NSBundle Class Reference

Cocoa > Resource Management

2007-07-19

Contents

NSBundle Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSBundle.h
Companion guides	Bundle Programming Guide Resource Programming Guide
Related sample code	CoreRecipes GLSLShowpiece NumberInput_IMKit_Sample Quartz Composer WWDC 2005 TextEdit TextEditPlus

Overview

An `NSBundle` object represents a location in the file system that groups code and resources that can be used in a program. `NSBundle` objects locate program resources, dynamically load and unload executable code, and assist in localization. You build a bundle in Xcode using one of these project types: Application, Framework, Loadable Bundle, Palette.

See also NSBundle Additions in the Application Kit framework, which defines methods for loading nib files and locating image resources.

Unlike some other Foundation classes with corresponding Core Foundation names (such as `NSString` and CFString), `NSBundle` objects cannot be cast ("toll-free bridged") to CFBundle references. If you need functionality provided in CFBundle, you can still create a CFBundle and use the CFBundle API. See Interchangeable Data Types for more information on toll-free bridging.

Tasks

Initializing an NSBundle

`– initWithPath:` (page 18)
: Returns an `NSBundle` object initialized to correspond to a given directory.

Getting an NSBundle

`+ bundleForClass:` (page 9)
: Returns the `NSBundle` object with which a given class is associated.

`+ bundleWithIdentifier:` (page 10)
: Returns the previously created `NSBundle` instance that has a given bundle identifier.

`+ bundleWithPath:` (page 11)
: Returns an `NSBundle` object that corresponds to the specified directory.

`+ mainBundle` (page 11)
: Returns the `NSBundle` object that corresponds to the directory where the current application executable is located.

`+ allBundles` (page 9)
: Returns an array of all the application's non-framework bundles.

`+ allFrameworks` (page 9)
: Returns an array of all of the application's bundles that represent frameworks.

Getting a Bundled Class

`– classNamed:` (page 16)
: Returns the `Class` object for the specified name.

`– principalClass` (page 29)
: Returns the receiver's principal class.

Finding a Resource

`+ pathForResource:ofType:inDirectory:` (page 12)
: Returns the full pathname for the resource file identified by a given name and extension and residing in a given bundle directory.

`– pathForResource:ofType:` (page 24)
: Returns the full pathname for the resource identified by a given name and specified file extension.

`– pathForResource:ofType:inDirectory:` (page 25)
: Returns the full pathname for the resource identified by the given name and file extension and located in the specified bundle subdirectory.

- `pathForResource:ofType:inDirectory:forLocalization:` (page 26)
  Returns the full pathname for the resource identified by the given name and file extension, located in the specified bundle subdirectory, and limited to global resources and those associated with the specified localization.

+ `pathsForResourcesOfType:inDirectory:` (page 13)
  Returns an array containing the pathnames for all bundle resources having a given extension and residing in the bundle directory specified by a given path.

- `pathsForResourcesOfType:inDirectory:` (page 27)
  Returns an array containing the pathnames for all bundle resources having the specified filename extension and residing in the resource subdirectory.

- `pathsForResourcesOfType:inDirectory:forLocalization:` (page 28)
  Returns an array containing the pathnames for all bundle resources having the specified filename extension, residing in the specified resource subdirectory, and limited to global resources and those associated with the specified localization.

- `resourcePath` (page 31)
  Returns the full pathname of the receiving bundle's subdirectory containing resources.

Getting the Bundle Directory

- `bundlePath` (page 16)
  Returns the full pathname of the receiver's bundle directory.

Getting Bundle Information

- `builtInPlugInsPath` (page 15)
  Returns the full pathname of the receiver's subdirectory containing plug-ins.

- `bundleIdentifier` (page 15)
  Returns the receiver's bundle identifier.

- `executablePath` (page 17)
  Returns the full pathname of the receiver's executable file.

- `infoDictionary` (page 18)
  Returns a dictionary that contains information about the receiver.

- `objectForInfoDictionaryKey:` (page 23)
  Returns the value associated with a given key in the receiver's property list.

- `pathForAuxiliaryExecutable:` (page 24)
  Returns the full pathname of the executable with a given name in the receiver's bundle.

- `privateFrameworksPath` (page 30)
  Returns the full pathname of the receiver's subdirectory containing private frameworks.

- `sharedFrameworksPath` (page 31)
  Returns the full pathname of the receiver's subdirectory containing shared frameworks.

- `sharedSupportPath` (page 32)
  Returns the full pathname of the receiver's subdirectory containing shared support files.

Managing Localized Resources

`– localizedStringForKey:value:table:` (page 22)
: Returns a localized version of the string designated by a given key in a given table.

Loading a Bundle's Code

`– executableArchitectures` (page 17)
: Returns an array of numbers indicating the architecture types supported by the bundle's executable.

`– preflightAndReturnError:` (page 29)
: Returns a Boolean value indicating whether the bundle's executable code could be loaded successfully.

`– load` (page 19)
: Dynamically loads the bundle's executable code into a running program, if the code has not already been loaded.

`– loadAndReturnError:` (page 20)
: Loads the bundle's executable code and returns any errors.

`– isLoaded` (page 19)
: Obtains information about the load status of a bundle.

`– unload` (page 32)
: Unloads the code associated with the receiver.

Managing Localizations

`+ preferredLocalizationsFromArray:` (page 14)
: Returns one or more localizations from the specified list that a bundle object would use to locate resources for the current user.

`+ preferredLocalizationsFromArray:forPreferences:` (page 14)
: Returns the localizations that a bundle object would prefer, given the specified bundle and user preference localizations.

`– localizations` (page 21)
: Returns a list of all the localizations contained within the receiver's bundle.

`– developmentLocalization` (page 17)
: Returns the localization used to create the bundle.

`– preferredLocalizations` (page 28)
: Returns one or more localizations contained in the receiver's bundle that the receiver uses to locate resources based on the user's preferences.

`– localizedInfoDictionary` (page 21)
: Returns a dictionary with the keys from the bundle's localized property list.

Class Methods

allBundles

Returns an array of all the application's non-framework bundles.

```
+ (NSArray *)allBundles
```

Return Value
An array of all the application's non-framework bundles.

Discussion
The returned array includes the main bundle and all bundles that have been dynamically created but doesn't contain any bundles that represent frameworks.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

allFrameworks

Returns an array of all of the application's bundles that represent frameworks.

```
+ (NSArray *)allFrameworks
```

Return Value
An array of all of the application's bundles that represent frameworks. Only frameworks with one or more Objective-C classes in them are included.

Discussion
The returned array includes frameworks that are linked into an application when the application is built and bundles for frameworks that have been dynamically created.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Core Data HTML Store

CoreRecipes

Declared In
`NSBundle.h`

bundleForClass:

Returns the `NSBundle` object with which a given class is associated.

```
+ (NSBundle *)bundleForClass:(Class)aClass
```

Parameters

aClass

A class.

Return Value

The `NSBundle` object that dynamically loaded *aClass* (a loadable bundle), the `NSBundle` object for the framework in which *aClass* is defined, or the main bundle object if *aClass* was not dynamically loaded or is not defined in a framework.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ mainBundle` (page 11)

`+ bundleWithPath:` (page 11)

Related Sample Code

BundleLoader

CIAnnotation

Core Data HTML Store

CoreRecipes

GLSLShowpiece

Declared In

`NSBundle.h`

bundleWithIdentifier:

Returns the previously created `NSBundle` instance that has a given bundle identifier.

```
+ (NSBundle *)bundleWithIdentifier:(NSString *)identifier
```

Parameters

identifier

The identifier for an existing `NSBundle` instance.

Return Value

The previously created `NSBundle` instance that has the bundle identifier *identifier*. Returns `nil` if the requested bundle is not found.

Discussion

This method is typically used by frameworks and plug-ins to locate their own bundle at runtime. This method may be somewhat more efficient than trying to locate the bundle using the `bundleForClass:` (page 9) method.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

JavaSplashScreen

PrefsPane

Declared In

`NSBundle.h`

bundleWithPath:

Returns an `NSBundle` object that corresponds to the specified directory.

```
+ (NSBundle *)bundleWithPath:(NSString *)fullPath
```

Parameters

fullPath

The path to a directory. This must be a full pathname for a directory; if it contains any symbolic links, they must be resolvable.

Return Value

The `NSBundle` object that corresponds to *fullPath*, or `nil` if *fullPath* does not identify an accessible bundle directory.

Discussion

This method allocates and initializes the returned object if there is no existing `NSBundle` associated with *fullPath*, in which case it returns the existing object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ mainBundle` (page 11)

`+ bundleForClass:` (page 9)

`- initWithPath:` (page 18)

Related Sample Code

BundleLoader

Core Data HTML Store

Declared In

`NSBundle.h`

mainBundle

Returns the `NSBundle` object that corresponds to the directory where the current application executable is located.

```
+ (NSBundle *)mainBundle
```

Return Value

The `NSBundle` object that corresponds to the directory where the application executable is located, or `nil` if a bundle object could not be created.

Discussion

This method allocates and initializes a bundle object if one doesn't already exist. The new object corresponds to the directory where the application executable is located. Be sure to check the return value to make sure you have a valid bundle. This method may return a valid bundle object even for unbundled applications.

In general, the main bundle corresponds to an application file package or application wrapper: a directory that bears the name of the application and is marked by a "`.app`" extension.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ bundleForClass:` (page 9)

`+ bundleWithPath:` (page 11)

Related Sample Code

CITransitionSelectorSample2

CoreRecipes

NewsReader

NumberInput_IMKit_Sample

StickiesExample

Declared In

`NSBundle.h`

pathForResource:ofType:inDirectory:

Returns the full pathname for the resource file identified by a given name and extension and residing in a given bundle directory.

```
+ (NSString *)pathForResource:(NSString *)name ofType:(NSString *)extension
    inDirectory:(NSString *)bundlePath
```

Parameters

name

The name of a resource file contained in the bundle specified by *bundlePath*.

extension

If *extension* is an empty string or `nil`, the returned pathname is the first one encountered that exactly matches *name*.

bundlePath

The path of a top-level bundle directory. This must be a valid path. For example, to specify the bundle directory for an application, you might specify the path `/Applications/MyApp.app`.

Return Value

The full pathname for the resource file or `nil` if the file could not be located. This method also returns `nil` if the bundle specified by the `bundlePath` parameter does not exist or is not a readable directory.

Discussion

The method first looks for a matching resource file in the nonlocalized resource directory (typically `Resources`) of the specified bundle. If a matching resource file is not found, it then looks in the top level of any available language-specific "`.lproj`" directories. (The search order for the language-specific directories corresponds to the user's preferences.) It does not recurse through other subdirectories at any of these locations. For more details see Bundles and Localization.

Note: This method is best suited only for the occasional retrieval of resource files. In most cases where you need to retrieve bundle resources, it is preferable to use the `NSBundle` instance methods instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- localizedStringForKey:value:table:` (page 22)

- `pathForResource:ofType:` (page 24)
- `pathForResource:ofType:inDirectory:` (page 25)

+ `pathsForResourcesOfType:inDirectory:` (page 13)

- `pathsForResourcesOfType:inDirectory:` (page 27)

Declared In
`NSBundle.h`

pathsForResourcesOfType:inDirectory:

Returns an array containing the pathnames for all bundle resources having a given extension and residing in the bundle directory specified by a given path.

```
+ (NSArray *)pathsForResourcesOfType:(NSString *)extension inDirectory:(NSString
    *)bundlePath
```

Parameters

extension
: If *extension* is an empty string or `nil`, all bundle resources in the top-level resource directories are returned.

bundlePath
: The top-level directory of a bundle. This must represent a valid path.

Return Value
An array containing the full pathnames for all bundle resources with the specified extension. This method returns an empty array of no matching resource files are found. It also returns an empty array if the bundle specified by the `bundlePath` parameter does not exist or is not a readable directory.

Discussion
This method provides a means for dynamically discovering multiple bundle resources of the same type.

The method first looks for matching resource files in the nonlocalized resource directory (typically `Resources`) of the specified bundle. It then looks in the top level of any available language-specific "`.lproj`" directories. It does not recurse through other subdirectories at any of these locations. For more details see Bundles and Localization.

Note: This method is best suited only for the occasional retrieval of resource files. In most cases where you need to retrieve bundle resources, it is preferable to use the `NSBundle` instance methods instead.

Availability
Available in Mac OS X v10.0 and later.

See Also

- `localizedStringForKey:value:table:` (page 22)
- `pathForResource:ofType:` (page 24)
- `pathForResource:ofType:inDirectory:` (page 25)

+ `pathForResource:ofType:inDirectory:` (page 12)

Declared In
`NSBundle.h`

preferredLocalizationsFromArray:

Returns one or more localizations from the specified list that a bundle object would use to locate resources for the current user.

```
+ (NSArray *)preferredLocalizationsFromArray:(NSArray *)localizationsArray
```

Parameters

localizationsArray

An array of `NSString` objects, each of which specifies the name of a localization that the bundle supports.

Return Value

An array of `NSString` objects containing the preferred localizations. These strings are ordered in the array according to the current user's language preferences and are taken from the strings in the *localizationsArray* parameter.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBundle.h`

preferredLocalizationsFromArray:forPreferences:

Returns the localizations that a bundle object would prefer, given the specified bundle and user preference localizations.

```
+ (NSArray *)preferredLocalizationsFromArray:(NSArray *)localizationsArray
    forPreferences:(NSArray *)preferencesArray
```

Parameters

localizationsArray

An array of `NSString` objects, each of which specifies the name of a localization that the bundle supports.

preferencesArray

An array of `NSString` objects containing the user's preferred localizations. If this parameter is `nil`, the method uses the current user's localization preferences.

Return Value

An array of `NSString` objects containing the preferred localizations. These strings are ordered in the array according to the specified preferences and are taken from the strings in the *localizationsArray* parameter.

Discussion

Use the argument *localizationsArray* to specify the supported localizations of the bundle and use *preferencesArray* to specify the user’s localization preferences.

If none of the user-preferred localizations are available in the bundle, this method chooses one of the bundle localizations and returns it.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSBundle.h`

Instance Methods

builtInPlugInsPath

Returns the full pathname of the receiver's subdirectory containing plug-ins.

```
- (NSString *)builtInPlugInsPath
```

Return Value
The full pathname of the receiving bundle's subdirectory containing plug-ins.

Discussion
This method returns the appropriate path for modern application and framework bundles. This method may not return a path for non-standard bundle formats or for some older bundle formats.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BundleLoader

CIAnnotation

Core Data HTML Store

Declared In
`NSBundle.h`

bundleIdentifier

Returns the receiver's bundle identifier.

```
- (NSString *)bundleIdentifier
```

Return Value
The receiver's bundle identifier, which is defined by the `CFBundleIdentifier` key in the bundle's information property list.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- infoDictionary` (page 18)

Related Sample Code
CoreRecipes

MungSaver

NumberInput_IMKit_Sample

Declared In
`NSBundle.h`

bundlePath

Returns the full pathname of the receiver's bundle directory.

```
- (NSString *)bundlePath
```

Return Value
The full pathname of the receiver's bundle directory.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
JavaSplashScreen
NSGLImage

Declared In
`NSBundle.h`

classNamed:

Returns the `Class` object for the specified name.

```
- (Class)classNamed:(NSString *)className
```

Parameters

`className`
The name of a class.

Return Value
The `Class` object for *`className`*. Returns `NIL` if *`className`* is not one of the classes associated with the receiver or if there is an error loading the executable code containing the class implementation.

Discussion
If the bundle's executable code is not yet loaded, this method dynamically loads it into memory. Classes (and categories) are loaded from just one file within the bundle directory; this code file has the same name as the directory, but without the extension ("`.bundle`", "`.app`", "`.framework`"). As a side effect of code loading, the receiver posts `NSBundleDidLoadNotification` (page 34) after all classes and categories have been loaded; see "Notifications" (page 34) for details.

The following example loads a bundle's executable code containing the class "`FaxWatcher`":

```
- (void)loadBundle:(id)sender
{
    Class exampleClass;
    id newInstance;
    NSString *str = @"~/BundleExamples/BundleExample.bundle";
    NSBundle *bundleToLoad = [NSBundle bundleWithPath:str];
    if (exampleClass = [bundleToLoad classNamed:@"FaxWatcher"])  {
        newInstance = [[exampleClass alloc] init];
    // [newInstance doSomething];
    }
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `principalClass` (page 29)
- `load` (page 19)

Declared In
`NSBundle.h`

developmentLocalization

Returns the localization used to create the bundle.

```
- (NSString *)developmentLocalization
```

Return Value
The localization used to create the bundle.

Discussion
The returned localization corresponds to the value in the `CFBundleDevelopmentRegion` key of the bundle's property list (`Info.plist`).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSBundle.h`

executableArchitectures

Returns an array of numbers indicating the architecture types supported by the bundle's executable.

```
- (NSArray *)executableArchitectures
```

Return Value
An array of `NSNumber` objects, each of which contains an integer value corresponding to a supported processor architecture. For a list of common architecture types, see the constants in "Mach-O Architecture" (page 33). If the bundle does not contain a Mach-O executable, this method returns `nil`.

Discussion
This method scans the bundle's Mach-O executable and returns all of the architecture types it finds. Because they are taken directly from the executable, the returned values may not always correspond to one of the well-known CPU types defined in "Mach-O Architecture" (page 33).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSBundle.h`

executablePath

Returns the full pathname of the receiver's executable file.

```
- (NSString *)executablePath
```

Return Value
The full pathname of the receiving bundle's executable file.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

infoDictionary

Returns a dictionary that contains information about the receiver.

```
- (NSDictionary *)infoDictionary
```

Return Value
A dictionary, constructed from the bundle's `Info.plist` file, that contains information about the receiver. If the bundle does not contain an `Info.plist` file, a valid dictionary is returned but this dictionary contains only private keys that are used internally by the `NSBundle` class.

Discussion
Common keys for accessing the values of the dictionary are `CFBundleIdentifier`, `NSMainNibFile`, and `NSPrincipalClass`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– principalClass` (page 29)

Related Sample Code
JavaSplashScreen
PrefsPane
VertexPerformanceTest

Declared In
`NSBundle.h`

initWithPath:

Returns an `NSBundle` object initialized to correspond to a given directory.

```
- (id)initWithPath:(NSString *)fullPath
```

Parameters
fullPath
The path to a directory. This must be a full pathname for a directory; if it contains any symbolic links, they must be resolvable.

Return Value
An `NSBundle` object initialized to correspond to *`fullPath`*. This method initializes and returns a new instance only if there is no existing bundle associated with *`fullPath`*, otherwise it deallocates `self` and returns the existing object. If *`fullPath`* doesn't exist or the user doesn't have access to it, returns `nil`.

Discussion
It's not necessary to allocate and initialize an instance for the main bundle; use the `mainBundle` (page 11) class method to get this instance. You can also use the `bundleWithPath:` (page 11) class method to obtain a bundle identified by its directory path.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ bundleForClass:` (page 9)

Declared In
`NSBundle.h`

isLoaded

Obtains information about the load status of a bundle.

```
- (BOOL)isLoaded
```

Return Value
`YES` if the bundle's code is currently loaded, otherwise `NO`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- load` (page 19)

Declared In
`NSBundle.h`

load

Dynamically loads the bundle's executable code into a running program, if the code has not already been loaded.

```
- (BOOL)load
```

Return Value
`YES` if the method successfully loads the bundle's code or if the code has already been loaded, otherwise `NO`.

Discussion
You can use this method to load the code associated with a dynamically loaded bundle, such as a plug-in or framework. Prior to Mac OS X version 10.5, a bundle would attempt to load its code—if it had any—only once. Once loaded, you could not unload that code. In Mac OS X version 10.5 and later, you can unload a bundle's executable code using the `unload` (page 32) method.

You don't need to load a bundle's executable code to search the bundle's resources.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `loadAndReturnError:` (page 20)
- `isLoaded` (page 19)
- `unload` (page 32)
- `classNamed:` (page 16)
- `principalClass` (page 29)

Related Sample Code
Core Data HTML Store

Declared In
`NSBundle.h`

loadAndReturnError:

Loads the bundle's executable code and returns any errors.

```
- (BOOL)loadAndReturnError:(NSError **)error
```

Parameters

`error`

On input, a pointer to an error object variable. On output, this variable may contain an error object indicating why the bundle's executable could not be loaded. If no error occurred, this parameter is left unmodified. You may specify `nil` for this parameter if you are not interested in the error information.

Return Value
`YES` if the bundle's executable code was loaded successfully or was already loaded; otherwise, `NO` if the code could not be loaded.

Discussion
If this method returns `NO` and you pass a value for the *`error`* parameter, a suitable error object is returned in that parameter. Potential errors returned are in the Cocoa error domain and include the types that follow. For a full list of error types, see `FoundationErrors.h`.

- `NSFileNoSuchFileError` - returned if the bundle's executable file was not located.
- `NSExecutableNotLoadableError` - returned if the bundle's executable file exists but could not be loaded. This error is returned if the executable is not recognized as a loadable executable. It can also be returned if the executable is a PEF/CFM executable but the current process does not support that type of executable.
- `NSExecutableArchitectureMismatchError` - returned if the bundle executable does not include code that matches the processor architecture of the current processor.
- `NSExecutableRuntimeMismatchError` - returned if the bundle's required Objective-C runtime information is not compatible with the runtime of the current process.

- `NSExecutableLoadError` - returned if the bundle's executable failed to load for some detectable reason prior to linking. This error might occur if the bundle depends on a framework or library that is missing or if the required framework or library is not compatible with the current architecture or runtime version.
- `NSExecutableLinkError` - returned if the executable failed to load due to link errors but is otherwise alright.

The error object may contain additional debugging information in its description that you can use to identify the cause of the error. (This debugging information should not be displayed to the user.) You can obtain the debugging information by invoking the error object's `description` method in your code or by using the `print-object` command on the error object in gdb.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– load` (page 19)
`– unload` (page 32)

Declared In
`NSBundle.h`

localizations

Returns a list of all the localizations contained within the receiver's bundle.

```
- (NSArray *)localizations
```

Return Value
An array, containing `NSString` objects, that specifies all the localizations contained within the receiver's bundle.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

localizedInfoDictionary

Returns a dictionary with the keys from the bundle's localized property list.

```
- (NSDictionary *)localizedInfoDictionary
```

Return Value
A dictionary with the keys from the bundle's localized property list (`InfoPlist.strings`).

Discussion
This method uses the preferred localization for the current user when determining which resources to return. If the preferred localization is not available, this method chooses the most appropriate localization found in the bundle.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
PrefsPane

Declared In
`NSBundle.h`

localizedStringForKey:value:table:

Returns a localized version of the string designated by a given key in a given table.

```
- (NSString *)localizedStringForKey:(NSString *)key value:(NSString *)value
    table:(NSString *)tableName
```

Parameters

`key`
The key for a string in the table identified by *`tableName`*.

`value`
The value to return if *`key`* is `nil` or if a localized string for *`key`* can't be found in the table.

`tableName`
The receiver's string table to search. If *`tableName`* is `nil` or is an empty string, the method attempts to use the table in `Localizable.strings`.

Return Value
A localized version of the string designated by *`key`* in table *`tableName`*. If *`value`* is `nil` or an empty string, and a localized string is not found in the table, returns *`key`*. If *`key`* and *`value`* are both `nil`, returns the empty string.

Discussion
For more details about string localization and the specification of a `.strings` file, see "Working With Localized Strings."

Using the user default `NSShowNonLocalizedStrings`, you can alter the behavior of `localizedStringForKey:value:table:` (page 22) to log a message when the method can't find a localized string. If you set this default to `YES` (in the global domain or in the application's domain), then when the method can't find a localized string in the table, it logs a message to the console and capitalizes *`key`* before returning it.

The following example cycles through a static array of keys when a button is clicked, gets the value for each key from a strings table named `Buttons.strings`, and sets the button title with the returned value:

```
- (void)changeTitle:(id)sender
{
    static int keyIndex = 0;
    NSBundle *thisBundle = [NSBundle bundleForClass:[self class]];

    NSString *locString = [thisBundle
        localizedStringForKey:assortedKeys[keyIndex++]
        value:@"No translation" table:@"Buttons"];
    [sender setTitle:locString];
    if (keyIndex == MAXSTRINGS) keyIndex=0;
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pathForResource:ofType:` (page 24)
- `pathForResource:ofType:inDirectory:` (page 25)
- `pathsForResourcesOfType:inDirectory:` (page 27)

+ `pathForResource:ofType:inDirectory:` (page 12)

+ `pathsForResourcesOfType:inDirectory:` (page 13)

Related Sample Code
BundleLoader
CocoaDVDPlayer
InstallerPluginSample
NewsReader
Sketch-112

Declared In
`NSBundle.h`

objectForInfoDictionaryKey:

Returns the value associated with a given key in the receiver's property list.

```
- (id)objectForInfoDictionaryKey:(NSString *)key
```

Parameters

`key`
A key in the receiver's property list.

Return Value
The value associated with *`key`* in the receiver's property list (`Info.plist`). The localized value of a key is returned when one is available.

Discussion
Use of this method is preferred over other access methods because it returns the localized value of a key when one is available.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
AutoUpdater
BundleLoader
FancyAbout
GridCalendar

Declared In
`NSBundle.h`

pathForAuxiliaryExecutable:

Returns the full pathname of the executable with a given name in the receiver's bundle.

```
- (NSString *)pathForAuxiliaryExecutable:(NSString *)executableName
```

Parameters

executableName

The name of an executable file.

Return Value

The full pathname of the executable *executableName* in the receiver's bundle.

Discussion

This method returns the appropriate path for modern application and framework bundles. This method may not return a path for non-standard bundle formats or for some older bundle formats.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBundle.h`

pathForResource:ofType:

Returns the full pathname for the resource identified by a given name and specified file extension.

```
- (NSString *)pathForResource:(NSString *)name ofType:(NSString *)extension
```

Parameters

name

The name of the resource file.

extension

The file extension of a resource with the name *name*.

Return Value

The full pathname for the resource file or `nil` if the file could not be located.

Discussion

If *extension* is an empty string or `nil`, the returned pathname is the first one encountered where the file name exactly matches *name*.

The method first looks for a matching resource file in the nonlocalized resource directory (typically `Resources`) of the specified bundle. If a matching resource file is not found, it then looks in the top level of any available language-specific "`.lproj`" directories. (The search order for the language-specific directories corresponds to the user's preferences.) It does not recurse through other subdirectories at any of these locations. For more details see Bundles and Localization.

The following code fragment gets the path to a localized sound, creates an `NSSound` instance from it, and plays the sound.

```
NSString *soundPath;
NSSound *thisSound;
NSBundle *thisBundle = [NSBundle bundleForClass:[self class]];
if (soundPath = [thisBundle pathForResource:@"Hello" ofType:@"snd"])  {
    thisSound = [[[NSSound alloc] initFromSoundfile:soundPath] autorelease];
    [thisSound play];
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- localizedStringForKey:value:table:` (page 22)
`- pathForResource:ofType:` (page 24)
`- pathForResource:ofType:inDirectory:` (page 25)
`+ pathForResource:ofType:inDirectory:` (page 12)
`+ pathsForResourcesOfType:inDirectory:` (page 13)

Related Sample Code
AttachAScript
CIAnnotation
CITransitionSelectorSample2
GLSLShowpiece
StickiesExample

Declared In
`NSBundle.h`

pathForResource:ofType:inDirectory:

Returns the full pathname for the resource identified by the given name and file extension and located in the specified bundle subdirectory.

```
- (NSString *)pathForResource:(NSString *)name ofType:(NSString *)extension
    inDirectory:(NSString *)subpath
```

Parameters

name
The name of the resource file.

extension
The file extension of the specified resource file.

subpath
The name of the bundle subdirectory.

Return Value
The full pathname for the resource file or `nil` if the file could not be located.

Discussion
If *extension* is an empty string or `nil`, the returned pathname is the first one encountered where the file name exactly matches *name*.

If *subpath* is `nil`, this method searches the top-level nonlocalized resource directory (typically `Resources`) and the top-level of any language-specific directories. For example, suppose you have a modern bundle and specify `@"Documentation"` for the *subpath* parameter. This method would first look in the `Contents/Resources/Documentation` directory of the bundle, followed by the `Documentation` subdirectories of each language-specific `.lproj` directory. (The search order for the language-specific directories corresponds to the user's preferences.) This method does not recurse through any other subdirectories at any of these locations. For more details see Bundles and Localization.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- localizedStringForKey:value:table:` (page 22)
`- pathForResource:ofType:` (page 24)
`- pathsForResourcesOfType:inDirectory:` (page 27)
`+ pathForResource:ofType:inDirectory:` (page 12)
`+ pathsForResourcesOfType:inDirectory:` (page 13)

Declared In
`NSBundle.h`

pathForResource:ofType:inDirectory:forLocalization:

Returns the full pathname for the resource identified by the given name and file extension, located in the specified bundle subdirectory, and limited to global resources and those associated with the specified localization.

```
- (NSString *)pathForResource:(NSString *)name ofType:(NSString *)extension
    inDirectory:(NSString *)subpath forLocalization:(NSString *)localizationName
```

Parameters

name
The name of the resource file.

extension
The file extension of the specified resource file.

subpath
The name of the bundle subdirectory to search.

localizationName
The name of the localization. This parameter should correspond to the name of one of the bundle's language-specific resource directories without the `.lproj` extension.

Return Value
The full pathname for the resource file or `nil` if the file could not be located.

Discussion
This method is equivalent to `pathForResource:ofType:inDirectory:` (page 25), except that only nonlocalized resources and those in the language-specific `.lproj` directory specified by *localizationName* are searched.

There should typically be little reason to use this method—see Getting the Current Language and Locale. See also preferredLocalizationsFromArray:forPreferences: (page 14) for how to determine what localizations are available.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

pathsForResourcesOfType:inDirectory:

Returns an array containing the pathnames for all bundle resources having the specified filename extension and residing in the resource subdirectory.

```
- (NSArray *)pathsForResourcesOfType:(NSString *)extension inDirectory:(NSString
    *)subpath
```

Parameters

`extension`
The file extension of the files to retrieve.

`subpath`
The name of the bundle subdirectory to search.

Return Value
An array containing the full pathnames for all bundle resources matching the specified criteria. This method returns an empty array of no matching resource files are found.

Discussion
This method provides a means for dynamically discovering multiple bundle resources of the same type. If *`extension`* is an empty string or `nil`, all bundle resources in the specified resource directory are returned.

The argument *`subpath`* specifies the name of a specific subdirectory to search within the current bundle's resource directory hierarchy. If *`subpath`* is `nil`, this method searches the top-level nonlocalized resource directory (typically `Resources`) and the top-level of any language-specific directories. For example, suppose you have a modern bundle and specify `@"Documentation"` for the *`subpath`* parameter. This method would first look in the `Contents/Resources/Documentation` directory of the bundle, followed by the `Documentation` subdirectories of each language-specific `.lproj` directory. (The search order for the language-specific directories corresponds to the user's preferences.) This method does not recurse through any other subdirectories at any of these locations. For more details see Bundles and Localization.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- localizedStringForKey:value:table:` (page 22)
`- pathForResource:ofType:` (page 24)
`- pathForResource:ofType:inDirectory:` (page 25)
`+ pathForResource:ofType:inDirectory:` (page 12)
`+ pathsForResourcesOfType:inDirectory:` (page 13)

Related Sample Code
AutoSample
CocoaCreateMovie
QTKitCreateMovie

Declared In
`NSBundle.h`

pathsForResourcesOfType:inDirectory:forLocalization:

Returns an array containing the pathnames for all bundle resources having the specified filename extension, residing in the specified resource subdirectory, and limited to global resources and those associated with the specified localization.

```
- (NSArray *)pathsForResourcesOfType:(NSString *)extension inDirectory:(NSString
    *)subpath forLocalization:(NSString *)localizationName
```

Parameters

extension
The file extension of the files to retrieve.

subpath
The name of the bundle subdirectory to search.

localizationName
The name of the localization. This parameter should correspond to the name of one of the bundle's language-specific resource directories without the `.lproj` extension.

Return Value
An array containing the full pathnames for all bundle resources matching the specified criteria. This method returns an empty array of no matching resource files are found.

Discussion
This method is equivalent to `pathsForResourcesOfType:inDirectory:` (page 27), except that only nonlocalized resources and those in the language-specific `.lproj` directory specified by *localizationName* are searched.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

preferredLocalizations

Returns one or more localizations contained in the receiver’s bundle that the receiver uses to locate resources based on the user’s preferences.

```
- (NSArray *)preferredLocalizations
```

Return Value
One or more localizations contained in the receiver’s bundle that the receiver uses to locate resources based on the user’s preferences.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ preferredLocalizationsFromArray:` (page 14)
`- localizations` (page 21)

Declared In
`NSBundle.h`

preflightAndReturnError:

Returns a Boolean value indicating whether the bundle's executable code could be loaded successfully.

```
- (BOOL)preflightAndReturnError:(NSError **)error
```

Parameters

error

On input, a pointer to an error object variable. On output, this variable may contain an error object indicating why the bundle's executable could not be loaded. If no error would occur, this parameter is left unmodified. You may specify `nil` for this parameter if you are not interested in the error information.

Return Value

`YES` if the bundle's executable code could be loaded successfully or is already loaded; otherwise, `NO` if the code could not be loaded.

Discussion

This method does not actually load the bundle's executable code. Instead, it performs several checks to see if the code could be loaded and with one exception returns the same errors that would occur during an actual load operation. The one exception is the `NSExecutableLinkError` error, which requires the actual loading of the code to verify link errors.

For a list of possible load errors, see the discussion for the `loadAndReturnError:` (page 20) method.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- loadAndReturnError:` (page 20)

Declared In

`NSBundle.h`

principalClass

Returns the receiver's principal class.

```
- (Class)principalClass
```

Return Value

The receiver's principal class—after ensuring that the code containing the definition of that class is dynamically loaded. If the receiver encounters errors in loading or if it can't find the executable code file in the bundle directory, returns `NIL`.

Discussion

The principal class typically controls all the other classes in the bundle; it should mediate between those classes and classes external to the bundle. Classes (and categories) are loaded from just one file within the bundle directory. `NSBundle` obtains the name of the code file to load from the dictionary returned from `infoDictionary` (page 18), using "`NSExecutable`" as the key. The bundle determines its principal class in one of two ways:

- It first looks in its own information dictionary, which extracts the information encoded in the bundle's property list (`Info.plist`). `NSBundle` obtains the principal class from the dictionary using the key `NSPrincipalClass`. For nonloadable bundles (applications and frameworks), if the principal class is not specified in the property list, the method returns `NIL`.
- If the principal class is not specified in the information dictionary, `NSBundle` identifies the first class loaded as the principal class. When several classes are linked into a dynamically loadable file, the default principal class is the first one listed on the `ld` command line. In the following example, Reporter would be the principal class:

```
ld -o myBundle -r Reporter.o NotePad.o QueryList.o
```

The order of classes in Xcode's project browser is the order in which they will be linked. To designate the principal class, control-drag the file containing its implementation to the top of the list.

As a side effect of code loading, the receiver posts `NSBundleDidLoadNotification` (page 34) after all classes and categories have been loaded; see "Notifications" (page 34) for details.

The following method obtains a bundle by specifying its path (`bundleWithPath:` (page 11)), then loads the bundle with `principalClass` (page 29) and uses the returned class object to allocate and initialize an instance of that class:

```
- (void)loadBundle:(id)sender
{
    Class exampleClass;
    id newInstance;
    NSString *path = @"/tmp/Projects/BundleExample/BundleExample.bundle";
    NSBundle *bundleToLoad = [NSBundle bundleWithPath:path];
    if (exampleClass = [bundleToLoad principalClass]) {
        newInstance = [[exampleClass alloc] init];
        [newInstance doSomething];
    }
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- classNamed:` (page 16)
`- infoDictionary` (page 18)
`- load` (page 19)

Related Sample Code
BundleLoader

Declared In
`NSBundle.h`

privateFrameworksPath

Returns the full pathname of the receiver's subdirectory containing private frameworks.

```
- (NSString *)privateFrameworksPath
```

Return Value
The full pathname of the receiver's subdirectory containing private frameworks.

Discussion
This method returns the appropriate path for modern application and framework bundles. This method may not return a path for non-standard bundle formats or for some older bundle formats.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MP3 Player

Declared In
`NSBundle.h`

resourcePath

Returns the full pathname of the receiving bundle's subdirectory containing resources.

```
- (NSString *)resourcePath
```

Return Value
The full pathname of the receiving bundle's subdirectory containing resources.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- bundlePath` (page 16)

Related Sample Code
StickiesExample
SurfaceVertexProgram
TexturePerformanceDemo
TextureRange
VertexPerformanceDemo

Declared In
`NSBundle.h`

sharedFrameworksPath

Returns the full pathname of the receiver's subdirectory containing shared frameworks.

```
- (NSString *)sharedFrameworksPath
```

Return Value
The full pathname of the receiver's subdirectory containing shared frameworks.

Discussion
This method returns the appropriate path for modern application and framework bundles. This method may not return a path for non-standard bundle formats or for some older bundle formats.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

sharedSupportPath

Returns the full pathname of the receiver's subdirectory containing shared support files.

```
- (NSString *)sharedSupportPath
```

Return Value
The full pathname of the receiver's subdirectory containing shared support files.

Discussion
This method returns the appropriate path for modern application and framework bundles. This method may not return a path for non-standard bundle formats or for some older bundle formats.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

unload

Unloads the code associated with the receiver.

```
- (BOOL)unload
```

Return Value
`YES` if the bundle was successfully unloaded or was not already loaded; otherwise, `NO` if the bundle could not be unloaded.

Discussion
This method attempts to unload a bundle’s executable code using the underlying dynamic loader (typically `dyld`). You may use this method to unload plug-in and framework bundles when you no longer need the code they contain. You should use this method to unload bundles that were loaded using the methods of the `NSBundle` class only. Do not use this method to unload bundles that were originally loaded using the bundle-manipulation functions in Core Foundation.

It is the responsibility of the caller to ensure that no in-memory objects or data structures refer to the code being unloaded. For example, if you have an object whose class is defined in a bundle, you must release that object prior to unloading the bundle. Similarly, your code should not attempt to access any symbols defined in an unloaded bundle.

Special Considerations

Prior to Mac OS X version 10.5, code could not be unloaded once loaded, and this method would always return `NO`. In Mac OS X version 10.5 and later, you can unload a bundle’s executable code using this method.

Availability
Available in Mac OS X v10.5 and later.

See Also

- `loadAndReturnError:` (page 20)
- `load` (page 19)

Declared In

`NSBundle.h`

Constants

Mach-O Architecture

These constants describe the CPU types that a bundle's executable code may support.

```
enum {
    NSBundleExecutableArchitectureI386      = 0x00000007,
    NSBundleExecutableArchitecturePPC       = 0x00000012,
    NSBundleExecutableArchitectureX86_64    = 0x01000007,
    NSBundleExecutableArchitecturePPC64     = 0x01000012
};
```

Constants

`NSBundleExecutableArchitectureI386`

Specifies the 32-bit Intel architecture.

Available in Mac OS X v10.5 and later.

Declared in `NSBundle.h`.

`NSBundleExecutableArchitecturePPC`

Specifies the 32-bit PowerPC architecture.

Available in Mac OS X v10.5 and later.

Declared in `NSBundle.h`.

`NSBundleExecutableArchitectureX86_64`

Specifies the 64-bit Intel architecture.

Available in Mac OS X v10.5 and later.

Declared in `NSBundle.h`.

`NSBundleExecutableArchitecturePPC64`

Specifies the 64-bit PowerPC architecture.

Available in Mac OS X v10.5 and later.

Declared in `NSBundle.h`.

Declared In

`NSBundle.h`

Notifications

NSBundleDidLoadNotification

`NSBundle` posts `NSBundleDidLoadNotification` to notify observers which classes and categories have been dynamically loaded. When a request is made to an `NSBundle` object for a class (`classNamed:` (page 16) or `principalClass` (page 29)), the bundle dynamically loads the executable code file that contains the class implementation and all other class definitions contained in the file. After the module is loaded, the bundle posts the `NSBundleDidLoadNotification`.

The notification object is the `NSBundle` instance that dynamically loads classes. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSLoadedClasses"`	An `NSArray` object containing the names (as `NSString` objects) of each class that was loaded

In a typical use of this notification, an object might want to enumerate the *`userInfo`* array to check if each loaded class conformed to a certain protocol (say, an protocol for a plug-and-play tool set); if a class does conform, the object would create an instance of that class and add the instance to another `NSArray` object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBundle.h`

Document Revision History

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

Date	Notes
2007-07-19	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Index