

Introduction to SwiftData

Reena Thakkar

@ Swift Ahmedabad on Saturday 21st December 2024

Today

- Introduction to SwiftData
- Core Data
- SwiftData
- Final Thoughts

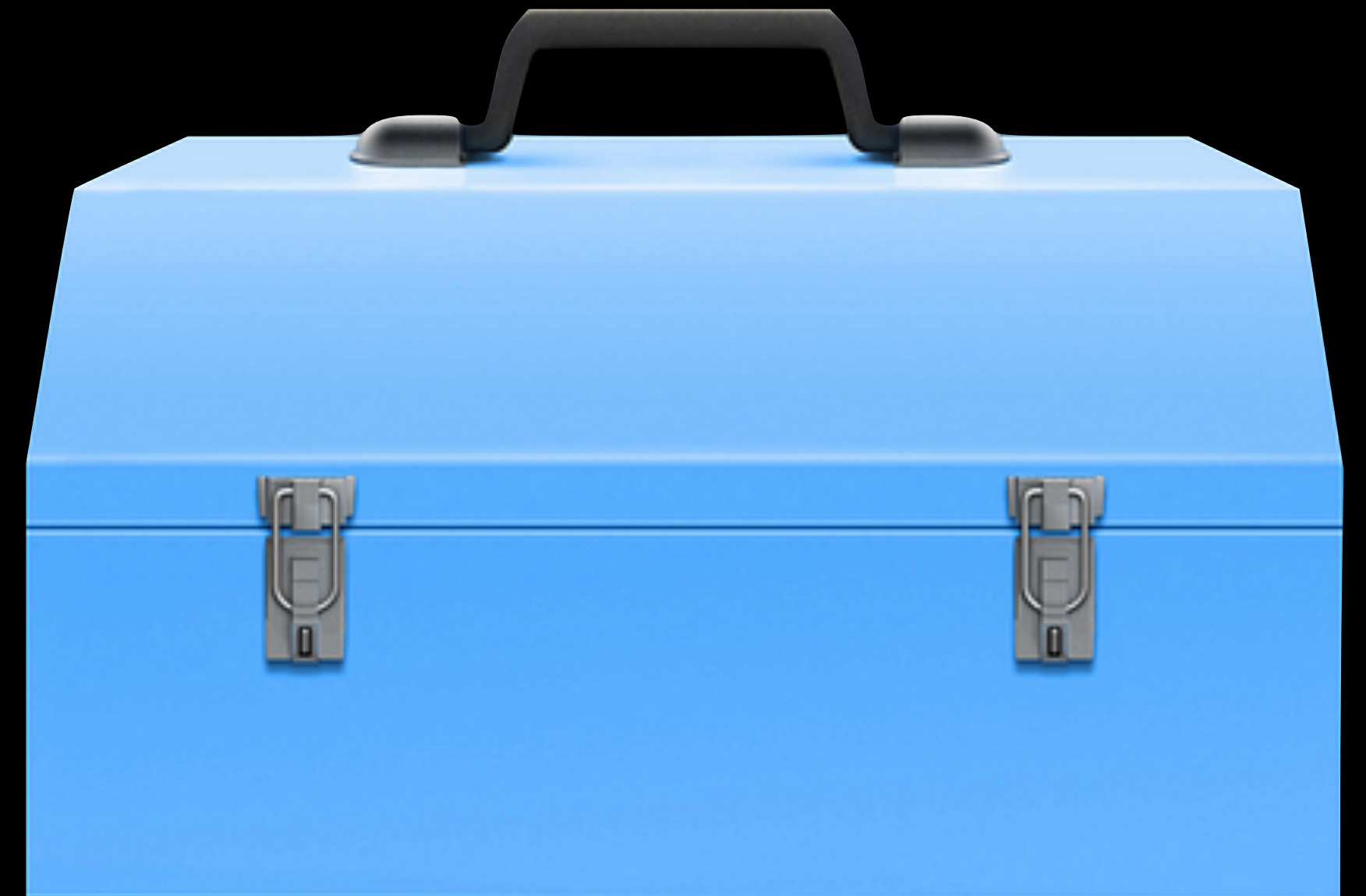
Introduction To SwiftData

- Introduced with iOS 17
- For below iOS 17
 - CoreData
 - Raw SQLite
 - GRDB
 - Other OpenSource Libraries



Introduction To SwiftData

- It is a Framework not Database
- Database used is “SQLite”
- It still uses “CoreData” internally
- Interoperate / Migrate to SwiftData Gradually



Introduction

To SwiftData

- It is a Framework not Database
- Database used is “SQLite”
- It still uses “CoreData” internally
- Interoperate / Migrate to SwiftData Gradually



Introduction To SwiftData

- It is a Framework not Database
- Database used is “SQLite”
- It still uses “CoreData” internally
- Interoperate / Migrate to SwiftData Gradually



Introduction To SwiftData

- It is a Framework not Database
- Database used is “SQLite”
- It still uses “CoreData” internally
- Interoperate / Migrate to SwiftData Gradually



WWDC 23: Migrate to SwiftData - <https://www.youtube.com/watch?v=olsjgo4Qb4A>

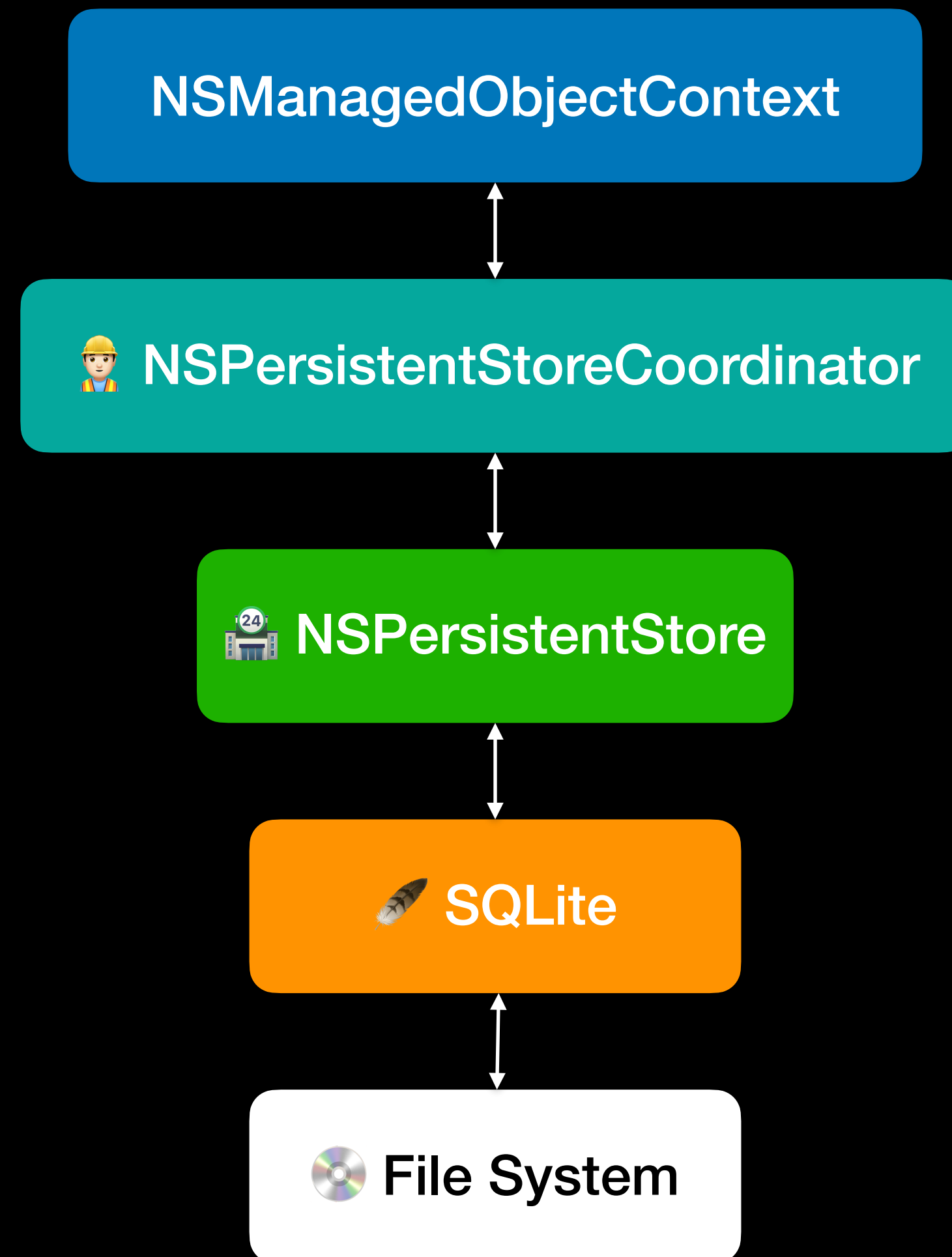
<https://www.hackingwithswift.com/quick-start/swiftdata/how-to-migrate-an-app-from-core-data-to-swiftdata>

Today

- Introduction to SwiftData
- Core Data
- SwiftData
- Final Thoughts

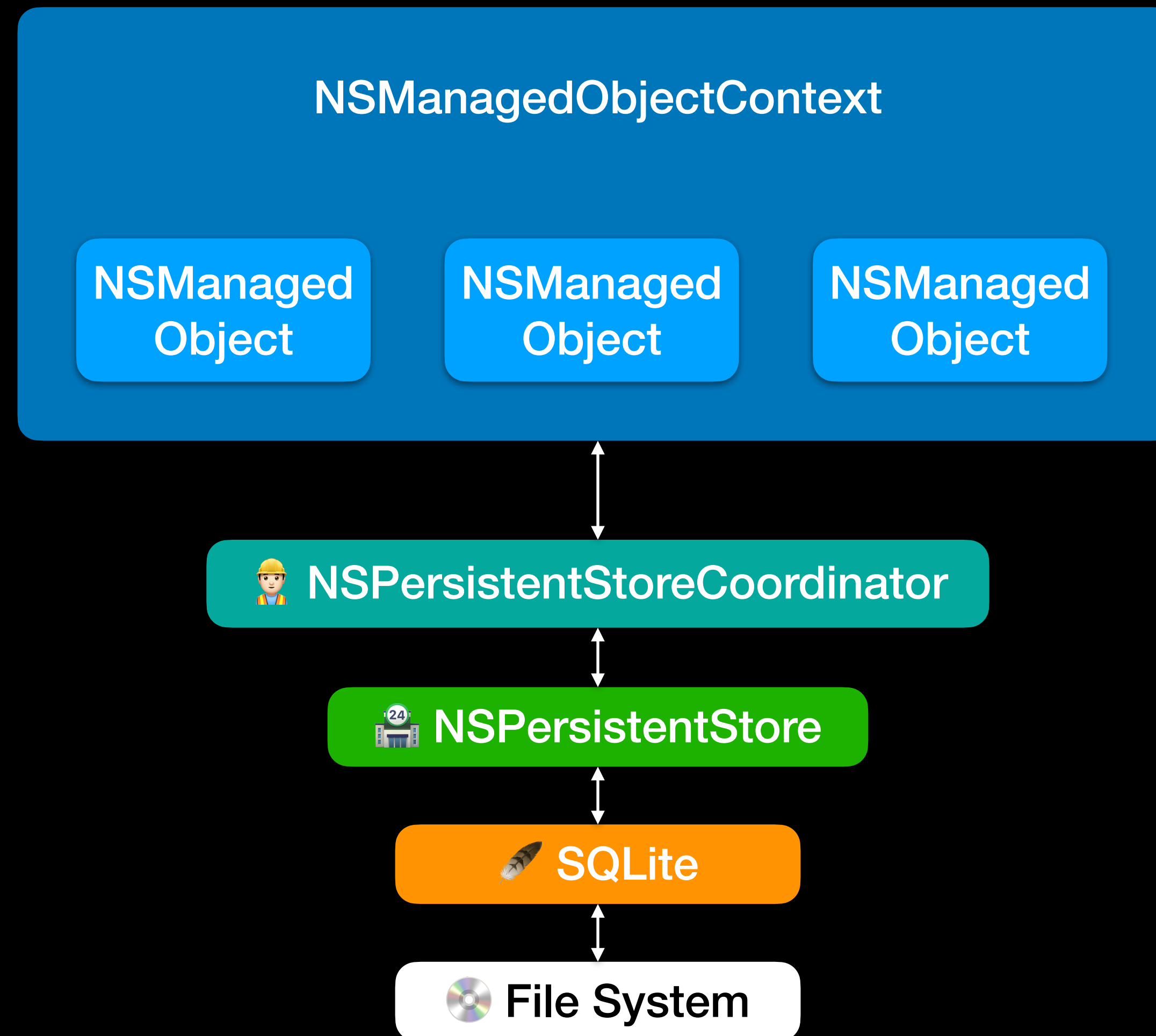
Core Data Stack

BEFORE iOS 9.0



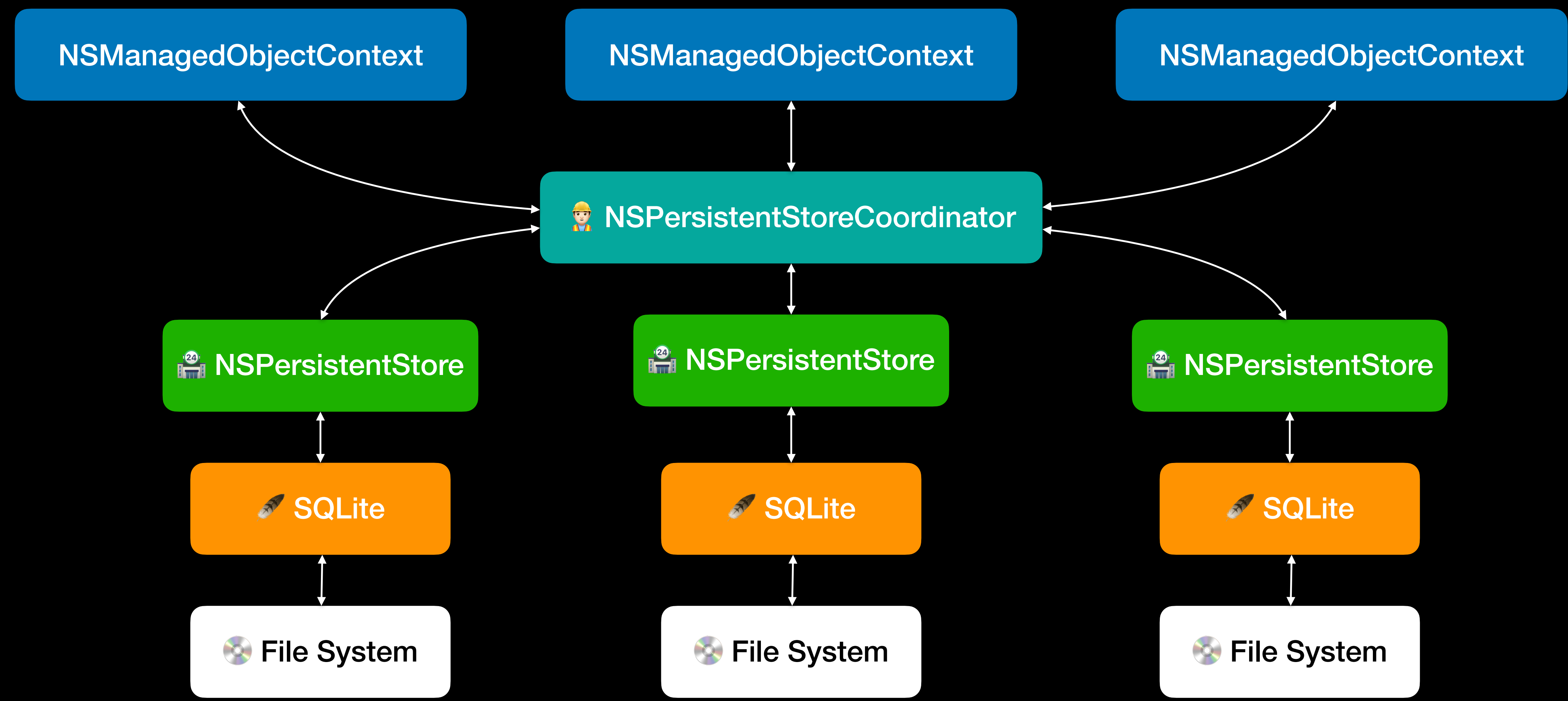
Core Data Stack

BEFORE iOS 9.0



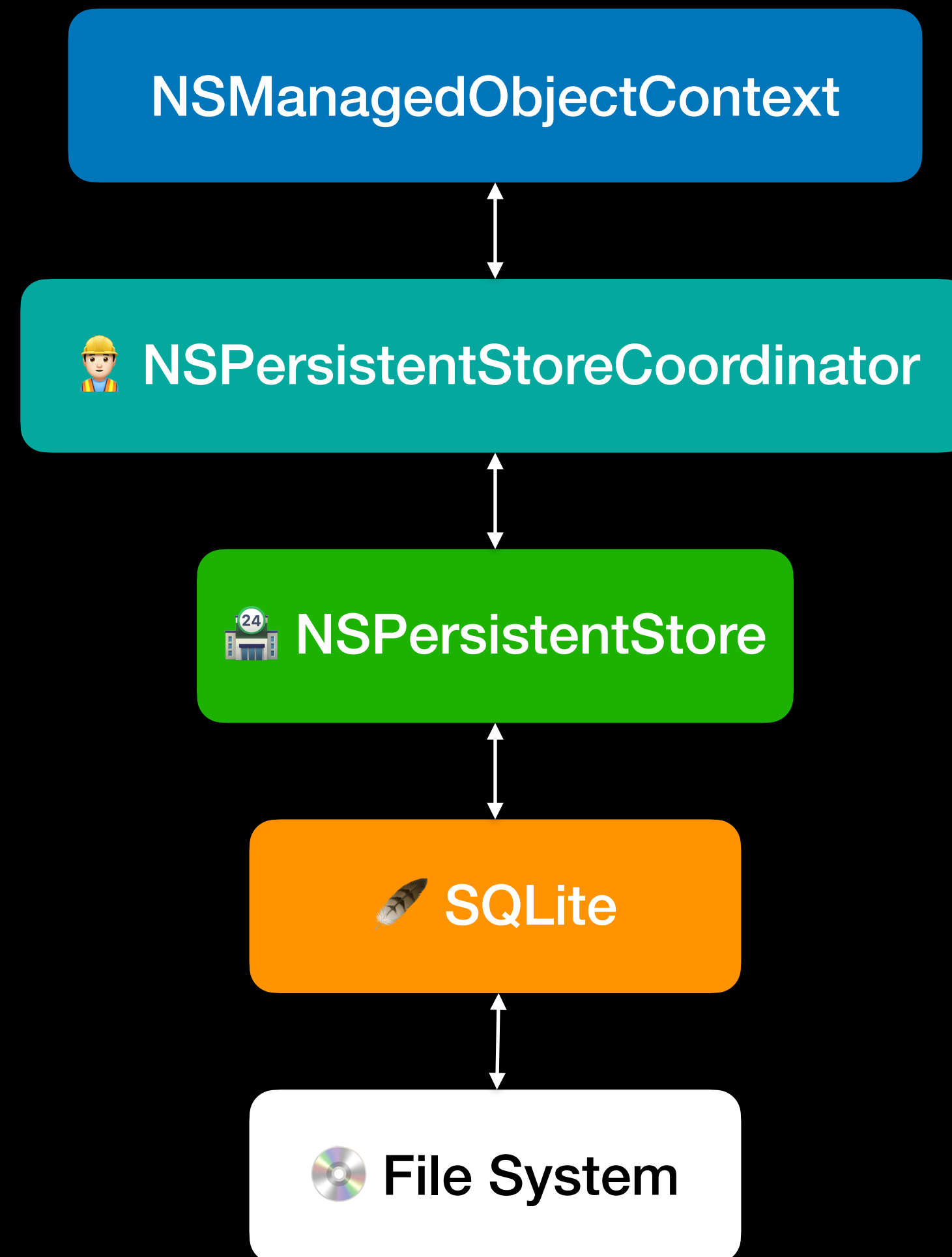
Core Data Stack

BEFORE iOS 9.0



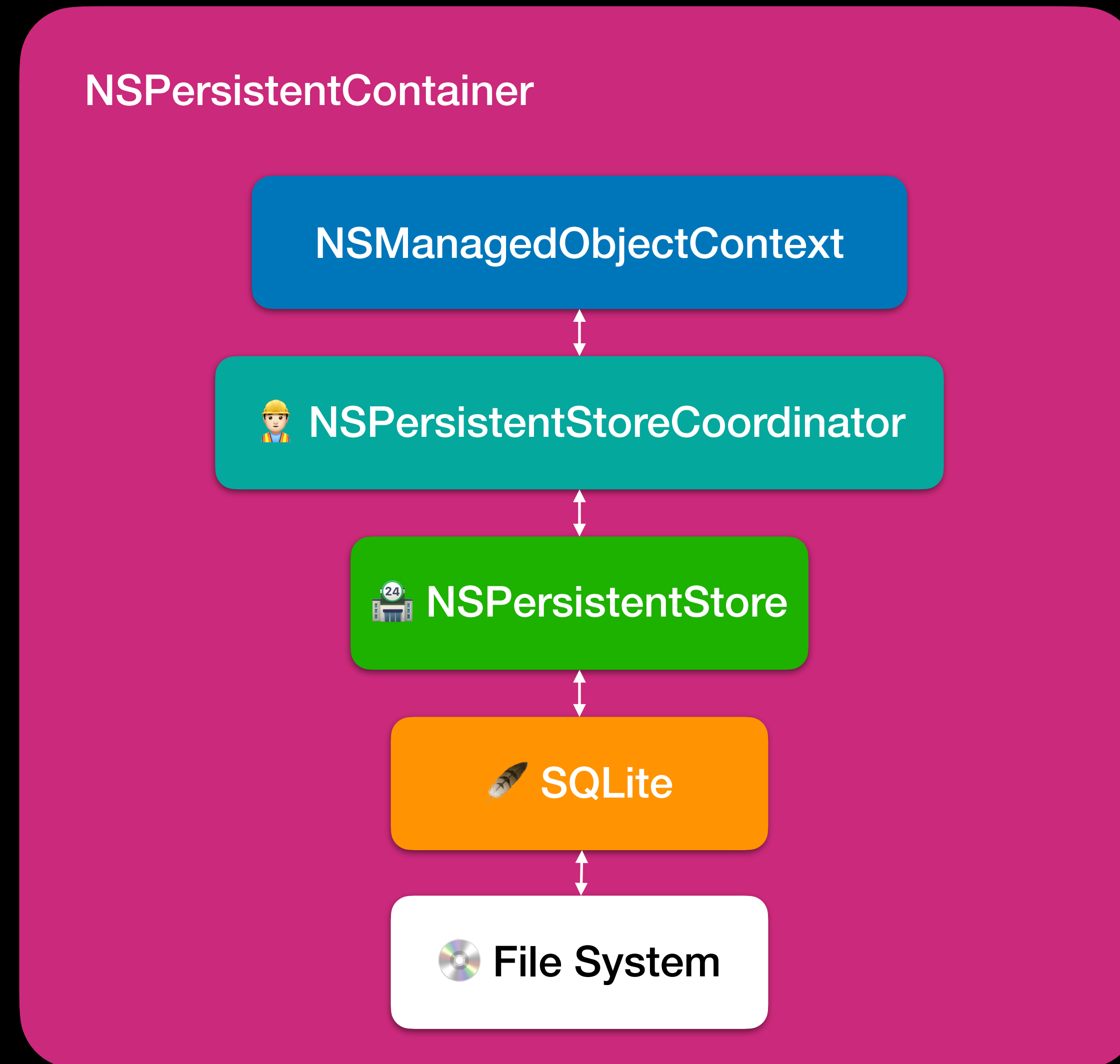
Core Data Stack

BEFORE iOS 9.0



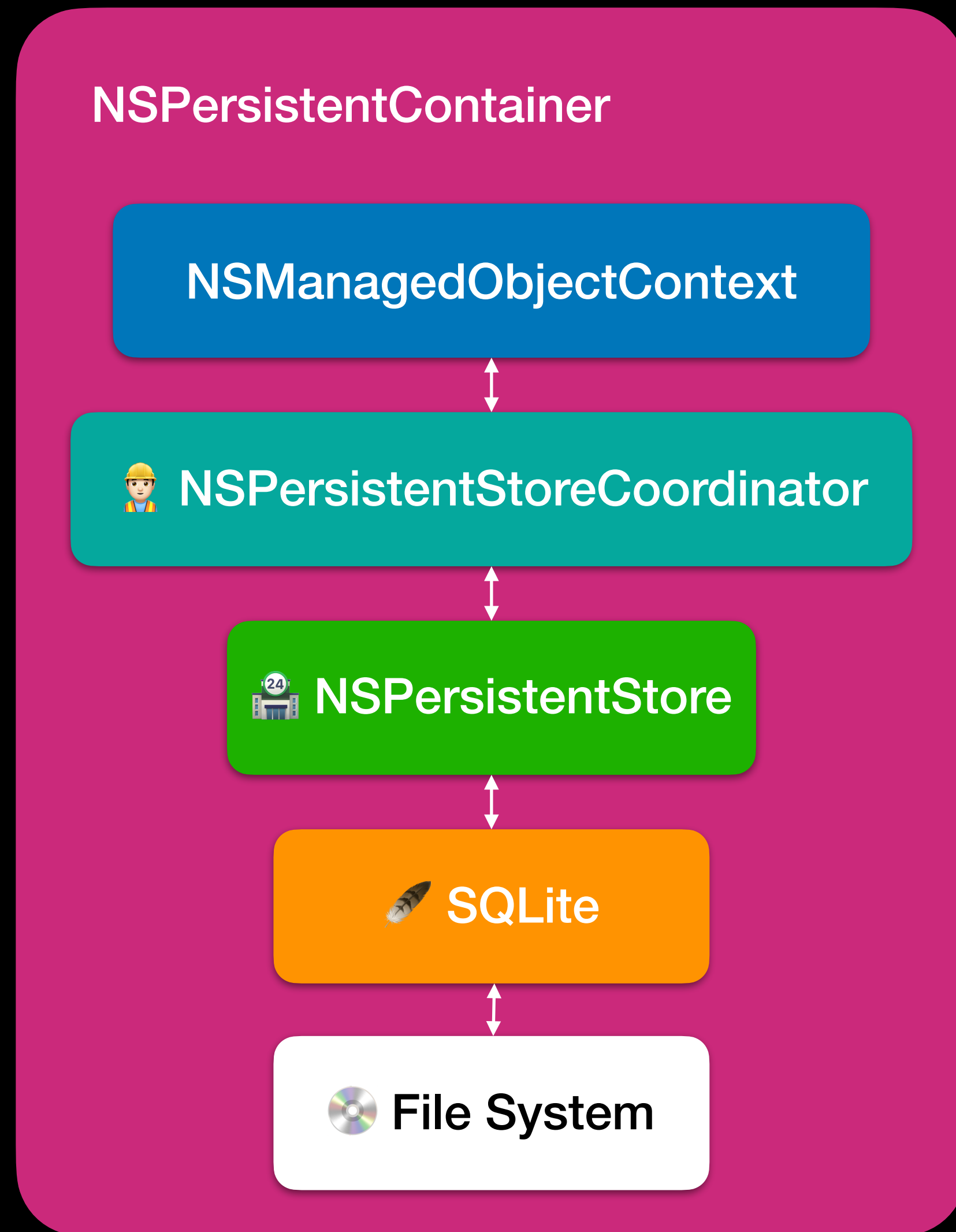
Core Data Stack

iOS 10.0



Core Data Stack

iOS 10.0



Let's Configure

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name: "PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {  
    public static let container: NSPersistentContainer = {  
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }  
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }  
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)  
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in  
            if let error = error as NSError? {  
                print("Unresolved error \(error), \(error.userInfo)")  
            }  
        })  
        return container  
    }()  
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Setup

iOS 10.0

```
public enum PersistentModelStore {
    public static let container: NSPersistentContainer = {
        guard let modelURL = Bundle.main.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = NSPersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Core Data Entities/Relationships/Indexing

ENTITIES

E Author

E Book

COMPOSITE TYPES

FETCH REQUESTS

CONFIGURATIONS

C Default

Attributes

Attribute	Type
id	UUID
name	String

+

—

Relationships

Relationship	Destination	Inverse
books	Book	authors

+

—

Fetches Properties

Fetches Property	Predicate
------------------	-----------

+

—

Entity

Name

Author

Abstract Entity

Parent Entity

No Parent Entity

Class

Name

Author

Module

Global namespace

Codegen

Class Definition

Constraints

No Content

+

—

Spotlight

Display Name

Expression

User Info

Key	Value
-----	-------

+

—

Versioning

Hash Modifier

Version Hash Modifier

Renaming ID

Renaming Identifier

Outline Style

Add Entity

Add Attribute

Core Data Entities/Relationships/Indexing

ENTITIES

E Author

E Book

COMPOSITE TYPES

FETCH REQUESTS

CONFIGURATIONS

C Default

Attributes

Attribute	Type
id	UUID
name	String

Relationships

Relationship	Destination	Inverse
books	Book	authors

Fetches Properties

Fetches Property	Predicate
------------------	-----------

Entity

Name

Author

Abstract Entity

Parent Entity

No Parent Entity

Class

Name

Author

Module

Global namespace

Codegen

Class Definition

Constraints

No Content

Spotlight

Display Name

Expression

User Info

Key	Value
-----	-------

Versioning

Hash Modifier

Version Hash Modifier

Renaming ID

Renaming Identifier

Outline Style

Add Entity

Add Attribute

Core Data Entities/Relationships/Indexing

ENTITIES

E Author

E Book

COMPOSITE TYPES

FETCH REQUESTS

CONFIGURATIONS

C Default

Attributes

Attribute	Type
id	UUID
name	String

+

−

Relationships

Relationship	Destination	Inverse
books	Book	authors

+

−

Fetches Properties

Fetches Property	Predicate
------------------	-----------

+

−

Entity

Name

Author

Abstract Entity

Parent Entity

No Parent Entity

Class

Name

Author

Module

Global namespace

Codegen

Class Definition

Constraints

No Content

+

−

Spotlight

Display Name

Expression

User Info

Key	Value
-----	-------

+

−

Versioning

Hash Modifier

Version Hash Modifier

Renaming ID

Renaming Identifier

Outline Style

Add Entity

Add Attribute

Core Data Entities/Relationships/Indexing

ENTITIES

E Author

E Book

COMPOSITE TYPES

FETCH REQUESTS

CONFIGURATIONS

C Default

Attributes

Attribute	Type
S desc	String
N edition	Integer 16
ID id	UUID
D publishedAt	Date
S title	String

+

—

Relationships

Relationship	Destination	Inverse
O authors	Author	books

+

—

Fetches Properties

Fetches Property	Predicate
------------------	-----------

+

—

Entity

Name

Book

Abstract Entity

Parent Entity

No Parent Entity

Class

Name

Book

Module

Global namespace

Codegen

Class Definition

Constraints

No Content

+

—

Spotlight

Display Name

Expression

User Info

Key	Value
-----	-------

+

—

Versioning

Hash Modifier

Version Hash Modifier

Renaming ID

Renaming Identifier

Outline Style

Add Entity

Add Attribute

Core Data Operations

```
private func performFetchRequest() {  
    let request = Book.fetchRequest()  
    let context = PersistentModelStore.container.viewContext  
    do {  
        let items = try context.fetch(request)  
        self.bookList = items  
    } catch {  
        print(error)  
    }  
}
```

Core Data Operations

```
private func performFetchRequest() {
    let request = Book.fetchRequest()
    let context = PersistentModelStore.container.viewContext
    do {
        let items = try context.fetch(request)
        self.bookList = items
    } catch {
        print(error)
    }
}
```

```
func createNew() throws {
    let context = PersistentModelStore.container.viewContext

    let author = Author(context: context)
    author.id = UUID()
    author.name = "The Author"

    let book = Book(context: context)
    book.id = UUID()
    book.title = "The Book"
    book.desc = "Authorization Book"
    book.edition = 1
    book.publishedAt = Date()
    book.authors = author
    try context.save()
}
```

Today

- Introduction to SwiftData
- Core Data
- SwiftData
- Final Thoughts

SwiftData

Declarative schema

Custom Datastore

@Attribute

#Expression

@Relationship

CloudKit Sync

@Transient

#Predicate

Compound Predicates

Versioning

@Query

#Unique

Relationship Management

Schema migration plan

Graph Management

Change tracking

History

#Index

ModalActor

Documents

SwiftData

```
class Book {  
    var id: UUID  
    var title: String  
    var author: Author  
  
    init(id: UUID = .init(), title: String, author: Author) {  
        self.id = id  
        self.title = title  
        self.author = author  
    }  
}
```

SwiftData

```
@Model
class Book {
    var id: UUID
    var title: String
    var author: Author

    init(id: UUID = .init(), title: String, author: Author) {
        self.id = id
        self.title = title
        self.author = author
    }
}
```

SwiftData

```
@Model
class Book: Identifiable {
    var id: UUID
    var title: String
    var author: Author

    init(id: UUID = .init(), title: String, author: Author) {
        self.id = id
        self.title = title
        self.author = author
    }
}
```

SwiftData

```
@Model
class Book: Identifiable {
    var id: UUID
    var title: String
    var author: Author

    init(id: UUID = .init(), title: String, author: Author) {
        self.id = id
        self.title = title
        self.author = author
    }
}

@Model
class Author: Identifiable {
    var id: UUID
    var name: String

    init(id: UUID, name: String) {
        self.id = id
        self.name = name
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(for: [Book.self, Author.self])
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(container: ModelContainer)
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var sharedModelContainer: ModelContainer = {
        let schema = Schema([Author.self, Book.self])
        let modelConfiguration = ModelConfiguration(schema: schema, isStoredInMemoryOnly: false)
        do {
            return try ModelContainer(for: schema, configurations: [modelConfiguration])
        } catch {
            fatalError("Could not create ModelContainer: \(error)")
        }
    }()

    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(container: ModelContainer)
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var sharedModelContainer: ModelContainer = {
        let schema = Schema([Author.self, Book.self])
        let modelConfiguration = ModelConfiguration(schema: schema, isStoredInMemoryOnly: false)
        do {
            return try ModelContainer(for: schema, configurations: [modelConfiguration])
        } catch {
            fatalError("Could not create ModelContainer: \(error)")
        }
    }()

    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(sharedModelContainer)
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(for: [Author.self, Book.self])
    }
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(for: [Author.self, Book.self])
    }
}

#Preview {
    ContentView()
}
```

SwiftData

```
import SwiftUI
import SwiftData

@main
struct SwiftDataStudyApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .modelContainer(for: [Author.self, Book.self])
    }
}

#Preview {
    ContentView()
        .modelContainer(for: [Book.self, Author.self], inMemory: true)
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    var body: some View {
        List {
        }
    }
}

#Preview {
    ContentView()
        .modelContainer(for: [Book.self, Author.self], inMemory: true)
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    var body: some View {
        List {

        }
    }
}

#Preview {
    ContentView()
        .modelContainer(for: [Book.self, Author.self], inMemory: true)
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {

    var body: some View {
        List {

        }
    }
}

#Preview {
    ContentView()
        .modelContainer(for: [Book.self, Author.self], inMemory: true)
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query private var items: [Book]

    var body: some View {
        List {

        }
    }
}

#Preview {
    ContentView()
        .modelContainer(for: [Book.self, Author.self], inMemory: true)
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query private var items: [Book]

    var body: some View {
        List {
            ForEach(items) { item in
                Text(item.title)
            }
        }
    }
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    @Environment(\.modelContext) private var modelContext
    @Query private var items: [Book]

    var body: some View {
        List {
            ForEach(items) { item in
                Text(item.title)
            }
        }
    }

    private func addItem() {
        withAnimation {
            let newItem = Book(title: "New Book", author: Author(name: "A Author"))
            modelContext.insert(newItem)
        }
    }
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    @Environment(\.modelContext) private var modelContext
    @Query private var items: [Book]

    var body: some View {
        List {
            ForEach(items) { item in
                Text(item.title)
            }
        }
    }

    private func deleteItems(offsets: IndexSet) {
        withAnimation {
            for index in offsets {
                modelContext.delete(items[index])
            }
        }
    }
}
```

SwiftData View

```
import SwiftUI
import SwiftData

struct ContentView: View {
    @Environment(\.modelContext) private var modelContext
    @Query private var items: [Book]

    var body: some View {
        List {
            ForEach(items) { item in
                Text(item.title)
            }
            .onDelete(perform: deleteItems)
        }
    }

    private func deleteItems(offsets: IndexSet) {
        withAnimation {
            for index in offsets {
                modelContext.delete(items[index])
            }
        }
    }
}
```

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

Schema Macro

#Unique

#Index

@Query

#Predicates

#Expression

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

```
@Attribute(.unique, .spotlight)
```

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

.spotlight

@Query

#Predicates

#Expression

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

.spotlight

@Query

#Predicates

#Expression

→ value is unique

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:) → b/n in-memory form and a persisted form

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

.spotlight

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

→ binary data beside model storage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

.spotlight

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption → Stored in encrypted form

.preserveValueOnDeletion

.ephemeral

.spotlight

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

→ persistent history when deleted

.ephemeral

.spotlight

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

→ Track changes but not stored

.spotlight

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

Schema.Attribute.Option

.unique

.transformable(by:)

.externalStorage

.allowsCloudEncryption

.preserveValueOnDeletion

.ephemeral

.spotlight

Indexes for spotlight search

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Relations with other models

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Track changes but not persist/store

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Avoids duplicate models

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Speed up queries

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Query using filter, sort &/or fetchDescription

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Rich & Optimised queries

SwiftData Macros

@Model

@Attribute

@Relationship

@Transient

#Unique

#Index

@Query

#Predicates

#Expression

- Express complex queries

Today

- Introduction to SwiftData
- Core Data
- SwiftData
- Final Thoughts

Final Thoughts

- Start using new tools/apis
- Think and ponder about why it was introduced
- Practice with study/demo projects
- Search for answers
- Ask for help
- Build it to learn it.

References (Highly Recommended)

- Pragmatic Core Data : by Florian Kugler (objc.io)

- <https://www.youtube.com/watch?v=pOSm-SDZI7A&t=279s>

- What's new in SwiftData: by Rishi Verma (WWDC 24: Apple)

- <https://www.youtube.com/watch?v=-lMOuvRQilc>

- Migrate to SwiftData: by Luvena Huo (WWDC 23: Apple)

- <https://www.youtube.com/watch?v=olsjgo4Qb4A>

- Migrate to SwiftData: by Paul Hudson (HackingWithSwift.com)

- <https://www.hackingwithswift.com/quick-start/swiftdata/how-to-migrate-an-app-from-core-data-to-swiftdata>

This was a brief Introduction to SwiftData

Reena Thakkar

@ Swift Ahmedabad on Saturday 21st December 2024

Thank you