NSAtomicStore 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

NSAtomicStore Class Reference

Inherits from	NSPersistentStore : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreData.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	NSAtomicStore.h
Companion guides	Atomic Store Programming Topics Core Data Programming Guide
Related sample code	Core Data HTML Store CustomAtomicStoreSubclass

Overview

`NSAtomicStore` is an abstract superclass that you can subclass to create a Core Data atomic store. It provides default implementations of some utility methods. A custom atomic store allows you to define a custom file format that integrates with a Core Data application.

The atomic stores are all intended to handle data sets that can be expressed in memory. The atomic store API favors simplicity over performance.

Subclassing Notes

Methods to Override

In a subclass of `NSAtomicStore`, you must override the following methods to provide behavior appropriate for your store:

`load:` (page 9)	Loads the cache nodes for the receiver.
`newReferenceObjectForManagedObject:` (page 11)	Returns a new reference object for a given managed object.
`save:` (page 13)	Saves the cache nodes.

`updateCacheNode:fromManagedObject:` (page 14)	Updates the given cache node using the values in a given managed object.

Note that these are in addition to the methods you must override for a subclass of `NSPersistentStore`:

`type`	Returns the type string of the receiver.
`identifier`	Returns the unique identifier for the receiver.
`setIdentifier:`	Sets the unique identifier for the receiver.
`metadata`	Returns the metadata for the receiver.
`metadataForPersistentStoreWithURL:error:`	Returns the metadata from the persistent store at the given URL.
`setMetadata:forPersistentStoreWithURL:error:`	Sets the metadata for the store at a given URL.

Tasks

Initializing a Store

- `initWithPersistentStoreCoordinator:configurationName:URL:options:` (page 8)
  Returns an atomic store, initialized with the given arguments.

Loading a Store

- `load:` (page 9)
  Loads the cache nodes for the receiver.
- `objectIDForEntity:referenceObject:` (page 12)
  Returns a managed object ID from the reference data for a specified entity.
- `addCacheNodes:` (page 7)
  Registers a set of cache nodes with the receiver.

Updating Cache Nodes

- `newCacheNodeForManagedObject:` (page 11)
  Returns a new cache node for a given managed object.
- `newReferenceObjectForManagedObject:` (page 11)
  Returns a new reference object for a given managed object.
- `updateCacheNode:fromManagedObject:` (page 14)
  Updates the given cache node using the values in a given managed object.
- `willRemoveCacheNodes:` (page 14)
  Method invoked before the store removes the given collection of cache nodes.

Saving a Store

- `save:` (page 13)
  Saves the cache nodes.

Utility Methods

- `cacheNodes` (page 8)
  Returns the set of cache nodes registered with the receiver.
- `cacheNodeForObjectID:` (page 8)
  Returns the cache node for a given managed object ID.
- `referenceObjectForObjectID:` (page 12)
  Returns the reference object for a given managed object ID.

Managing Metadata

- `metadata` (page 10)
  Returns the metadata for the receiver.
- `setMetadata:` (page 13)
  Sets the metadata for the receiver.

Instance Methods

addCacheNodes:

Registers a set of cache nodes with the receiver.

```
- (void)addCacheNodes:(NSSet *)cacheNodes
```

Parameters

cacheNodes
  A set of cache nodes.

Discussion
You should invoke this method in a subclass during the call to `load:` (page 9) to register the loaded information with the store.

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

Declared In
`NSAtomicStore.h`

cacheNodeForObjectID:

Returns the cache node for a given managed object ID.

```
- (NSAtomicStoreCacheNode *)cacheNodeForObjectID:(NSManagedObjectID *)objectID
```

Parameters

objectID

A managed object ID.

Return Value

The cache node for *objectID*.

Discussion

This method is normally used by cache nodes to locate related cache nodes (by relationships).

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

CustomAtomicStoreSubclass

Declared In

NSAtomicStore.h

cacheNodes

Returns the set of cache nodes registered with the receiver.

```
- (NSSet *)cacheNodes
```

Return Value

The set of cache nodes registered with the receiver.

Discussion

You should modify this collection using `addCacheNodes:` (page 7): and `willRemoveCacheNodes:` (page 14).

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

CustomAtomicStoreSubclass

Declared In

NSAtomicStore.h

initWithPersistentStoreCoordinator:configurationName:URL:options:

Returns an atomic store, initialized with the given arguments.

```
- (id)initWithPersistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    configurationName:(NSString *)configurationName
    URL:(NSURL *)url
    options:(NSDictionary *)options
```

Parameters

coordinator

A persistent store coordinator.

configurationName

The name of the managed object model configuration to use.

url

The URL of the store to load. This value must not be `nil`.

options

A dictionary containing configuration options.

Return Value

An atomic store, initialized with the given arguments, or `nil` if the store could not be initialized.

Discussion

You typically do not invoke this method yourself; it is invoked by the persistent store coordinator during `addPersistentStoreWithType:configuration:URL:options:error:`, both when a new store is created and when an existing store is opened.

In your implementation, you should check whether a file already exists at *url*; if it does not, then you should either create a file here or ensure that your `load:` (page 9) method does not fail if the file does not exist.

Any subclass of `NSAtomicStore` must be able to handle being initialized with a URL pointing to a zero-length file. This serves as an indicator that a new store is to be constructed at the specified location and allows you to securely create reservation files in known locations which can then be passed to Core Data to construct stores. You may choose to create zero-length reservation files during `initWithPersistentStoreCoordinator:configurationName:URL:options:` or `load:` (page 9). If you do so, you must remove the reservation file if the store is removed from the coordinator before it is saved.

You should ensure that you load metadata during initialization and set it using `setMetadata:` (page 13).

Special Considerations

You must invoke `super`'s implementation to ensure that the store is correctly initialized.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `load:` (page 9)
- `setMetadata:` (page 13)

Declared In

`NSAtomicStore.h`

load:

Loads the cache nodes for the receiver.

```
- (BOOL)load:(NSError **)error
```

Parameters

error

If an error occurs, upon return contains an `NSError` object that describes the problem.

Return Value

`YES` if the cache nodes were loaded correctly, otherwise `NO`.

Discussion

You override this method to to load the data from the URL specified in `initWithPersistentStoreCoordinator:configurationName:URL:options:` (page 8) and create cache nodes for the represented objects. You must respect the configuration specified for the store, as well as the options.

Any subclass of `NSAtomicStore` must be able to handle being initialized with a URL pointing to a zero-length file. This serves as an indicator that a new store is to be constructed at the specified location and allows you to securely create reservation files in known locations which can then be passed to Core Data to construct stores. You may choose to create zero-length reservation files during `initWithPersistentStoreCoordinator:configurationName:URL:options:` (page 8) or `load:`. If you do so, you must remove the reservation file if the store is removed from the coordinator before it is saved.

Special Considerations

You must override this method.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- addCacheNodes:` (page 7)

Declared In

`NSAtomicStore.h`

metadata

Returns the metadata for the receiver.

```
- (NSDictionary *)metadata
```

Return Value

The metadata for the receiver.

Discussion

`NSAtomicStore` provides a default dictionary of metadata. This dictionary contains the store type and identifier (`NSStoreTypeKey` and `NSStoreUUIDKey`) as well as store versioning information. Subclasses must ensure that the metadata is saved along with the store data.

See Also

`- metadata` (`NSPersistentStore`)

newCacheNodeForManagedObject:

Returns a new cache node for a given managed object.

```
- (NSAtomicStoreCacheNode *)newCacheNodeForManagedObject:(NSManagedObject
    *)managedObject
```

Parameters

managedObject

A managed object.

Return Value

A new cache node for *managedObject*.

Following normal rules for Cocoa memory management (see Memory Management Rules), the returned object has a retain count of 1.

Discussion

This method is invoked by the framework after a save operation on a managed object content, once for each newly-inserted `NSManagedObject` instance.

`NSAtomicStore` provides a default implementation that returns a suitable cache node. You can override this method to take the information from the managed object and return a cache node with a retain count of 1 (the node will be registered by the framework).

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSAtomicStore.h`

newReferenceObjectForManagedObject:

Returns a new reference object for a given managed object.

```
- (id)newReferenceObjectForManagedObject:(NSManagedObject *)managedObject
```

Parameters

managedObject

A managed object. At the time this method is called, it has a temporary ID.

Return Value

A new reference object for *managedObject*.

Following normal rules for Cocoa memory management (see Memory Management Rules), the returned object has a retain count of 1.

Discussion

This method is invoked by the framework after a save operation on a managed object context, once for each newly-inserted managed object. The value returned is used to create a permanent ID for the object and must be unique for an instance within its entity's inheritance hierarchy (in this store), and must have a retain count of 1.

Special Considerations

You must override this method.

This method must return a stable (unchanging) value for a given object, otherwise Save As and migration will not work correctly. This means that you can use arbitrary numbers, UUIDs, or other random values only if they are persisted with the raw data. If you cannot save the originally-assigned reference object with the data, then the method must derive the reference object from the managed object's values. For more details, see *Atomic Store Programming Topics*.

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

Declared In
`NSAtomicStore.h`

objectIDForEntity:referenceObject:

Returns a managed object ID from the reference data for a specified entity.

```
- (NSManagedObjectID *)objectIDForEntity:(NSEntityDescription *)entity
    referenceObject:(id)data
```

Parameters

entity
An entity description object.

data
Reference data for which the managed object ID is required.

Return Value
The managed object ID from the reference data for a specified entity

Discussion
You use this method to create managed object IDs which are then used to create cache nodes for information being loaded into the store.

Special Considerations

You should not override this method.

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

See Also
`- addCacheNodes:` (page 7)

Related Sample Code
CustomAtomicStoreSubclass

Declared In
`NSAtomicStore.h`

referenceObjectForObjectID:

Returns the reference object for a given managed object ID.

```
- (id)referenceObjectForObjectID:(NSManagedObjectID *)objectID
```

Parameters

objectID

A managed object ID.

Return Value

The reference object for *objectID*.

Discussion

Subclasses should invoke this method to extract the reference data from the object ID for each cache node if the data is to be made persistent.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSAtomicStore.h

save:

Saves the cache nodes.

```
- (BOOL)save:(NSError **)error
```

Parameters

error

If an error occurs, upon return contains an `NSError` object that describes the problem.

Discussion

You override this method to make persistent the necessary information from the cache nodes to the URL specified for the receiver.

Special Considerations

You must override this method.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `newReferenceObjectForManagedObject:` (page 11)
- `updateCacheNode:fromManagedObject:` (page 14)
- `willRemoveCacheNodes:` (page 14)

Declared In

NSAtomicStore.h

setMetadata:

Sets the metadata for the receiver.

```
- (void)setMetadata:(NSDictionary *)storeMetadata
```

Parameters

storeMetadata

The metadata for the receiver.

See Also

- metadata (page 10)

updateCacheNode:fromManagedObject:

Updates the given cache node using the values in a given managed object.

```
- (void)updateCacheNode:(NSAtomicStoreCacheNode *)node
    fromManagedObject:(NSManagedObject *)managedObject
```

Parameters

node

The cache node to update.

managedObject

The managed object with which to update *node*.

Discussion

This method is invoked by the framework after a save operation on a managed object context, once for each updated NSManagedObject instance.

You override this method in a subclass to take the information from *managedObject* and update *node*.

Special Considerations

You must override this method.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSAtomicStore.h

willRemoveCacheNodes:

Method invoked before the store removes the given collection of cache nodes.

```
- (void)willRemoveCacheNodes:(NSSet *)cacheNodes
```

Parameters

cacheNodes

The set of cache nodes to remove.

Discussion

This method is invoked by the store before the call to save: (page 13) with the collection of cache nodes marked as deleted by a managed object context. You can override this method to track the nodes which will not be made persistent in the save: (page 13) method.

You should not invoke this method directly in a subclass.

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

See Also
- `save:` (page 13)

Declared In
`NSAtomicStore.h`

Document Revision History

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

Date	Notes
2008-10-15	Corrected discussion of save: method.
2007-10-31	New document that describes the Core Data class used to represent an atomic persistent store.

Document Revision History

Index