

La barra multifunzione introdotta in Office 2007 è sicuramente la novità più d'impatto di questa suite. In questa guida vedremo come è possibile personalizzare la barra nelle nostre applicazioni Access.

Personalizzare la barra multifunzione significa creare un file XML che è conforme ad uno schema predefinito.

E' possibile scaricare lo schema da questo link: <http://www.microsoft.com/downloads/details.aspx?FamilyId=15805380-F2C0-4B80-9AD1-2CB0C300AEF9&displaylang=en>

In Access 2007 ci sono due modi per personalizzare la barra multifunzione (o ribbon):

1. Creare una tabella di sistema USysRibbons contenente i campi

- ID
- RibbonName (text)
- RibbonXML (memo)

2. Usare il metodo LoadCustomUI dell'oggetto Application , questo è anche l'unico metodo utilizzabile nei progetti ADP.

Al fine di individuare più facilmente gli errori nella definizione del Ribbon, come prima cosa andate su Office Menu – Opzioni Access – Impostazioni Avanzate – Sezione Generale e mettete la spunta a Mostra errori dell'interfaccia utente dei componenti aggiuntivi, in questo modo se c'è un errore nella struttura del file XML, vi verrà segnalato con un messaggio d'errore.

Per scrivere il codice XML del Ribbon è consigliabile procurarsi Visual Studio 2005 o 2008 Express che grazie all'Intellisense ci suggeriscono proprietà e metodi supportati da ciascun tag. Poiché in XML è importante la distinzione maiuscole/minuscole l'intellisense ci aiuterà a costruire un XML formalmente corretto.

In Access potete creare una barra multifunzione valida per l'intero database, impostandola in **Office Menu – Opzioni di Access – Database corrente – Nome barra multifunzione**, oppure a livello di singola Maschera o Report impostando la proprietà **Barra Multifunzione** della Maschera o del report al nome del Ribbon che volete utilizzare.

Per far partire Access con la barra standard basta avviare il DB tenendo premuto il tasto Shift.

All'interno della barra potete usare le icone che sono usate in tutte le applicazioni Office, quindi non solo quelle presenti in Access, oppure delle immagini personalizzate a vostra scelta.

Per trovare facilmente il nome dell'immagine Office da associare al vostro pulsante è possibile usare questo trucco: Menu Office – Opzioni di Applicazione – Personalizzazione – rilasciate il mouse sopra l'icona di cui volete conoscere il nome e comparirà una finestrella dove l'ultimo elemento tra parentesi tonde è il nome dell'immagine.

In alternativa è possibile scaricare questo file con la lista di tutte le icone

<http://www.microsoft.com/downloadS/details.aspx?familyid=12B99325-93E8-4ED4-8385-74D0F7661318&displaylang=en>

Una barra multifunzione è tipicamente costituita da un insieme di schede <tabs>

Ogni scheda <tab> è costituita da uno o più gruppi <group>

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
```

```
<ribbon startFromScratch="true">
```

```
<tabs>
```

```
<tab id="Inserimento" visible="true"
label="Inserimento/Modifica">
```

```
<group id="Utenti" label="Utenti"> .... </group>
```

```
</tab>
```

```
</tabs>
```

```
</ribbon>
```

Quando una scheda appare in un particolare “contesto” si parla di scheda contestuale , per cui se volete che una certa scheda sia visualizzata solo per maschere e report userete una istruzione di questo tipo:

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
```

```
xmlns:x="myNameSpace">
<ribbon>
  <contextualTabs
    <tabSet idMso="TabSetFormReportExtensibility" >
      <tab id="MyTab" label="Maschera" >
        <group id="Comandi" label="Comandi" >
          <button id="idStampaEtichette" label="Stampa Etichette"
imageMso="ViewsReportView"
onAction="MacroUtenti.StampaEtichette" showImage="true"
showLabel="true" size="large"/>
        </group>
        <group idMso="GroupClipboard"/></group>
        <group idMso="GroupRecords"/></group>
        <group idMso="GroupSortAndFilter"/></group>
        <group idMso="GroupFindAccess"/></group>
      </tab>
    </tabSet>
  </contextualTabs>
</ribbon>
</customUI>
```

All'interno di un gruppo possono esserci vari elementi come:

- <button>
- <comboBox>
- <dropDown>
- <dynamicMenu>
- <box>
- <labelControl>
- <toggleButton>
- <splitButton>
- <DialogBoxLauncher>
- <gallery>
- <buttonGroup>
- <checkBox>
- <menuSeparator>

Tutta la programmazione del ribbon avviene attraverso l'utilizzo di procedure di CallBack.

All'atto della definizione del ribbon dichiaro i nomi delle procedure che dovranno essere eseguite allo scatenarsi di un particolare evento o che impostano alcune proprietà dell'oggetto. Ogni oggetto che compone il ribbon dispone di un suo insieme di "eventi" e "proprietà" che scatenano la procedura di callback.

Evento OnLoad

```
<customUI xmlns=http://schemas ... onLoad="onRibbonLoad">
```

La procedura scatenata dall'evento onLoad dell'elemento customUI è molto importante perché ci consente di ottenere un puntatore al ribbon personalizzato, attraverso il quale potremo invalidare dei controlli durante l'esecuzione dell'applicazione, al fine di farli rispecchiare la situazione "corrente" che potrebbe essere diversa rispetto a quando il ribbon è stato caricato.

```
Global g_ribbonInv as IRibbonUI Public Sub onRibbonLoad (theRibbon As IRibbonUI)  
Set g_ribbonInv = theRibbon End Sub
```

L'oggetto IRibbonUI espone due metodi :

- Invalidate
- InvalidateControl(ControlID)

A livello del tag customUI è possibile definire anche la funzione che si occuperà del caricamento delle immagini a livello globale in tutto il ribbon, anziché definirla a livello del singolo oggetto.

Es. Supponendo che le immagini si trovino in una cartella Immagini nello stesso percorso del db:

```
Sub LoadImage(ImageName as string,Byref Image) Dim strPath as String strPath =  
CurrentProject.Path & "immagini" & imageName
```

```
Set  
Image=LoadPicture(strPath)
```

```
End sub
```

L'elemento <RibbonUI> dispone di una sola proprietà:

startFromScratch =True/False

Se impostata a true le schede standard vengono nascoste, il pulsante Office viene ridotto e così anche la Quick Access ToolBar

Tutti gli oggetti hanno una proprietà che consente di individuarli in maniera univoca, che può essere una di queste tre:

- id (identificatore univoco per gli oggetti custom)
- idMSO (per gli oggetti built-in)
- idQ (per gli oggetti prefissati da un Namespace)

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
xmlns:x="myNameSpace">
  <button
    idQ="x:myButton" ...=" " />
</customUI>
Evento OnAction
```

Questa procedura viene eseguita quando l'utente clicca sull'oggetto e prende diversi parametri a seconda dell'oggetto a cui si riferisce.

Button

```
Sub OnAction(control As IRibbonControl)
```

Dropdown

```
Sub OnAction(control As IRibbonControl, selectedId As String, selectedIndex As Integer)
```

Gallery

```
Sub OnAction(control As IRibbonControl, selectedId As String, selectedIndex As Integer)
```

CheckBox ☐

```
Sub OnAction(control As IRibbonControl, pressed As Boolean)
```

ToggleButton -

Sub OnAction(control As IRibbonControl, pressed As Boolean) **Evento OnChange**

Questa procedura viene eseguita sulla selezione di un elemento o dalla modifica di una casella di testo, si applica a:

- ComboBox

- EditBox

Sub OnChange(control As IRibbonControl, text As String) **Altre procedure di callback**

- getEnabled
 - Sub GetEnabled(control As IRibbonControl, ByRef enabled)
- getDescription
 - Sub GetDescription(control As IRibbonControl, ByRef description)
- getVisible
 - Sub GetVisible(control As IRibbonControl, ByRef visible)
- getImage
 - Sub GetImage(control As IRibbonControl, ByRef image)
- getLabel
 - Sub GetLabel(control As IRibbonControl, ByRef label)
- getSize
 - sub GetSize(control As IRibbonControl, ByRef size)
- getScreentip

Scritto da Cinzia Pagani

Mercoledì 19 Novembre 2008 01:00 - Ultimo aggiornamento Sabato 19 Settembre 2009 21:03

- Sub GetScreentip(control As IRibbonControl, ByRef screentip)
- getSupertip
- Sub GetSupertip(control As IRibbonControl, ByRef screentip) **Evento**

getItemCount

Si applica a comboBox, dropDown, gallery e permette di impostare il numero di elementi che costituiscono l'oggetto

- Sub GetItemCount(control As IRibbonControl, ByRef count) **Evento**

getItemLabel

Si applica a comboBox, dropDown e gallery e consente di impostare ogni singola etichetta degli elementi

Sub GetItemLabel(control As IRibbonControl, index As Integer, ByRef label) **OfficeMenu e Quick Access Toolbar**

E' possibile personalizzare anche il menu Office e la Quick Access Toolbar, utilizzando i tag:

<officeMenu> e <qat>

figli del tag <ribbon>

Es:

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"> <ribbon
startFromScratch
=
"true">
```

```
<
officeMenu
>
```

```
<
button
```


Scritto da Cinzia Pagani

Mercoledì 19 Novembre 2008 01:00 - Ultimo aggiornamento Sabato 19 Settembre 2009 21:03

idMso

=

"FileNewDatabase" visible="false"/>

<

splitButton

idMso

=

"FileSaveAsMenuAccess" visible="false"/>

<

button

idMso

=

"FileOpenDatabase" visible="false"/>

<

button

idMso

=

"FileCompactAndRepairDatabase"/>

</

officeMenu

>

<

qat

>

<

documentControls

>

<

button

idMso

=

"FilePrintQuick" />

<

button

idMso

=

"FilePrintPreview"/>

Scritto da Cinzia Pagani

Mercoledì 19 Novembre 2008 01:00 - Ultimo aggiornamento Sabato 19 Settembre 2009 21:03

```
<
button
idMso
=
"FileCompactAndRepairDatabase"/>
```

```
</documentControls> </qat> </ribbon> </customUI>
```

Strumenti

Esistono diversi tools che consentono di personalizzare la barra multifunzione, che generano il codice XML in modo "visuale":

- Ribbon Creator (Gunter Avenius MVP)
 - <http://www.accessribbon.de/en/index.php>
- Ribbon Customizer (Patrick Schmid MVP)
 - <http://pschmid.net/office2007/ribboncustomizer/support.php>

Se invece decidete di cimentarvi con l'XML, queste risorse vi saranno senz'altro utili:

- [Schema da usare per convalidare il ribbon](#)
- [Lista dei control ID dei comandi standard](#)
- [Una guida per utilizzare il ribbon in modo consistente](#)
- [Access developer portal](#)
- [Open XML developer \(Custom UI Editor\)](#)