Accessing Collection Properties

Key-value coding compliant objects expose their to-many properties in the same way that they expose other properties. You can get or set a collection object just as you would any other object using `valueForKey:` and `setValue:forKey:` (or their key path equivalents). However, when you want to manipulate the content of these collections, it's usually most efficient to use the mutable proxy methods defined by the protocol.

The protocol defines three different proxy methods for collection object access, each with a key and a key path variant:

- `mutableArrayValueForKey:` and `mutableArrayValueForKeyPath:`

  These return a proxy object that behaves like an `NSMutableArray` object.

- `mutableSetValueForKey:` and `mutableSetValueForKeyPath:`

  These return a proxy object that behaves like an `NSMutableSet` object.

- `mutableOrderedSetValueForKey:` and `mutableOrderedSetValueForKeyPath:`

  These return a proxy object that behaves like an `NSMutableOrderedSet` object.

When you operate on the proxy object, adding objects to, removing objects from, or replacing objects in it, the default implementation of the protocol modifies the underlying property accordingly. This is more efficient than obtaining a non-mutable collection object with `valueForKey:`, creating a modified one with altered content, and then storing it back to the object with a `setValue:forKey:` message. In many cases, it is also more efficient than working directly with a mutable property. These methods provide the additional benefit of maintaining key-value observing compliance for the objects held in the collection object (see *Key-Value Observing Programming Guide* for details.