NSArchiver Class Reference

Cocoa > Data Management

2006-05-23

Contents

NSArchiver Class Reference

Inherits from	NSCoder : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Archives and Serializations Programming Guide for Cocoa
Declared in	NSArchiver.h
Related sample code	Departments and Employees MenuItemView QTMetadataEditor Sketch-112 StickiesExample

Overview

`NSArchiver`, a concrete subclass of `NSCoder`, provides a way to encode objects into an architecture-independent format that can be stored in a file. When you archive a graph of objects, the class information and instance variables for each object are written to the archive. `NSArchiver`'s companion class, NSUnarchiver, decodes the data in an archive and creates a graph of objects equivalent to the original set.

`NSArchiver` stores the archive data in a mutable data object (`NSMutableData`). After encoding the objects, you can have the `NSArchiver` object write this mutable data object immediately to a file, or you can retrieve the mutable data object for some other use.

In Mac OS X v10.2 and later, `NSArchiver` and `NSUnarchiver` have been replaced by `NSKeyedArchiver` and `NSKeyedUnarchiver` respectively—see *Archives and Serializations Programming Guide for Cocoa*.

Tasks

Initializing an NSArchiver

`– initForWritingWithMutableData:` (page 10)
  Returns an archiver, initialized to encode stream and version information into a given mutable data object.

Archiving Data

+ `archivedDataWithRootObject:` (page 6)
  Returns a data object containing the encoded form of the object graph whose root object is given.

+ `archiveRootObject:toFile:` (page 7)
  Creates a temporary instance of `NSArchiver` and archives an object graph by encoding it into a data object and writing the resulting data object to a specified file.

- `encodeRootObject:` (page 9)
  Archives a given object along with all the objects to which it is connected.

- `encodeConditionalObject:` (page 9)
  Conditionally archives a given object.

Getting the Archived Data

- `archiverData` (page 8)
  Returns the receiver's archive data.

Substituting Classes or Objects

- `classNameEncodedForTrueClassName:` (page 8)
  Returns the name of the class used to archive instances of the class with a given true name.

- `encodeClassName:intoClassName:` (page 8)
  Encodes a substitute name for the class with a given true name.

- `replaceObject:withObject:` (page 10)
  Causes the receiver to treat subsequent requests to encode a given object as though they were requests to encode another given object.

Class Methods

archivedDataWithRootObject:

Returns a data object containing the encoded form of the object graph whose root object is given.

```
+ (NSData *)archivedDataWithRootObject:(id)rootObject
```

Parameters

rootObject
  The root object of the object graph to archive.

Return Value
A data object containing the encoded form of the object graph whose root object is *rootObject*.

Discussion
This method invokes `initForWritingWithMutableData:` (page 10) and `encodeRootObject:` (page 9) to create a temporary archiver that encodes the object graph.

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

See Also
`- initForWritingWithMutableData:` (page 10)
`- encodeRootObject:` (page 9)

Related Sample Code
Departments and Employees
MenuItemView
QTMetadataEditor
Sketch-112
StickiesExample

Declared In
`NSArchiver.h`

archiveRootObject:toFile:

Creates a temporary instance of `NSArchiver` and archives an object graph by encoding it into a data object and writing the resulting data object to a specified file.

```
+ (BOOL)archiveRootObject:(id)rootObject toFile:(NSString *)path
```

Parameters

`rootObject`
The root object of the object graph to archive.

`path`
The location of the the file into which to write the archive.

Return Value
`YES` if the archive was written successfully, otherwise `NO`.

Discussion
This convenience method invokes `archivedDataWithRootObject:` (page 6) to get the encoded data, and then sends that data object the message `writeToFile:atomically:`, using *`path`* for the first argument and `YES` for the second.

The archived data should be retrieved from the archive by an NSUnarchiver object.

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

See Also
`+ archivedDataWithRootObject:` (page 6)
`- writeToFile:atomically:` (NSData)

Declared In
`NSArchiver.h`

Instance Methods

archiverData

Returns the receiver's archive data.

```
- (NSMutableData *)archiverData
```

Return Value
The receiver's archive data.

Discussion
The returned data object is the same one specified as the argument to `initForWritingWithMutableData:` (page 10). It contains whatever data has been encoded thus far by invocations of the various encoding methods. It is safest not to invoke this method until after `encodeRootObject:` (page 9) has returned. In other words, although it is possible for a class to invoke this method from within its `encodeWithCoder:` method, that method must not alter the data.

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

Declared In
`NSArchiver.h`

classNameEncodedForTrueClassName:

Returns the name of the class used to archive instances of the class with a given true name.

```
- (NSString *)classNameEncodedForTrueClassName:(NSString *)trueName
```

Parameters

trueName
: The real name of an encoded class.

Return Value
The name of the class used to archive instances of the class *trueName*.

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

See Also
`- encodeClassName:intoClassName:` (page 8)

Declared In
`NSArchiver.h`

encodeClassName:intoClassName:

Encodes a substitute name for the class with a given true name.

```
- (void)encodeClassName:(NSString *)trueName intoClassName:(NSString *)inArchiveName
```

Parameters

`trueName`
: The real name of a class in the object graph being archived.

`inArchiveName`
: The name of the class to use in the archive in place of *`trueName`*.

Discussion

Any subsequently encountered objects of class *`trueName`* are archived as instances of class *`inArchiveName`*. It is safest not to invoke this method during the archiving process (that is, within an `encodeWithCoder:` method). Instead, invoke it before `encodeRootObject:` (page 9).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- classNameEncodedForTrueClassName:` (page 8)

Declared In

`NSArchiver.h`

encodeConditionalObject:

Conditionally archives a given object.

```
- (void)encodeConditionalObject:(id)object
```

Parameters

`object`
: The object to archive.

Discussion

This method overrides the superclass implementation to allow *`object`* to be encoded only if it is also encoded unconditionally by another object in the object graph. Conditional encoding lets you encode one part of a graph detached from the rest. (See *Archives and Serializations Programming Guide for Cocoa* for more information.)

This method should be invoked only from within an `encodeWithCoder:` method. If *`object`* is `nil`, the `NSArchiver` object encodes it unconditionally as `nil`. This method raises an `NSInvalidArgumentException` if no root object has been encoded.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSArchiver.h`

encodeRootObject:

Archives a given object along with all the objects to which it is connected.

```
- (void)encodeRootObject:(id)rootObject
```

Parameters

`rootObject`
: The root object of the object graph to archive.

Discussion

If any object is encountered more than once while traversing the graph, it is encoded only once, but the multiple references to it are stored. (See *Archives and Serializations Programming Guide for Cocoa* for more information.)

This message must not be sent more than once to a given `NSArchiver` object; an `NSInvalidArgumentException` is raised if a root object has already been encoded. If you need to encode multiple object graphs, therefore, don't attempt to reuse an `NSArchiver` instance; instead, create a new one for each graph.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSArchiver.h`

initForWritingWithMutableData:

Returns an archiver, initialized to encode stream and version information into a given mutable data object.

```
- (id)initForWritingWithMutableData:(NSMutableData *)data
```

Parameters

`data`
: The mutable data object into which to write the archive. This value must not be `nil`.

Return Value

An archiver object, initialized to encode stream and version information into *`data`*.

Discussion

Raises an `NSInvalidArgumentException` if *`data`* is `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- archiverData` (page 8)

Declared In

`NSArchiver.h`

replaceObject:withObject:

Causes the receiver to treat subsequent requests to encode a given object as though they were requests to encode another given object.

```
- (void)replaceObject:(id)object withObject:(id)newObject
```

Parameters

`object`

An object in the object graph being archived.

`newObject`

The object with which to replace *`object`* in the archive.

Discussion

Both *`object`* and *`newObject`* must be valid objects.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSArchiver.h`

Constants

Archiving Exception Names

Raised by `NSArchiver` if there are problems initializing or encoding.

```
extern NSString *NSInconsistentArchiveException;
```

Constants

`NSInconsistentArchiveException`

The name of an exception raised by `NSArchiver` if there are problems initializing or encoding.

Available in Mac OS X v10.0 and later.

Declared in `NSArchiver.h`.

Declared In

`NSArchiver.h`

Document Revision History

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

Date	Notes
2006-05-23	First publication of this content as a separate document.

Index