NSManagedObjectModel Class Reference

Cocoa > Data Management

2007-01-26

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

Simultaneously published in the United States and Canada.

Contents

Tables

NSManagedObjectModel Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSFastEnumeration NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreData.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	NSManagedObjectModel.h
Companion guides	Core Data Programming Guide Core Data Utility Tutorial Core Data Model Versioning and Data Migration Programming Guide
Related sample code	Core Data HTML Store CoreRecipes

Overview

An `NSManagedObjectModel` object describes a schema—a collection of entities (data models) that you use in your application.

The model contains one or more `NSEntityDescription` objects representing the entities in the schema. Each `NSEntityDescription` object has property description objects (instances of subclasses of `NSPropertyDescription`) that represent the properties (or fields) of the entity in the schema. The Core Data framework uses this description in several ways:

- Constraining UI creation in Interface Builder
- Validating attribute and relationship values at runtime
- Mapping between your managed objects and a database or file-based schema for object persistence.

A managed object model maintains a mapping between each of its entity objects and a corresponding managed object class for use with the persistent storage mechanisms in the Core Data Framework. You can determine the entity for a particular managed object with the `entity` method.

You typically create managed object models using the data modeling tool in Xcode, but it is possible to build an model programmatically if needed.

Loading a Model File

Managed object model files are typically stored in a project or a framework. To load a model, you provide an URL to the constructor. Note that loading a model doesn't have the effect of loading all of its entities.

Stored Fetch Requests

It is often the case that in your application you want to get hold of a collection of objects that share features in common. Sometimes you can define those features (property values) in advance; sometimes you need to be able to supply values at runtime. For example, you might want to be able to retrieve all movies owned by Pixar; alternatively you might want to be able to retrieve all movies that earned more than an amount specified by the user at runtime.

Fetch requests are often predefined in a managed object model as templates. They allow you to pre-define named queries and their parameters in the model. Typically they contain variables that need to be substituted at runtime. `NSManagedObjectModel` provides API to retrieve a stored fetch request by name, and to perform variable substitution—see `fetchRequestTemplateForName:` (page 16) and `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15). You can create fetch request templates programmatically, and associate them with a model using `setFetchRequestTemplate:forName:` (page 19); typically, however, you define them using the Xcode design tool.

Configurations

Sometimes a model—particularly one in a framework—may be used in different situations, and you may want to specify different sets of entities to be used in different situations. There might, for example, be certain entities that should only be available if a user has administrative privileges. To support this requirement, a model may have more than one configuration. Each configuration is named, and has an associated set of entities. The sets may overlap. You establish configurations programmatically using `setEntities:forConfiguration:` (page 19) or using the Xcode design tool, and retrieve the entities for a given configuration name using `entitiesForConfiguration:` (page 14).

Changing Models

Since a model describes the structure of the data in a persistent store, changing any parts of a model that alters the schema renders it incompatible with (and so unable to open) the stores it previously created. If you change your schema, you therefore need to migrate the data in existing stores to new version (see Versioning in *Core Data Programming Guide*). For example, if you add a new entity or a new attribute to an existing entity, you will not be able to open old stores; if you add a validation constraint or set a new default value for an attribute, you will be able to open old stores.

Editing Models Programmatically

Managed object models are editable until they are used by an object graph manager (a managed object context or a persistent store coordinator). This allows you to create or modify them dynamically. However, once a model is being used, it *must not* be changed. This is enforced at runtime—when the object manager

first fetches data using a model, the whole of that model becomes uneditable. Any attempt to mutate a model or any of its sub-objects after that point 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.

Fast Enumeration

In Mac OS X v10.5 and later, `NSManagedObjectModel` supports the `NSFastEnumeration` protocol. You can use this to enumerate over a model's entities, as illustrated in the following example:

```
NSManagedObjectModel *aModel = ...;
for (NSEntityDescription *entity in aModel) {
    // entity is each instance of NSEntityDescription in aModel in turn
}
```

Tasks

Initializing a Model

`- initWithContentsOfURL:` (page 17)
Initializes the receiver using the model file at the specified URL.

`+ mergedModelFromBundles:` (page 10)
Returns a model created by merging all the models found in given bundles.

`+ mergedModelFromBundles:forStoreMetadata:` (page 11)
Returns a merged model from a specified array for the version information in provided metadata.

`+ modelByMergingModels:` (page 12)
Creates a single model from an array of existing models.

`+ modelByMergingModels:forStoreMetadata:` (page 12)
Returns, for the version information in given metadata, a model merged from a given array of models.

Entities and Configurations

`- entities` (page 13)
Returns the entities in the receiver.

`- entitiesByName` (page 14)
Returns the entities of the receiver in a dictionary.

`- setEntities:` (page 18)
Sets the entities array of the receiver.

`- configurations` (page 13)
Returns all the available configuration names of the receiver.

`- entitiesForConfiguration:` (page 14)
Returns the entities of the receiver for a specified configuration.

`- setEntities:forConfiguration:` (page 19)
Associates the specified entities with the receiver using the given configuration name.

Getting Fetch Request Templates

- `fetchRequestTemplatesByName` (page 16)
  Returns a dictionary of the receiver's fetch request templates.
- `fetchRequestTemplateForName:` (page 16)
  Returns the fetch request with a specified name.
- `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15)
  Returns a copy of the fetch request template with the variables substituted by values from the substitutions dictionary.
- `setFetchRequestTemplate:forName:` (page 19)
  Associates the specified fetch request with the receiver using the given name.

Localization

- `localizationDictionary` (page 18)
  Returns the localization dictionary of the receiver.
- `setLocalizationDictionary:` (page 20)
  Sets the localization dictionary of the receiver.

Versioning and Migration

- `isConfiguration:compatibleWithStoreMetadata:` (page 17)
  Returns a Boolean value that indicates whether a given configuration in the receiver is compatible with given metadata from a persistent store.
- `entityVersionHashesByName` (page 15)
  Returns a dictionary of the version hashes for the entities in the receiver.
- `versionIdentifiers` (page 21)
  Returns the collection of developer-defined version identifiers for the receiver.
- `setVersionIdentifiers:` (page 21)
  Sets the identifiers for the receiver.

Class Methods

mergedModelFromBundles:

Returns a model created by merging all the models found in given bundles.

```
+ (NSManagedObjectModel *)mergedModelFromBundles:(NSArray *)bundles
```

Parameters

bundles

An array of instances of `NSBundle` to search. If you specify `nil`, then the main bundle is searched.

Return Value
A model created by merging all the models found in *bundles*.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ mergedModelFromBundles:forStoreMetadata: (page 11)
+ modelByMergingModels: (page 12)
+ modelByMergingModels:forStoreMetadata: (page 12)
- initWithContentsOfURL: (page 17)

Related Sample Code
Core Data HTML Store
CoreRecipes

Declared In
NSManagedObjectModel.h

mergedModelFromBundles:forStoreMetadata:

Returns a merged model from a specified array for the version information in provided metadata.

```
+ (NSManagedObjectModel *)mergedModelFromBundles:(NSArray *)bundles
    forStoreMetadata:(NSDictionary *)metadata
```

Parameters

bundles
An array of bundles.

metadata
A dictionary containing version information from the metadata for a persistent store.

Return Value
The managed object model used to create the store for the metadata. If a model cannot be created to match the version information specified by *metadata*, returns nil.

Discussion
This method is a companion to mergedModelFromBundles: (page 10).

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

See Also
+ mergedModelFromBundles: (page 10)
+ modelByMergingModels: (page 12)
+ modelByMergingModels:forStoreMetadata: (page 12)
- initWithContentsOfURL: (page 17)

Declared In
NSManagedObjectModel.h

modelByMergingModels:

Creates a single model from an array of existing models.

```
+ (NSManagedObjectModel *)modelByMergingModels:(NSArray *)models
```

Parameters

models

An array of instances of `NSManagedObjectModel`.

Return Value

A single model made by combining the models in *models*.

Discussion

You use this method to combine multiple models (typically from different frameworks) into one.

Availability

Available in Mac OS X v10.4 and later.

See Also

`+ mergedModelFromBundles:` (page 10)

`+ mergedModelFromBundles:forStoreMetadata:` (page 11)

`+ modelByMergingModels:forStoreMetadata:` (page 12)

`- initWithContentsOfURL:` (page 17)

Declared In

`NSManagedObjectModel.h`

modelByMergingModels:forStoreMetadata:

Returns, for the version information in given metadata, a model merged from a given array of models.

```
+ (NSManagedObjectModel *)modelByMergingModels:(NSArray *)models
    forStoreMetadata:(NSDictionary *)metadata
```

Parameters

models

An array of instances of `NSManagedObjectModel`.

metadata

A dictionary containing version information from the metadata for a persistent store.

Return Value

A merged model from *models* for the version information in *metadata*. If a model cannot be created to match the version information in *metadata*, returns `nil`.

Discussion

This is the companion method to `mergedModelFromBundles:forStoreMetadata:` (page 11).

Availability

Available in Mac OS X v10.5 and later.

See Also

`+ mergedModelFromBundles:` (page 10)

`+ mergedModelFromBundles:forStoreMetadata:` (page 11)

+ modelByMergingModels: (page 12)
- initWithContentsOfURL: (page 17)

Declared In
NSManagedObjectModel.h

Instance Methods

configurations

Returns all the available configuration names of the receiver.

```
- (NSArray *)configurations
```

Return Value
An array containing the available configuration names of the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- entitiesForConfiguration: (page 14)
- setEntities:forConfiguration: (page 19)

Declared In
NSManagedObjectModel.h

entities

Returns the entities in the receiver.

```
- (NSArray *)entities
```

Return Value
An array containing the entities in the receiver.

Discussion
Entities are instances of NSEntityDescription.

Availability
Available in Mac OS X v10.4 and later.

See Also
- entitiesByName (page 14)
- entitiesForConfiguration: (page 14)
- setEntities: (page 18)
- setEntities:forConfiguration: (page 19)

Related Sample Code
CoreRecipes

Declared In
NSManagedObjectModel.h

entitiesByName

Returns the entities of the receiver in a dictionary.

```
- (NSDictionary *)entitiesByName
```

Return Value
The entities of the receiver in a dictionary, where the keys in the dictionary are the names of the corresponding entities.

Availability
Available in Mac OS X v10.4 and later.

See Also
- entities (page 13)
- entitiesForConfiguration: (page 14)
- setEntities: (page 18)
- setEntities:forConfiguration: (page 19)
+ entityForName:inManagedObjectContext: (NSEntityDescription)

Related Sample Code
Core Data HTML Store
CoreRecipes
CustomAtomicStoreSubclass

Declared In
NSManagedObjectModel.h

entitiesForConfiguration:

Returns the entities of the receiver for a specified configuration.

```
- (NSArray *)entitiesForConfiguration:(NSString *)configuration
```

Parameters
configuration
The name of a configuration in the receiver.

Return Value
An array containing the entities of the receiver for the configuration specified by *configuration*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- entities (page 13)
- entitiesByName (page 14)
- setEntities: (page 18)
- setEntities:forConfiguration: (page 19)

Declared In
NSManagedObjectModel.h

entityVersionHashesByName

Returns a dictionary of the version hashes for the entities in the receiver.

```
- (NSDictionary *)entityVersionHashesByName
```

Return Value
A dictionary of the version hashes for the entities in the receiver, keyed by entity name.

Discussion
The dictionary of version hash information is used by Core Data to determine schema compatibility.

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

See Also
- isConfiguration:compatibleWithStoreMetadata: (page 17)

Declared In
NSManagedObjectModel.h

fetchRequestFromTemplateWithName:substitutionVariables:

Returns a copy of the fetch request template with the variables substituted by values from the substitutions dictionary.

```
- (NSFetchRequest *)fetchRequestFromTemplateWithName:(NSString *)name
    substitutionVariables:(NSDictionary *)variables
```

Parameters

name

A string containing the name of a fetch request template.

variables

A dictionary containing key-value pairs where the keys are the names of variables specified in the template; the corresponding values are substituted before the fetch request is returned. The dictionary must provide values for all the variables in the template.

Return Value
A copy of the fetch request template with the variables substituted by values from *variables*.

Discussion
The *variables* dictionary must provide values for all the variables. If you want to test for a nil value, use [NSNull null].

This method provides the usual way to bind an “abstractly” defined fetch request template to a concrete fetch. For more details on using this method, see Creating Predicates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fetchRequestTemplatesByName` (page 16)
- `fetchRequestTemplateForName:` (page 16)
- `setFetchRequestTemplate:forName:` (page 19)

Declared In
`NSManagedObjectModel.h`

fetchRequestTemplateForName:

Returns the fetch request with a specified name.

```
- (NSFetchRequest *)fetchRequestTemplateForName:(NSString *)name
```

Parameters

name

A string containing the name of a fetch request template.

Return Value
The fetch request named *name*.

Discussion
If the template contains substitution variables, you should instead use `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15) to create a new fetch request.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fetchRequestTemplatesByName` (page 16)
- `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15)
- `setFetchRequestTemplate:forName:` (page 19)

Declared In
`NSManagedObjectModel.h`

fetchRequestTemplatesByName

Returns a dictionary of the receiver's fetch request templates.

```
- (NSDictionary *)fetchRequestTemplatesByName
```

Return Value
A dictionary of the receiver's fetch request templates, keyed by name.

Discussion
If the template contains a predicate with substitution variables, you should instead use `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15) to create a new fetch request.

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

See Also
- `fetchRequestTemplateForName:` (page 16)
- `fetchRequestFromTemplateWithName:substitutionVariables:` (page 15)

Declared In
`NSManagedObjectModel.h`

initWithContentsOfURL:

Initializes the receiver using the model file at the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)url
```

Parameters

url

An URL object specifying the location of a model file.

Return Value
A managed object model initialized using the file at *url*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `+ mergedModelFromBundles:` (page 10)
- `+ mergedModelFromBundles:forStoreMetadata:` (page 11)
- `+ modelByMergingModels:` (page 12)
- `+ modelByMergingModels:forStoreMetadata:` (page 12)

Declared In
`NSManagedObjectModel.h`

isConfiguration:compatibleWithStoreMetadata:

Returns a Boolean value that indicates whether a given configuration in the receiver is compatible with given metadata from a persistent store.

```
- (BOOL)isConfiguration:(NSString *)configuration
    compatibleWithStoreMetadata:(NSDictionary *)metadata
```

Parameters

configuration

The name of a configuration in the receiver. Pass `nil` to specify no configuration.

metadata

Metadata for a persistent store.

Return Value
`YES` if the configuration in the receiver specified by *configuration* is compatible with the store metadata given by *metadata*, otherwise `NO`.

Discussion
This method compares the version information in the store metadata with the entity versions of a given configuration. For information on specific differences, use `entityVersionHashesByName` (page 15) and perform an entity-by-entity comparison.

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

See Also
`- entityVersionHashesByName` (page 15)

Declared In
`NSManagedObjectModel.h`

localizationDictionary

Returns the localization dictionary of the receiver.

```
- (NSDictionary *)localizationDictionary
```

Return Value
The localization dictionary of the receiver.

Discussion
The key-value pattern is described in `setLocalizationDictionary:` (page 20).

Note that in the implementation in Mac OS X v10.4, `localizationDictionary` may return `nil` until Core Data lazily loads the dictionary for its own purposes (for example, reporting a localized error).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setLocalizationDictionary:` (page 20)

Declared In
`NSManagedObjectModel.h`

setEntities:

Sets the entities array of the receiver.

```
- (void)setEntities:(NSArray *)entities
```

Parameters
entities
An array of instances of `NSEntityDescription`.

Special Considerations
This method raises an exception if the receiver has been used by an object graph manager.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `entities` (page 13)
- `entitiesByName` (page 14)
- `entitiesForConfiguration:` (page 14)
- `setEntities:forConfiguration:` (page 19)

Declared In
`NSManagedObjectModel.h`

setEntities:forConfiguration:

Associates the specified entities with the receiver using the given configuration name.

```
- (void)setEntities:(NSArray *)entities forConfiguration:(NSString *)configuration
```

Parameters

entities
An array of instances of `NSEntityDescription`.

configuration
A name for the configuration.

Special Considerations

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

Availability
Available in Mac OS X v10.4 and later.

See Also
- `entities` (page 13)
- `entitiesByName` (page 14)
- `entitiesForConfiguration:` (page 14)
- `setEntities:` (page 18)

Declared In
`NSManagedObjectModel.h`

setFetchRequestTemplate:forName:

Associates the specified fetch request with the receiver using the given name.

```
- (void)setFetchRequestTemplate:(NSFetchRequest *)fetchRequest forName:(NSString
    *)name
```

Parameters

fetchRequest
A fetch request, typically containing predicates with variables for substitution.

name
A string that specifies the name of the fetch request template.

Discussion
For more details on using this method, see Creating Predicates.

Special Considerations

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

Availability

Available in Mac OS X v10.4 and later.

See Also

- `- fetchRequestTemplatesByName` (page 16)
- `- fetchRequestTemplateForName:` (page 16)
- `- fetchRequestFromTemplateWithName:substitutionVariables:` (page 15)

Declared In

`NSManagedObjectModel.h`

setLocalizationDictionary:

Sets the localization dictionary of the receiver.

```
- (void)setLocalizationDictionary:(NSDictionary *)localizationDictionary
```

Parameters

localizationDictionary

A dictionary containing localized string values for entities, properties, and error strings related to the model. The key and value pattern is described in Table 1 (page 20).

Discussion

Table 1 (page 20) describes the key and value pattern for the localization dictionary.

Table 1 Key and value pattern for the localization dictionary.

Key	Value	Note
"Entity/NonLocalizedEntityName"	"LocalizedEntityName"	
"Property/NonLocalizedPropertyName/Entity/EntityName"	"LocalizedPropertyName"	(1)
"Property/NonLocalizedPropertyName"	"LocalizedPropertyName"	
"ErrorString/NonLocalizedErrorString"	"LocalizedErrorString"	

(1) For properties in different entities with the same non-localized name but which should have different localized names.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `- localizationDictionary` (page 18)

Declared In

`NSManagedObjectModel.h`

setVersionIdentifiers:

Sets the identifiers for the receiver.

```
- (void)setVersionIdentifiers:(NSSet *)identifiers
```

Parameters

identifiers

An array of identifiers for the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- versionIdentifiers (page 21)

Declared In

NSManagedObjectModel.h

versionIdentifiers

Returns the collection of developer-defined version identifiers for the receiver.

```
- (NSSet *)versionIdentifiers
```

Return Value

The collection of developer-defined version identifiers for the receiver. Merged models return the combined collection of identifiers.

Discussion

The Core Data framework does not give models a default identifier, nor does it depend this value at runtime. For models created in Xcode, you set this value in the model inspector.

This value is meant to be used as a debugging hint to help you determine the models that were combined to create a merged model.

Availability

Available in Mac OS X v10.5 and later.

See Also

- setVersionIdentifiers: (page 21)

Declared In

NSManagedObjectModel.h

Document Revision History

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

Date	Notes
2007-01-26	Updated for Mac OS X v10.5
2006-12-05	Added a caveat regarding changing model schema.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index