

# Core Data

# Core Data

- Core Data ?
- Création du modèle
- Utilisation

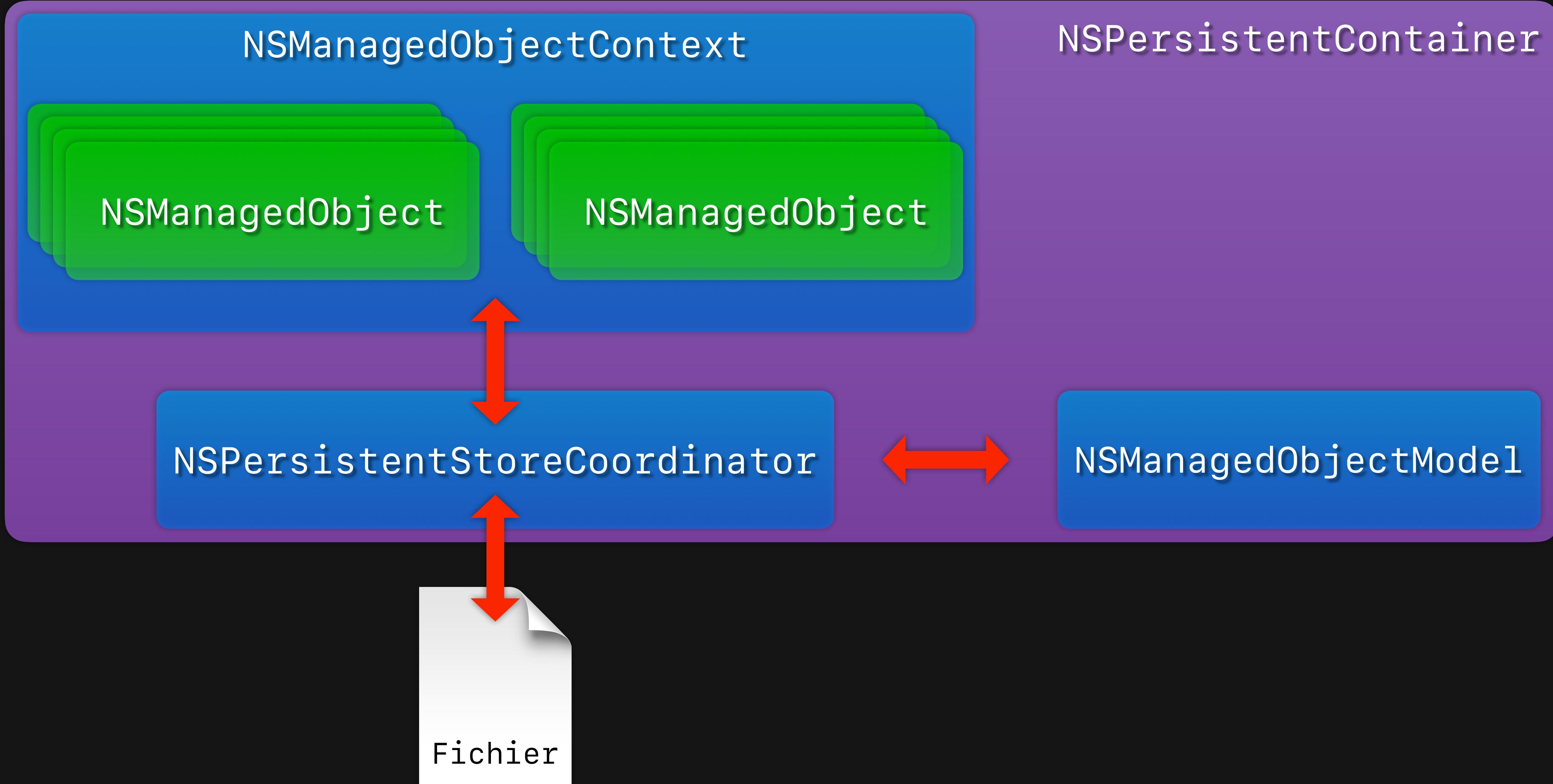
Core Data

Core Data ?

## Core Data ?

- ORM (object-relational mapping) disponible depuis iOS 3
- Gère la persistance d'un graphe d'objet
- Validation automatique des propriétés
- SQLite ou XML

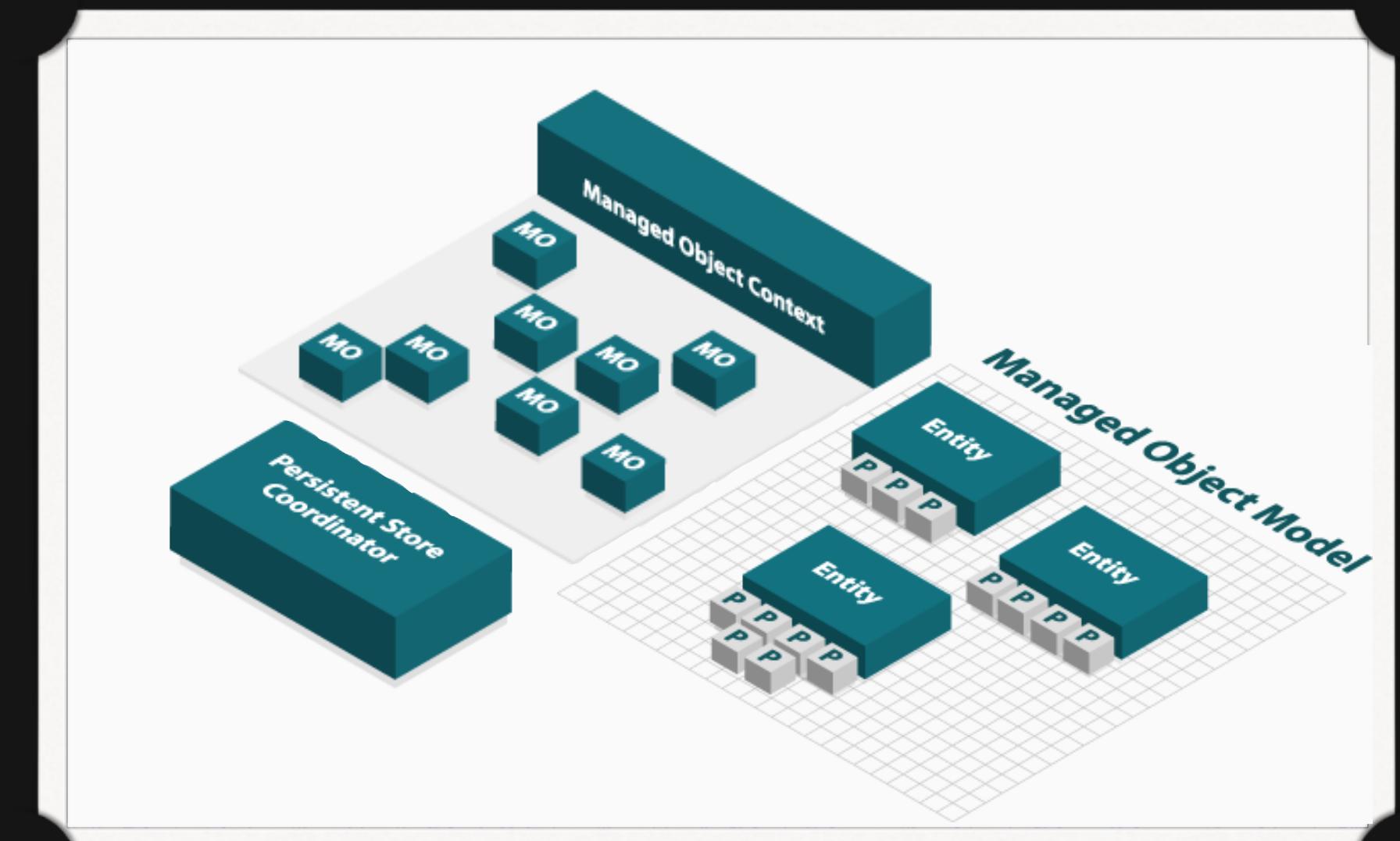
Core Data ?



Core Data ?

# NSPersistentContainer

- Introduit dans iOS 10 / macOS Sierra
- Simplifie la gestion de la stack Core Data
  - S'occupe de l'initialisation des différents éléments qui composent Core Data



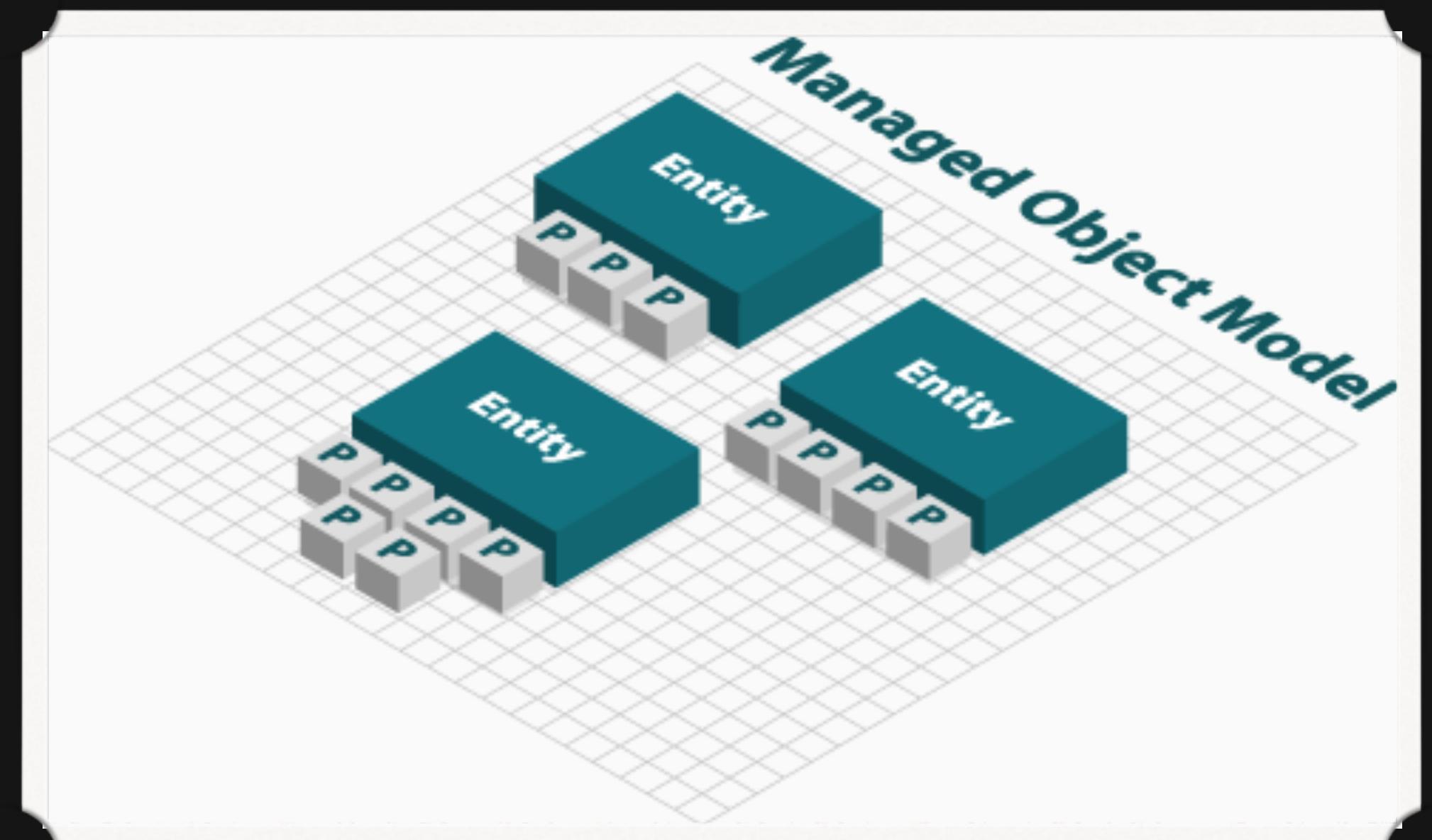
# NSManagedObjectContext

- Chef d'orchestre de Core Data dans notre application
- Surveille tout les objets sous sa responsabilité
- Peut transmettre les objets pour sauvegarde sur disque
- Gère le “Undo-Redo” et le rollback
- Récupère les données par requêtes



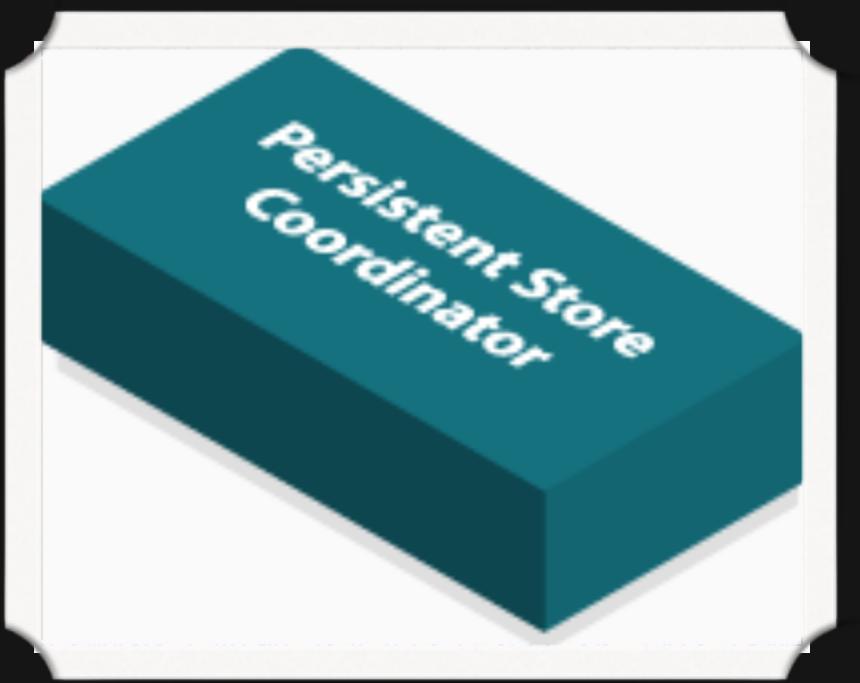
# NSManagedObjectModel

- Chargé depuis un fichier .xcdatamodel
- Contient la description des données
- Défini des entités qui possèdent des propriétés
  - Attributs
  - Relations
  - Requêtes



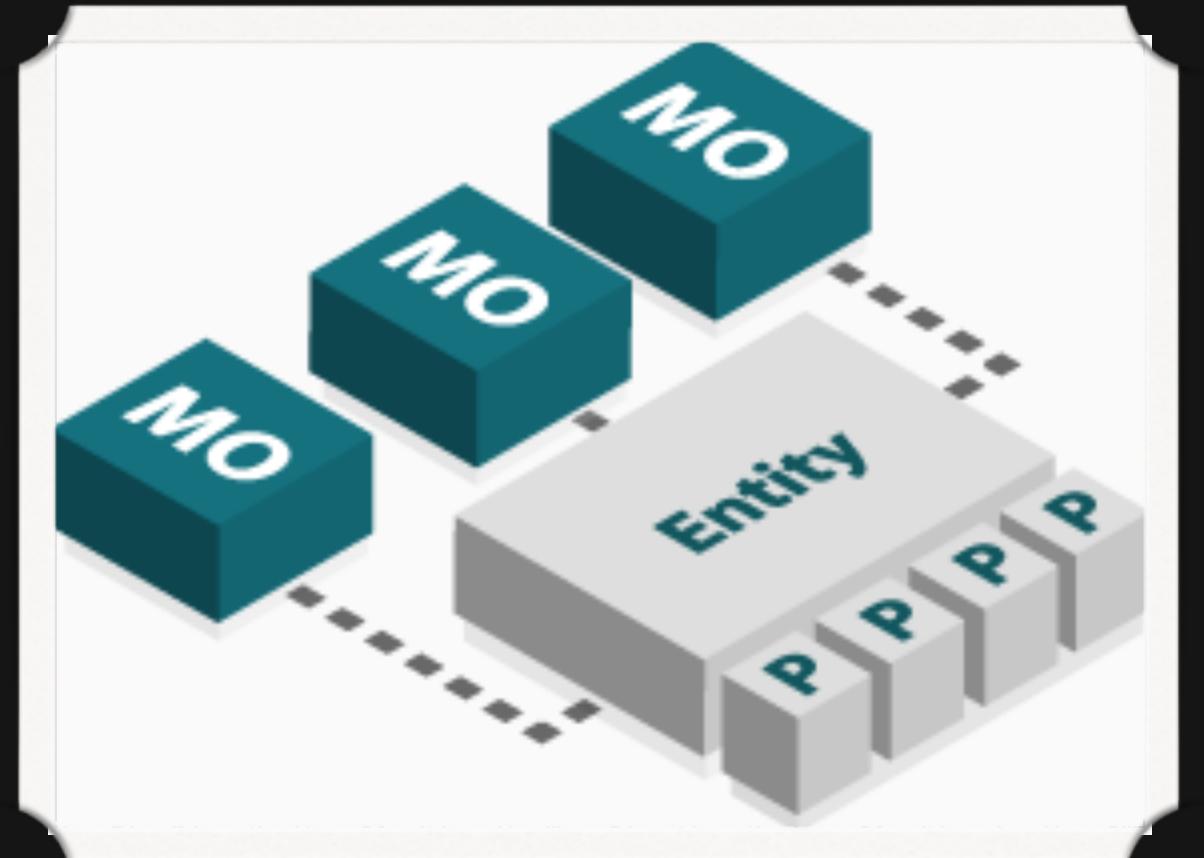
# NSPersistentStoreCoordinator

- Gère la persistance des données sur le disque
- Écrit les données dans des fichiers (SQLite par défaut)
- Utilisé directement uniquement dans des utilisations avancées de Core Data

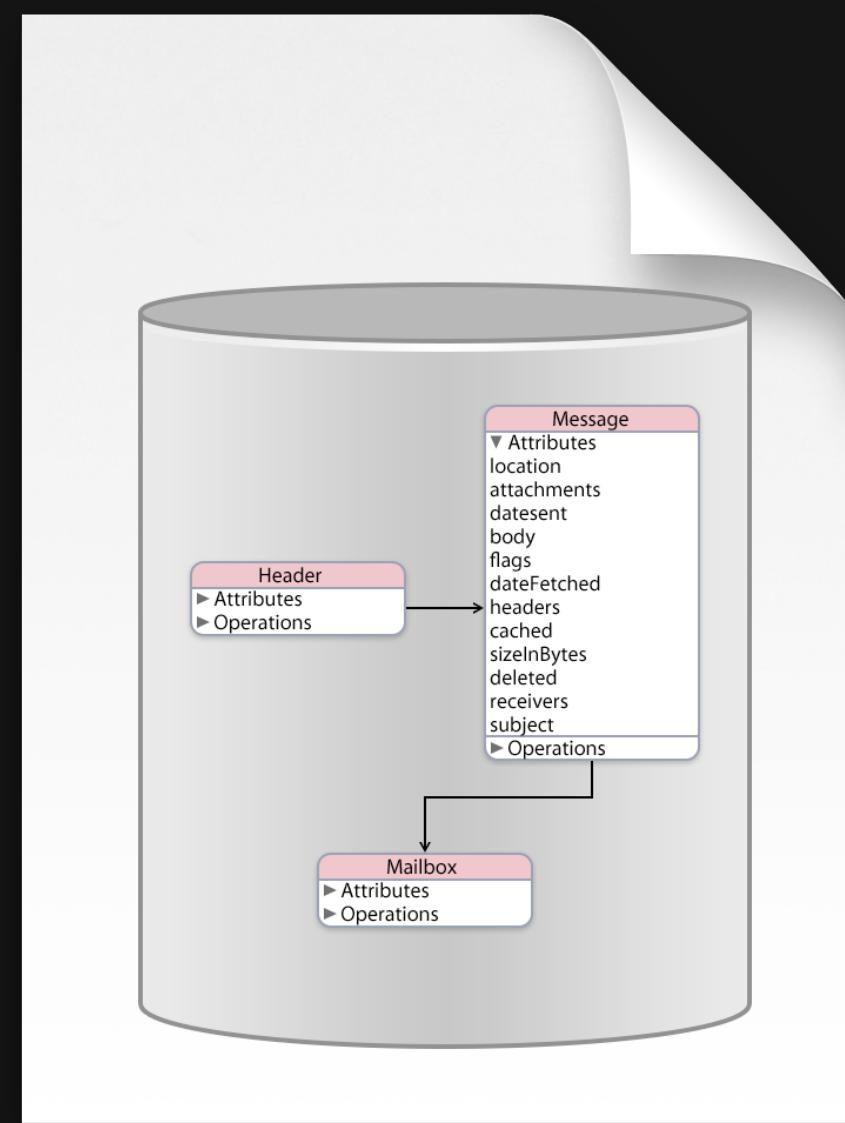


# NSManagedObject

- Représente un objet dans le contexte
- Chaque NSManagedObject est lié à une entité du modèle
- Implémente les méthodes nécessaire à la gestion des données
- Chaque objet possède un identifiant unique
- À sous-classer pour créer vos propres objets managés

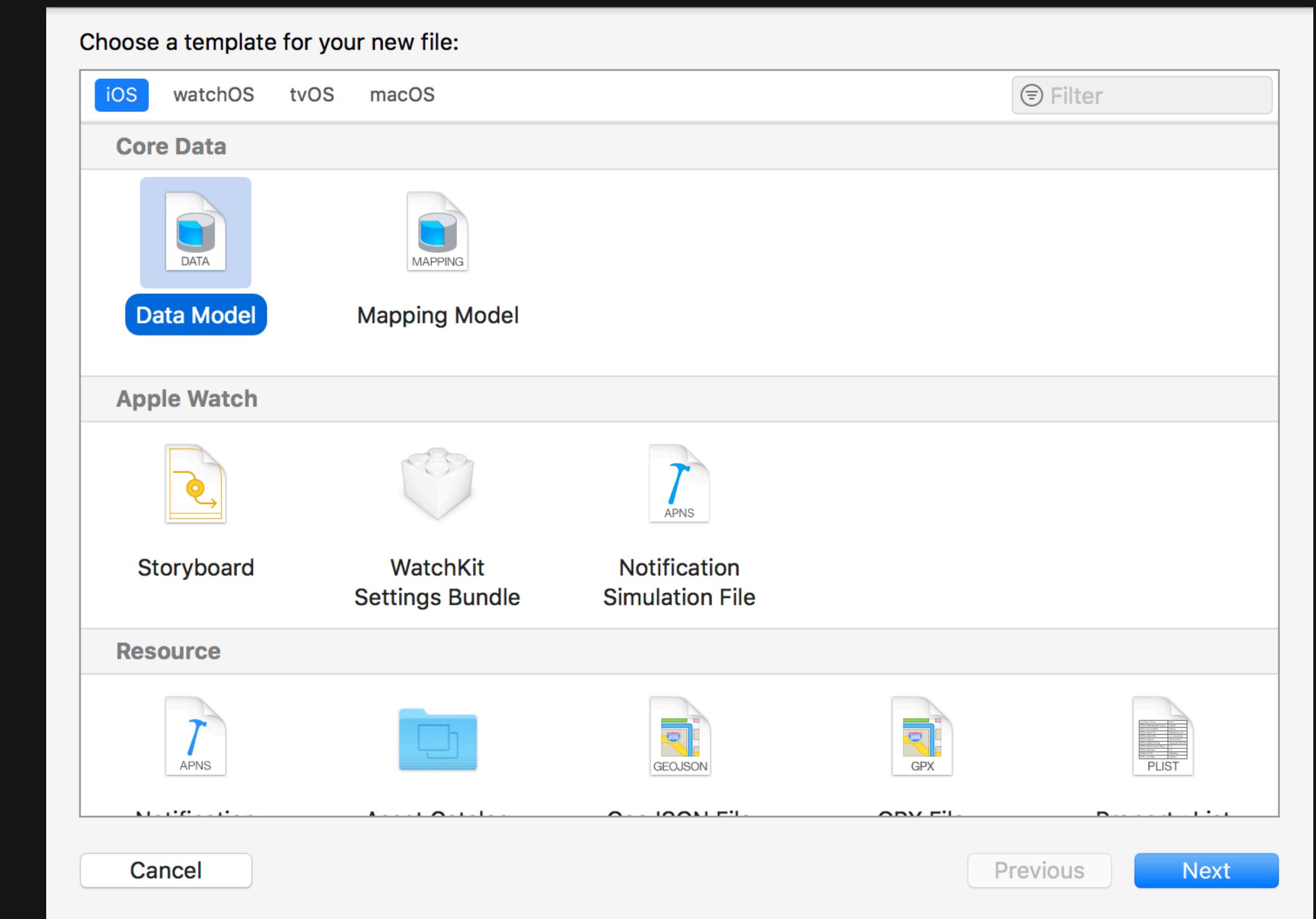


# Création du modèle



## Création du modèle

- Création graphique depuis Xcode
- Ressemble à de l'UML
- Définition de chaque entité
  - Attributs
  - Relations



# Création du modèle

The screenshot shows the Xcode Model Editor interface for creating a data model. The left sidebar lists entities: Author and Book (selected). The main area is divided into three sections: Attributes, Relationships, and Fetched Properties.

**Attributes Section:**

- Attributes Table:** Shows columns for Attribute, Type, and Value.
- bookNotation** is selected, highlighted in blue.
- Type:** Integer 16.
- Value:** bookNotation.

**Relationships Section:**

- Relationships Table:** Shows columns for Relationship, Destination, and Inverse.
- author** is selected.
- Destination:** Author.
- Inverse:** books.

**Fetched Properties Section:**

- Fetched Properties Table:** Shows columns for Fetched Property and Predicate.
- Empty table.

**Right Panel (Inspector):**

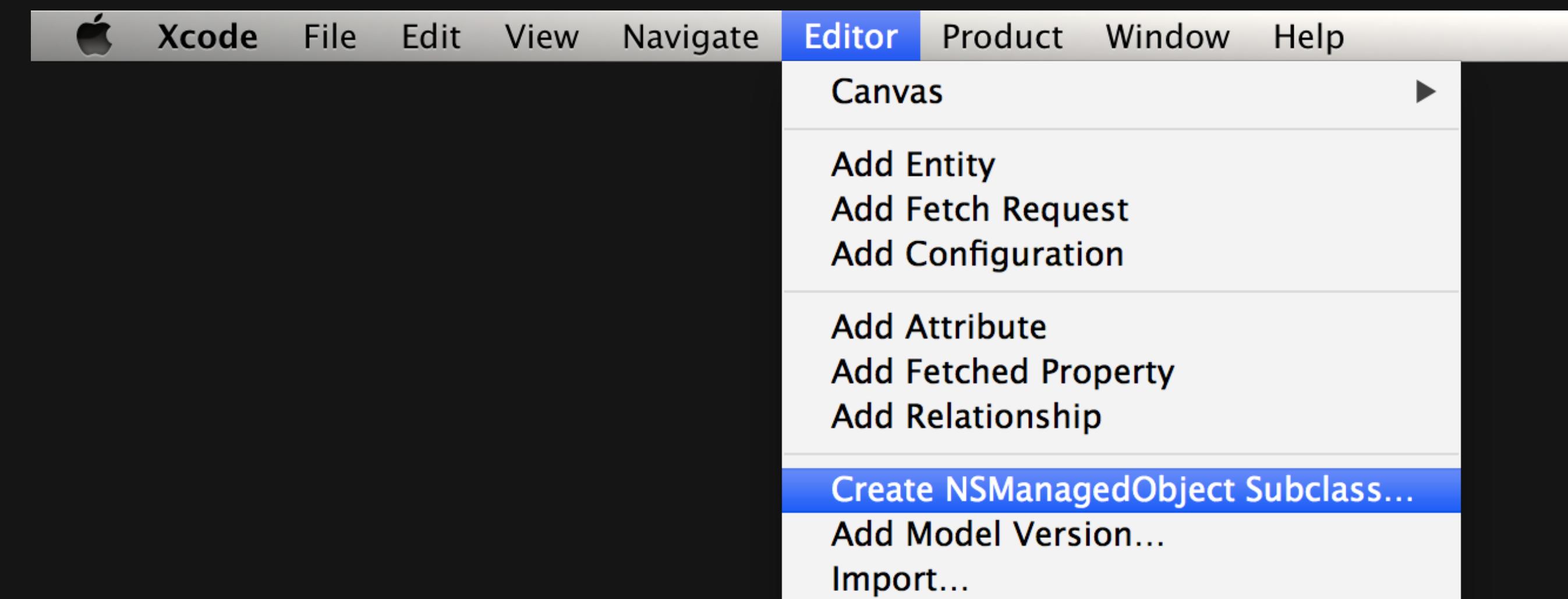
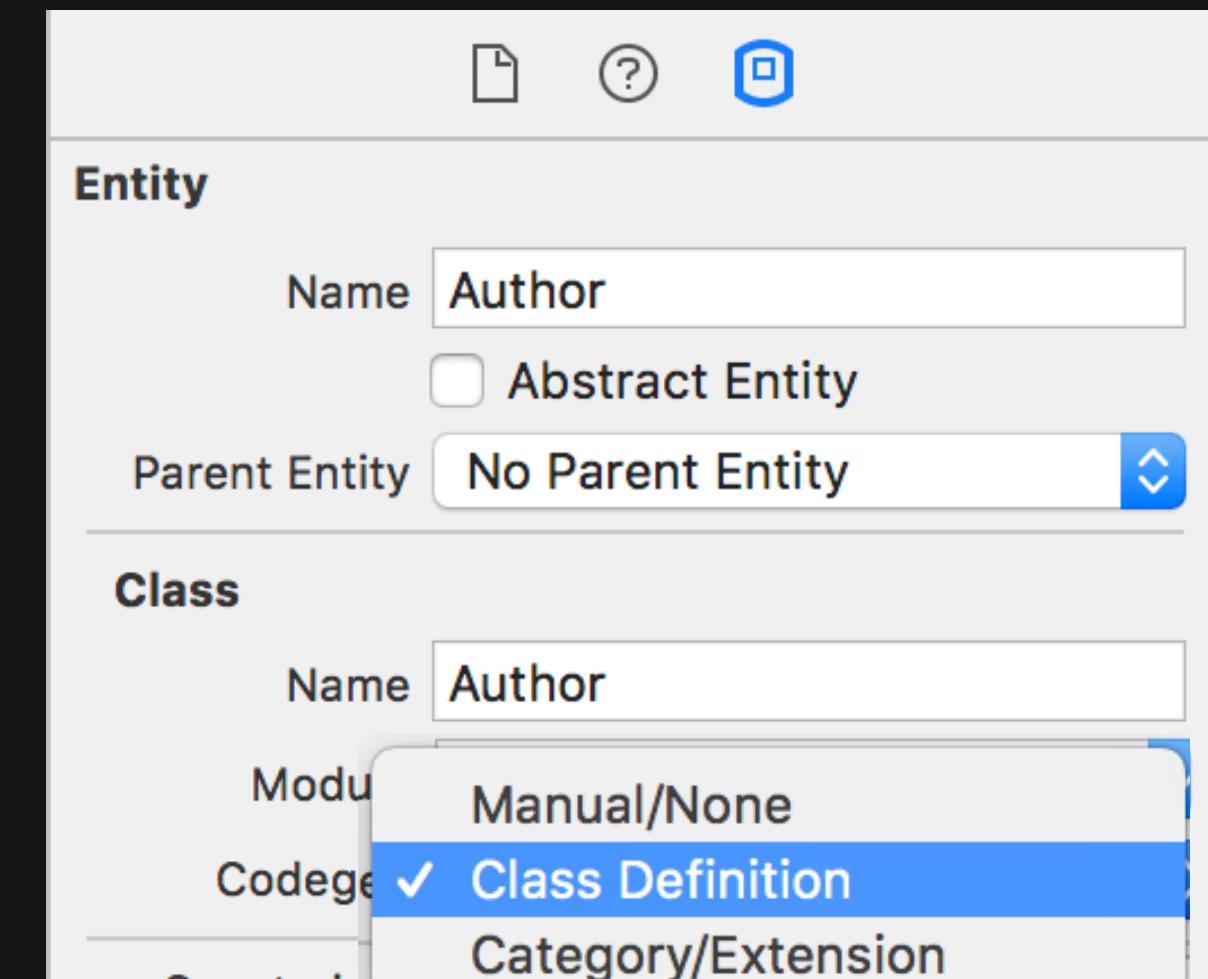
- Attribute Section:**
  - Name:** bookNotation.
  - Properties:** Transient (unchecked), Optional (unchecked), Indexed (unchecked).
  - Attribute Type:** Integer 16.
  - Validation:** Minimum (0), Maximum (5), Default (0).
  - Advanced:** Index in Spotlight (unchecked), Store in External Record File (unchecked).
- User Info Section:**
  - Key:** Value.
- Versioning Section:**
  - Hash Modifier:** Version Hash Modifier.
  - Renaming ID:** Renaming Identifier.

**Bottom Bar:**

- Outline Style
- Add Fetch Request
- + (Add Entity)
- Add Fetched Property
- Editor Style (highlighted with a blue circle)

## Création du modèle

- Classes de nos NSManagedObject générées automatiquement
- Plus de simplicité dans le code
- Autocomplétion des propriétés

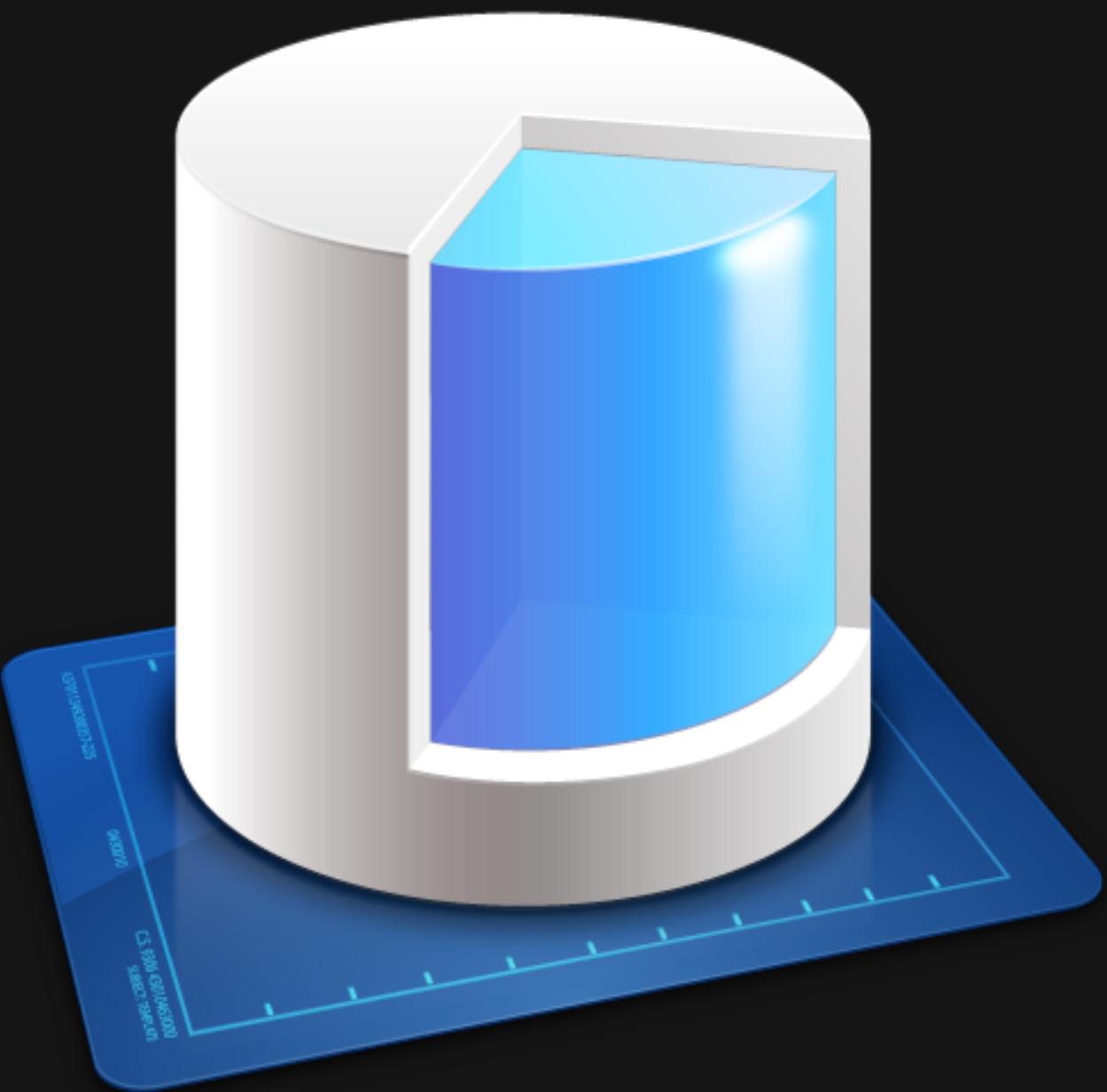


Création du modèle

Démo

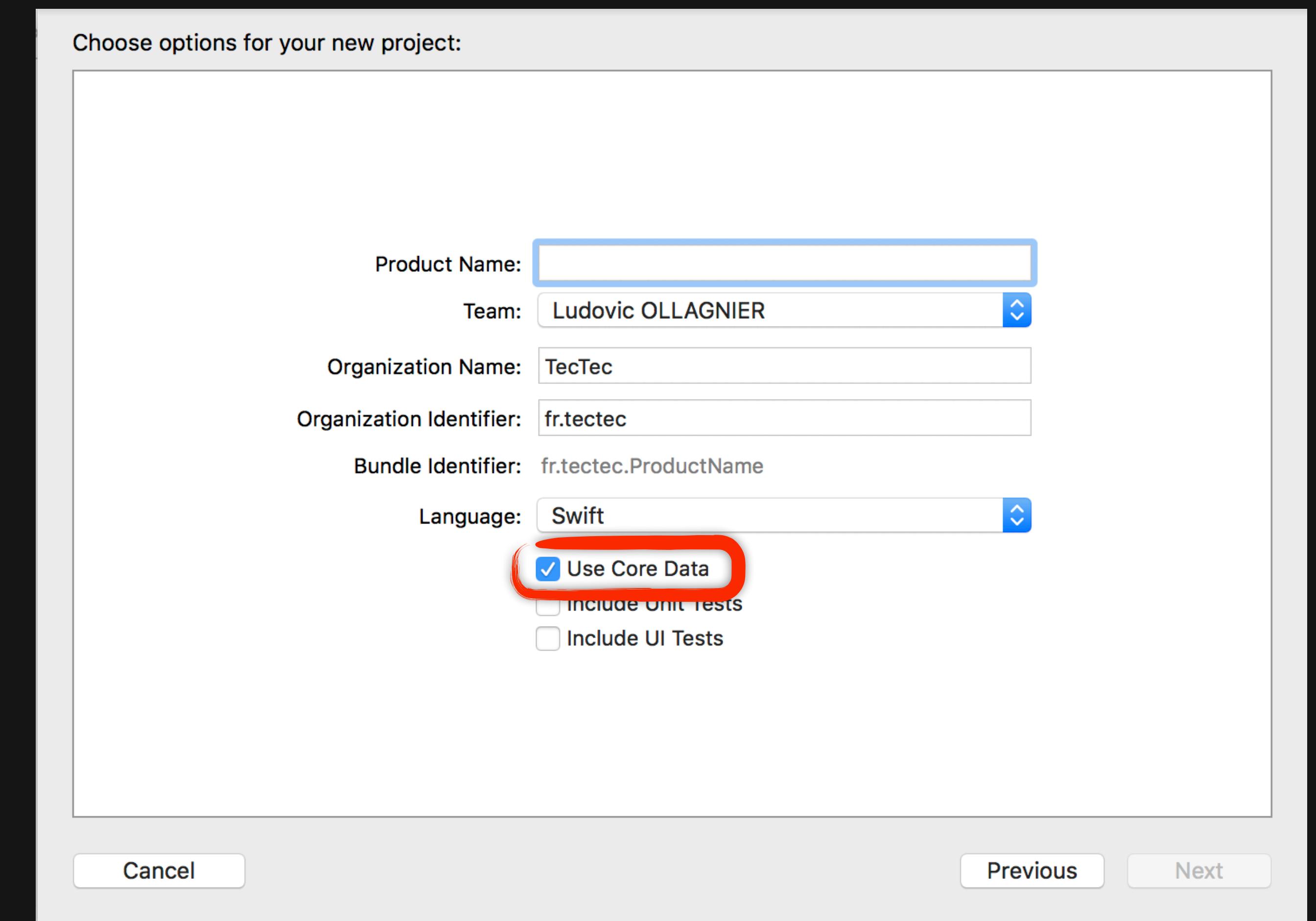
# Core Data

# Utilisation



- Chargement obligatoire de :
  - NSManagedObjectModel
  - NSPersistentStoreCoordinator
  - NSManagedObjectContext

→ Laissez faire Xcode



- Implémentation d'Apple
  - Mise en place dans le App Delegate
  - Présence d'une méthode -saveContext pour gérer l'échec de l'écriture

**→ Pas la meilleure implémentation !**

- L'App Delegate ne fait pas parti du modèle
- Le code de gestion de Core Data doit être dans le modèle
- Possibilité d'utiliser un singleton, ou de passer le modèle de View Controller en View Controller



# Ajout d'un objet

- Utiliser le constructeur de l'objet, en lui indiquant le contexte à utiliser
- Personnaliser les propriétés de l'objet
- Demander au modèle d'enregistrer le contexte

```
let aBook = Book(context: context)
aBook.name = "De la Terre à la Lune"
saveContext()
```



# Récupération d'objets

- Récupérer une NSFetchedRequest
- Demander au contexte de récupérer les objets avec la fetchRequest
- Récupérer un tableau des objets correspondants !

```
let fetchRequest: NSFetchedRequest<Book> = Book.fetchRequest()
guard let allBooks = try? context.fetch(fetchRequest) else { return }
```



# Récupération d'objets

- Une NSFetchedRequest peut être précisée par un NSPredicate
- Sans NSPredicate, tout les éléments sont retournés
- La fetchRequest est exécutée sur le contexte, pas sur le persistentStore

```
let fetchRequest: NSFetchedRequest<Book> = Book.fetchRequest()  
let predicate = NSPredicate(format: "name == \\"name")  
fetchRequest.predicate = predicate  
guard let allBooks = try? context.fetch(fetchRequest) else { return }
```

Pour aller plus loin . . .



- [Core Data Tutorial for iOS](#)
- [Core Data Programming Guide](#)