API Collection

NSKeyValueBindingCreation

A set of methods that you can use to create and remove bindings between view objects and controllers, or between controllers and model objects.

Overview

The NSKeyValueBindingCreation informal protocol also provides a means for a view subclass to advertise the bindings that it exposes. The protocol is implemented by `NSObject` and its methods can be overridden by view and controller subclasses.

When a new binding is created it relates the receiver's binding (for example, a property of the view object) to a property of the observable object specified by a key path. When the value of the specified property of the observable object changes, the receiver is notified using the key-value observing mechanism. A binding also specifies binding options that can further customize how the observing and the observed objects interact.

Bindings are considered to be a property of the object which is bound, and all information related to bindings should be owned by the object. All standard bindings on AppKit objects (views, cells, table columns, controllers) unbind their bindings automatically when they are deallocated, but if you create key-value bindings for other kind of objects, you need to make sure that you remove those bindings before deallocation (observed objects have weak references to their observers, so controllers/model objects might continue referencing and messaging the objects that were bound to them).

Bindings between objects are typically established in Interface Builder using the Bindings inspector. However, there are times it must be done programmatically, such as when establishing a binding between objects in different nib files.

`NSView` subclasses can expose additional key-value-coding/key-value-observing compliant properties as bindings by calling the class method `exposeBinding(_:)` for each of the properties. This is typically done in the class's `initialize` method. By exposing the bindings

that an object supports and creating an Interface Builder palette, you can make instances of your own classes bindable in Interface Builder.

Topics

Exposing bindings

`class func exposeBinding(NSBindingName)`

Exposes the specified `binding`, advertising its availability.

`var exposedBindings: [NSBindingName]`

Returns an array containing the bindings exposed by the receiver.

Managing bindings

`func valueClassForBinding(NSBindingName) -> AnyClass?`

Returns the class of the value that will be returned for the specified binding.

`func bind(NSBindingName, to: Any, withKeyPath: String, options: [NSBindingOption : Any]?)`

Establishes a binding between a given property of the receiver and the property of a given object specified by a given key path.

`func optionDescriptionsForBinding(NSBindingName) -> [NSAttributeDescription]`

Returns an array describing the options for the specified binding.

`func infoForBinding(NSBindingName) -> [NSBindingInfoKey : Any]?`

Returns a dictionary describing the receiver's `binding`.

`struct NSBindingInfoKey`

`func unbind(NSBindingName)`

Removes a given binding between the receiver and a controller.

`func NSIsControllerMarker(_ object: Any?) -> Bool`

Tests whether a given object is special marker object used for indicating the state of a selection in relation to a key.

Constants

`struct` NSBindingName

Values that specify a binding for certain methods.

`struct` NSBindingOption

Binding Dictionary Keys

The following values are used as keys in the dictionary returned by `infoForBinding(_:)`

See Also

Related Documentation

Cocoa Bindings Reference

Cocoa Bindings Programming Topics

Key-Value Coding

NSKeyValueCoding

A mechanism by which you can access the properties of an object indirectly by name or key.

NSScriptKeyValueCoding

A collection of methods that provide additional capabilities for working with key-value coding.

NSScriptKeyValueCoding Exception Names

Exceptions raised by key-value coding methods.