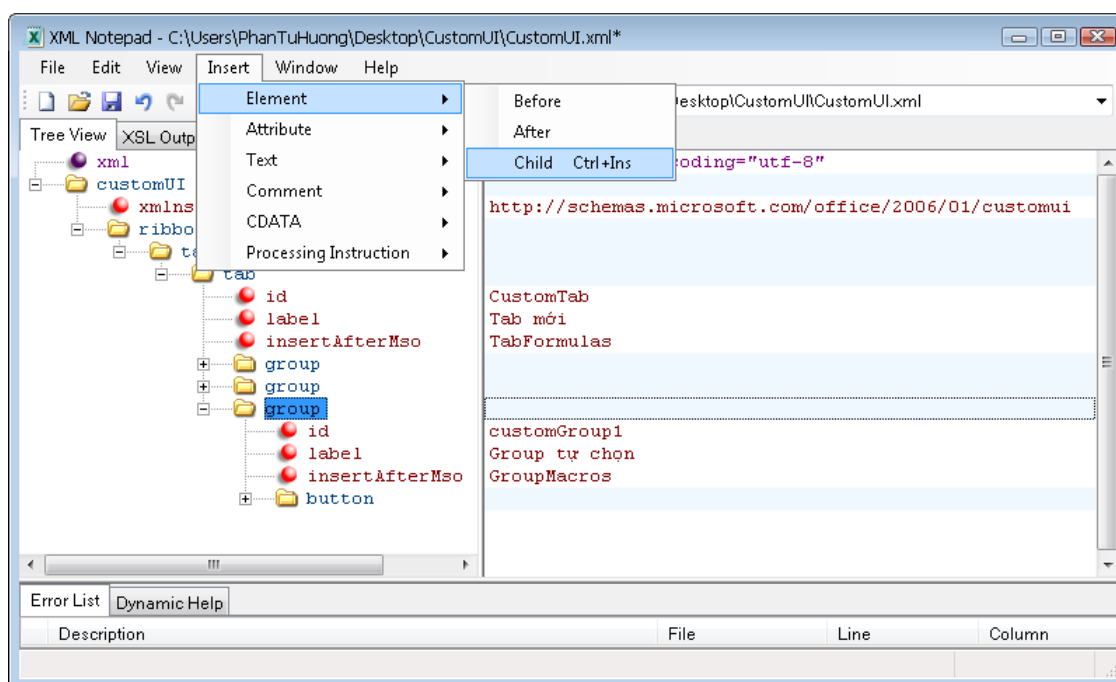
Hình 21-49: Di chuyển nút (a) và danh sách đối tượng trong group (b)

4.3. Giảm độ hiệu lực XML:

Chức năng này xác định hiệu lực của dữ liệu trong quá trình soạn thảo mã XML và hiển thị lỗi hoặc cảnh báo trong phần Error List nếu mắc phải. Ngoài ra, chúng còn hỗ trợ khi làm việc nhờ các dấu nhắc các thuộc tính, giá trị của các phần tử bởi trình cảm ứng thông minh Intellisense (hình 21-49b). Khi xây dựng mã XML, chúng ta dễ dàng lựa chọn điều khiển cũng như các thuộc tính của chúng mà không sợ sai sót. Để có được sự hỗ trợ này, chúng ta phải cài đặt thêm công cụ “Office 2007 XML Schema” như đã trình bày ở trên.

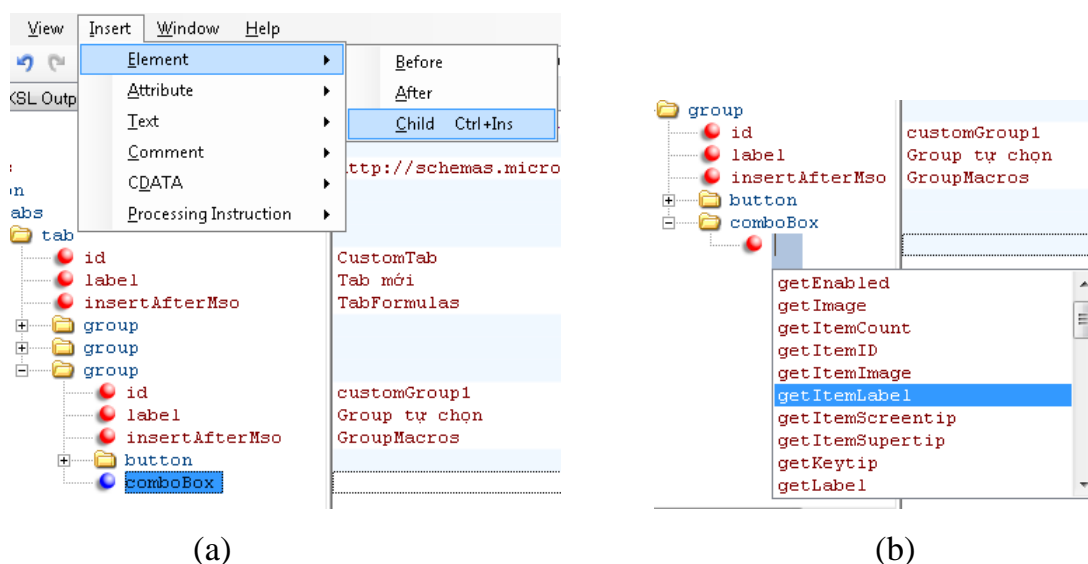
4.4. Tạo các phần tử và thuộc tính của chúng:

Để tạo điều khiển mới trong điều kiện quản lý chúng (ví dụ: group), vào menu/Insert/Element/Childs hoặc phím tắt Ctrl+Ins (hình 21-50). Trong trường hợp đã có điều khiển đồng mức với điều khiển mới (ví dụ: button), hãy chọn điều khiển đó và vào menu/Insert/Element/After (hoặc Before) hoặc bấm phím phải chuột rồi chọn Insert để tạo mới điều khiển đứng sau (hoặc đứng trước) điều khiển được chọn đó (hình 21-50). Trong danh sách thả xuống (hình 21-49b), chọn điều khiển comboBox. Nếu điều khiển mới tạo không phù hợp, bấm trái chuột vào điều khiển đó và danh sách điều khiển hiện ra để chúng ta lựa chọn thay thế. Bây giờ chúng ta xây dựng các thuộc tính cho điều khiển mới tạo đó.



Hình 21-50: Tạo điều khiển mới nằm trong Group quản lý

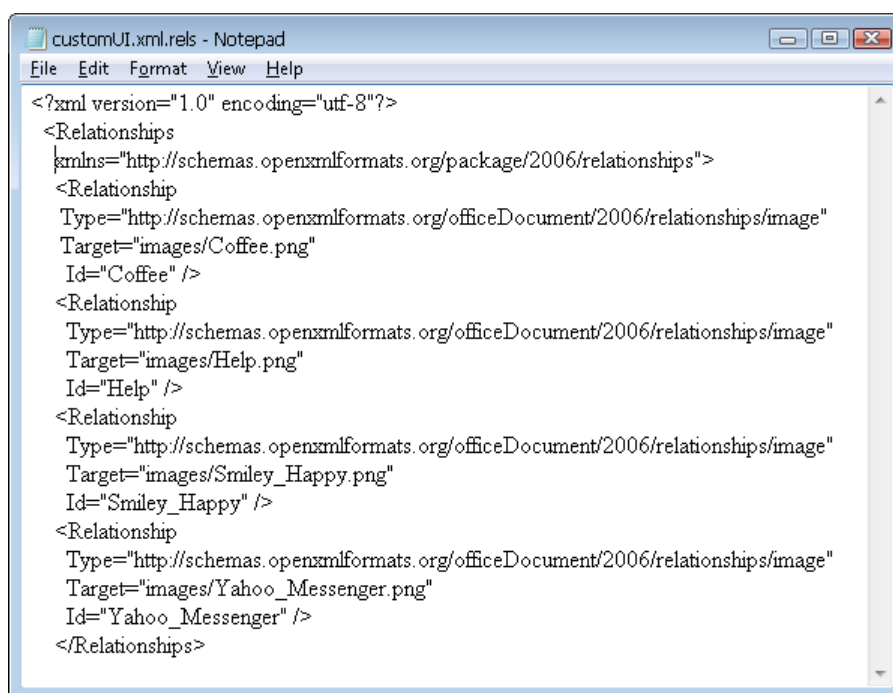
Điều khiển ban đầu mới tạo ra có biểu tượng nút tròn màu xanh, khi đã có thuộc tính bên trong thì trở thành biểu tượng thư mục (hình 21-51). Với điều khiển có nhiều phần tử (item), hãy nhớ không đặt trùng giá trị id cho các phần tử đó. Đây cũng là nguyên nhân để Ribbon không hoạt động.



Hình 21-51: Xây dựng thuộc tính cho điều khiển mới comboBox

4.5. Xây dựng hình ảnh của điều khiển Ribbon:

Như trong mục 21.3.3 đã đề cập, chúng ta có thể tạo hình ảnh từ bên ngoài cho đối tượng Ribbon nhờ chức năng “Insert Icons” của CUE. Còn trường hợp xây dựng Ribbon trong XN2007 thì thực hiện như thế nào? Trong thư mục CustomUI của tập tin khi đã chuyển đổi sang đuôi zip, chúng ta tạo hai thư mục con Images và _rels. Sau đó, sao chép toàn bộ hình ảnh cần sử dụng vào trong thư mục Images (lưu ý những hình ảnh đó phải đúng quy định của Ribbon). Tạo tập tin customUI.xml.rels trong thư mục _rels để khai báo toàn bộ hình ảnh trong thư mục Image ở trên. Hình 21-52 thể hiện cấu trúc tập tin customUI.xml.rels khai báo hình ảnh bên ngoài của tập tin CustomImage.xlsm (hình 21-39).



Hình 21-52: Cấu trúc tập tin customUI.xml.rels khai báo hình ảnh bên ngoài

Như vậy, để khai báo một hình ảnh bên ngoài (ví dụ: Logo.png), cấu trúc tập tin customUI.xml.rels như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<Relationships
  xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship
    Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image"
    Target="images/Logo.png"
    Id="Logo" />
</Relationships>
```

Khi sử dụng CUE để tải hình ảnh bên ngoài, công việc tạo thư mục cùng các tập tin bên trong được thiết lập một cách tự động. Do đó, chúng ta có thể phối hợp giữa XN2007 với CUE trong công việc xây dựng điều khiển Ribbon sao cho đạt hiệu quả cao nhất, hạn chế tối thiểu sai sót.

5. Xây dựng các điều khiển Ribbon

Như đã thống kê ở bảng 21-6, có nhiều loại điều khiển Ribbon mà chúng ta có thể xây dựng tùy theo yêu cầu làm việc. Mục này trình bày cách xây dựng một số đối tượng Ribbon hay sử dụng.

5.1. Xây dựng điều khiển button, checkBox, editBox:

Chúng ta sử dụng XN2007 để xây dựng group mới trong tab Home, đứng trước group Alignment. Group đó có chứa các điều khiển button “Upper Case”, checkBox “Gridline” và editBox “Zoom (%)”.

Đầu tiên, chúng ta tạo tập tin CustomControl1.xlsm và thực hiện như các bước ở trên. Sau đó xây dựng group cùng các điều khiển theo các bước sau:

- Bước 1: Copy nội dung mã XML bên dưới vào tập tin customUI.xml của CustomControl1.xlsm.zip. Kiểm tra mã XML có bị lỗi hay không trong Error List. Lưu lại, bỏ đuôi .zip để trở về tập tin ban đầu.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab idMso="TabHome">
        <group id="customGroup1" label="Custom Format" insertAfterMso="GroupFont" >
          <button id="Button1"
            label="Upper Case"
            size="normal"
            onAction="UpperCase"
            imageMso="BevelTextGallery" />
          <checkBox id="Checkbox1"
            label="Gridlines"
            onAction="Checkbox1_Change"/>
          <editBox id="EditBox1"
            showLabel="true"
            label="Zoom (%):"
            onChange="EditBox1_Change"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

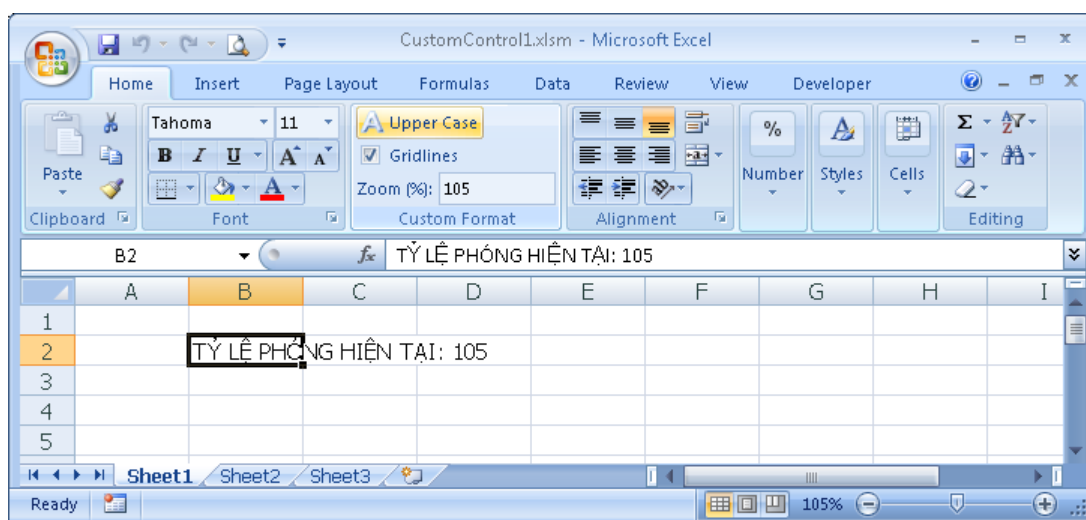
- Bước 2: Mở CustomControl1.xlsm trong Excel, tạo các thủ tục để thi hành các điều khiển trên khi bấm vào (onAction) hay khi thay đổi (onChange) như sau:

```
'Gán cho nút UpperCase, dùng để biến thành chữ hoa ô hiện hành
Sub UpperCase(control As IRibbonControl)
    ActiveCell.Value = UCase(ActiveCell.Value)
End Sub
```

```
'Gán cho checkBox Gridlines khi có sự thay đổi (Change)
Sub Checkbox1_Change(control As IRibbonControl, pressed As Boolean)
    If pressed = False Then
        ActiveWindow.DisplayGridlines = False    'Tắt lưới ô
    Else
        ActiveWindow.DisplayGridlines = True     'Bật lưới ô
    End If
End Sub
```

```
'Gán cho editBox Zoom khi có sự thay đổi
Sub EditBox1_Change(control As IRibbonControl, text As String)
    If IsNumeric(text) = False Then
        MsgBox "Bạn phải nhập số! Yêu cầu nhập lại!",
            vbExclamation, "Custom Group"
    Else
        ActiveWindow.Zoom = text
        Range("B2").Value = "Tỷ lệ phóng hiện tại: " & text
    End If
End Sub
```

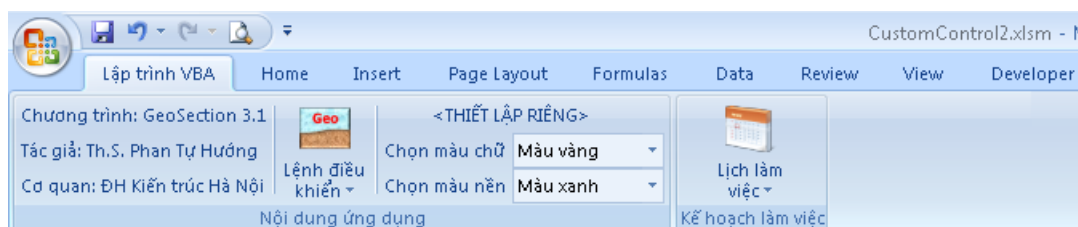
Hãy lưu lại, đóng tập tin CustomControl1.xlsm và mở lại. Chúng ta sẽ thấy group cùng với các điều khiển được tạo ra. Bây giờ hãy thử sự hoạt động của các điều khiển đó xem sao? Chúng hoạt động tốt đấy chứ!



Hình 21-53: Group và điều khiển Ribbon bên trong được tạo mới

5.2. Xây dựng điều khiển label, menu, combobox, gallery:

Công việc tiếp theo là chúng ta xây dựng tab “Lập trình VBA” đứng trước tab Home. Tab này chứa 2 group “Nội dung ứng dụng”, “Kế hoạch làm việc”. Group “Nội dung ứng dụng” chứa 3 nhóm được tách riêng gồm điều khiển label, menu “Lệnh điều khiển”, comboBox “Chọn màu chữ”, “Chọn màu nền”. Group “Kế hoạch làm việc” gallery “Lịch làm việc” (hình 21-54). Ribbon này có chứa hai hình ảnh bên ngoài (xem cách khai báo, quản lý hình ảnh ở mục trước).



Hình 21-54: Xây dựng điều khiển Ribbon trong CustomControl2.xlsm

Nội dung mã trong CustomUI.xml như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="CustomTab" label="Lập trình VBA" insertBeforeMso="TabHome">
        <group id="customGroup1" label="Nội dung ứng dụng" insertAfterMso="GroupMacros">
          <labelControl id="Label1" label="Chương trình: GeoSection 3.1" />
          <labelControl id="Label2" label="Tác giả: Th.S. Phan Tự Hường" />
          <labelControl id="Label3" label="Cơ quan: ĐH Kiến trúc Hà Nội" />
          <separator id="Sep1"></separator>
          <menu id="Menu1" label="Lệnh điều khiển" image="Logo" enabled="true" size="large">
            <button id="button1" label="Vẽ hình trụ hồ khoan" imageMso="Chart3DColumnChart" />
            <button id="button2" label="Vẽ mặt cắt Địa chất CT" imageMso="ChartAreaChart" />
          </menu>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

```

<button id="button3" label="Lập bảng tổng hợp Excel" imageMso="ExportExcel" />
<button id="button4" label="Thông tin các lớp đất đá"
    imageMso="TentativeAcceptInvitation" />
<button id="button5" label="Xuất nội dung sang Word" imageMso="ExportWord" />
</menu>
<separator id="Sep2"></separator>
<labelControl id="Label4" label="<THIẾT LẬP RIÊNG>" />
<comboBox id="combobox1" enabled="true" label="Chọn màu chữ"
    screentip="Chọn màu chữ trong ô hiện hành"
    supertip="Chọn một trong bốn màu chữ trong ô hiện hành gồm xanh, đỏ, tím, vàng."
    onChange="FontColor_Change">
    <item id="Item1" label="Màu xanh" imageMso="AppointmentBusy"></item>
    <item id="Item2" label="Màu đỏ" imageMso="AppointmentColor1"></item>
    <item id="Item3" label="Màu tím" imageMso="AppointmentOutOfOffice"></item>
    <item id="Item4" label="Màu vàng" imageMso="AppointmentColor10"></item>
</comboBox>
<comboBox id="combobox2" enabled="true" label="Chọn màu nền"
    screentip="Thiết lập màu trong danh sách chọn" onChange="GroundColor_Change">
    <item id="Item5" label="Màu xanh" imageMso="AppointmentBusy"></item>
    <item id="Item6" label="Màu đỏ" imageMso="AppointmentColor1"></item>
    <item id="Item7" label="Màu tím" imageMso="AppointmentOutOfOffice"></item>
    <item id="Item8" label="Màu vàng" imageMso="AppointmentColor10"></item>
</comboBox>
</group>
<group id="customGroup2" label="Kế hoạch làm việc">
    <gallery id="Gallery1" image="Calendar" size="large" label="Lịch làm việc"
        columns="1" rows="8" onAction="TimeSelected">
        <item id="Progress1" label="Từ 8h30' đến 9h30'" imageMso="FileManageMenu" />
        <item id="Progress2" label="Từ 9h30' đến 10h30'" imageMso="AutoDial" />
        <item id="Progress3" label="Từ 10h30' đến 11h30'" imageMso="OpenStartPage" />
        <item id="Progress4" label="Từ 11h30' đến 12h30'" imageMso="MeetingsWorkspace" />
        <item id="Progress5" label="Từ 12h30' đến 14h00'" imageMso="FileBackupDatabase" />
        <item id="Progress6" label="Từ 14h00' đến 16h00'"
            imageMso="FormulaMoreFunctionsMenu" />
        <item id="Progress7" label="Từ 16h00' đến 17h00'" imageMso="ManageReplies" />
        <item id="Progress8" label="Từ 17h00' trở đi" imageMso="BlogHomePage" />
        <button id="Nowis" label="Thời điểm hiện tại ..."
            imageMso="ViewAppointmentInCalendar" onAction="ShowNow" />
    </gallery>
</group>
</tab>
</tabs>
</ribbon>
</customUI>

```

Tiếp theo, chúng ta xây dựng các thủ tục khi bấm vào điều khiển Ribbon cho một số điều khiển như sau:

```

'Thực hiện lệnh khi bấm thay đổi màu chữ (combobox1)
Sub FontColor_Change(control As IRibbonControl, ByRef returnedVal)
    If returnedVal = "Màu xanh" Then
        ActiveCell.Font.Color = -4165632
    ElseIf returnedVal = "Màu đỏ" Then
        ActiveCell.Font.Color = -16776961
    ElseIf returnedVal = "Màu tím" Then
        ActiveCell.Font.Color = -16777024
    Else
        ActiveCell.Font.Color = -16727809
    End If
End Sub

```

```
End Sub
```

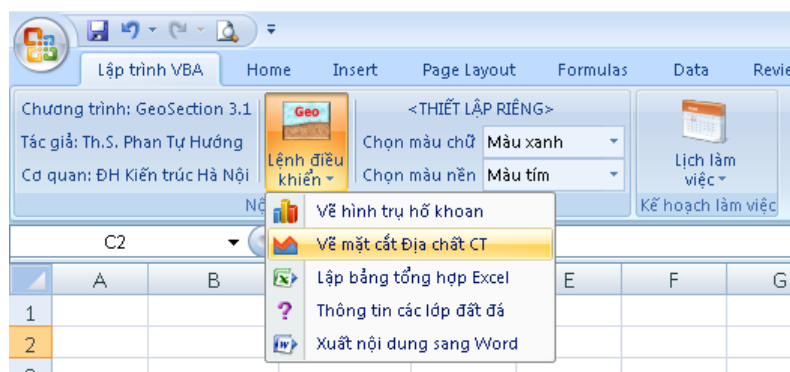
```
'Thực hiện lệnh khi bấm thay đổi màu nền (combobox2)
Sub GroundColor_Change(control As IRibbonControl, ByRef returnedVal)
    With ActiveCell.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        If returnedVal = "Màu xanh" Then
            .Color = 12611584
        ElseIf returnedVal = "Màu đỏ" Then
            .Color = 255
        ElseIf returnedVal = "Màu tím" Then
            .Color = 192
        Else
            .Color = 49407
        End If
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
End Sub
```

```
'Thực hiện khi chọn phần tử trong danh sách của Gallery1
Sub TimeSelected(control As IRibbonControl, id As String,
    index As Integer)
    MsgBox index      'Hiện số thứ tự trong danh sách (bắt đầu từ 0)
End Sub
```

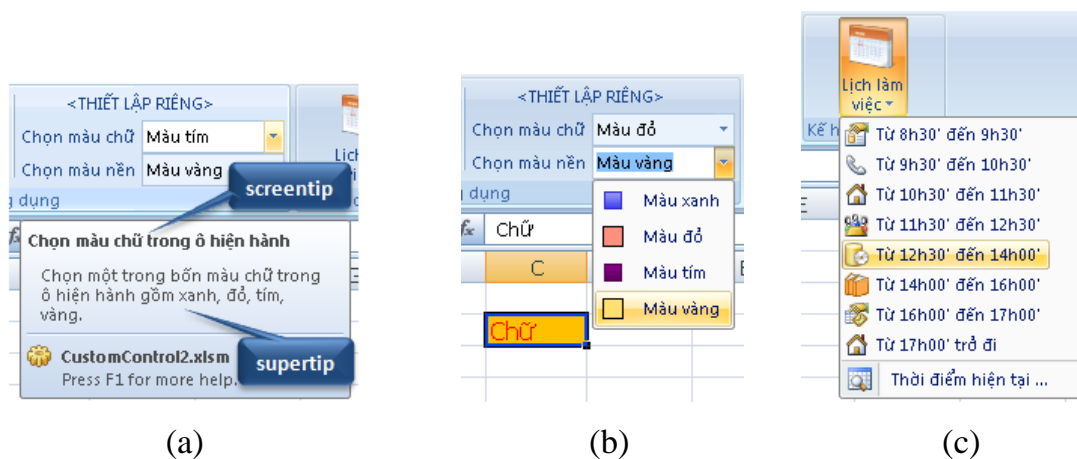
```
'Thực hiện khi bấm nút Nowis nằm dưới cùng của Gallery1
Sub ShowNow(control As IRibbonControl)
    MsgBox Now      'Hiện thời gian và ngày hiện tại
End Sub
```

Hãy mở CustomControl2.xlsm trong Excel, kết quả thể hiện như hình 21-55. Khi di chuyển chuột lên comboBox “Chọn màu chữ”, thông tin về chức năng làm việc của điều khiển đó được thể hiện như hình 21-56a nhờ các thuộc tính screentip, supertip. Với điều khiển gallery “Lịch làm việc”, khi chọn phần tử nào thì thứ tự (index) của nó hiện ra, phần tử đầu tiên bắt đầu từ 0 (hình 21-56c). Điều khiển button

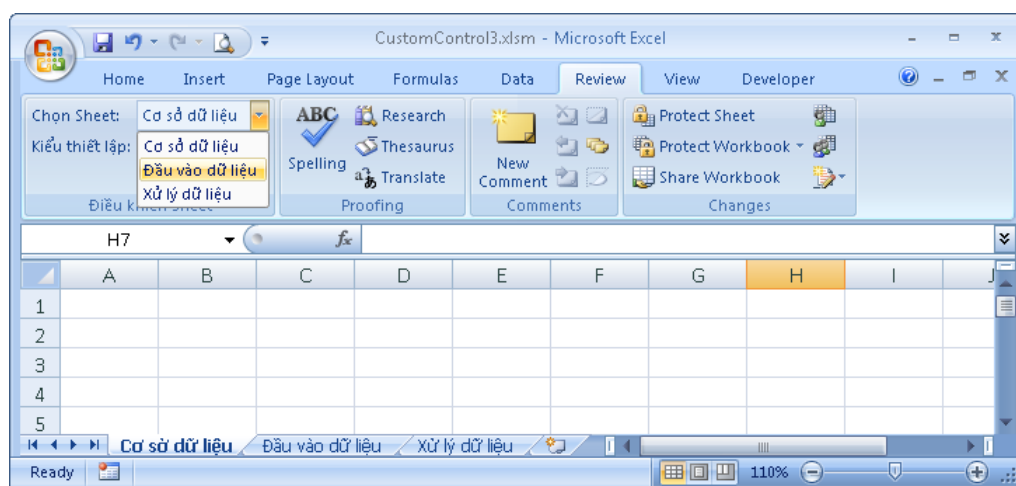
Nowis (label là “Thời điểm hiện tại ...”) nằm dưới cùng trong danh sách hiển thị thời gian hiện tại.



Hình 21-55: Xây dựng điều khiển Ribbon trong CustomControl2.xlsm



Hình 21-56: Chi tiết các điều khiển Ribbon



Hình 21-57: Xây dựng hai điều khiển dropDown trong CustomControl3.xlsm

5.3. Xây dựng điều khiển *dropDown*:

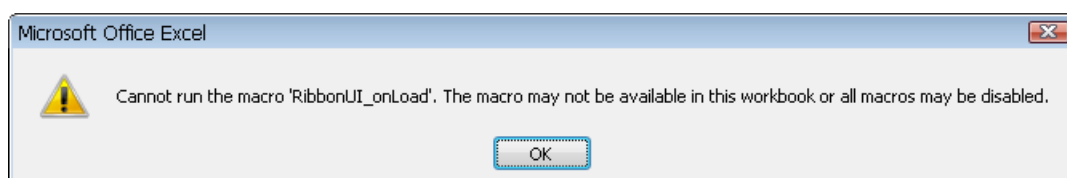
Ở phần trước, chúng ta sử dụng các lệnh và chức năng đã có sẵn khi xây dựng Ribbon. Tức là đa số các phần tử của điều khiển được xác định cụ thể trong XML, chúng ta không thể bổ sung hay xoá bớt khi cần thiết. Trong phần này, thay vì sử dụng một danh sách tĩnh thả xuống (ví dụ *comboBox* trước đó), chúng ta tạo ra điều khiển *dropDown* chứa danh sách động trong khi thả xuống.

Điều khiển thứ nhất *dropDown* “Chọn Sheet:” liệt kê tất cả các bảng tính trong Workbook. Khi chọn Sheet nào trong danh sách thả xuống, Sheet đó được kích hoạt. Trong trường hợp này, chúng ta sử dụng callback để cập nhật danh sách Sheet thả xuống như thêm hoặc gỡ bỏ bảng tính từ Workbook.

Điều khiển thứ hai *dropDown* “Kiểu thiết lập:” hiện danh sách thuộc tính của Sheet được chọn như “Hiển thị” (Visible), Ẩn (Hidden) và “Siêu ẩn” (VeryHidden). Ví dụ này khá thú vị bởi vì sử dụng hai điều khiển *dropDown* có liên hệ với nhau. Để cập nhật danh sách tự động (là các Sheet trong Workbook), chúng ta cần nắm bắt đối tượng *RibbonUI*. Đầu tiên, bổ sung phần tử *onLoad* trong *CustomUI* (dòng thứ 2). Nội dung mã XML như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<customUI onLoad="RibbonUI_onLoad"
xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab idMso="TabReview">
        <group id="customGroup1" label="Điều khiển Sheet" insertBeforeMso="GroupProofing">
          <dropDown id="SelectSheet" label="Chọn Sheet:" onAction="SelectSheet_Click"
            getItemID="SelectSheet_getItemId"
            getItemCount="SelectSheet_getItemCount"
            getItemLabel="SelectSheet_getItemLabel"></dropDown>
          <dropDown id="SheetVisible" label="Kiểu thiết lập:"
            onAction="SheetVisible_Click">
            <item id="SheetVisible1" label="Hiển thị"></item>
            <item id="SheetVisible2" label="Ẩn"></item>
            <item id="SheetVisible3" label="Siêu ẩn"></item>
          </dropDown>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Khi đối tượng chứa mã XML trên được mở, thủ tục *RibbonUI_onLoad* trong VBA callback đến mã XML để thay đổi các thiết lập trong chúng. Hãy lưu lại và mở *CustomControl3.xlsm* trong Excel, chúng ta nhận được thông báo như hình 21-58.



Hình 21-58: Xây dựng hai điều khiển dropDown trong CustomControl3.xlsm

Thông báo này đã được biết trước, vì chúng ta đã khai báo onLoad nhưng chưa cung cấp chương trình hoạt động. Để thực hiện điều đó, mở cửa sổ MVB, tạo module thường và xây dựng các đoạn mã trong đó (có thể sử dụng CUE để tạo mã ban đầu rồi copy sang VBA).

Ở đầu của module, chúng ta khai báo 2 biến. Biến đầu tiên tham chiếu tới đối tượng Ribbon, biến thứ hai lưu giữ tên các Sheet trong Workbook. Đảm bảo rằng đối tượng RibbonUI được giữ lại trong thời gian tải bằng cách thiết lập callback cho onLoad như sau:

```
Option Explicit
Public RibbonUI As IRibbonUI
Dim SheetName As String
'Thiết lập callback cho customUI.onLoad
Sub RibbonUI_onLoad(ribbon As IRibbonUI)
    Set RibbonUI = ribbon
End Sub
```

Tiếp theo, chúng ta xây dựng thủ tục callback cho các điều khiển bên trong đối tượng dropDown “Chọn Sheet.”:

```
'Thiết lập callback getItemLabel cho dropDown SelectSheet
Sub SelectSheet_getItemLabel(control As IRibbonControl,
    Index As Integer, ByRef returnedVal)
    returnedVal = Worksheets(Index + 1).Name
End Sub
```

```
'Thiết lập callback getItemID cho dropDown SelectSheet
Sub SelectSheet_getItemID(control As IRibbonControl,
    Index As Integer, ByRef id)
    id = "SelectSheet" & Index
End Sub
```

```
'Thiết lập callback onAction cho dropDown SelectSheet
Sub SelectSheet_Click(control As IRibbonControl, id As String,
    Index As Integer)
    On Error Resume Next
    Call SelectSheet_getItemLabel(control, Index, SheetName)
    Worksheets(Index + 1).Select
    If Err.Number <> 0 Then
        MsgBox "Sheet này không tồn tại! Sẽ thiết lập lại toàn bộ Sheet
            hiện hành ngay bây giờ.", vbInformation, "Ribbon in Excel"
        RibbonUI.InvalidateControl "SelectSheet"
    End If
End Sub
```

Chúng ta đã sử dụng các callback để điều khiển dropDown “Chọn Sheet:” như sau:

- getItemCount: xác định tổng các mục con.
- getItemLabel: xác định nhãn các mục con.
- getItemID: xác định chỉ danh các mục con.

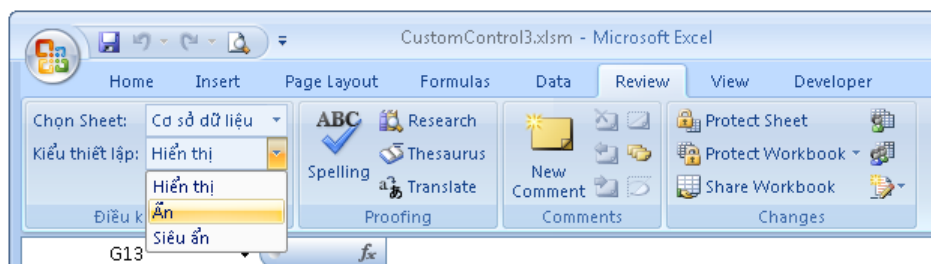
Thủ tục SelectSheet_Click sử dụng thuộc tính InvalidateControl của đối tượng RibbonUI. Thuộc tính này cập nhật đối tượng được chọn (là “SelectSheet” ở thủ tục trên). Chúng ta sử dụng thuộc tính trên vì trong quá trình làm việc với CustomControl3.xlsm, khi số lượng hoặc tên Sheet có sự thay đổi không giống ban đầu, thủ tục SelectSheet_Click sẽ bị lỗi. Khi bị lỗi, thủ tục sẽ cập nhật lại danh sách Sheet hiện tại trong Workbook.

Như vậy, điều khiển dropDown “Chọn Sheet:” có số mục con có thể thay đổi (danh sách động), phụ thuộc vào số lượng Sheet trong Workbook hiện hành.

Công việc tiếp theo thiết lập thuộc tính Visible cho Sheet được chọn trong danh sách trên.

```
'Thiết lập callback onAction cho dropDown SheetVisible
Sub SheetVisible_Click(control As IRibbonControl, id As String,
    index As Integer)
    'Kiểm tra xem Sheet đã được chọn chưa?
    On Error Resume Next
    SheetName = Worksheets(SheetName).Name
    If Err.Number <> 0 Then
```

```
MsgBox "Bạn phải chọn đúng Sheet trước khi thực hiện!"
Exit Sub
End If
'Thiết lập trường hợp chọn
Select Case id
Case "SheetVisible1"
    Worksheets(SheetName).Visible = xlSheetVisible
    Worksheets(SheetName).Select
Case "SheetVisible2"
    Worksheets(SheetName).Visible = xlSheetHidden
Case "SheetVisible3"
    Worksheets(SheetName).Visible = xlSheetVeryHidden
End Select
'Trường hợp còn Sheet cuối cùng (không thể ẩn được)
If Err.Number <> 0 Then
    MsgBox "Bạn không thể ẩn toàn bộ Sheet!"
End If
On Error GoTo 0
End Sub
```



Hình 21-59: Thiết lập chế độ ẩn/hiện cho Sheet

Chúng ta có thể tải các bộ cài hỗ trợ việc xây dựng Ribbon như đã đề cập ở các mục trên, các tập tin ví dụ đã được tạo Ribbon trong trang www.giaiphapexcel.com hoặc www.bluesofts.net.

Tài liệu tham khảo

- [1]. Mark Dodge, Craig Stinson, 2000. Excel 2000 toàn tập. Nhà xuất bản trẻ.
- [2]. Nguyễn Thị Ngọc Mai (Chủ biên) và nnk, 2000. Microsoft Visual Basic 6.0 và lập trình cơ sở dữ liệu. Nhà xuất bản Lao động - Xã hội.
- [3]. Introduction to using macros in Excel 2003, 2005. University of Durham Information Technology Service.
- [4]. Julitta Korol, 2002. Learn Excel 2002 VBA Programming with XML and ASP. Wordware.Publishing.
- [5]. John Walkenbach, 2007. Excel 2007 VBA Programming for Dummies. Wiley Publishing, Ins.
- [6]. David Boctor, 1999. Microsoft Office 2000/Visual Basic for Application/Fundamentals, Microsoft Office.
- [7]. Steven M. Hansen, 2004. Mastering Excel 2003 Programming with VBA. SYBEX.
- [8]. Stephen Bullen, Rob Bovey, John Green, 2009. Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft® Excel, VBA® and .NET. Addison Wesley Professional.
- [9]. Excel VBA Simulation Basic Tutorial. Nguồn Internet.
- [10]. Microsoft Visual Basic Help, 2010, Microsoft.
- [11]. Chip Pearson & John Walkenbach, 2001. Excel Macro. Nguồn Internet.
- [12]. Bill Jelen, Tracy Syrstad, 2010. VBA and Macros: Microsoft® Excel® 2010. Que Publishing.
- [13]. Robert L. McDonald, 2000. An Introduction to VBA in Excel. Finance Dept, KelloggSchool, Northwestern University.
- [14]. Các tư liệu trên trang web: www.giaiphapexcel.com, www.caulacbovb.com, www.ketcau.com, www.cadviet.com,...
- [15]. John Green, Stephen Bullen, Rob Bovey, Michael Alexander, 2007. Excel ® 2007 VBA Programmer's Reference. Wiley Publishing.

[16]. Robert Martin, Ken Puls, Teresa Hennig, 2007. RibbonX: Customizing the Office 2007 Ribbon. Wiley Publishing.

[17]. David & Raina Hawley, 2007. Excel Hacks – Tips and Tools for Streamlining Your Spreadsheets (second edition), O'Reilly Media.

[18]. Phan Tụ Hường, 2006. Ứng dụng ngôn ngữ VBA trong Excel để giải một số bài toán trong Địa chất công trình. Tạp chí khoa học Mỏ - Địa chất.

[19]. Phan Tụ Hường, 2007. Ứng dụng ngôn ngữ lập trình VBA trong Excel để tự động hoá tính toán và xử lý thống kê chỉ tiêu cơ lý đất dính. Đề tài nghiên cứu khoa học cấp trường - Trường Đại học Mỏ Địa chất.

[20]. Phan Tụ Hường, 2010. Xây dựng mô hình tự động hoá thiết kế trong AutoCad bằng ngôn ngữ lập trình VB6. Tạp chí khoa học, kiến trúc và xây dựng - Trường ĐH Kiến trúc Hà Nội.

[21]. Phan Tụ Hường, 2011. Giới thiệu phần mềm lập báo cáo khảo sát địa chất công trình GeoSection. Tạp chí khoa học Mỏ - Địa chất, số 35.