Validating Properties

The key-value coding protocol defines methods to support property validation. Just as you use key-based accessors to read and write properties of a key-value coding compliant object, you can also validate properties by key (or key path). When you call the `validateValue:forKey:error:` (or the `validateValue:forKeyPath:error:`) method, the default implementation of the protocol searches the object receiving the validation message (or the one at the end of the key path) for a method whose name matches the pattern `validate<Key>:error:`. If the object has no such method, the validation succeeds by default, and the default implementation returns `YES`. When a property-specific validation method exists, the default implementation returns the result of calling that method instead.

> NOTE
>
> You typically use the validation described here only in Objective-C. In Swift, property validation is more idiomatically handled by relying on compiler support for optionals and strong type checking, while using the built-in `willSet` and `didSet` property observers to test any run-time API contracts, as described in the Property Observers section of *The Swift Programming Language (Swift 3.0.1)*.

Because proper
has three possit

1. The validation method deems the value object valid and returns `YES` without altering the value or the error.
2. The validation method deems the value object invalid, but chooses not to alter it. In this case, the method returns `NO` and sets the error reference (if provided by the caller) to an `NSError` object that indicates the reason for failure.
3. The validation method deems the value object invalid, but creates a new, valid one as a replacement. In this case, the method returns `YES` while leaving the error object untouched. Before returning, the method modifies the value reference to point at the new value object. When it makes a modification, the method always creates a new object, rather than modifying the old one, even if the value object is mutable.

Listing 6-1 shows an example of how to call validation for a name string.

Listing 6-1 Validation of the name property

```
Person* person = [[Person alloc] init];
NSError* error;
NSString* name = @"John";
if (![person validateValue:&name forKey:@"name" error:&error]) {
    NSLog(@"%@",error);
}
```

Automatic Validation

In general, neither the key-value coding protocol nor its default implementation define any mechanism to perform validation automatically. Instead, you make use of the validation methods when appropriate for your app.

Certain other Cocoa technologies do perform validation automatically in some circumstances. For example, Core Data automatically performs validation when the managed object context is saved (see *Core Data Programming Guide*). Also, in macOS, Cocoa bindings allow you to specify that validation should occur automatically (see *Cocoa Bindings Programming Topics* for more information).