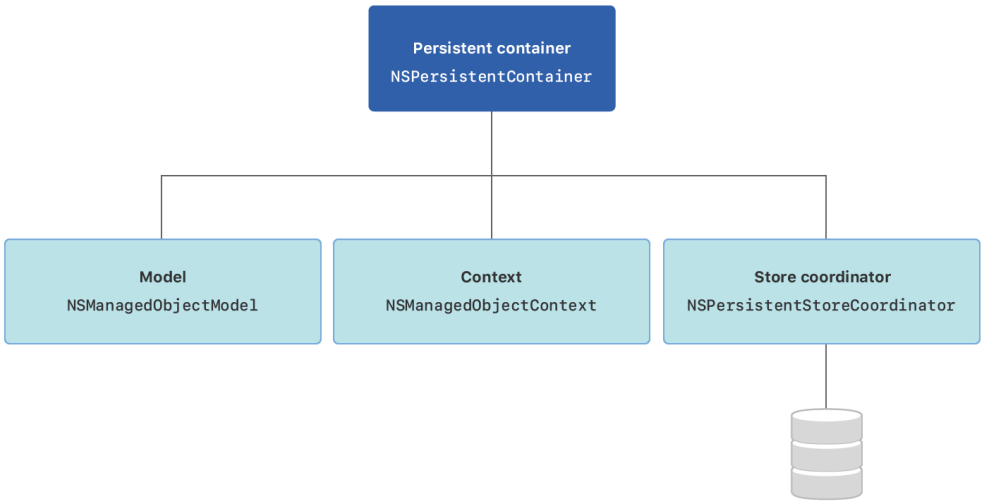
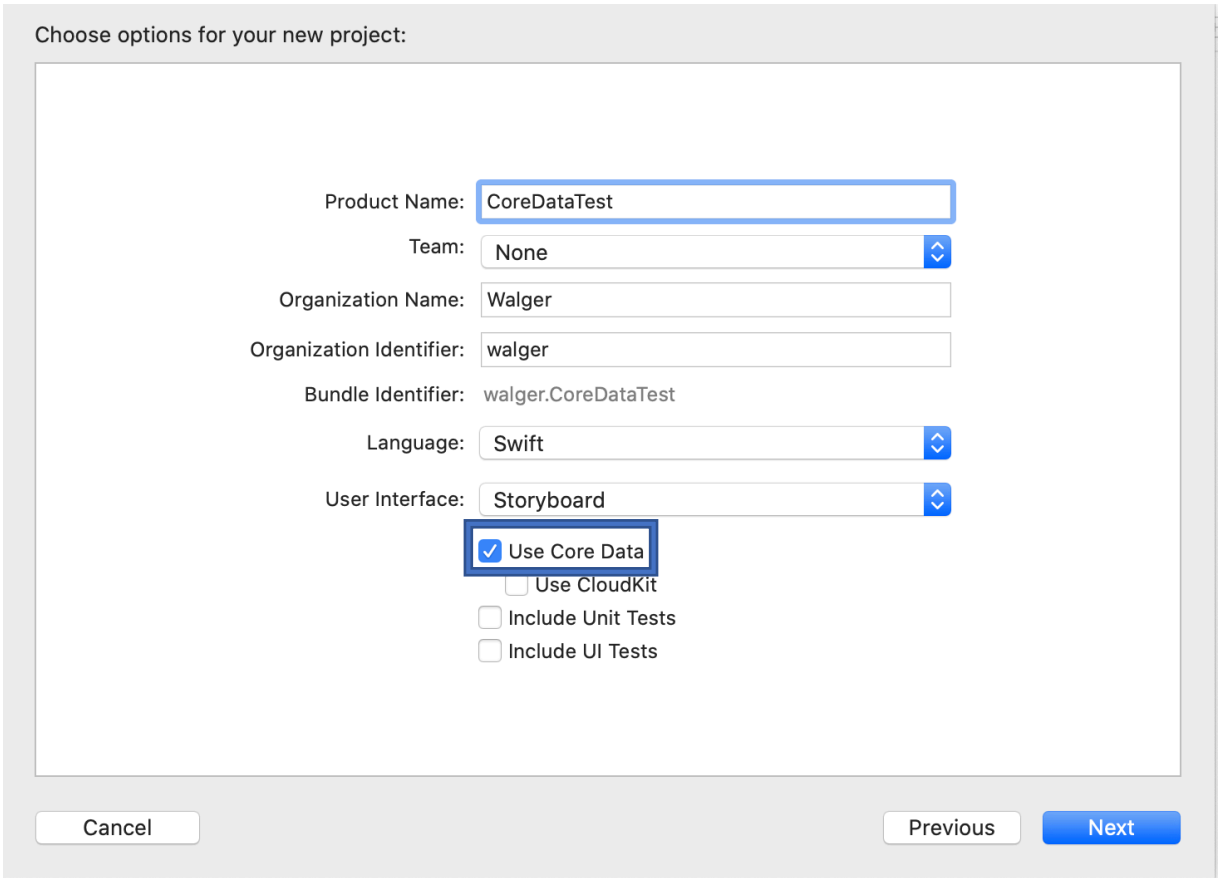
CoreData CheatSheet

Stack



iOS-App mit CoreData erstellen: Haken bei „Use CoreData“ setzen



Erzeugt:

- PersistentContainer mit Zugriff auf das Datenmodell - hier „CoreDataTest“ – in der AppDelegate

```
// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {
    /*
     The persistent container for the application. This implementation
     creates and returns a container, having loaded the store for the
     application to it.
    */
    let container = NSPersistentContainer(name: "CoreDataTest")
    container.loadPersistentStores(completionHandler: { (storeDescription, error) in
        if let error = error as NSError? {
            fatalError("Unresolved error \(error), \(error.userInfo)")
        }
    })
    return container
}()
```

Über den PersistentContainer wird der Context aufgerufen:

```
let context =  (UIApplication.shared.delegate as!
    AppDelegate).persistentContainer.viewContext
```

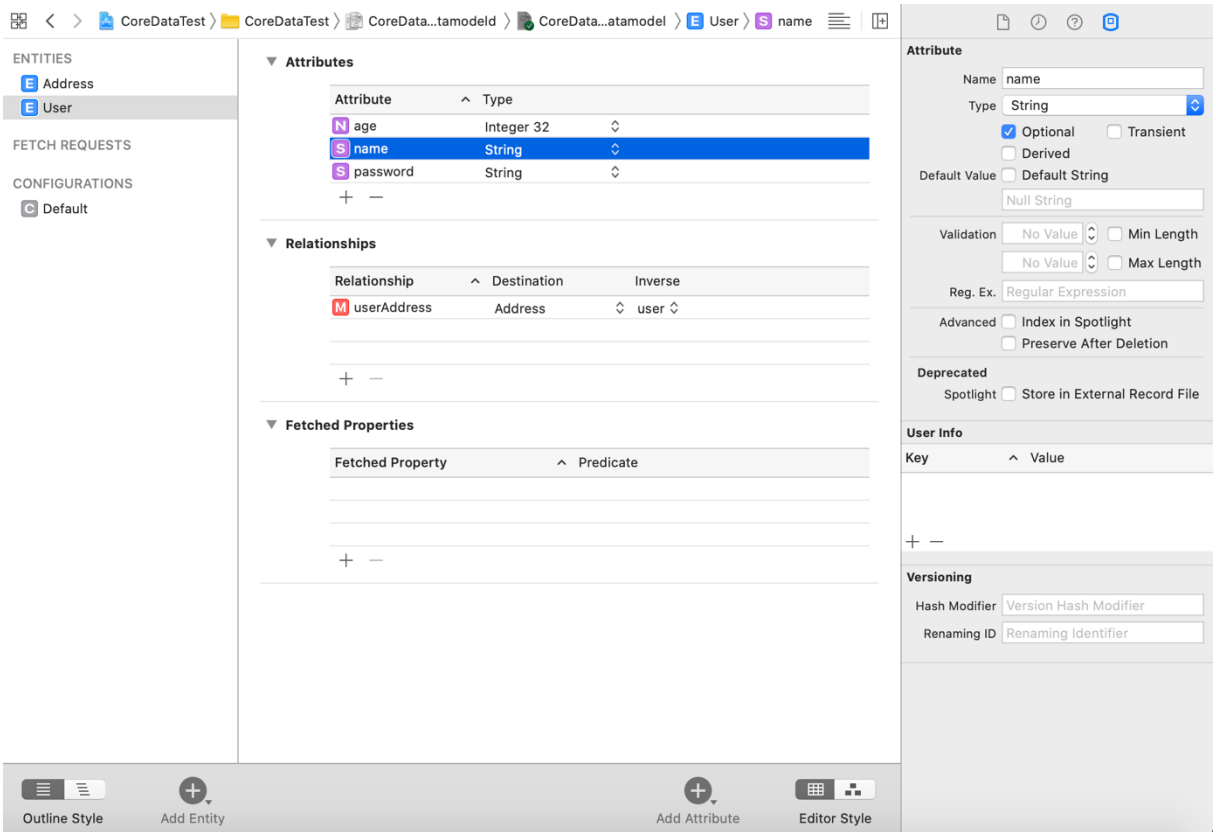
Methode saveContext() speichert Änderungen im Context persistent

```
// MARK: - Core Data Saving support

func saveContext () {
    let context = persistentContainer.viewContext
    if context.hasChanges {
        do {
            try context.save()
        } catch {
            let nserror = error as NSError
            fatalError("Unresolved error \(nserror), \(nserror.userInfo)")
        }
    }
}
```

Datenmodell

- gespeichert in der Datei „Projektname“.xcdatamodeld
- ermöglicht Definition von Entitäten



Ablauf:

1. Neue Entität hinzufügen mit „Add Entity“
2. Attribute definieren mit „Add Attribute“

In jeder Klasse, in der man CoreData verwendet:

```
import CoreData
```

Daten speichern mit CoreData

1. AppDelegate nach dem Kontext fragen
2. Neues ManagedObject erstellen und in den Kontext einfügen
3. Mit Key-Value Coding die Attribute setzen, dabei sehr genau sein
4. Änderungen „committen“ und Objekt zum Interface hinzufügen

```
guard let appDelegate =
    UIApplication.shared.delegate as? AppDelegate else {
    return
  }
  // 1
  let managedContext =
    appDelegate.persistentContainer.viewContext
  // 2
    let entity = NSEntityDescription.entity(forEntityName: "User",
                                            in: managedContext)!
      let user = NSManagedObject(entity: entity,
                                 insertInto: managedContext)
      // 3
      user.setValue(name, forKeyPath: "name")

      // 4
      do {
        try managedContext.save()
        [redacted]
      } catch let error as NSError {
        print("Could not save. \(error), \(error.userInfo)")
      }
}
```

Daten laden mit CoreData

1. AppDelegate nach dem Context fragen
2. FetchRequest aufsetzen mit Rückgabetyp (hier: NSManagedObject)
3. Fetch-Methode vom Context aufrufen und Ergebnis speichern

```
//1
  guard let appDelegate =
    UIApplication.shared.delegate as? AppDelegate else {
      return
  }

  let managedContext =
    appDelegate.persistentContainer.viewContext

  //2
  let fetchRequest =
    NSFetchRequest<NSManagedObject>(entityName: "User")

  //3
  do {
    users = try managedContext.fetch(fetchRequest)
  } catch let error as NSError {
    print("Could not fetch. \(error), \(error.userInfo)")
  }
```

Predicates

```
let name = "Peter";
fetchRequest.predicate = NSPredicate(format: "name == %@", name);
```

```
fetchRequest.predicate = NSPredicate(format: "age >= 30")
```

SortDescriptors

```
fetchRequest.sortDescriptors = [NSSortDescriptor(key: "name", ascending: true)]
```

```
let sortDescriptor1 = NSSortDescriptor(key: "name", ascending: true)
let sortDescriptor2 = NSSortDescriptor(key: "date", ascending: true)
fetchRequest.sortDescriptors = [sortDescriptor1, sortDescriptor2]
```