

BloodMagic

Custom property attributes

WHOAMI

Github: AlexDenisov

IRC: AlexDenisov

Twitter: 1101_debian

Outline

- What BloodMagic is
- How to use it
- How to extend it
- How does it work
- Q & A

What BloodMagic is

Problem

```
@interface ViewController : UIViewController
```

```
@property (nonatomic) NSMutableArray *resources;
```

```
@property (nonatomic) ProgressView *progressView;
```

```
@property (nonatomic) ErrorView *errorView;
```

```
@property (nonatomic) DataLoader *dataLoader;
```

```
@end
```

Lazy Initialization

```
@implementation ViewController

- (NSMutableArray *)resources
{
    if (!_resources) {
        _resources = [NSMutableArray new];
    }

    return _resources;
}

- (ProgressView *)progressView
{
    if (!_progressView) {
        _progressView = [ProgressView new];
    }

    return _progressView;
}

- (ErrorView *)errorView
{
    if (!_errorView) {
        _errorView = [ErrorView new];
    }

    return _errorView;
}

- (DataLoader *)dataLoader
{
    if (!_dataLoader) {
        _dataLoader = [DataLoader new];
    }

    return _dataLoader;
}

@end
```

A lot of Boilerplate Code

```
- (PropertyClass *)propertyName
{
    if (!_propertyName) {
        _propertyName = [PropertyClass new];
    }

    return _propertyName;
}
```

Just add 'lazy' attribute

```
@interface ViewController : UIViewController
```

```
@property (nonatomic, lazy) NSMutableArray *resources;
```

```
@property (nonatomic, lazy) CenterProgress *centerProgress;
```

```
@property (nonatomic, lazy) BottomProgress *bottomProgress;
```

```
@property (nonatomic, lazy) DataLoader *dataLoader;
```

```
@end
```


Just add 'lazy' attribute

```
@interface ViewController : UIViewController
```

```
@property (nonatomic, lazy) NSMutableArray *resources;
```

```
@property (nonatomic, lazy) CenterProgress *centerProgress;
```

```
@property (nonatomic, lazy) BottomProgress *bottomProgress;
```

```
@property (nonatomic, lazy) DataLoader *dataLoader;
```

```
@end
```

```
/tmp/lazy_test → make
```

```
...
```

```
error: unknown property attribute 'lazy'
```

```
@property (nonatomic, lazy) NSMutableArray *resources;
```

```
^
```

```
...
```

BloodMagic

a mechanism for creating
custom property attributes
based on your code

Let's add magic

```
#import <BloodMagic/Lazy.h>
```

```
@interface ViewController : UIViewController  
    <BMLazy>
```

```
@property (nonatomic) NSMutableArray *resources;
```

```
@property (nonatomic) ProgressView *progressView;
```

```
@property (nonatomic) ErrorView *errorView;
```

```
@property (nonatomic) DataLoader *dataLoader;
```

```
@end
```

Let's add magic

```
@implementation ViewController
```

```
@dynamic resources;
```

```
@dynamic progressView;
```

```
@dynamic errorView;
```

```
@dynamic dataLoader;
```

```
- (void)actOnSomething
```

```
{
```

```
    [self.dataLoader loadNextPage];
```

```
    // ^ new object created
```

```
}
```

```
@end
```

Magic Happened

```
@implementation ViewController

- (NSMutableArray *)resources
{
    if (!_resources) {
        _resources = [NSMutableArray new];
    }

    return _resources;
}

- (ProgressView *)progressView
{
    if (!_progressView) {
        _progressView = [ProgressView new];
    }

    return _progressView;
}

- (ErrorView *)errorView
{
    if (!_errorView) {
        _errorView = [ErrorView new];
    }

    return _errorView;
}

- (DataLoader *)dataLoader
{
    if (!_dataLoader) {
        _dataLoader = [DataLoader new];
    }

    return _dataLoader;
}

@end
```



BloodMagic

```
@implementation ViewController

@dynamic resources;
@dynamic progressView;
@dynamic errorView;
@dynamic dataLoader;

@end
```

How to use BloodMagic

Single Custom Attribute

```
#import <BloodMagic/Lazy.h>
```

```
@interface ViewController : UIViewController  
    <BMLazy>
```

```
@property (nonatomic, bm_lazy) DataLoader *dataLoader;
```

```
@end
```

Single Custom Attribute

```
#import "ViewController.h"
```

```
@implementation ViewController
```

```
@dynamic dataLoader;
```

```
- (void)viewDidLoad
```

```
{
```

```
    [super viewDidLoad];
```

```
    [self.dataLoader loadNextPage];
```

```
}
```

```
@end
```


Single Custom Attribute

```
#import "ViewController.h"
```

```
@implementation ViewController
```

```
@dynamic dataLoader;
```

```
- (void)viewDidLoad
```

```
{
```

```
    [super viewDidLoad];
```

```
    [self.dataLoader loadNextPage];
```

```
    // ^ new object created
```

```
}
```

```
@end
```

Multiple Custom Attributes

```
#import <BloodMagic/Lazy.h>  
#import <BloodMagic/Partial.h>
```

```
@interface ViewController : UIViewController  
    <BMLazy,  
    BMPartial>
```

```
@property (nonatomic, bm_lazy) DataLoader *dataLoader;  
@property (nonatomic, bm_partial) ErrorView *errorView;
```

```
@end
```

Multiple Custom Attributes

```
#import "ViewController.h"
```

```
@implementation ViewController
```

```
@lazy(dataLoader)
```

```
@partial(errorView)
```

```
- (void)viewDidLoad
```

```
{
```

```
    [super viewDidLoad];
```

```
    [self.dataLoader loadNextPage];
```

```
    [self.view addSubview:self.errorView];
```

```
}
```

```
@end
```

How to extend it

```
@property (nonatomic, bm_preference) NSString *cocoaHeads;
```

Module structure

BloodMagic/Sources/Modules/Preference (master ✓) → tree

|-- Private

| |-- BMPreferenceHook.h

| |-- BMPreferenceHook.m

| |-- BMPreferenceModuleLoader.h

| `-- BMPreferenceModuleLoader.m

`-- Public

`-- BMPreference.h

...

|-- BloodMagic/Sources

`-- Preference.h

Public Protocol

```
@protocol BMPreference  
    <NSObject>
```

```
@end
```

Public Header

```
#import <BloodMagic/Sources/Modules/Core/Public/BMPublicCoreDefnitions.h>
#import <BloodMagic/Sources/Modules/Preference/Public/BMPreference.h>

#ifdef bm_preference
#define bm_preference
#endif

#ifdef preference
#define preference(property_name) register_module(BMPreference, property_name)
#endif
```


Public Header

```
#import <BloodMagic/Sources/Modules/Core/Public/BMPublicCoreDefnitions.h>  
#import <BloodMagic/Sources/Modules/Preference/Public/BMPreference.h>
```

```
#ifndef bm_preference  
#define bm_preference  
#endif
```

```
#ifndef preference  
#define preference(property_name) register_module(BMPreference, property_name)  
#endif
```

Private Module Loader

```
#import <Foundation/Foundation.h>
```

```
@interface BMPreferenceModuleLoader : NSObject
```

```
@end
```

Private Module Loader

```
#import "BMPreferenceModuleLoader.h"  
#import "BMPreference.h"  
#import "BMBloodMagicInjector.h"  
  
@implementation BMPreferenceModuleLoader  
  
+ (void)load  
{  
    @autoreleasepool {  
        BMBloodMagicInjector *injector = [BMBloodMagicInjector new];  
        [injector injectBloodMagicInto:@protocol(BMPreference)];  
    }  
}  
  
@end
```

Private Hook

```
#import "BMHook.h"
```

```
#import "BMPreference.h"
```

```
@interface BMPreferenceHook : NSObject  
    <BMHook,  
    BMPreference>
```

```
@end
```

Private Hook

```
#import "BMPreferenceHook.h"
#import "BMProperty.h"

@implementation BMPreferenceHook

static inlineNSUserDefaults *bm_defaults()
{
    return [NSUserDefaults standardUserDefaults];
}

+ (void)accessorHook:(id *)value
    withProperty:(const BMProperty *)property
    sender:(__unused id)sender
{
    *value = [bm_defaults() objectForKey:property.name];
}

+ (void)mutatorHook:(id *)value
    withProperty:(const BMProperty *)property
    sender:(__unused id)sender
{
    [bm_defaults() setObject:*value forKey:property.name];
}

@end
```

Private Hook

Accessor

```
static inline NSUserDefaults *bm_defaults()  
{  
    return [NSUserDefaults standardUserDefaults];  
}  
  
+ (void)accessorHook:(id *)value  
    withProperty:(const BMProperty *)property  
    sender:(__unused id)sender  
{  
    *value = [bm_defaults() objectForKey:property.name];  
}
```

Private Hook

Mutator

```
static inlineNSUserDefaults *bm_defaults()
{
    return [NSUserDefaults standardUserDefaults];
}

+ (void)mutatorHook:(id *)value
    withProperty:(const BMProperty *)property
    sender:(__unused id)sender
{
    [bm_defaults() setObject:*value forKey:property.name];
}
```

BMPreference Usage

```
#import <BloodMagic/Preference.h>
```

```
@interface Settings : NSObject  
    <BMPreference>
```

```
@property (nonatomic, bm_preference) NSString *name;
```

```
@property (nonatomic, bm_preference) NSUInteger age;
```

```
@end
```


BMPreference Usage

```
#import "Settings.h"
```

```
@implementation Settings
```

```
@dynamic name;
```

```
@dynamic age;
```

```
@end
```

BMPreference Usage

```
Settings *settings = [Settings new];  
settings.name = @"AlexDenisov";  
settings.age = 26;  
  
// ...  
NSLog(@"%@@", defaults);
```

BMPreference Usage

```
Settings *settings = [Settings new];
settings.name = @"AlexDenisov";
settings.age = 26;

// ...
NSLog(@"%@@", defaults);

{
...
    age = 26;
    name = AlexDenisov;
...
}
```

How does it work

BloodMagic is modular

~/Projects/BloodMagic/Sources (master ✓) → tree -L 2

```
.
├──
└── -- Modules
    |-- Core
    |-- Final
    |-- Injectable
    |-- Initializers
    |-- Lazy
    |-- Partial
    └── -- Preference
```

BloodMagic is modular

~/Projects/BloodMagic/Sources (master ✓) → tree -L 2

```
.
├── Modules
│   ├── Core
│   ├── Final
│   ├── Injectable
│   ├── Initializers
│   ├── Lazy
│   ├── Partial
│   └── Preference
```

Property attributes
implementation

BloodMagic is modular

`~/Projects/BloodMagic/Sources (master ✓) → tree -L 2`

```
.
├── Modules
│   ├── Core
│   ├── Final
│   ├── Injectable
│   ├── Initializers
│   ├── Lazy
│   ├── Partial
│   └── Preference
```

Low level logic

- hooks
- injections
- runtime routines

Core Module: Hooks

```
@protocol BMHook  
    <NSObject>
```

```
@optional
```

```
+ (void)mutatorHook:(id *)value  
    withProperty:(const BMProperty *)property  
    sender:(id)sender;
```

```
+ (void)accessorHook:(id *)value  
    withProperty:(const BMProperty *)property  
    sender:(id)sender;
```

```
@end
```


Core Module: Injections

```
@interface BMBloodMagicInjector : NSObject
```

```
- (void)injectBloodMagicInto:(Protocol *)protocol;
```

```
@end
```

Core Module: Injections

```
@implementation BMBloodMagicInjector
```

```
// pseudocode
```

```
- (void)injectBloodMagicInto:(Protocol *)protocol
```

```
{
```

```
    class_list_t classes = collectClasses(protocol);
```

```
    property_list_t properties = collectDynamicProperties(classes);
```

```
    for (Property *property in properties) {
```

```
        Hook *hook = hookForProperty(property);
```

```
        property.accessor = hook.accessor;
```

```
        property.mutator = hook.mutator;
```

```
    }
```

```
}
```

```
@end
```

Sources:

<https://github.com/railsware/BloodMagic>

Blog-post:

<http://l.rw.rw/dibm>

Slides:

<https://speakerdeck.com/alexdenisov/bloodmagic>

<https://speakerdeck.com/0xc010d/dependency-injection-ftw>

Questions?