Class

NSObjectController

A controller that can manage an object's properties referenced by key-value paths.

macOS

```
class NSObjectController
```

Overview

`NSObjectController` is a Cocoa bindings–compatible controller class. Properties of the content object of instances 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 `objectClass`, which subclasses must override. Your application should use a custom data class that is key-value compliant whenever possible.

Object Controllers, Entity Mode, and Lazy Fetching

`NSObjectController` and its subclasses, when in entity mode, can now fetch lazily. With lazy fetching enabled using the property `usesLazyFetching`, the controller will try to fetch only a small amount of data from available persistent stores. This can provide a significant improvement in memory use when a large amount of content is stored on disk but just a subset of that data is required in memory.

When set to use lazy fetching, a controller will fetch objects in batches. You can change the default batch size for your application by setting a value for the the user default "`com.apple`

`.CocoaBindings.LazyFetchBatchSize`". If you have table views bound to an array controller set to use lazy fetching, the size of the controller's batch size will grow as the table views' visible row count grows.

Add, Insert, and Remove operations on controllers that use lazy fetching behave similarly to the same operations on a regular controller. The difference is that it is faster to sort an array controller using lazy fetching if:

- All of the keys in the `sortDescriptors` array are modeled, non transient properties.
- All of the selectors in the `sortDescriptors` array are `compare:` or `caseInsensitive Compare:`.
- There are no changes in the controller's managed object context

---

Topics

Initializing an object controller

`init(content: Any?)`

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

Managing content

`var content: Any?`

The receiver's content object.

`var automaticallyPreparesContent: Bool`

A Boolean that shows whether the receiver automatically creates and inserts new content objects automatically when loading from a nib file.

`func prepareContent()`

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

Setting the content class

`var objectClass: AnyClass!`

The object class to use when creating new objects.

Managing objects

`func newObject() -> Any`

Creates and returns a new object of the appropriate class.

`func addObject(Any)`

Sets the receiver's content object.

`func removeObject(Any)`

Removes a given object from the receiver's content.

`func add(Any?)`

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

`var canAdd: Bool`

A Boolean value that indicates whether an object can be added to the receiver using `add( _:)`.

`func remove(Any?)`

Removes the receiver's content object.

`var canRemove: Bool`

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

Managing editing

`var isEditable: Bool`

A Boolean that indicates whether the receiver allows adding and removing objects.

Core Data support

`var entityName: String?`

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

`func fetch(Any?)`

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

`var usesLazyFetching: Bool`

A Boolean that indicates whether the receiver uses lazy fetching.

`func defaultFetchRequest() -> NSFetchRequest<any NSFetchRequestResult>`

Returns the default fetch request used by the receiver.

`var fetchPredicate: NSPredicate?`

The receiver's fetch predicate.

`var managedObjectContext: NSManagedObjectContext?`

The receiver's managed object context.

`func fetch(with: NSFetchRequest<any NSFetchRequestResult>?, merge: Bool) throws`

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

Obtaining selections

`var selectedObjects: [Any]`

An array of all objects to be affected by editing.

`var selection: Any`

A proxy object representing the receiver's selection.

Validating user interface items

`func validateUserInterfaceItem(any NSValidatedUserInterfaceItem) -> Bool`

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

Initializers

`init?(coder: NSCoder)`

Relationships

Inherits From

`NSController`

Inherited By

NSArrayController, NSTreeController

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSCoding
NSEditor
NSEditorRegistration
NSObjectProtocol
Sendable
SendableMetatype

See Also

Core Controllers

class NSController

An abstract class that implements the NSEditor and NSEditorRegistration informal protocols required for controller classes.