Data Storage

Dr. Sarah Abraham
Model Layer of MVC

- Contains the data to be displayed

- Data can be:
  - Stored on device
  - Pulled down from a server

- Data displayed in app should be:
  - Personalized
  - Secure
User Defaults and Plists

- Both provide storage on the device itself
- User Defaults holds persistent key/value pairs
  - Good for small amounts of data
  - Usually related to device user
- Plists provide XML input
  - Good for data that is consistent between users
Core Data

- Framework for modeling data in object-oriented way
- Allows for data persistence on device
- Used for non-trivial storage
- Not a database in of itself
- Can be mapped to a true database management system like SQL/SQLite
Core Data Features

- Models data efficiently
- Manages data object life cycles
- Tracks changes to data
- Supports undo functionality
- Saves data to disk
Managed Object Model

- Defines structure of data
- Data types
- Relationships
- Xcode provides design tools to build object model
Managed Object Context

- Temporary scratch space in memory
- Objects fetched from persistent store placed in context for manipulation
- Monitors for changes to data
- Can save data back to Persistent Store
Entities, Attributes and Relationships

- Entities are data model instances in Core Data
  - Table in relational database
  - Example: Employee entity defines a company employee
- Attributes are properties stored in entities
  - A column in a relationship database table
  - Example: Employee entity has attributes name, position, salary
- Relationships are connections between entities
  - One-to-One (Country to Capital; Capital to Country)
  - One-to-Many (Manager to Employee)
  - Many-to-One (Employee to Manager)
Using Core Data

- Select “Use Core Data” as option for new project
- `.xcdatamodeld` file defines entities, attributes and relationships
Displaying Core Data

- Create variable to hold instances of managed objects:
  ```swift
  var managedObjects = [NSManagedObject]()
  ```
- Allows other objects in program to access and display managed objects
Writing to Core Data

```swift
func addPerson(name: String, occupation: String, age: Int) {
    let appDelegate = UIApplication.shared.delegate as! AppDelegate
    let managedContext = appDelegate.managedObjectContext
    let entity = NSEntityDescription.entity(forEntityName: "Person", in: managedContext)

    let person = NSManagedObject(entity: entity!, insertInto: managedContext)

    person.setValue(name, forKey: "name")
    person.setValue(age, forKey: "age")
    person.setValue(occupation, forKey: "occupation")

    do {
        try managedContext.save()
    } catch {
        let nserror = error as NSError
        NSLog("Unable to save \(nserror), \(nserror.userInfo)")
        abort()
    }

    people.append(person) // people contains NSManagedObjects
}
```
KVC

- Key Value Coding
- Ability to read and set a property using its name
- NSObject contains default methods:
  - `setValue(AnyObject?, forKey: String)`
  - `value(forKey: String)`
- Any class derived from NSObject can use KVC
- Managed Objects must be accessed with key-value coding
let appDelegate = UIApplication.shared.delegate as! AppDelegate
let managedContext = appDelegate.managedObjectContext
let fetchRequest = NSFetchRequest<NSFetchRequestResult>(entityName: "Person")
var fetchedResults: [NSManagedObject]? = nil

do {
    try fetchedResults = managedContext.fetch(fetchRequest) as? [NSManagedObject]
} catch {
    let nserror = error as NSError
    NSLog("Unable to fetch \(nserror), \(nserror.userInfo")")
    abort()
}

if let results = fetchedResults {
    people = results
}
Core Data Demo
Quiz Question!

What part of the Core Data system saves data back to persistent storage?

A. Managed Object Model
B. Managed Object Context
C. Key-value Coding
Quiz Answer

- B: The Managed Object Context allows for data stored in a Managed Object Model to be written back in persistent storage