

App Agenda Persistência

Parte 2 - Gravando e recuperando dados com UserDefaults

X-Code com Objective-C
Prof. Agesandro Scarpioni

Atenção

- Para iniciar esse conjunto de slides é necessário fazer e preparar as telas do conjunto de slides anterior chamado: Aula_XX_AAAA_App_Agenda_Tela.

NSUserDefaults

- Uma das formas mais simples de gravar dados e recuperar as informações sem utilizar banco é com a classe `NSUserDefaults`, é possível gravar números, booleanos, strings e outros objetos, ela é muito utilizada para armazenar preferências do usuário em um aplicativo.
- A classe `NSUserDefaults` internamente utiliza um banco de dados, mas tudo isso fica encapsulado e o acesso é transparente para o desenvolvedor, a classe utiliza uma estrutura de chave e valor.

NSUserDefaults

- Todos os dados retornados pela classe NSUserDefaults são imutáveis e não podem ser alterados, se for necessário alterar algum dado, é preciso salvar novamente os objetos.
- Vamos utilizar a tela de agenda apenas para fazermos nossos testes, porém, NSUserDefaults não é a forma correta para armazenar uma lista de contatos, para esse tipo de gravação vamos usar sqlite no próximo conjunto de slides. Lembre-se que estamos gravando as preferências do usuário isso vai funcionar como se estivéssemos gravando no aparelho o nome, sobrenome e telefone do proprietário do dispositivo.

Implementando NSUserDefaults

- No método Gravar, inclua as linhas sinalizadas abaixo para armazenar os dados e após a mensagem limpe os campos.

```
31 - (IBAction)Gravar:(id)sender {
32
33     [txtNome resignFirstResponder];
34     [txtSobrenome resignFirstResponder];
35     [txtTelefone resignFirstResponder];
36     if ([txtNome.text isEqualToString:@""] || [txtSobrenome.text isEqualToString:@""] ||
37         [txtTelefone.text isEqualToString:@""]){
38         UIAlertView *a = [[UIAlertView alloc] initWithTitle:@"Erro" message:@"Preencha todos os
39             campos" delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil];
40         [a show];
41     }
42
43     NSUserDefaults *dados = [NSUserDefaults standardUserDefaults];
44     [dados setValue:txtNome.text forKey:@"nome"];
45     [dados setValue:txtSobrenome.text forKey:@"sobrenome"];
46     [dados setValue:txtTelefone.text forKey:@"telefone"];
47     [dados synchronize];
48
49
50     // uma forma diferente de concatenar uma string
51     NSString *texto = @"Ok agenda gravada para ";
52     texto = [texto stringByAppendingString:txtNome.text];
53     texto = [texto stringByAppendingString:@" "];
54     texto = [texto stringByAppendingString:txtSobrenome.text];
55
56     UIAlertView *a = [[UIAlertView alloc] initWithTitle:@"Aviso" message:texto delegate:self
57         cancelButtonTitle:@"Ok" otherButtonTitles:nil];
58     [a show];
59
60     txtNome.text = @"";
61     txtSobrenome.text = @"";
62     txtTelefone.text = @"";
63
64
65 }
```


NSUserDefaults

- Foi criado um objeto chamado "dados" do tipo NSUserDefaults, nesse objeto foi armazenado (setValue) as 3 caixas de texto com as seguintes chaves nome, sobrenome e telefone (forkey). Poderíamos ter criado 3 variáveis, passado as 3 caixas de texto para as mesmas e depois usá-las no setValue ao invés de usar diretamente a caixa de texto.
- Depois de passar os valores e suas respectivas chaves chamamos o método synchronize para sincronizar as informações com o banco de dados interno.

```
42  
43     NSUserDefaults *dados = [NSUserDefaults standardUserDefaults];  
44     [dados setValue:txtNome.text forKey:@"nome"];  
45     [dados setValue:txtSobrenome.text forKey:@"sobrenome"];  
46     [dados setValue:txtTelefone.text forKey:@"telefone"];  
47     [dados synchronize];  
48
```

OBS: Como foi informado no início a classe NSUserDefaults é muito simples de ser usada, e é ideal para armazenar informações de preferências do usuário em um determinado aplicativo.

Testando

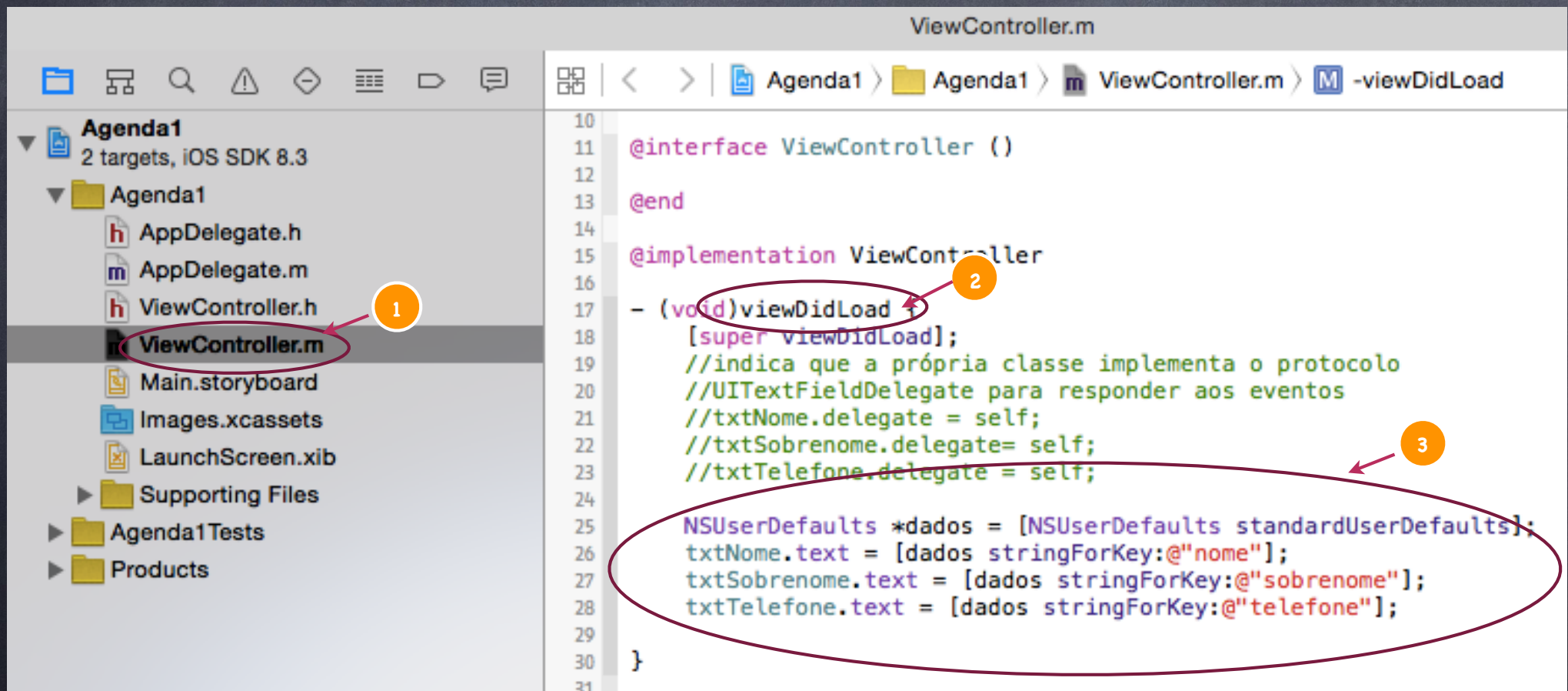
- Execute o aplicativo Command + R, digite um nome, um sobrenome e um telefone. Feche o aplicativo.
- Pronto os dados foram salvos.
- Para resgatarmos as informações vamos recuperá-las no viewDidLoad. Para resgatar as informações é tão simples quanto armazená-las.

OBS: Como foi informado no início a classe UserDefaults é muito simples de ser usada, e é ideal para armazenar informações de preferências do usuário em um determinado aplicativo.

Resgatando os dados

FIAP

- Vamos ler as informações quando a interface for carregada, por isso escolhemos o método viewDidLoad.



Resgatando os dados

- As linhas abaixo criam um objeto do tipo `NSUserDefaults` chamado "dados", passamos os valores de cada chave (nome, sobrenome, telefone) armazenados no objeto para as respectivas caixas de texto. Aqui também poderíamos ter passado as informações para variáveis e depois para as caixas de texto.

```
24  
25     NSUserDefaults *dados = [NSUserDefaults standardUserDefaults];  
26     txtNome.text = [dados objectForKey:@"nome"];  
27     txtSobrenome.text = [dados objectForKey:@"sobrenome"];  
28     txtTelefone.text = [dados objectForKey:@"telefone"];  
29
```

OBS: Como foi informado no início a classe `NSUserDefaults` é muito simples de ser usada, e é ideal para armazenar informações de preferências do usuário em um determinado aplicativo.

Testando

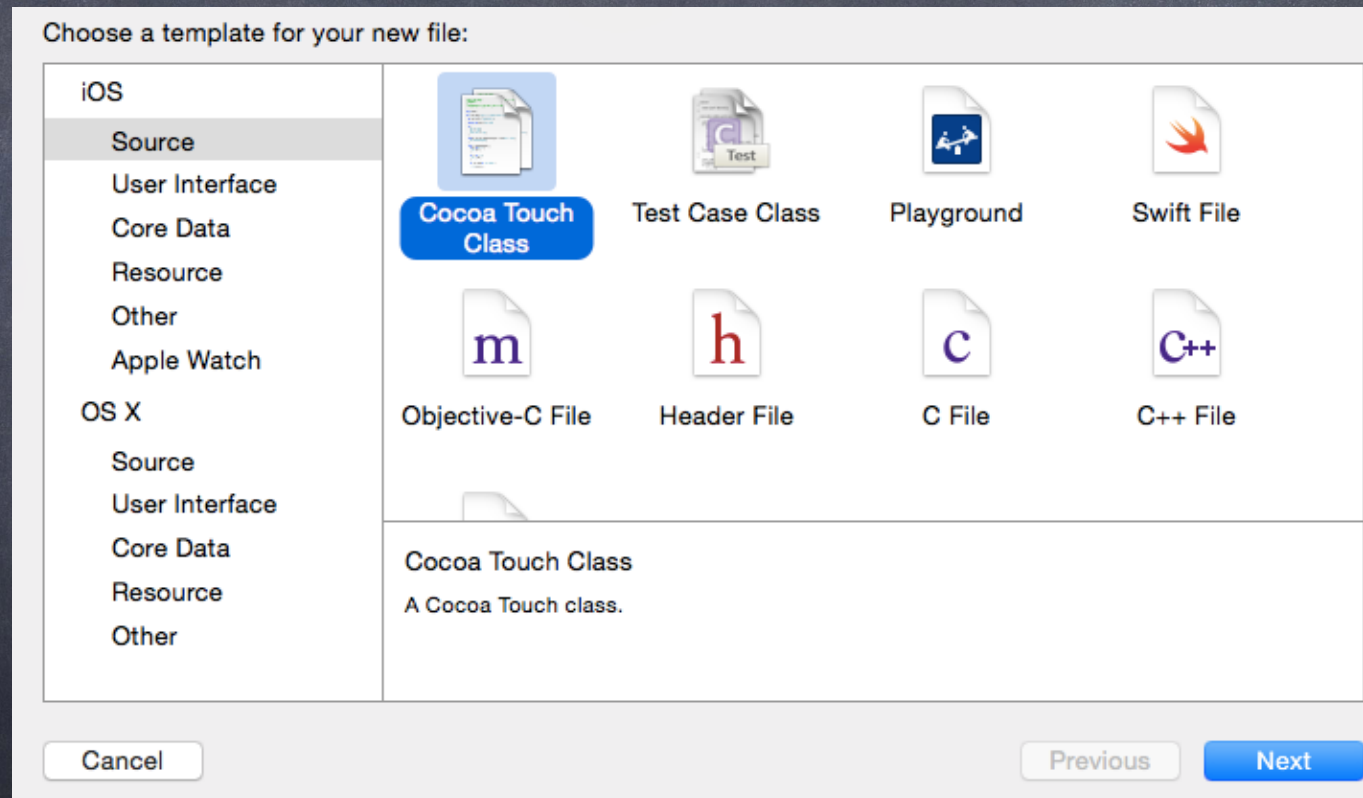
- Ao executar o aplicativo novamente, os dados armazenados com UserDefaults irão aparecer.



The screenshot shows an iOS Simulator window titled "iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.3...". The status bar at the top displays "Carrier" with a signal icon, the time "4:33 PM", and a battery icon. The app interface consists of three text input fields with labels above them: "Nome" (containing "Agesandro"), "Sobrenome" (containing "Scarpioni"), and "Telefone" (containing "123456"). At the bottom of the form is a blue button labeled "Gravar".

Encapsulando o acesso ao UserDefaults

- Adicione uma nova classe chamada "Preferencias" clicando em File->New->File-> ou Command+N, escolha Cocoa Touch Class e Clique em Next.



Encapsulando o acesso ao UserDefaults

- Classe "Preferencias", subclasse de NSObject

Choose options for your new file:

Class:

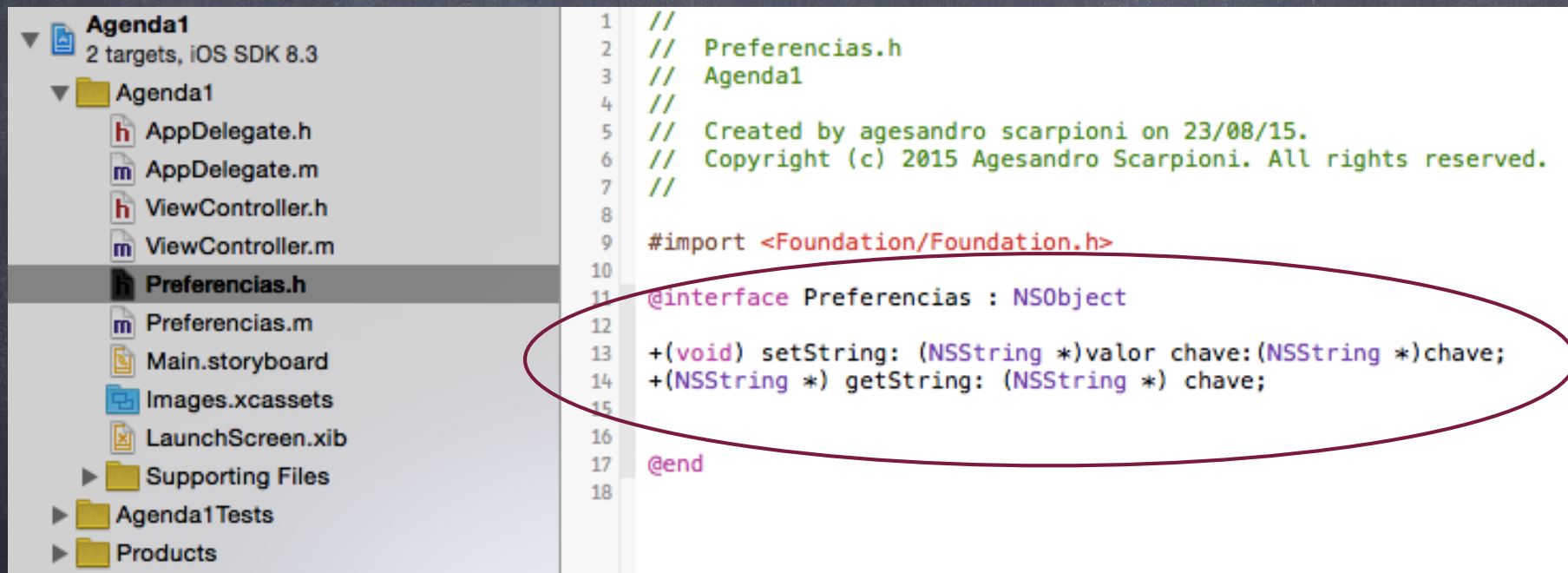
Subclass of:

☐ Also create XIB file

Language:

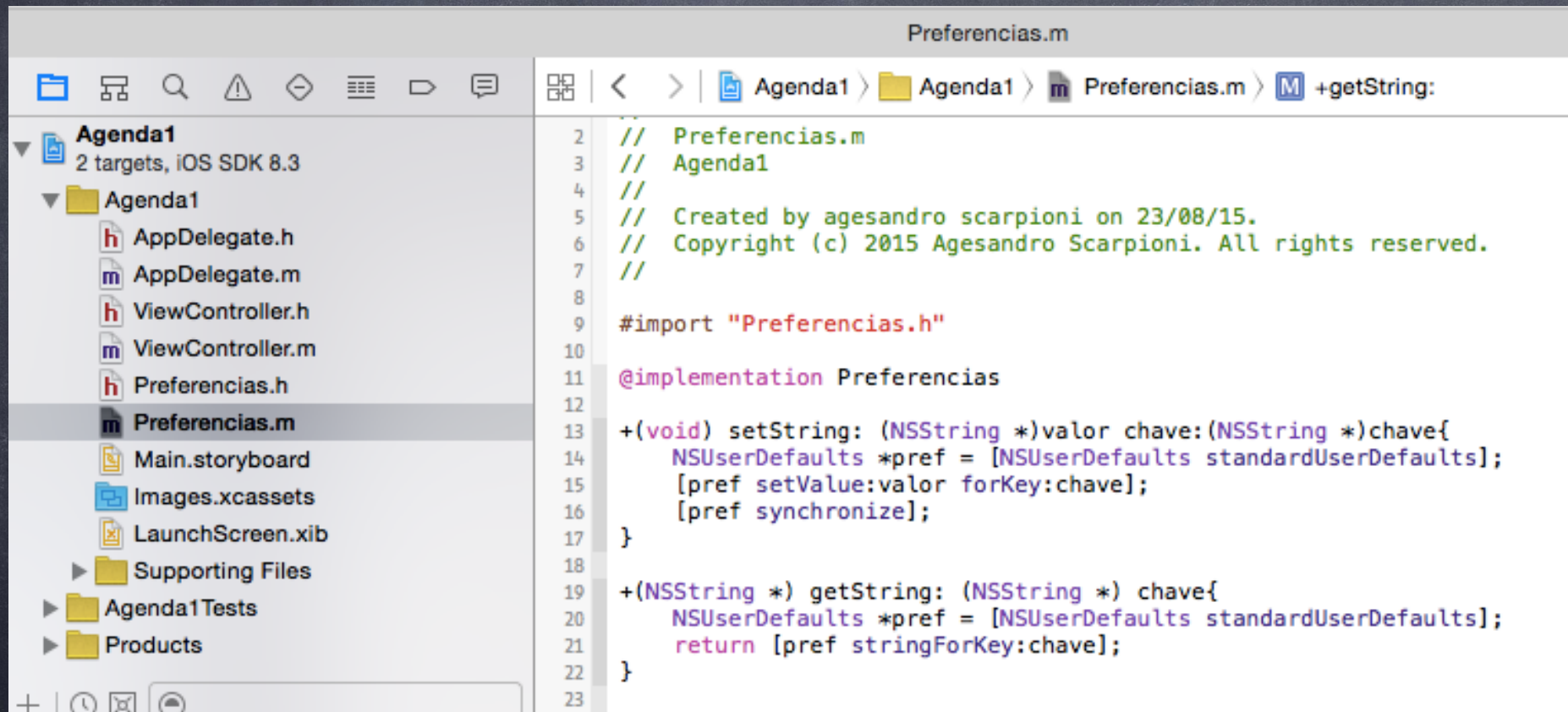
Encapsulando o acesso ao UserDefaults

- Declarar no Preferencias.h os dois métodos abaixo :



Encapsulando o acesso ao UserDefaults

- Implementar no Preferencias.m os dois métodos um para o set e outro para o get, veja o exemplo abaixo:



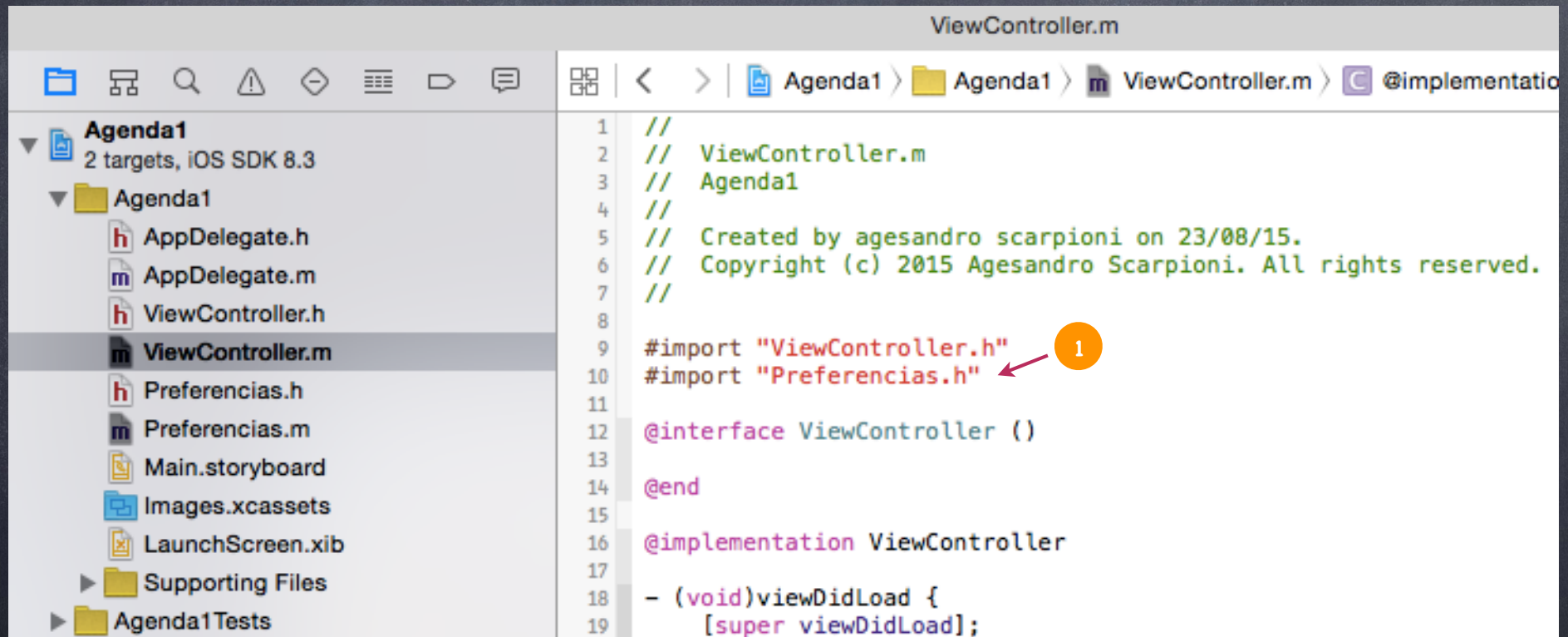
The screenshot shows the Xcode interface with the project 'Agenda1' selected in the left sidebar. The file 'Preferencias.m' is open in the editor. The code implements two methods: `setString:` and `getString:`, both using `NSUserDefaults` to store and retrieve data. The `setString:` method takes a string value and a key, sets the value, and synchronizes. The `getString:` method takes a key and returns the stored string value.

```
2 // Preferencias.m
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 23/08/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "Preferencias.h"
10
11 @implementation Preferencias
12
13 +(void) setString: (NSString *)valor chave:(NSString *)chave{
14     NSUserDefaults *pref = [NSUserDefaults standardUserDefaults];
15     [pref setValue:valor forKey:chave];
16     [pref synchronize];
17 }
18
19 +(NSString *) getString: (NSString *) chave{
20     NSUserDefaults *pref = [NSUserDefaults standardUserDefaults];
21     return [pref stringForKey:chave];
22 }
23
```

Dica: Para implementar copie as declarações do método no .h, cole no .m retire o (;) e coloque { }

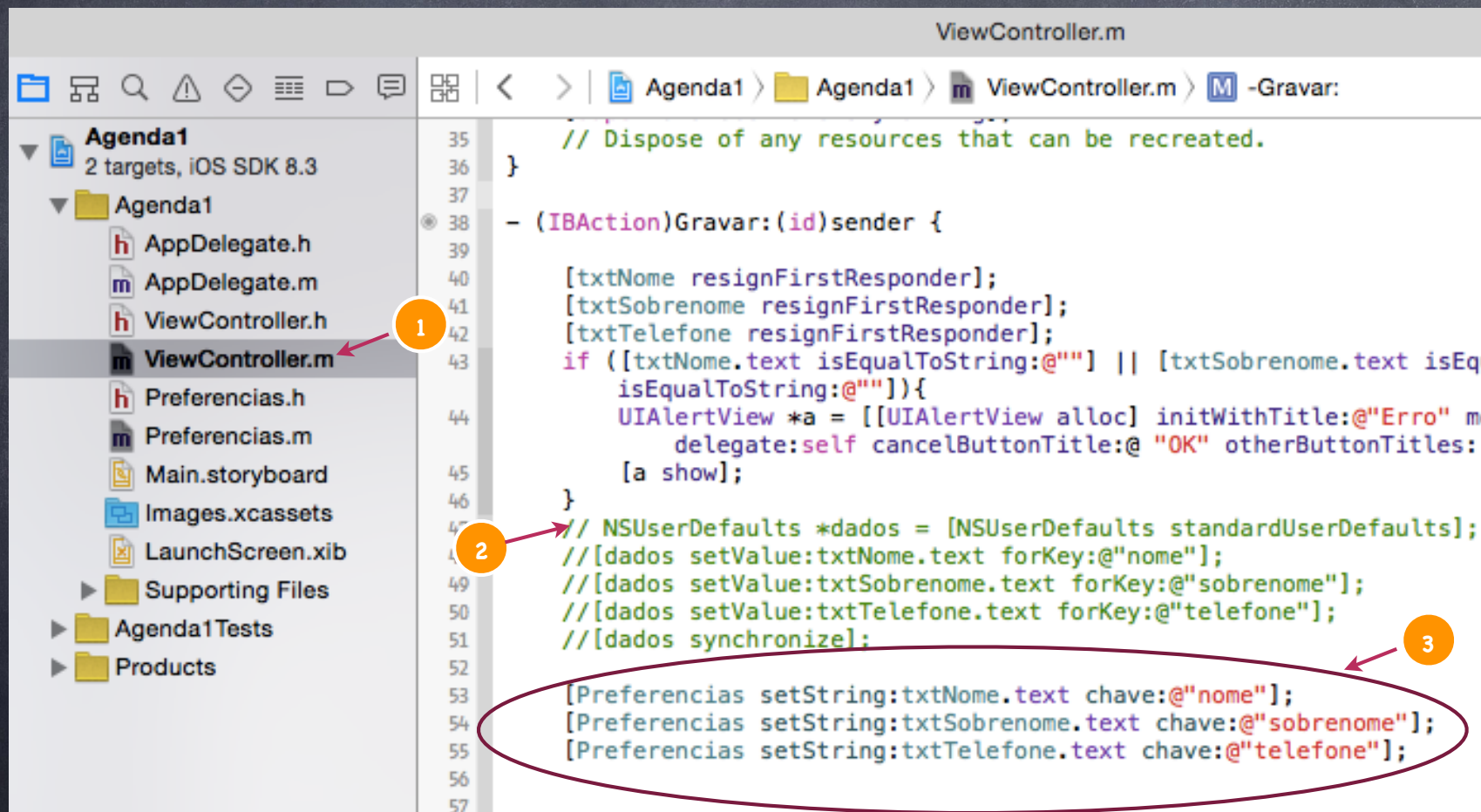
Encapsulando o acesso ao UserDefaults

- Faça o import da classe Preferencias no ViewController.m



Encapsulando o acesso ao UserDefaults

- Ainda no ViewController.m após o import, comente a antiga forma de acessar o UserDefaults e reescreva a forma encapsulada.



Testando

- Execute o aplicativo Command + R, digite um outro nome, um outro sobrenome e um outro telefone. Feche o aplicativo.
- Pronto os dados foram substituídos (salvos), utilizando a classe "Preferencias".
- Feche o aplicativo e rode novamente, veja que irá aparecer os novos dados salvos pela nova classe, pois a antiga forma está comentada.

Dica: Com a nova classe, criamos um método que salva e recupera string, podemos incluir métodos que salvam inteiros, booleanos, NSArray, double, float, etc.

Encapsulando o acesso ao UserDefaults

- Falta apenas comentar a antiga forma de exibição dos dados no método viewDidLoad do ViewController.m e chamar a forma encapsulada pela classe "Preferencias".

```
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "ViewController.h"
10 #import "Preferencias.h"
11
12 @interface ViewController ()
13
14 @end
15
16 @implementation ViewController
17
18 - (void)viewDidLoad {
19     [super viewDidLoad];
20     //indica que a própria classe implementa o protocolo
21     //UITextFieldDelegate para responder aos eventos
22     //txtNome.delegate = self;
23     //txtSobrenome.delegate = self;
24     //txtTelefone.delegate = self;
25
26     // UserDefaults *dados = [NSUserDefaults standardUserDefaults];
27     // txtNome.text = [dados stringForKey:@"nome"];
28     // txtSobrenome.text = [dados stringForKey:@"sobrenome"];
29     // txtTelefone.text = [dados stringForKey:@"telefone"];
30
31     txtNome.text = [Preferencias getString:@"nome"];
32     txtSobrenome.text = [Preferencias getString:@"sobrenome"];
33     txtTelefone.text = [Preferencias getString:@"telefone"];
34
35 }
```

OBS: Aqui também poderíamos criar variáveis para receber os dados e depois passá-las para as caixas de texto, eu optei por passar diretamente pois todas eram do tipo String.

NSUserDefaults

- Aprendemos de forma simples a manipular dados com NSUserDefaults, o treino com este componente é importante, pois muitas tarefas com relação a armazenamento podem ser resolvidas rapidamente aumentando a produtividade do desenvolvedor.

Mais métodos da classe NSUserDefaults

- Abaixo segue uma lista de diversos métodos para salvar tipos simples de dados como números, booleanos, strings mas também podem salvar objetos complexos como NSArray e NSData.
- Todos estes métodos funcionam da mesma forma e salvam determinado tipo de dado para a chave fornecida.

```
23 -(void) setBool: (BOOL) value forKey:(NSString *)defaultName;  
24  
25 -(void) setDouble: (double) value forKey:(NSString *)defaultName;  
26  
27 -(void) setFloat: (float) value forKey:(NSString *)defaultName;  
28 |  
29 -(void) setInteger: (NSInteger) value forKey:(NSString *)defaultName;  
30  
31 -(void) setObject: (id) value forKey:(NSString *)defaultName;  
32  
33 -(void) setURL: (NSURL *)url forKey:(NSString *)defaultName;
```

OBS: O setObject permite salvar qualquer tipo de objeto, como NSString, NSNumber, NSDate, NSData, NSArray, NSDictionary.

Mais métodos da classe NSUserDefaults

- Os métodos abaixo servem para ler os valores a partir da chave.

```
39  
40 - (NSArray *) arrayForKey: (NSString *) defaultName;  
41  
42 - (BOOL) boolForKey: (NSString *) defaultName;  
43  
44 - (NSData *) dataForKey: (NSString *) defaultName;  
45  
46 - (NSDictionary *) dictionaryForKey: (NSString *) defaultName;  
47  
48 - (float) floatForKey: (NSString *) defaultName;  
49  
50 - (NSInteger) integerForKey: (NSString *) defaultName;  
51  
52 - (id) objectForKey: (NSString *) defaultName;  
53  
54 - (NSString *) stringForKey: (NSString *) defaultName;  
55  
56 - (double) doubleForKey: (NSString *) defaultName;  
57  
58 - (NSURL *) URLForKey: (NSString *) defaultName;  
59
```

OBS: Lembre-se que todos os dados retornados pela classe NSUserDefaults são imutáveis e não podem ser alterados, se for necessário alterar algum dado, é preciso salvar novamente os objetos.

Prática

Criação de um programa para testarmos todos os conceitos deste tópico.

- Crie um projeto novo com um campo string, um campo float, um campo booleano e outro integer, desenvolva uma interface rapidamente sem muitos recursos de teclado virtual.
- Em um botão gravar, faça a gravação e a exibição dos dados utilizando UserDefaults, utilize uma classe para as preferências como fizemos com o programa anterior.
- Como uma sugestão para outro exercício de fixação, você pode utilizar a tela da aula prática anterior onde tínhamos 4 campos (Funcionário, cargo, departamento e salário), grave e exiba os dados com NSUserDefaults.

Próxima aula

- Utilizando Core Data para persistir dados