App Frameworks

#WWDC17

What’s New in Core Data?

Persisting since 2004

Session 210

Melissa Turner, Core Data Engineer
Rishi Verma, Core Data Engineer

Roadmap

Core Spotlight integration

New indexing APIs

Persistent history tracking

Alphonse Says “Hi”

CoreSpotlight

What is Spotlight

Original search API on macOS

Document-centric

- Search results identify document
- Not location in document

Not ideal for databases

What is Core Spotlight

Original search tool on iOS

Database-friendly Spotlight API

Search for content in apps

Allows deep linking

What is Core Spotlight

NEW

Original search tool on iOS

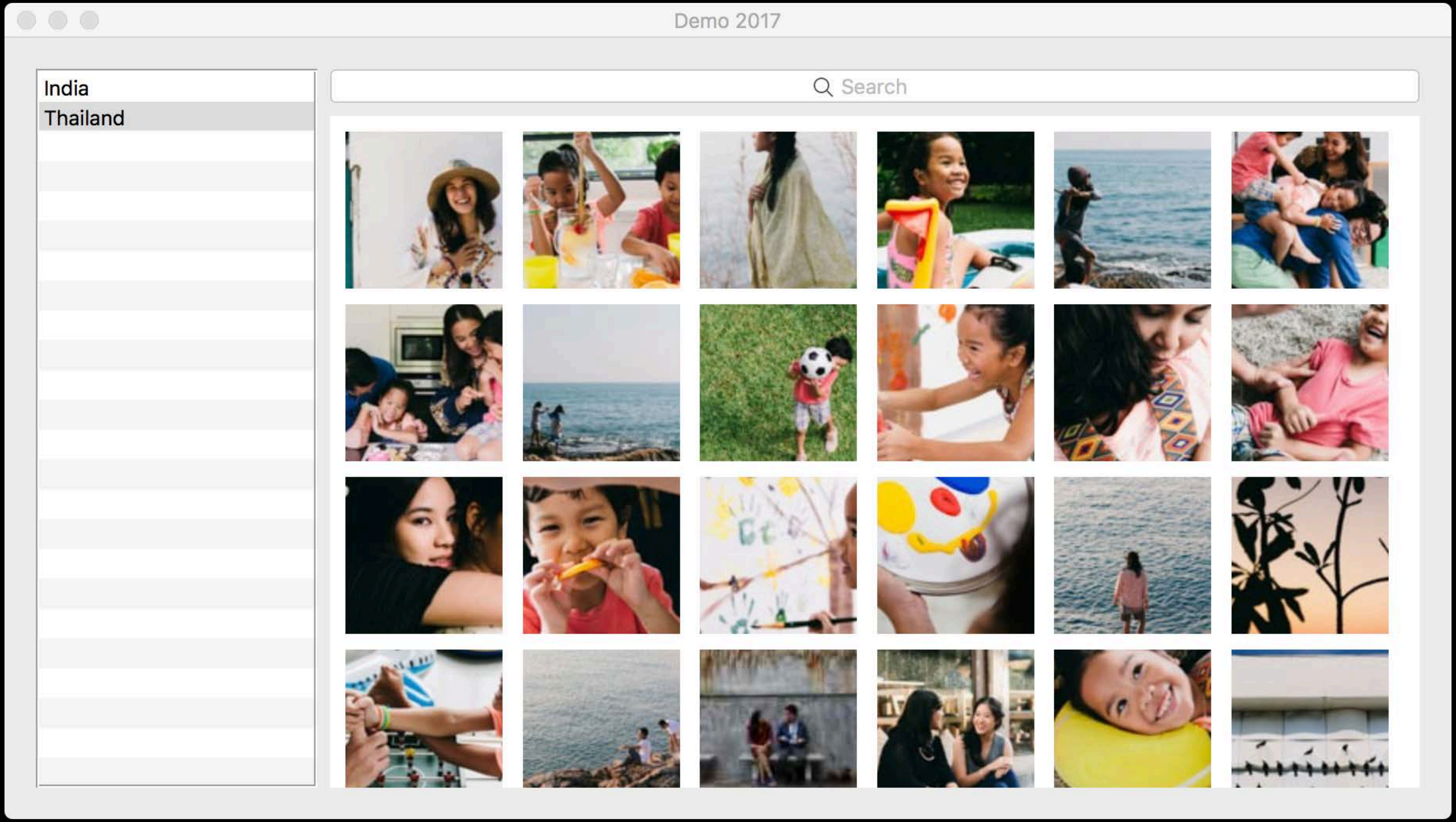
Database-friendly Spotlight API

Search for content in apps

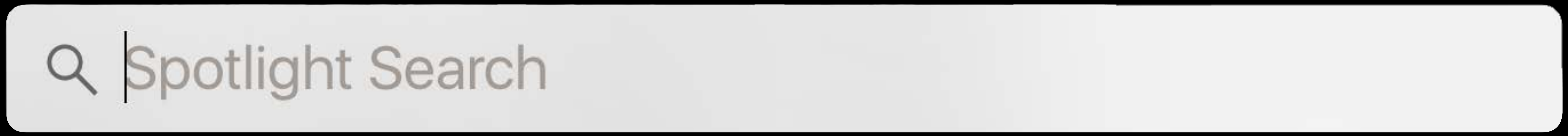
Allows deep linking

Ships on Mac OS and iOS

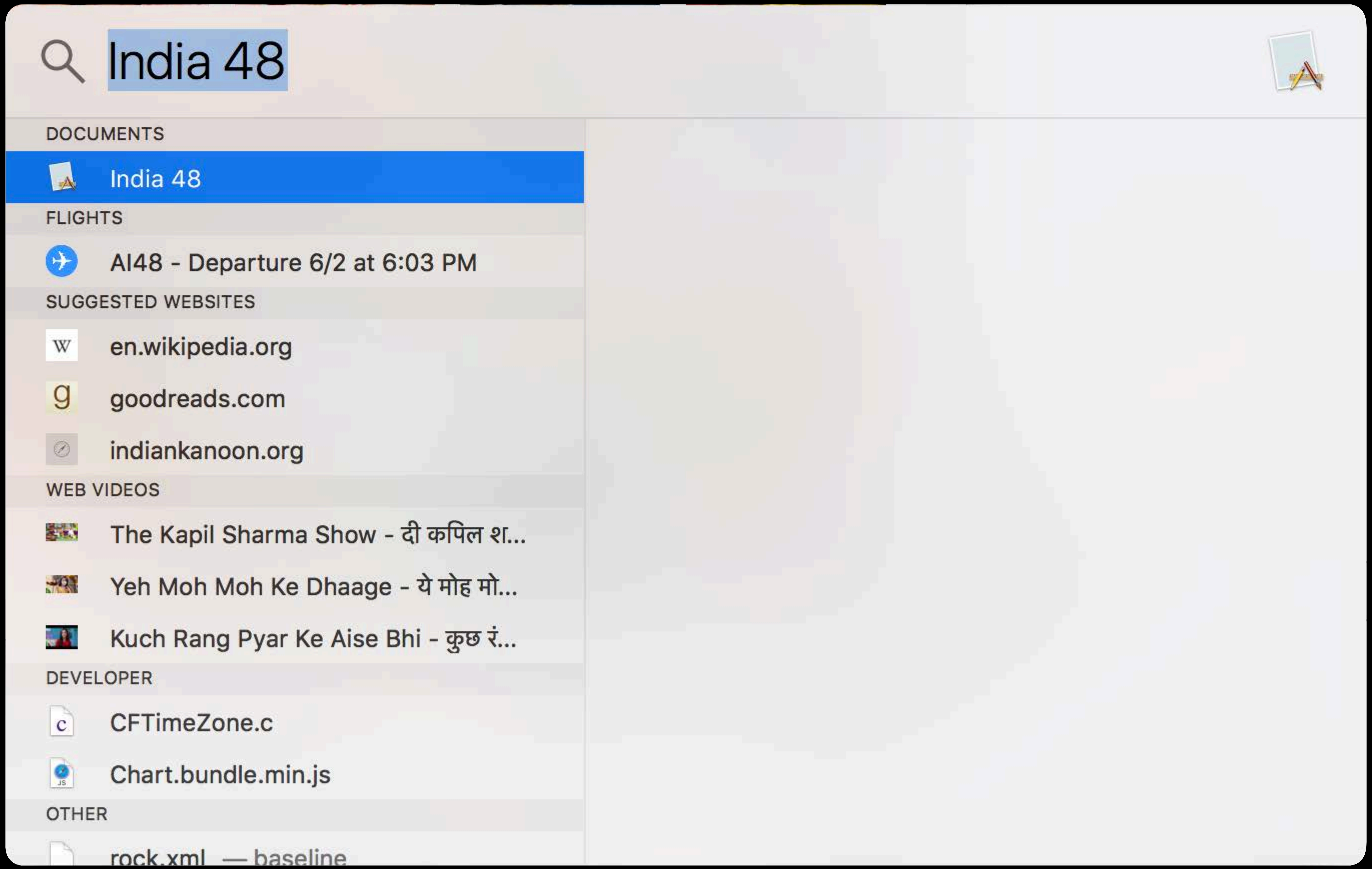
Core Spotlight



Core Spotlight



Core Spotlight



India 48

DOCUMENTS

India 48

FLIGHTS

AI48 - Departure 6/2 at 6:03 PM

SUGGESTED WEBSITES

en.wikipedia.org

goodreads.com

indiankanoon.org

WEB VIDEOS

The Kapil Sharma Show - दी कपिल श...

Yeh Moh Moh Ke Dhaage - ये मोह मो...

Kuch Rang Pyar Ke Aise Bhi - कुछ रं...

DEVELOPER

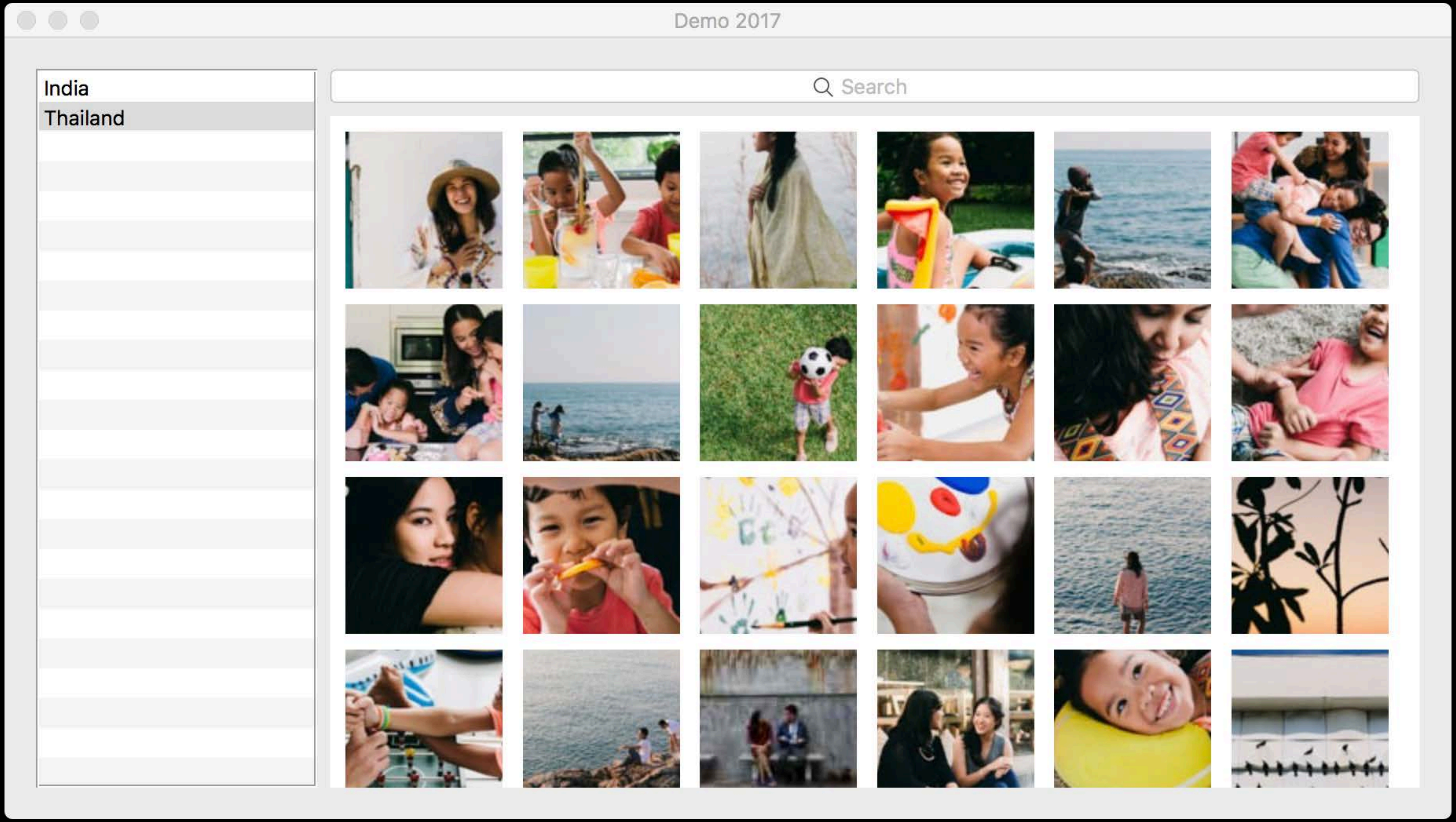
CFTimeZone.c

Chart.bundle.min.js

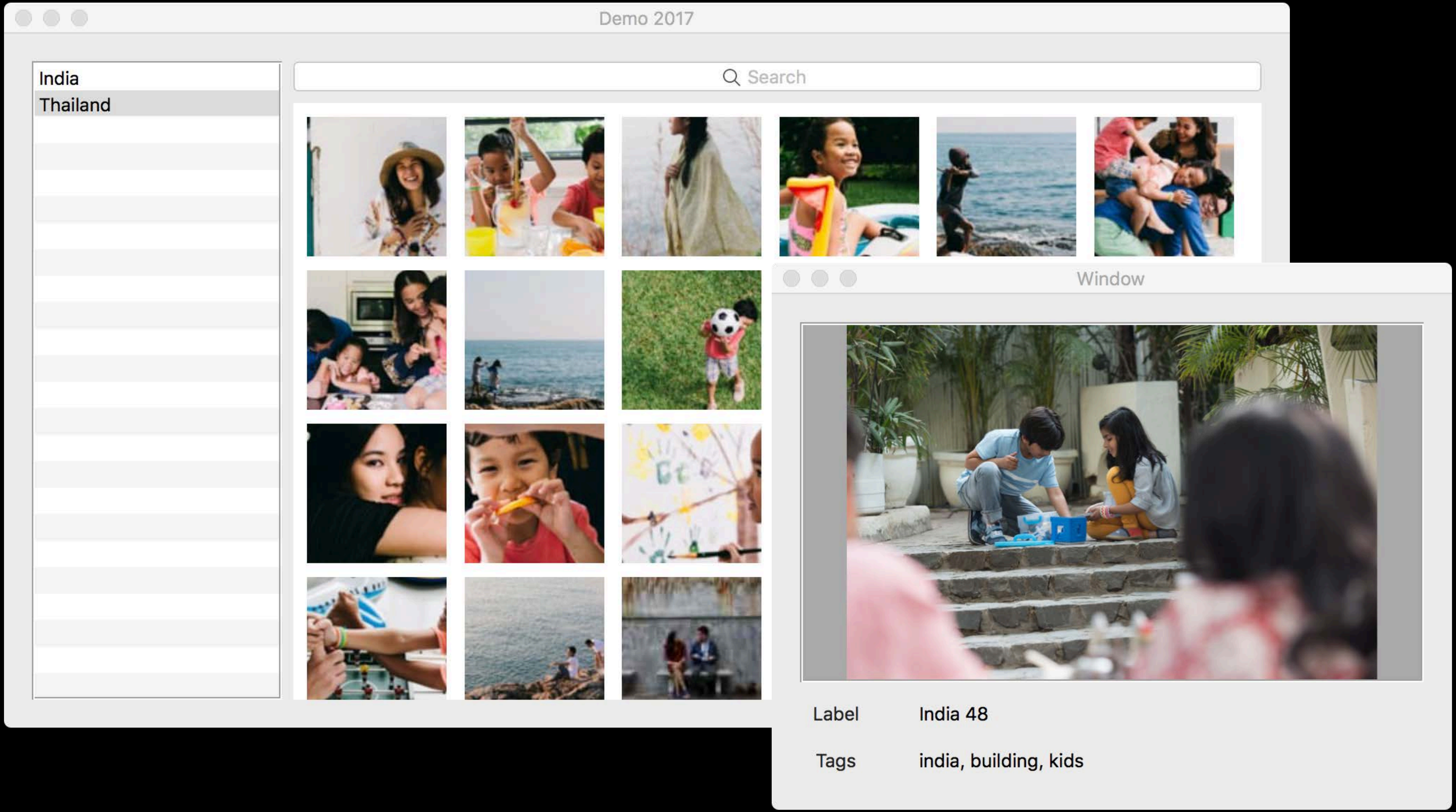
OTHER

rock.xml — baseline

Core Spotlight



Core Spotlight



Label India 48

Tags india, building, kids

What is Core Spotlight

Original search tool on iOS

Database-friendly Spotlight API

Search for content in apps

Allows deep linking

Ships on Mac OS and iOS

What is Core Spotlight

Original search tool on iOS

Database-friendly Spotlight API

Search for content in apps

Allows deep linking

Ships on Mac OS and iOS

Access CoreSpotlight search data from your app

What is Core Spotlight

Original search tool on iOS

Database-friendly Spotlight API

Search for content in apps

Allows deep linking

Ships on Mac OS and iOS

Access CoreSpotlight search data from your app

No zero-length files

Three Pieces

Way to indicate property participation

```
NSPropertyDescription.indexedBySpotlight
```

Way to specify display name

```
NSEntityDescription.coreSpotlightDisplayNameExpression
```

Exporter

`coreSpotlightDisplayNameExpression`

NSExpression

Evaluated during Core Spotlight update

`object` parameter will be an instance of NSManagedObject

Can return anything Core Spotlight can accept

- CSLocalizedString

coreSpotlightDisplayNameExpression

Keypath

- Property name

Function

```
lowercase:(displayName)
```

```
FUNCTION:(castTo:(@“Class”, @“MyUtilityClass”), @“displayNameFor:”, self))
```

NSCoreDataCoreSpotlightDelegate

Implements `CSSearchableIndexDelegate`

- Default implementations of all methods

Uses a separate store on a background thread

- Initialize with `NSPersistentStoreDescription` + model

Override point is `NSCDCSD.attributeSet(for object:)`

```
override func attributeSet(for object: NSManagedObject) -> CSSearchableItemAttributeSet? {

    guard let attributeSet = super.attributeSet(for: object) else { return nil }

    if "Photo" == object.entity.name  {
        let photo = object as! Photo
        let tags = photo.tags
        let localizedString = localizedTagString(photo)
        attributeSet.setValue(localizedString, forCustomKey: attributeKey!)
    }

    return attributeSet;

}
```

```
override func attributeSet(for object: NSManagedObject) -> CSSearchableItemAttributeSet? {

    guard let attributeSet = super.attributeSet(for: object) else { return nil }

    if "Photo" == object.entity.name  {
        let photo = object as! Photo
        let tags = photo.tags
        let localizedString = localizedTagString(photo)
        attributeSet.setValue(localizedString, forCustomKey: attributeKey!)
    }

    return attributeSet;

}
```

```swift
override func attributeSet(for object: NSManagedObject) -> CSSearchableItemAttributeSet? {

    guard let attributeSet = super.attributeSet(for: object) else { return nil }

    if "Photo" == object.entity.name  {
        let photo = object as! Photo
        let tags = photo.tags
        let localizedString = localizedTagString(photo)
        attributeSet.setValue(localizedString, forCustomKey: attributeKey!)
    }

    return attributeSet;

}
```

```swift
override func attributeSet(for object: NSManagedObject) -> CSSearchableItemAttributeSet? {

    guard let attributeSet = super.attributeSet(for: object) else { return nil }

    if "Photo" == object.entity.name  {
        let photo = object as! Photo
        let tags = photo.tags
        let localizedString = localizedTagString(photo)
        attributeSet.setValue(localizedString, forCustomKey: attributeKey!)
    }

    return attributeSet;

}
```

Demo

Non-Sharp Corners

Will automatically export to Core Spotlight on first launch

Uses Persistent History Tracking to ensure all data is pushed

Sharpish Corners

Core Spotlight and Spotlight integration mutually exclusive

`entity.coreSpotlightDisplayNameExpression` OR

`property.storedInExternalRecord`

Doesn’t track results of batch operations

Sharpish Corners

Entitlement to talk to Core Spotlight

- com.apple.application-identifier

You need to handle the search result

```
public func application(_ application: NSApplication, continue userActivity:
NSUserActivity, restorationHandler: @escaping ([Any]) -> Void) -> Bool
```

```swift
public func application(_ application: NSApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([Any]) -> Void) -> Bool {

  let userInfo = userActivity.userInfo as! NSDictionary

  let identifier = userInfo.value(forKey: “kCSSearchableItemActivityIdentifier”)

  let uri = URL(string: identifier as! String)
  let coordinator = self.persistentContainer.persistentStoreCoordinator
  let managedObjectID = managedObjectID(forURIRepresentation: uri!)

  // insert your code here
}
```

```
public func application(_ application: NSApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([Any]) -> Void) -> Bool {

  let userInfo = userActivity.userInfo as! NSDictionary

  let identifier = userInfo.value(forKey: ”kCSSearchableItemActivityIdentifier”)

  let uri = URL(string: identifier as! String)
  let coordinator = self.persistentContainer.persistentStoreCoordinator
  let managedObjectID = managedObjectID(forURIRepresentation: uri!)

  // insert your code here
}
```

```swift
public func application(_ application: NSApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([Any]) -> Void) -> Bool {

  let userInfo = userActivity.userInfo as! NSDictionary

  let identifier = userInfo.value(forKey: “kCSSearchableItemActivityIdentifier”)

  let uri = URL(string: identifier as! String)
  let coordinator = self.persistentContainer.persistentStoreCoordinator
  let managedObjectID = managedObjectID(forURIRepresentation: uri!)

  // insert your code here
}
```

```
public func application(_ application: NSApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([Any]) -> Void) -> Bool {

  let userInfo = userActivity.userInfo as! NSDictionary

  let identifier = userInfo.value(forKey: ”kCSSearchableItemActivityIdentifier”)

  let uri = URL(string: identifier as! String)
  let coordinator = self.persistentContainer.persistentStoreCoordinator
  let managedObjectID = managedObjectID(forURIRepresentation: uri!)

  // insert your code here
}
```

```
public func application(_ application: NSApplication, continue userActivity: NSUserActivity,
restorationHandler: @escaping ([Any]) -> Void) -> Bool {

  let userInfo = userActivity.userInfo as! NSDictionary

  let identifier = userInfo.value(forKey: “kCSSearchableItemActivityIdentifier”)

  let uri = URL(string: identifier as! String)
  let coordinator = self.persistentContainer.persistentStoreCoordinator
  let managedObjectID = managedObjectID(forURIRepresentation: uri!)

  // insert your code here
}
```

Sharp Corners

Up to you to delete old reference files

x86_64 only

New Indexing API

Indexing

Configure database for faster search

One or more columns per index

One or more indexes per column

Database will choose most efficient index based on query

Digression: Indexing

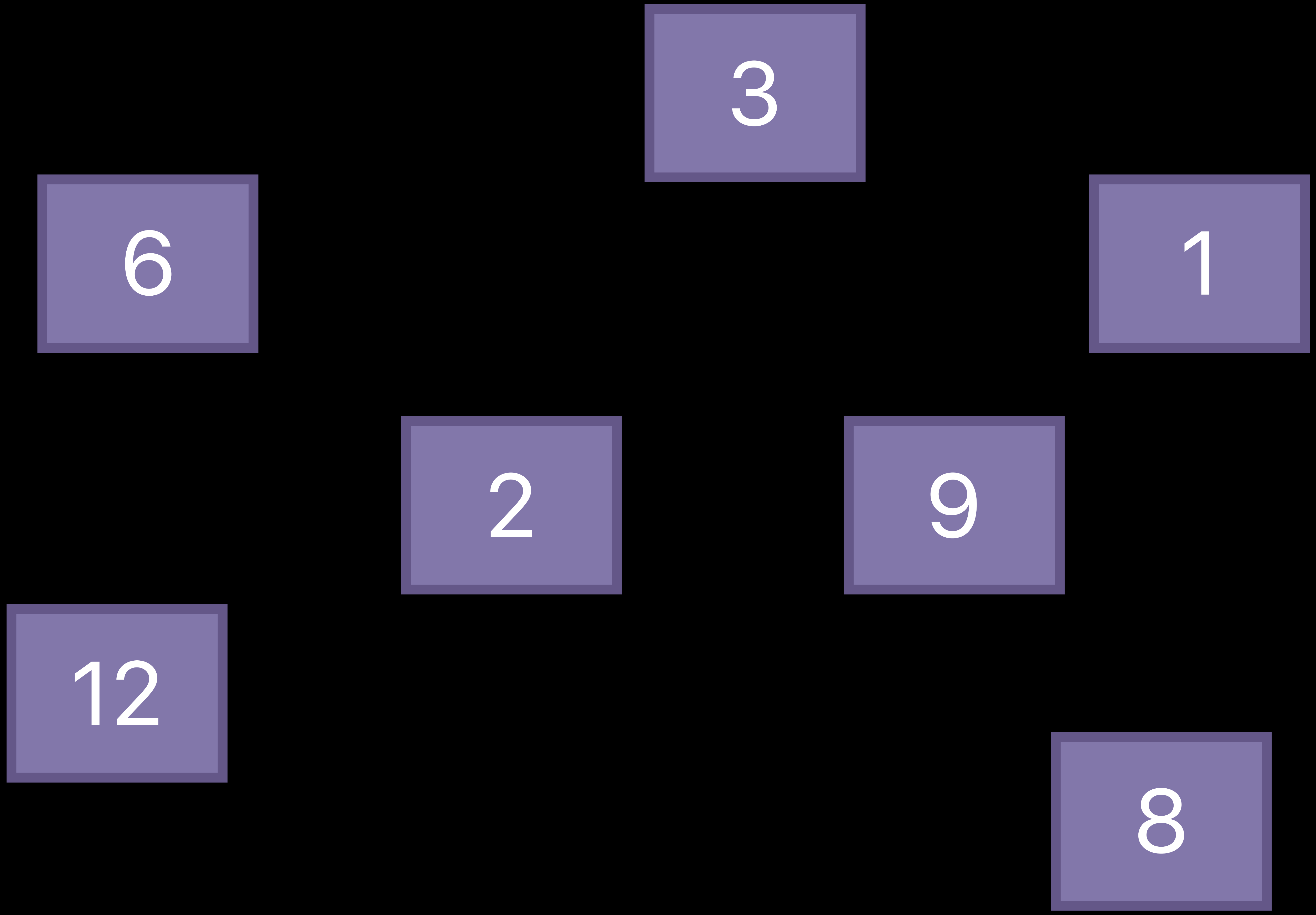
Binary indexes

- Basic form
- Strict comparison only

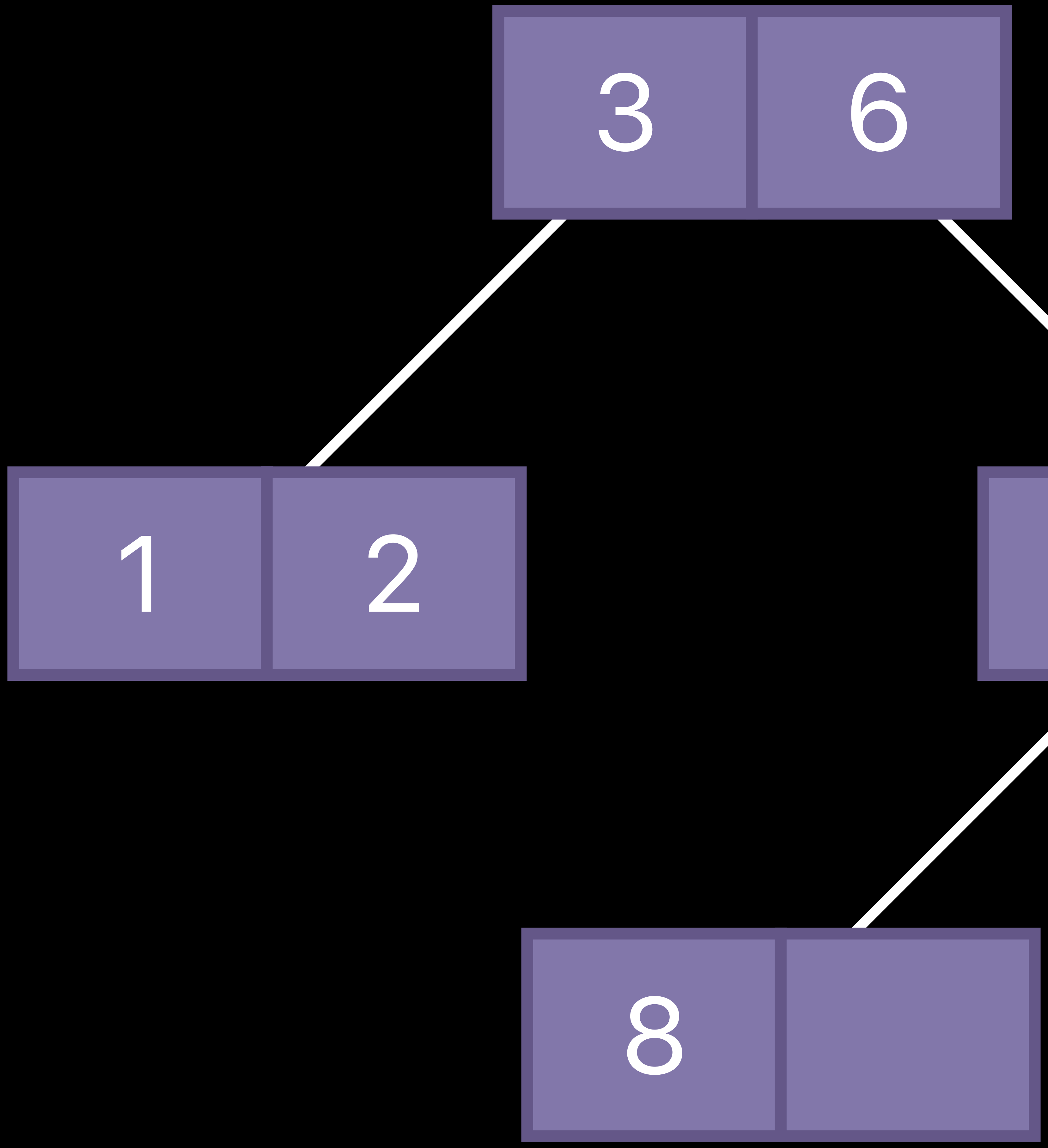
RTree indexes

- Optimized for ranged based searching

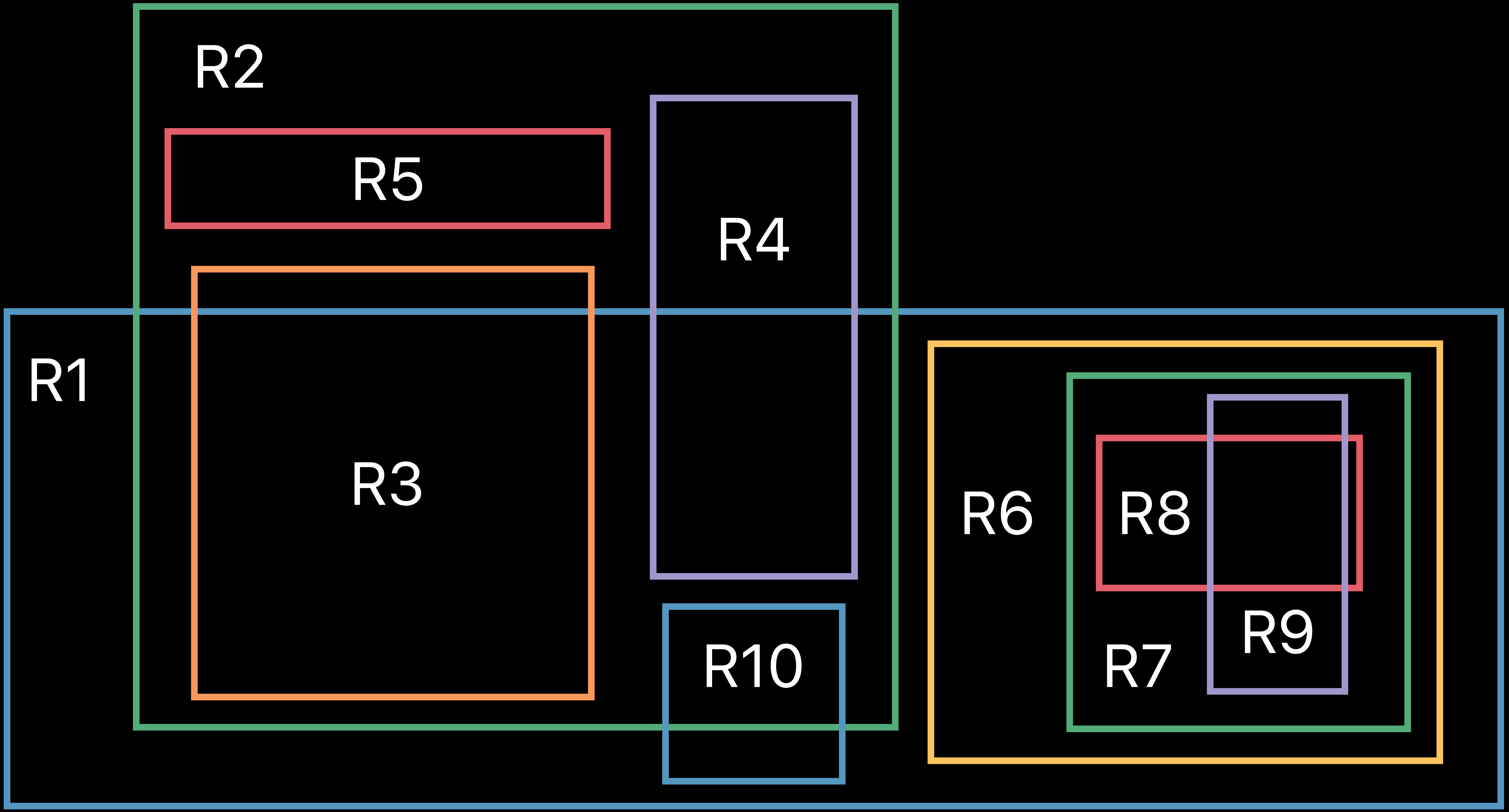
BTree Data



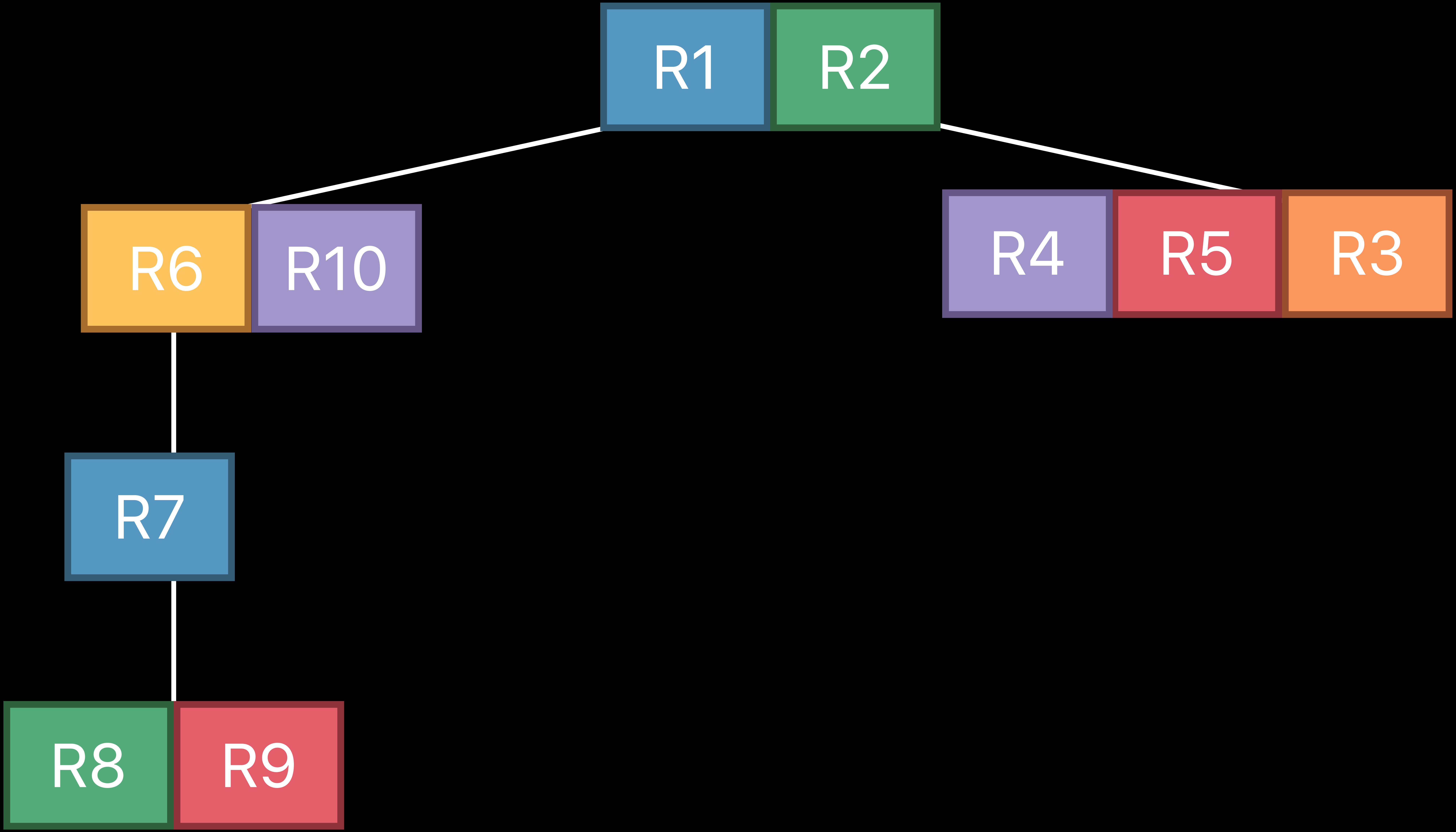
BTree Index



RTree Data



RTree Index



Current Indexing APIs

Single property index

```
NSAttributeDescription.isIndexed
```

Multiple column index

```
NSEntityDescription.compoundIndexes
```

Evolving Core Data Indexing

Ability to support more index types

More configurable indexing

More Powerful API

NSFetchIndexDescription

NSFetchIndexElementDescription

NSEntityDescription.indexes

Functions to control index use

```
indexed:by:(indexName, propertyName)
```

```
noindex:(propertyName)
```

NSFetchIndexElementDescription

Property

Type

- Binary
- RTree

Direction

NSFetchIndexDescription

Specify a single index

- Name
- Elements
- Predicate

Validates index composition

Sharp Corners

Can’t mix and match element types

Range indexes require numeric properties <= 32-bit

- Which can’t be optional

Create indexes last

- Changing entity hierarchy structurally drops indexes

Maybe Sharp Corners?

Doesn’t affect version hash

- Won’t cause migration (with perf hit)
- Won’t cause migration (user won’t get updated indexes)

Update model’s `versionHashModifier` to trigger migration

SQL store will ignore unsupported indexes

Demo

Miscellaneous

New Attribute Types

NSUUIDAttributeType

- NSUUID

NSURIAttributeType

- NSURL

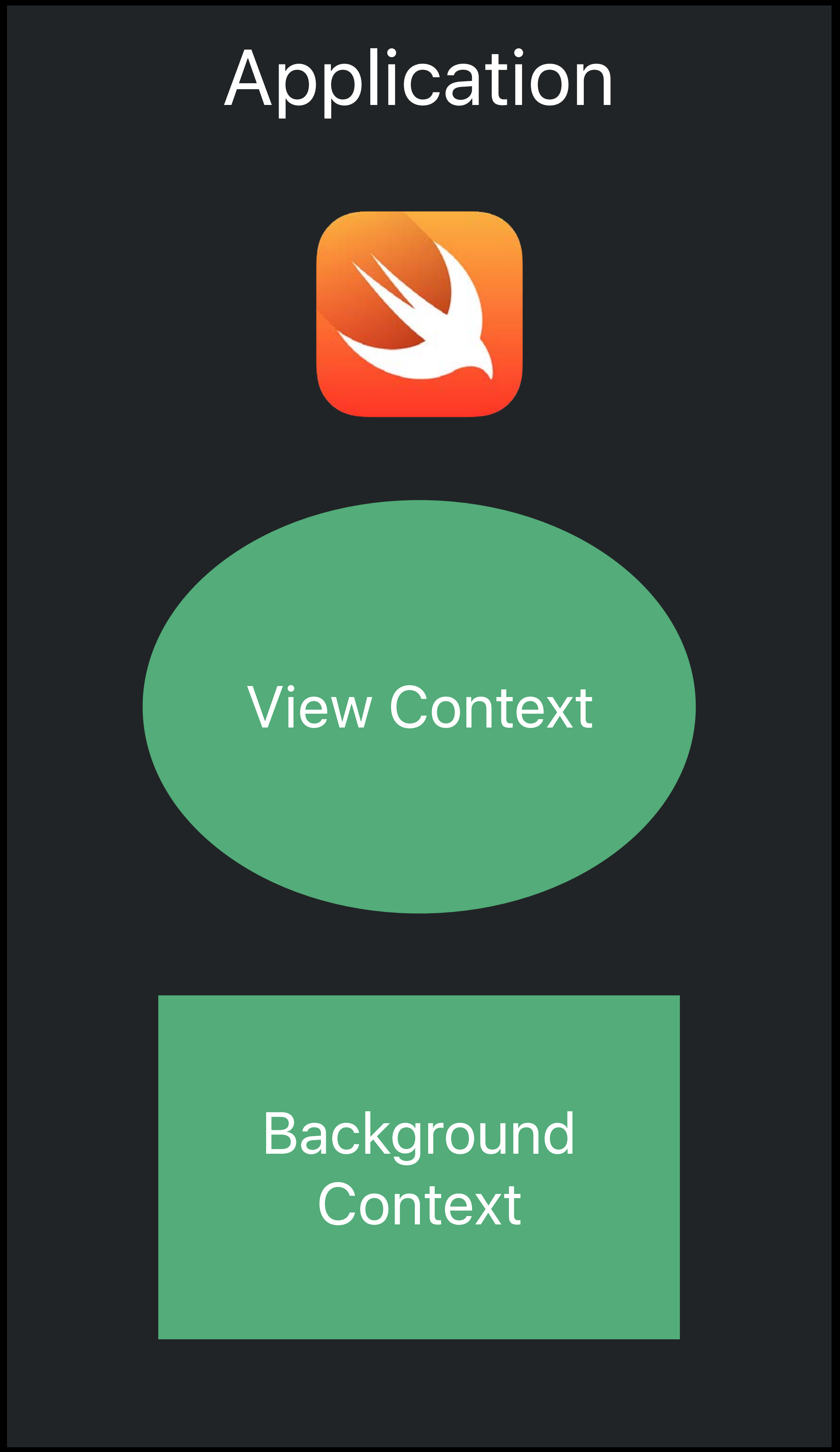
Persistent History Tracking

History 201

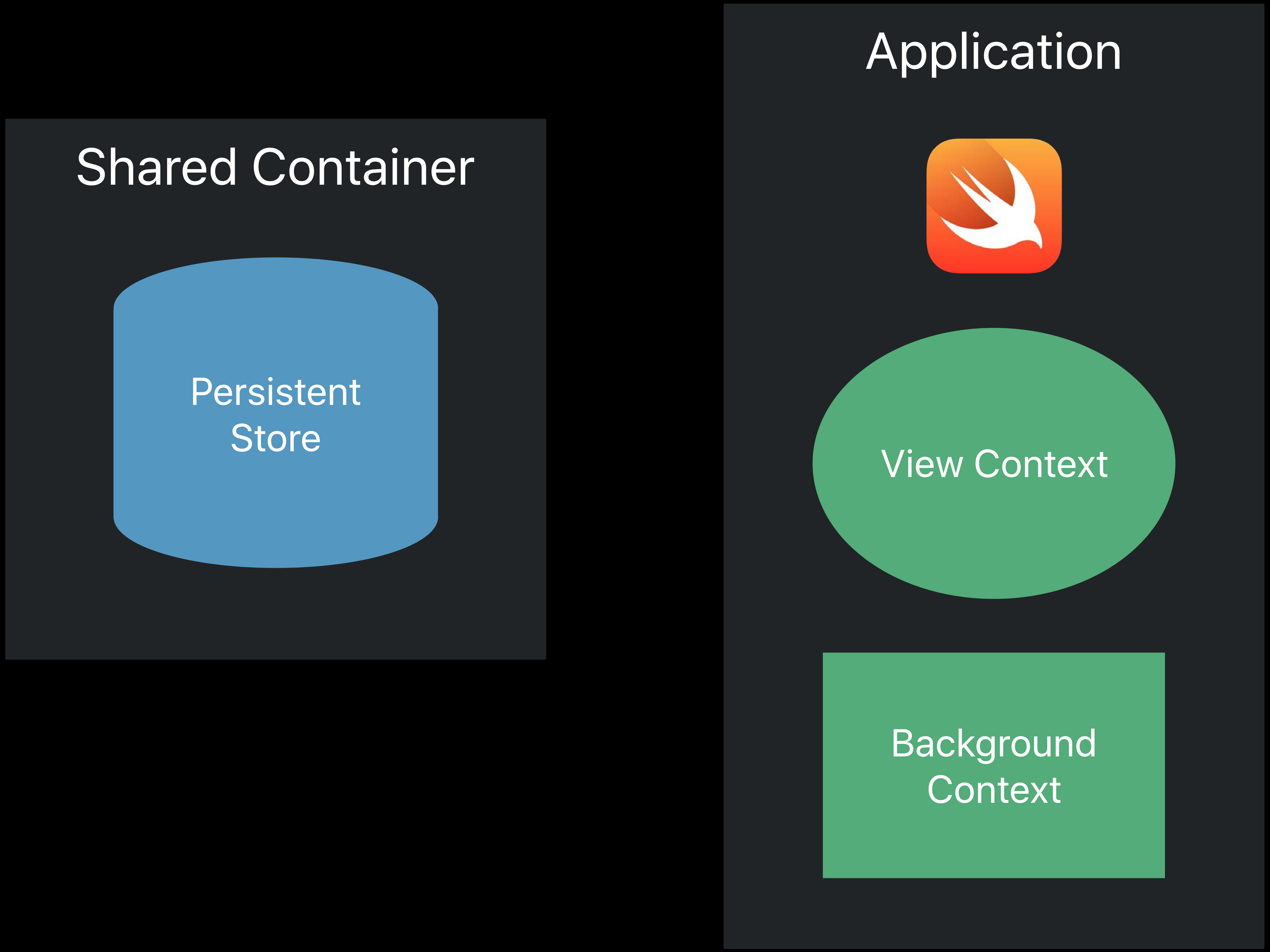
Rishi Verma, Core Data Engineer

Story of the App

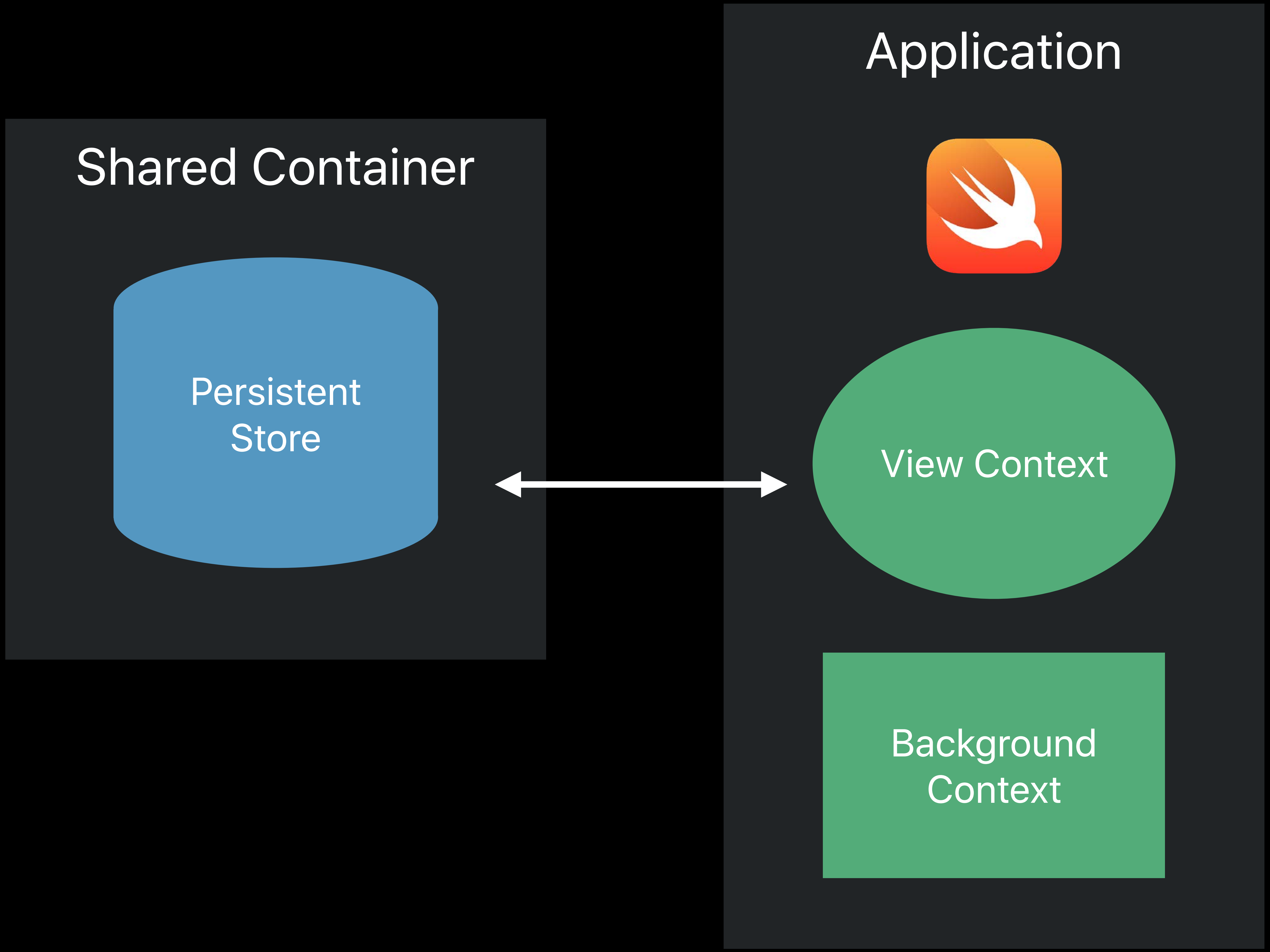
Story of the App



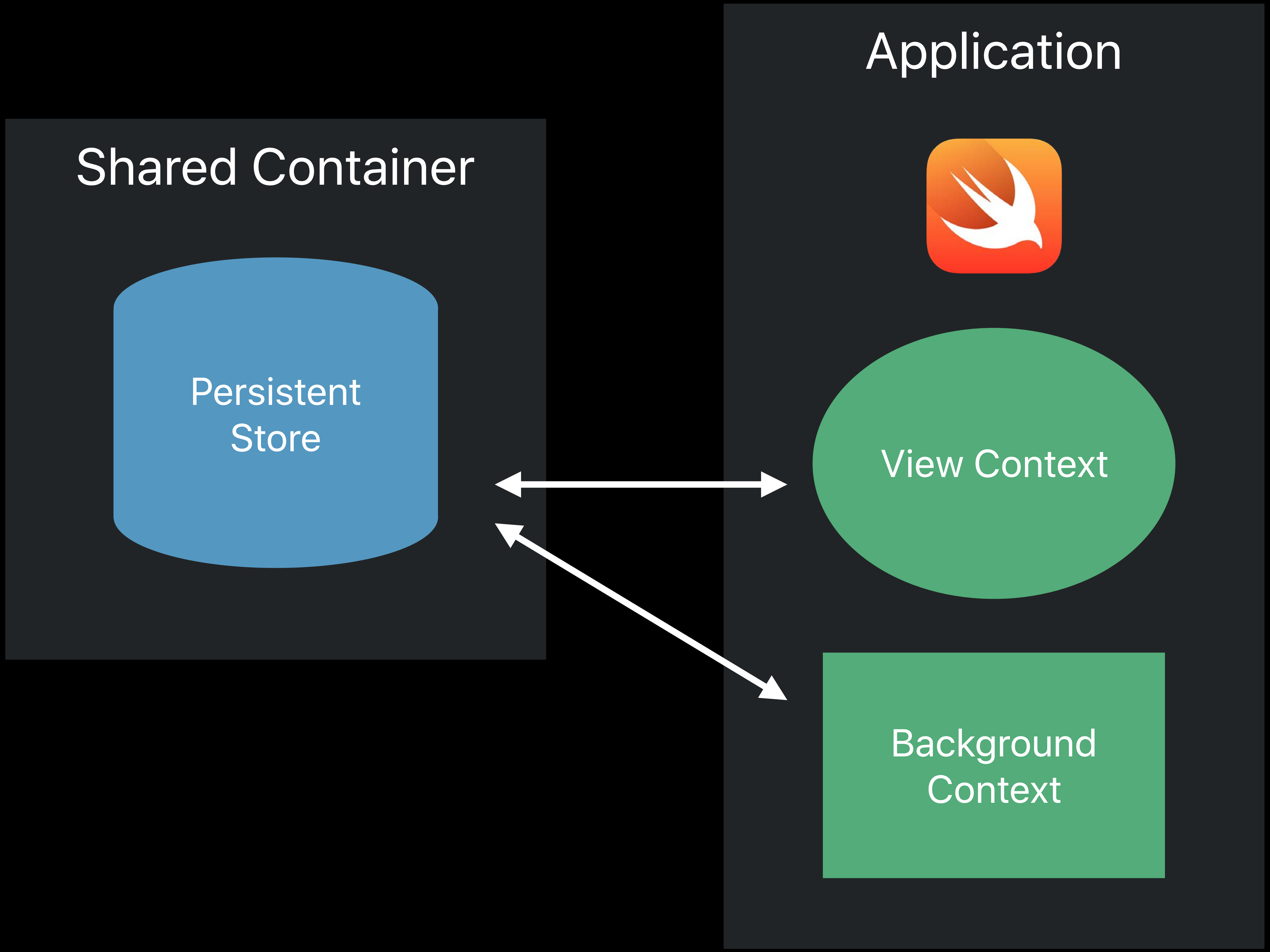
Story of the App



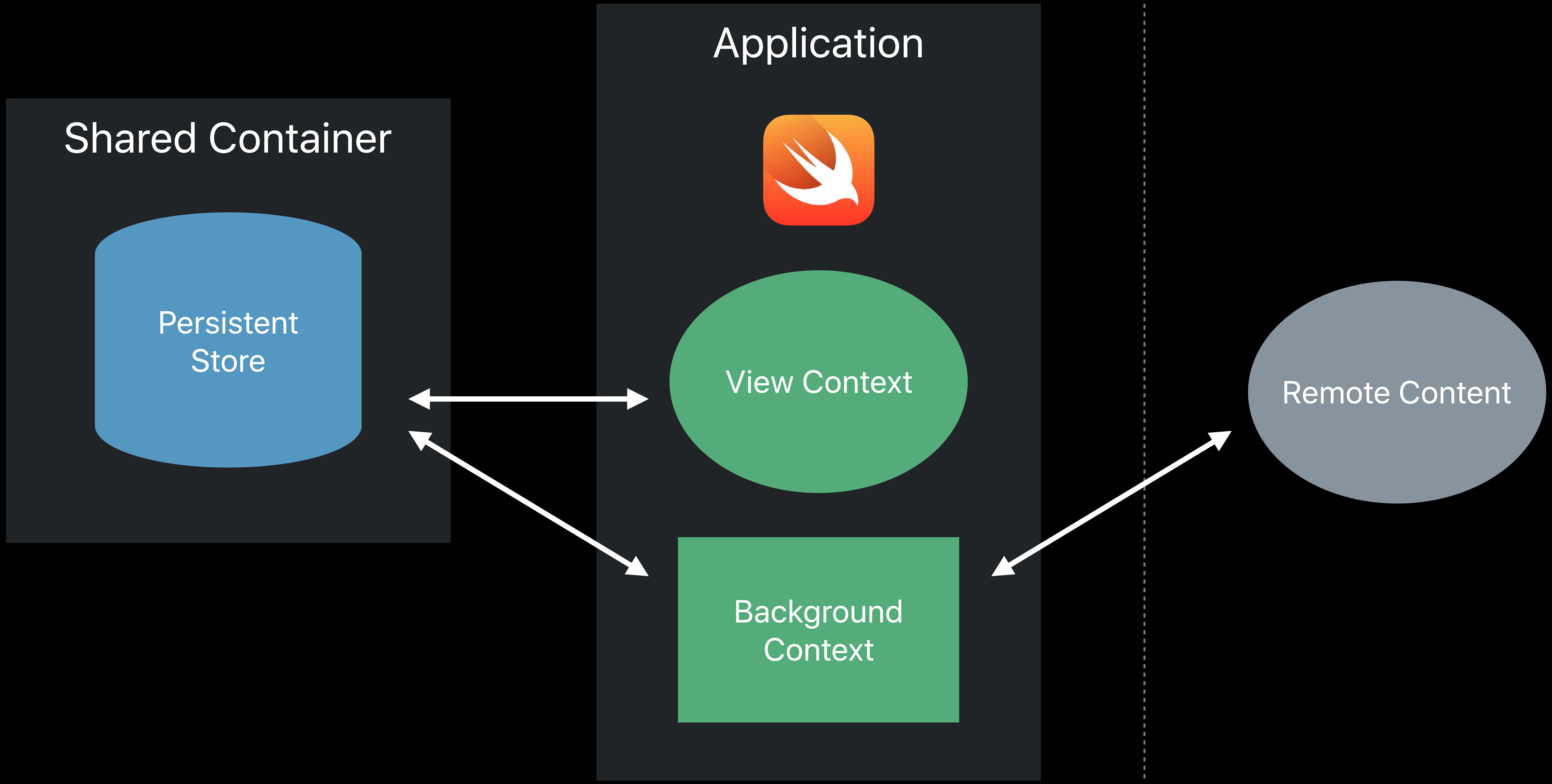
Story of the App



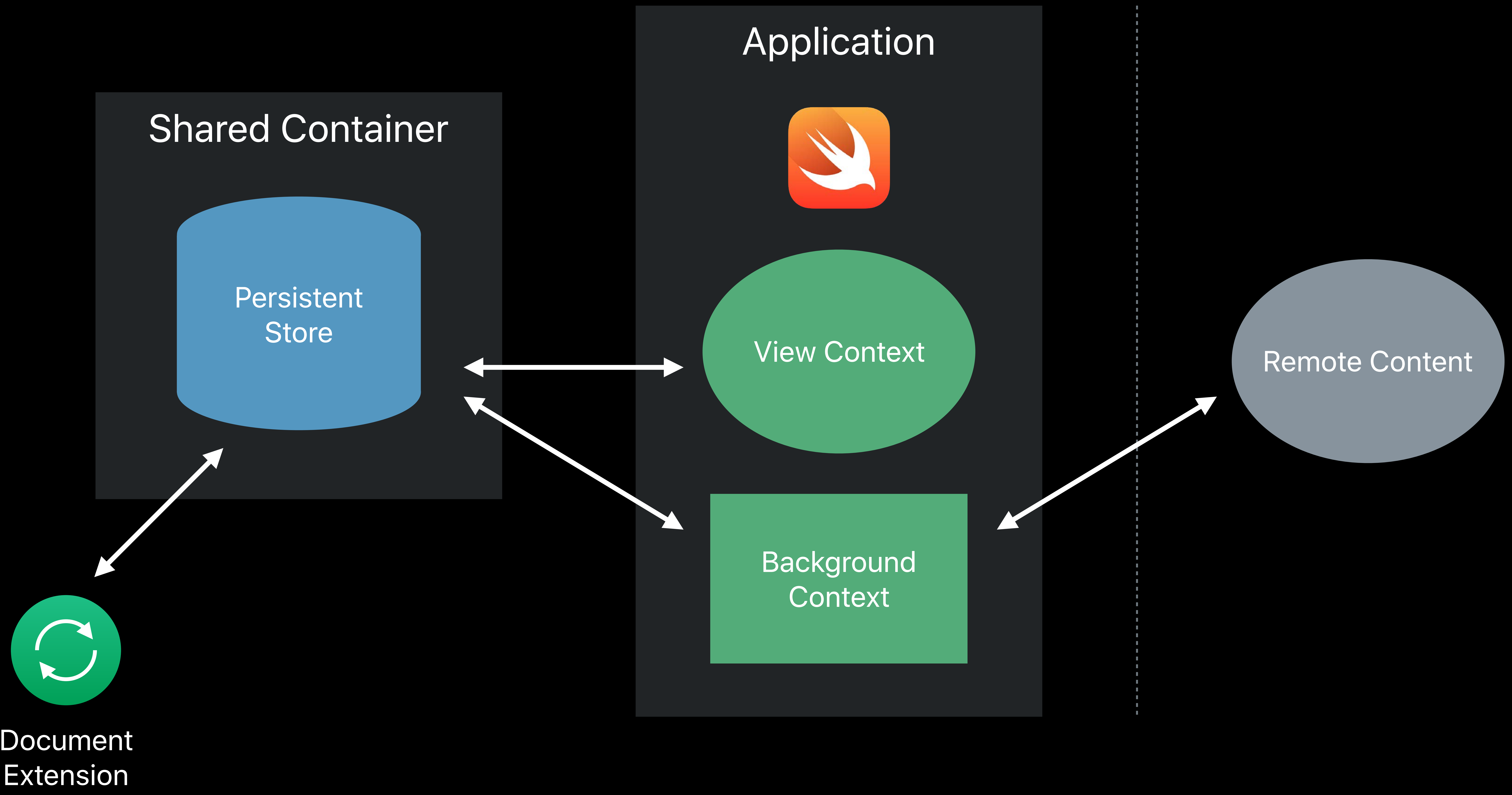
Story of the App



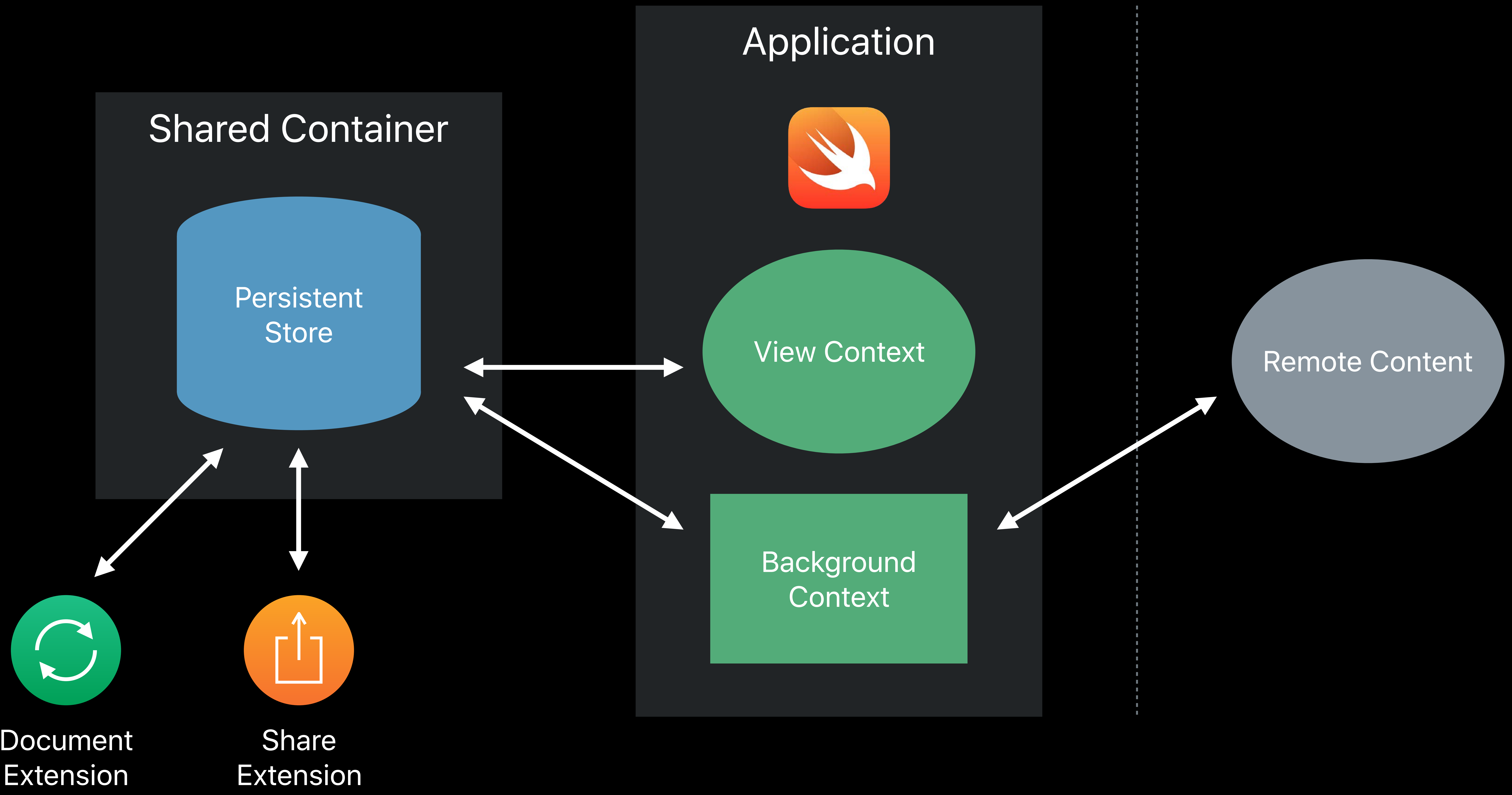
Story of the App



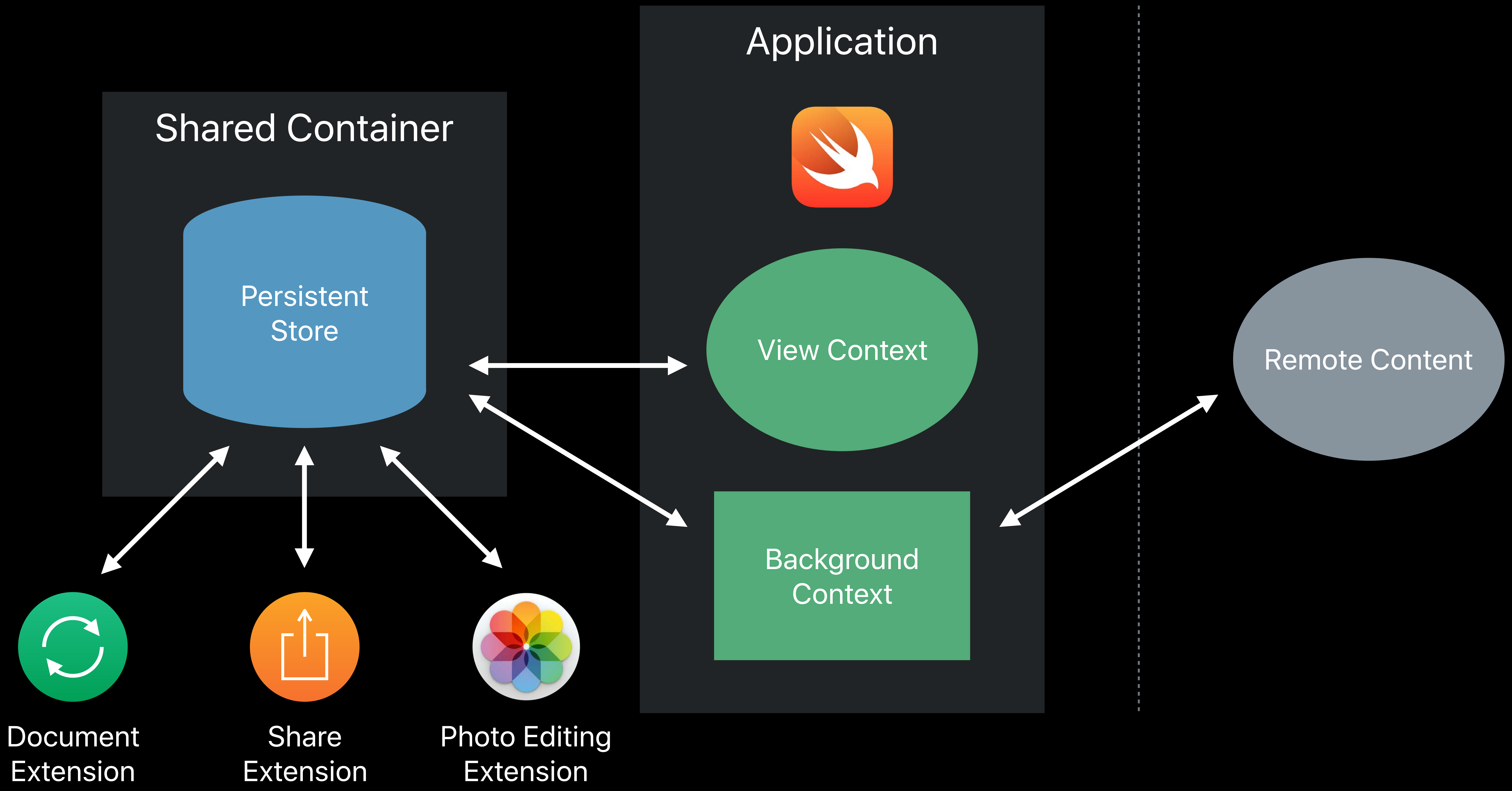
Story of the App



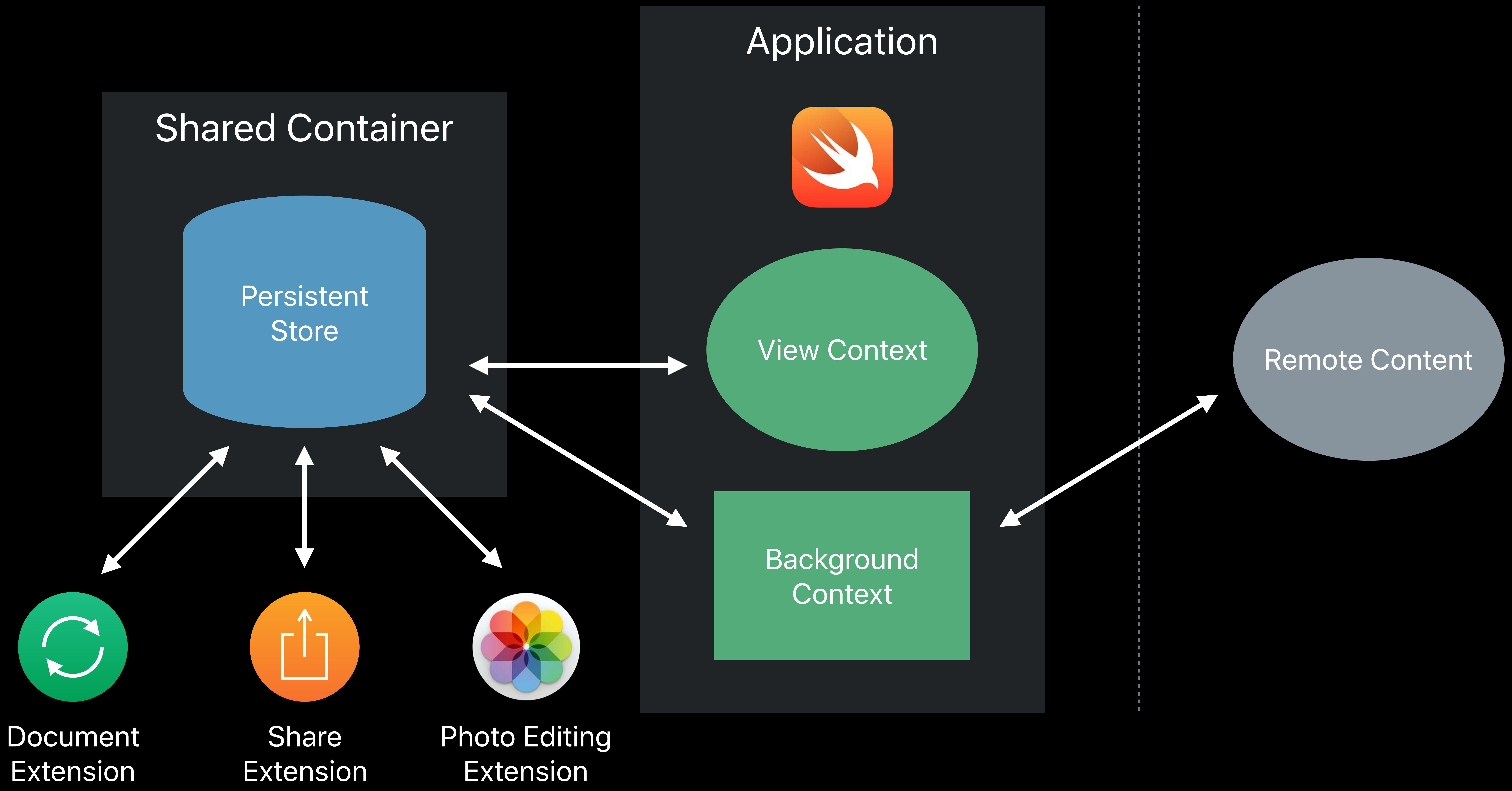
Story of the App



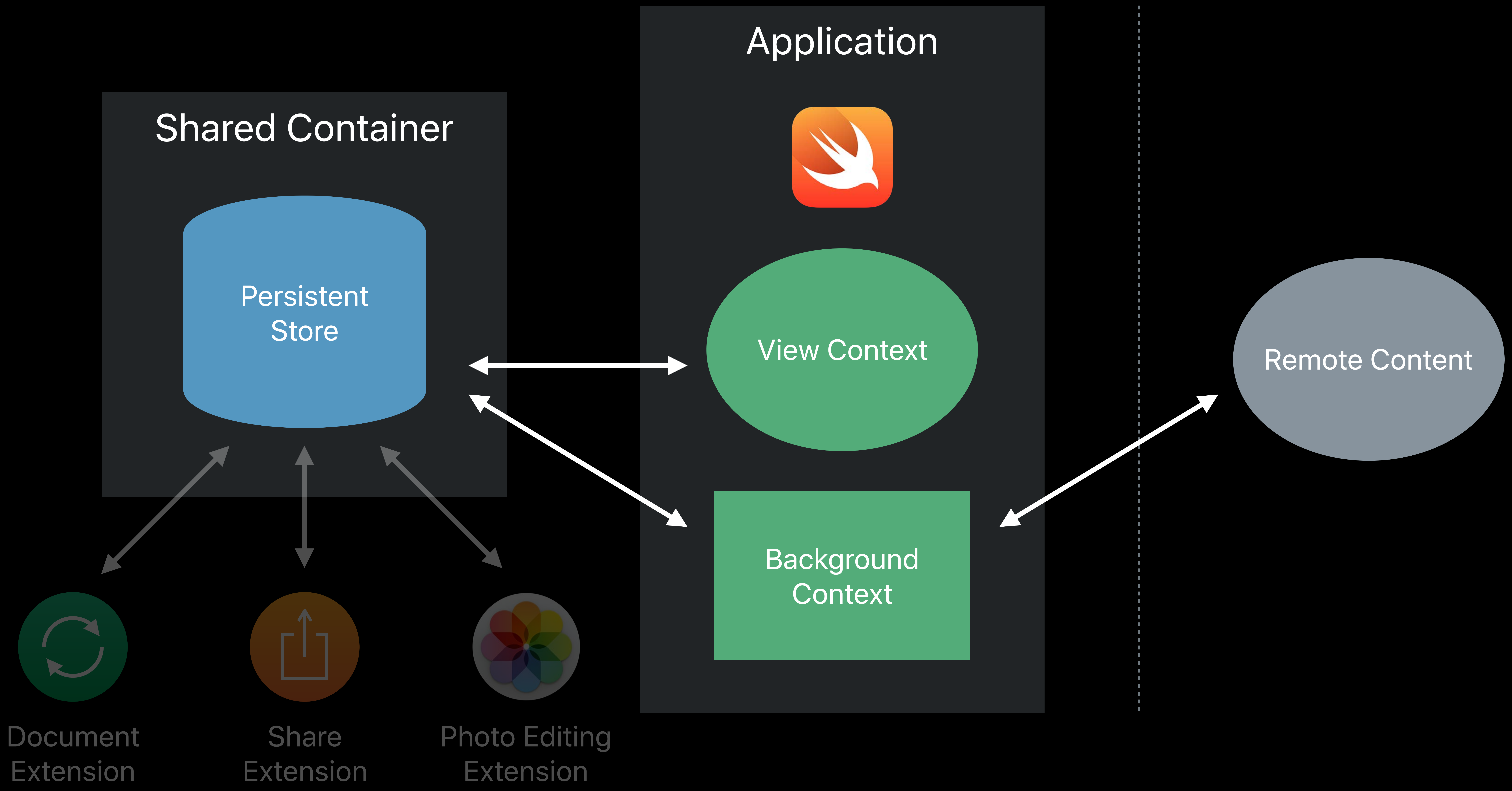
Story of the App



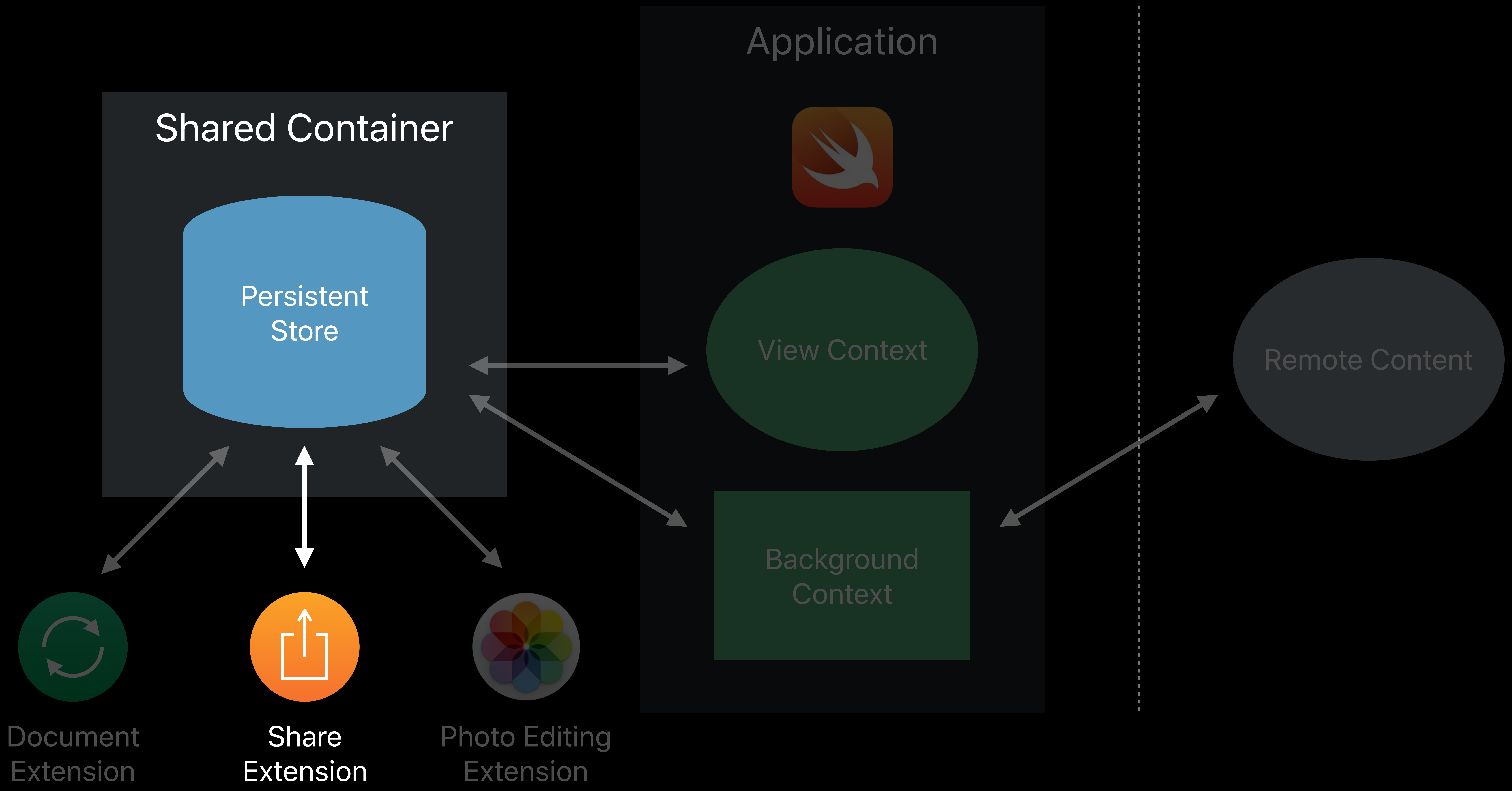
Trains in the Night



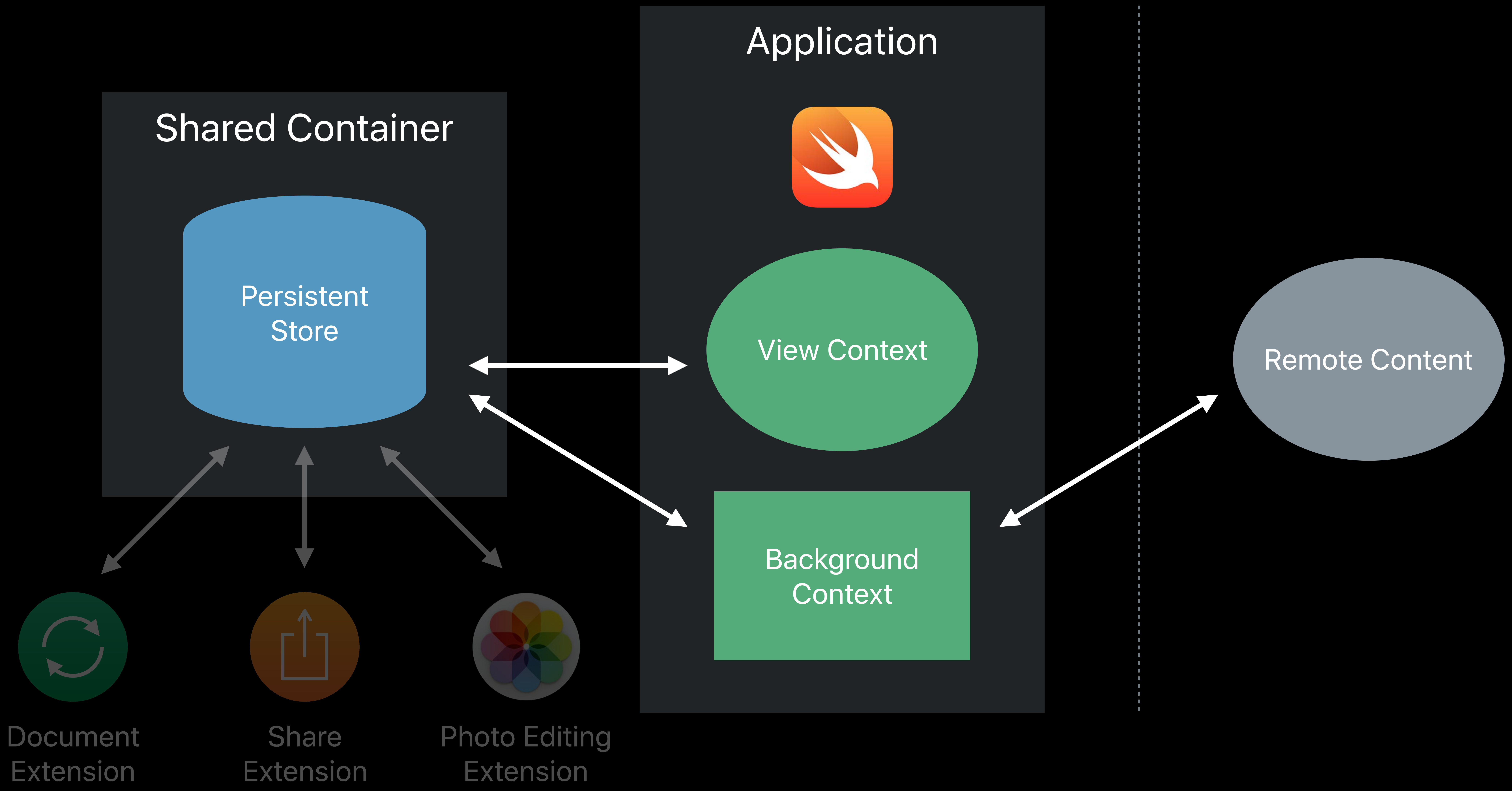
Trains in the Night



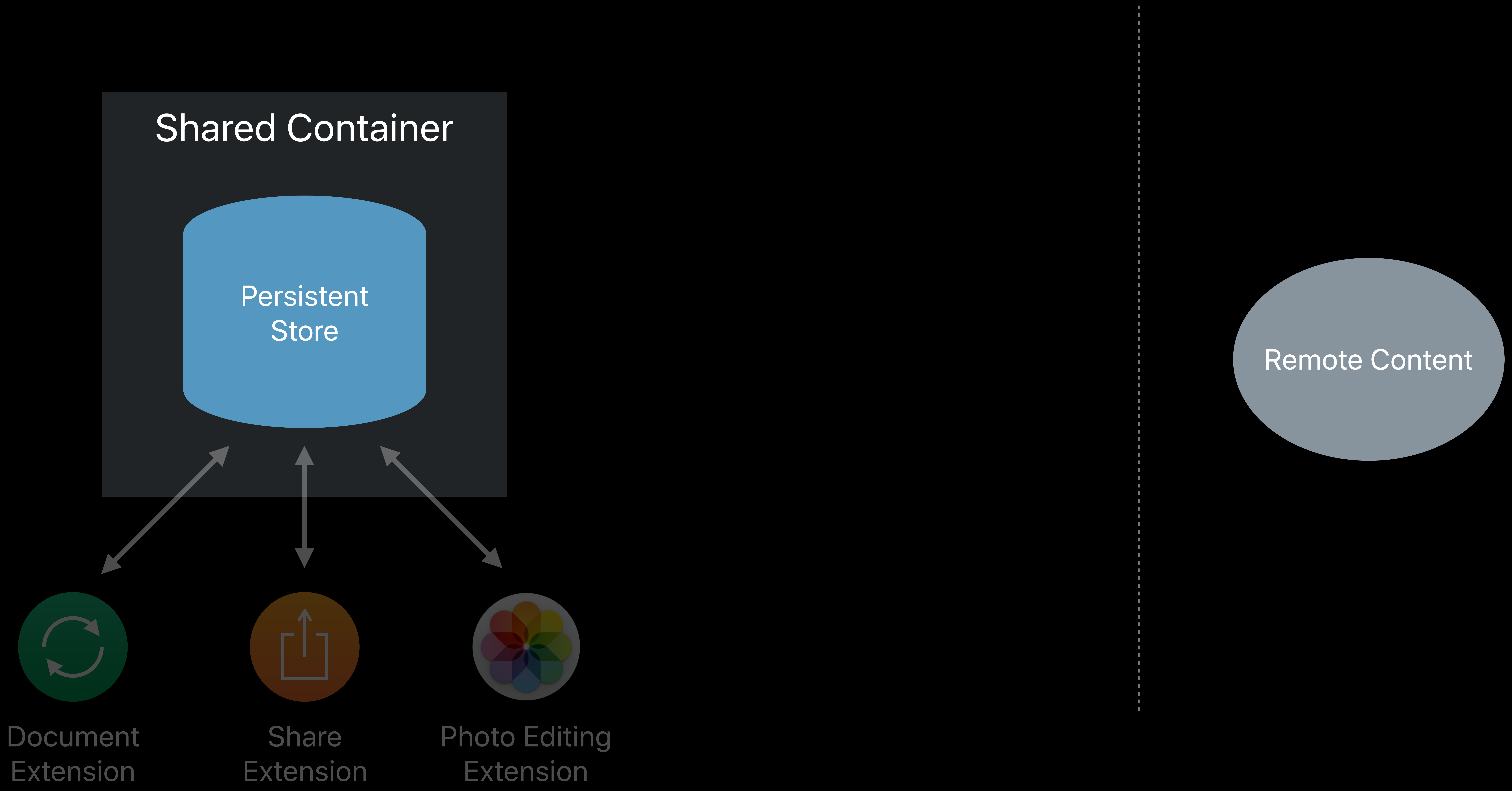
Trains in the Night



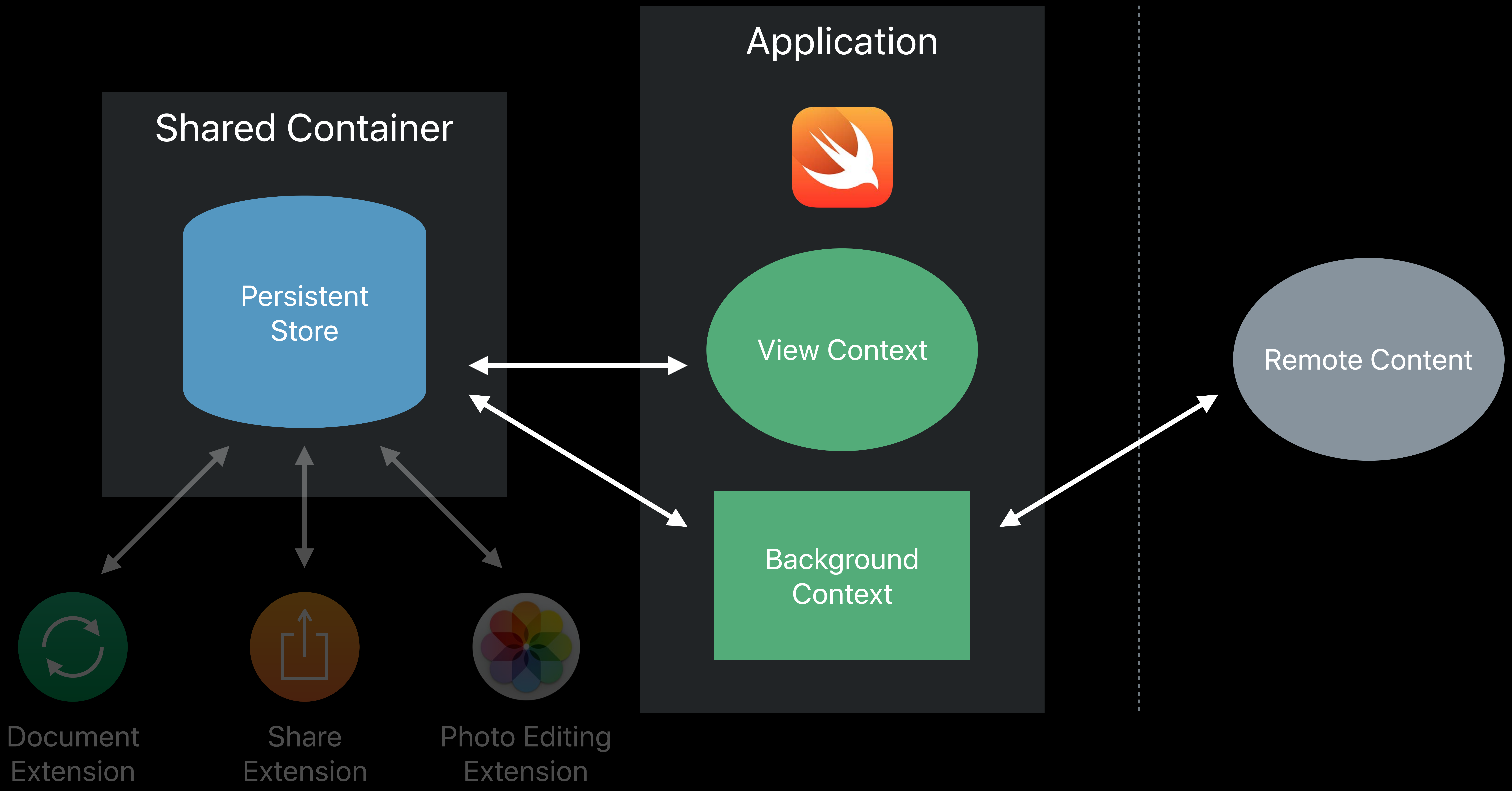
Trains in the Night



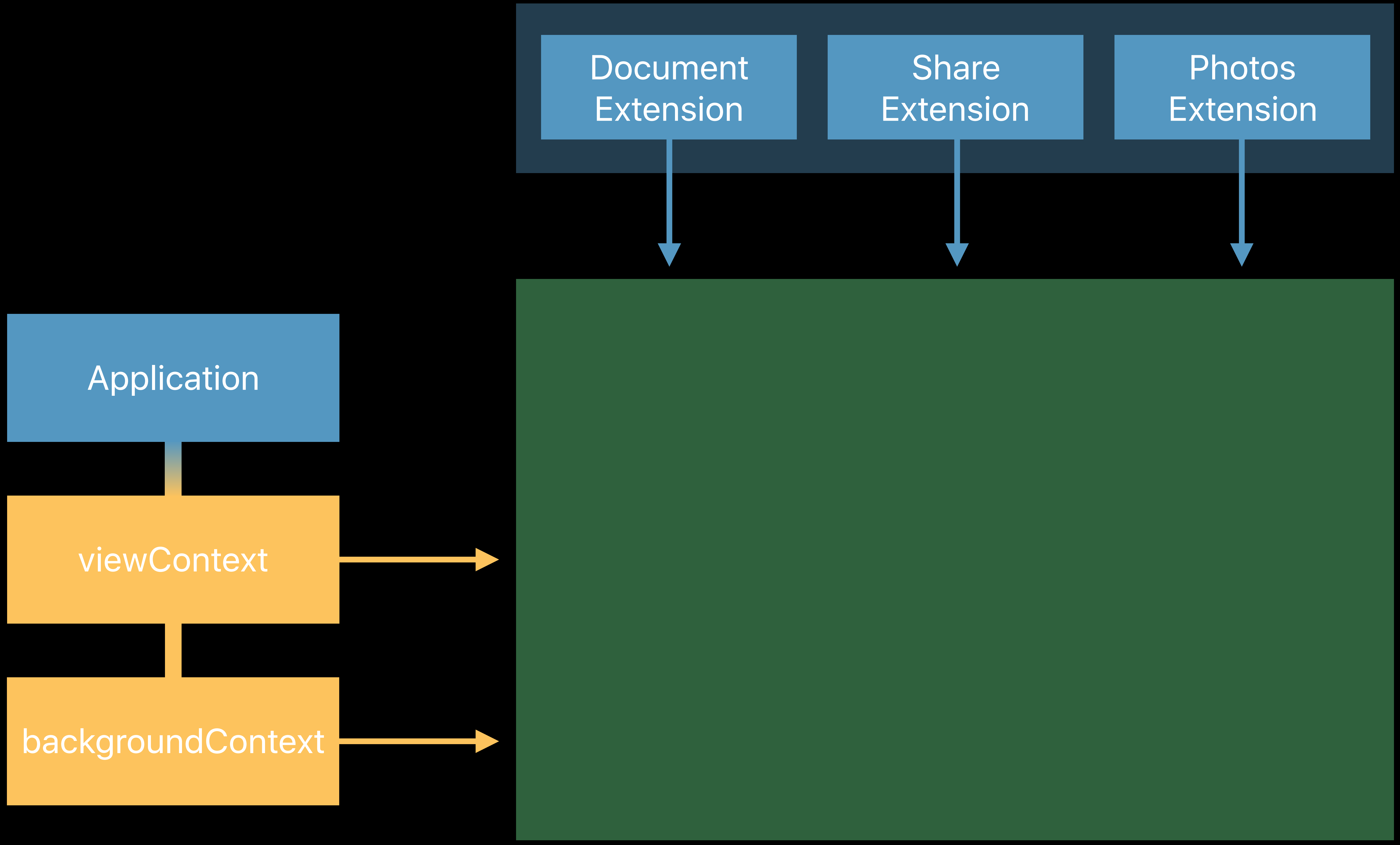
Trains in the Night



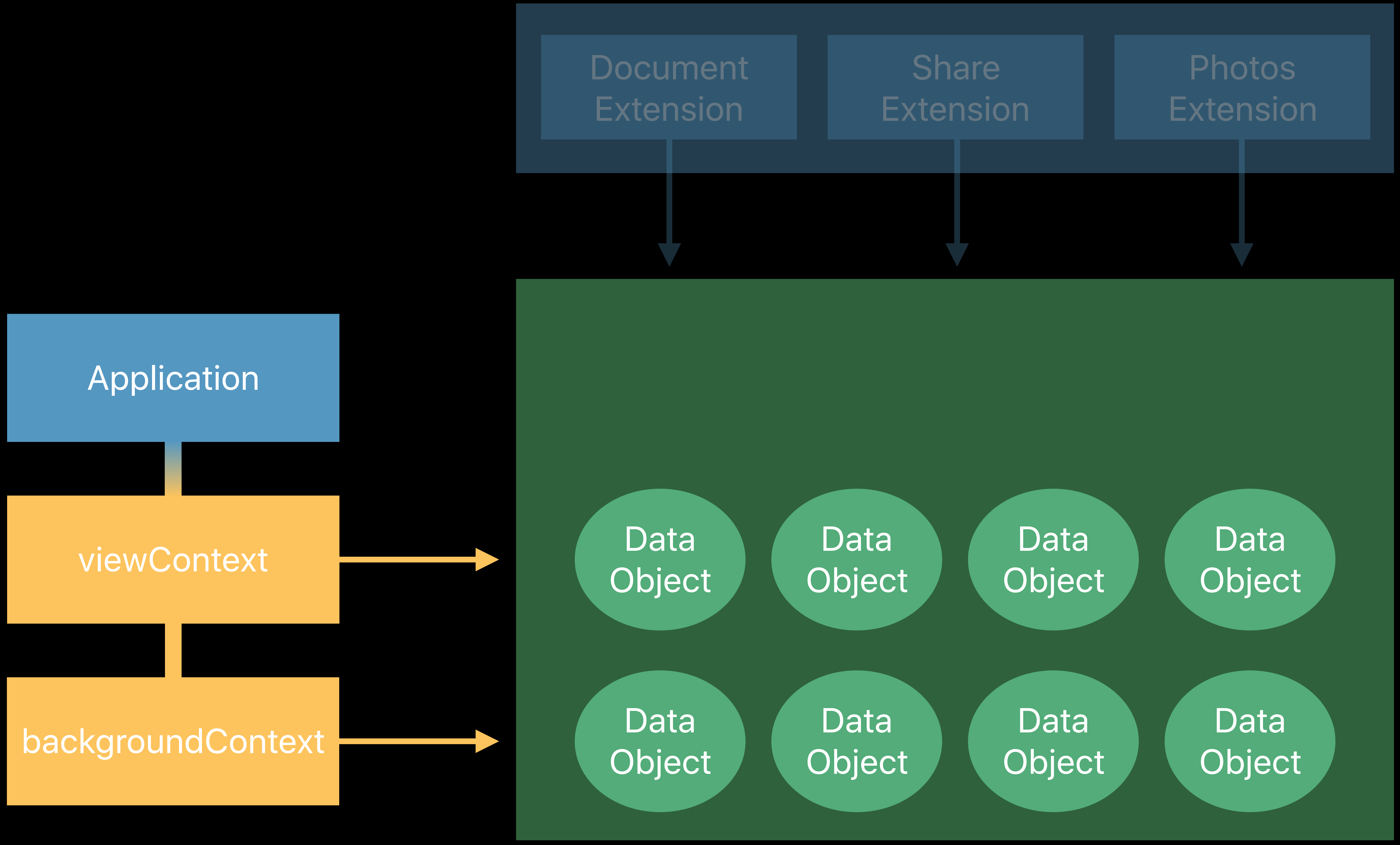
Trains in the Night



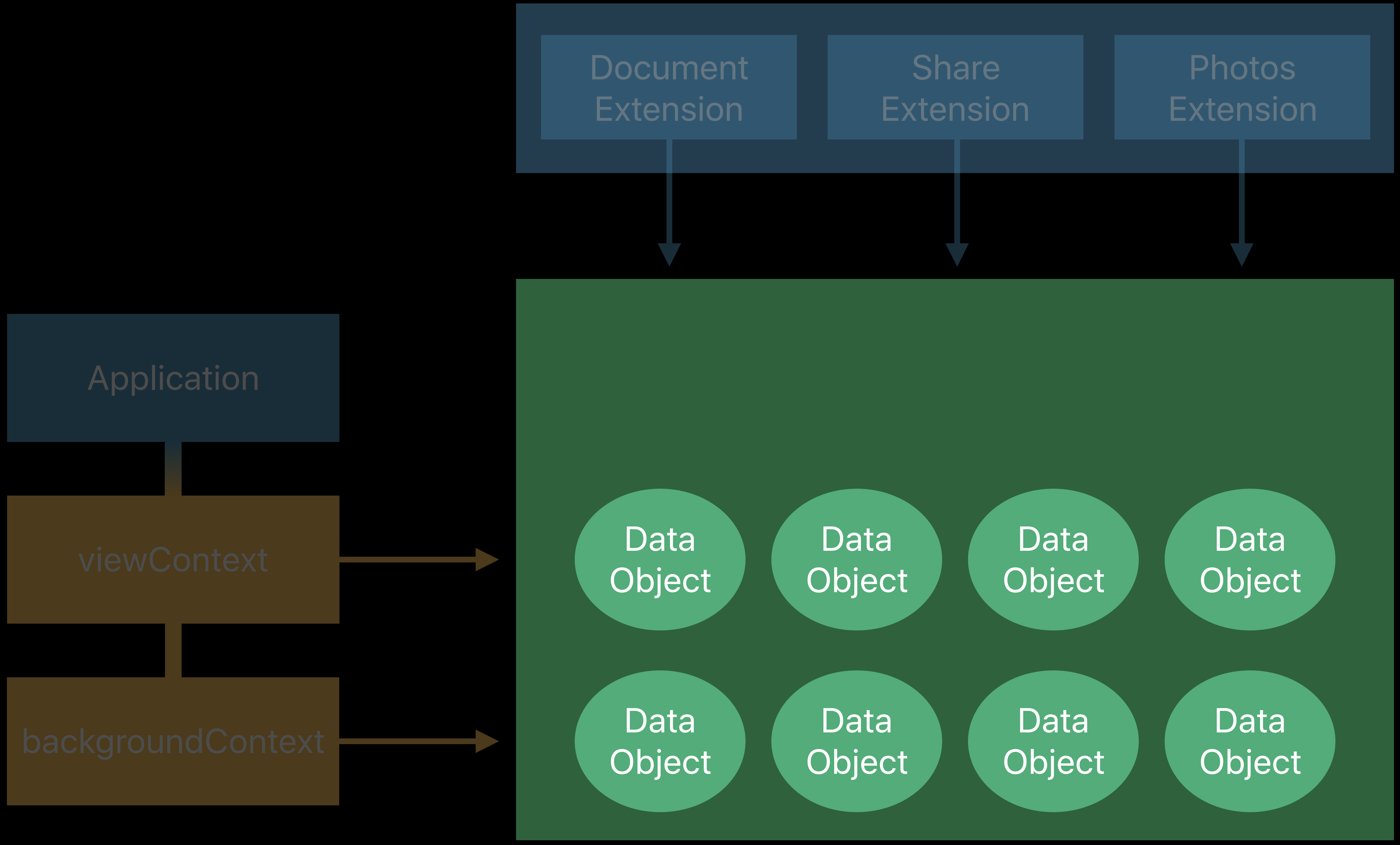
Ghost in the Database



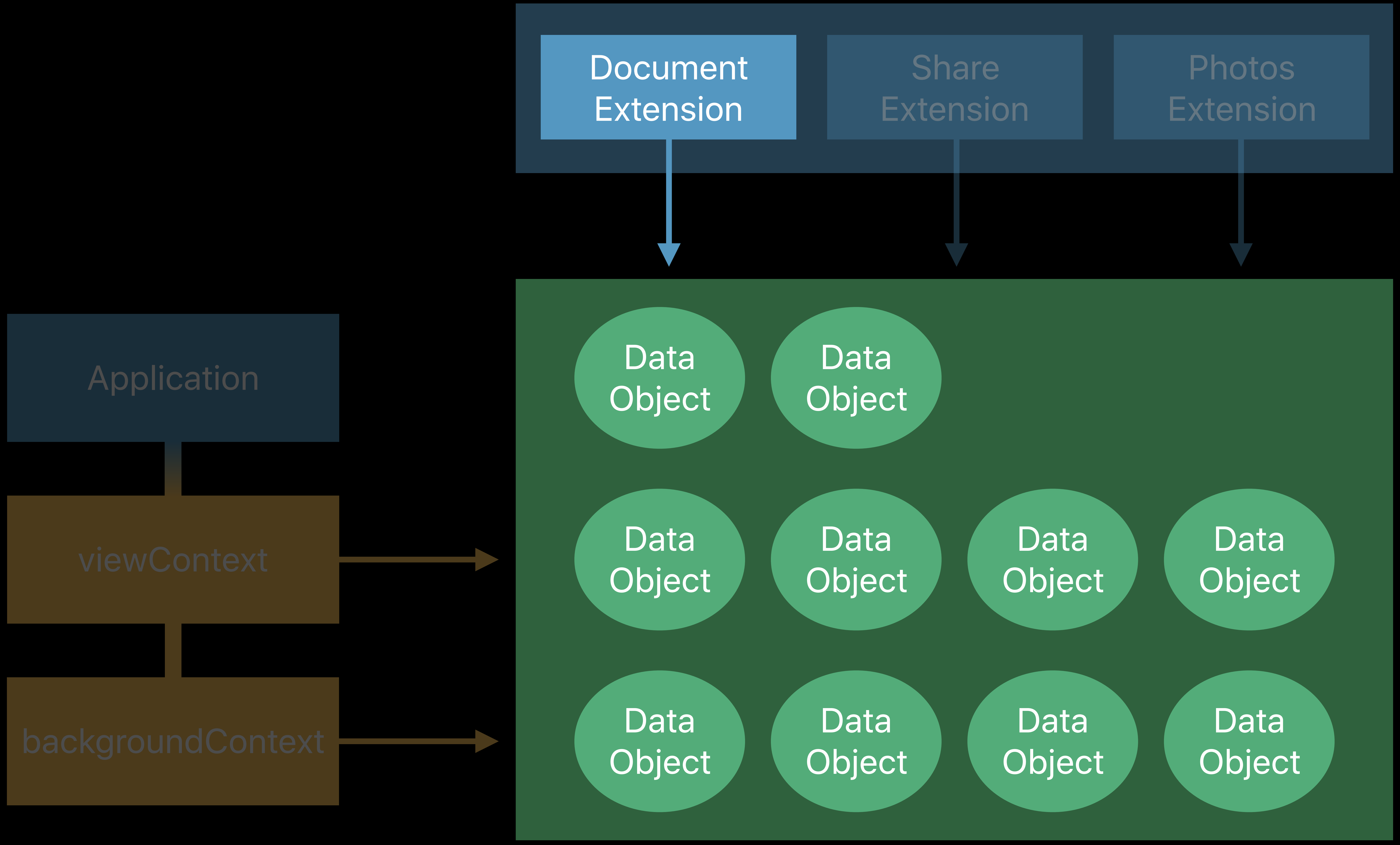
Ghost in the Database



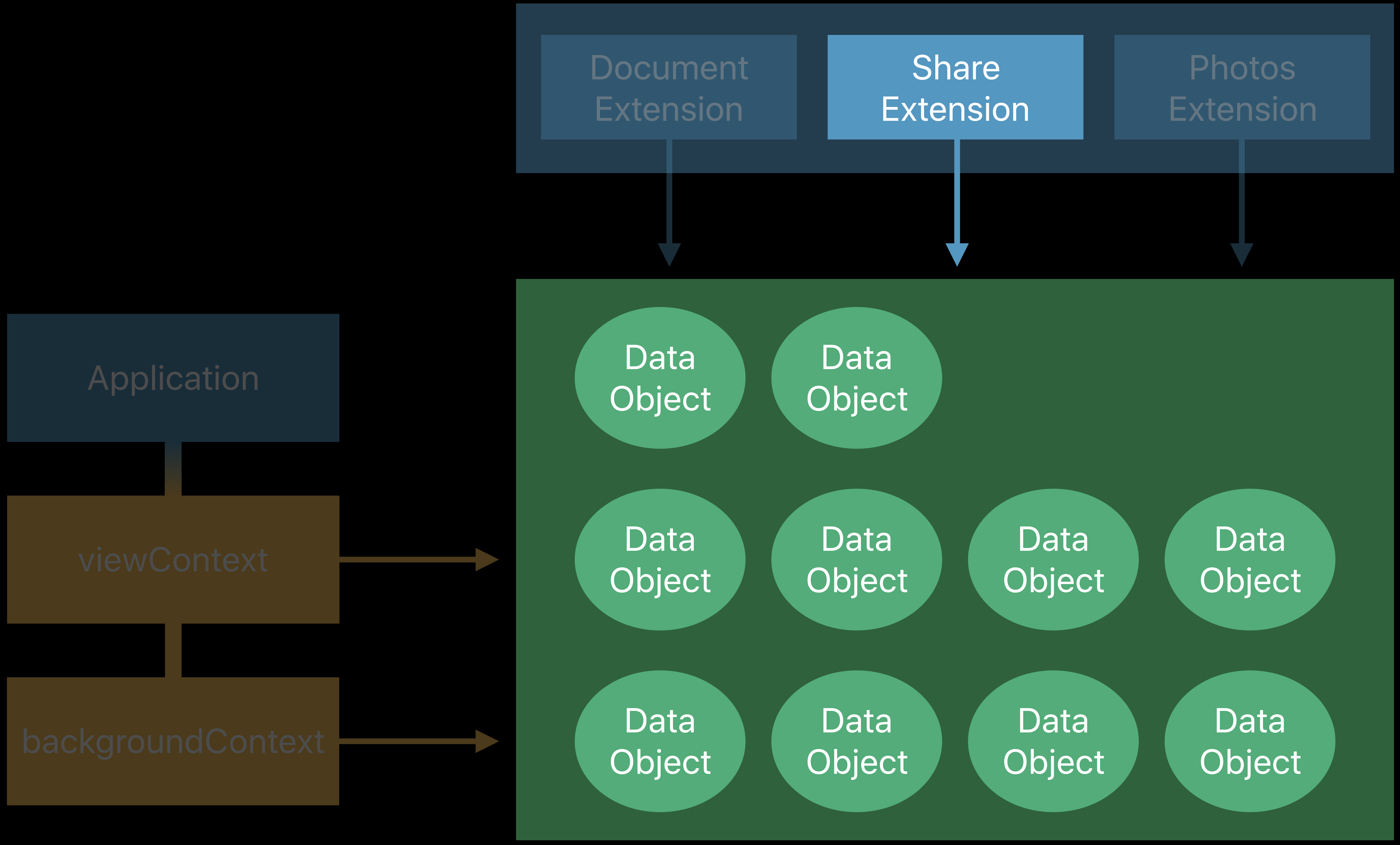
Ghost in the Database



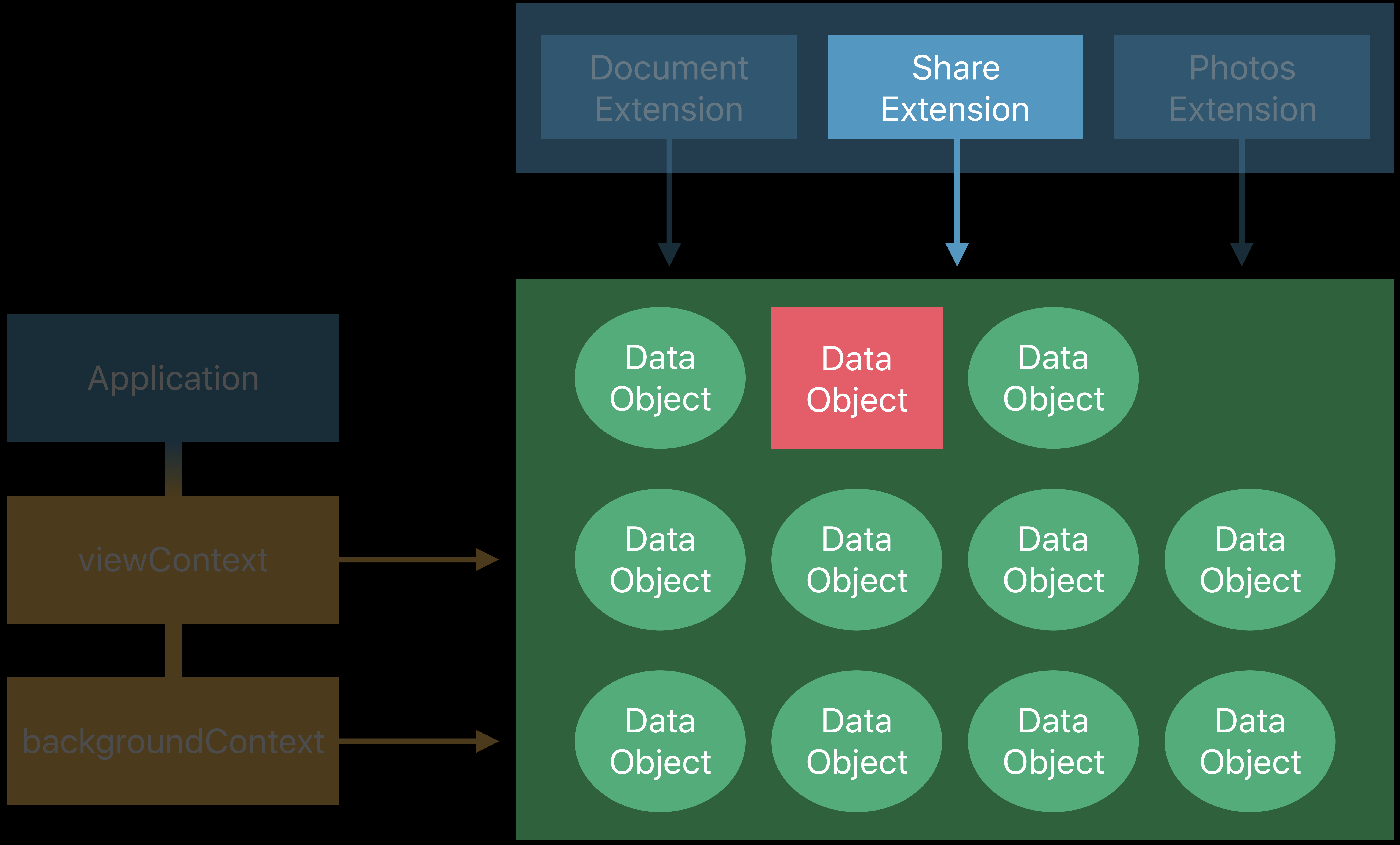
Ghost in the Database



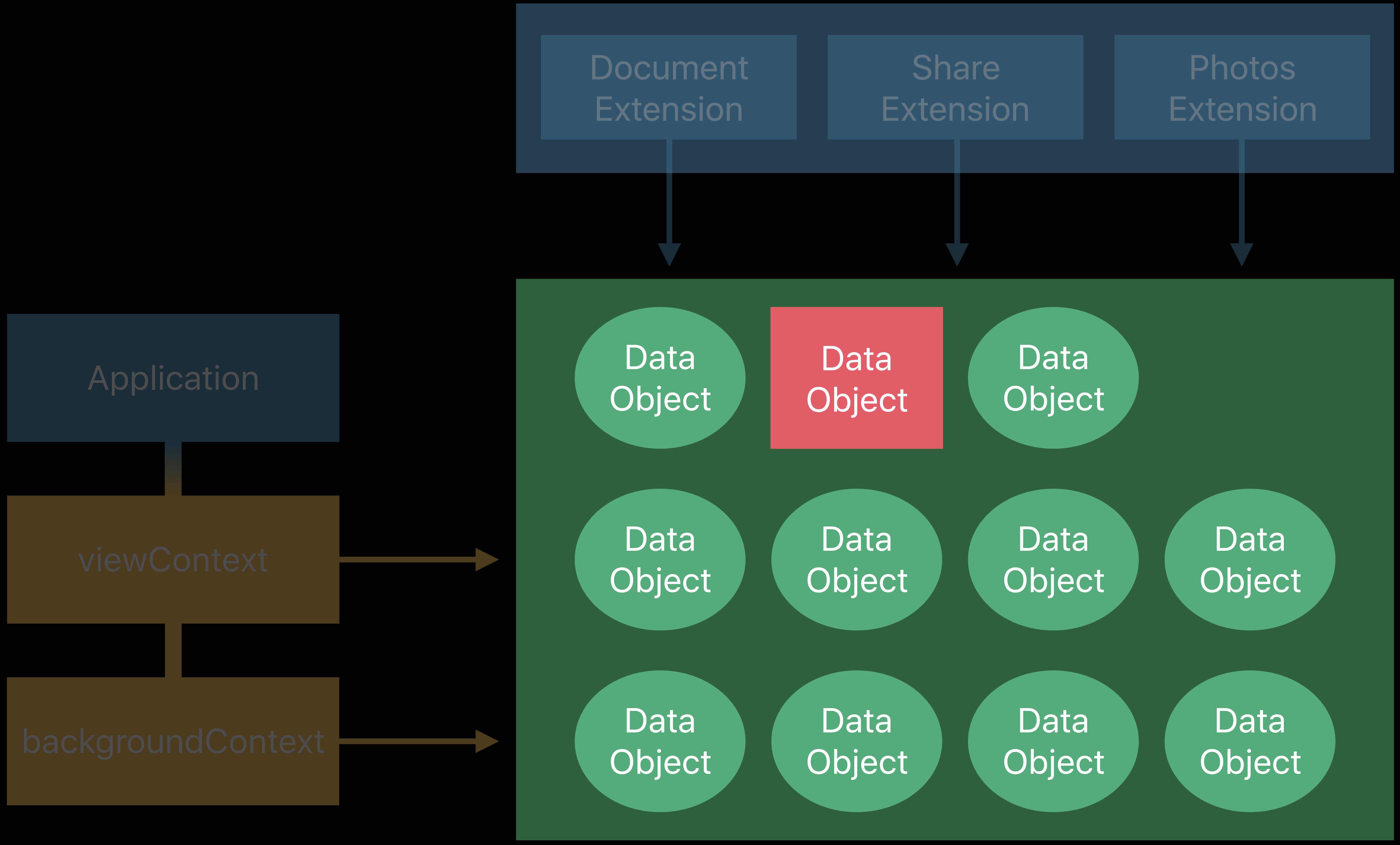
Ghost in the Database



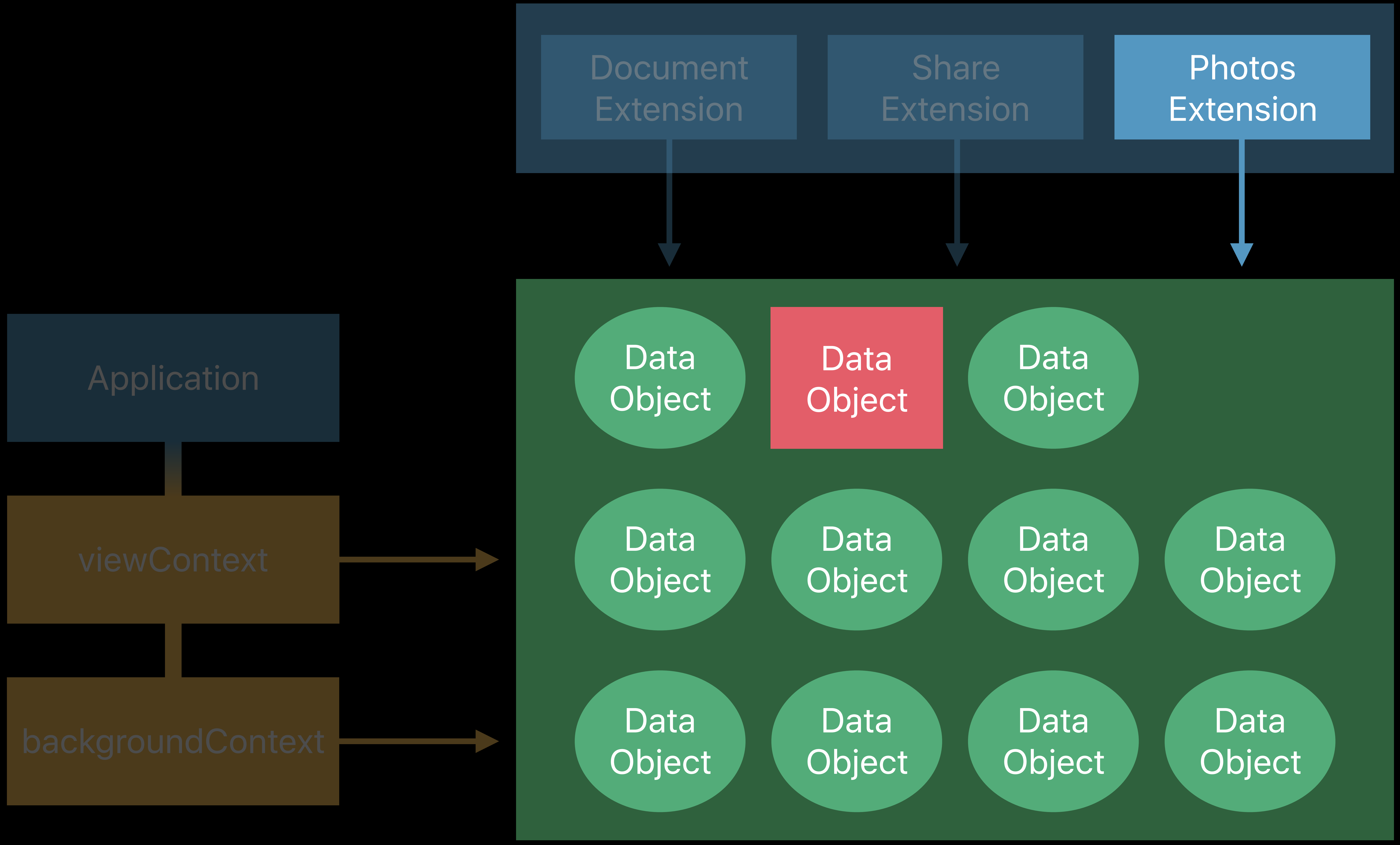
Ghost in the Database



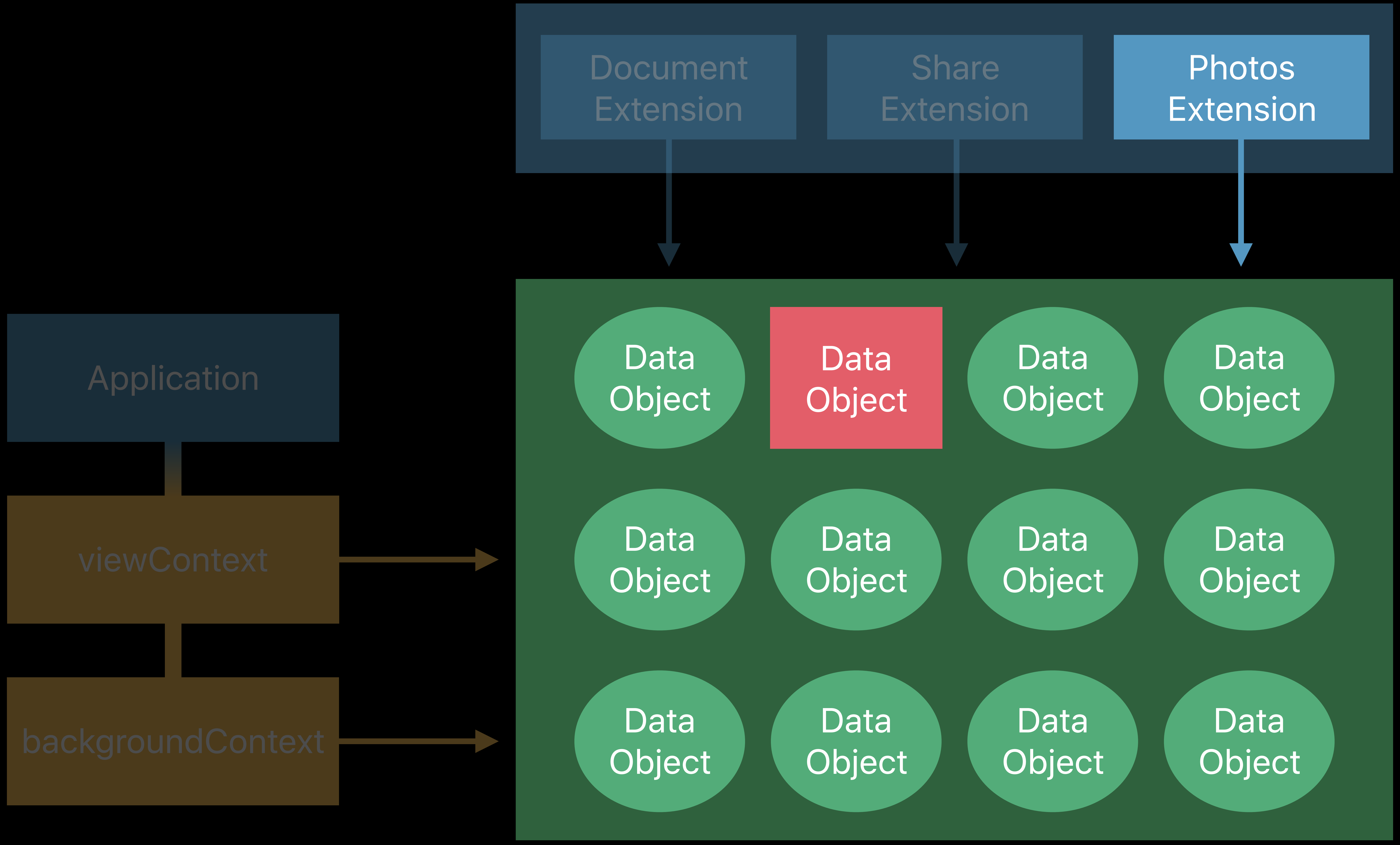
Ghost in the Database



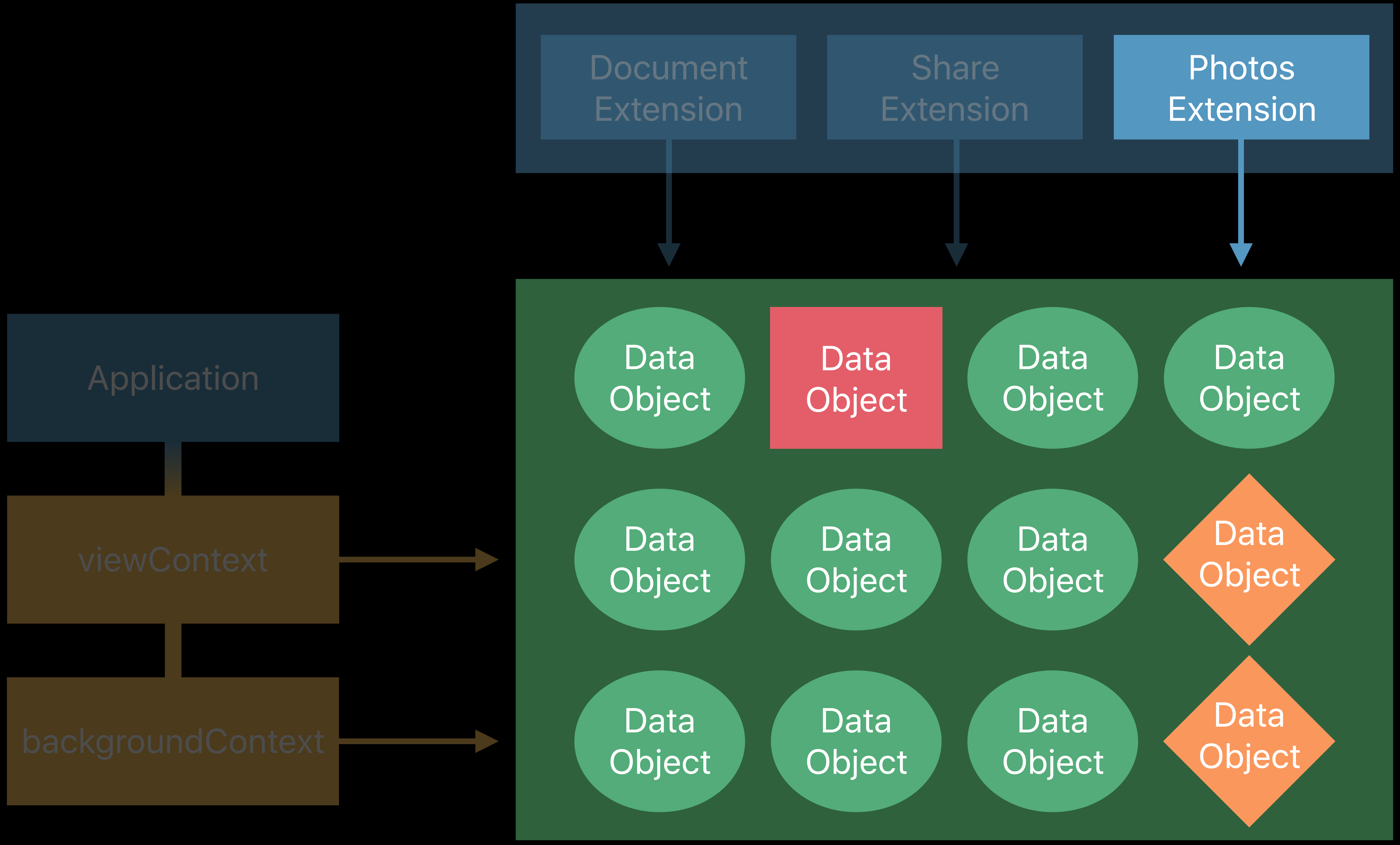
Ghost in the Database



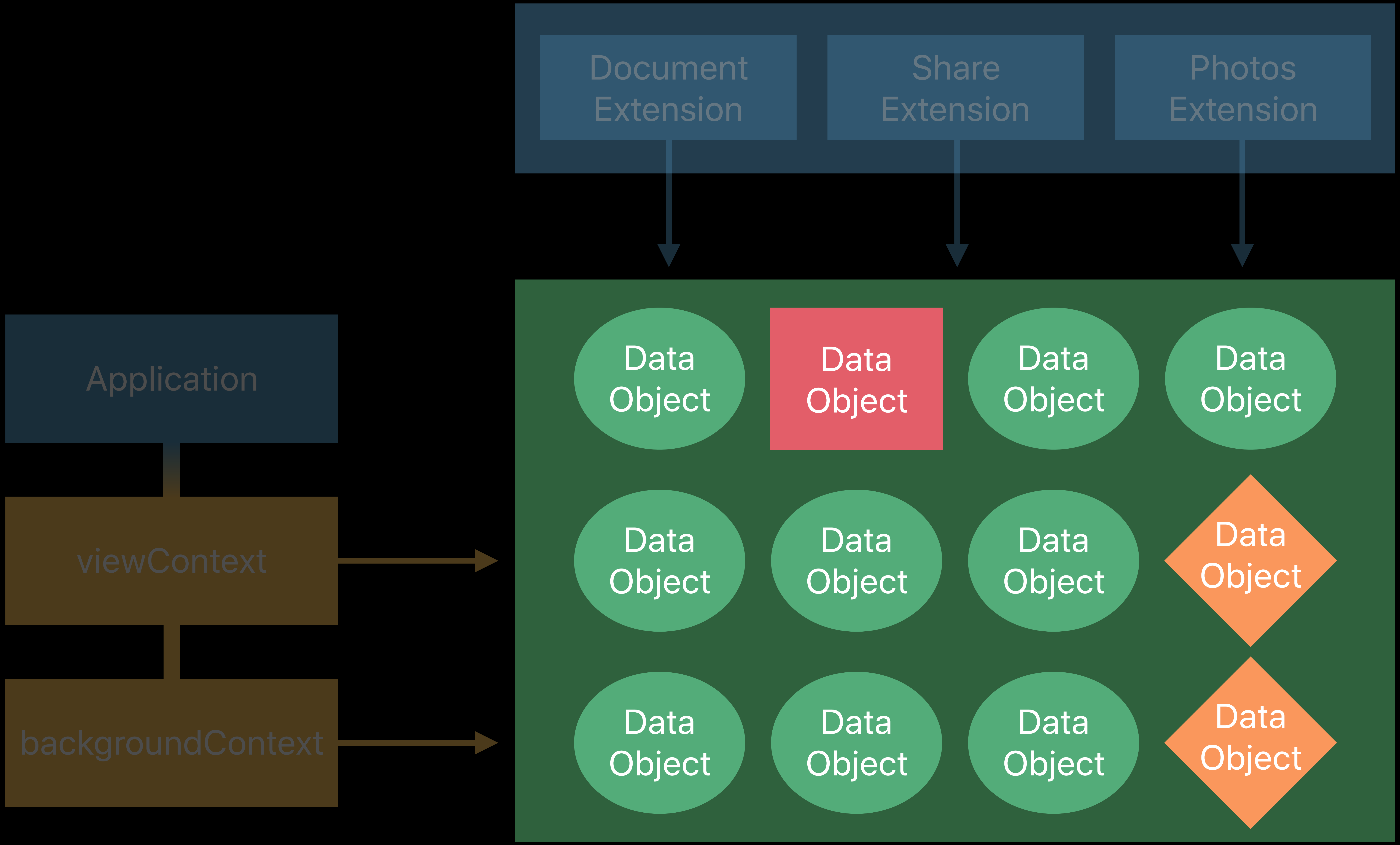
Ghost in the Database



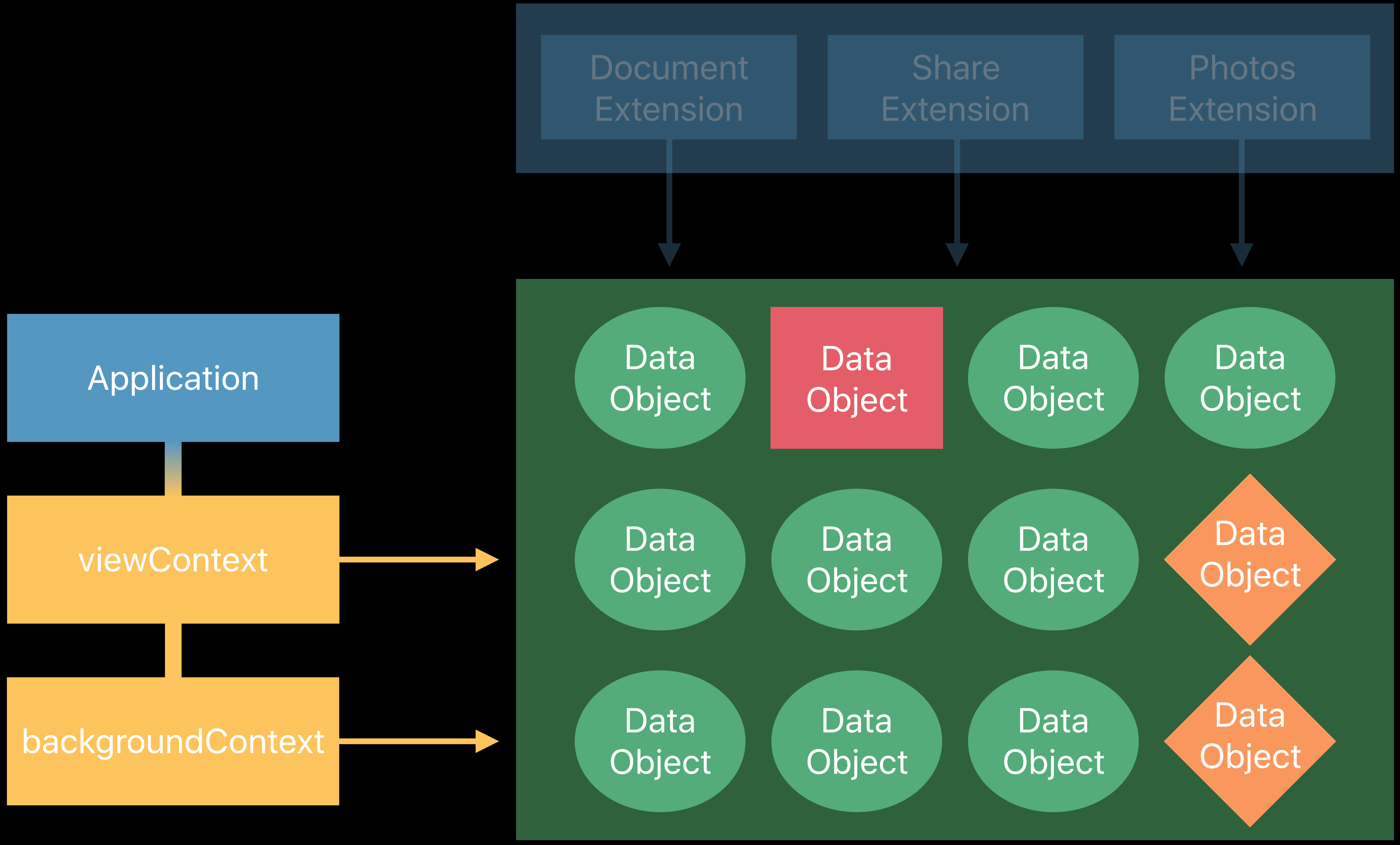
Ghost in the Database



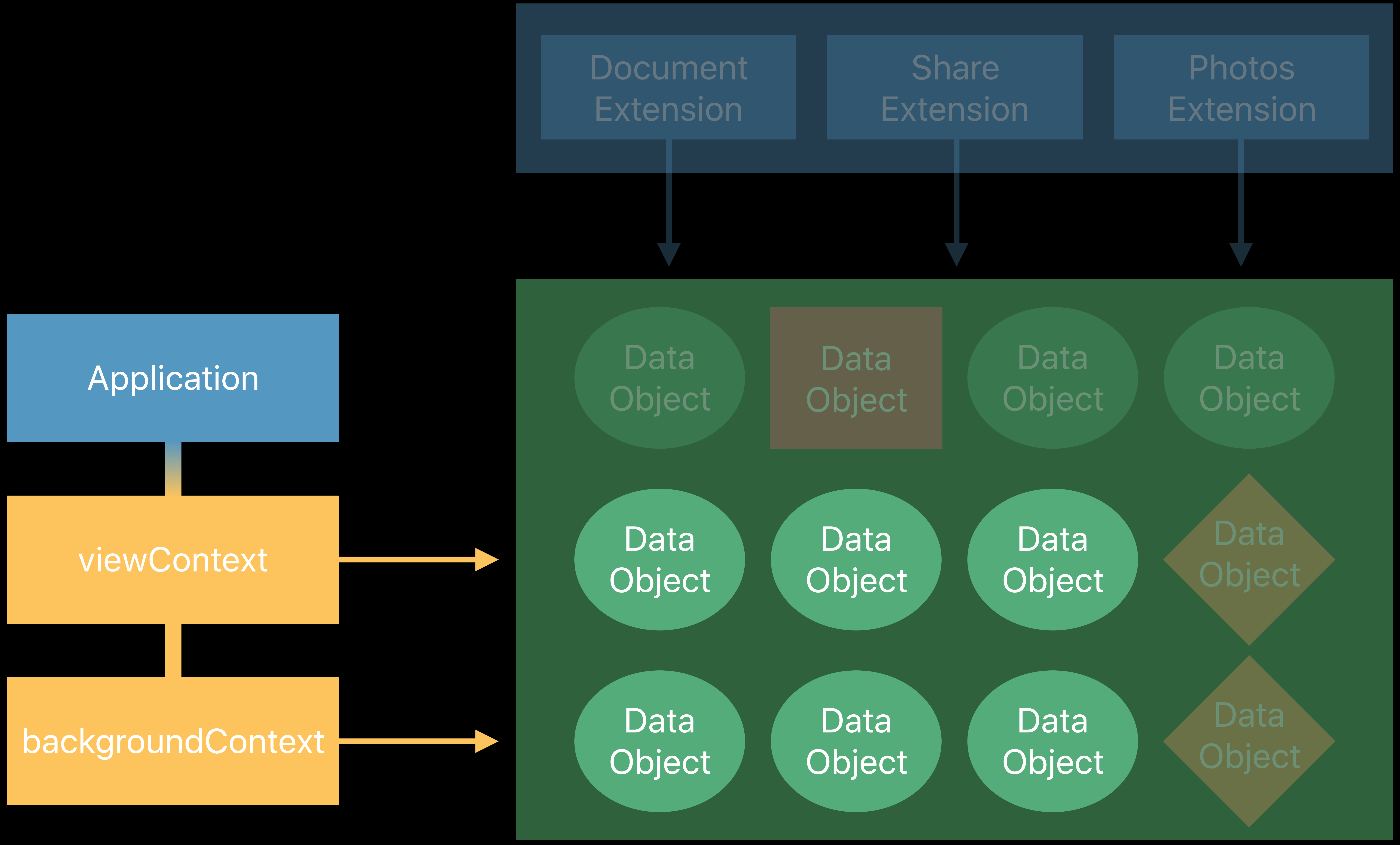
Ghost in the Database



Ghost in the Database



Ghost in the Database



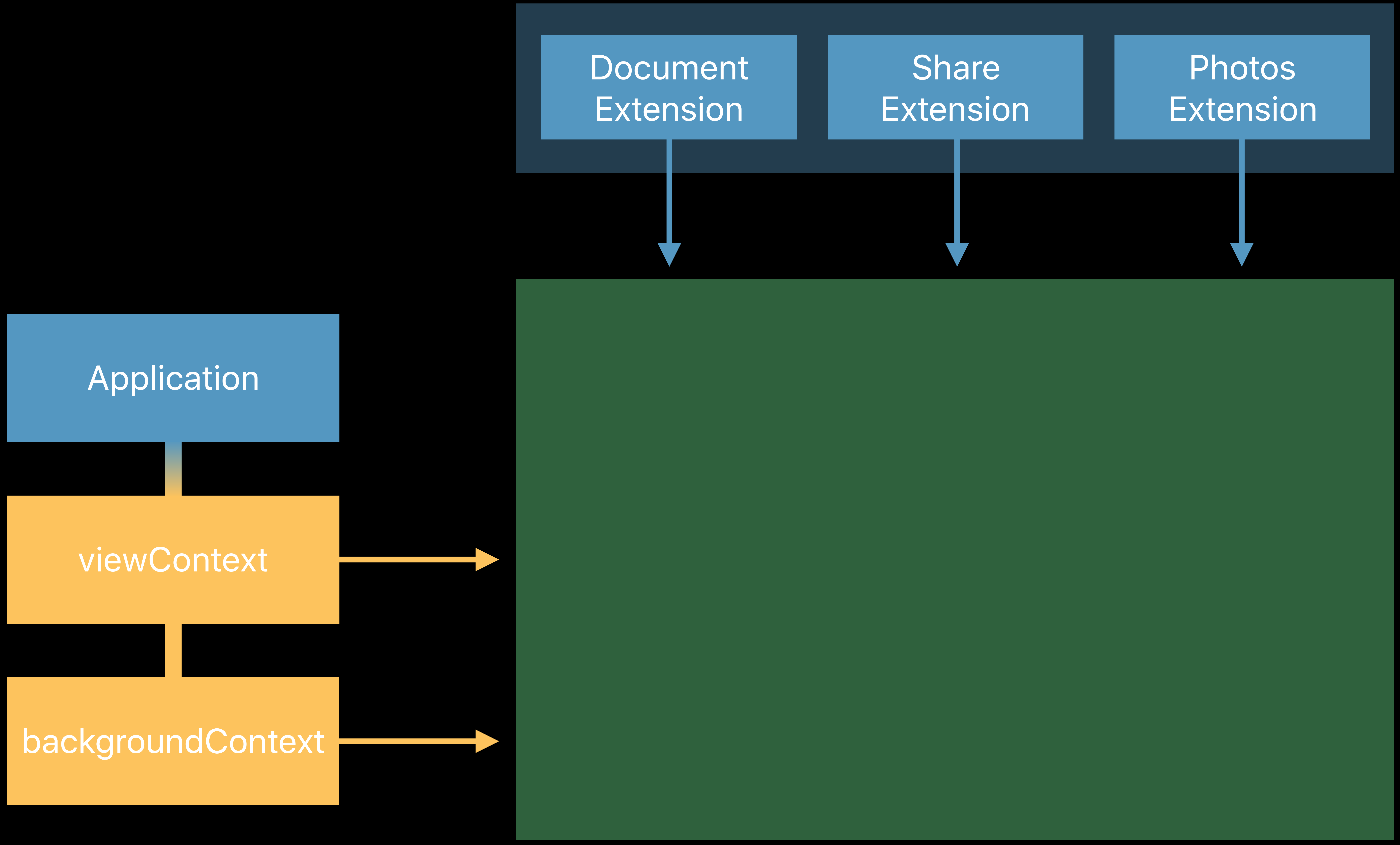
```swift
//Uncovering The Mystery – Made Easy, as simple as ABC

let container = NSPersistentContainer(name: "WWDCDemo")
let storeDescription = container.persistentStoreDescriptions.first

storeDescription?.setOption(true as NSNumber forKey:NSPersistentHistoryTrackingKey)

container.loadPersistentStores { storeDescription, error in
    …
}
```

```swift
//Uncovering The Mystery - Made Easy, as simple as ABC

let container = NSPersistentContainer(name: "WWDCDemo")
let storeDescription = container.persistentStoreDescriptions.first

storeDescription?.setOption(true as NSNumber forKey:NSPersistentHistoryTrackingKey)

container.loadPersistentStores { storeDescription, error in
    …
}
```

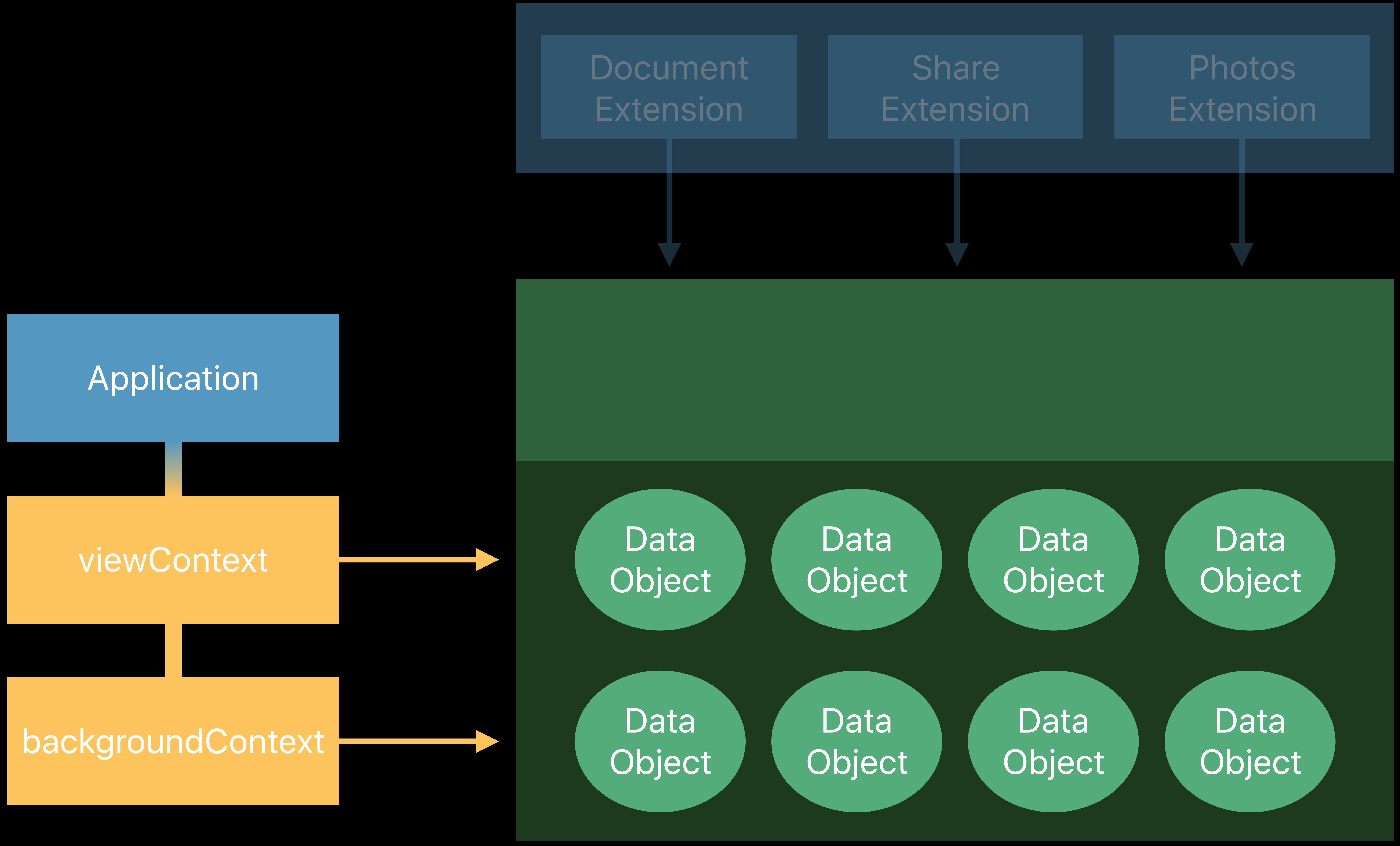
```
//Uncovering The Mystery - Made Easy, as simple as ABC

let container = NSPersistentContainer(name: "WWDCDemo")
let storeDescription = container.persistentStoreDescriptions.first

storeDescription?.setOption(true as NSNumber forKey:NSPersistentHistoryTrackingKey)

container.loadPersistentStores { storeDescription, error in
    …
}
```

```swift
//Uncovering The Mystery - Made Easy, as simple as ABC

let container = NSPersistentContainer(name: "WWDCDemo")
let storeDescription = container.persistentStoreDescriptions.first

storeDescription?.setOption(true as NSNumber forKey:NSPersistentHistoryTrackingKey)

container.loadPersistentStores { storeDescription, error in
     …
}
```

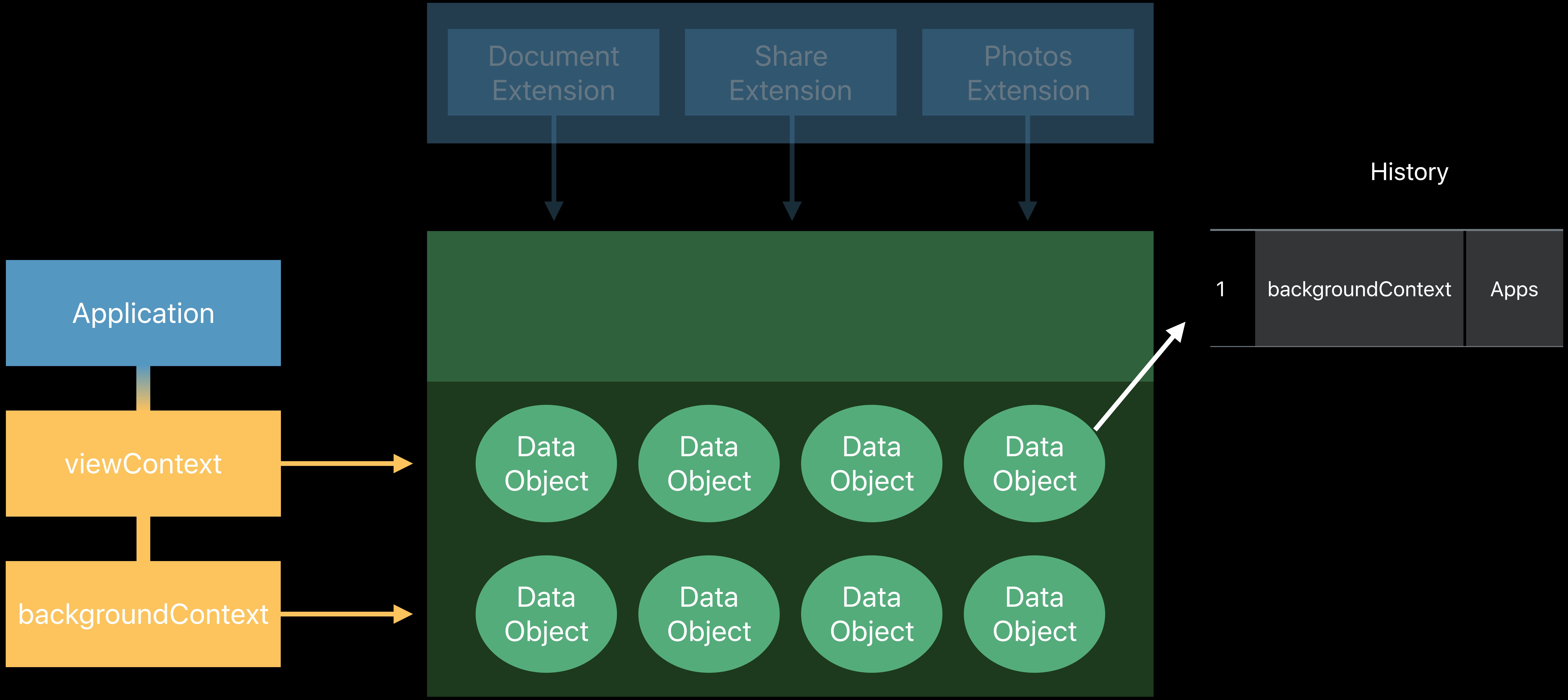
Ghost in the Database



Ghost in the Database

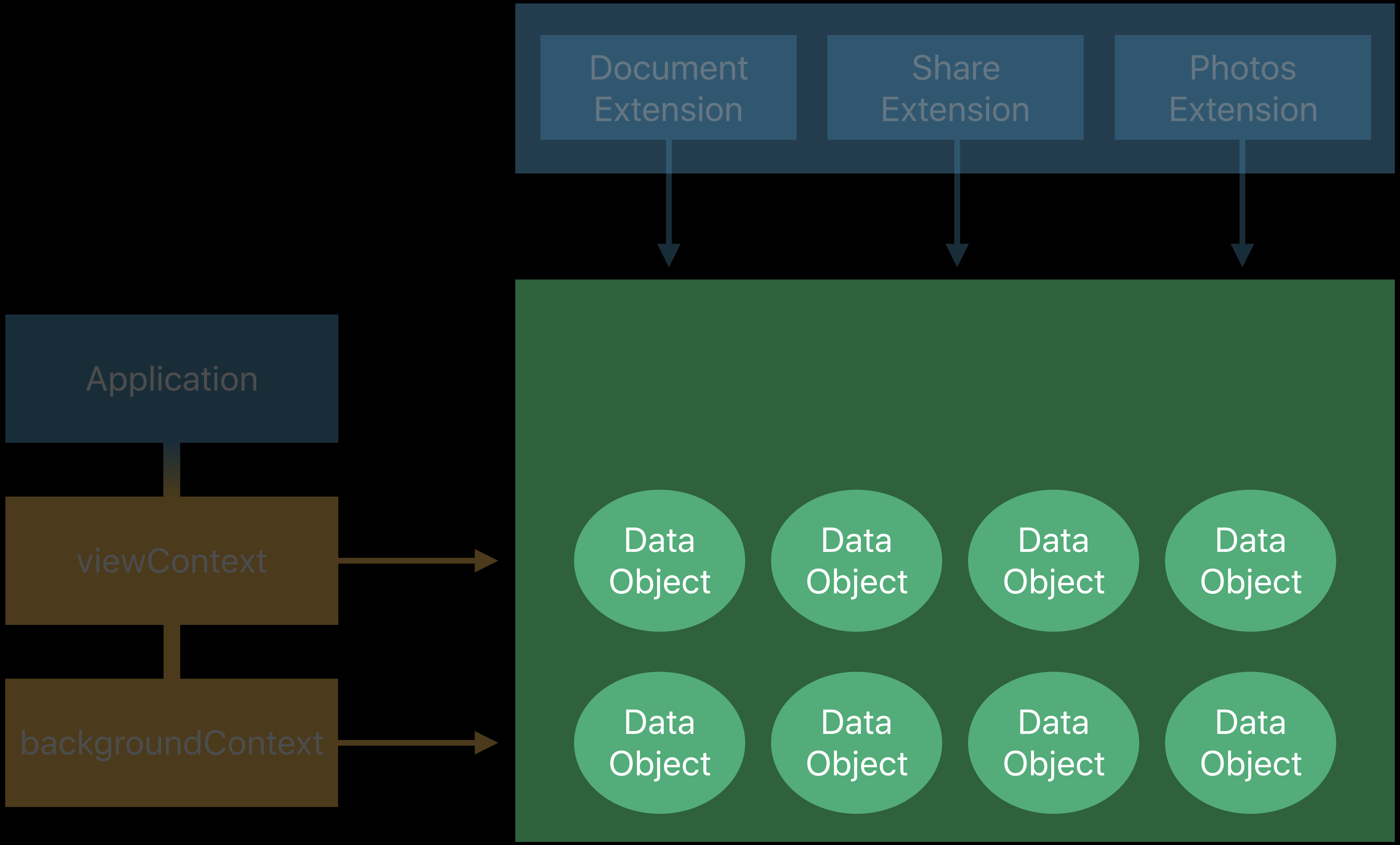


Ghost in the Database



1	backgroundContext	Apps

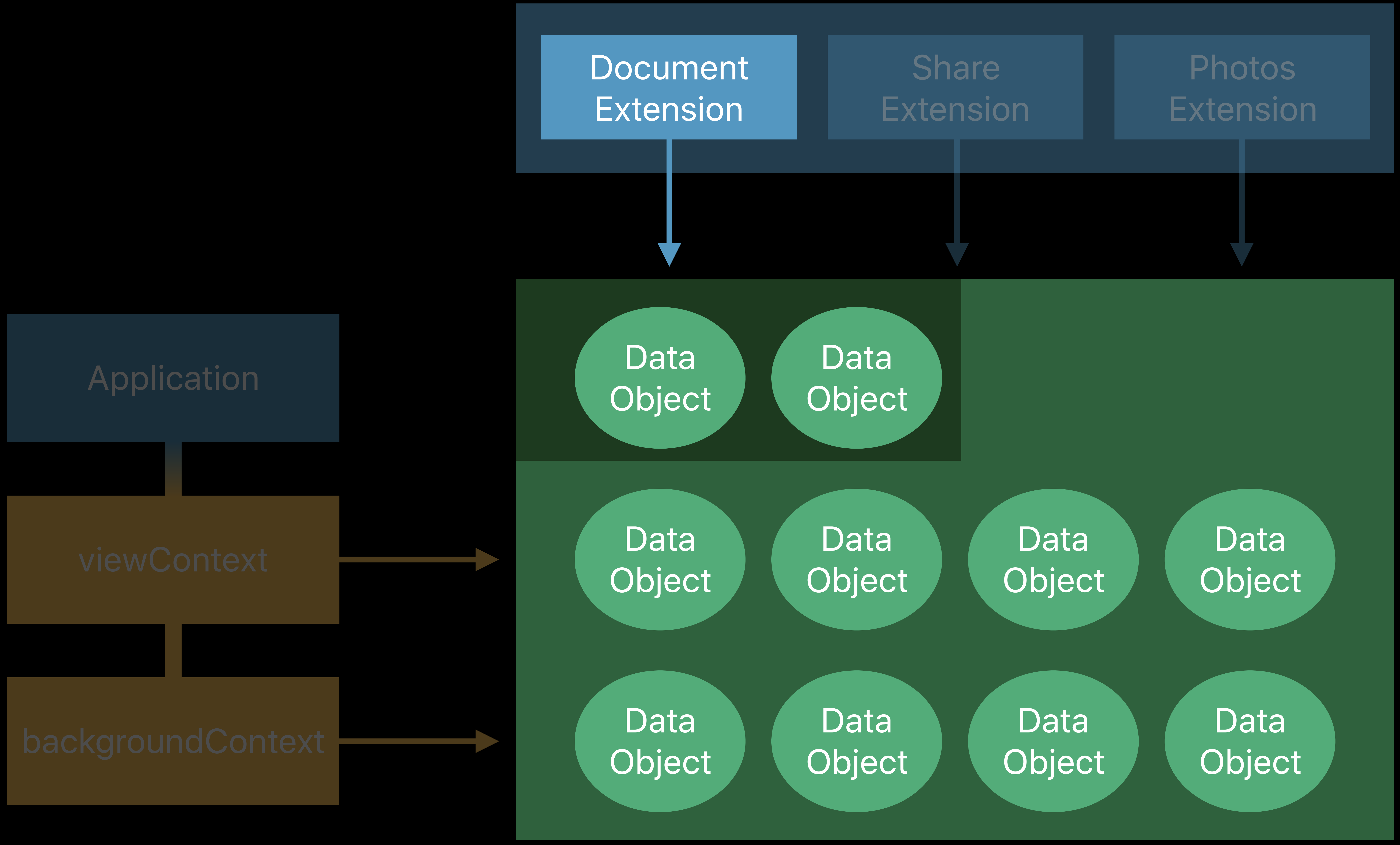
Ghost in the Database



History

1	backgroundContext	Apps

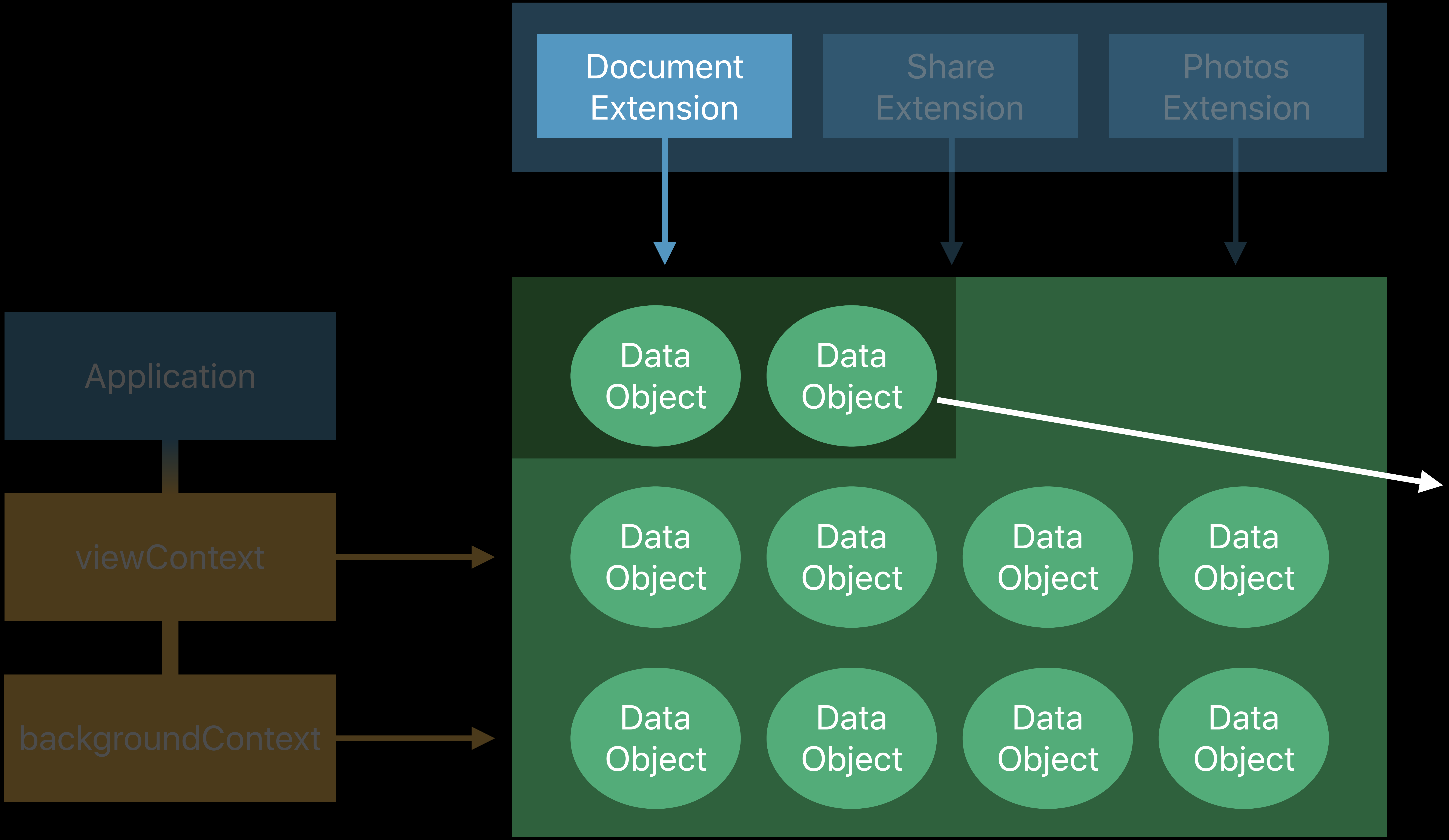
Ghost in the Database



History

1	backgroundContext	Apps

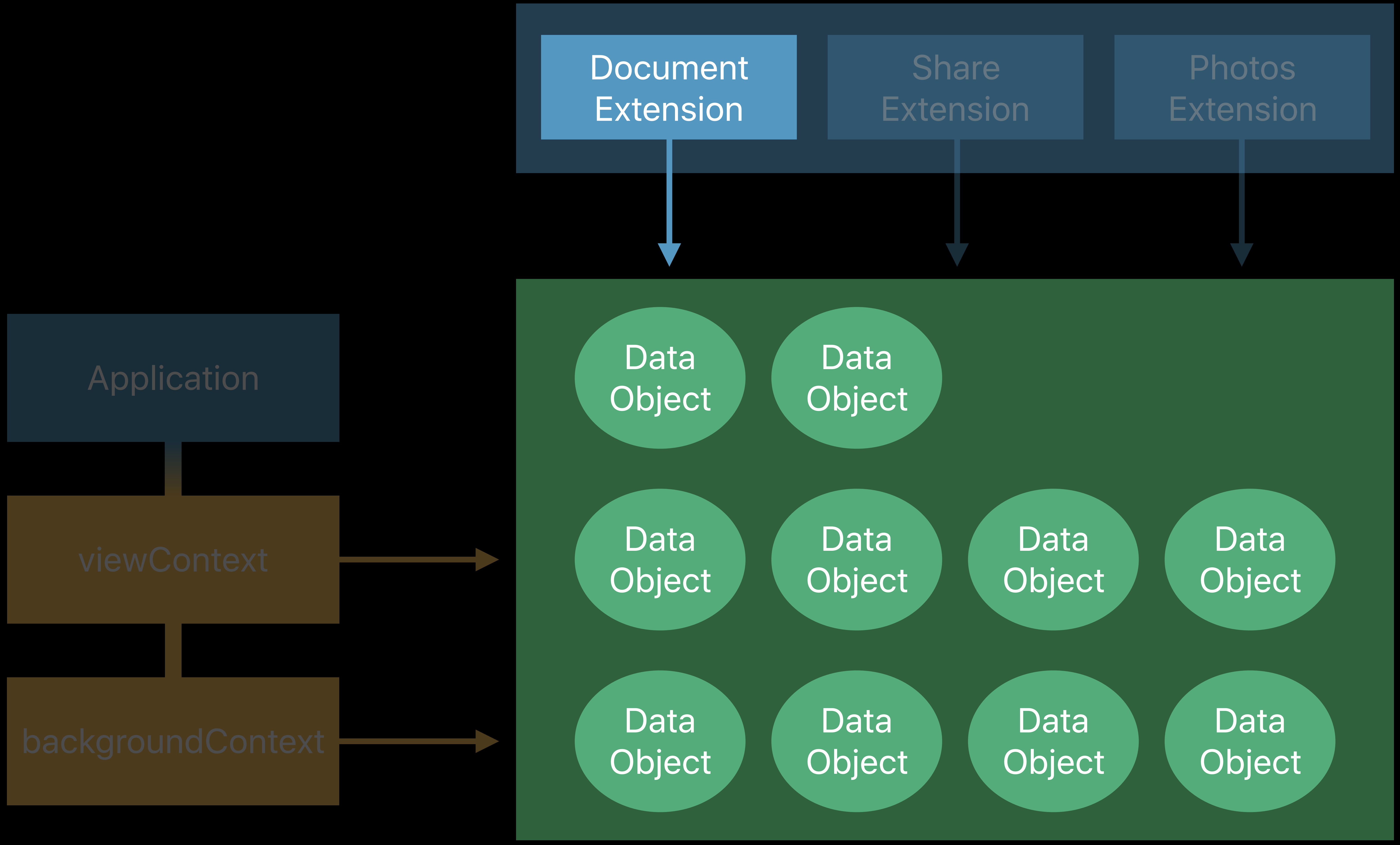
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files

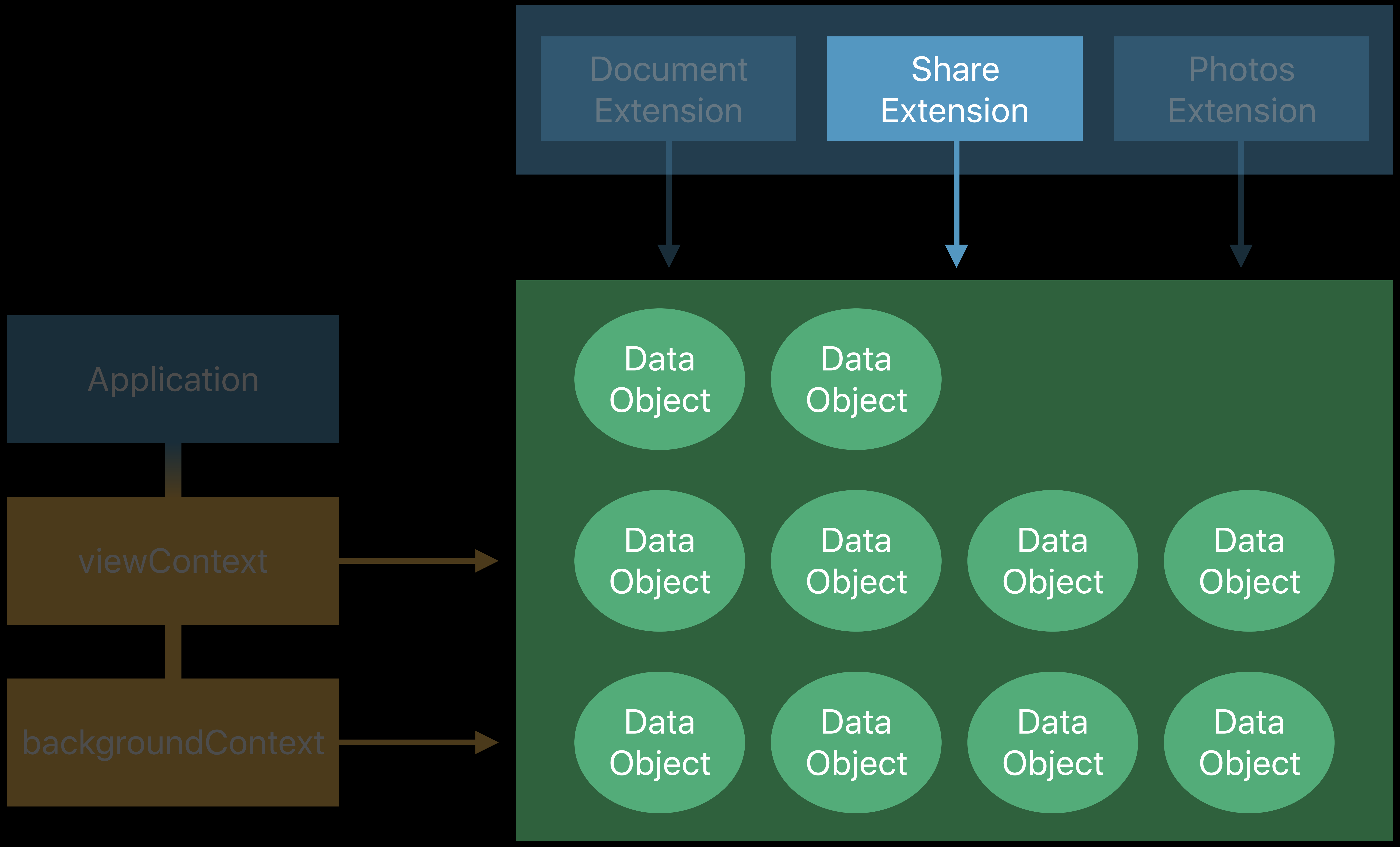
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files

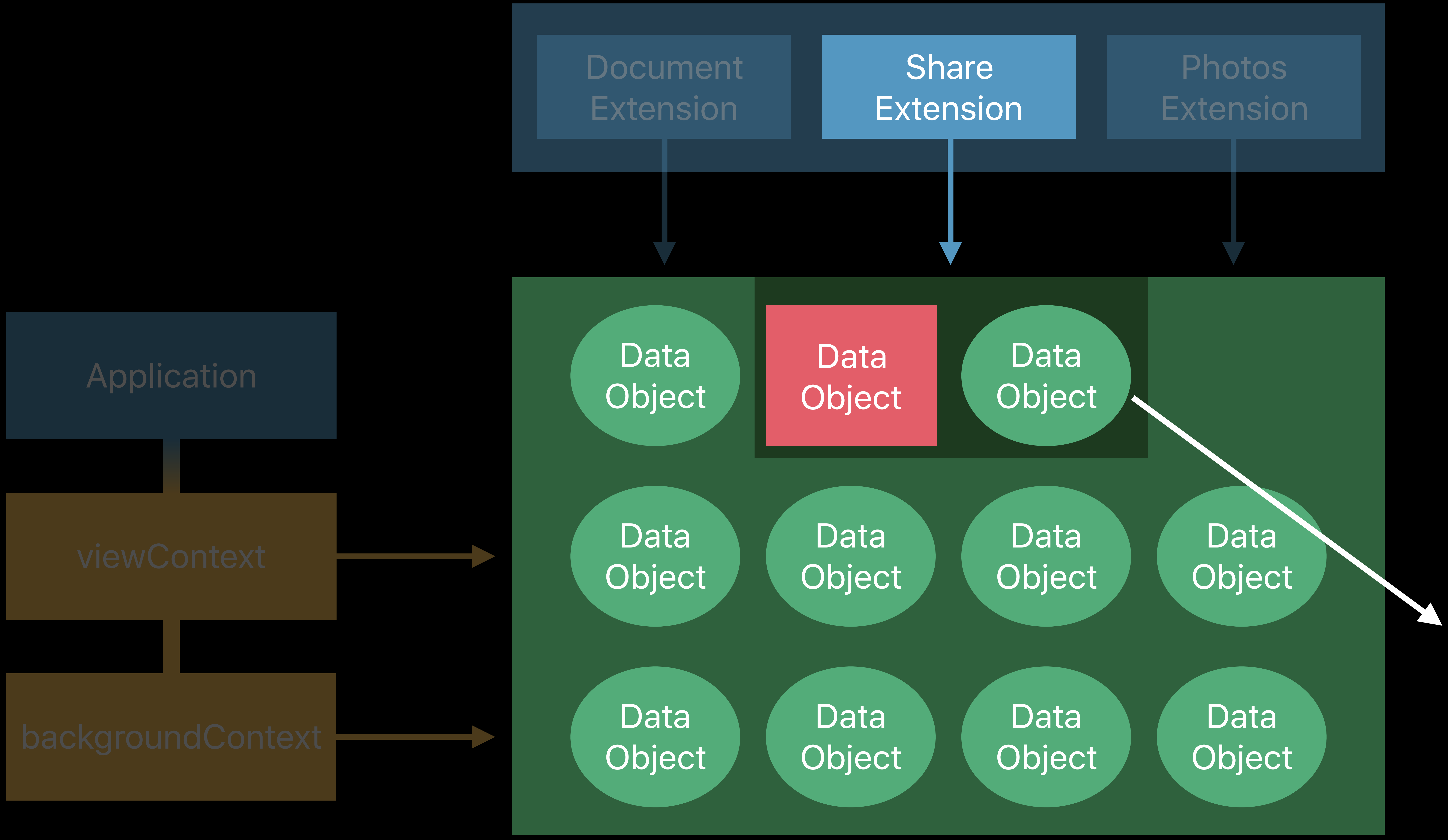
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files

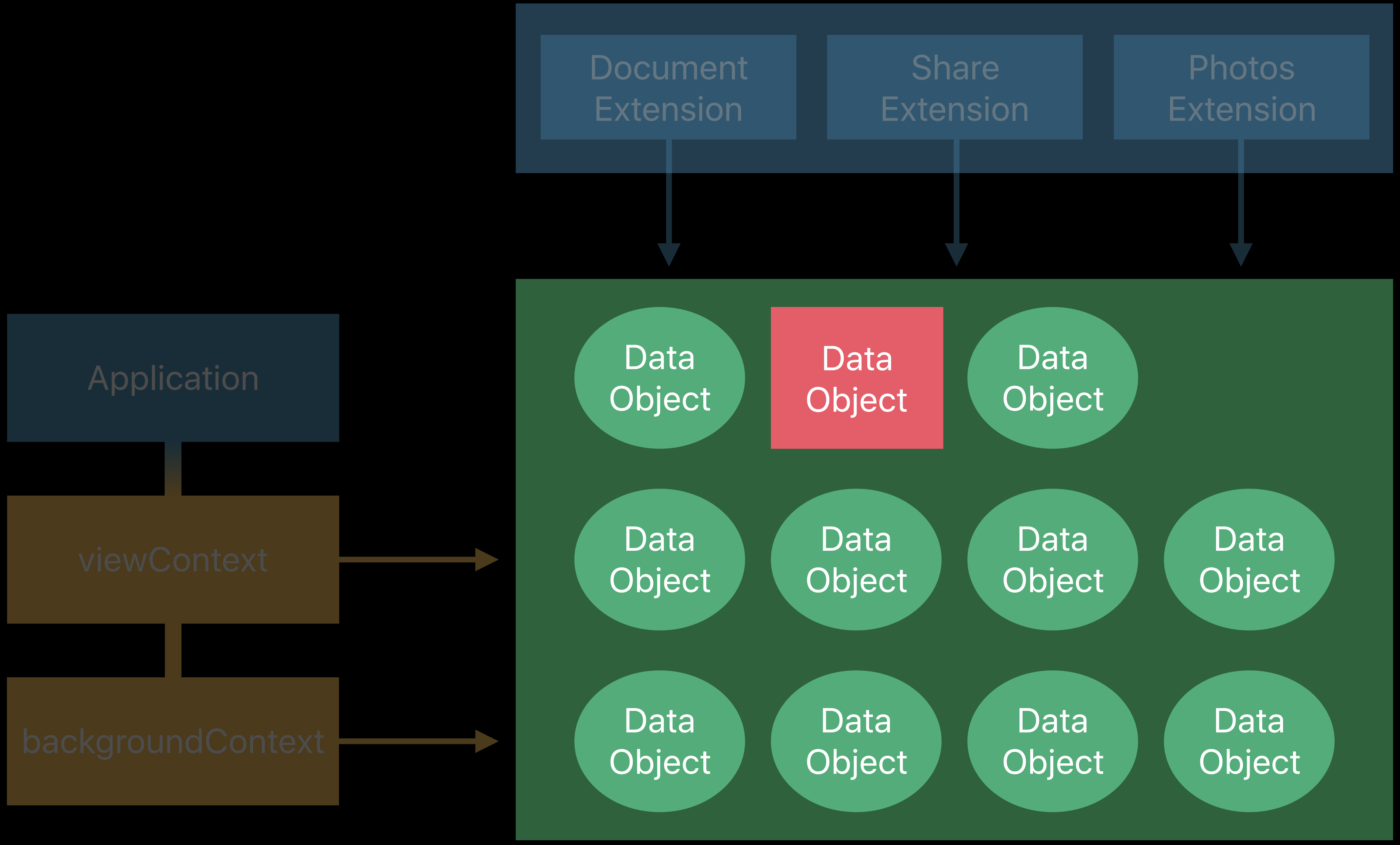
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari

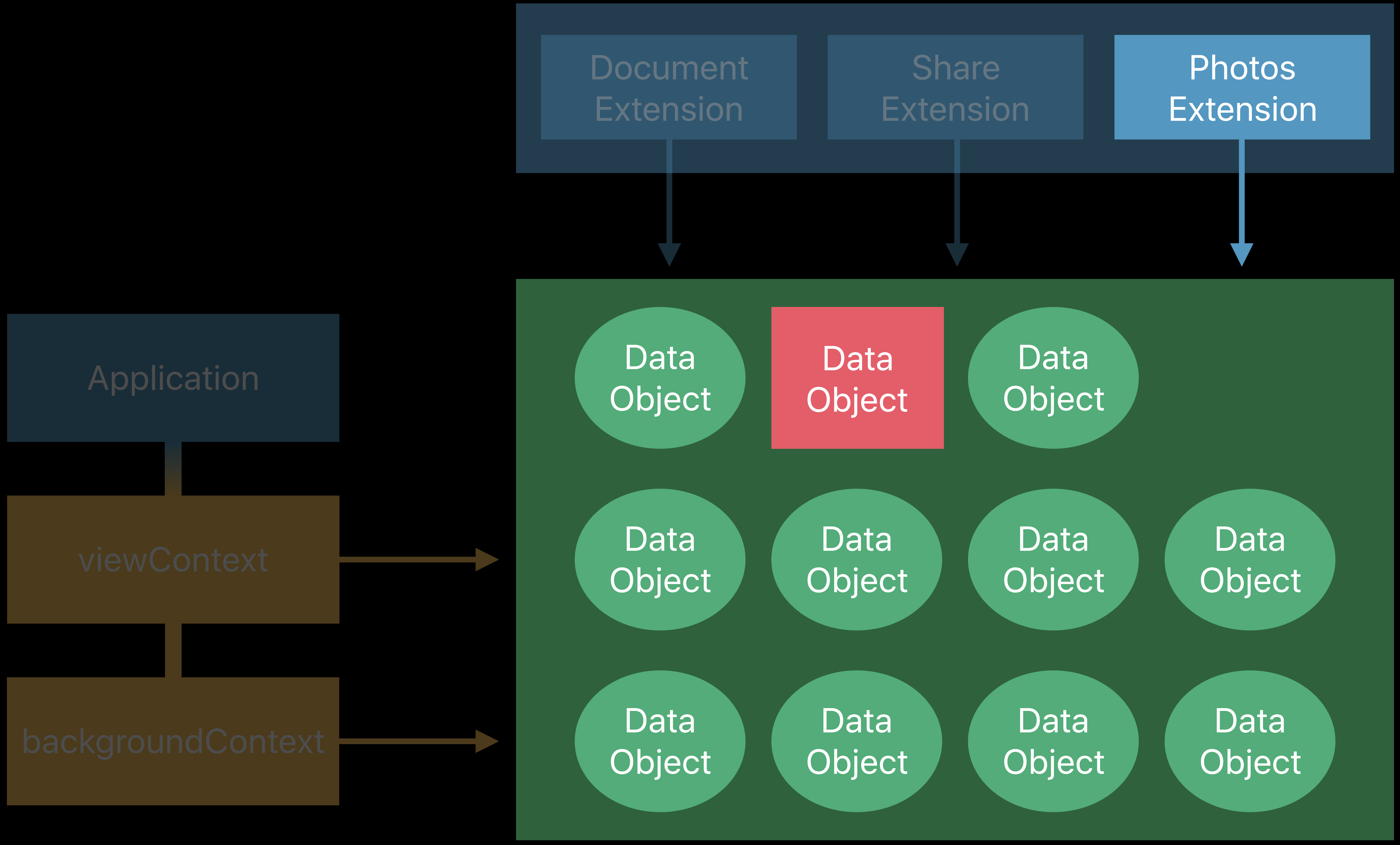
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari

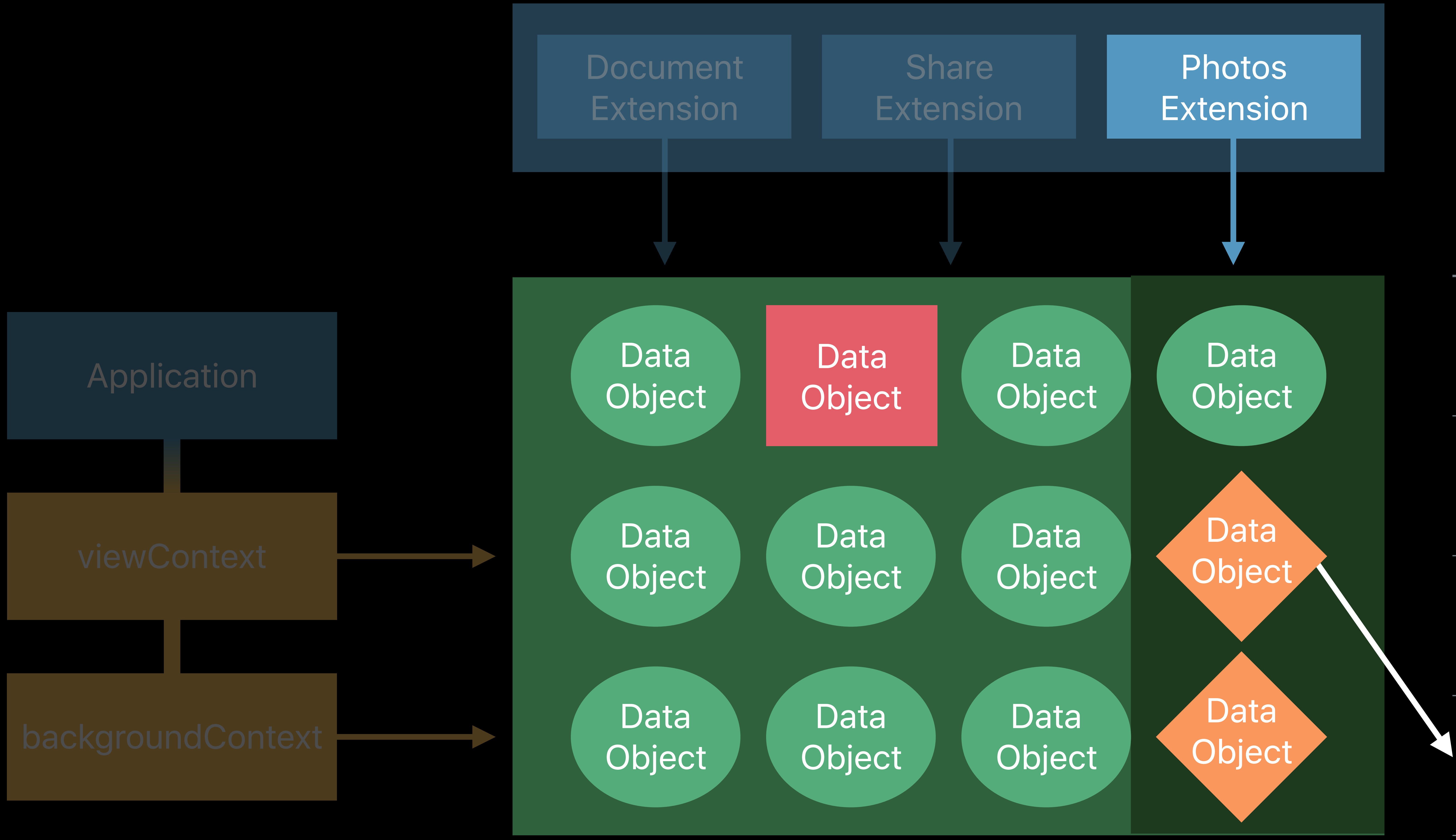
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari

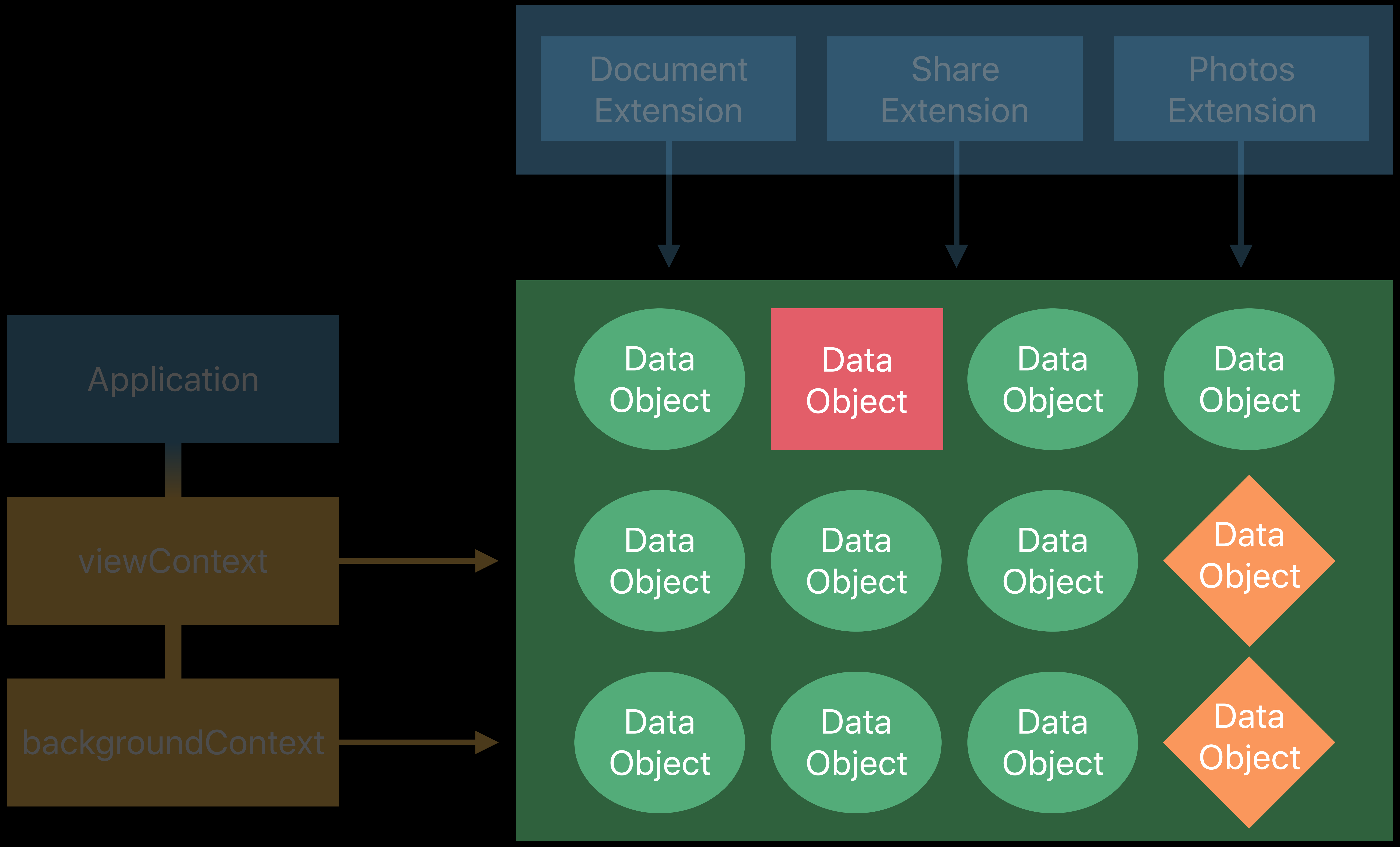
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari
4	PhotosExtension	Photos

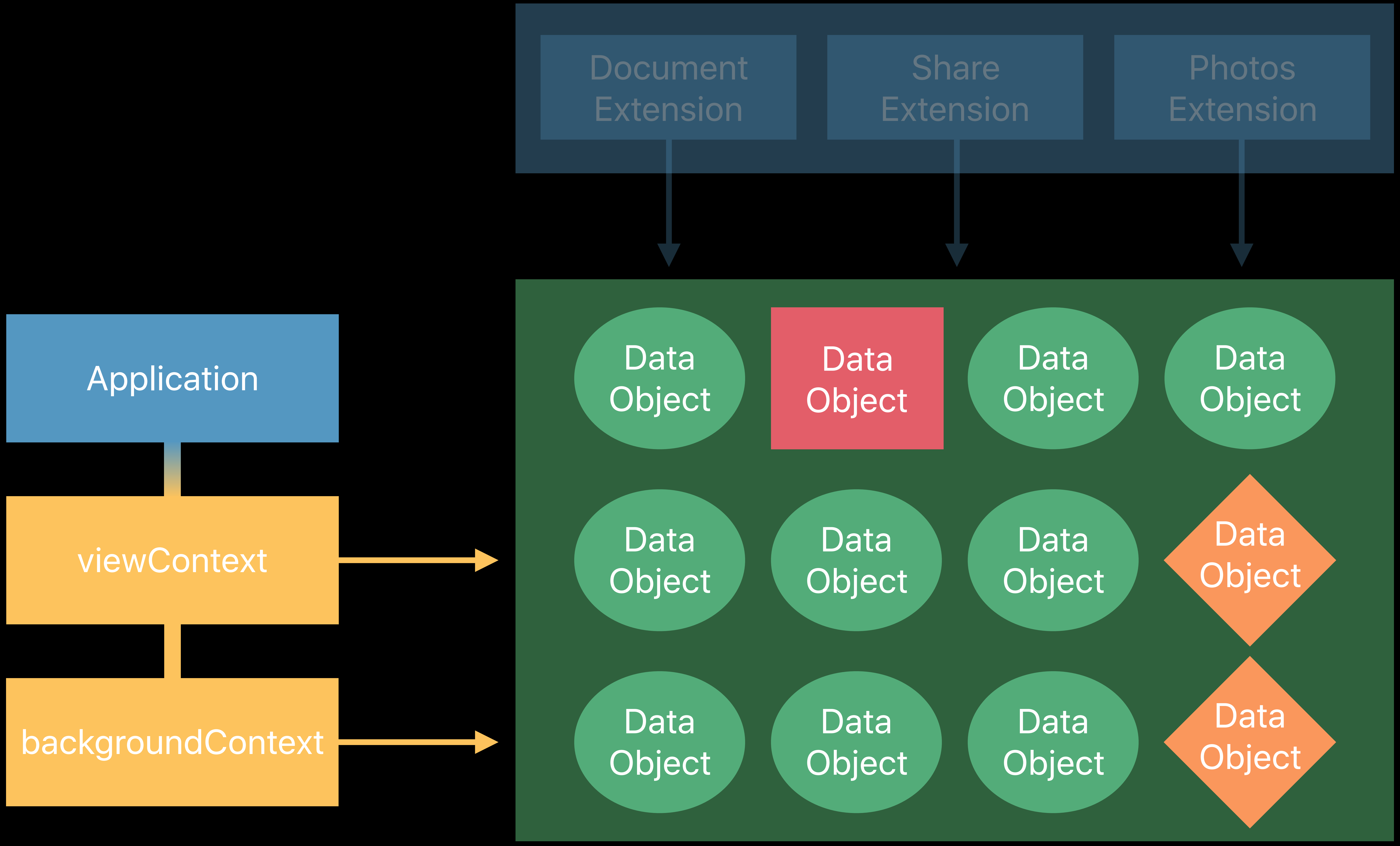
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari
4	PhotosExtension	Photos

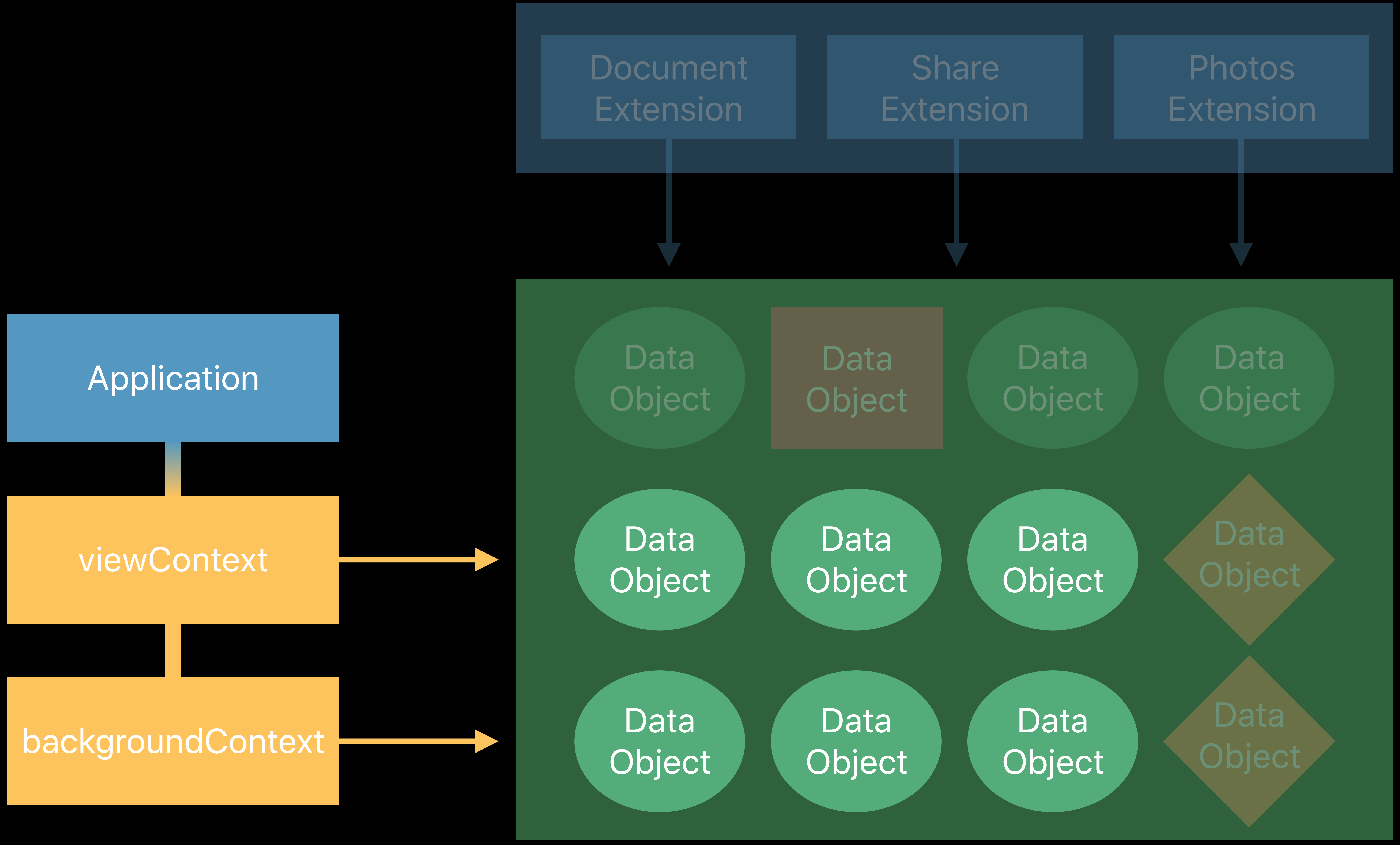
Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari
4	PhotosExtension	Photos

Ghost in the Database



History

1	backgroundContext	Apps
2	DocumentExtension	Files
3	ShareExtension	Safari
4	PhotosExtension	Photos

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```
//Reading History, the journey of a historian

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func fetchHistory(after date: Date) -> Self
    open class func fetchHistory(after token: NSPersistentHistoryToken?) -> Self
    open class func fetchHistory(after transaction: NSPersistentHistoryTransaction?) -> Self
    open var resultType: NSPersistentHistoryResultType
}
```

NEW

```swift
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//Reducing History, history is purged by the consumers

open class NSPersistentHistoryChangeRequest : NSPersistentStoreRequest {
    open class func deleteHistory(before date: Date) -> Self
    open class func deleteHistory(before token: NSPersistentHistoryToken?) -> Self
    open class func deleteHistory(before transaction: NSPersistentHistoryTransaction?) -> Self
}


//A single gatekeeper should be in charge of purging history
```

NEW

```
//NSPersistentHistoryTransaction

open class NSPersistentHistoryTransaction : NSObject, NSCopying {
    open var timestamp: Date { get }
    open var changes: [NSPersistentHistoryChange]? { get }
    open var transactionNumber: Int64 { get }
    open var storeID: String { get }
    open var bundleID: String { get }
    open var processID: String { get }
    open var contextName: String? { get }
    open var author: String? { get }
    open var token: NSPersistentHistoryToken { get }

    // Get a notification that can be consumed by a NSManagedObjectContext
    open func objectIDNotification() -> Notification
}
```

NEW

```
//NSPersistentHistoryTransaction

open class NSPersistentHistoryTransaction : NSObject, NSCopying {
    open var timestamp: Date { get }
    open var changes: [NSPersistentHistoryChange]? { get }
    open var transactionNumber: Int64 { get }
    open var storeID: String { get }
    open var bundleID: String { get }
    open var processID: String { get }
    open var contextName: String? { get }
    open var author: String? { get }
    open var token: NSPersistentHistoryToken { get }

    // Get a notification that can be consumed by a NSManagedObjectContext
    open func objectIDNotification() -> Notification
}
```

NEW

```
//NSPersistentHistoryTransaction

open class NSPersistentHistoryTransaction : NSObject, NSCopying {
    open var timestamp: Date { get }
    open var changes: [NSPersistentHistoryChange]? { get }
    open var transactionNumber: Int64 { get }
    open var storeID: String { get }
    open var bundleID: String { get }
    open var processID: String { get }
    open var contextName: String? { get }
    open var author: String? { get }
    open var token: NSPersistentHistoryToken { get }

    // Get a notification that can be consumed by a NSManagedObjectContext
    open func objectIDNotification() -> Notification
}
```

NEW

```swift
//NSPersistentHistoryTransaction

open class NSPersistentHistoryTransaction : NSObject, NSCopying {
    open var timestamp: Date { get }
    open var changes: [NSPersistentHistoryChange]? { get }
    open var transactionNumber: Int64 { get }
    open var storeID: String { get }
    open var bundleID: String { get }
    open var processID: String { get }
    open var contextName: String? { get }
    open var author: String? { get }
    open var token: NSPersistentHistoryToken { get }

    // Get a notification that can be consumed by a NSManagedObjectContext
    open func objectIDNotification() -> Notification
}
```

NEW

```
//NSManagedObjectContext

/* Set the author for the context, this will be used as an identifier in
the Persistent History Transactions (NSPersistentHistoryTransaction) */
open var transactionAuthor: String?
```

NEW

```
//NSPersistentHistoryChange

public enum NSPersistentHistoryChangeType : Int {
    case insert
    case update
    case delete
}

open class NSPersistentHistoryChange : NSObject, NSCopying {
    open var changeID: Int64 { get }
    @NSCopying open var changedObjectID: NSManagedObjectID { get }
    open var changeType: NSPersistentHistoryChangeType { get }
    open var tombstone: [AnyHashable : Any]? { get }
    open var transaction: NSPersistentHistoryTransaction? { get }
    open var updatedProperties: Set<NSPropertyDescription>? { get }
}
```

NEW

```
//NSPersistentHistoryChange

public enum NSPersistentHistoryChangeType : Int {
    case insert
    case update
    case delete
}

open class NSPersistentHistoryChange : NSObject, NSCopying {
    open var changeID: Int64 { get }
    @NSCopying open var changedObjectID: NSManagedObjectID { get }
    open var changeType: NSPersistentHistoryChangeType { get }
    open var tombstone: [AnyHashable : Any]? { get }
    open var transaction: NSPersistentHistoryTransaction? { get }
    open var updatedProperties: Set<NSPropertyDescription>? { get }
}
```

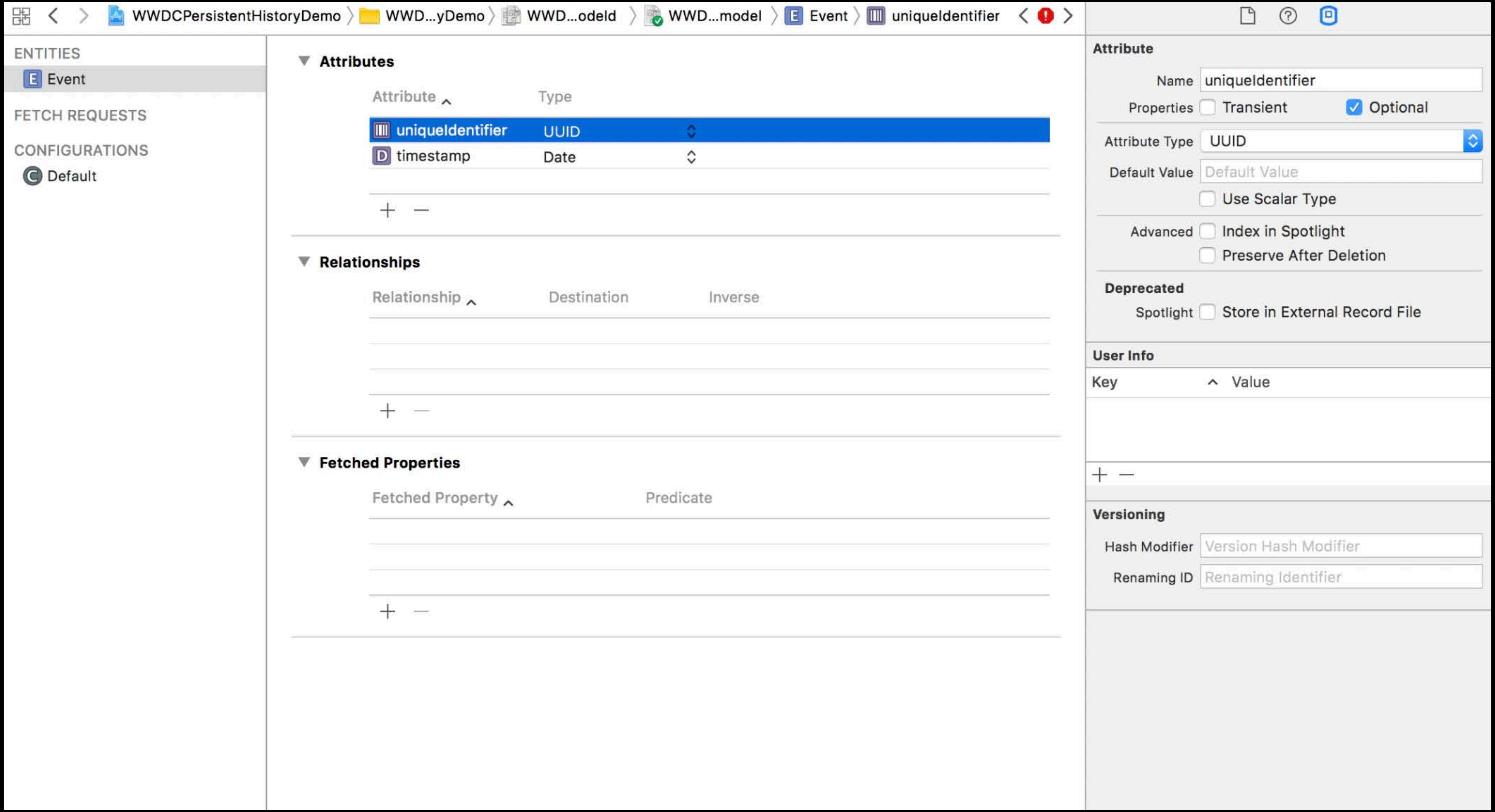
NEW

```
//NSPersistentHistoryChange

public enum NSPersistentHistoryChangeType : Int {
    case insert
    case update
    case delete
}

open class NSPersistentHistoryChange : NSObject, NSCopying {
    open var changeID: Int64 { get }
    @NSCopying open var changedObjectID: NSManagedObjectID { get }
    open var changeType: NSPersistentHistoryChangeType { get }
    open var tombstone: [AnyHashable : Any]? { get }
    open var transaction: NSPersistentHistoryTransaction? { get }
    open var updatedProperties: Set<NSPropertyDescription>? { get }
}
```

NEW

```
//NSPersistentHistoryChange

public enum NSPersistentHistoryChangeType : Int {
    case insert
    case update
    case delete
}

open class NSPersistentHistoryChange : NSObject, NSCopying {
    open var changeID: Int64 { get }
    @NSCopying open var changedObjectID: NSManagedObjectID { get }
    open var changeType: NSPersistentHistoryChangeType { get }
    open var tombstone: [AnyHashable : Any]? { get }
    open var transaction: NSPersistentHistoryTransaction? { get }
    open var updatedProperties: Set<NSPropertyDescription>? { get }
}
```

Model Editor



WWDCPersistentHistoryDemo 〉 WWD...yDemo 〉 WWD...odeld 〉 WWD...model 〉 Event 〉 uniqueIdentifier

ENTITIES
- E Event

FETCH REQUESTS

CONFIGURATIONS
- C Default

Attributes

Attribute	Type
uniqueIdentifier	UUID
timestamp	Date

Relationships

Relationship	Destination	Inverse

Fetched Properties

Fetched Property	Predicate

Attribute

Name: uniqueIdentifier
Properties: □ Transient ☑ Optional
Attribute Type: UUID
Default Value: Default Value
□ Use Scalar Type
Advanced: □ Index in Spotlight
□ Preserve After Deletion

Deprecated

Spotlight: □ Store in External Record File

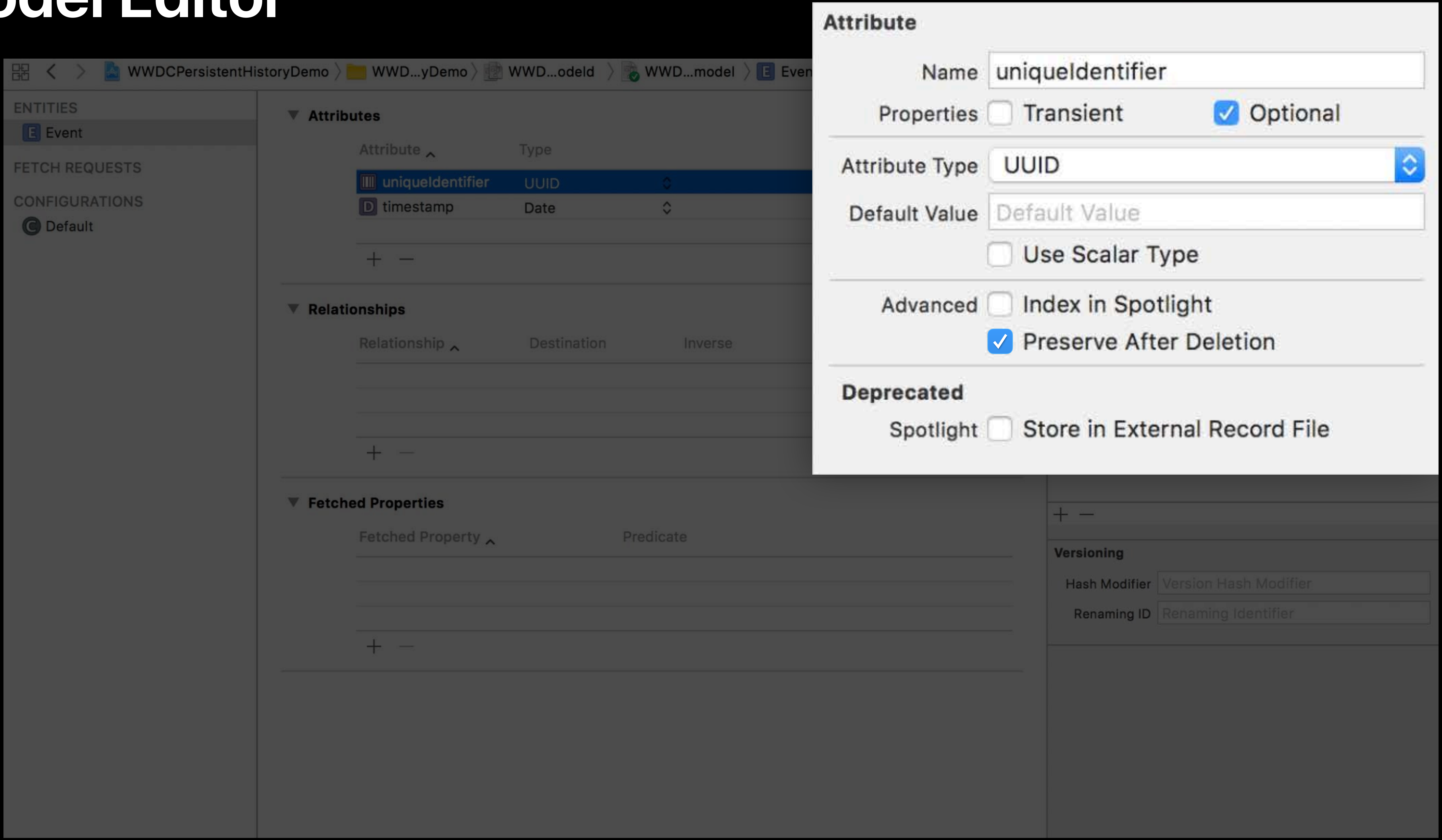
User Info

Key	Value

Versioning

Hash Modifier: Version Hash Modifier
Renaming ID: Renaming Identifier

Model Editor



Attributes

Attribute	Type
uniqueIdentifier	UUID
timestamp	Date

Attribute

Name: uniqueIdentifier

Properties: ☐ Transient ☑ Optional

Attribute Type: UUID

Default Value: Default Value

☐ Use Scalar Type

Advanced: ☐ Index in Spotlight

☑ Preserve After Deletion

Deprecated

Spotlight: ☐ Store in External Record File

Demo

History In Action!

Recap

History is fun!

Recap

History is fun!

Simple to enable Persistent History

Recap

History is fun!

Simple to enable Persistent History

Uncover missing changes

Recap

History is fun!

Simple to enable Persistent History

Uncover missing changes

Never be in the dark again!

Migration

Easy as pie

Migration

Easy as pie

We preserve as much history as possible

Migration

Easy as pie

We preserve as much history as possible

Except when removing:

Migration

Easy as pie

We preserve as much history as possible

Except when removing:

- Entities

Migration

Easy as pie

We preserve as much history as possible

Except when removing:

- Entities
- Tombstones

Migration

Easy as pie

We preserve as much history as possible

Except when removing:

- Entities
- Tombstones

We will preserve the history until the last transaction that is complete

Performance

Everything has consequences

Performance

Everything has consequences

Slight impact on save-and-batch operations

Performance

Everything has consequences

Slight impact on save-and-batch operations

An increase in memory during such operations

Performance

Everything has consequences

Slight impact on save-and-batch operations

An increase in memory during such operations

Small storage overhead, but history can be purged

Summary

Summary

CoreSpotlight—easily share your app data with Spotlight

Summary

CoreSpotlight—easily share your app data with Spotlight

New Indexing API—faster searches, faster apps

Summary

CoreSpotlight—easily share your app data with Spotlight

New Indexing API—faster searches, faster apps

Persistent History—always know “who dun it” https://bugreport.apple.com

More Information

https://developer.apple.com/wwdc17/210

Related Sessions

What’s New in Cocoa	Video	Online
What’s New in Foundation	Hall 2	Wednesday 11:00 AM
Build Better Apps with CloudKit Dashboard	Grand Ballroom B	Thursday 10:00 AM
What’s New in Core Spotlight for iOS and macOS	Grand Ballroom B	Thursday 4:10 PM
Cocoa Development Tips	Grand Ballroom B	Friday 9:00 AM

Labs

Cocoa Lab	Technology Lab C	Wed 11:00AM-1:00PM
Foundation Lab	Technology Lab C	Wed 1:00PM-3:30PM
Core Data Lab	Technology Lab C	Wed 3:10PM-6:00PM
CloudKit and iCloud Lab	Technology Lab C	Thu 11:00AM-1:00PM
Core Data Lab	Technology Lab H	Thu 4:10PM-6:00PM
CoreSpotlight and Search Lab	Technology Lab H	Fri 9:00AM-11:00AM
Cocoa Lab	Technology Lab B	Fri 1:50PM-3:20PM

WWDC17