IBPlugin Class Reference

Tools > Interface Builder

2007-04-02

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

Simultaneously published in the United States and Canada.

Contents

IBPlugin Class Reference

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

Overview

The `IBPlugin` class provides the basic functionality required by all Interface Builder plug-ins. The methods of this class provide hooks for you to configure your plug-in when it is loaded by Interface Builder.

Subclassing Notes

Interface Builder uses this class as the entry point to your plug-in bundle. You must implement a custom subclass that provides basic information about your plug-in to Interface Builder.

Methods to Override

There is only one method you are required to override in this class:

- `libraryNibNames` (page 8)

You may override other methods of this class to support the initialization and customization of your plug-in object, but doing so is optional.

Tasks

Getting the Shared Plug-in Object

`+ sharedInstance` (page 6)
Returns the shared plugin object.

Loading and Unloading Plug-in Resources

- `didLoad` (page 7)
  Notifies the receiver that it has been loaded into the Interface Builder environment.
- `willUnload` (page 9)
  Notifies your plug-in that it has been removed from the Interface Builder environment.

Getting the Plug-in's Custom Objects

- `libraryNibNames` (page 8)
  Returns an array of strings containing the names of your plug-in's custom nib files.

Configuring Your Plug-in

- `label` (page 7)
  Returns the user-readable name displayed for your plug-in object in the Interface Builder preferences window.
- `preferencesView` (page 8)
  Returns the custom view used to display your plug-in's preferences.
- `requiredFrameworks` (page 9)
  Returns the framework bundles required by your plug-in code to operate.

Pasteboard Notifications

- `pasteboardObjectsForDraggedLibraryView:` (page 8)
  Notifies the receiver that one of its library items is about to be added to a document.
- `document:didAddDraggedObjects:fromDraggedLibraryView:` (page 7)
  Notifies the receiver that one or more objects were added to the specified document from the library.

Class Methods

sharedInstance

Returns the shared plugin object.

```
+ (id)sharedInstance
```

Return Value
The shared `IBPlugin` object.

Discussion
You do not need to override this method. You may call it at any time from your plug-in code to retrieve your own plug-in object.

Instance Methods

didLoad

Notifies the receiver that it has been loaded into the Interface Builder environment.

```
- (void)didLoad
```

Discussion
You can override this method to initialize any variables or resources that your plug-in requires whenever it is loaded by Interface Builder. If you do override this method, you must call `super` at some point in your implementation.

If you implement this method, you should not rely on Interface Builder calling the `willUnload` method to undo your plug-in object's initialization. If your `didLoad` method acquires any resources that must be freed later, you should release those resources in your plug-in object's `dealloc` or `finalize` method instead.

See Also
- `willUnload` (page 9)

document:didAddDraggedObjects:fromDraggedLibraryView:

Notifies the receiver that one or more objects were added to the specified document from the library.

```
- (void)document:(IBDocument *) documentdidAddDraggedObjects:(NSArray *)
    rootsfromDraggedLibraryView:(NSView *) view
```

Parameters

document
The document that received the specified objects.

roots
The root objects that were added to the document. This array does not contain any child objects attached to the root objects.

view
The library view (owned by the receiver) from which the objects were added.

Discussion
You can use this method to create additional associations, connections, or bindings among a group of objects after they are added to a document. For example, you might use this method to create connections that are not present in the library view version of the objects.

label

Returns the user-readable name displayed for your plug-in object in the Interface Builder preferences window.

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

Return Value
A string containing the user-readable name of your plug-in.

Discussion
If you do not provide a name for your plug-in, the default implementation returns a formatted version of the receiver's class name by default.

libraryNibNames

Returns an array of strings containing the names of your plug-in's custom nib files.

```
- (NSArray *)libraryNibNames
```

Return Value
An array of `NSString` objects, each of which corresponds to the name of a nib file in your plug-in's `Resources` directory. The nib file name does not require any leading path information or the `.nib` filename extension.

Discussion
The nib files you return should contain one or more library-object template views. These views act as containers for your own custom objects. Each library-object template view contains one or more objects to add the library window. For information on how to configure these views in your nib file, see *Interface Builder Plug-In Programming Guide*.

You must override this method in your plug-in subclass.

pasteboardObjectsForDraggedLibraryView:

Notifies the receiver that one of its library items is about to be added to a document.

```
- (NSArray *)pasteboardObjectsForDraggedLibraryView:(NSView *) view
```

Parameters

view

The view that was dragged from the library.

Return Value
The array of objects that should actually be added to the document. Only the root object of a given object hierarchy need be returned; you do not need to include any associated child objects of that root object in the array.

Discussion
You can use this method to replace a library view with the actual objects that the view represents. For example, you might use an image view in the library to provide a better visual representation of the underlying objects. In such a case, though, you would not want the image view to be added to a document. Instead, you would implement this method and use it to exchange the image view for the actual objects.

If you do not implement this method, the default version returns an array containing the object in the *view* parameter.

preferencesView

Returns the custom view used to display your plug-in's preferences.

```
- (NSView *)preferencesView
```

Return Value
The plug-ins custom preferences view.

Discussion
If your plug-in supports configurable preferences, you can override this method to return the view used to display those preferences. When your plug-in is selected in the Interface Builder preferences window, your custom view replaces the list of plug-ins and frameworks normally displayed for plug-ins.

requiredFrameworks

Returns the framework bundles required by your plug-in code to operate.

```
- (NSArray *)requiredFrameworks
```

Return Value
An array of `NSBundle` objects, each of which is initialized to a framework directory.

Discussion
Interface Builder uses the information returned from this method to load your plug-in's required frameworks during simulation. If your plug-in requires any custom frameworks, such as those containing your custom view code, you must override this method and return those frameworks.

Your implementation of this method should return only those frameworks containing your custom object code. You do not need to return any system frameworks your objects depend on.

willUnload

Notifies your plug-in that it has been removed from the Interface Builder environment.

```
- (void)willUnload
```

Discussion
You can override this method to handle any cleanup that might be required when your plug-in is removed from Interface Builder by the user. This method is called only when the user removes your plug-in from the list of plug-ins in the preferences window. It is not called when the Interface Builder application quits. If you do override this method, you must call `super` at some point in your implementation.

See Also
- `didLoad` (page 7)

Document Revision History

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

Date	Notes
2007-04-02	New document describing the base interface for managing custom plug-ins.

Document Revision History

Index