Persistência

Cada problema exige uma solução

- `NSUserDefaults`
- Keychain
- `NSCoding` + `NSKeyedArchiver`
- Plist / JSON
- SQLite
- Core Data
- Realm
- YapDatabase

NSUserDefaults

❗ No fim, salva em um Plist

😨 Não é seguro

💩 Carrega todo o conteúdo para memória

😉 Bom para poucos dados (preferências/configurações, etc)

```
let defaults = NSUserDefaults.standardUserDefaults()

userDefaults.setObject("Palmeiras", forKey: "time")
userDefaults.synchronize()

let palmeiras = userDefaults.objectForKey("time")
```

```
pod 'SwiftyUserDefaults'
```

https://github.com/radex/SwiftyUserDefaults

Keychain

🔐 Área mais segura do iOS

😱 API em C

😎 Vários wrappers disponíveis

```
pod 'Locksmith'
```

https://github.com/matthewpalmer/Locksmith

	Core Data	NSKeyedArchiver
Entity Modeling	Yes	No
Querying	Yes	No
Speed	Fast	Slow
Serialization Format	SQLite, XML, or NSData	NSData
Migrations	Automatic	Manual
Undo Manager	Automatic	Manual

	Core Data	NSKeyedArchiver
Persists State	Yes	Yes
Pain in the Ass	Yes	No

http://nshipster.com/nscoding/

Plist / JSON

- 🙊 Mesmas características do `NSCoding`
- 😭 Serialização fica a cargo do desenvolvedor
- 💖 Mantle (https://github.com/Mantle/Mantle)

SQLite

- 👊 Todo mundo conhece
- 😱 Biblioteca em C
- 😜 Cross-platform
- 😎 Vários wrappers disponíveis
- 😬 Criação de tabelas, selects, inserts, etc na mão

Core Data

👀 Grafo de objetos

😄 Pode ser persistindo em um arquivo SQLite

😉 Mantido pela Apple

💩 Difícil de usar

Realm

😍 Bem mais simples de usar

😜 Cross-platform

❗ Muito recente

💔 Faltam algumas features importantes (optionals, notificações de mudança granulares, etc)

YapDatabase

- 😏 Feito em cima do SQLite
- 💭 Chave-valor

https://github.com/yapstudios/YapDatabase

https://medium.com/the-way-north/ditching-core-data-865c1bb5564c

Isolar persistência

```
class FavoritesManager {
    var favoritesIdentifiers: Set<Int> {
        // ??
    }

    func addIdentifier(identifier: Int) {
        // ??
    }

    func removeIdentifier(identifier: Int) {
        // ??
    }
}
```

Tente implementar usando `NSUserDefaults`!

Lembre-se que só precisamos guardar os ids

E se quisermos atualizar a listagem sempre que os favoritos mudarem?

Delegate

Bloco

Notificação

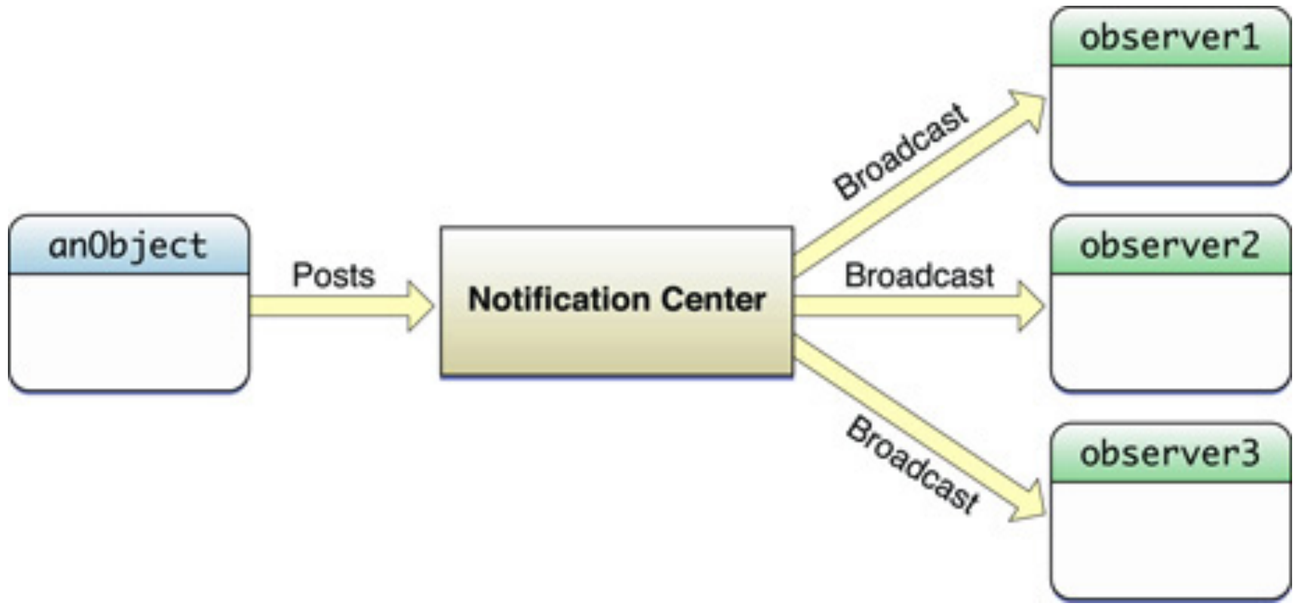
~~Delegate~~

~~Bloco~~

Notificação

NSNotificationCenter

- Alguém posta uma notificação
- Os interessados nessa notificação são avisados



Registrar

```
let name = FavoritesManager.favoritesChangedNotificationName
let notificationCenter = NSNotificationCenter.defaultCenter()

notificationCenter.addObserver(self, selector: "favoritesChanged",
    name: name, object: nil)
```

`object` é quem gerou a notificação.

Passar `nil` indica que queremos ouvir independentemente do gerador.

Desregistrar

- É importante se desregistrar antes do objeto ser desalocado

```
deinit {
    let name = FavoritesManager.favoritesChangedNotificationName
    let notificationCenter = NSNotificationCenter.defaultCenter()
    notificationCenter.removeObserver(self, name: name, object: nil)
}
```

Postar notificação

```
let name = self.dynamicType.favoritesChangedNotificationName
let notificationCenter = NSNotificationCenter.defaultCenter()

notificationCenter.postNotificationName(name, object: self)
```

É possível passar um `userInfo` que pode ser acessado no método que escuta a notificação