SPARC iOS Training

Lesson 4: Intro to Core Data

Agenda

- Basics of Core Data
- Terminology
- Coding Demo

What is Core Data?

- Core Data is a schema-driven object graph management and persistence framework
- Provides an infrastructure for managing all the changes to your model objects.
- Allows you to keep just a subset of your model objects in memory at any given time.
- Has an infrastructure for data store versioning and migration.

Database Equivalents

- In simple terms, a class would (more or less) represent a table in your database.
- A data object would represent a record (row) in your database
- An instance variable in that data object would represent a field in that database (column in that row)

Core Data's Role

- Core Data presents the records from your database to your application as an object
- When objects are edited, Core Data is responsible for updating the respective record in the database table
- Core data is also responsible for creating new records or destroying existing ones when objects are created or destroyed.

Components

- **Entity** = The Class (or db table)
- **Attribute** = iVar (or db column)
- **Relationship** = A key join between entities

- Both attributes and relationships are represented by properties in your data model class.
  - “employee.department” == @“Sales”
  - “employee.department” relationship to associated department object

Important 

- Don’t think of Core Data as a SQL database
- Think of it as a way to persist your objects.

Core Data Concepts

- Persistent Store
- Data Model
- Persistent Store Coordinator
- Managed Object Context
  - Entities
  - Managed Objects
- Fetch Request
  - Predicates

Persistent Store / Data Model

- The Persistent Store is where Core Data stores its data
- Every persistent store is associated with a single data model
- The data model defines the types of data that can be stored in its associated persistent store

Persistent Store Coordinator

- Handles calls from different classes that trigger data access (read/writes)
- Manages all requests to access data to prevent conflicts database locking which can happen if you have multiple classes trying to access the data store at the same time

Managed Object Context

- Serves as the gateway between your entities and the rest of Core Data
- Maintains state of all managed objects
- Tracks changes to managed objects
- Coordinates data changes with the persistent store coordinator

Fetch Requests

- Essentially a query for what we want to get from the persistent store
- As a minimum you must specify an entity for the request.
- Predicates filter your requests
- Sort descriptors sort your results

Faulting and Uniquing

- Faulting is a mechanism Core Data employs to reduce your application's memory usage.
- Uniquing ensures that, in a given managed object context, you never have more than one managed object to represent a given record.
- Faulting allows Core Data to put boundaries on the object graph
- Transparent (it fetches automagically)

Demo