IBInspector Class Reference

Tools > Interface Builder

2007-07-11

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

IBInspector Class Reference

Inherits from	NSObject
Conforms to	`NSObject` (`NSObject`)
Framework	/System/Library/Frameworks/InterfaceBuilderKit.framework
Declared in	InterfaceBuilderKit/IBInspector.h
Companion guide	Interface Builder Plug-In Programming Guide

Overview

The `IBInspector` class manages the display and synchronization of custom attributes for a particular class in the inspector window. If you are implementing a custom view, control, or object whose class has user-editable attributes, you should provide a custom subclass of `IBInspector`.

Subclassing Notes

An `IBInspector` object is a controller that manages a custom user interface you define. The main job of your inspector object is to synchronize the changes made in its user interface with the data values in the underlying objects and vice versa. Each inspector object you create manages the interface for a single slice in the attributes pane of the inspector window. If your custom objects derive from multiple custom parent classes, you would typically implement a separate inspector object for each class.

To manage the custom user interface for an inspector, you can use bindings or outlets and actions. When using bindings, the synchronization is automatic. When using outlets and actions, the inspector object should push changes from inspector controls to the current objects being inspected. Conversely, when the selection itself changes, Interface Builder sends your inspector a `refresh` message, which your inspector object can use to update its interface.

For more information on creating a custom inspector object, see *Interface Builder Plug-In Programming Guide*.

Methods to Override

To implement an inspector, you typically override the following methods:

- `viewNibName` (page 9)
- `refresh` (page 8)

Overriding the `refresh` method is necessary if you are using actions and outlets to synchronize your inspector interface or if you need to perform some additional actions in your inspector whenever the selection changes. If you use Cocoa bindings exclusively to synchronize your inspector user interface with the current selection, you do not need to override the `refresh` method.

You may also want to override the `label` method to provide a custom label for your inspector slice. If you do not override this method, Interface Builder provides a reasonable default.

Tasks

Getting the Shared Inspector Object

`+ sharedInstance` (page 7)
Returns the shared instance of your custom inspector object.

Getting the Inspector Attributes

`+ supportsMultipleObjectInspection` (page 7)
Returns a Boolean value indicating whether the receiver supports the selection of multiple objects.

`- document` (page 7)
Returns the document object that contains the currently inspected objects.

`- label` (page 8)
Returns the user-readable string to display in your inspector slice.

`- view` (page 9)
Returns the content view of the receiver.

`- viewNibName` (page 9)
Returns the name of the nib file containing the receiver's content view.

Getting the Inspected Objects

`- inspectedObjects` (page 8)
Returns the currently selected objects that should be inspected.

`- inspectedObjectsController` (page 8)
Returns an array controller you can use to bind to the inspected objects.

Updating the Inspector View

`- refresh` (page 8)
Notifies the receiver that some aspect of the selection changed.

Class Methods

sharedInstance

Returns the shared instance of your custom inspector object.

```
+ (id)sharedInstance
```

Return Value
Your shared inspector object.

Discussion
You do not need to override this method. The first time this method is called, Interface Builder creates an instance of your inspector object and caches it for future access.

supportsMultipleObjectInspection

Returns a Boolean value indicating whether the receiver supports the selection of multiple objects.

```
+ (BOOL)supportsMultipleObjectInspection
```

Return Value
`YES` if the receiver supports multiple selections; otherwise, `NO`. The default implementation of this method returns `YES`.

Discussion
Multiple selection applies only when the selected objects share a common class. Whenever possible, you should design your inspector's user interface to display appropriate information when multiple objects are selected. If for some reason, your inspector cannot support multiple selection, you should override this method and return `NO`.

Instance Methods

document

Returns the document object that contains the currently inspected objects.

```
- (IBDocument *)document
```

Return Value
The active document.

Discussion
The current document maintains information that might be useful to your inspector, such as the objects currently in the nib file and the relationships between those objects.

You should not override this method.

inspectedObjects

Returns the currently selected objects that should be inspected.

```
- (NSArray *)inspectedObjects
```

Return Value
The currently selected objects, or an empty set if no objects are selected.

Discussion
This method returns always returns the current selection, regardless of whether that selection contains zero, one, or multiple objects.

See Also
- `inspectedObjectsController` (page 8)

inspectedObjectsController

Returns an array controller you can use to bind to the inspected objects.

```
- (NSArrayController *)inspectedObjectsController
```

Return Value
The array controller for the selected objects.

See Also
- `inspectedObjects` (page 8)

label

Returns the user-readable string to display in your inspector slice.

```
- (NSString *)label
```

Return Value
A string describing the receiver. By default, this method returns a formatted version of the receiver's class name.

Discussion
You can override this method to return a custom label for your inspector.

refresh

Notifies the receiver that some aspect of the selection changed.

```
- (void)refresh
```

Discussion
You should override this method in your inspector classes if you want to synchronize your inspector's content view manually with the values in the currently selected objects. In your implementation of this method, you should get the currently selected objects and use their current values to update the controls in the receiver's content view. You do not need to implement this method if you synchronize your inspector contents exclusively using Cocoa bindings.

If you override this method, you must call `super` at some point in your implementation.

view

Returns the content view of the receiver.

```
- (NSView *)view
```

Return Value
The content view containing the inspector's visible content. This view typically contains controls used to display the attributes of the currently inspected object.

Discussion
You do not need to override this method if your view is connected to the `inspectorView` outlet of your inspector class. (This outlet is exposed by the `IBInspector` class.) If your view is connected to that outlet, this method returns your view automatically. If you do not connect your view to that outlet, you must override this method and return your view object.

See Also
- `viewNibName` (page 9)

viewNibName

Returns the name of the nib file containing the receiver's content view.

```
- (NSString *)viewNibName
```

Return Value
A string identifying the receiver's nib file. This string should not include the `.nib` extension or any pathname information. Interface Builder looks for the specified nib file in the `Resources` directory of the plug-in bundle.

Discussion
You should override this method to return the name of the nib file containing your inspector's content view. If your inspector uses a single content view, you should set the File's Owner of the nib file to your custom `IBInspector` subclass and connect the `inspectorView` outlet of that class to your view. If your inspector supports multiple views, you should use a single master view and swap out its contents as needed when the inspector is refreshed.

If you prefer not to use a nib file to load your view, you can return `nil` from this method. If you do so, however, you must override the `view` method to return your view.

See Also
- `refresh` (page 8)
- `view` (page 9)

Document Revision History

This table describes the changes to *IBInspector Class Reference*.

Date	Notes
2007-07-11	New document describing the methods for managing an inspector view.

Document Revision History

Index