NSPageLayout

Inherits From: NSObject

Conforms To: NSObject (NSObject)

Declared In: AppKit/NSPageLayout.h

Note: On Mach platforms, NSPageLayout inherits from NSPanel and conforms to NSCoding.

Class Description

NSPageLayout is a panel that queries the user for information such as paper type and orientation. This information is stored in an NSPrintInfo object, and is later used when printing. The NSPageLayout panel is created, displayed, and run (in a modal loop) when a **runPageLayout:** message is sent to the NSApplication object. By default, this message is sent up the responder chain when the user chooses the Page Setup menu item (on Mach platforms the menu item is called Page Layout).

Typically, you access an NSPageLayout panel by invoking the **pageLayout** method. When the class receives a **pageLayout** message, it returns an existing panel rather than create a new one. If a panel is reused, its attributes are reset to the default values so that the effect is the same as receiving a new panel. Because an NSPageLayout object may be reused, you shouldn't modify the instance returned by **pageLayout**, except using the methods below. If you must modify an NSPageLayout object in other ways than those allowed by its methods, create and manage your own instance using the **alloc...** and **init...** methods rather than modify the object returned by **pageLayout**.

In most cases it is unnecessary to subclass NSPageLayout—you can customize an NSPageLayout by specifying your own accessory view. You can add your own controls to an NSPageLayout through the **setAccessoryView:** method. The panel is automatically resized to accommodate the NSView that you've added. Note that NSPageLayout does not have accessor methods to obtain the state of its controls. If controls you add through an accessory view need to know the values of existing controls (or vice versa) use the **viewWithTag:** method. You obtain a specific control object by sending **viewWithTag:** to the NSPageLayout object passing one of the tags defined in AppKit/NSPageLayout.h. The value can then be obtained by sending an appropriate accessor message to the returned control object.

Method Types

Creating an NSPageLayout	+ pageLayout
Running an NSPageLayout	– runModal – runModalWithPrintInfo:

Customizing an NSPageLayout	– accessoryView – setAccessoryView:
Accessing the NSPrintInfo	– printInfo – readPrintInfo – writePrintInfo
Updating the display	– convertOldFactor:newFactor: – pickedButton: – pickedOrientation: – pickedPaperSize: – pickedUnits:

Class Methods

pageLayout

+ (NSPageLayout *)**pageLayout**

Returns a shared NSPageLayout object or a newly created one if it doesn't already exist.

Instance Methods

accessoryView

– (NSView *)**accessoryView**

Returns the receiver's accessory view (used to customize the receiver).

See also: – **setAccessoryView:**

convertOldFactor:newFactor:

– (void)**convertOldFactor:**(float *)*old* **newFactor:**(float *)*new*

This method is for Mach platforms only—it is not defined for other platforms. The standard unit used to measure a paper's dimensions is a point (for example, NSPrintInfo defines a paper's size in points). However, the user can select a different unit of measurement from the NSPageLayout panel. Use this method to get the ratio between a point and the currently selected unit of measurement. Unless this method is invoked by **pickedUnits:** both *old* and *new* will be set to the same ratio value.

The **pickedUnits:** method is invoked when the user selects a new unit of measurement from the NSPageLayout panel. Subclasses should override the **pickedUnits:** method to update any controls, located on the accessory view, that display dimensional values. Use this method to get the old and new ratios. See **pickedUnits:** for details.

pickedButton:

– (void)**pickedButton:**(id)*sender*

This method is for Mach platforms only—it is not defined for other platforms. Invoked when either the OK or Cancel buttons are clicked, and stops the receiver's modal loop. If the OK button was clicked, this method verifies that the height, width and scale entries are acceptable (they must hold positive numbers). If not, the unacceptable entry is selected and the panel isn't stopped. Subclasses should override this method to verify that the controls on the accessory view contain acceptable values.

See also: – **pickedOrientation:**, – **pickedPaperSize:**, – **pickedUnits:**

pickedOrientation:

– (void)**pickedOrientation:**(id)*sender*

This method is for Mach platforms only—it is not defined for other platforms. Invoked when the user selects a page orientation (i.e., protrait or landscape). This method updates the height and width fields, and redraws the paper view.

See also: – pickedButton:, – pickedPaperSize:, – pickedUnits:

pickedPaperSize:

– (void)**pickedPaperSize:**(id)*sender*

This method is for Mach platforms only—it is not defined for other platforms. Invoked when the user selects a paper size from the paper size list. Updates the height and width fields, redraws the paper view, and may switch the protrait/landscape orientation.

See also: – **pickedButton:**, – **pickedOrientation:**, – **pickedUnits:**

pickedUnits:

– (void)**pickedUnits:**(id)*sender*

This method is for Mach platforms only—it is not defined for other platforms. Invoked when the user selects a new unit of measurement from the Units list. The height and width fields are updated.

Subclasses should override this method to update controls in the accessory view that contain unit values. The ratios returned by the **convertOldFactor:newFactor:** method should be used to calculate the new values as shown below, where **myField** is an NSTextField located on the accessory view that needs to be updated:

```
- pickedUnitsLsender
{
```

```
    float old, new;

    /* At this point the units have been selected but not set. */
    [self convertOldFactor:&old newFactor:&new];

    /* Update myField based on the conversion factors. */
    [myField setFloatValue:([myField floatValue]*new/old)];

    /* Set the selected units. */
    return [super pickedUnits:sender];
}
```

See also: – **pickedButton:**, – **pickedOrientation:**, – **pickedPaperSize:**

printInfo

– (NSPrintInfo *)**printInfo**

Returns the NSPrintInfo object that is used when the receiver is run (set using the **runModal** or **runModalWithPrintInfo:** methods).

See also: – **readPrintInfo**, – **writePrintInfo**

readPrintInfo

– (void)**readPrintInfo**

Sets the receiver's values to those stored in the NSPrintInfo object used when the receiver is run.

See also: – **printInfo**, – **writePrintInfo**, – **runModal**, – **runModalWithPrintInfo:**

runModal

– (int)**runModal**

Displays the receiver and begins the modal loop. The receiver's values are recorded in the shared NSPrintInfo object. Returns NSCancelButton if the user clicks the Cancel button, otherwise returns NSOKButton.

See also: – **pickedButton:**, – **runModalWithPrintInfo:**

runModalWithPrintInfo:

– (int)**runModalWithPrintInfo:**(NSPrintInfo *)*printInfo*

Displays the receiver and begins the modal loop. The receiver's values are recorded in *printInfo*. Returns NSCancelButton if the user clicks the Cancel button, otherwise returns NSOKButton.

See also: – **pickedButton:**, – **runModal:**

setAccessoryView:

– (void)**setAccessoryView:**(NSView *)*aView*

Adds an NSView to the receiver. Invoke this method to add a custom view containing your controls. The receiver is automatically resized to accommodate *aView*. This method can be invoked repeatedly to change the accessory view depending on the situation. If *aView* is **nil**, then the receiver's current accessory view, if any, is removed.

See also: – **accessoryView**

writePrintInfo

– (void)**writePrintInfo**

Writes the receiver's values to the NSPrintInfo object used when the receiver is run.

See also: – **printInfo**, – **readPrintInfo**, – **runModal**, – **runModalWithPrintInfo:**