



Desenvolvimento para iPhone

Usando Objective-C e iOS
SDK 8

Agenda

- Core Data
 - Quando usar e Como usar;
- Entidades;
- Atributos;
- Relacionamentos;
- CRUD
- Integração com UITableView;
- Prática
- Se der tempo...
 - UIManagedDocument e iCloud (requer iDevice)

Objetivo do Dia

- Dominar o Core Data.

Banco de Dados

- Em algum momento, você vai precisar guardar uma quantidade maior de dados e/ou precisar buscar dados de uma forma mais sofisticada;
- Essa é a hora de adotar um banco de dados local!;
- Os arquivos PLIST são bons para uma pequena massa de dados e não são têm capacidade de indexação. Portanto, **os arquivos PLIST não são adequados** para armazenar dados transacionais e complexos inerentes ao app.

Banco de Dados (Core Data)

- A abordagem padrão para banco de dados local na plataforma iOS é o Core Data;
- Um framework extremamente poderoso que fornece um banco de dados orientado à objetos!

Core Data

- Oferece um meio para criar sua árvore de objetos suportada por um banco de dados;
- Na frieza dos bits, os dados podem ser armazenados em SQLite, XML ou mesmo em memória, mas, na maioria dos casos, isso é transparente para o programador.

Como funciona o Core Data?

- Em quatro passos básicos:
 1. Criar um mapeamento visual com o Xcode, onde a relação entre entidades de objetos é definida;
 2. Criar objetos (registros) e queries através da API orientada à objetos;
 3. Acessar as “colunas” da “tabela” usando as **@property** do objeto (registro);
 4. Ser feliz!

Core Data

- Na prática, vamos focar em criar entidades, atributos e relacionamentos;
- Para acessar todas essas coisas maravilhosas, precisamos de um contexto: o **NSManagedObjectContext**;
 - Ele é o centralizador de todas as interações com o Core Data.



Core Data

- Como obter o contexto?
- Duas formas:
 1. Criando um **UIManagedDocument** e acessando a **@property managedObjectContext**;
 2. Ao criar um novo projeto, marcar a opção “Use Core Data”
- Na prática de hoje, veremos como fazer a primeira opção.

UIManagedDocument

- Herda de **UIDocument**, que oferece uma série de mecanismos para persistência;
- Quando você usa o **UIManagedDocument**, você tem meio caminho andado para integrar com o **iCloud**;
- Entenda-o como sendo um simples contêiner para seu banco de dados Core Data.

```
NSFileManager *fileManager = [NSFileManager defaultManager];

//A "pastaDocumentos" representa o caminho, em disco, da raiz do sandbox do app
NSURL *pastaDocumentos = [[fileManager URLsForDirectory:NSDocumentDirectory
                                                         inDomains:NSUserDomainMask] firstObject];

NSString *nomeDoArquivo = @"MeuBancoDeDados";

NSURL *caminhoDoArquivo = [pastaDocumentos URLByAppendingPathComponent:nomeDoArquivo];

//Cria a instância do objeto, mas não abre ou cria o arquivo referenciado.
UIManagedDocument *arquivoBD = [[UIManagedDocument alloc] initWithFileURL:caminhoDoArquivo];

BOOL arquivoExiste = [fileManager fileExistsAtPath:[caminhoDoArquivo path]];

if (arquivoExiste) {
    [arquivoBD openWithCompletionHandler:^(BOOL success) {
        if (success) {
            NSLog(@"Arquivo aberto com sucesso! Banco de dados existe!");
        }
    }];
} else {
    [arquivoBD saveToURL:caminhoDoArquivo
                  forSaveOperation:UIDocumentSaveForCreating
                  completionHandler:^(BOOL success) {
        if (success) {
            NSLog(@"Arquivo não existia. Arquivo criado com sucesso!");
        }
    }];
}
```

UIManagedDocument

- Salvam a si mesmos sozinhos!
- Se, por ventura, você quiser salvar manualmente, a operação será assíncrona;
- Eles se fecham sozinhos, quando não existir mais nenhum ponteiro **strong**!
- Se, por ventura, você quiser fechar manualmente, a operação será assíncrona.

```
//Salva o documento de forma assíncrona
[arquivoBD saveToURL:arquivoBD.fileURL
  forSaveOperation:UIDocumentSaveForOverwriting
  completionHandler:^(BOOL success) {
    if (success) {
      NSLog(@"Salvou com sucesso!");
    }
  }];
```

```
//fecha o documento de forma assíncrona
[arquivoBD closeWithCompletionHandler:^(BOOL success) {
  if (success) {
    NSLog(@"Fechou com sucesso!");
  }
}];
```

UIManagedDocument

- Cuidado com múltiplas instâncias de **UIManagedDocument** que apontam para o mesmo arquivo!
- Elas não vão compartilhar o mesmo contexto;
- Mudanças em uma instância não serão refletidas automaticamente em outra;
- Eventuais conflitos terão que ser tratados individualmente pelo programador.

Core Data

- Agora que você já tem um **NSManagedObjectContext**... O que fazer?
- Definir um modelo de dados, graficamente através do Xcode; e
- Agora você está pronto para fazer todas as operações do CRUD!

Prática

- Criar um modelo de dados (xcdatamodel);
- Criar o contexto e fazer todas as operações do CRUD;
- Integrar com o UITableViewController usando o NSFetchedResultsController

Desafio

- Recuperar do Flickr (esta parte será fornecida pelo professor, já que ainda não vimos a parte de chamada a webservices) uma lista de fotos recentes;
- Armazenar, usando Core Data com duas entidades: Foto e Fotógrafo;
- Num UITableView, mostrar uma lista de fotógrafos. Ao clicar em um, mostrar as fotos em um UICollectionView.
 - Em ambos, usar o NSFetchedResultsController