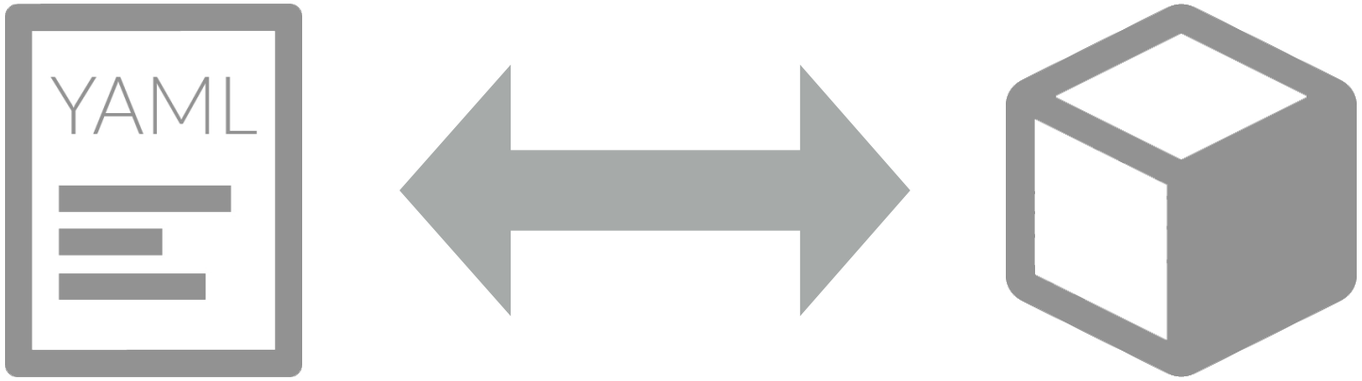
Trabalhando com YAML



Kaê Angeli Coutinho

O que é YAML?

- Padrão de serialização de dados
- Muito intuitivo, similar a uma lista de compras
- Suporte a diversas linguagens
- JSON turbinado

Exemplo YAML

```yaml
Name: YAML Tutorial
Creator: Kaê Angeli Coutinho
Platform: iOS 8.0
ListExample:
    - 1
    - 2
    - 3
    - 4
    - 5
YetAnotherListExample: [Both,Literal,And,Numerical,Types,Are,Accepted]
```

CocoaPods

- Gerenciador de dependencias (bibliotecas) do Objective-C
- Repositório padrão e central para varias bibliotecas de terceiros
- Melhor desempenho na inclusão de bibliotecas
- Manutenção e atualização muito mais ágil e fácil
- Alta escalabilidade

Como adquirir CocoaPods?

- Abra a aplicação Terminal

- Atualize o RubyGems

```
sudo gem update --system
```

- Instale o CocoaPods

```
sudo gem install cocoapods -y
```

- Inicialize o CocoaPods

```
pod setup
```

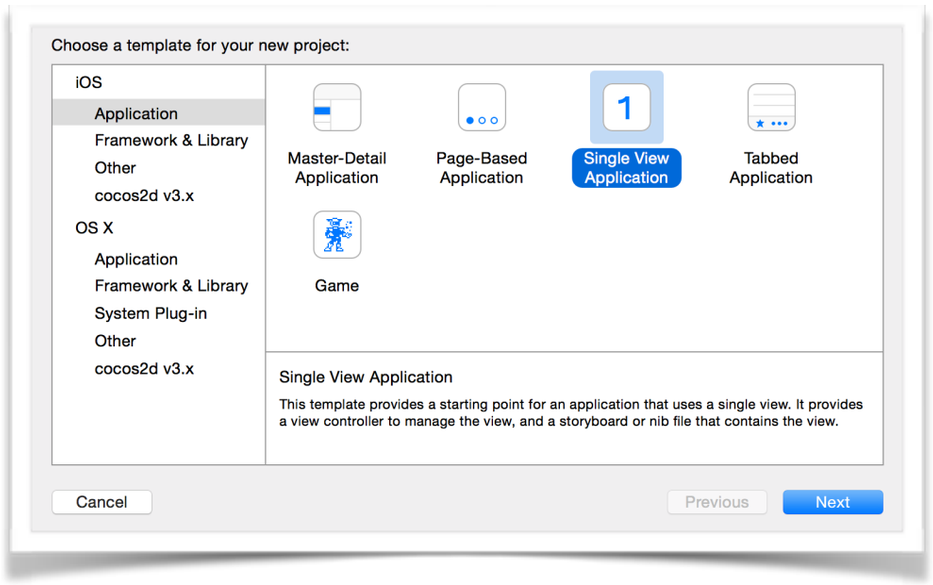
Mãos à obra

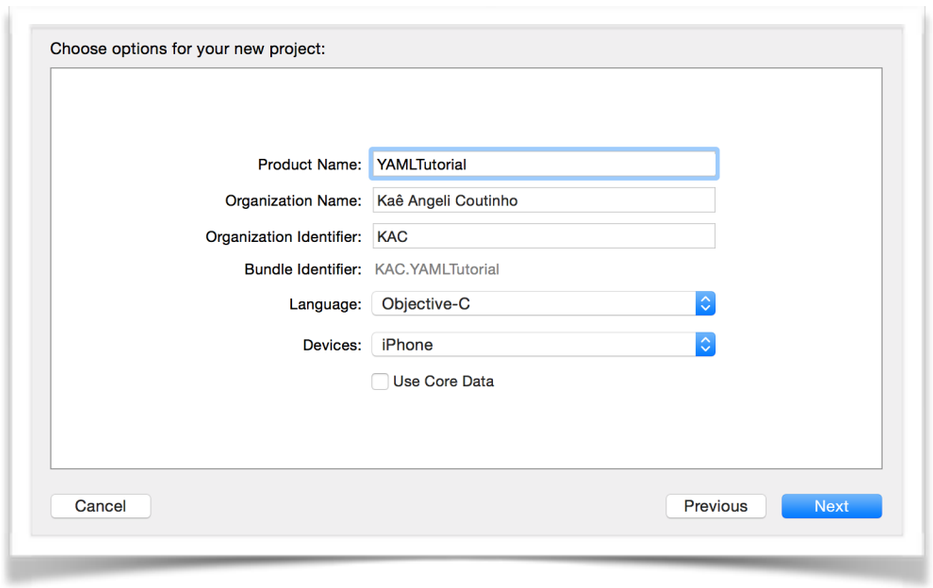
- Neste projeto, iremos fazer uso do YAML Framework, uma biblioteca para iOS que permite serializar e desserializar arquivos YAML e objetos

- Para tal, vamos usar o nosso novo gerenciador de dependências, vulgo CocoaPods

- Iremos delegar a responsabilidade de encontrar a biblioteca (versão mais recente) e agrega-la ao projeto, à ele

- Vamos criar um projeto inicialmente vazio, apenas para podermos acionar o gerenciador de dependências

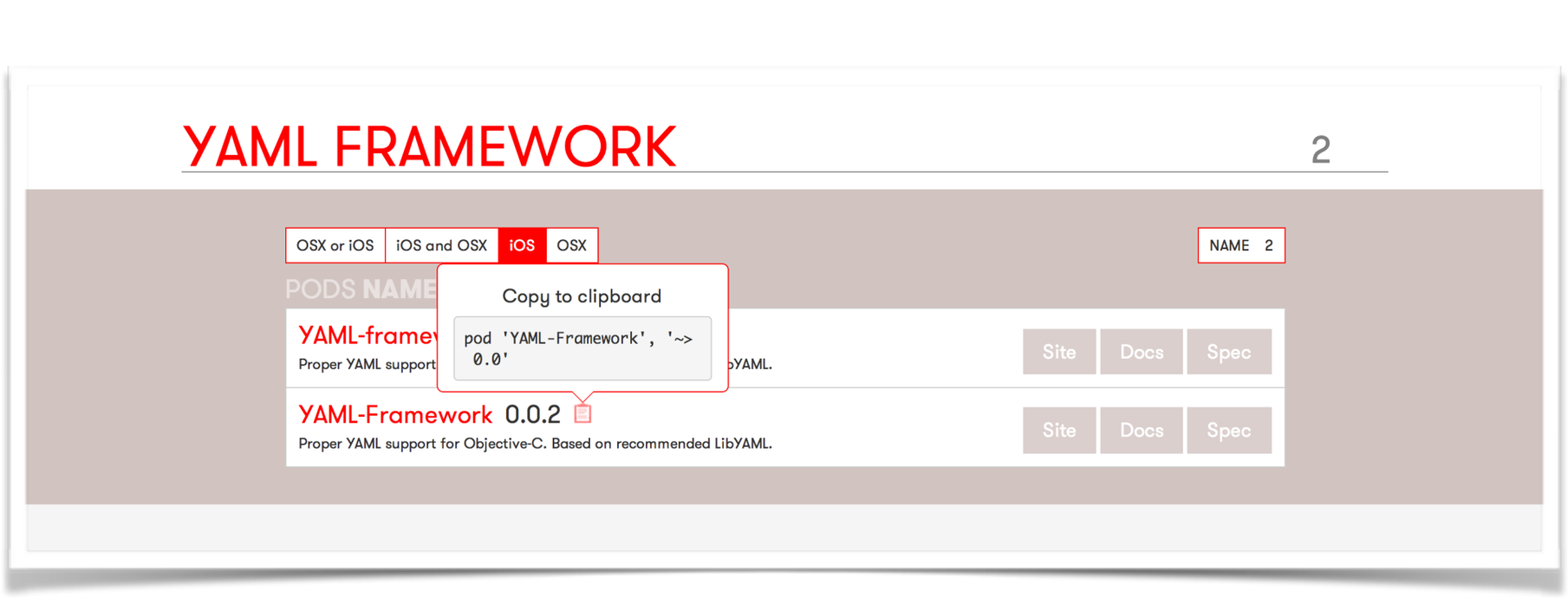




- Todo projeto que utiliza CocoaPods deve conter um Podfile

- Podfiles especificam para o gerenciador quais bibliotecas serão utilizadas no projeto, e muitas outras configurações de projeto (plataforma alvo, fonte, etc)

- Acesso as bibliotecas disponíveis pode ser feito através do site do CocoaPods (www.cocoapods.org)



- Sabendo o nome da biblioteca necessária, chegou a hora de criar o Podfile para o nosso projeto

- Abra a aplicação Terminal

    - Navegue até a pasta raiz do projeto

```
cd ~/caminho/YAMLTutorial
```

    - Habilite o CocoaPods para o projeto

```
pod init
```

- Crie o Podfile

```
touch Podfile
```

- O arquivo Podfile fora criado

- Lembrando que o Podfile deve estar contido na pasta raiz do projeto

- Utilize seu editor de texto preferido, neste projeto fora utilizado o Sublime Text 3

```
# Podfile
# Created by Kaê Angeli Coutinho

platform :ios, "7.0"

source 'https://github.com/CocoaPods/Specs.git'

target "YAMLTutorial" do

pod 'YAML-Framework', '~> 0.0'

end
```

- Ja com o Podfile corretamente estruturado, precisamos mandar o CocoaPods instalar nossas dependências

- Abra a aplicação Terminal

  - Navegue até a pasta raiz do projeto

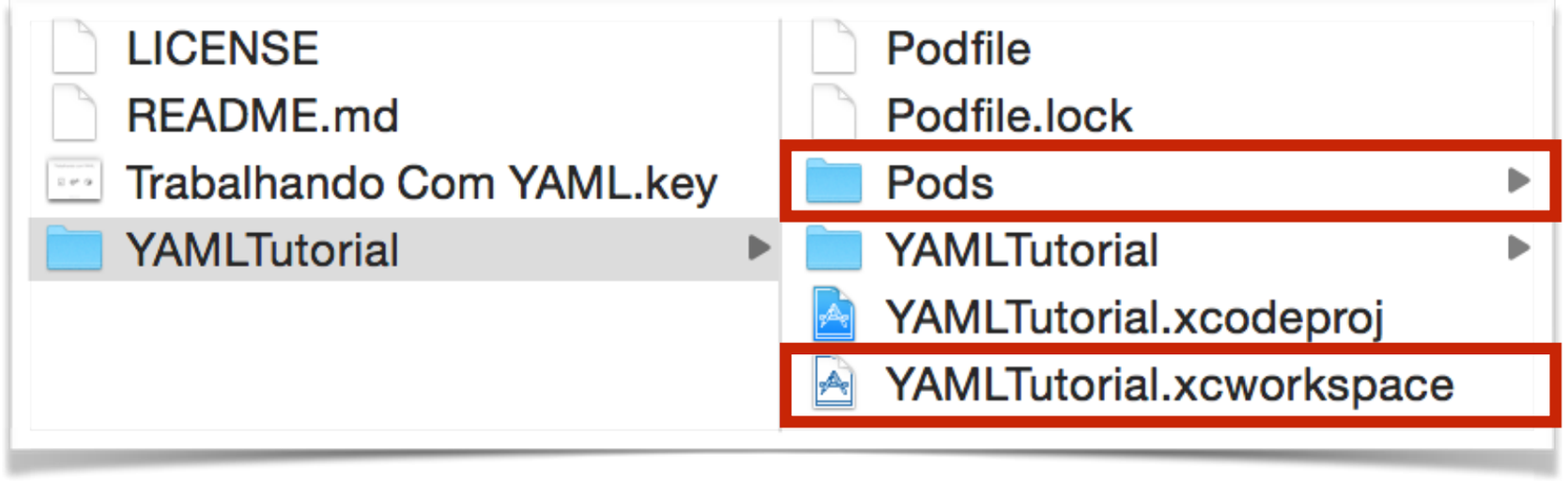
```
cd ~/caminho/YAMLTutorial
```

  - Instale as dependências

```
pod install
```

- Note que uma pasta Pods fora criada, juntamente com o YAMLTutorial.xcworkspace

- A partir de agora, utilizaremos o arquivo xcworkspace para trabalharmos no projeto, já que o mesmo é um invólucro do nosso projeto com as dependências que necessitamos

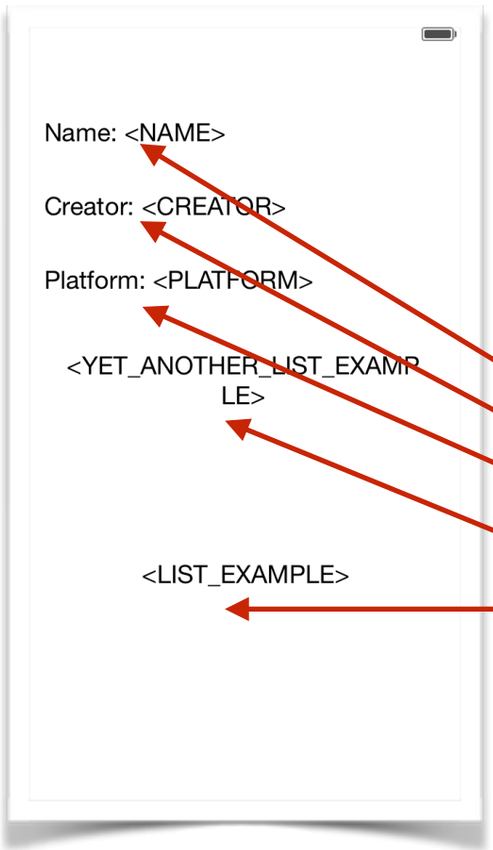


- Primeiramente, vamos gerar um arquivo YAML para podemos popular nossa interface com os dados obtidos a partir dele

- Usaremos o exemplo já apresentado, e o denominaremos de Data.yml, para saber que se trata de um arquivo YAML

```yaml
# Data.yml
# Created by Kaê Angeli Coutinho
---
Name: YAML Tutorial
Creator: Kaê Angeli Coutinho
Platform: iOS 8.0
ListExample:
    - 1
    - 2
    - 3
    - 4
    - 5
YetAnotherListExample: [Both,Literal,And,Numerical,Types,Are,Accepted]
```

- Agora, vamos começar o desenvolvimento da aplicação pela interface gráfica e suas conexões com o ViewController.h




```
//
//  ViewController.h
//  YAMLTutorial
//
//  Created by Kaê Coutinho on 9/27/14.
//  Copyright (c) 2014 Kaê Angeli Coutinho. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <YAML-Framework/YAMLSerialization.h>
#define YAML_DATA_FILE_PATH [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"yaml"]

@interface ViewController : UIViewController

@property (weak,nonatomic) IBOutlet UILabel * nameLabel;
@property (weak,nonatomic) IBOutlet UILabel * creatorLabel;
@property (weak,nonatomic) IBOutlet UILabel * platformLabel;
@property (weak,nonatomic) IBOutlet UITextView * listExampleTextField;
@property (weak,nonatomic) IBOutlet UITextView * yetAnotherListExampleTextField;

@end
```

- IBOutlet

- Declarar os métodos privados e iVars necessários

```
@interface ViewController()
{
    id YAMLData;
}

-(void)initialize;
-(void)parseYAMLDataFile;

@end

@implementation ViewController

@synthesize nameLabel;
@synthesize creatorLabel;
@synthesize platformLabel;
@synthesize listExampleTextField;
@synthesize yetAnotherListExampleTextField;
```

- Vamos implementar os métodos privados declarados anteriormente

```objc
-(void)initialize
{
    [self parseYAMLDataFile];
    if(YAMLData)
    {
        [nameLabel setText:[NSString stringWithFormat:@"Name: %@",YAMLData[@"Name"]]];
        [creatorLabel setText:[NSString stringWithFormat:@"Creator: %@",YAMLData[@"Creator"]]];
        [platformLabel setText:[NSString stringWithFormat:@"Platform: %@",YAMLData[@"Platform"]]];
        [listExampleTextField setText:@"List Example:\n"];
        [yetAnotherListExampleTextField setText:@"Yet Another List Example:\n"];
        for(id content in YAMLData[@"ListExample"])
        {
            [listExampleTextField setText:[NSString stringWithFormat:@"%@\n%@",[listExampleTextField text],content]];
        }
        for(id content in YAMLData[@"YetAnotherListExample"])
        {
            [yetAnotherListExampleTextField setText:[NSString stringWithFormat:@"%@\n%@",[yetAnotherListExampleTextField text],content]];
        }
    }
}
```

```objc
-(void)parseYAMLDataFile
{
    NSError * parseError;
    NSInputStream * YAMLDataInputStream = [NSInputStream inputStreamWithFileAtPath:YAML_DATA_FILE_PATH];
    YAMLData = [[YAMLSerialization objectsWithYAMLStream:YAMLDataInputStream options:kYAMLReadOptionStringScalars error:&parseError] firstObject];
    if(parseError)
    {
        NSLog(@"\n Impossible to read YAML file. Reason:\n\n %@",[parseError description]);
        YAMLData = nil;
    }
}
```

- Ultima alteração a se fazer é sobrescrever o método de ciclo da View Controller, viewDidLoad

```
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self initialize];
}
```

Concluindo

- Neste tutorial, você aprendeu alguns conceitos importantes, como:
  - Serialização e desserialização de dados
  - Padrão YAML
  - Teoria e aplicação de CocoaPods