DEMYSTIFYING THE OBJECTIVE-C RUNTIME

Sébastien Morel - Senior developer at **Wherecloud**

april 10, 2012

WHAT IS RUNTIME ?

Library on top of C language.

Foundation of the Objective-C dynamic language.

TRUST YOURSELF !

"This reference is useful primarily for developing bridge layers between Objective-C and other languages, or for low-level debugging."

TRUE

"You typically do not need to use the Objective-C runtime library directly when programming in Objective-C."

FALSE

People on the web uses words like "Faking" or "Hacking".

FALSE

Using **Runtime** is **LEGAL** !

WHAT DOES IT ALLOW?

Introspection / **Creation** / **Modification**

Classes
Instance Variables
Properties
Associated Objects

Messages / Selectors / Implementations
Protocols

WHAT DO WE **HATE** IMPLEMENTING AS **DEVELOPERS** ?

From Apple Sample : LocateMe

```
- (void)dealloc {

    [setupViewController release];

    [startButton release];

    [descriptionLabel release];

    [locationManager release];

    [locationMeasurements release];

    [bestEffortAtLocation release];

    [stateString release];

    [dateFormatter release];

    [tableView release];

    [locationDetailViewController release];

    [setupViewController release];

    [super dealloc];

}
```

USELESS

From Apple Sample : LocateMe

```
- (void)viewDidUnload {
    self.startButton = nil;
    self.descriptionLabel = nil;
    self.stateString = nil;
    self.tableView = nil;
    // For the readonly properties, they must be released and set to nil directly.
    [setupViewController release];
    setupViewController = nil;
    [locationDetailViewController release];
    locationDetailViewController = nil;
    [dateFormatter release];
    dateFormatter = nil;
}
```

USELESS

From ASIHTTPRequest

```
- (id)copyWithZone:(NSZone *)zone
{
    // Don't forget - this will return a retained copy!
    ASIHTTPRequest *newRequest = [[[self class] alloc] initWithURL:[self url]];
    [newRequest setDelegate:[self delegate]];
    [newRequest setRequestMethod:[self requestMethod]];
    [newRequest setPostBody:[self postBody]];
    [newRequest setShouldStreamPostDataFromDisk:[self shouldStreamPostDataFromDisk]];
    [newRequest setPostBodyFilePath:[self postBodyFilePath]];
    [newRequest setRequestHeaders:[[[self requestHeaders] mutableCopyWithZone:zone] autorelease]];
    [newRequest setRequestCookies:[[[self requestCookies] mutableCopyWithZone:zone] autorelease]];
    [newRequest setUseCookiePersistence:[self useCookiePersistence]];
    [newRequest setUseKeychainPersistence:[self useKeychainPersistence]];
    [newRequest setUseSessionPersistence:[self useSessionPersistence]];
    [newRequest setAllowCompressedResponse:[self allowCompressedResponse]];
    [newRequest setDownloadDestinationPath:[self downloadDestinationPath]];
    [newRequest setTemporaryFileDownloadPath:[self temporaryFileDownloadPath]];
    [newRequest setUsername:[self username]];
    [newRequest setPassword:[self password]];
    [newRequest setDomain:[self domain]];
    [newRequest setProxyUsername:[self proxyUsername]];
    [newRequest setProxyPassword:[self proxyPassword]];
    [newRequest setProxyDomain:[self proxyDomain]];
    [newRequest setProxyHost:[self proxyHost]];
    [newRequest setProxyPort:[self proxyPort]];
    [newRequest setProxyType:[self proxyType]];
    [newRequest setUploadProgressDelegate:[self uploadProgressDelegate]];
    [newRequest setDownloadProgressDelegate:[self downloadProgressDelegate]];
    [newRequest setShouldPresentAuthenticationDialog:[self shouldPresentAuthenticationDialog]];
    [newRequest setShouldPresentProxyAuthenticationDialog:[self shouldPresentProxyAuthenticationDialog]];
    [newRequest setPostLength:[self postLength]];
    [newRequest setHaveBuiltPostBody:[self haveBuiltPostBody]];
    [newRequest setDidStartSelector:[self didStartSelector]];
    [newRequest setDidFinishSelector:[self didFinishSelector]];
    [newRequest setDidFailSelector:[self didFailSelector]];
    [newRequest setTimeOutSeconds:[self timeOutSeconds]];
    [newRequest setShouldResetDownloadProgress:[self shouldResetDownloadProgress]];
    [newRequest setShouldResetUploadProgress:[self shouldResetUploadProgress]];
    [newRequest setShowAccurateProgress:[self showAccurateProgress]];
    [newRequest setDefaultResponseEncoding:[self defaultResponseEncoding]];
    [newRequest setAllowResumeForFileDownloads:[self allowResumeForFileDownloads]];
    [newRequest setUserInfo:[[[self userInfo] copyWithZone:zone] autorelease]];
    [newRequest setUseHTTPVersionOne:[self useHTTPVersionOne]];
    [newRequest setShouldRedirect:[self shouldRedirect]];
    [newRequest setValidatesSecureCertificate:[self validatesSecureCertificate]];
   [newRequest setClientCertificateIdentity:clientCertificateIdentity];
    [newRequest setClientCertificates:[[clientCertificates copy] autorelease]];
    [newRequest setPACurl:[self PACurl]];
    [newRequest setShouldPresentCredentialsBeforeChallenge:[self shouldPresentCredentialsBeforeChallenge]];
    [newRequest setNumberOfTimesToRetryOnTimeout:[self numberOfTimesToRetryOnTimeout]];
    [newRequest setShouldUseRFC2616RedirectBehaviour:[self shouldUseRFC2616RedirectBehaviour]];
    [newRequest setShouldAttemptPersistentConnection:[self shouldAttemptPersistentConnection]];
    [newRequest setPersistentConnectionTimeoutSeconds:[self persistentConnectionTimeoutSeconds]];
    return newRequest;
}
```

BIG BURP !

From Apple Documentation

```
- (void)encodeWithCoder:(NSCoder *)coder {

    [super encodeWithCoder:coder];

    [coder encodeObject:mapName forKey:@"MVMapName"];

    [coder encodeFloat:magnification forKey:@"MVMagnification"];

    [coder encodeObject:legendView forKey:@"MVLegend"];

    [coder encodeConditionalObject:auxiliaryView forKey:@"MVAuxView"];

}
```

USELESS

```
- (NSString*) description

- (id)initWithCoder:(NSCoder *)aDecoder

- (BOOL)isEqual:(id)object
```

USELESS

AND A LOT MORE !

EVERYTHING CAN BE AUTOMATED !

A simple class should looks like:

```
@interface LUDownload : CKObject
@property(nonatomic,retain) LUFilm* film;
@property(nonatomic,retain) NSURL* downloadURL;
@property(nonatomic,assign,readonly) BOOL complete;
@property(nonatomic,assign) CKDownloader* downloader;
@end


@implementation LUDownload
@synthesize film,downloadURL,complete,downloader;
@end
```

NOTHING MORE !

ADVANTAGES

Do it once

=

Test it once

Use it everywhere !

No more Copy/Paste

=

No more distraction errors !

ABOUT PROPERTIES

Name

Attributes

- **Type**: Native (int, short, ...), Pointer, SEL, Class, struct, ...
- **Assignment Type** : Assign, Retain, Copy, Weak.
- **ReadOnly**

* Attributes is a **string** that has to be **parsed**.

```
@property(nonatomic, readonly, copy) id idReadonlyCopyNonatomic;
“T@,R,C,VidReadonlyCopyNonatomic”
```

PROPERTY BASED ALGORITHMS

Runtime provides:

class_getSuperClass
class_copyPropertyList
property_getName
property_getAttributes

SAMPLE

```
- (void) encodeWithCoder:(NSCoder *)aCoder {
  NSArray* allProperties = [[self class] allPropertyDescriptors];
  for(CKClassPropertyDescriptor* property in allProperties){
     if(property.isReadOnly == NO){

        CKPropertyExtendedAttributes* attributes = [property
            extendedAttributesForInstance:self];
        if([attributes isSerializable]){

           id object = [self valueForKey:property.name];
           [aCoder encodeObject:object forKey:property.name];
        }
     }
  }
}
```

Extended Property Attributes

DYNAMIC INFORMAL PROTOCOL

1. Generates a selector name.

```
NSString* selectorName = [NSString
        stringWithFormat:@"%@ExtendedAttributes:",property.name];
```

2. Gets a selector.

```
SEL attributesSelector = NSSelectorFromString(selectorName);
```

3. Performs the selector.

```
CKPropertyExtendedAttributes* attributes =
        [CKPropertyAttributes attributes];
if([instance respondsToSelector:attributesSelector]){
      [instance performSelector:attributesSelector
                     withObject:attributes];
}
```

SAMPLE

```
@interface MyClass : CKObject {}
@property (nonatomic, copy) NSString* myProperty;
@end


@implementation MyClass
@synthesize myProperty;

- (void) myPropertyExtendedAttributes:(CKPropertyExtendedAttributes*)attributes{

   //Customize representation of myProperty in Forms
   [attributes setTableViewCellControllerClass : [CKMultilineTextTableViewCellController class]];
   [attributes setMaximumLength : 500];

   //Do not serialize myProperty when serializing MyClass Objects
   [attributes setSerializable : NO];

   //Copy myProperty when Copying MyClass Objects
   [attributes setCopiable : YES];

   //Customize validation when validating MyClass Objects
   [attributes setValidationPredicate : [NSPredicate predicateWithBlock:
        ^BOOL(id evaluatedObject, NSDictionary *bindings) {
            NSString* myPropertyValue = (NSString*)evaluatedObject;
            if ([myPropertyValue length] > 0 && [myPropertyValue length] <= 500)
                return YES;
            return NO;
   }]];
}

- (void) myPropertyChanged{
    //Do some stuff
}

@end
```

DYNAMIC PROPERTIES

Why ?

1. **Unit Tests** property based algorithm.
2. **Data Driven** class description.

class_addProperty

3. **Extends** the framework's classes.

Associated Objects

SAMPLE

```
@interface UIView (MyCategory)
@property(nonatomic,assign)CGPoint myProperty;
@end


static char UIViewMyPropertyKey;

@implementation UIView (MyCategory)
@dynamic myProperty;

- (void)setMyProperty:(CGPoint)property{
    [self willChangeValueForKey:@"myProperty"];

    objc_setAssociatedObject(self, &UIViewMyPropertyKey,
                             [NSValue valueWithCGPoint:property],
                             OBJC_ASSOCIATION_RETAIN_NONATOMIC);

    [self didChangeValueForKey:@"myProperty"];
}

- (CGPoint) myProperty{
    NSValue* value = objc_getAssociatedObject(self, & UIViewMyPropertyKey);
    return value ? [value CGPointValue] : CGPointMake(0,0);
}

@end
```

CONCLUSION

Uses the **Runtime** to:

1. **Extends** vendors framework classes.

2. **Automates** boring and useless methods based on properties.

3. **KVO/KVC** + property based approach allow to build complex systems in an easy way !

Properties are the key !

THINGS WE'VE BUILT USING THE RUNTIME

Object Graph / Single Value Conversion System

Auto Synchronized User Defaults

Generic key/value CoreData model + Auto serialization

Native IOS Cascading Stylesheets

Property Grids

InApp - Interactive Debugging Tools

END

Going deeper

http://en.wikipedia.org/wiki/Type_introspection

http://en.wikipedia.org/wiki/Reflection_(computer_programming)

https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Introduction/Introduction.html

https://developer.apple.com/library/mac/#documentation/Cocoa/Reference/ObjCRuntimeRef/Reference/reference.html

https://developer.apple.com/library/mac/#documentation/cocoa/reference/foundation/miscellaneous/foundation_functions/reference/reference.html

http://cocoasamurai.blogspot.ca/2010/01/understanding-objective-c-runtime.html

http://macdevcenter.com/pub/a/mac/2002/05/24/runtime_partone.html

http://www.jot.fm/issues/issue_2009_01/article4.pdf

http://mobileorchard.com/podcast-mike-ash-on-the-objective-c-runtime-objects-and-the-runtime-message-sending-and-no-such-method/

Understanding Property Attributes

https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Articles/ocrtPropertyIntrospection.html#//apple_ref/doc/uid/TP40008048-CH101-SW1

https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Articles/ocrtTypeEncodings.html#//apple_ref/doc/uid/TP40008048-CH100-SW1