Saving, Accessing, and Deleting Data

Andrew Bancroft

@andrewcbancroft www.andrewcbancroft.com

Overview

Saving Data with NSEntityDescription

Accessing Data with NSFetchRequest

Filtering Data with NSPredicate

Sorting Data with NSSortDescriptor

Implement Editor Screen

Implement Delete ShoutOut on Details Screen

Saving Data

Call the Type’s initializer!

Task: Create a New ShoutOut

What might be a rational line of code to write in order to create a new ShoutOut instance?

```
let shoutOut = ~~ShoutOut~~()
```

Task: Create a New ShoutOut

What might be a rational line of code to write in order to create a new ShoutOut instance?

Inserting Data with NSEntityDescription

```
let shoutOut = NSEntityDescription.insertNewObject(
               forEntityName: “ShoutOut”,
               into: mainContext) as! ShoutOut

shoutOut.from = “Andrew”
```

Insert Data with NSEntityDescription

Supply the entity name in the form of a String

Supply an instance of NSManagedObjectContext

Returns an NSManagedObject instance

Must *cast result* to NSManagedObject subclass Type

```swift
let shoutOut = NSEntityDescription.insertNewObject(
               forEntityName: “ShoutOut”,
               into: mainContext) as! ShoutOut

shoutOut.from = “Andrew”

mainContext.save()
```

Save with NSManagedObjectContext

Use NSManagedObjectContext’s save() method to insert objects into the persistent store.

The save() method can throw, so you must wrap the call in a do-catch block.

```
let shoutOut = NSEntityDescription.insertNewObject(
               forEntityName: “ShoutOut”,
               into: mainContext) as! ShoutOut

shoutOut.from = “Andrew”

do { try mainContext.save() } catch _ {}
```

Save with NSManagedObjectContext

Use NSManagedObjectContext’s save() method to insert objects into the persistent store.

The save() method can throw, so you must wrap the call in a do-catch block.

Inserting Data with NSEntityDescription

1. Initialize a new Entity instance with NSEntityDescription.insertNewObject
2. Cast the returned NSManagedObject to the correct NSManagedObject subclass Type and set properties
3. Save!

Demo

Practice using NSEntityDescription

Seed a set of Employee objects into the persistent store

Retrieving Data

Retrieving Data with NSFetchRequest

Workflow for Retrieving Data

1. **Initialize an NSFetchRequest**
2. **Use NSManagedObjectContext instance to perform the fetch request**
3. **Work with the retrieved data**

```
let shoutOutsFetchRequest = NSFetchRequest<ShoutOut>(
                               entityName: “ShoutOut”)


let shoutOuts = mainContext.fetch(shoutOutsFetchRequest)
```

Retrieve Data with NSFetchRequest

Initialize an NSFetchRequest

Perform the fetch with an instance of NSManagedObjectContext

The fetch() method can throw, so you must wrap the call in a do-catch block.

```
let shoutOutsFetchRequest = NSFetchRequest<ShoutOut>(
                          entityName: “ShoutOut”)

do {
  let shoutOuts = try mainContext.fetch(shoutOutsFetchRequest)
} catch _ {}
```

Retrieve Data with NSFetchRequest

Initialize an NSFetchRequest

Perform the fetch with an instance of NSManagedObjectContext

The fetch() method can throw, so you must wrap the call in a do-catch block.

Demo

Set up a place to experiment with in the unit test target

Practice retrieving an object from the persistent store

Filtering Data

Filtering Data with NSPredicate

What Is a Predicate?

Criteria

Conditions

True/False Expressions

```
if firstName == “Luke” {

  shoutOut.shoutCategory =
  “Great Job!”

}
```

```
if firstName == “Luke” {

  shoutOut.shoutCategory =
  “Great Job!”

}
```

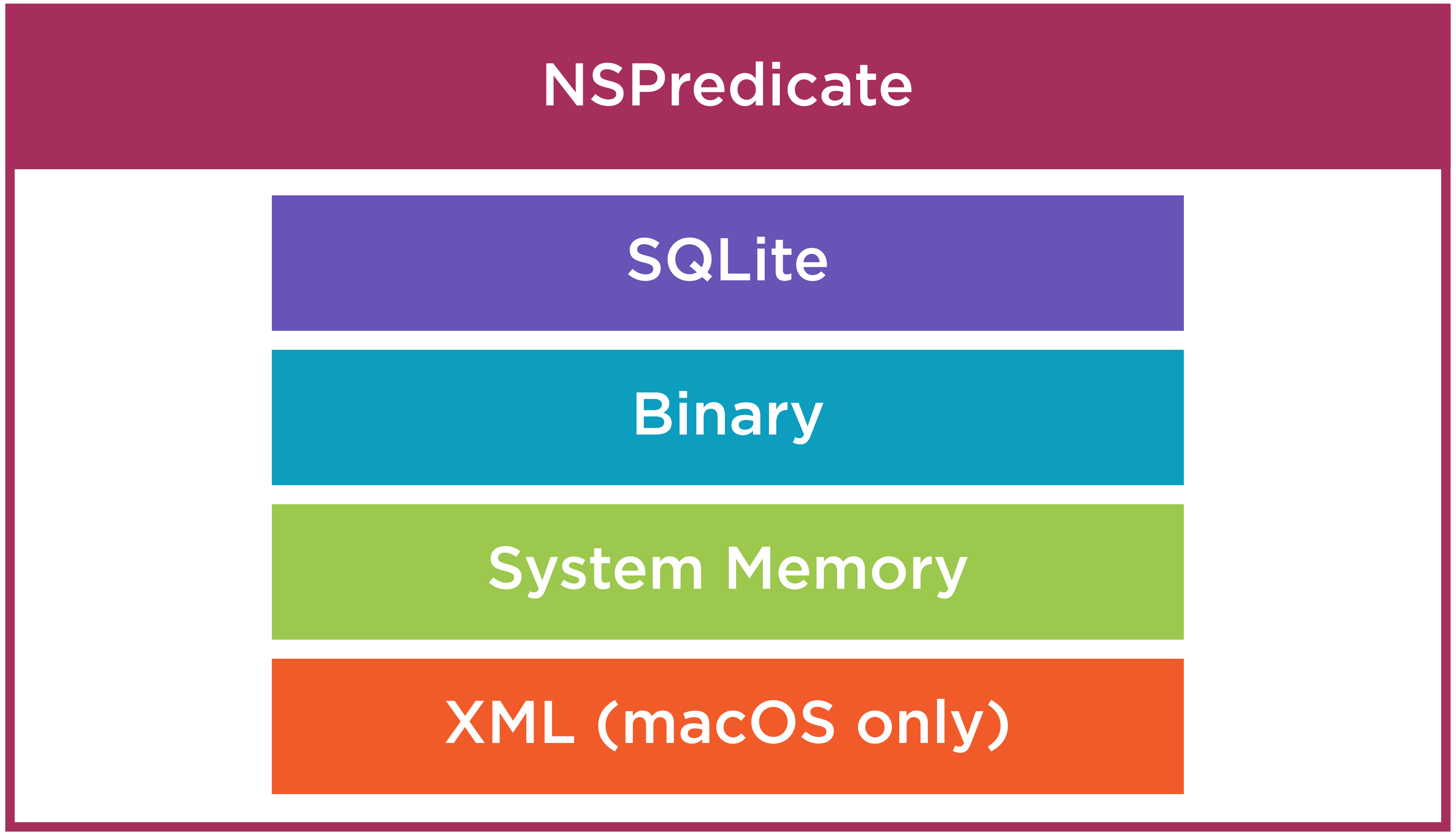
- firstName == “Luke” is a predicate in the general sense of the term

```
SELECT
  *
FROM
  ShoutOuts
WHERE
  shoutCategory = ‘Great Job!’
```

```
SELECT
  *
FROM
  ShoutOuts
WHERE
  shoutCategory = ‘Great Job!’
```

- WHERE clause of a SQL statement is also a predicate

NSPredicate as an Abstraction for Filtering



```swift
let shoutOutsFetchRequest = NSFetchRequest<ShoutOut>(
                    entityName: “ShoutOut”)

let predicate = NSPredicate(format: “%K == %@”,
                   #keyPath(ShoutOut.shoutCategory),
                   “Great Job!”)

shoutOutsFetchRequest.predicate = predicate
```

Initialize and Apply an NSPredicate

Initialize an NSFetchRequest

Initialize an NSPredicate

- **%K is a placeholder for the key (Attribute) containing values for comparison**
- **== is a comparison operator**
- **%@ is an object placeholder that values in the key (%K) placeholder will be compared to**

Assign predicate to NSFetchRequest instance

NSPredicate Comparison Operators

Basic	String	Aggregate
=, ==	CONTAINS	ANY
>=, =>	BEGINSWITH	ALL
<=, =<	ENDSWITH	NONE
<	LIKE	IN
>	MATCHES	
!=, <>		
BETWEEN		

Apple Developer Documentation

http://bit.ly/PredicateProgrammingGuide

Demo

New experiment in unit test target

Practice filtering data with NSPredicate

Sorting Data

Sorting Data with NSSortDescriptor

```swift
let shoutOutsFetchRequest = NSFetchRequest<ShoutOut>(
                entityName: “ShoutOut”)

let categorySort = NSSortDescriptor(
                key: #keyPath(ShoutOut.shoutCategory),
                ascending: true)

shoutOutsFetchRequest.sortDescriptors = [categorySort]
```

Initialize and Apply an NSSortDescriptor

Initialize an NSFetchRequest

Initialize one or more NSSortDescriptor instances

Assign array of NSSortDescriptor instances to NSFetchRequest instance

Demo

Use Core Data skills to implement Editor screen of app

End goal: Populate “to Employee” picker with sorted list of Employees

Deleting Data

Workflow for Deleting Data

1. **Perform fetch request**
2. **Call delete on NSManagedObjectContext**
3. **Call save on NSManagedObjectContext**

```
do {

  let shoutOuts = try mainContext.fetch(shoutOutsFetchRequest)
  let firstShoutOut = shoutOuts[0]
  mainContext.delete(firstShoutOut)
  do { try mainContext.save() } catch _ {}

} catch _ {}
```

Delete a Single Object

Perform fetch request

Call delete on NSManagedObjectContext instance (pass object to delete)

Call save on NSManagedObjectContext instance

```
do {
  let shoutOuts = try mainContext.fetch(shoutOutsFetchRequest)
  for shoutOut in shoutOuts {
     mainContext.delete(shoutOut)
  }
  do { try mainContext.save() } catch _ {}
} catch _ {}
```

Delete Multiple Objects

Perform fetch request

Loop over each instance, calling delete on NSManagedObjectContext instance

Call save on NSManagedObjectContext instance

Demo

Practice deleting an object from the persistent store

Implement delete ShoutOut behavior of the ShoutOutDetailsViewController

Summary

Saved a list of Employees to the persistent store

Learned how to filter and sort data that's retrieved from the persistent store

Implemented Editor Screen

Implemented Delete ShoutOut on Details Screen

Coming up: Showing and Synchronizing data!