Class

NSControl

A specialized view, such as a button or text field, that notifies your app of relevant events using the target-action design pattern.

macOS

```
@MainActor
class NSControl
```

Overview

The NSControl class is abstract and must be subclassed to be used. Although you can subclass it yourself, more often you use one of the subclasses already defined by AppKit. A control draws content on the screen, automatically handles user interactions with that content, and calls the action method of its target object for any significant user interactions.

About delegate methods

The NSControl class provides several delegate methods for its subclasses that allow text editing, such as NSTextField and NSMatrix. These include: controlTextDidBeginEditing:, controlTextDidChange:, and controlTextDidEndEditing:.

Note that although NSControl defines delegate methods, it doesn't itself have a delegate. Any subclass that uses these methods must have a delegate and the methods to get and set it. In addition, a formal delegate protocol NSControlTextEditingDelegate also defines delegate methods used by control delegates.

Responding to mouse events

When the mouse button is pressed while the cursor is within the bounds of the receiver, the system calls `mouseDown(with:)`. This method highlights the receiver's cell and sends it a `trackMouse(with:in:of:untilMouseUp:)` message. Whenever the cell finishes tracking the mouse (for example, because the cursor has left the cell's bounds), the cell is unhighlighted. If the mouse button is still down and the cursor reenters the bounds, the cell is again highlighted and a new `trackMouse(with:in:of:untilMouseUp:)` message is sent. This behavior repeats until the mouse button goes up. If it goes up with the cursor in the control, the state of the control is changed, and the action message is sent to the target. If the mouse button goes up when the cursor is outside the control, no action message is sent.

Topics

Creating a Control

`init(frame: NSRect)`

Initializes a control with the specified frame rectangle.

`init?(coder: NSCoder)`

Initializes a control with data in an unarchiver.

Enabling and Disabling the Control

`var isEnabled: Bool`

A Boolean value that indicates whether the receiver reacts to mouse events.

Accessing the Control's Value

`var doubleValue: Double`

The value of the receiver's cell as a double-precision floating-point number.

`var floatValue: Float`

The value of the receiver's cell as a single-precision floating-point number.

`var intValue: Int32`

The value of the receiver's cell as an integer.

`var integerValue: Int`

The value of the receiver's cell as an integer value.

`var objectValue: Any?`

The value of the receiver’s cell as an Objective-C object.

`var stringValue: String`

The value of the receiver’s cell as an `NSString` object.

`var attributedStringValue: NSAttributedString`

The value of the receiver’s cell as an attributed string.

Interacting with Other Controls

`func takeDoubleValueFrom(Any?)`

Sets the value of the receiver’s cell to a double-precision floating-point value obtained from the specified object.

`func takeFloatValueFrom(Any?)`

Sets the value of the receiver’s cell to a single-precision floating-point value obtained from the specified object.

`func takeIntValueFrom(Any?)`

Sets the value of the receiver’s cell to an integer value obtained from the specified object.

`func takeIntegerValueFrom(Any?)`

Sets the value of the receiver’s cell to an `NSInteger` value obtained from the specified object.

`func takeObjectValueFrom(Any?)`

Sets the value of the receiver’s cell to the object value obtained from the specified object.

`func takeStringValueFrom(Any?)`

Sets the value of the receiver’s cell to the string value obtained from the specified object.

Formatting Text

`var alignment: NSTextAlignment`

The alignment mode of the text in the receiver’s cell.

`var font: NSFont?`

The font used to draw text in the receiver’s cell.

`var lineBreakMode: NSLineBreakMode`

The line break mode to use for text in the control’s cell.

```
var usesSingleLineMode: Bool
```

A Boolean value that indicates whether the text in the control's cell uses single line mode.

```
var formatter: Formatter?
```

The receiver's formatter.

```
var baseWritingDirection: NSWritingDirection
```

The initial writing direction used to determine the actual writing direction for text.

Managing Expansion Tool Tips

```
func draw(withExpansionFrame: NSRect, in: NSView)
```

Performs custom expansion tool tip drawing.

```
var allowsExpansionToolTips: Bool
```

A Boolean value that indicates whether expansion tool tips are shown when the control is hovered over.

```
func expansionFrame(withFrame: NSRect) -> NSRect
```

The frame in which a tool tip can be displayed, if needed.

Managing the Field Editor

```
func abortEditing() -> Bool
```

Terminates the current editing operation and discards any edited text.

```
func currentEditor() -> NSText?
```

Returns the current field editor for the control.

```
func validateEditing()
```

Validates changes to any user-typed text.

```
func edit(withFrame: NSRect, editor: NSText, delegate: Any?, event:
NSEvent)
```

Begins editing of the receiver's text using the specified field editor.

```
func endEditing(NSText)
```

Ends the editing of text in the receiver using the specified field editor.

```
func select(withFrame: NSRect, editor: NSText, delegate: Any?, start:
Int, length: Int)
```

Selects the specified text range in the receiver’s field editor.

Control-Editing Notifications

An NSControl object posts the following notifications to interested observers and its delegate. Note that although the NSControl class defines delegate methods, it doesn’t itself have a delegate. Any subclass that uses these methods must have a delegate and the methods to get and set it.

`class let textDidBeginEditingNotification: NSNotification.Name`

Sent when a control with editable cells begins an edit session.

`class let textDidChangeNotification: NSNotification.Name`

Sent when the text in the receiving control changes.

`class let textDidEndEditingNotification: NSNotification.Name`

Sent when a control with editable cells ends an editing session.

Resizing the Control

`var controlSize: NSControl.ControlSize`

The size of the control.

`enum ControlSize`

A constant for specifying a cell’s size.

`func sizeThatFits(NSSize) -> NSSize`

Asks the control to calculate and return the size that best fits the specified size.

`func sizeToFit()`

Resizes the receiver’s frame so that it’s the minimum size needed to contain its cell.

Displaying a Cell

`var isHighlighted: Bool`

A Boolean value that indicates whether the cell is highlighted.

`enum ImagePosition`

A constant for specifying the position of a button’s image relative to its title.

`struct StateValue`

A constant that indicates whether a control is on, off, or in a mixed state.

Implementing the Target-Action Mechanism

`var action: Selector?`

The default action-message selector associated with the control.

`var target: AnyObject?`

The target object that receives action messages from the cell.

`var isContinuous: Bool`

A Boolean value indicating whether the receiver’s cell sends its action message continuously to its target during mouse tracking.

`func sendAction(Selector?, to: Any?) -> Bool`

Causes the specified action to be sent to the target.

`func sendAction(on: NSEvent.EventTypeMask) -> Int`

Sets the conditions on which the receiver sends action messages to its target.

Accessing Tags

`var tag: Int`

The tag identifying the receiver (not the tag of the receiver’s cell).

Activating from the Keyboard

`func performClick(Any?)`

Simulates a single mouse click on the receiver.

`var refusesFirstResponder: Bool`

A Boolean value indicating whether the receiver refuses the first responder role.

Tracking the Mouse

`func mouseDown(with: NSEvent)`

Informs the receiver that the user has pressed the left mouse button.

`var ignoresMultiClick: Bool`

A Boolean value indicating whether the receiver ignores multiple clicks made in rapid succession.

Supporting Constraint-Based Layout

`func invalidateIntrinsicContentSize(for: NSCell)`

Notifies the control that the intrinsic content size for its cell is no longer valid.

Deprecated

Deprecated Symbols

Review unsupported symbols and their replacements.

Relationships

Inherits From

`NSView`

Inherited By

`NSBrowser`
`NSButton`
`NSColorWell`
`NSComboButton`
`NSDatePicker`
`NSImageView`
`NSLevelIndicator`
`NSMatrix`
`NSPathControl`
`NSRuleEditor`
`NSScroller`
`NSSegmentedControl`
`NSSlider`
`NSStepper`
`NSSwitch`
`NSTableView`
`NSTextField`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSAccessibilityElementProtocol`
`NSAccessibilityProtocol`
`NSAnimatablePropertyContainer`
`NSAppearanceCustomization`
`NSCoding`
`NSDraggingDestination`
`NSObjectProtocol`
`NSStandardKeyBindingResponding`
`NSTouchBarProvider`
`NSUserActivityRestoring`
`NSUserInterfaceItemIdentification`
`Sendable`
`SendableMetatype`

See Also

View fundamentals

`class NSView`

The infrastructure for drawing, printing, and handling events in an app.

`class NSCell`

A mechanism for displaying text or images in a view object without the overhead of a full `NSView` subclass.

`class NSActionCell`

An active area inside a control.