NSKeyValueCoding Protocol Reference

Data Management: Event Handling

2010-01-20

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

Simultaneously published in the United States and Canada.

Contents

NSKeyValueCoding Protocol Reference

(informal protocol)

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

Overview

The `NSKeyValueCoding` informal protocol defines a mechanism by which you can access the properties of an object indirectly by name (or key), rather than directly through invocation of an accessor method or as instance variables. Thus, all of an object's properties can be accessed in a consistent manner.

The basic methods for accessing an object's values are `setValue:forKey:` (page 11), which sets the value for the property identified by the specified key, and `valueForKey:` (page 15), which returns the value for the property identified by the specified key. The default implementation uses the accessor methods normally implemented by objects (or to access instance variables directly if need be).

Tasks

Getting Values

- `valueForKey:` (page 15)
  Returns the value for the property identified by a given key.
- `valueForKeyPath:` (page 16)
  Returns the value for the derived property identified by a given key path.
- `dictionaryWithValuesForKeys:` (page 8)
  Returns a dictionary containing the property values identified by each of the keys in a given array.
- `valueForUndefinedKey:` (page 16)
  Invoked by `valueForKey:` (page 15) when it finds no property corresponding to a given key.
- `mutableArrayValueForKey:` (page 8)
  Returns a mutable array proxy that provides read-write access to an ordered to-many relationship specified by a given key.
- `mutableArrayValueForKeyPath:` (page 9)
  Returns a mutable array that provides read-write access to the ordered to-many relationship specified by a given key path.

- `mutableSetValueForKey:` (page 10)
  Returns a mutable set proxy that provides read-write access to the unordered to-many relationship specified by a given key.
- `mutableSetValueForKeyPath:` (page 10)
  Returns a mutable set that provides read-write access to the unordered to-many relationship specified by a given key path.

Setting Values

- `setValue:forKeyPath:` (page 12)
  Sets the value for the property identified by a given key path to a given value.
- `setValuesForKeysWithDictionary:` (page 13)
  Sets properties of the receiver with values from a given dictionary, using its keys to identify the properties.
- `setNilValueForKey:` (page 11)
  Invoked by `setValue:forKey:` (page 11) when it's given a `nil` value for a scalar value (such as an `int` or `float`).
- `setValue:forKey:` (page 11)
  Sets the property of the receiver specified by a given key to a given value.
- `setValue:forUndefinedKey:` (page 13)
  Invoked by `setValue:forKey:` (page 11) when it finds no property for a given key.

Changing Default Behavior

+ `accessInstanceVariablesDirectly` (page 7)
  Returns a Boolean value that indicates whether the key-value coding methods should access the corresponding instance variable directly on finding no accessor method for a property.

Validation

- `validateValue:forKey:error:` (page 14)
  Returns a Boolean value that indicates whether the value specified by a given pointer is valid for the property identified by a given key.
- `validateValue:forKeyPath:error:` (page 14)
  Returns a Boolean value that indicates whether the value specified by a given pointer is valid for a given key path relative to the receiver.

Deprecated Methods

- `handleQueryWithUnboundKey:` (page 21) Deprecated in Mac OS X v10.3
  Invoked by `valueForKey:` (page 15) when it finds no property corresponding to *`key`*. (Deprecated. Use `valueForUndefinedKey:` (page 16) instead.)

- `handleTakeValue:forUnboundKey:` (page 21) Deprecated in Mac OS X v10.3
  Invoked by `takeValue:forKey:` (page 21) when it finds no property binding for *key*. (Deprecated. Use `setValue:forUndefinedKey:` (page 13) instead.)
- `takeValue:forKey:` (page 21) Deprecated in Mac OS X v10.3
  Sets the value for the property identified by *key* to *value*. (Deprecated. Use `setValue:forKey:` (page 11) instead.)
- `takeValue:forKeyPath:` (page 22) Deprecated in Mac OS X v10.3
  Sets the value for the property identified by *keyPath* to *value*. (Deprecated. Use `setValue:forKeyPath:` (page 12) instead.)
- `takeValuesFromDictionary:` (page 22) Deprecated in Mac OS X v10.3
  Sets properties of the receiver with values from a given dictionary, using its keys to identify the properties (Deprecated. Use `setValuesForKeysWithDictionary:` (page 13) instead.)
- `unableToSetNilForKey:` (page 22) Deprecated in Mac OS X v10.3
  Invoked if *key* is represented by a scalar attribute. (Deprecated. Use `setNilValueForKey:` (page 11) instead.)
- `valuesForKeys:` (page 23) Deprecated in Mac OS X v10.3
  Returns a dictionary containing as keys the property names in *keys*, with corresponding values being the corresponding property values. (Deprecated. Use `dictionaryWithValuesForKeys:` (page 8) instead.)
+ `useStoredAccessor` (page 23) Deprecated in Mac OS X v10.4
  Returns `YES` if the stored value methods `storedValueForKey:` (page 23) and `takeStoredValue:forKey:` (page 24) should use private accessor methods in preference to public accessors. (Deprecated. This method has no direct replacement, although see accessInstanceVariablesDirectly (page 7).)
- `storedValueForKey:` (page 23) Deprecated in Mac OS X v10.4
  Returns the property identified by a given key. (Deprecated. If you are using the `NSManagedObject` class, use `primitiveValueForKey:` instead.)
- `takeStoredValue:forKey:` (page 24) Deprecated in Mac OS X v10.4
  Sets the value of the property identified by a given key. (Deprecated. If you are using the `NSManagedObject` class, use `setPrimitiveValue:forKey:` instead.)

Class Methods

accessInstanceVariablesDirectly

Returns a Boolean value that indicates whether the key-value coding methods should access the corresponding instance variable directly on finding no accessor method for a property.

```
+ (BOOL)accessInstanceVariablesDirectly
```

Return Value

`YES` if the key-value coding methods should access the corresponding instance variable directly on finding no accessor method for a property, otherwise `NO`.

Discussion

The default returns `YES`. Subclasses can override it to return `NO`, in which case the key-value coding methods won't access instance variables.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchFractal

Declared In
`NSKeyValueCoding.h`

Instance Methods

dictionaryWithValuesForKeys:

Returns a dictionary containing the property values identified by each of the keys in a given array.

```
- (NSDictionary *)dictionaryWithValuesForKeys:(NSArray *)keys
```

Parameters

keys

An array containing `NSString` objects that identify properties of the receiver.

Return Value
A dictionary containing as keys the property names in *keys*, with corresponding values being the corresponding property values.

Discussion
The default implementation invokes `valueForKey:` (page 15) for each key in *keys* and substitutes `NSNull` values in the dictionary for returned `nil` values.

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

See Also
`- setValuesForKeysWithDictionary:` (page 13)

Related Sample Code
Departments and Employees
QTMetadataEditor

Declared In
`NSKeyValueCoding.h`

mutableArrayValueForKey:

Returns a mutable array proxy that provides read-write access to an ordered to-many relationship specified by a given key.

```
- (NSMutableArray *)mutableArrayValueForKey:(NSString *)key
```

Parameters

key

The name of an ordered to-many relationship.

Return Value

A mutable array proxy that provides read-write access to the ordered to-many relationship specified by *key*.

Discussion

Objects added to the mutable array become related to the receiver, and objects removed from the mutable array become unrelated. The default implementation recognizes the same simple accessor methods and array accessor methods as `valueForKey:` (page 15), and follows the same direct instance variable access policies, but always returns a mutable collection proxy object instead of the immutable collection that `valueForKey:` would return.

The search pattern that `mutableArrayValueForKey:` uses is described in Accessor Search Implementation Details in *Key-Value Coding Programming Guide*.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- mutableArrayValueForKeyPath:` (page 9)

Related Sample Code

SimpleCalendar

SimpleStickies

Declared In

`NSKeyValueCoding.h`

mutableArrayValueForKeyPath:

Returns a mutable array that provides read-write access to the ordered to-many relationship specified by a given key path.

```
- (NSMutableArray *)mutableArrayValueForKeyPath:(NSString *)keyPath
```

Parameters

keyPath

A key path, relative to the receiver, to an ordered to-many relationship.

Return Value

A mutable array that provides read-write access to the ordered to-many relationship specified by *keyPath*.

Discussion

See `mutableArrayValueForKey:` (page 8) for additional details.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- mutableArrayValueForKey:` (page 8)

Declared In

`NSKeyValueCoding.h`

mutableSetValueForKey:

Returns a mutable set proxy that provides read-write access to the unordered to-many relationship specified by a given key.

```
- (NSMutableSet *)mutableSetValueForKey:(NSString *)key
```

Parameters

key

The name of an unordered to-many relationship.

Return Value

A mutable set that provides read-write access to the unordered to-many relationship specified by *key*.

Discussion

Objects added to the mutable set proxy become related to the receiver, and objects removed from the mutable set become unrelated. The default implementation recognizes the same simple accessor methods and set accessor methods as `valueForKey:` (page 15), and follows the same direct instance variable access policies, but always returns a mutable collection proxy object instead of the immutable collection that `valueForKey:` would return.

The search pattern that `mutableSetValueForKey:` uses is described in Accessor Search Implementation Details in *Key-Value Coding Programming Guide*.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- mutableArrayValueForKeyPath:` (page 9)

Related Sample Code

CoreRecipes

QTMetadataEditor

Declared In

`NSKeyValueCoding.h`

mutableSetValueForKeyPath:

Returns a mutable set that provides read-write access to the unordered to-many relationship specified by a given key path.

```
- (NSMutableSet *)mutableSetValueForKeyPath:(NSString *)keyPath
```

Parameters

keyPath

A key path, relative to the receiver, to an unordered to-many relationship.

Return Value

A mutable set that provides read-write access to the unordered to-many relationship specified by *keyPath*.

Discussion

See `mutableSetValueForKey:` (page 10) for additional details.

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

See Also
`- mutableArrayValueForKey:` (page 8)

Declared In
`NSKeyValueCoding.h`

setNilValueForKey:

Invoked by `setValue:forKey:` (page 11) when it's given a `nil` value for a scalar value (such as an `int` or `float`).

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

Parameters

key

The name of one of the receiver's properties.

Discussion
Subclasses can override this method to handle the request in some other way, such as by substituting `0` or a sentinel value for `nil` and invoking `setValue:forKey:` again or setting the variable directly. The default implementation raises an `NSInvalidArgumentException`.

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

Declared In
`NSKeyValueCoding.h`

setValue:forKey:

Sets the property of the receiver specified by a given key to a given value.

```
- (void)setValue:(id)value forKey:(NSString *)key
```

Parameters

value

The value for the property identified by *key*.

key

The name of one of the receiver's properties.

Discussion
If *key* identifies a to-one relationship, relate the object specified by *value* to the receiver, unrelating the previously related object if there was one. Given a collection object and a *key* that identifies a to-many relationship, relate the objects contained in the collection to the receiver, unrelating previously related objects if there were any.

The search pattern that `setValue:forKey:` uses is described in Accessor Search Implementation Details in *Key-Value Coding Programming Guide*.

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

Related Sample Code
CIAnnotation
CocoaSlides
FunHouse
GridCalendar
ImageApp

Declared In
`NSKeyValueCoding.h`

setValue:forKeyPath:

Sets the value for the property identified by a given key path to a given value.

```
- (void)setValue:(id)value forKeyPath:(NSString *)keyPath
```

Parameters

value

The value for the property identified by *keyPath*.

keyPath

A key path of the form *relationship.property* (with one or more relationships): for example "department.name" or "department.manager.lastName."

Discussion
The default implementation of this method gets the destination object for each relationship using `valueForKey:` (page 15), and sends the final object a `setValue:forKey:` message.

Special Considerations

When using this method, and the destination object does not implement an accessor for the value, the default behavior is for that object to retain *value* rather than copy or assign *value*.

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

See Also
`- valueForKeyPath:` (page 16)

Related Sample Code
BindingsJoystick
CoreRecipes
Fire
Fireworks
GeekGameBoard

Declared In
`NSKeyValueCoding.h`

setValue:forUndefinedKey:

Invoked by `setValue:forKey:` (page 11) when it finds no property for a given key.

```
- (void)setValue:(id)value forUndefinedKey:(NSString *)key
```

Parameters

value

The value for the key identified by *key*.

key

A string that is not equal to the name of any of the receiver's properties.

Discussion

Subclasses can override this method to handle the request in some other way. The default implementation raises an `NSUndefinedKeyException`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- valueForUndefinedKey:` (page 16)

Declared In

`NSKeyValueCoding.h`

setValuesForKeysWithDictionary:

Sets properties of the receiver with values from a given dictionary, using its keys to identify the properties.

```
- (void)setValuesForKeysWithDictionary:(NSDictionary *)keyedValues
```

Parameters

keyedValues

A dictionary whose keys identify properties in the receiver. The values of the properties in the receiver are set to the corresponding values in the dictionary.

Discussion

The default implementation invokes `setValue:forKey:` (page 11) for each key-value pair, substituting `nil` for `NSNull` values in *keyedValues*.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- dictionaryWithValuesForKeys:` (page 8)

Related Sample Code

Core Data HTML Store

Departments and Employees

QTMetadataEditor

QuickLookSketch

Declared In

`NSKeyValueCoding.h`

validateValue:forKey:error:

Returns a Boolean value that indicates whether the value specified by a given pointer is valid for the property identified by a given key.

```
- (BOOL)validateValue:(id *)ioValue forKey:(NSString *)key error:(NSError **)outError
```

Parameters

ioValue

A pointer to a new value for the property identified by *key*. This method may modify or replace the value in order to make it valid.

key

The name of one of the receiver's properties. The key must specify an attribute or a to-one relationship.

outError

If validation is necessary and *ioValue* is not transformed into a valid value, upon return contains an `NSError` object that describes the reason that *ioValue* is not a valid value.

Return Value

YES if **ioValue* is a valid value for the property identified by *key*, or of the method is able to modify the value to **ioValue* to make it valid; otherwise `NO`.

Discussion

The default implementation of this method searches the class of the receiver for a validation method whose name matches the pattern `validate<Key>:error:`. If such a method is found it is invoked and the result is returned. If no such method is found, `YES` is returned.

The sender of the message is never given responsibility for releasing *ioValue* or *outError*.

See "Key-Value Validation" for more information.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- validateValue:forKeyPath:error:` (page 14)

Declared In

`NSKeyValueCoding.h`

validateValue:forKeyPath:error:

Returns a Boolean value that indicates whether the value specified by a given pointer is valid for a given key path relative to the receiver.

```
- (BOOL)validateValue:(id *)ioValue forKeyPath:(NSString *)inKeyPath error:(NSError
    **)outError
```

Parameters

ioValue

A pointer to a new value for the property identified by *keyPath*. This method may modify or replace the value in order to make it valid.

key

The name of one of the receiver's properties. The key path must specify an attribute or a to-one relationship. The key path has the form *relationship.property* (with one or more relationships); for example “department.name” or “department.manager.lastName”.

outError

If validation is necessary and *ioValue* is not transformed into a valid value, upon return contains an `NSError` object that describes the reason that *ioValue* is not a valid value.

Discussion
The default implementation gets the destination object for each relationship using `valueForKey:` (page 15) and returns the result of a `validateValue:forKey:error:` message to the final object.

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

See Also
`- validateValue:forKey:error:` (page 14)

Declared In
`NSKeyValueCoding.h`

valueForKey:

Returns the value for the property identified by a given key.

```
- (id)valueForKey:(NSString *)key
```

Parameters

key

The name of one of the receiver's properties.

Return Value
The value for the property identified by *key*.

Discussion
The search pattern that `valueForKey:` uses to find the correct value to return is described in Accessor Search Implementation Details in *Key-Value Coding Programming Guide*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- valueForKeyPath:` (page 16)

Related Sample Code
CIAnnotation
CIVideoDemoGL
CustomAtomicStoreSubclass
FunHouse
ImageApp

Declared In
`NSKeyValueCoding.h`

valueForKeyPath:

Returns the value for the derived property identified by a given key path.

```
- (id)valueForKeyPath:(NSString *)keyPath
```

Parameters

keyPath

A key path of the form *relationship.property* (with one or more relationships); for example "department.name" or "department.manager.lastName".

Return Value

The value for the derived property identified by *keyPath*.

Discussion

The default implementation gets the destination object for each relationship using `valueForKey:` (page 15) and returns the result of a `valueForKey:` message to the final object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setValue:forKeyPath:` (page 12)

Related Sample Code

Aperture Edit Plugin - Borders & Titles

Core Data HTML Store

CoreRecipes

Dicey

Spotlight

Declared In

`NSKeyValueCoding.h`

valueForUndefinedKey:

Invoked by `valueForKey:` (page 15) when it finds no property corresponding to a given key.

```
- (id)valueForUndefinedKey:(NSString *)key
```

Parameters

key

A string that is not equal to the name of any of the receiver's properties.

Discussion

Subclasses can override this method to return an alternate value for undefined keys. The default implementation raises an `NSUndefinedKeyException`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setValue:forUndefinedKey:` (page 13)

Declared In
`NSKeyValueCoding.h`

Constants

Key Value Coding Exception Names

This constant defines the name of an exception raised when a key value coding operation fails.

```
extern NSString *NSUndefinedKeyException;
```

Constants
`NSUndefinedKeyException`
Raised when a key value coding operation fails. *`userInfo`* keys are described in "NSUndefinedKeyException userInfo Keys" (page 17)

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueCoding.h`.

Declared In
`NSKeyValueCoding.h`

NSUndefinedKeyException userInfo Keys

These constants are keys into an `NSUndefinedKeyException` *`userInfo`* dictionary

```
extern NSString *NSTargetObjectUserInfoKey;
extern NSString *NSUnknownUserInfoKey;
```

Constants
`NSTargetObjectUserInfoKey`
The object on which the key value coding operation failed.

`NSUnknownUserInfoKey`
The key for which the key value coding operation failed.

Discussion
For additional information see "Key Value Coding Exception Names" (page 17).

Declared In
`NSKeyValueCoding.h`

Array operators

These constants define the available array operators. See Set and Array Operators for more information.

```
NSString *const NSAverageKeyValueOperator;
NSString *const NSCountKeyValueOperator;
NSString *const NSDistinctUnionOfArraysKeyValueOperator;
NSString *const NSDistinctUnionOfObjectsKeyValueOperator;
NSString *const NSDistinctUnionOfSetsKeyValueOperator;
NSString *const NSMaximumKeyValueOperator;
NSString *const NSMinimumKeyValueOperator;
NSString *const NSSumKeyValueOperator;
NSString *const NSUnionOfArraysKeyValueOperator;
NSString *const NSUnionOfObjectsKeyValueOperator;
NSString *const NSUnionOfSetsKeyValueOperator;
```

Constants

`NSAverageKeyValueOperator`

The `@avg` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSCountKeyValueOperator`

The `@count` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSDistinctUnionOfArraysKeyValueOperator`

The `@distinctUnionOfArrays` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSDistinctUnionOfObjectsKeyValueOperator`

The `@distinctUnionOfObjects` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSDistinctUnionOfSetsKeyValueOperator`

The `@distinctUnionOfSets` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSMaximumKeyValueOperator`

The `@max` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSMinimumKeyValueOperator`

The `@min` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSSumKeyValueOperator`

The `@sum` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSUnionOfArraysKeyValueOperator`
The `@unionOfArrays` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSUnionOfObjectsKeyValueOperator`
The `@unionOfObjects` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

`NSUnionOfSetsKeyValueOperator`
The `@unionOfSets` array operator.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueCoding.h`.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`NSKeyValueCoding.h`

Deprecated NSKeyValueCoding Methods

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

Deprecated in Mac OS X v10.3

handleQueryWithUnboundKey:

Invoked by `valueForKey:` (page 15) when it finds no property corresponding to *key*. (Deprecated in Mac OS X v10.3. Use `valueForUndefinedKey:` (page 16) instead.)

```
- (id)handleQueryWithUnboundKey:(NSString *)key
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`NSKeyValueCoding.h`

handleTakeValue:forUnboundKey:

Invoked by `takeValue:forKey:` (page 21) when it finds no property binding for *key*. (Deprecated in Mac OS X v10.3. Use `setValue:forUndefinedKey:` (page 13) instead.)

```
- (void)handleTakeValue:(id)value forUnboundKey:(NSString *)key
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`NSKeyValueCoding.h`

takeValue:forKey:

Sets the value for the property identified by *key* to *value*. (Deprecated in Mac OS X v10.3. Use `setValue:forKey:` (page 11) instead.)

```
- (void)takeValue:(id)value forKey:(NSString *)key
```

Availability
Deprecated in Mac OS X v10.3.

Related Sample Code
SimpleStickies

Declared In
`NSKeyValueCoding.h`

takeValue:forKeyPath:

Sets the value for the property identified by *keyPath* to *value*. (Deprecated in Mac OS X v10.3. Use `setValue:forKeyPath:` (page 12) instead.)

```
- (void)takeValue:(id)value forKeyPath:(NSString *)keyPath
```

Availability
Deprecated in Mac OS X v10.3.

Declared In
`NSKeyValueCoding.h`

takeValuesFromDictionary:

Sets properties of the receiver with values from a given dictionary, using its keys to identify the properties (Deprecated in Mac OS X v10.3. Use `setValuesForKeysWithDictionary:` (page 13) instead.)

```
- (void)takeValuesFromDictionary:(NSDictionary *)aDictionary
```

Availability
Deprecated in Mac OS X v10.3.

Declared In
`NSKeyValueCoding.h`

unableToSetNilForKey:

Invoked if *key* is represented by a scalar attribute. (Deprecated in Mac OS X v10.3. Use `setNilValueForKey:` (page 11) instead.)

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

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`NSKeyValueCoding.h`

valuesForKeys:

Returns a dictionary containing as keys the property names in *keys*, with corresponding values being the corresponding property values. (Deprecated in Mac OS X v10.3. Use `dictionaryWithValuesForKeys:` (page 8) instead.)

```
- (NSDictionary *)valuesForKeys:(NSArray *)keys
```

Availability
Deprecated in Mac OS X v10.3.

Related Sample Code
Core Data HTML Store
Departments and Employees

Declared In
`NSKeyValueCoding.h`

Deprecated in Mac OS X v10.4

useStoredAccessor

Returns `YES` if the stored value methods `storedValueForKey:` (page 23) and `takeStoredValue:forKey:` (page 24) should use private accessor methods in preference to public accessors. (Deprecated in Mac OS X v10.4. This method has no direct replacement, although see accessInstanceVariablesDirectly (page 7).)

```
+ (BOOL)useStoredAccessor
```

Discussion
Returning `NO` causes the stored value methods to use the same accessor method or instance variable search order as the corresponding basic key-value coding methods (`valueForKey:` (page 15) and `takeValue:forKey:` (page 21)). The default implementation returns `YES`.

Applications should use the `valueForKey:` and `setValue:forKey:` methods instead of `storedValueForKey:` and `takeStoredValue:forKey:`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSKeyValueCoding.h`

storedValueForKey:

Returns the property identified by a given key. (Deprecated in Mac OS X v10.4. If you are using the `NSManagedObject` class, use `primitiveValueForKey:` instead.)

```
- (id)storedValueForKey:(NSString *)key
```

Discussion
This method is used when the value is retrieved for storage in an object store (generally, this storage is ultimately in a database) or for inclusion in a snapshot. The default implementation is similar to the implementation of `valueForKey:` (page 15), but it resolves *key* with a different method/instance variable search order:

1. Searches for a private accessor method based on *key* (a method preceded by an underbar). For example, with a *key* of "lastName", `storedValueForKey:` looks for a method named `_getLastName` or `_lastName`.

2. If a private accessor is not found, searches for an instance variable based on *key* and returns its value directly. For example, with a `key` of "lastName", `storedValueForKey:` looks for an instance variable named `_lastName` or `lastName`.

3. If neither a private accessor nor an instance variable is found, `storedValueForKey:` searches for a public accessor method based on *key*. For the *key* "lastName", this would be `getLastName` or `lastName`.

4. If *key* is unknown, `storedValueForKey:` calls `handleTakeValue:forUnboundKey:` (page 21).

This different search order allows an object to bypass processing that is performed before returning a value through a public API. However, if you always want to use the search order in `valueForKey:` (page 15), you can implement the class method `useStoredAccessor` (page 23) to return `NO`. And as with `valueForKey:` (page 15), you can prevent direct access of an instance variable with the class method `accessInstanceVariablesDirectly` (page 7).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSKeyValueCoding.h`

takeStoredValue:forKey:

Sets the value of the property identified by a given key. (Deprecated in Mac OS X v10.4. If you are using the `NSManagedObject` class, use `setPrimitiveValue:forKey:` instead.)

```
- (void)takeStoredValue:(id)value forKey:(NSString *)key
```

Discussion
This method is used to initialize the receiver with values from an object store (generally, this storage is ultimately from a database) or to restore a value from a snapshot. The default implementation is similar to the implementation of `takeValue:forKey:` (page 21), but it resolves *key* with a different method/instance variable search order:

1. Searches for a private accessor method based on *key* (a method preceded by an underbar). For example, with a *key* of "lastName", `takeStoredValue:forKey:` looks for a method named `_setLastName:`.

2. If a private accessor is not found, searches for an instance variable based on *key* and sets its *value* directly. For example, with a *key* of "lastName", `takeStoredValue:forKey:` looks for an instance variable named `_lastName` or `lastName`.

3. If neither a private accessor nor an instance variable is found, `takeStoredValue:forKey:` searches for a public accessor method based on *key*. For the *key* "lastName", this would be `setLastName:`.

4. If *key* is unknown, `takeStoredValue:forKey:` calls `handleTakeValue:forUnboundKey:` (page 21).

This different search order allows an object to bypass processing that is performed before setting a value through a public API. However, if you always want to use the search order in `takeValue:forKey:` (page 21), you can implement the class method `useStoredAccessor` (page 23) to return `NO`. And as with `valueForKey:` (page 15), you can prevent direct access of an instance variable with the class method `accessInstanceVariablesDirectly` (page 7).

Availability
Deprecated in Mac OS X v10.4.

Related Sample Code
SimpleStickies

Declared In
`NSKeyValueCoding.h`

Document Revision History

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

Date	Notes
2010-01-20	Clarified setValue:forKey: and memory management.
2009-11-13	Updated optional status of protocol methods.
2009-02-04	Accessor search descriptions of setValue:forKey:, valueForKey:, mutableArrayValueForKey:, and mutableSetValueForKey: now point to the Key-Value Coding Programming Guide.
2007-02-23	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History