Workspace Services Programming Topics

Cocoa > Interapplication Communication

2007-03-06

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

Simultaneously published in the United States and Canada.

Contents

Listings

Introduction to Workspace Services

The NSWorkspace class provides a variety of services to Cocoa applications, including retrieving information from the file system, launching applications, and miscellaneous other tasks.

Some of NSWorkspace's methods are currently unimplemented:

- `noteUserDefaultsChanged`
- `checkForRemovableMedia`
- `extendPowerOffBy:`

Additionally, the compression and encryption file operations defined for `performFileOperation:source:destination:files:tag:` are currently unavailable.

Organization of This Document

This programming topic contains the following articles:

- "About Workspace Services" (page 9) contains a discussion of NSWorkspace's services.
- "Use of .app Extension" (page 11) describes how Mac OS X uses the `.app` extension to identify applications.
- "Retrieving Information About Files" (page 13) describes how to find file types, application information, and retrieve icons.
- "Manipulating Files" (page 15) describes how to perform Finder-like operations on files.
- "Manipulating Applications" (page 17) describes how to manipulate, locate, and hide applications.
- "Manipulating Devices" (page 19) describes how to track available storage devices and their associated volumes.
- "Tracking Workspace Changes" (page 21) describes how to inform the file system of changes.
- "Receiving Workspace Notifications" (page 23) describes how to use the notification center provided by NSWorkspace to receive workspace notifications.

About Workspace Services

The NSWorkspace class provides an interface between Cocoa applications and the Mac OS X "workspace," which consists primarily of the services provided by the Finder.

NSWorkspace provides access to services for files, applications, devices, user defaults, and a few other system features. Each application has one shared instance of NSWorkspace, which you access through the `sharedWorkspace` method.

NSWorkspace also provides notifications related to its services. Unlike most notifications, all NSWorkspace notifications are posted to NSWorkspace's own notification center instead of the application's default notification center.

Use of .app Extension

One of the ways Mac OS X determines if a package is an application is through the use of file extensions. The rules to determine if a package is an application are:

- On all file systems: the presence of a `.app` suffix
- On HFS+ only: the `.app` suffix is optional, if the package bit is set and the folder contains a new style `info.xml`.

Retrieving Information About Files

This document explains how to use NSWorkspace to retrieve information about files.

Finding File Types and Applications

To retrieve the file type and what application opens a file, use NSWorkspace's `getInfoForFile:application:type:` method. The string passed to the method must be the full pathname of the desired file. This code fragment retrieves the application and type for the file at `fullPath`:

```
NSString *fullPath;    // Assume this exists.
NSString *theApplication;
NSString *theType;

[[NSWorkspace sharedWorkspace] getInfoForFile:fullPath
                               application:&theApplication
                               type:&theType];
[theApplication retain];
[theType retain];
```

To retrieve other file information, use the NSFileManager methods `displayNameAtPath:`, `fileExtensionHidden`, `fileHFSCreatorCode`, `fileHFSTypeCode`, and `fileAttributesAtPath:traverseLink:`.

To retrieve the full pathname for an application, use the `fullPathForApplication:` method, available in both languages. The provided application name can either include or omit the `.app` extension.

To find out if a pathname points to a file package, use `the isFilePackageAtPath:` method.

Retrieving Icons

The methods `iconForFile:` and `iconForFiles:` retrieve the icon for a file or the icons for an NSArray of files. Files should be specified with full pathnames. The `iconForFileType:` method provides the icon for a given file extension or encoded HFS file type. This code fragment retrieves the icon for the file at `fullPath`, and resizes it to full 128 pixels by 128 pixels resolution:

```
NSString *fullPath;    // Assume this exists.
NSImage *theIcon;

theIcon = [[[NSWorkspace sharedWorkspace] iconForFile:fullPath] retain];
[theIcon setSize:NSMakeSize(128.0,128.0)];
```

To retrieve a generic icon, call the `NSFileTypeForHFSTypeCode` function with one of the icon constants defined by Icon Services in the Carbon framework (see "Standard Finder Icon Constants" in *Icon Services and Utilities Reference*), then use the `iconForFileType:` method with the result. This code fragment retrieves the generic application icon at full size:

```
#import <Carbon/Carbon.h>
// ...
NSImage *theIcon;

theIcon = [[[NSWorkspace sharedWorkspace]
              iconForFileType:
              NSFileTypeForHFSTypeCode(kGenericApplicationIcon)] retain];
[theIcon setSize:NSMakeSize(128.0,128.0)];
```

Manipulating Files

This task explains how to perform Finder-like operations on files using the NSWorkspace class.

Opening and Showing Files

NSWorkspace provides several methods for opening files:

- To open a file with default behavior, as if the user had opened it from the Finder, use the `openFile:` method.
- To open the file with a specific application, use `openFile:withApplication:`.
- To open the file with a specific application and specify if the current application should deactivate (allowing the new application to become active), use `openFile:withApplication:andDeactivate:`.

To show a file in the Finder, use the `selectFile:inFileViewerRootedAtPath:` method.

Opening URLs

To open a URL with the default handler for the resource type, use the `openURL:` method. The URL can be either local or remote. For example, a local files are opened as if double-clicked in the Finder, and a web addresses are opened in the default web browser.

Performing File Operations

The NSWorkspace method `performFileOperation:source:destination:files:tag:` performs various file system operations on files, such as moving and copying. The following Objective-C code fragment shows how to copy a file at `fullPath` from `source` to `destination`:

```
int tag;
BOOL succeeded;
NSString *source, *destination, *fullPath;    // Assume these exist
NSWorkspace *workspace = [NSWorkspace sharedWorkspace];
NSArray *files = [NSArray arrayWithObject:fullPath];

succeeded = [workspace performFileOperation:NSWorkspaceCopyOperation
                     source:source destination:destination
                     files:files tag:&tag];
```

In this code fragment, on return `succeeded` contains `YES` if the operation succeeded, `NO` otherwise. Also, the method sets `tag` to a negative integer if the operation fails, `0` if the operation is performed synchronously and succeeds, and a positive integer if the operation is performed asynchronously and succeeds.

Manipulating Applications

This task explains how to use NSWorkspace to manipulate applications. For information on how to use the `.app` extension, see "Use of .app Extension" (page 11).

Manipulating Applications

The NSWorkspace methods `launchApplication:` and `launchApplication:showIcon:autoLaunch:` launch applications using Launch Services.. The application name can include or omit the `.app` extension.

Locating Applications

The `fullPathForApplication:` method returns the full path for an application, specified with or without the `.app` extension.

Hiding Applications

To hide all other applications, you can use the `hideOtherApplications` method. Since the user usually has access to this functionality through other means, you should rarely have to invoke this method.

Manipulating Devices

This task explains how to track available storage devices and their associated volumes.

Tracking Available Devices

NSWorkspace provides several methods for tracking the status of storage devices:

- To retrieve the names of local mounted volumes, use the `mountedRemovableMedia` and `mountedLocalVolumePaths` methods.
- To wait until new removable devices have been mounted and then retrieve their pathnames, use the `mountNewRemovableMedia` method.

Also, to retrieve the Finder display names for volumes, use NSFileManager's `displayNameAtPath:` method.

Tracking Workspace Changes

This task explains how to inform the file system of changes using NSWorkspace.

Tracking File System Changes

If you create a file directly, use the `noteFileSystemChanged` method to inform NSWorkspace that it needs to update itself. The `noteFileSystemChanged:` method informs NSWorkspace that the file system at a specific path has changed.

NSDocument and NSSavePanel call `noteFileSystemChanged` automatically, so you don't need to use the NSWorkspace methods if you save a file using either of these classes.

Receiving Workspace Notifications

Workspace notifications are posted when:

- applications are launched and terminated
- volumes are mounted or unmounted
- the Finder performs file operations
- the Finder becomes the active application or resigns as the active application
- the user logs out or shuts down the computer
- the computer wakes from sleep.

Receiving Workspace Notifications

NSWorkspace notifications are posted to a notification center provided by the NSWorkspace object, instead of going through the application's default notification center as most notifications do. To receive NSWorkspace notifications, your application must register an observer with the NSWorkspace notification center, returned by the `notificationCenter` method.

The code fragment in Listing 1 registers a method `observerMethod` with the NSWorkspace notification center to receive all NSWorkspace notifications:

Listing 1 Registering for workspace notifications

```
NSNotificationCenter *notCenter;
SEL observerMethodSEL;

// Assume -observerMethod:(id)aNotification exists
observerMethodSEL = @selector(observerMethod:);
notCenter = [[NSWorkspace sharedWorkspace] notificationCenter];
[notCenter addObserver:self selector:observerMethodSEL
           name:nil object:nil]; // Register for all notifications
```

Document Revision History

This table describes the changes to *Workspace Services Programming Topics*.

Date	Notes
2007-03-06	Corrected minor typos.
2004-06-28	Reorganized introduction. Updated reference from `<HIToolbox/Icons.h>` to *Icon Services and Utilities Reference* in "Retrieving Information About Files" (page 13)
2002-11-12	Revision history was added to existing topic.