Segmented Controls Programming Guide for Cocoa

Cocoa > User Experience

2006-02-07

Contents

Figures and Listings

Introduction to Segmented Controls

This programming topic describes the use of NSSegmentedControl and NSSegmentedCell to implement a horizontal control made of multiple segments.

Note: NSSegmentedControl and NSSegmentedCell are only available in Mac OS X version 10.3 and later.

Organization of This Document

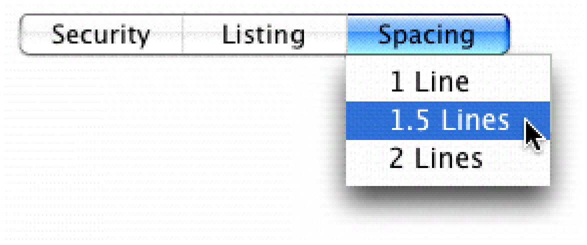
This programming contains the following articles:

- "Segmented Controls Overview" (page 9) provides basic information on segmented controls.
- "Using Segmented Controls" (page 11) shows how to configure and interact with segmented controls programatically.

Segmented Controls Overview

NSSegmentedControl is a subclass of `NSControl` that implements a horizontal control made of multiple segments. It provides a compact means of grouping together a number of controls, as shown in Figure 1.

Figure 1 An example of a segmented control



Segments may contain text, images, and menus. One of the simplest applications of a segmented control is its use as a compact alternative to a group of radio buttons. Here, the user makes a single selection from a number of options. In a more complex example, segments can be independently selected or have associated menus.

`NSSegmentedControl` has these features:

- Each segment can have an image, text (label), menu, tooltip, and tag.
- Either the whole control or individual segments can be enabled or disabled.
- There are three modes: radio button (as illustrated by Finder's view mode selection control), momentary (as illustrated by Safari's toolbar buttons), or any on/off.
- Each segment can either be a fixed width or be autosized to fit the contents.
- It provides full keyboard control of the user interface.

A segmented control's appearance and behavior are controlled by the class `NSSegmentedCell`. Most `NSSegmentedCell` methods have covers in `NSSegmentedControl`, which simply call the `NSSegmentedCell` equivalent. For more information, see the NSSegmentedCell and NSSegmentedControl class specifications.

Basic Configuration

A segmented control requires some basic configuration. You must specify its tracking mode—the way in which selections are handled; how many segments it contains; and, optionally, the width of each segment. Segmented controls support three different tracking modes:

- Only one segment can be selected
- Any segment can be selected

- Segments are selected only while tracking

You can configure a segmented control programatically or using Interface Builder. In Cocoa, the tracking mode is set using an `NSSegmentedCell` method. Since connections are usually made to the control that contains the cell, you typically have to access the cell first and then send the message, if you are setting the tracking mode programmatically.

Text Labels and Images

You can set a text label (title) or image independently for each segment. By default, each segment autosizes to fit its contents. If you want to ensure a fixed size—perhaps with each segment the same width—you must set the width of each segment individually. In this case it is up to you to ensure that each segment is wide enough for its contents. If a segment is too narrow for its contents, text is truncated and images are clipped.

Menus

A segment can have a menu; you can configure this menu programatically or in Interface Builder. You then set the menu for a segment programatically, using the `setMenu:forSegment:` method. You typically set a menu together with a label or an image. If you do not specify either, the segment remains blank, although the menu will appear when the user clicks on it.

Note: A segment with a menu always acts as a momentary control and cannot be toggled on, even if the control is configured as “select one” or “select any”.

Using Segmented Controls

You can create and configure a segmented control programatically or using Interface Builder. You can also interact with the control programatically, for example to set the selection, and to enable and disable segments.

Creating and Configuring a Segmented Control

There are a number of aspects of a segmented control that you should configure: its tracking mode—the way in which selections are handled; how many segments it contains; and the properties of each segment. Segmented controls support three different tracking modes, specified by the following constants:

`NSSegmentSwitchTrackingSelectOne`
: Only one button can be selected

`NSSegmentSwitchTrackingSelectAny`
: Any button can be selected

`NSSegmentSwitchTrackingMomentary`
: Segments are selected only while tracking

Configuring a Segmented Control in Interface Builder

The easiest way to create a segmented control is in Interface Builder. To do so, simply drag a segmented control from the Cocoa-Controls palette to your window. Use the Attributes panes of the Inspector window to configure the attributes of the segmented control. These are:

- **Size**. Specifies the control size. Like other controls, segmented controls come in three sizes: regular, small, and mini.
- **Selection**. Specifies the tracking mode of the segmented control. The possible options are:
  - **Single**. This corresponds to the constant`NSSegmentSwitchTrackingSelectOne` and specifies that only one segment can be selected.
  - **Multiple**. This corresponds to the constant `NSSegmentSwitchTrackingSelectAny` and specifies that any segment can be selected.
  - **Momentary**. This corresponds to the constant `NSSegmentSwitchTrackingMomentary` and indicates that segments are selected only while tracking.
- **Text Direction**. Specifies the direction of any label text. This defaults to Natural, indicating the text direction native to the user's current language settings.
- **Segments**. Specifies the number of segments in the segmented control.
- **Tag**. Specifies a tag identifying the segmented control.

- **Segment Widths**. Specifies the width of the individual segments in the segmented control. By default, segments are autosized to fit their contents. If you want to ensure a fixed width for each segment, you can specify that width here. You can type in a specific width or you can use the Avg and Max buttons to calculate the current average or maximum segment width, respectively, and set the fixed segment width to that number.

Certain attributes must be configured individually for each segment. To configure a segment, double-click to select it. The Inspector window for an `NSSegmentedCell` lets you configure these attributes:

- **Label**. Specifies the text label of the segment, if any.
- **Tag**. Specifies a tag identifying the individual segment.
- **Icon**. A file containing the image for the segment to display. A segment should have either an icon or a text label, but not both. See Controls in *Apple Human Interface Guidelines*.

Configuring a Segmented Control Programatically

You can also create a segmented control programmatically. One of the simplest applications of a segmented control is its use as a compact alternative to a group of radio buttons. Here, the user makes a single selection from a number of options. Listing 1 (page 12) shows code to create a control like one found in iPhoto.

Listing 1 Simple text-based example

```
// Set the number of segments
[segControl setSegmentCount:4];
// Set titles for each segment
[segControl setLabel:@"Import" forSegment:0];
[segControl setLabel:@"Organize" forSegment:1];
[segControl setLabel:@"Edit" forSegment:2];
[segControl setLabel:@"Book" forSegment:3];
```

By default a segmented control autosizes to fit. To programatically mark a segment as autosizing, set its width to `0`. If any segments are marked as autosizing, then the widths are adjusted so that all segments are displayed completely in the control.

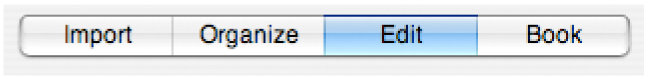
If you want to ensure a fixed size, perhaps with each segment the same width, you must set the width of each segment individually, as shown in Listing 2 (page 12).

Listing 2 Setting segments to equal widths

```
[segControl setWidth:70 forSegment:0];
[segControl setWidth:70 forSegment:1];
[segControl setWidth:70 forSegment:2];
[segControl setWidth:70 forSegment:3];
```

Combining code from Listing 1 (page 12) and Listing 2 (page 12) produces a controller like that shown in Figure 1 (page 12).

Figure 1 Control with segments of equal width



It is up to you to ensure that each segment is wide enough for its contents. If an image is too large to fit, it is clipped on the right. If the text, or the combination of the text and image, is too wide for a segment, the text is truncated using an ellipsis.

Some of the methods you may require for configuration (for example the tracking mode) are implemented only in `NSSegmentedCell`, so you must access the control's cell, as in the following example:

```
[[segControl cell] setTrackingMode:NSSegmentedSwitchTrackingSelectAny];
```

In common with other controls, `NSSegmentedControl` is available in three sizes—regular, small, and mini (although textured windows only support regular-sized segmented controls). The size is also configured by the cell:

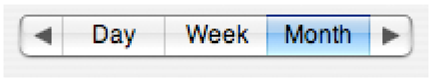
```
[[segControl cell] setControlSize:NSMiniControlSize];
```

Combining Segments With Different Attributes

There is no requirement that all segments within a segmented control be of the same type. The following code creates a segmented control like that found in iCal, as illustrated in Figure 2 (page 13):

```
[segControl setSegmentCount:5];
// Configure each segment
[segControl setImage:MYLeftArrowImage forSegment:0];
[segControl setLabel:@"Day" forSegment:1];
[segControl setLabel:@"Week" forSegment:2];
[segControl setLabel:@"Month" forSegment:3];
[segControl setImage:MYRightArrowImage forSegment:4];
```

Figure 2 Segmented control showing a combination of segment types



Note: In Interface Builder, you can edit the label and image for each segment independently, as described in "Configuring a Segmented Control in Interface Builder" (page 11).

It is not possible, however, to specify different tracking modes for different segments. You cannot specify momentary selection for some segments and radio button behavior for others. Thus, although the control in Figure 2 (page 13)*looks* like one found in iCal, it cannot function in the same way (that is, where the arrows are selected momentarily and advance through the calendar, and one of the text buttons remains continuously selected to choose the display option). The exception to this rule is a segment that contains a menu. A segment with a menu always acts as a momentary control and cannot be toggled on, even if the control is configured as "select one" or "select any".

Although it is possible to set both a label and an image for a segment, a segment should have one or the other, as described in *Apple Human Interface Guidelines*.

Note: In Mac OS X v10.4 and earlier, images are not automatically centered in the segment. To ensure that a segment's image is centered, use the following workaround:

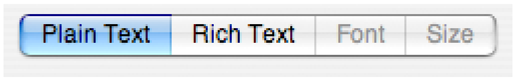
```
[segmentedControl setLabel: nil forSegment: n];
```

Interacting With a Segmented Control

There are a number of methods for interacting with a segmented control programatically—for example, to change the selected segment, and to enable and disable individual segments.

You can find out which segment, if any, is selected, using `selectedSegment`. If no segment is selected, the method returns `-1`. When a control is configured to allow multiple selections, you can find out if an individual segment is selected using `isSelectedForSegment:`. Conversely, you can set the selection using `setSelectedSegment:` and `setSelected:forSegment:`.

Similarly, you can enable and disable individual segments within the control using `setEnabled:forSegment:`. Doing so may be useful to restrict the user to making selections that are valid for the current environment, as illustrated below.



Tags and Target-Action

As with any control, you can set a segmented control's target, action, and tag. Each segment within the control, however, may have its own tag. When any segment in the control is clicked, the action message is sent to the target—unless the segment is disabled or has a menu. You can determine which segment was clicked by finding the tag for the selected segment, as shown in the following example:

```
- (void)awakeFromNib
{
    [segControl setSegmentCount:3];
    [[segControl cell] setTag:0 forSegment:0];
    [[segControl cell] setTag:1 forSegment:1];
    [[segControl cell] setTag:2 forSegment:2];
    [segControl setTarget:self];
    [segControl setAction:@selector(segControlClicked:)];
}

- (IBAction)segControlClicked:(id)sender
{
    int clickedSegment = [sender selectedSegment];
    int clickedSegmentTag = [[sender cell] tagForSegment:clickedSegment];
    //...
}
```

Note that the cell's `selectedSegment` method returns the segment that was last clicked. This means you can find out which segment was clicked, even when tracking mode is `NSSegmentSwitchTrackingSelectAny` or `NSSegmentSwitchTrackingMomentary`. Note also that if a segment contains a menu, the action message is not sent when it is clicked.

Document Revision History

This table describes the changes to *Segmented Controls Programming Guide for Cocoa*.

Date	**Notes**
2006-02-07	Added section on how to create a segmented control using Interface Builder. Changed title from Segmented Controls.
	Added "Configuring a Segmented Control in Interface Builder" (page 11).
	Clarified statement on configuring a segment's menu in "Menus" (page 10).
	Changed Figure 1 (page 9) so the example follows the guidelines in *Apple Human Interface Guidelines*.
2004-02-02	Clarified that textured window only support regular-sized segmented controls.
2003-10-15	First release of this document.