

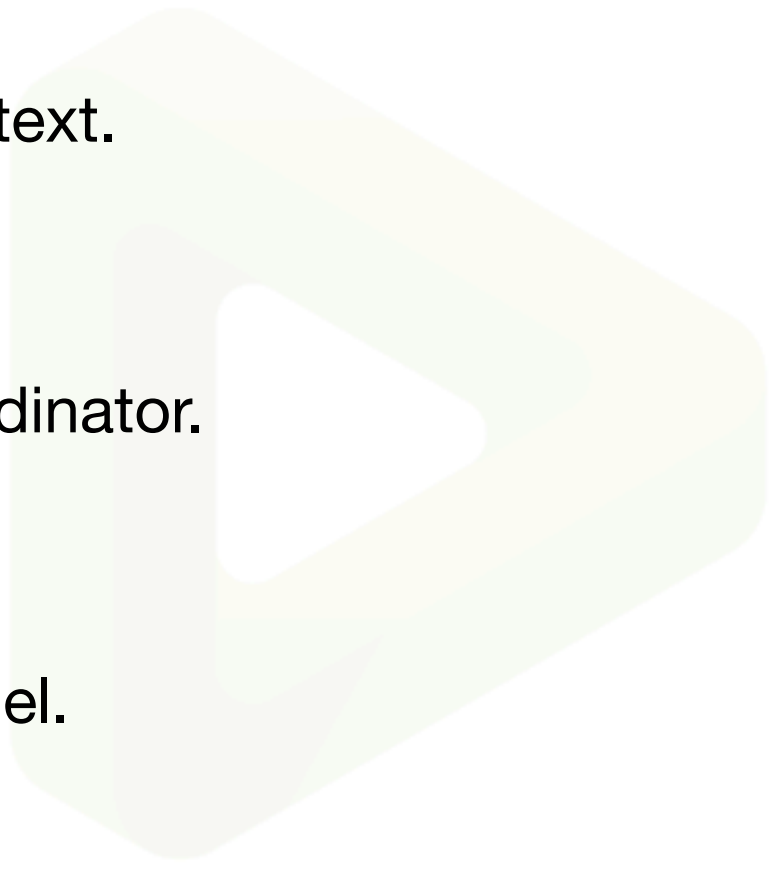
# Tema 11

## Introducción a Core Data

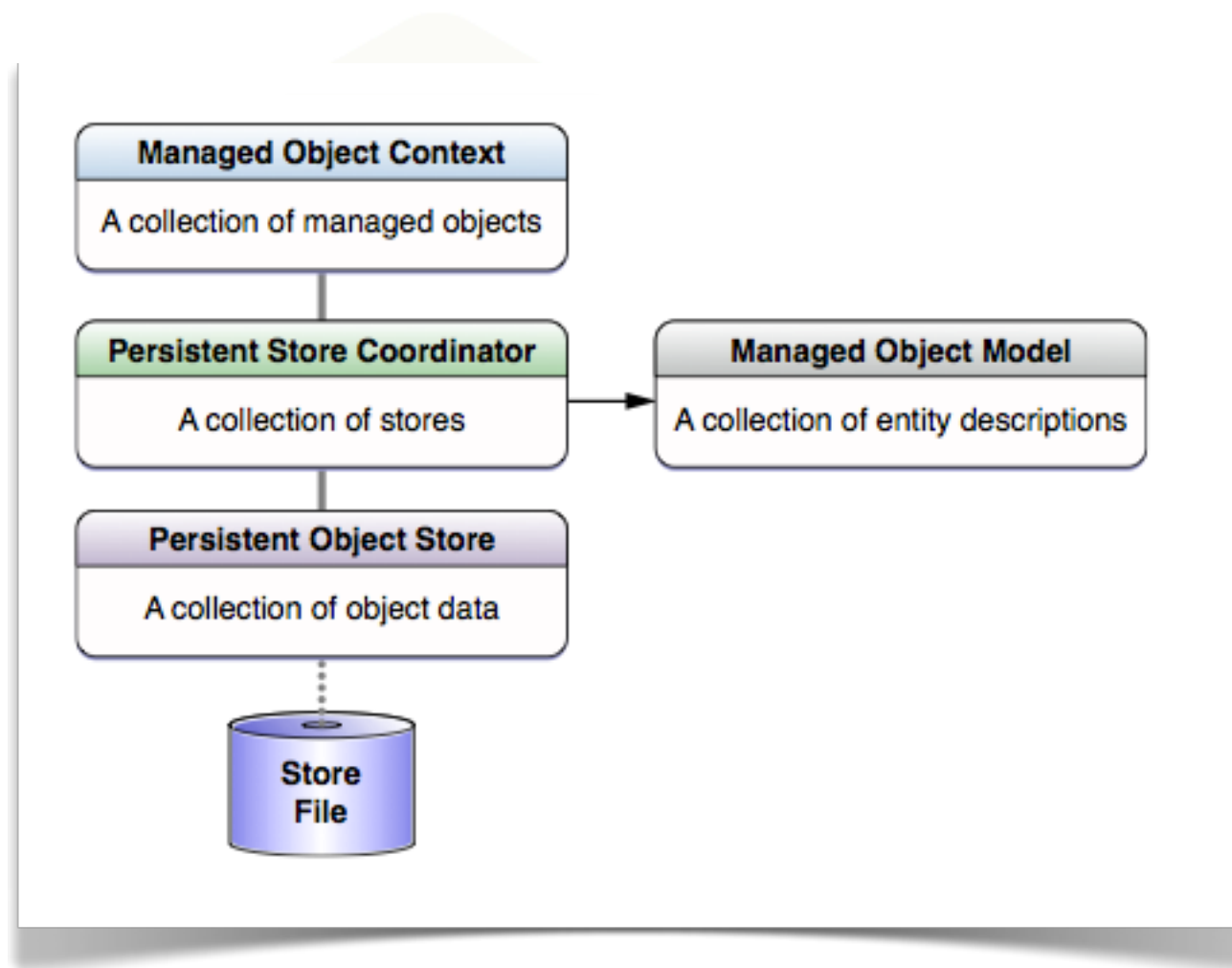
# Core Data Foundation

- Es un *framework* que garantiza la persistencia de los datos de nuestra aplicación.
- Nos abstrae de la implementación final del almacenamiento.
- Permite enlazar el modelo lógico con el modelo físico.

# Core Data Stack

- Managed Object Context.
  - Persistent Store Coordinator.
  - Managed Object Model.
  - Persistent Object Store.
- 

# Core Data Stack

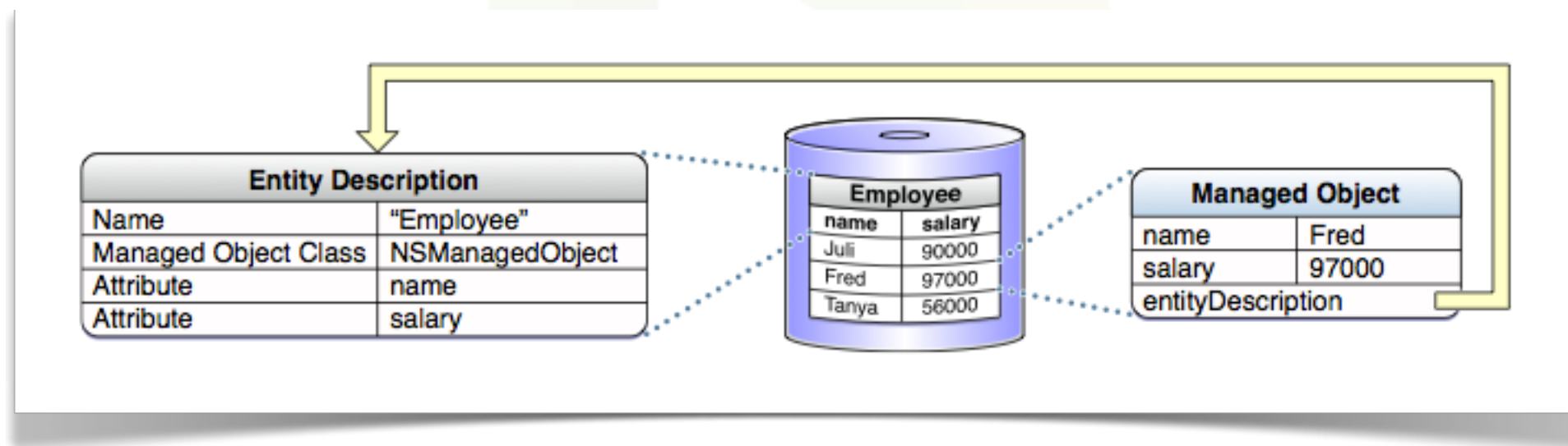


# Managed Object Context

- Es una instancia de **NSManagedObjectContext**.
- Un contexto representa el espacio de un único objeto.
- Es el responsable de gestionar una colección de objetos.

# Managed Object Model

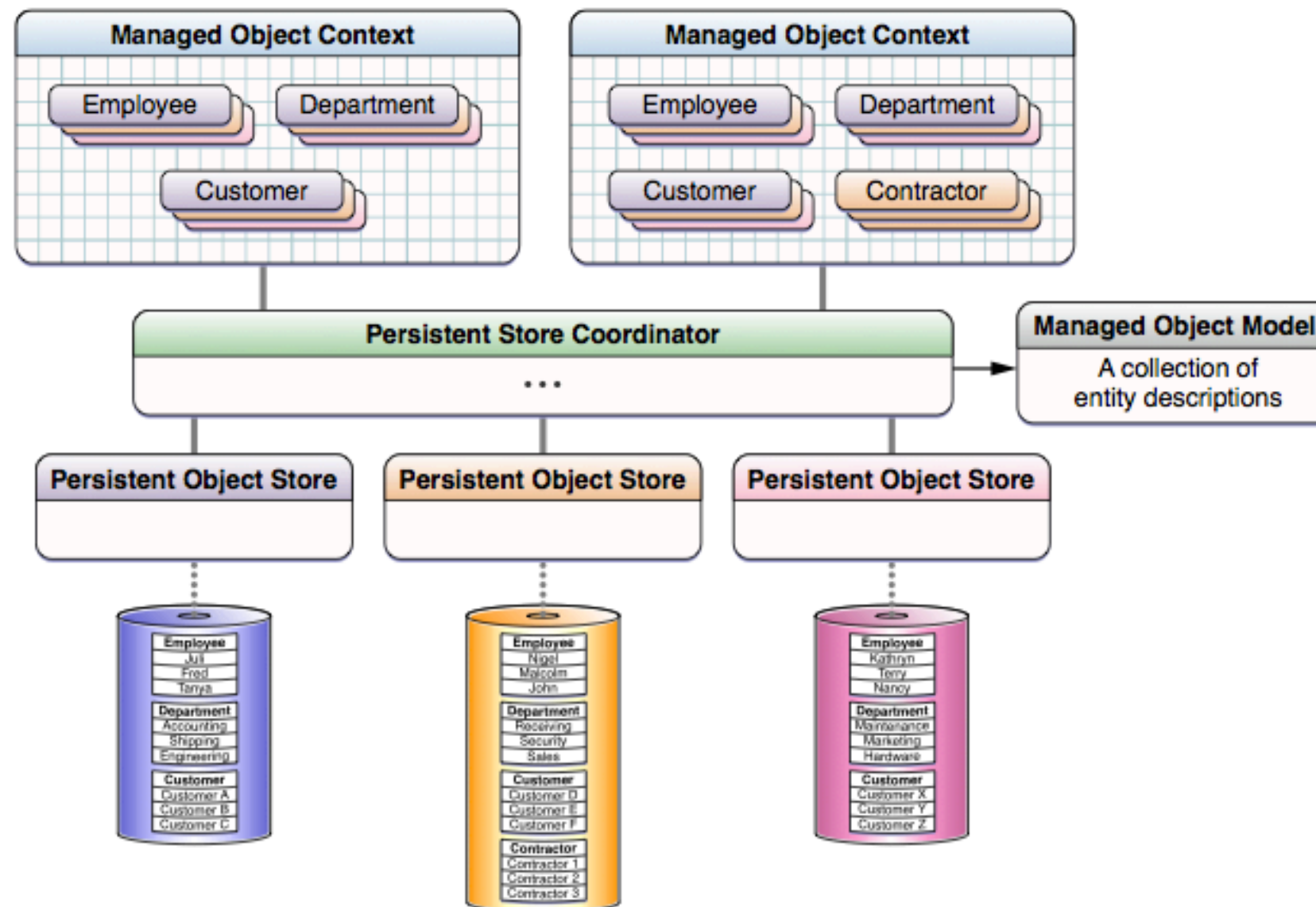
- Es la representación de un esquema que coincide con la base de datos y los '*managed objects*' de la aplicación.



# Persistent Store Coordinator

- Es una instancia de **NSPersistentStoreCoordinator**.
- Gestiona una colección de '*Persistent Object Stores*'. Es decir, un conjunto de ficheros de datos persistentes.
- Asocia los objetos de la aplicación con los registros de la BD.

# Core Data

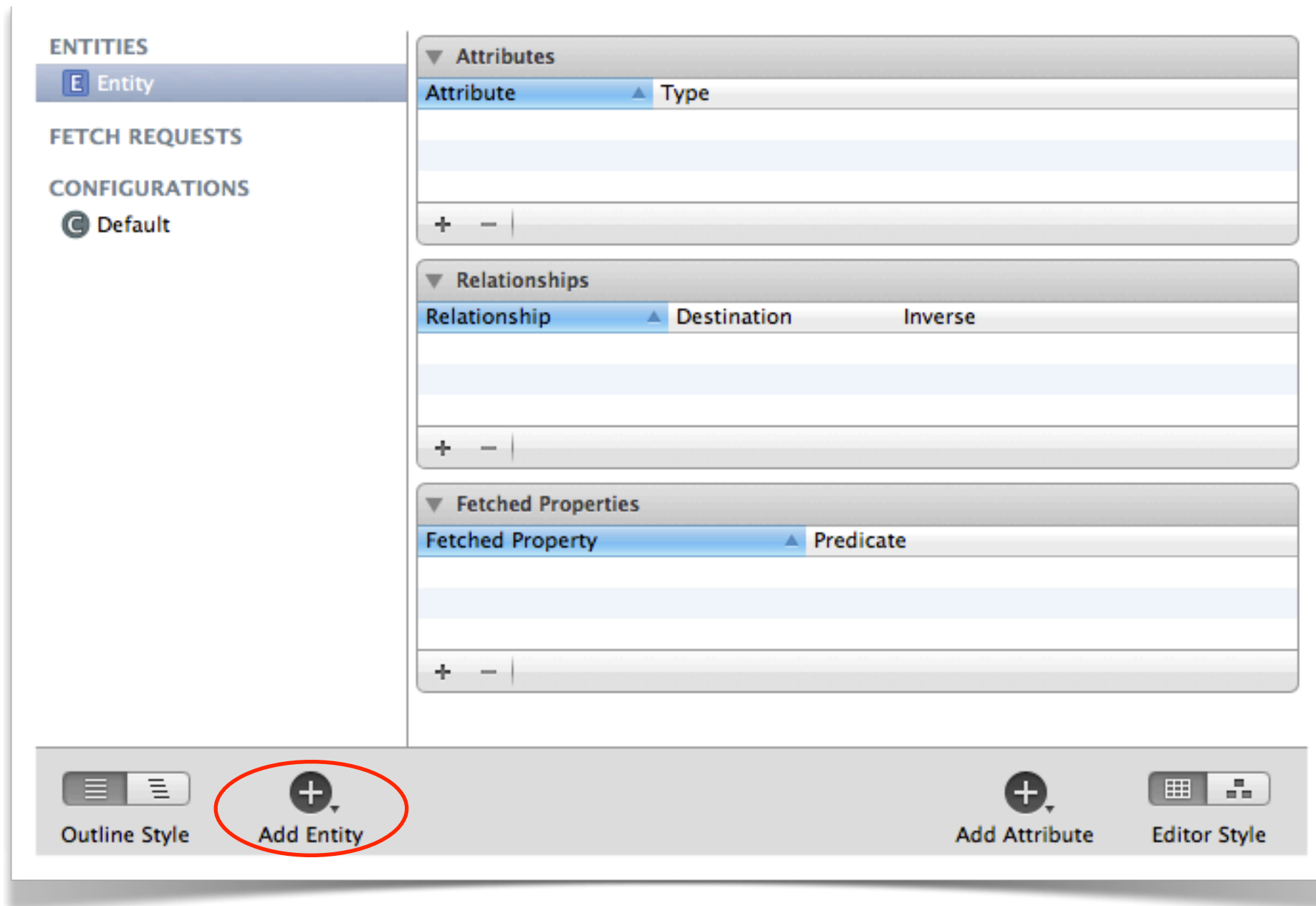




# Añadiendo Entidades

- Los modelos se encuentran en *.xcdatamodel*.
- Para añadir hacemos click en el botón **Add entity** de la barra de herramientas inferior.

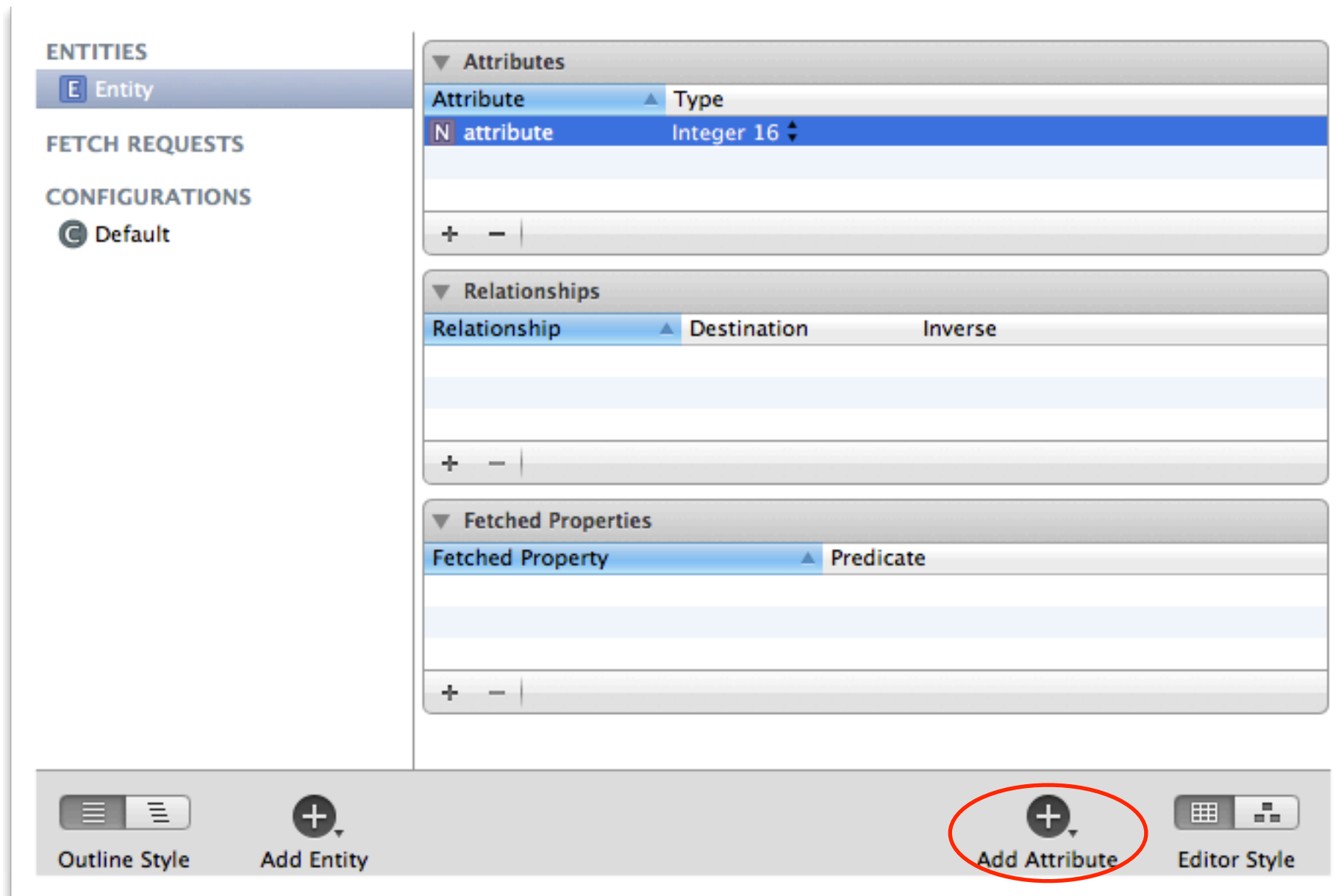
# Añadiendo Entidades



# Añadiendo Atributos

- Para añadir un atributo hacemos click en **Add attribute**.
- Se debe seleccionar el nombre y el tipo del atributo.

# Añadiendo Atributos



# Creando objetos

- Es necesario invocar a **NSEntityDescriptor** con el tipo de objeto y el contexto.

```
// Create and configure a new instance of the Event entity
Event *event = (Event *)[NSEntityDescription insertNewObjectForEntityForName:@"Event"
inManagedObjectContext:managedObjectContext];
CLLocationCoordinate2D coordinate = [location coordinate];
```

```
[event setLatitude:[NSNumber numberWithDouble:coordinate.latitude]];
[event setLongitude:[NSNumber numberWithDouble:coordinate.longitude]];
[event setCreationDate:[NSDate date]];
```

# Salvando objetos

- Se utiliza el método **save** de **NsManagedObjectContext**.
- En caso de fallo, encontraremos en la variable **NSError** el motivo y otros parámetros.

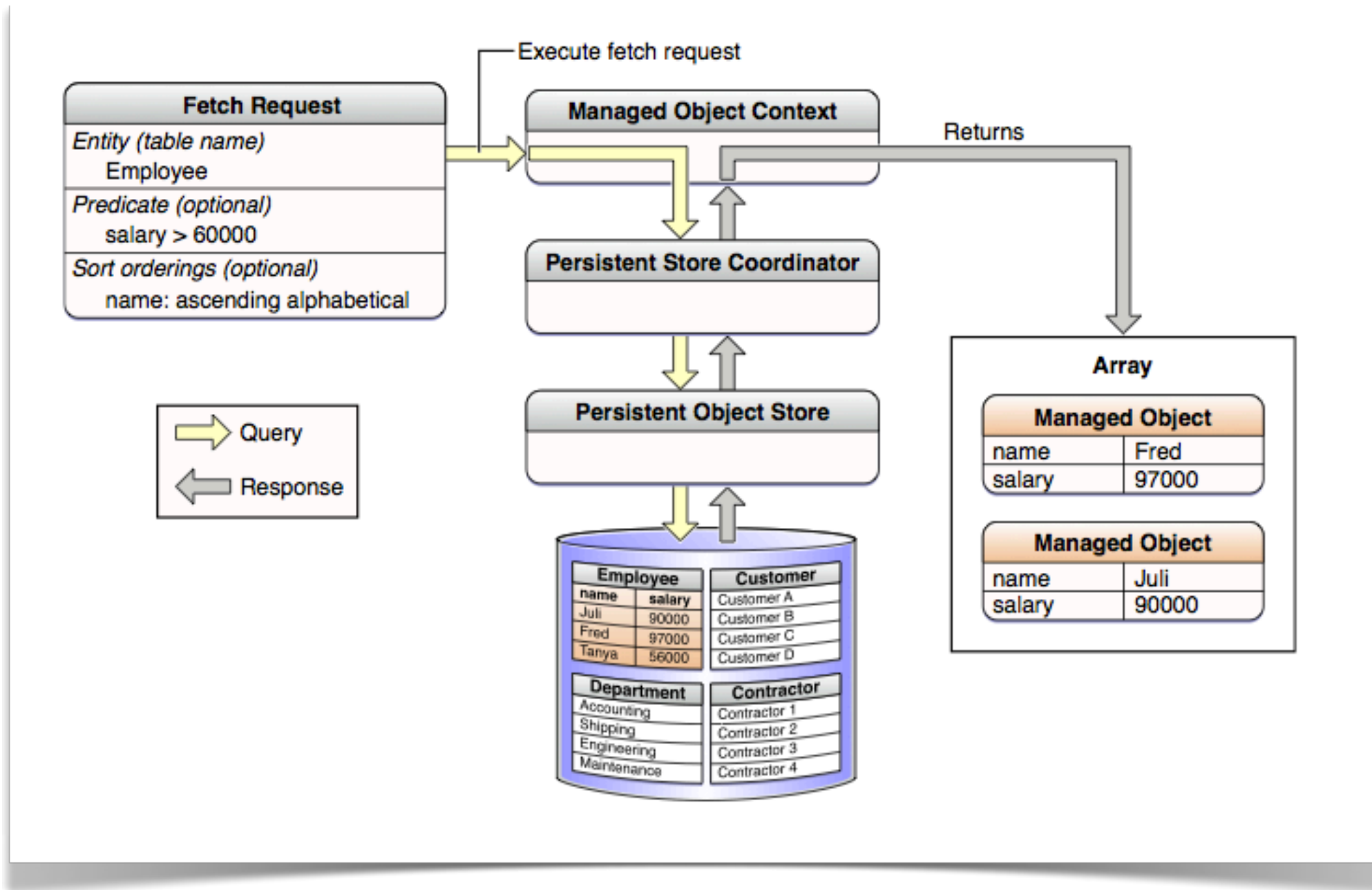
```
NSError *error;

if (![managedObjectContext save:&error]) {
    // Handle the error.
}
```

# Recuperando Objetos

- Para recuperar objetos de un sistema de persistencia, es necesario un **ManagedObjectContext** y una petición de obtención (**NSFetchRequest**).
- Se debe especificar la entidad de manera obligatoria y opcionalmente, una serie de parámetros.

# Esquema de Recuperación





# Creando la petición

- Se crea el objeto **NSFetchRequest**.
- Se crea el objeto de entidad.
- Se le asocia la entidad mediante **setEntity**.

```
NSFetchRequest *request = [[NSFetchRequest alloc] init];  
  
NSEntityDescription *entity = [NSEntityDescription entityForName:@"Event"  
inManagedObjectContext:managedObjectContext];  
[request setEntity:entity];
```

# Ordenación de registros

- Se crea el objeto de tipo **NSSortDescriptor**.
- Se inicia con el atributo por el que deseamos ordenar la colección e indicamos el modo (ascendente/descendente).

```
NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"creationDate"  
ascending:NO];
```

```
NSArray *sortDescriptors = [[NSArray alloc] initWithObjects:sortDescriptor, nil];
```

```
[request setSortDescriptors:sortDescriptors];  
[sortDescriptors release];  
[sortDescriptor release];
```

# Ejecutando la petición

- Se utiliza el contexto, indicando la petición (**request**).
- Devuelve un conjunto de resultados (en una variable **NSMutableArray**).

```
NSError *error;

NSMutableArray *mutableFetchResults =
[[managedObjectContext executeFetchRequest:request error:&error] mutableCopy];

if (mutableFetchResults == nil) {
    // Handle the error.
}
```

# Borrado de Objetos

- Se indica al contexto que objeto debe borrar.
- Posteriormente, se realiza el **commit**.

```
[managedObjectContext deleteObject:eventToDelete];  
NSError *error;  
if (![managedObjectContext save:&error]) {  
    // Handle the error.  
}
```

# Core Data Manager

- Vamos a explicar como configurar un Core Data Manager.
- Lo primero que necesitamos crear es el **NSManagedObjectModel**, que es el objeto que definirá el esquema de la base de datos de tu aplicación

```
NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"Model" withExtension:@"momd"];
NSManagedObjectModel managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
return __managedObjectModel;
```

# Core Data Manager

- Después tenemos que definir el **NSPersistentStoreCoordinator** que es quien asocia las "stores" (que reciben las órdenes del Coordinator y las trasladan a la base de datos) con el modelo (**NSManagedObjectModel**).

```

NSPersistentStoreCoordinator *persistentStoreCoordinator;
NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@"SongsDatabase.sqlite"]; //nombre de la base de datos

persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:
[self managedObjectModel]];
[persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:nil
URL:storeURL options:nil error:&error]

```

# Core Data Manager

- Para interactuar con la base de datos necesitamos un **NSManagedObjectContext** que se responsabiliza de manejar una colección de objetos.
- Estos objetos son instancias al modelo, después de ejecutar un **save** el **Coordinator** que creamos anteriormente actualiza la base de datos.

```
NSManagedObjectContext * managedObjectContext;  
managedObjectContext = [[NSManagedObjectContext alloc] init];  
[managedObjectContext setPersistentStoreCoordinator: persistentStoreCoordinator];
```

# Integración con iCloud

- En iOS 5 se incluyó la sincronización de los datos de nuestras aplicaciones con la nube (**iCloud**).
- Existe la posibilidad de realizar la persistencia de los datos en nuestro dispositivo mediante **Core Data** y de manera adicional, sincronizar con **iCloud**.



# Integración con iCloud

- Primero de todo, nuestra aplicación debe estar activada en el *Provisioning Portal* como apta para **iCloud**.
- Posteriormente, modificaremos el objeto de contexto de nuestra aplicación creada con **Core Data**.
- Debemos añadir código para que se realicen cambios de manera asíncrona cuando algún dato almacenado en **iCloud** se modifique.
- Además, debemos notificar a **iCloud** cuando modifiquemos los datos localmente.