NSEntityDescription Class Reference

Data Management

2010-04-06

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

Simultaneously published in the United States and Canada.

Contents

NSEntityDescription Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSFastEnumeration NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreData.framework
Availability	Available in iOS 3.0 and later.
Declared in	NSEntityDescription.h
Companion guides	Core Data Programming Guide Core Data Utility Tutorial

Overview

An `NSEntityDescription` object describes an entity in Core Data. Entities are to managed objects what `Class` is to `id`, or—to use a database analogy—what tables are to rows. An instance specifies an entity's name, its properties (its attributes and relationships, expressed by instances of `NSAttributeDescription` and `NSRelationshipDescription`) and the class by which it is represented.

An `NSEntityDescription` object is associated with a specific class whose instances are used to represent entries in a persistent store in applications using the Core Data Framework. Minimally, an entity description should have:

- A name
- The name of a managed object class

  (If an entity has no managed object class name, it defaults to `NSManagedObject`.)

You usually define entities in an `NSManagedObjectModel` using the data modeling tool in Xcode. `NSEntityDescription` objects are primarily used by the Core Data Framework for mapping entries in the persistent store to managed objects in the application. You are not likely to interact with them directly unless you are specifically working with models. Like the other major modeling classes, `NSEntityDescription` provides you with a user dictionary in which you can store any application-specific information related to the entity.

Editing Entity Descriptions

Entity descriptions are editable until they are used by an object graph manager. This allows you to create or modify them dynamically. However, once a description is used (when the managed object model to which it belongs is associated with a persistent store coordinator), it *must not* (indeed cannot) be changed. This is enforced at runtime: any attempt to mutate a model or any of its sub-objects after the model is associated with a persistent store coordinator causes an exception to be thrown. If you need to modify a model that is in use, create a copy, modify the copy, and then discard the objects with the old model.

If you want to create an entity hierarchy, you need to consider the relevant API. You can only set an entity's sub-entities (see `setSubentities:` (page 17)), you cannot set an entity's super-entity directly. To set a super-entity for a given entity, you must therefore set an array of subentities on that super entity and include the current entity in that array. So, the entity hierarchy needs to be built top-down.

Using Entity Descriptions in Dictionaries

`NSEntityDescription`'s `copy` (page 11) method returns an entity such that

```
[[entity copy] isEqual: entity] == NO
```

Since `NSDictionary` copies its keys and requires that keys both conform to the `NSCopying` protocol and have the property that `copy` returns an object for which `[[object copy] isEqual:object]` is true, you should not use entities as keys in a dictionary. Instead, you should either use the entity's name as the key, or use a map table (`NSMapTable`) with retain callbacks.

Fast Enumeration

In Mac OS v10.5 and later and on iOS, `NSEntityDescription` supports the `NSFastEnumeration` protocol. You can use this to enumerate over an entity's properties, as illustrated in the following example:

```
NSEntityDescription *anEntity = ...;
for (NSPropertyDescription *property in anEntity) {
    // property is each instance of NSPropertyDescription in anEntity in turn
}
```

Tasks

Information About an Entity Description

- `name` (page 12)
  Returns the entity name of the receiver.
- `setName:` (page 16)
  Sets the entity name of the receiver.
- `managedObjectModel` (page 12)
  Returns the managed object model with which the receiver is associated.

- `managedObjectClassName` (page 12)
  Returns the name of the class that represents the receiver's entity.
- `setManagedObjectClassName:` (page 15)
  Sets the name of the class that represents the receiver's entity.
- `renamingIdentifier` (page 14)
  Returns the renaming identifier for the receiver.
- `setRenamingIdentifier:` (page 17)
  Sets the renaming identifier for the receiver.
- `isAbstract` (page 11)
  Returns a Boolean value that indicates whether the receiver represents an abstract entity.
- `setAbstract:` (page 15)
  Sets whether the receiver represents an abstract entity.
- `userInfo` (page 20)
  Returns the user info dictionary of the receiver.
- `setUserInfo:` (page 18)
  Sets the user info dictionary of the receiver.

Managing Inheritance

- `subentitiesByName` (page 19)
  Returns the sub-entities of the receiver in a dictionary.
- `subentities` (page 18)
  Returns an array containing the sub-entities of the receiver.
- `setSubentities:` (page 17)
  Sets the subentities of the receiver.
- `superentity` (page 19)
  Returns the super-entity of the receiver.
- `isKindOfEntity:` (page 11)
  Returns a Boolean value that indicates whether the receiver is a sub-entity of another given entity.

Working with Properties

- `propertiesByName` (page 13)
  Returns a dictionary containing the properties of the receiver.
- `properties` (page 13)
  Returns an array containing the properties of the receiver.
- `setProperties:` (page 16)
  Sets the properties array of the receiver.
- `attributesByName` (page 10)
  Returns the attributes of the receiver in a dictionary, where the keys in the dictionary are the attribute names.
- `relationshipsByName` (page 14)
  Returns the relationships of the receiver in a dictionary, where the keys in the dictionary are the relationship names.

Retrieving an Entity with a Given Name

Creating a New Managed Object

Supporting Versioning

Copying Entity Descriptions

Class Methods

entityForName:inManagedObjectContext:

Returns the entity with the specified name from the managed object model associated with the specified managed object context's persistent store coordinator.

```
+ (NSEntityDescription *)entityForName:(NSString *)entityName
    inManagedObjectContext:(NSManagedObjectContext *)context
```

Parameters

entityName

The name of an entity.

context

The managed object context to use.

Return Value
The entity with the specified name from the managed object model associated with *context*'s persistent store coordinator.

Discussion
This method is functionally equivalent to the following code example.

```
NSManagedObjectModel *managedObjectModel = [[context persistentStoreCoordinator]
 managedObjectModel];
NSEntityDescription *entity = [[managedObjectModel entitiesByName]
objectForKey:entityName];
return entity;
```

Availability
Available in iOS 3.0 and later.

See Also
- entitiesByName

Declared In
NSEntityDescription.h

insertNewObjectForEntityForName:inManagedObjectContext:

Creates, configures, and returns an instance of the class for the entity with a given name.

```
+ (id)insertNewObjectForEntityForName:(NSString *)entityName
    inManagedObjectContext:(NSManagedObjectContext *)context
```

Parameters

entityName

The name of an entity.

context

The managed object context to use.

Return Value
A new, autoreleased, fully configured instance of the class for the entity named *entityName*. The instance has its entity description set and is inserted it into *context*.

Discussion
This method makes it easy for you to create instances of a given entity without worrying about the details of managed object creation.

The method is particularly useful on Mac OS X v10.4, as you can use it to create a new managed object without having to know the class used to represent the entity. This is especially beneficial early in the development life-cycle when classes and class names are volatile. The method is conceptually similar to the following code example.

```
NSManagedObjectModel *managedObjectModel =
        [[context persistentStoreCoordinator] managedObjectModel];
NSEntityDescription *entity =
        [[managedObjectModel entitiesByName] objectForKey:entityName];
```

```
NSString *className = [entity managedObjectClassName];
Class entityClass = [[NSBundle mainBundle] classNamed:className];
id newObject = [[entityClass alloc]
                initWithEntity:entity insertIntoManagedObjectContext:context];
return [newObject autorelease];
```

On Mac OS X v10.5 and later and on iOS, you can instead use `initWithEntity:insertIntoManagedObjectContext:` which returns an instance of the appropriate class for the entity. The equivalent code for Mac OS X v10.5 and on iOS is as follows:

```
NSManagedObjectModel *managedObjectModel =
        [[context persistentStoreCoordinator] managedObjectModel];
NSEntityDescription *entity =
        [[managedObjectModel entitiesByName] objectForKey:entityName];
NSManagedObject *newObject = [[NSManagedObject alloc]
            initWithEntity:entity insertIntoManagedObjectContext:context];
return [newObject autorelease];
```

Important: Despite the presence of the word "new" in the method name, in a reference counted environment you are not responsible for releasing the returned object. ("new" is not the *first* word in the method name—see Memory Management Rules).

Availability
Available in iOS 3.0 and later.

See Also
`- initWithEntity:insertIntoManagedObjectContext:`

Declared In
`NSEntityDescription.h`

Instance Methods

attributesByName

Returns the attributes of the receiver in a dictionary, where the keys in the dictionary are the attribute names.

```
- (NSDictionary *)attributesByName
```

Return Value
The attributes of the receiver in a dictionary, where the keys in the dictionary are the attribute names and the values are instances of `NSAttributeDescription`..

Availability
Available in iOS 3.0 and later.

See Also
- `propertiesByName` (page 13)
- `relationshipsByName` (page 14)
- `relationshipsWithDestinationEntity:` (page 14)

Declared In
NSEntityDescription.h

copy

Returns a copy of the receiver

```
- (id)copy
```

Return Value
A copy of the receiver.

Special Considerations
NSEntityDescription's implementation of copy returns an entity such that:

```
[[entity copy] isEqual:entity] == NO
```

You should not, therefore, use an entity as a key in a dictionary (see "Using Entity Descriptions in Dictionaries" (page 6)).

isAbstract

Returns a Boolean value that indicates whether the receiver represents an abstract entity.

```
- (BOOL)isAbstract
```

Return Value
YES if the receiver represents an abstract entity, otherwise NO.

Discussion
An abstract entity might be Shape, with concrete sub-entities such as Rectangle, Triangle, and Circle.

Availability
Available in iOS 3.0 and later.

See Also
- setAbstract: (page 15)

Declared In
NSEntityDescription.h

isKindOfEntity:

Returns a Boolean value that indicates whether the receiver is a sub-entity of another given entity.

```
- (BOOL)isKindOfEntity:(NSEntityDescription *)entity
```

Parameters
entity
An entity.

Return Value
YES if the receiver is a sub-entity of *entity*, otherwise NO.

Availability
Available in iOS 3.0 and later.

Declared In
NSEntityDescription.h

managedObjectClassName

Returns the name of the class that represents the receiver's entity.

```
- (NSString *)managedObjectClassName
```

Return Value
The name of the class that represents the receiver's entity.

Availability
Available in iOS 3.0 and later.

See Also
- setManagedObjectClassName: (page 15)

Declared In
NSEntityDescription.h

managedObjectModel

Returns the managed object model with which the receiver is associated.

```
- (NSManagedObjectModel *)managedObjectModel
```

Return Value
The managed object model with which the receiver is associated.

Availability
Available in iOS 3.0 and later.

See Also
setEntities:(NSManagedObjectModel)
setEntities:forConfiguration::(NSManagedObjectModel)

Declared In
NSEntityDescription.h

name

Returns the entity name of the receiver.

```
- (NSString *)name
```

Return Value
The entity name of receiver.

Availability
Available in iOS 3.0 and later.

See Also
- `setName:` (page 16)

Declared In
`NSEntityDescription.h`

properties

Returns an array containing the properties of the receiver.

```
- (NSArray *)properties
```

Return Value
An array containing the properties of the receiver. The elements in the array are instances of `NSAttributeDescription`, `NSRelationshipDescription`, and/or `NSFetchedPropertyDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `propertiesByName` (page 13)
- `setProperties:` (page 16)
- `attributesByName` (page 10)
- `relationshipsByName` (page 14)

Declared In
`NSEntityDescription.h`

propertiesByName

Returns a dictionary containing the properties of the receiver.

```
- (NSDictionary *)propertiesByName
```

Return Value
A dictionary containing the receiver's properties, where the keys in the dictionary are the property names and the values are instances of `NSAttributeDescription` and/or `NSRelationshipDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `attributesByName` (page 10)
- `relationshipsByName` (page 14)
- `relationshipsWithDestinationEntity:` (page 14)

Declared In
`NSEntityDescription.h`

relationshipsByName

Returns the relationships of the receiver in a dictionary, where the keys in the dictionary are the relationship names.

```
- (NSDictionary *)relationshipsByName
```

Return Value
The relationships of the receiver in a dictionary, where the keys in the dictionary are the relationship names and the values are instances of `NSRelationshipDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `attributesByName` (page 10)
- `propertiesByName` (page 13)
- `relationshipsWithDestinationEntity:` (page 14)

Declared In
`NSEntityDescription.h`

relationshipsWithDestinationEntity:

Returns an array containing the relationships of the receiver where the entity description of the relationship is a given entity.

```
- (NSArray *)relationshipsWithDestinationEntity:(NSEntityDescription *)entity
```

Parameters
entity
An entity description.

Return Value
An array containing the relationships of the receiver where the entity description of the relationship is *entity*. Elements in the array are instances of `NSRelationshipDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `attributesByName` (page 10)
- `propertiesByName` (page 13)
- `relationshipsByName` (page 14)

Declared In
`NSEntityDescription.h`

renamingIdentifier

Returns the renaming identifier for the receiver.

```
- (NSString *)renamingIdentifier
```

Return Value
The renaming identifier for the receiver.

Discussion
The renaming identifier is used to resolve naming conflicts between models. When creating a mapping model between two managed object models, a source entity and a destination entity that share the same identifier indicate that an entity mapping should be configured to migrate from the source to the destination.

If you do not set this value, the identifier will return the entity's name.

Availability
Available in iOS 3.0 and later.

See Also
`- setRenamingIdentifier:` (page 17)

Declared In
`NSEntityDescription.h`

setAbstract:

Sets whether the receiver represents an abstract entity.

```
- (void)setAbstract:(BOOL)flag
```

Parameters

flag

A Boolean value indicating whether the receiver is abstract (`YES`) or not (`NO`).

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability
Available in iOS 3.0 and later.

See Also
`- isAbstract` (page 11)

Declared In
`NSEntityDescription.h`

setManagedObjectClassName:

Sets the name of the class that represents the receiver's entity.

```
- (void)setManagedObjectClassName:(NSString *)name
```

Parameters

name

The name of the class that represents the receiver's entity.

Discussion
The class specified by *name* must either be, or inherit from, `NSManagedObject`.

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability
Available in iOS 3.0 and later.

See Also
- `managedObjectClassName` (page 12)

Declared In
`NSEntityDescription.h`

setName:

Sets the entity name of the receiver.

```
- (void)setName:(NSString *)name
```

Parameters

name

The name of the entity the receiver describes.

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability
Available in iOS 3.0 and later.

See Also
- `name` (page 12)

Declared In
`NSEntityDescription.h`

setProperties:

Sets the properties array of the receiver.

```
- (void)setProperties:(NSArray *)properties
```

Parameters

properties

An array of properties (instances of `NSAttributeDescription`, `NSRelationshipDescription`, and/or `NSFetchedPropertyDescription`).

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability
Available in iOS 3.0 and later.

See Also
- `properties` (page 13)

- propertiesByName (page 13)
- attributesByName (page 10)
- relationshipsByName (page 14)

Declared In
NSEntityDescription.h

setRenamingIdentifier:

Sets the renaming identifier for the receiver.

```
- (void)setRenamingIdentifier:(NSString *)value
```

Parameters

value

The renaming identifier for the receiver.

Availability
Available in iOS 3.0 and later.

See Also
- renamingIdentifier (page 14)

Declared In
NSEntityDescription.h

setSubentities:

Sets the subentities of the receiver.

```
- (void)setSubentities:(NSArray *)array
```

Parameters

array

An array containing sub-entities for the receiver. Objects in the array must be instances of NSEntityDescription.

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability
Available in iOS 3.0 and later.

See Also
- subentities (page 18)
- subentitiesByName (page 19)
- superentity (page 19)

Declared In
NSEntityDescription.h

setUserInfo:

Sets the user info dictionary of the receiver.

```
- (void)setUserInfo:(NSDictionary *)dictionary
```

Parameters

dictionary

A user info dictionary.

Special Considerations

This method raises an exception if the receiver's model has been used by an object graph manager.

Availability

Available in iOS 3.0 and later.

See Also

- userInfo (page 20)

Declared In

NSEntityDescription.h

setVersionHashModifier:

Sets the version hash modifier for the receiver.

```
- (void)setVersionHashModifier:(NSString *)modifierString
```

Parameters

modifierString

The version hash modifier for the receiver.

Discussion

This value is included in the version hash for the entity. You use it to mark or denote an entity as being a different "version" than another even if all of the values which affect persistence are equal. (Such a difference is important in cases where, for example, the structure of an entity is unchanged but the format or content of data has changed.)

Availability

Available in iOS 3.0 and later.

See Also

- versionHash (page 20)
- versionHashModifier (page 20)

Declared In

NSEntityDescription.h

subentities

Returns an array containing the sub-entities of the receiver.

```
- (NSArray *)subentities
```

Return Value
An array containing the receiver's sub-entities. The sub-entities are instances of `NSEntityDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `setSubentities:` (page 17)
- `subentitiesByName` (page 19)
- `superentity` (page 19)

Declared In
`NSEntityDescription.h`

subentitiesByName

Returns the sub-entities of the receiver in a dictionary.

```
- (NSDictionary *)subentitiesByName
```

Return Value
A dictionary containing the receiver's sub-entities. The keys in the dictionary are the sub-entity names, the corresponding values are instances of `NSEntityDescription`.

Availability
Available in iOS 3.0 and later.

See Also
- `setSubentities:` (page 17)
- `subentities` (page 18)
- `superentity` (page 19)

Declared In
`NSEntityDescription.h`

superentity

Returns the super-entity of the receiver.

```
- (NSEntityDescription *)superentity
```

Return Value
The receiver's super-entity. If the receiver has no super-entity, returns `nil`.

Availability
Available in iOS 3.0 and later.

See Also
- `setSubentities:` (page 17)
- `subentities` (page 18)
- `subentitiesByName` (page 19)

Declared In
NSEntityDescription.h

userInfo

Returns the user info dictionary of the receiver.

```
- (NSDictionary *)userInfo
```

Return Value
The receiver's user info dictionary.

Availability
Available in iOS 3.0 and later.

See Also
- setUserInfo: (page 18)

Declared In
NSEntityDescription.h

versionHash

Returns the version hash for the receiver.

```
- (NSData *)versionHash
```

Return Value
The version hash for the receiver.

Discussion
The version hash is used to uniquely identify an entity based on the collection and configuration of properties for the entity. The version hash uses only values which affect the persistence of data and the user-defined versionHashModifier (page 20) value. (The values which affect persistence are: the name of the entity, the version hash of the superentity (if present), if the entity is abstract, and all of the version hashes for the properties.) This value is stored as part of the version information in the metadata for stores which use this entity, as well as a definition of an entity involved in an NSEntityMapping object.

Availability
Available in iOS 3.0 and later.

See Also
- versionHashModifier (page 20)
- setVersionHashModifier: (page 18)

Declared In
NSEntityDescription.h

versionHashModifier

Returns the version hash modifier for the receiver.

```
- (NSString *)versionHashModifier
```

Return Value
The version hash modifier for the receiver.

Discussion
This value is included in the version hash for the entity. See `setVersionHashModifier:` (page 18) for a full discussion.

Availability
Available in iOS 3.0 and later.

See Also
- `versionHash` (page 20)
- `setVersionHashModifier:` (page 18)

Declared In
`NSEntityDescription.h`

Document Revision History

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

Date	Notes
2010-04-06	Corrected repeated text.
2009-05-01	Corrected typographical errors.
2009-02-19	Updated for iOS 3.0.
2008-06-02	Updated for Mac OS X v10.6.
2008-02-08	Updated the discussion of the method insertNewObjectForEntityForName: inManagedObjectContext: for Mac OS X v10.5.
2007-07-23	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History