Desenvolvimento para iPhone

Usando Objective-C e iOS SDK 8

Agenda

- Formas de construir um novo objeto;
- Tipagem dinâmica;
- Instrospecção;
- Selector;
- Mais sobre Protocolos;

Objetivos do dia

- Sanar todas as dúvidas na criação de objetos;
- Entender os mecanismos de tipagem dinâmica;
- Compreender os conceitos de introspecção;
- Saber como e quando usar o @selector;
- Entender as novidades a respeito dos Protocolos;

Criação de Objetos - Construtores

```
NSArray *array = [[NSArray alloc] init];
```

```
NSMutableDictionary *dictionary = [[NSMutableDictionary alloc] initWithCapacity:10];
```

```
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Mensagem"
                                                message:@"Declare variables, not war."
                                               delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
```

Novidade do dia!

```
NSSet *set = [NSSet new]; = NSSet *set = [[NSSet alloc] init];
```

Criação de Objetos - Construtores

```
NSMutableDictionary *dictionary = [[NSMutableDictionary alloc] initWithCapacity:10];
```

Designated Initializer

Designated Initializer

```
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Mensagem"
                                                message:@"Declare variables, not war."
                                               delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
```

Criação de Objetos - Construtores

```
NSSet *set = [NSSet new];
```

Somente usa o inicializador padrão. Dito isso, nenhum *designated initizalizer* poderá ser usado com esta sintaxe.

Criação de Objetos – Métodos de Classe

```
NSString *minhaIdade = [NSString stringWithFormat:@"Minha idade é: %d", 18];

UIButton *botao = [UIButton buttonWithType:UIButtonTypeSystem];

NSArray *array = [NSArray arrayWithObjects:@"Carro", @"Bike", @"Moto", nil];
```

- Internamente, estes métodos usam um inicializador, seja o padrão ou um *designated* para construir e entregar uma nova instância do objeto;
- Esta construção existe para oferecer um atalho na hora de codificar. É aplicável a outras linguagens!

Criação de Objetos - Exemplos

```
12 @interface Exemplo ()
13 @property (strong, nonatomic) NSString *nome;
14 @property (strong, nonatomic) NSNumber *idade;
15 @end
16
17 @implementation Exemplo
18
19 - (instancetype) initWithNome: (NSString *) nome {
20     self = [super init];
21     if (self) {
22         [self setNome:nome];
23     }
24     return self;
25 }
26
27 + (instancetype) exemploComNome: (NSString *) nome {
28     return [[Exemplo alloc] initWithNome:nome];
29 }
30
31 + (instancetype) exemploComNome: (NSString *) nome idade: (NSNumber *) idade {
32     Exemplo *exemplo = [[Exemplo alloc] init];
33     [exemplo setNome:nome];
34     [exemplo setIdade:idade];
35     return exemplo;
36 }
37
38 @end
```

Símbolo "+" representa método de classe.

Tipagem Dinâmica

- O Objective-C tem um tipo coringa muito importante, que se chama **id**;
  - O significado de **id** é:
    - Ponteiro para um objeto de tipo desconhecido ou não especificado.
- Em tempo de execução, TODOS os objetos são tratados como **id**;
- Requer muito cuidado ao usar!

Tipagem Dinâmica

```
38 - (void) cuidadoComTipagemDinamica {
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57 }
```

Tipagem Dinâmica

```
20 @interface Veiculo : NSObject
21 - (void) mover;
22 @end
23
24 @interface Tanque : Veiculo
25 - (void) atirar;
26 @end
27
28
29 @implementation Exemplo
30 - (void) exemplo {
31
32     Tanque *tanque = [[Tanque al      init];
33     [tanque mover];
34     [tanque atirar];
35
36
37     Veiculo *v =  anque;
38     [v atirar];
39 }
40
41
42 @end
```

Erro de compilação!

- Esse exemplo específico funcionaria, porque **v** é um Tanque;
- Mas o compilador não sabe disso.

Introspecção e Selector

- Todos os objetos filhos de NSObject sabem fazer introspecção:
- **isKindOfClass:**
  - Retorna **YES**, se o objeto for do mesmo tipo do parâmetro (incluindo a árvore de herança)
- **isMemberOfClass:**
  - Idem anterior, porém ignora a herança
- **respondsToSelector:**
  - Retorna **YES** se o objeto for capaz de responder a uma dada mensagem (método)

Introspecção e Selector

```
NSString *s = @"s";

[s isKindOfClass: [NSObject class]]; //YES

[s isMemberOfClass: [NSNumber class]]; //NO

[s respondsToSelector: @selector(rangeOfString:)]; //YES
```

Introspecção e Selector

Sintaxe para armazenar um **@selector** em uma variável

```
NSString *s = @"x";
NSArray *numeros = @[@1, @2, @3, @4];

SEL umSelector = @selector(description);
SEL selectorComArgumento = @selector(stringByAppendingString:);
SEL maisDeUmArgumento = @selector(arrayWithObjects:count:);

[s respondsToSelector:umSelector]; // retorna BOOL

[numeros makeObjectsPerformSelector:umSelector]; //retorna void

[s performSelector:umSelector]; //retorna id
NSString *s1 = [s performSelector:selectorComArgumento withObject:@"1"]; //retorna id
//s1 contém a string "x1"
```

Protocolos

- Juntando os conceitos de tipagem dinâmica e introspecção, o conceito de protocolo ganha um novo significado!


ExemploAula02 › iPhone 6
ExemploAula02 | Build ExemploAula02: Succeeded | 30/04/15 at 21:56
ExemploAula02 › E...02 › Exemplo.m › -exemplo
Manual › Exemplo.h › No Selection
#import "Exemplo.h"
#import <UIKit/UIKit.h>
@interface Exemplo ()
@end
@implementation Exemplo
- (void) exemplo {
NSString *s = @"x";
/*
...
várias
...
*/
[self.delegate
}
@end
#import <Foundation/Foundation.h>
@protocol ExemploDelegate <NSObject>
- (void) trabalhoDelegado;
@end
@interface Exemplo : NSObject
@property (strong, nonatomic) id<ExemploDelegate> delegate;
@end
É assim que funciona a comunicação cega entre as camadas do MVVC!
delegate é do tipo id, e deve estar em conformidade com o protocolo ExemploDelegate.
A definição do protocolo pertence a Exemplo.

Hora de brincar!

- NSRange
- NSUserDefaults
- Literais;
- Arquivo de propriedes (.plist)

Desafio

- Carregar um NSArray de strings a partir de um arquivo plist;
- Mostrar todos os elementos em um UITableView;
- Ao clicar em uma célula, navegar para uma outra tela, onde o "detalhe" daquela célula é apresentado.