Class

NSKeyedArchiver

An encoder that stores an object's data to an archive referenced by keys.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.0+ | macOS 10.0+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class NSKeyedArchiver
```

Overview

`NSKeyedArchiver`, a concrete subclass of `NSCoder`, provides a way to encode objects (and scalar values) into an architecture-independent format suitable for storage in a file. When you archive a set of objects, the archiver writes the class information and instance variables for each object to the archive. The 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 have names, or keys. When decoding a non-keyed archive, the decoder must decode values in the same order the original encoder used. When decoding a keyed archive, the decoder requests values by name, meaning it can decode values 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 currently-encoding object. A keyed archive is hierarchical, so the keys used by object A to encode its instance variables don't conflict with the keys used by object B. This is true 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.

Topics

Creating a Keyed Archiver

`init(requiringSecureCoding: Bool)`

Creates an archiver to encode data, and optionally disables secure coding.

~~`init()`~~

Initializes an archiver to encode data.

Deprecated

~~`init(forWritingWith: NSMutableData)`~~

Initializes an archiver to encode data into a given a mutable-data object.

Deprecated

Archiving Data

`class func archivedData(withRootObject: Any, requiringSecureCoding: Bool) throws -> Data`

Encodes an object graph with the given root object into a data representation, optionally requiring secure coding.

`func finishEncoding()`

Instructs the receiver to construct the final data stream.

`var encodedData: Data`

The encoded data for the archiver.

`var outputFormat: PropertyListSerialization.PropertyListFormat`

The format in which the receiver encodes its data.

`var requiresSecureCoding: Bool`

Indicates whether the archiver requires all archived classes to resist object substitution attacks.

~~`class func archivedData(withRootObject: Any) -> Data`~~

Returns a data object that contains the encoded form of the object graph formed by the given root object.

Deprecated

~~`class func archiveRootObject(Any, toFile: String) -> Bool`~~

Archives an object graph rooted at a given object to a file at a given path.

Deprecated

Encoding Data and Objects

`func encodeEncodable<T>(T, forKey: String) throws`

Encodes a given value and associates it with a key.

`func encode(Bool, forKey: String)`

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

`func encodeBytes(UnsafePointer<UInt8>?, length: Int, forKey: String)`

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

`func encodeConditionalObject(Any?, forKey: String)`

Encodes a reference to a given object and associates it with a key only if it has been unconditionally encoded elsewhere in the archive.

`func encode(Double, forKey: String)`

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

`func encode(Float, forKey: String)`

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

`func encode(Int32, forKey: String)`

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

`func encode(Int64, forKey: String)`

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

`func encode(Any?, forKey: String)`

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

Managing the Delegate

`var delegate: (any NSKeyedArchiverDelegate)?`

The archiver's delegate.

Managing Classes and Class Names

`class func setClassName(String?, for: AnyClass)`

Sets a global translation mapping to encode instances of a given class with the provided name, rather than their real name.

`class func className(for: AnyClass) -> String?`

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

`func setClassName(String?, for: AnyClass)`

Sets a mapping for this archiver to encode instances of a given class with the provided name, rather than their real name.

`func className(for: AnyClass) -> String?`

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

Constants

Keyed Archiving Exception Names

Names of exceptions raised by this class if problems occur while creating an archive.

Keyed Archiver Root Object Key

Keys that the archiver uses in the hierarchy of encoded objects.

Relationships

Inherits From

`NSCoder`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`

NSObjectProtocol
Sendable
SendableMetatype

See Also

Keyed Archivers

`protocol` NSKeyedArchiverDelegate

The optional methods implemented by the delegate of a keyed archiver.

`class` NSKeyedUnarchiver

A decoder that restores data from an archive referenced by keys.

`protocol` NSKeyedUnarchiverDelegate

The optional methods implemented by the delegate of a keyed unarchiver.

`class` NSCoder

An abstract class that serves as the basis for objects that enable archiving and distribution of other objects.

`class` NSSecureUnarchiveFromDataTransformer

A value transformer that converts data to and from classes that support secure coding.