Overview

When you think of modern user interfaces, you usually visualize some kind of toolbar at the top of your window. In fact, I'm sure many of us have gone through the effort of creating static toolbars at the top of our most important forms. Microsoft took toolbars to a whole new level when they introduced the ribbon into their Microsoft Office products. Now, even their most common built-in tools, such as Windows Explorer, incorporate it. This page will familiarize you with some core concepts that are important in understanding how to use the SRP Ribbon Control.

Commands, Not Buttons

The most notable aspect of the SRP Ribbon Control is its use of commands instead of buttons. Commands are actions you define and name. These actions are defined along with a caption and icons. For example, you can define a command called "SAVE" whose caption is "Save" whose icon is an image of a disk. The unique thing about commands is that they are defined independently from any button control on the ribbon.

The reason is simple: sometimes we want a save button in more than one location. Let's look at how OpenInsight works today. Let's say you have a form with both a menu and a toolbar. You have a Save item in the File menu and a button in your toolbar with a disk icon. Both of them do the same action. However, you have to set up event handlers for each control and keep their enabled states synchronized. It's not the end of the world, but if there's an easier way, wouldn't you take it?

In the SRP Ribbon Control, you define a save command and process one event called OnCommand. That event passes you the name of the command, and you process it accordingly. When you add buttons to the ribbon, you associate them to a command. So, you can have three save buttons on your ribbon and each one would fire the same event. Better yet, when you disable the save command, all three buttons are disabled in unison.

Commands are defined in the initialization XML. See the <Command> tag for details.

Keys

Almost everything you define in the SRP Ribbon Control has a key. A key is a string that is unique to that item. Each command has a unique key as does each control. By using keys instead of numerical indexes, it makes it much easier to manipulate the ribbon at runtime. For example, it's much more intuitive to disable a command by name, such as "SAVE", than by index, such as "3". If something is not working, you might want to double check your key names.

Your keys only have to be unique across the same type of element. That is, every command key must be unique and every control key must be unique, but you can use the same key for a command that you use for a control. However, as a general practice, we recommend keeping all your keys as unique as possible to avoid mistakes. This is easily done by using prefixes for visual elements and no prefixes for commands. For example, the save command could be called "SAVE" and one of your save buttons could be called "BTN_SAVE".

Ribbon Elements

The SRP Ribbon Control is composed of several visual components: tabs, groups, controls, and the backstage.

Tabs are usually the first things you define. Tabs do not have icons, only captions. Most of your tabs will have the same visual theme, but you can also add special tabs that are custom colored. For example, in Microsoft Word when you select a table, two new tabs appear highlighted in yellow. Those are called context sensitive tabs. See the <Tabs> XML tag for details on defining tabs.

Groups are the little panes that appear within a single tab. Every group must have a unique key even if the groups are not in the same tab. Groups can have an option button in the bottom corner. When that button is clicked, an OnGroupOptionClick event is fired. Also, as the ribbon shrinks in size and groups get smaller, they will eventually collapse into a popup, so you can associate an icon with the group which will be displayed in that scenario. The <Group> tag is used in the setup XML to define groups.

Controls are placed within groups. There are many kinds of controls supported, and more controls will be added in the future. Some controls are simple, like buttons. Others are more complex, such as split buttons, which display a popup which can contain more controls. Yes, you can have a popup within a popup within a popup. This is why unique keys are useful and why setting up the control can be complex. See the <Group> tag for a list of available controls.

The last element of the ribbon control is the Backstage. The backstage is what appears when the user clicks the system button. The system button is either a round button in the top left corner of the ribbon or a blue tab at the beginning of the tab bar. The backstage fills up the entire form when visible. It has a side bar along the left which can be populated with either buttons or pages. Buttons and pages are both associated with commands. Buttons simply fire the OnCommand event when clicked. Pages display an embedded OI form on the right, which are defined by including the <Page> tag in the XML setup and calling the EmbedWindow method during the CREATE event of your form.