NSKeyedArchiver Class Reference

Cocoa > Data Management

2008-10-15

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

Simultaneously published in the United States and Canada.

Contents

NSKeyedArchiver Class Reference

Inherits from	NSCoder : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.2 and later.
Companion guide	Archives and Serializations Programming Guide for Cocoa
Declared in	NSKeyedArchiver.h
Related sample code	CoreRecipes CustomAtomicStoreSubclass iSpend QTQuartzPlayer Squiggles

Overview

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

A keyed archive differs from a non-keyed archive in that all the objects and values encoded into the archive are given names, or keys. When decoding a non-keyed archive, values have to be decoded in the same order in which they were encoded. When decoding a keyed archive, because values are requested by name, values can be decoded out of sequence or not at all. Keyed archives, therefore, provide better support for forward and backward compatibility.

The keys given to encoded values must be unique only within the scope of the current object being encoded. A keyed archive is hierarchical, so the keys used by object A to encode its instance variables do not conflict with the keys used by object B, even if A and B are instances of the same class. Within a single object, however, the keys used by a subclass can conflict with keys used in its superclasses.

An `NSArchiver` object can write the archive data to a file or to a mutable-data object (an instance of `NSMutableData`) that you provide.

Tasks

Initializing an NSKeyedArchiver Object

`- initForWritingWithMutableData:` (page 14)
Returns the receiver, initialized for encoding an archive into a given a mutable-data object.

Archiving Data

`+ archivedDataWithRootObject:` (page 7)
Returns an `NSData` object containing the encoded form of the object graph whose root object is given.

`+ archiveRootObject:toFile:` (page 8)
Archives an object graph rooted at a given object by encoding it into a data object then atomically writes the resulting data object to a file at a given path, and returns a Boolean value that indicates whether the operation was successful.

`- finishEncoding` (page 14)
Instructs the receiver to construct the final data stream.

`- outputFormat` (page 15)
Returns the format in which the receiver encodes its data.

`- setOutputFormat:` (page 16)
Sets the format in which the receiver encodes its data.

Encoding Data and Objects

`- archiver:didEncodeObject:` (page 17) *delegate method*
Informs the delegate that a given object has been encoded.

`- archiverDidFinish:` (page 18) *delegate method*
Notifies the delegate that encoding has finished.

`- archiver:willEncodeObject:` (page 17) *delegate method*
Informs the delegate that *`object`* is about to be encoded.

`- archiverWillFinish:` (page 18) *delegate method*
Notifies the delegate that encoding is about to finish.

`- archiver:willReplaceObject:withObject:` (page 18) *delegate method*
Informs the delegate that one given object is being substituted for another given object.

`- encodeBool:forKey:` (page 10)
Encodes a given Boolean value and associates it with a given key.

`- encodeBytes:length:forKey:` (page 11)
Encodes a given number of bytes from a given C array of bytes and associates them with the a given key.

`- encodeConditionalObject:forKey:` (page 11)
Encodes a reference to a given object and associates it with a given key only if it has been unconditionally encoded elsewhere in the archive with `encodeObject:forKey:` (page 14).

- `encodeDouble:forKey:` (page 12)
  Encodes a given `double` value and associates it with a given key.
- `encodeFloat:forKey:` (page 12)
  Encodes a given `float` value and associates it with a given key.
- `encodeInt:forKey:` (page 13)
  Encodes a given `int` value and associates it with a given key.
- `encodeInt32:forKey:` (page 12)
  Encodes a given 32-bit integer value and associates it with a given key.
- `encodeInt64:forKey:` (page 13)
  Encodes a given 64-bit integer value and associates it with a given key.
- `encodeObject:forKey:` (page 14)
  Encodes a given object and associates it with a given key.

Managing Delegates

- `delegate` (page 10)
  Returns the receiver's delegate.
- `setDelegate:` (page 16)
  Sets the delegate for the receiver.

Managing Classes and Class Names

+ `setClassName:forClass:` (page 9)
  Adds a class translation mapping to `NSKeyedArchiver` whereby instances of of a given class are encoded with a given class name instead of their real class names.
+ `classNameForClass:` (page 8)
  Returns the class name with which `NSKeyedArchiver` encodes instances of a given class.
- `setClassName:forClass:` (page 15)
  Adds a class translation mapping to the receiver whereby instances of of a given class are encoded with a given class name instead of their real class names.
- `classNameForClass:` (page 9)
  Returns the class name with which the receiver encodes instances of a given class.

Class Methods

archivedDataWithRootObject:

Returns an `NSData` object containing the encoded form of the object graph whose root object is given.

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

Parameters

rootObject

The root of the object graph to archive.

Return Value

An `NSData` object containing the encoded form of the object graph whose root object is *rootObject*. The format of the archive is `NSPropertyListBinaryFormat_v1_0`.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

CoreRecipes

CustomAtomicStoreSubclass

iSpend

QTQuartzPlayer

Squiggles

Declared In

`NSKeyedArchiver.h`

archiveRootObject:toFile:

Archives an object graph rooted at a given object by encoding it into a data object then atomically writes the resulting data object to a file at a given path, and returns a Boolean value that indicates whether the operation was successful.

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

Parameters

rootObject

The root of the object graph to archive.

path

The path of the file in which to write the archive.

Return Value

`YES` if the operation was successful, otherwise `NO`.

Discussion

The format of the archive is `NSPropertyListBinaryFormat_v1_0`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

classNameForClass:

Returns the class name with which `NSKeyedArchiver` encodes instances of a given class.

```
+ (NSString *)classNameForClass:(Class)cls
```

Parameters

cls

The class for which to determine the translation mapping.

Return Value

The class name with which `NSKeyedArchiver` encodes instances of *cls*. Returns `nil` if `NSKeyedArchiver` does not have a translation mapping for *cls*.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ setClassName:forClass:` (page 9)

`- classNameForClass:` (page 9)

Declared In

`NSKeyedArchiver.h`

setClassName:forClass:

Adds a class translation mapping to `NSKeyedArchiver` whereby instances of of a given class are encoded with a given class name instead of their real class names.

```
+ (void)setClassName:(NSString *)codedName  forClass:(Class)cls
```

Parameters

codedName

The name of the class that `NSKeyedArchiver` uses in place of `cls`.

cls

The class for which to set up a translation mapping.

Discussion

When encoding, the class's translation mapping is used only if no translation is found first in an instance's separate translation map.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ classNameForClass:` (page 8)

`- setClassName:forClass:` (page 15)

Declared In

`NSKeyedArchiver.h`

Instance Methods

classNameForClass:

Returns the class name with which the receiver encodes instances of a given class.

```
- (NSString *)classNameForClass:(Class)cls
```

Parameters

cls

The class for which to determine the translation mapping.

Return Value

The class name with which the receiver encodes instances of *cls*. Returns `nil` if the receiver does not have a translation mapping for *cls*. The class's separate translation map is not searched.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- setClassName:forClass:` (page 15)

`+ classNameForClass:` (page 8)

Declared In

`NSKeyedArchiver.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value

The receiver's delegate.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- setDelegate:` (page 16)

Declared In

`NSKeyedArchiver.h`

encodeBool:forKey:

Encodes a given Boolean value and associates it with a given key.

```
- (void)encodeBool:(BOOL)boolv  forKey:(NSString *)key
```

Parameters

boolv

The value to encode.

key

The key with which to associate *boolv*. This value must not be `nil`.

Availability

Available in Mac OS X v10.2 and later.

See Also
`decodeBoolForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeBytes:length:forKey:

Encodes a given number of bytes from a given C array of bytes and associates them with the a given key.

```
- (void)encodeBytes:(const uint8_t *)bytesp  length:(NSUInteger)lenv
    forKey:(NSString *)key
```

Parameters

bytesp
A C array of bytes to encode.

lenv
The number of bytes from *bytesp* to encode.

key
The key with which to associate the encoded value. This value must not be `nil`.

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

See Also
`decodeBytesForKey:returnedLength:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeConditionalObject:forKey:

Encodes a reference to a given object and associates it with a given key only if it has been unconditionally encoded elsewhere in the archive with `encodeObject:forKey:` (page 14).

```
- (void)encodeConditionalObject:(id)objv  forKey:(NSString *)key
```

Parameters

objv
The object to encode.

key
The key with which to associate the encoded value. This value must not be `nil`.

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

Declared In
`NSKeyedArchiver.h`

encodeDouble:forKey:

Encodes a given `double` value and associates it with a given key.

```
- (void)encodeDouble:(double)realv  forKey:(NSString *)key
```

Parameters

realv

The value to encode.

key

The key with which to associate *realv*. This value must not be `nil`.

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

See Also
`decodeDoubleForKey:` (`NSKeyedUnarchiver`)
`decodeFloatForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeFloat:forKey:

Encodes a given `float` value and associates it with a given key.

```
- (void)encodeFloat:(float)realv  forKey:(NSString *)key
```

Parameters

realv

The value to encode.

key

The key with which to associate *realv*. This value must not be `nil`.

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

See Also
`decodeFloatForKey:` (`NSKeyedUnarchiver`)
`decodeDoubleForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeInt32:forKey:

Encodes a given 32-bit integer value and associates it with a given key.

```
- (void)encodeInt32:(int32_t)intv  forKey:(NSString *)key
```

Parameters

intv

The value to encode.

key

The key with which to associate *intv*. This value must not be `nil`.

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

See Also
`decodeInt32ForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeInt64:forKey:

Encodes a given 64-bit integer value and associates it with a given key.

```
- (void)encodeInt64:(int64_t)intv  forKey:(NSString *)key
```

Parameters

intv

The value to encode.

key

The key with which to associate *intv*. This value must not be `nil`.

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

See Also
`decodeInt64ForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeInt:forKey:

Encodes a given `int` value and associates it with a given key.

```
- (void)encodeInt:(int)intv  forKey:(NSString *)key
```

Parameters

intv

The value to encode.

key

The key with which to associate *intv*. This value must not be `nil`.

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

See Also
`decodeIntForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

encodeObject:forKey:

Encodes a given object and associates it with a given key.

```
- (void)encodeObject:(id)objv  forKey:(NSString *)key
```

Parameters

objv

The value to encode. This value may be `nil`.

key

The key with which to associate *objv*. This value must not be `nil`.

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

See Also
`decodeObjectForKey:` (`NSKeyedUnarchiver`)

Declared In
`NSKeyedArchiver.h`

finishEncoding

Instructs the receiver to construct the final data stream.

```
- (void)finishEncoding
```

Discussion
No more values can be encoded after this method is called. You must call this method when finished.

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

See Also
`- initForWritingWithMutableData:` (page 14)

Declared In
`NSKeyedArchiver.h`

initForWritingWithMutableData:

Returns the receiver, initialized for encoding an archive into a given a mutable-data object.

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

Parameters

data

The mutable-data object into which the archive is written.

Return Value

The receiver, initialized for encoding an archive into *data*.

Discussion

When you finish encoding data, you must invoke `finishEncoding` (page 14) at which point *data* is filled. The format of the receiver is `NSPropertyListBinaryFormat_v1_0`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

outputFormat

Returns the format in which the receiver encodes its data.

```
- (NSPropertyListFormat)outputFormat
```

Return Value

The format in which the receiver encodes its data. The available formats are `NSPropertyListXMLFormat_v1_0` and `NSPropertyListBinaryFormat_v1_0`.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- setOutputFormat:` (page 16)

Declared In

`NSKeyedArchiver.h`

setClassName:forClass:

Adds a class translation mapping to the receiver whereby instances of of a given class are encoded with a given class name instead of their real class names.

```
- (void)setClassName:(NSString *)codedName  forClass:(Class)cls
```

Parameters

codedName

The name of the class that the receiver uses uses in place of `cls`.

cls

The class for which to set up a translation mapping.

Discussion

When encoding, the receiver's translation map overrides any translation that may also be present in the class's map.

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

See Also
- `classNameForClass:` (page 9)
+ `setClassName:forClass:` (page 9)

Declared In
`NSKeyedArchiver.h`

setDelegate:

Sets the delegate for the receiver.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate
The delegate for the receiver.

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

See Also
- `delegate` (page 10)

Declared In
`NSKeyedArchiver.h`

setOutputFormat:

Sets the format in which the receiver encodes its data.

```
- (void)setOutputFormat:(NSPropertyListFormat)format
```

Parameters

format
The format in which the receiver encodes its data. *format* can be `NSPropertyListXMLFormat_v1_0` or `NSPropertyListBinaryFormat_v1_0`.

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

See Also
- `outputFormat` (page 15)

Declared In
`NSKeyedArchiver.h`

Delegate Methods

archiver:didEncodeObject:

Informs the delegate that a given object has been encoded.

```
- (void)archiver:(NSKeyedArchiver *)archiver  didEncodeObject:(id)object
```

Parameters

archiver

The archiver that sent the message.

object

The object that has been encoded. *object* may be `nil`.

Discussion

The delegate might restore some state it had modified previously, or use this opportunity to keep track of the objects that are encoded.

This method is not called for conditional objects until they are actually encoded (if ever).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

archiver:willEncodeObject:

Informs the delegate that *object* is about to be encoded.

```
- (id)archiver:(NSKeyedArchiver *)archiver  willEncodeObject:(id)object
```

Parameters

archiver

The archiver that sent the message.

object

The object that is about to be encoded. This value is never `nil`.

Return Value

Either *object* or a different object to be encoded in its stead. The delegate can also modify the coder state. If the delegate returns `nil`, `nil` is encoded.

Discussion

This method is called after the original object may have replaced itself with `replacementObjectForKeyedArchiver:`.

This method is called whether or not the object is being encoded conditionally.

This method is not called for an object once a replacement mapping has been set up for that object (either explicitly, or because the object has previously been encoded). This method is also not called when `nil` is about to be encoded.

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

Declared In
NSKeyedArchiver.h

archiver:willReplaceObject:withObject:

Informs the delegate that one given object is being substituted for another given object.

```
- (void)archiver:(NSKeyedArchiver *)archiver  willReplaceObject:(id)object
    withObject:(id)newObject
```

Parameters

archiver
The archiver that sent the message.

object
The object being replaced in the archive.

newObject
The object replacing *object* in the archive.

Discussion
This method is called even when the delegate itself is doing, or has done, the substitution. The delegate may use this method if it is keeping track of the encoded or decoded objects.

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

Declared In
NSKeyedArchiver.h

archiverDidFinish:

Notifies the delegate that encoding has finished.

```
- (void)archiverDidFinish:(NSKeyedArchiver *)archiver
```

Parameters

archiver
The archiver that sent the message.

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

Declared In
NSKeyedArchiver.h

archiverWillFinish:

Notifies the delegate that encoding is about to finish.

```
- (void)archiverWillFinish:(NSKeyedArchiver *)archiver
```

Parameters

archiver

The archiver that sent the message.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

Constants

Keyed Archiving Exception Names

Names of exceptions that are raised by `NSKeyedArchiver` if there is a problem creating an archive.

```
extern NSString *NSInvalidArchiveOperationException;
```

Constants

`NSInvalidArchiveOperationException`

The name of the exception raised by `NSKeyedArchiver` if there is a problem creating an archive.

Available in Mac OS X v10.2 and later.

Declared in `NSKeyedArchiver.h`.

Declared In

NSKeyedArchiver.h

Document Revision History

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

Date	Notes
2008-10-15	Clarified behavior of encodeObject:forKey: for nil values.
2007-01-30	Merged delegate method group with regular method groups.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index