NSKeyValueObserving Protocol Reference

Cocoa > Data Management

2007-10-31

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

Simultaneously published in the United States and Canada.

Contents

NSKeyValueObserving Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/Foundation.framework
Companion guide	Key-Value Observing Programming Guide
Declared in	NSKeyValueObserving.h

Overview

The `NSKeyValueObserving` (KVO) informal protocol defines a mechanism that allows objects to be notified of changes to the specified properties of other objects.

You can observe any object properties including simple attributes, to-one relationships, and to-many relationships. Observers of to-many relationships are informed of the type of change made — as well as which objects are involved in the change.

`NSObject` provides an implementation of the `NSKeyValueObserving` protocol that provides an automatic observing capability for all objects. You can further refine notifications by disabling automatic observer notifications and implementing manual notifications using the methods in this protocol.

Note: Key-value observing is not available for Java applications.

Tasks

Change Notification

- `observeValueForKeyPath:ofObject:change:context:` (page 10)
  This message is sent to the receiver when the value at the specified key path relative to the given object has changed.

Registering for Observation

- `addObserver:forKeyPath:options:context:` (page 8)
  Registers *anObserver* to receive KVO notifications for the specified key-path relative to the receiver.
- `removeObserver:forKeyPath:` (page 11)
  Stops a given object from receiving change notifications for the property specified by a given key-path relative to the receiver.

Notifying Observers of Changes

`- willChangeValueForKey:` (page 13)
Invoked to inform the receiver that the value of a given property is about to change.

`- didChangeValueForKey:` (page 9)
Invoked to inform the receiver that the value of a given property has changed.

`- willChange:valuesAtIndexes:forKey:` (page 12)
Invoked to inform the receiver that the specified change is about to be executed at given indexes for a specified ordered to-many relationship.

`- didChange:valuesAtIndexes:forKey:` (page 8)
Invoked to inform the receiver that the specified change has occurred on the indexes for a specified ordered to-many relationship.

`- willChangeValueForKey:withSetMutation:usingObjects:` (page 13)
Invoked to inform the receiver that the specified change is about to be made to a specified unordered to-many relationship.

`- didChangeValueForKey:withSetMutation:usingObjects:` (page 9)
Invoked to inform the receiver that the specified change was made to a specified unordered to-many relationship.

Observing Customization

`+ automaticallyNotifiesObserversForKey:` (page 6)
Returns a Boolean value that indicates whether the receiver supports automatic key-value observation for the given key.

`+ keyPathsForValuesAffectingValueForKey:` (page 7)
Returns a set of key paths for properties whose values affect the value of the specified key.

`- setObservationInfo:` (page 12)
Sets the observation info for the receiver.

`- observationInfo` (page 10)
Returns a pointer that identifies information about all of the observers that are registered with the receiver.

`+ setKeys:triggerChangeNotificationsForDependentKey:` (page 19) Deprecated in Mac OS X v10.5 and later
Configures the receiver to post change notifications for a given property if any of the properties specified in a given array changes. (Deprecated. You should use the method `keyPathsForValuesAffectingValueForKey:` (page 7) instead.)

Class Methods

automaticallyNotifiesObserversForKey:

Returns a Boolean value that indicates whether the receiver supports automatic key-value observation for the given key.

```
+ (BOOL)automaticallyNotifiesObserversForKey:(NSString *)key
```

Return Value
`YES` if the key-value observing machinery should automatically invoke `willChangeValueForKey:` (page 13)/`didChangeValueForKey:` (page 9) and `willChange:valuesAtIndexes:forKey:` (page 12)/`didChange:valuesAtIndexes:forKey:` (page 8) whenever instances of the class receive key-value coding messages for the *key*, or mutating key-value-coding-compliant methods for the *key* are invoked; otherwise `NO`.

Discussion
The default implementation returns `YES`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSKeyValueObserving.h`

keyPathsForValuesAffectingValueForKey:

Returns a set of key paths for properties whose values affect the value of the specified key.

```
+ (NSSet *)keyPathsForValuesAffectingValueForKey:(NSString *)key
```

Parameters

key

The key whose value is affected by the key paths.

Return Value

Discussion
When an observer for the key is registered with an instance of the receiving class, key-value observing itself automatically observes all of the key paths for the same instance, and sends change notifications for the key to the observer when the value for any of those key paths changes.

The default implementation of this method searches the receiving class for a method whose name matches the pattern `+keyPathsForValuesAffecting<Key>`, and returns the result of invoking that method if it is found. Any such method must return an NSSet. If no such method is found, an NSSet that is computed from information provided by previous invocations of the now-deprecated `setKeys:triggerChangeNotificationsForDependentKey:` (page 19) method is returned, for backward binary compatibility.

You can override this method when the getter method of one of your properties computes a value to return using the values of other properties, including those that are located by key paths. Your override should typically invoke super and return a set that includes any members in the set that result from doing that (so as not to interfere with overrides of this method in superclasses).

Note: You must not override this method when you add a computed property to an existing class using a category, overriding methods in categories is unsupported. In that case, implement a matching `+keyPathsForValuesAffecting<Key>` to take advantage of this mechanism.

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

Declared In
`NSKeyValueObserving.h`

Instance Methods

addObserver:forKeyPath:options:context:

Registers *anObserver* to receive KVO notifications for the specified key-path relative to the receiver.

```
- (void)addObserver:(NSObject *)anObserver
    forKeyPath:(NSString *)keyPath
    options:(NSKeyValueObservingOptions)options
    context:(void *)context
```

Parameters

anObserver

The object to register for KVO notifications. The observer must implement the key-value observing method `observeValueForKeyPath:ofObject:change:context:` (page 10).

keyPath

The key path, relative to the receiver, of the property to observe. This value must not be `nil`.

options

A combination of the `NSKeyValueObservingOptions` values that specifies what is included in observation notifications. For possible values, see NSKeyValueObservingOptions (page 15).

context

Arbitrary data that is passed to *anObserver* in observeValueForKeyPath:ofObject:change:context: (page 10).

Discussion
Neither the receiver, nor *anObserver*, are retained.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- removeObserver:forKeyPath:` (page 11)

Declared In
`NSKeyValueObserving.h`

didChange:valuesAtIndexes:forKey:

Invoked to inform the receiver that the specified change has occurred on the indexes for a specified ordered to-many relationship.

```
- (void)didChange:(NSKeyValueChange)change
    valuesAtIndexes:(NSIndexSet *)indexes
    forKey:(NSString *)key
```

Parameters

change

The type of change that was made.

indexes

The indexes of the to-many relationship that were affected by the change.

key

The name of a property that is an ordered to-many relationship.

Discussion

You should invoke this method when implementing key-value-observing compliance manually.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `willChange:valuesAtIndexes:forKey:` (page 12)
- `didChangeValueForKey:` (page 9)

Declared In

`NSKeyValueObserving.h`

didChangeValueForKey:

Invoked to inform the receiver that the value of a given property has changed.

```
- (void)didChangeValueForKey:(NSString *)key
```

Parameters

key

The name of the property that changed.

Discussion

You should invoke this method when implementing key-value observer compliance manually.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `willChangeValueForKey:` (page 13)
- `didChange:valuesAtIndexes:forKey:` (page 8)

Declared In

`NSKeyValueObserving.h`

didChangeValueForKey:withSetMutation:usingObjects:

Invoked to inform the receiver that the specified change was made to a specified unordered to-many relationship.

```
- (void)didChangeValueForKey:(NSString *)key
    withSetMutation:(NSKeyValueSetMutationKind)mutationKind
    usingObjects:(NSSet *)objects
```

Parameters

key

The name of a property that is an unordered to-many relationship

mutationKind

The type of change that was made.

objects

The objects that were involved in the change (see NSKeyValueSetMutationKind (page 17)).

Discussion
You invoke this method when implementing key-value observer compliance manually.

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

See Also
`- willChangeValueForKey:withSetMutation:usingObjects:` (page 13)

Declared In
`NSKeyValueObserving.h`

observationInfo

Returns a pointer that identifies information about all of the observers that are registered with the receiver.

```
- (void *)observationInfo
```

Return Value
A pointer that identifies information about all of the observers that are registered with the receiver, the options that were used at registration-time, and so on.

Discussion
The default implementation of this method retrieves the information from a global dictionary keyed by the receiver's pointers.

For improved performance, this method and `setObservationInfo:` can be overridden to store the opaque data pointer in an instance variable. Overrides of this method must not attempt to send Objective-C messages to the stored data, including `retain` and `release`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setObservationInfo:` (page 12)

Declared In
`NSKeyValueObserving.h`

observeValueForKeyPath:ofObject:change:context:

This message is sent to the receiver when the value at the specified key path relative to the given object has changed.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
    ofObject:(id)object
    change:(NSDictionary *)change
    context:(void *)context
```

Parameters

keyPath

The key path, relative to *object*, to the value that has changed.

object

The source object of the key path *keyPath*.

change

A dictionary that describes the changes that have been made to the value of the property at the key path *keyPath* relative to *object*. Entries are described in "Keys used by the change dictionary" (page 16).

context

The value that was provided when the receiver was registered to receive key-value observation notifications.

Discussion

The receiver must be registered as an observer for the specified *keyPath* and *object*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSKeyValueObserving.h

removeObserver:forKeyPath:

Stops a given object from receiving change notifications for the property specified by a given key-path relative to the receiver.

```
- (void)removeObserver:(NSObject *)anObserver
    forKeyPath:(NSString *)keyPath
```

Parameters

anObserver

The object to remove as an observer.

keyPath

A key-path, relative to the receiver, for which *anObserver* is registered to receive KVO change notifications.

Availability

Available in Mac OS X v10.3 and later.

See Also

- addObserver:forKeyPath:options:context: (page 8)

Related Sample Code

Departments and Employees

Declared In

NSKeyValueObserving.h

setObservationInfo:

Sets the observation info for the receiver.

```
- (void)setObservationInfo:(void *)observationInfo
```

Parameters

observationInfo

The observation info for the receiver.

Discussion

The *observationInfo* is a pointer that identifies information about all of the observers that are registered with the receiver. The default implementation of this method stores *observationInfo* in a global dictionary keyed by the receiver's pointers.

For improved performance, this method and `observationInfo` can be overridden to store the opaque data pointer in an instance variable. Classes that override this method must not attempt to send Objective-C messages to *observationInfo*, including `retain` and `release`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- observationInfo` (page 10)

Declared In

`NSKeyValueObserving.h`

willChange:valuesAtIndexes:forKey:

Invoked to inform the receiver that the specified change is about to be executed at given indexes for a specified ordered to-many relationship.

```
- (void)willChange:(NSKeyValueChange)change
    valuesAtIndexes:(NSIndexSet *)indexes
    forKey:(NSString *)key
```

Parameters

change

The type of change that is about to be made.

indexes

The indexes of the to-many relationship that will be affected by the change.

key

The name of a property that is an ordered to-many relationship.

Discussion

You should invoke this method when implementing key-value-observing compliance manually.

Important: After the values have been changed, a corresponding `didChange:valuesAtIndexes:forKey:` (page 8) must be invoked with the same parameters.

Availability

Available in Mac OS X v10.3 and later.

See Also
- `didChange:valuesAtIndexes:forKey:` (page 8)
- `willChangeValueForKey:` (page 13)

Declared In
`NSKeyValueObserving.h`

willChangeValueForKey:

Invoked to inform the receiver that the value of a given property is about to change.

```
- (void)willChangeValueForKey:(NSString *)key
```

Parameters

key

The name of the property that will change.

Discussion

You should invoke this method when implementing key-value observer compliance manually.

The change type of this method is `NSKeyValueChangeSetting`.

Important: After the values have been changed, a corresponding `didChangeValueForKey:` (page 9) must be invoked with the same parameter.

Availability

Available in Mac OS X v10.3 and later.

See Also
- `didChangeValueForKey:` (page 9)
- `willChange:valuesAtIndexes:forKey:` (page 12)

Declared In
`NSKeyValueObserving.h`

willChangeValueForKey:withSetMutation:usingObjects:

Invoked to inform the receiver that the specified change is about to be made to a specified unordered to-many relationship.

```
- (void)willChangeValueForKey:(NSString *)key
    withSetMutation:(NSKeyValueSetMutationKind)mutationKind
    usingObjects:(NSSet *)objects
```

Parameters

key

The name of a property that is an unordered to-many relationship

mutationKind

The type of change that will be made.

objects

The objects that are involved in the change (see NSKeyValueSetMutationKind (page 17)).

Discussion
You invoke this method when implementing key-value observer compliance manually.

Important: After the values have been changed, a corresponding `didChangeValueForKey:withSetMutation:usingObjects:` (page 9) must be invoked with the same parameters.

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

See Also
`- didChangeValueForKey:withSetMutation:usingObjects:` (page 9)

Declared In
`NSKeyValueObserving.h`

Constants

NSKeyValueChange

These constants are returned as the value for a `NSKeyValueChangeKindKey` key in the change dictionary passed to `observeValueForKeyPath:ofObject:change:context:` (page 10) indicating the type of change made:

```
enum {
    NSKeyValueChangeSetting = 1,
    NSKeyValueChangeInsertion = 2,
    NSKeyValueChangeRemoval = 3,
    NSKeyValueChangeReplacement = 4
};
typedef NSUInteger NSKeyValueChange;
```

Constants

`NSKeyValueChangeSetting`

Indicates that the value of the observed key path was set to a new value. This change can occur when observing an attribute of an object, as well as properties that specify to-one and to-many relationships.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeInsertion`

Indicates that an object has been inserted into the to-many relationship that is being observed.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeRemoval`
Indicates that an object has been removed from the to-many relationship that is being observed.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeReplacement`
Indicates that an object has been replaced in the to-many relationship that is being observed.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

Declared In
`NSKeyValueObserving.h`

NSKeyValueObservingOptions

These constants are passed to `addObserver:forKeyPath:options:context:` (page 8) and determine the values that are returned as part of the change dictionary passed to an `observeValueForKeyPath:ofObject:change:context:` (page 10). You can pass `0` if you require no change dictionary values.

```
enum {
    NSKeyValueObservingOptionNew = 0x01,
    NSKeyValueObservingOptionOld = 0x02,
    NSKeyValueObservingOptionInitial = 0x04,
    NSKeyValueObservingOptionPrior = 0x08
};
typedef NSUInteger NSKeyValueObservingOptions;
```

Constants
`NSKeyValueObservingOptionNew`
Indicates that the change dictionary should provide the new attribute value, if applicable.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueObservingOptionOld`
Indicates that the change dictionary should contain the old attribute value, if applicable.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueObservingOptionInitial`
If specified, a notification should be sent to the observer immediately, before the observer registration method even returns. The change dictionary in the notification will always contain an `NSKeyValueChangeNewKey` entry if `NSKeyValueObservingOptionNew` is also specified but will never contain an `NSKeyValueChangeOldKey` entry. (In an initial notification the current value of the observed property may be old, but it's new to the observer.) You can use this option instead of explicitly invoking, at the same time, code that is also invoked by the observer's `observeValueForKeyPath:ofObject:change:context:` method. When this option is used with`addObserver:forKeyPath:options:context:` a notification will be sent for each indexed object to which the observer is being added.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueObservingOptionPrior`

Whether separate notifications should be sent to the observer before and after each change, instead of a single notification after the change. The change dictionary in a notification sent before a change always contains an `NSKeyValueChangeNotificationIsPriorKey` entry whose value is `[NSNumber numberWithBool:YES]`, but never contains an `NSKeyValueChangeNewKey` entry. When this option is specified the change dictionary in a notification sent after a change contains the same entries that it would contain if this option were not specified. You can use this option when the observer's own key-value observing-compliance requires it to invoke one of the `-willChange...` methods for one of its own properties, and the value of that property depends on the value of the observed object's property. (In that situation it's too late to easily invoke `-willChange...` properly in response to receiving an `observeValueForKeyPath:ofObject:change:context:` message after the change.)

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueObserving.h`.

Declared In

`NSKeyValueObserving.h`

Keys used by the change dictionary

These constants are used as keys in the change dictionary passed to `observeValueForKeyPath:ofObject:change:context:` (page 10).

```
NSString *const NSKeyValueChangeKindKey;
NSString *const NSKeyValueChangeNewKey;
NSString *const NSKeyValueChangeOldKey;
NSString *const NSKeyValueChangeIndexesKey;
```

Constants

`NSKeyValueChangeKindKey`

An `NSNumber` object that contains a value corresponding to one of the `NSKeyValueChangeKindKey` enumerations, indicating what sort of change has occurred.

A value of `NSKeyValueChangeSetting` indicates that the observed object has received a `setValue:forKey:` message, or that the key-value-coding-compliant set method for the key has been invoked, or that `willChangeValueForKey:` (page 13)/`didChangeValueForKey:` (page 9) has otherwise been invoked.

A value of `NSKeyValueChangeInsertion`, `NSKeyValueChangeRemoval`, or `NSKeyValueChangeReplacement` indicates that mutating messages have been sent to the array returned by a `mutableArrayValueForKey:` message sent to the object, or that one of the key-value-coding-compliant array mutation methods for the key has been invoked, or that `willChange:valuesAtIndexes:forKey:` (page 12)/`didChange:valuesAtIndexes:forKey:` (page 8) has otherwise been invoked.

You can use `NSNumber`'s `intValue` method to retrieve the integer value of the change kind.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeNewKey`

If the value of the `NSKeyValueChangeKindKey` entry is `NSKeyValueChangeSetting`, and `NSKeyValueObservingOptionNew` was specified when the observer was registered, the value of this key is the new value for the attribute.

For `NSKeyValueChangeInsertion` or `NSKeyValueChangeReplacement`, if `NSKeyValueObservingOptionNew` was specified when the observer was registered, the value for this key is an `NSArray` instance that contains the objects that have been inserted or replaced other objects, respectively.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeOldKey`

If the value of the `NSKeyValueChangeKindKey` entry is `NSKeyValueChangeSetting`, and `NSKeyValueObservingOptionOld` was specified when the observer was registered, the value of this key is the value before the attribute was changed.

For `NSKeyValueChangeRemoval` or `NSKeyValueChangeReplacement`, if `NSKeyValueObservingOptionOld` was specified when the observer was registered, the value is an `NSArray` instance that contains the objects that have been removed or have been replaced by other objects, respectively.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueChangeIndexesKey`

If the value of the `NSKeyValueChangeKindKey` entry is `NSKeyValueChangeInsertion`, `NSKeyValueChangeRemoval`, or `NSKeyValueChangeReplacement`, the value of this key is an `NSIndexSet` object that contains the indexes of the inserted, removed, or replaced objects.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueObserving.h`.

Declared In

`NSKeyValueObserving.h`

NSKeyValueSetMutationKind

These constants are specified as the parameter to the methods `willChangeValueForKey:withSetMutation:usingObjects:` (page 13) and `didChangeValueForKey:withSetMutation:usingObjects:` (page 9).

```
enum {
    NSKeyValueUnionSetMutation = 1,
    NSKeyValueMinusSetMutation = 2,
    NSKeyValueIntersectSetMutation = 3,
    NSKeyValueSetSetMutation = 4
```

```
};
typedef NSUInteger NSKeyValueSetMutationKind;
```

Constants

`NSKeyValueUnionSetMutation`

Indicates that objects in the specified set are being added to the receiver. This mutation kind results in a `NSkeyValueChangeKindKey` value of `NSKeyValueChangeInsertion`.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueMinusSetMutation`

Indicates that the objects in the specified set are being removed from the receiver. This mutation kind results in a `NSkeyValueChangeKindKey` value of `NSKeyValueChangeRemoval`.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueIntersectSetMutation`

Indicates that the objects not in the specified set are being removed from the receiver. This mutation kind results in a `NSkeyValueChangeKindKey` value of `NSKeyValueChangeRemoval`.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueObserving.h`.

`NSKeyValueSetSetMutation`

Indicates that set of objects are replacing the existing objects in the receiver. This mutation kind results in a `NSkeyValueChangeKindKey` value of `NSKeyValueChangeReplacement`.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueObserving.h`.

Declared In

`NSKeyValueObserving.h`

Deprecated NSKeyValueObserving Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.5 and later

setKeys:triggerChangeNotificationsForDependentKey:

Configures the receiver to post change notifications for a given property if any of the properties specified in a given array changes. (Deprecated in Mac OS X v10.5 and later. You should use the method `keyPathsForValuesAffectingValueForKey:` (page 7) instead.)

```
+ (void)setKeys:(NSArray *)keys
    triggerChangeNotificationsForDependentKey:(NSString *)dependentKey
```

Parameters

keys

The names of the properties upon which the value of the property identified by *dependentKey* depends.

dependentKey

The name of a property whose value depends on the properties specified by *keys*.

Discussion

Invocations of will- and did-change KVO notification methods for any key in *keys* will automatically invoke the corresponding change notification methods for *dependentKey*. The receiver will not receive `willChange`/`didChange` messages to generate the notifications.

Dependencies should be registered before any instances of the receiving class are created, so you typically invoke this method in a class's `initialize` method, as illustrated in the following example.

```
+ (void)initialize
{
    [self setKeys:[NSArray arrayWithObjects:@"firstName", @"lastName", nil]
          triggerChangeNotificationsForDependentKey:@"fullName"];
}
```

Availability

Deprecated in Mac OS X v10.5 and later.

Related Sample Code

CoreRecipes

Dicey

iSpend

QTRecorder

Reducer

Declared In
`NSKeyValueObserving.h`

Document Revision History

This table describes the changes to *NSKeyValueObserving Protocol Reference*.

Date	Notes
2007-10-31	Marked setKeys:triggerChangeNotificationsForDependentKey: as deprecated in Mac OS X v10.5 and later.
2007-04-01	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index