
NSObjectController Class Reference

Data Management: Data Types & Collections





Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY

DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSObjectController Class Reference 5

Overview	5
Tasks	5
Initializing an Object Controller	5
Managing Content	6
Setting the Content Class	6
Managing Objects	6
Managing Editing	6
Core Data Support	7
Obtaining Selections	7
Validating User Interface Items	7
Instance Methods	8
add:	8
addObject:	8
automaticallyPreparesContent	9
canAdd	9
canRemove	10
content	10
defaultFetchRequest	11
entityName	11
fetch:	11
fetchPredicate	12
fetchWithRequest:merge:error:	12
initWithContent:	13
isEditable	13
managedObjectContext	13
newObject	14
objectClass	15
prepareContent	15
remove:	16
removeObject:	16
selectedObjects	17
selection	17
setAutomaticallyPreparesContent:	18
setContent:	18
setEditable:	18
setEntityName:	19
setFetchPredicate:	19
setManagedObjectContext:	20
setObjectClass:	20
setUsesLazyFetching:	21

usesLazyFetching 21
validateUserInterfaceItem: 21

Document Revision History 23

Index 25

NSObjectController Class Reference

Inherits from	NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.3 and later.
Declared in	NSObjectController.h
Companion guides	Cocoa Bindings Programming Topics Predicate Programming Guide Core Data Programming Guide
Related sample code	CIRAWFilterSample Departments and Employees DispatchFractal GridCalendar Simple Bindings Adoption

Overview

`NSObjectController` is a Cocoa bindings-compatible controller class. Properties of the content object of an instance of this class can be bound to user interface elements to access and modify their values.

By default, the content of an `NSObjectController` instance is an `NSMutableDictionary` object. This allows a single `NSObjectController` instance to be used to manage many different properties referenced by key value paths. The default content object class can be changed by calling `setObjectClass:` (page 20), which subclassers must override.

Tasks

Initializing an Object Controller

- `initWithContent:` (page 13)

Initializes and returns an `NSObjectController` object with the given content.

Managing Content

- [setContent:](#) (page 18)
Sets the receiver's content object.
- [content](#) (page 10)
Returns the receiver's content object.
- [setAutomaticallyPreparesContent:](#) (page 18)
Sets whether the receiver automatically creates and inserts new content objects automatically when loading from a nib file.
- [automaticallyPreparesContent](#) (page 9)
Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.
- [prepareContent](#) (page 15)
Typically overridden by subclasses that require additional control over the creation of new objects.

Setting the Content Class

- [setObjectClass:](#) (page 20)
Sets the object class to use when creating new objects.
- [objectClass](#) (page 15)
Returns the class used when creating new non-Core Data objects.

Managing Objects

- [newObject](#) (page 14)
Creates and returns a new object of the appropriate class.
- [addObject:](#) (page 8)
Sets the receiver's content object.
- [removeObject:](#) (page 16)
Removes a given object from the receiver's content.
- [add:](#) (page 8)
Creates a new object and sets it as the receiver's content object.
- [canAdd](#) (page 9)
Returns a Boolean value that indicates whether an object can be added to the receiver using [add:](#) (page 8).
- [remove:](#) (page 16)
Removes the receiver's content object.
- [canRemove](#) (page 10)
Returns a Boolean value that indicates whether an object can be removed from the receiver.

Managing Editing

- [setEditable:](#) (page 18)
Sets whether the receiver allows adding and removing objects.

- [isEditable](#) (page 13)
Returns a Boolean value that indicates whether the receiver allows adding and removing objects.

Core Data Support

- [entityName](#) (page 11)
Returns the entity name used by the receiver to create new objects.
- [setEntityName:](#) (page 19)
Sets the entity name used by the receiver to create new objects.
- [fetch:](#) (page 11)
Causes the receiver to fetch the data objects specified by the entity name and fetch predicate.
- [setUsesLazyFetching:](#) (page 21)
Sets whether the receiver uses lazy fetching.
- [usesLazyFetching](#) (page 21)
Returns a Boolean indicating whether the receiver uses lazy fetching.
- [defaultFetchRequest](#) (page 11)
Returns the default fetch request used by the receiver.
- [fetchPredicate](#) (page 12)
Returns the receiver's fetch predicate.
- [setFetchPredicate:](#) (page 19)
Sets the receiver's fetch predicate.
- [managedObjectContext](#) (page 13)
Returns the receiver's managed object context.
- [setManagedObjectContext:](#) (page 20)
Sets the receiver's managed object context.
- [fetchWithRequest:merge:error:](#) (page 12)
Subclasses should override this method to customize a fetch request, for example to specify fetch limits.

Obtaining Selections

- [selectedObjects](#) (page 17)
Returns an array of all objects to be affected by editing.
- [selection](#) (page 17)
Returns a proxy object representing the receiver's selection.

Validating User Interface Items

- [validateUserInterfaceItem:](#) (page 21)
Returns whether the receiver can handle the action method for a user interface item.

Instance Methods

add:

Creates a new object and sets it as the receiver's content object.

- (void)add:(id) *sender*

Parameters

sender

Typically the object that invoked this method.

Discussion

Creates a new object of the appropriate entity (specified by [entityName](#) (page 11)) or class (specified by [objectClass](#) (page 15))—see [newObject](#) (page 14)—and sets it as the receiver's content object using [addObject:](#) (page 8).

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [canAdd](#) (page 9)
- [remove:](#) (page 16)

Declared In

NSObjectController.h

addObject:

Sets the receiver's content object.

- (void)addObject:(id) *object*

Parameters

object

The content object for the receiver.

Discussion

If the receiver's content is bound to another object or controller through a relationship key, the relationship of the "master" object is changed.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [removeObject:](#) (page 16)

Related Sample Code

Departments and Employees

Declared In

NSObjectController.h

automaticallyPreparesContent

Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.

- (BOOL)automaticallyPreparesContent

Return Value

YES if the receiver automatically prepares its content when loaded from a nib, otherwise NO.

Discussion

See [setAutomaticallyPreparesContent:](#) (page 18) for a full explanation of "automatically prepares content."

The default is NO.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setAutomaticallyPreparesContent:](#) (page 18)
- [prepareContent](#) (page 15)

Declared In

NSObjectController.h

canAdd

Returns a Boolean value that indicates whether an object can be added to the receiver using [add:](#) (page 8).

- (BOOL)canAdd

Return Value

YES if an object can be added to the receiver using [add:](#) (page 8), otherwise NO.

Discussion

Bindings can use this method to control the enabling of user interface objects.

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [canRemove](#) (page 10)
- [add:](#) (page 8)

Declared In

NSObjectController.h

canRemove

Returns a Boolean value that indicates whether an object can be removed from the receiver.

- (BOOL)canRemove

Return Value

YES if an object can be removed from the receiver using [remove:](#) (page 16), otherwise NO.

Discussion

Bindings can use this method to control the enabling of user interface objects.

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [canAdd](#) (page 9)
- [remove:](#) (page 16)

Declared In

NSObjectController.h

content

Returns the receiver's content object.

- (id)content

Return Value

The receiver's content object.

Discussion

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setContent:](#) (page 18)

Related Sample Code

Aperture Edit Plugin - Borders & Titles

Declared In

NSObjectController.h

defaultFetchRequest

Returns the default fetch request used by the receiver.

- (NSFetchRequest *)defaultFetchRequest

Return Value

The default NSFetchedResult used by the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setUsesLazyFetching:](#) (page 21)
- [usesLazyFetching](#) (page 21)

Declared In

NSObjectController.h

entityName

Returns the entity name used by the receiver to create new objects.

- (NSString *)entityName

Return Value

The entity name used by the receiver to create new objects.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setEntityName:](#) (page 19)

Declared In

NSObjectController.h

fetch:

Causes the receiver to fetch the data objects specified by the entity name and fetch predicate.

- (void)fetch:(id)sender

Parameters

sender

Typically the object that invoked this method.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setFetchPredicate:](#) (page 19)
- [fetchPredicate](#) (page 12)

Declared In

NSObjectController.h

fetchPredicate

Returns the receiver's fetch predicate.

```
- (NSPredicate *)fetchPredicate
```

Return Value

The receiver's fetch predicate.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [fetch:](#) (page 11)
- [setFetchPredicate:](#) (page 19)

Declared In

NSObjectController.h

fetchWithRequest:merge:error:

Subclasses should override this method to customize a fetch request, for example to specify fetch limits.

```
- (BOOL)fetchWithRequest:(NSFetchRequest *)fetchRequest merge:(BOOL)merge
    error:(NSError **)error
```

Parameters

fetchRequest

The fetch request to use for the fetch. Pass `nil` to use the default fetch request.

merge

If YES, the receiver merges the existing content with the fetch result, otherwise the receiver replaces the entire content with the fetch result.

error

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

Return Value

YES if the fetch completed successfully, otherwise NO.

Discussion

This method performs a number of actions that you cannot reproduce. To customize this method, you should therefore create your own fetch request and then invoke `super`'s implementation with the new fetch request.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [fetch:](#) (page 11)

Declared In

NSObjectController.h

initWithContent:

Initializes and returns an NSObjectController object with the given content.

- (id)initWithContent:(id)content

Parameters

content

The content for the receiver.

Return Value

The initialized object controller, with its content object set to *content*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSObjectController.h

isEditable

Returns a Boolean value that indicates whether the receiver allows adding and removing objects.

- (BOOL)isEditable

Return Value

YES if the receiver allows adding and removing objects, otherwise NO.

Discussion

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setEditable:](#) (page 18)

Declared In

NSObjectController.h

managedObjectContext

Returns the receiver's managed object context.

- (NSManagedObjectContext *)managedObjectContext

Return Value

The receiver's managed object context.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setManagedObjectContext:](#) (page 20)

Related Sample Code

Departments and Employees

Declared In

NSObjectController.h

newObject

Creates and returns a new object of the appropriate class.

```
- (id)newObject
```

Return Value

A new object of the appropriate class. The returned object is implicitly retained, the sender is responsible for releasing it (with either `release` or `autorelease`).

If an entity name is set (see [setEntityName:](#) (page 19)), the object created is an instance of the class specified for that entity (and the object is inserted into the receiver's managed object context). Otherwise the object created is an instance of the class returned by [objectClass](#) (page 15).

Discussion

This method is called when adding and inserting objects if [automaticallyPreparesContent](#) (page 9) is YES.

The default implementation assumes the class returned by [objectClass](#) (page 15) has a standard `init` method without arguments. If the object class being controlled is `NSManagedObject` (or a subclass thereof) its designated initializer (`initWithEntity:insertIntoManagedObjectContext:`) is called instead, using the entity and managed object context specified for the receiver.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setObjectClass:](#) (page 20)
 - [objectClass](#) (page 15)
 - [setEntityName:](#) (page 19)
 - [entityName](#) (page 11)

Related Sample Code

DemoMonkey

Departments and Employees

With and Without Bindings

Declared In

NSObjectController.h

objectClass

Returns the class used when creating new non-Core Data objects.

- (Class)objectClass

Return Value

The object class used when creating new non-Core Data objects (that is, if no entity has been set)—see [newObject](#) (page 14).

Discussion

If an entity has been set, then the class returned by this method does not automatically reflect the class for the entity.

The default class is `NSMutableDictionary`.

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setObjectClass:](#) (page 20)
- [entityName](#) (page 11)
- [managedObjectContext](#) (page 13)

Declared In

NSObjectController.h

prepareContent

Typically overridden by subclasses that require additional control over the creation of new objects.

- (void)prepareContent

Discussion

Subclasses that implement this method are responsible for creating the new content object and setting it as the receiver's content object. This method is only called if [automaticallyPreparesContent](#) (page 9) has been set to YES.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [automaticallyPreparesContent](#) (page 9)
- [setAutomaticallyPreparesContent:](#) (page 18)

Related Sample Code

QTMetadataEditor

Declared In

NSObjectController.h

remove:

Removes the receiver's content object.

- (void)remove:(id)sender

Parameters*sender*

Typically the object that invoked this method.

DiscussionRemoves the receiver's content object using [removeObject:](#) (page 16).**Special Considerations**

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [canRemove](#) (page 10)
- [add:](#) (page 8)

Declared In

NSObjectController.h

removeObject:

Removes a given object from the receiver's content.

- (void)removeObject:(id)object

Parameters*object*

The object to remove from the receiver.

DiscussionIf *object* is the receiver's content object, the receiver's content is set to `nil`. If the receiver's content is bound to another object or controller through a relationship key, the relationship of the 'master' object is cleared.**Availability**

Available in Mac OS X v10.3 and later.

See Also

- [addObject:](#) (page 8)

Declared In

NSObjectController.h

selectedObjects

Returns an array of all objects to be affected by editing.

- (NSArray *)selectedObjects

Return Value

An array of all objects to be affected by editing. If the receiver supports a selection mechanism, the array contains key value coding compliant proxies of the selected objects; otherwise proxies for all content objects are returned. If the receiver is a concrete instance of `NSObjectController`, returns an array containing the receiver's content object.

Discussion

You should avoid registering for key-value observing changes for key paths that pass *through* this method, (for example, `selectedObjects.firstName`). Using the proxy returned by the [selection](#) (page 17) method is better for performance.

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [selection](#) (page 17)

Declared In

`NSObjectController.h`

selection

Returns a proxy object representing the receiver's selection.

- (id)selection

Return Value

A proxy object representing the receiver's selection. This object is fully key-value coding compliant, but note that it is a proxy and so does not provide the full range of functionality that might be available in the source object.

Discussion

If a value requested from the selection proxy using key-value coding returns multiple objects, the controller has no selection, or the proxy is not key-value coding compliant for the requested key, the appropriate marker (`NSMultipleValuesMarker`, `NSNoSelectionMarker` or `NSNotApplicableMarker`) is returned. Otherwise, the value of the key is returned.

This property is observable using key-value observing.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [selectedObjects](#) (page 17)

Declared In

`NSObjectController.h`

setAutomaticallyPreparesContent:

Sets whether the receiver automatically creates and inserts new content objects automatically when loading from a nib file.

- (void)setAutomaticallyPreparesContent:(BOOL)*flag*

Parameters

flag

A flag that specifies whether the receiver automatically prepares its content.

Discussion

If *flag* is YES and the receiver is not using a managed object context, [prepareContent](#) (page 15) is used to create the content object. If *flag* is YES and a managed object context is set, the initial content is fetched from the managed object context using the current fetch predicate. The default is NO.

Setting *flag* to YES is the same as checking the “Automatically Prepares Content” option in the Interface Builder controller inspector.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [automaticallyPreparesContent](#) (page 9)
- [prepareContent](#) (page 15)

Declared In

NSObjectController.h

setContent:

Sets the receiver’s content object.

- (void)setContent:(id)*content*

Parameters

content

The content object for the receiver.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [content](#) (page 10)

Declared In

NSObjectController.h

setEditable:

Sets whether the receiver allows adding and removing objects.

- (void)setEditable:(BOOL)*flag*

Parameters*flag*

YES if the the receiver should allow adding and removing objects, otherwise NO.

Discussion

The default is YES.

Availability

Available in Mac OS X v10.3 and later.

See Also- [isEditable](#) (page 13)**Declared In**

NSObjectController.h

setEntityName:

Sets the entity name used by the receiver to create new objects.

- (void)setEntityName:(NSString *)*entityName***Parameters***entityName*

The entity name used by the receiver to create new objects.

Availability

Available in Mac OS X v10.4 and later.

See Also- [entityName](#) (page 11)**Declared In**

NSObjectController.h

setFetchPredicate:

Sets the receiver's fetch predicate.

- (void)setFetchPredicate:(NSPredicate *)*predicate***Parameters***predicate*

The fetch predicate for the receiver.

DiscussionThe receiver uses *predicate* when fetching its content, for example in [fetch:](#) (page 11). If you need to customize the fetching behavior further, you can override [fetchWithRequest:merge:error:](#) (page 12).**Availability**

Available in Mac OS X v10.4 and later.

See Also- [fetch:](#) (page 11)

- [fetchPredicate](#) (page 12)
- [fetchWithRequest:merge:error:](#) (page 12)

Declared In

NSObjectController.h

setManagedObjectContext:

Sets the receiver's managed object context.

- (void)setManagedObjectContext:(NSManagedObjectContext *)*managedObjectContext*

Parameters

managedObjectContext

The managed object context for the receiver.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [managedObjectContext](#) (page 13)

Declared In

NSObjectController.h

setObjectClass:

Sets the object class to use when creating new objects.

- (void)setObjectClass:(Class)*objectClass*

Parameters

objectClass

The object class to use when creating new objects.

Discussion

NSObjectController's default implementation assumes that instances of *objectClass* are initialized using a standard `init` method that takes no arguments.

If an entity name has been set (see [setEntityName:](#) (page 19)), this method has no effect.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [objectClass](#) (page 15)
- [setEntityName:](#) (page 19)
- [managedObjectContext](#) (page 13)

Declared In

NSObjectController.h

setUsesLazyFetching:

Sets whether the receiver uses lazy fetching.

- (void)setUsesLazyFetching:(BOOL)*enabled*

Parameters

enabled

Boolean value that indicates whether the receiver uses lazy fetching.

Discussion

When enabled the controller uses a number of techniques that typically make managing large data sets more efficient. As with all optimizations, you should use suitable performance analysis tools (such as Instruments) to determine the best solution.

Note: Setting `setUsesLazyFetching:` to YES will cause an exception if the receiving controller is not bound to a managed object context.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [defaultFetchRequest](#) (page 11)
- [usesLazyFetching](#) (page 21)

Declared In

NSObjectController.h

usesLazyFetching

Returns a Boolean indicating whether the receiver uses lazy fetching.

- (BOOL)usesLazyFetching

Return Value

YES if the receiver uses lazy fetching, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [defaultFetchRequest](#) (page 11)
- [setUsesLazyFetching:](#) (page 21)

Declared In

NSObjectController.h

validateUserInterfaceItem:

Returns whether the receiver can handle the action method for a user interface item.

- (BOOL)validateUserInterfaceItem:(id < NSValidatedUserInterfaceItem >) *item*

Parameters

item

The user interface item to validate. You can send *item* the `action` and `tag` messages.

Return Value

YES if the receiver can handle the action method; NO if it cannot.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSObjectController.h

Document Revision History

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

Date	Notes
2007-12-11	Updated newObject description with more memory information. Added exception warning to setUsesLazyFetching:.
2007-05-16	Updated to include new API introduced in Mac OS X v10.5. Corrected default value of automaticallyPreparesContent:.
2006-07-24	Clarified use of fetchWithRequest:merge:error:.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

add: [instance method 8](#)
addObject: [instance method 8](#)
automaticallyPreparesContent [instance method 9](#)

C

canAdd [instance method 9](#)
canRemove [instance method 10](#)
content [instance method 10](#)

D

defaultFetchRequest [instance method 11](#)

E

entityName [instance method 11](#)

F

fetch: [instance method 11](#)
fetchPredicate [instance method 12](#)
fetchWithRequest:merge:error: [instance method 12](#)

I

initWithContent: [instance method 13](#)
isEditable [instance method 13](#)

M

managedObjectContext [instance method 13](#)

N

newObject [instance method 14](#)

O

objectClass [instance method 15](#)

P

prepareContent [instance method 15](#)

R

remove: [instance method 16](#)
removeObject: [instance method 16](#)

S

selectedObjects [instance method 17](#)
selection [instance method 17](#)
setAutomaticallyPreparesContent: [instance method 18](#)
setContent: [instance method 18](#)
setEditable: [instance method 18](#)
setEntityName: [instance method 19](#)
setFetchPredicate: [instance method 19](#)
setManagedObjectContext: [instance method 20](#)
setObjectClass: [instance method 20](#)
setUsesLazyFetching: [instance method 21](#)

U

usesLazyFetching **instance method** [21](#)

V

validateUserInterfaceItem: **instance method** [21](#)