Core Data

MatthewMorey.com | @xzolian

9:41 AM
LTE
41012
ST. AUGUSTINE, FL 40NM ENE OF...
Last Updated: Today at 8:50 AM
4.3 ft
8 secs
80.1 °F
E
8.9 mph
81.0 °F
ENE

BuoyExplorer.com

Wrist Presenter

WristPresenter.com

ChaiOne

Methodist

HOUSTON DYNAMO

DR

Agenda

Data Persistence

Data Persistence

- Raw Files

Raw Files

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSData *data = [fileManager contentsAtPath:
                                       @"/path/to/file.txt"];

...

[fileManager createFileAtPath:@"/path/to/file.txt"
                     contents:data
                   attributes:nil];

...

NSString *data = [NSString stringWithContentsOfFile:
                                         @"/path/to/file.txt"
                               encoding:NSUTF8StringEncoding
                                  error:nil];
```

Raw Files

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSData *data = [fileManager contentsAtPath:
                                       @"/path/to/file.txt"];


...

[fileManager createFileAtPath:@"/path/to/file.txt"
                     contents:data
                   attributes:nil];
```

```
...

NSString *data = [NSString stringWithContentsOfFile:
                                         @"/path/to/file.txt"
                                encoding:NSUTF8StringEncoding
                                   error:nil];
```

Raw Files

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSData *data = [fileManager contentsAtPath:
                                        @”/path/to/file.txt”];

...

[fileManager createFileAtPath:@”/path/to/file.txt”
                     contents:data
                   attributes:nil];

...
```

```
NSString *data = [NSString stringWithContentsOfFile:
                                          @”/path/to/file.txt”
                                       encoding:NSUTF8StringEncoding
                                          error:nil];
```

Data Persistence

- Raw Files
- NSPropertyListSerialization

NSPropertyListSerialization

```
NSDictionary *userDataDictionary = [NSPropertyListSerialization
                    propertyListWithData:userData
                                 options:NSPropertyListImmutable
                                  format:NULL
                                   error:nil];

...

NSDictionary *userDataDictionary = [[NSDictionary alloc]
                  initWithContentsOfFile:@"/path/to/file.plist"];
```

Data Persistence

- Raw Files
- NSPropertyListSerialization
- NSUserDefaults

NSUserDefaults

```
[[NSUserDefaults standardUserDefaults] setObject:data
                                            forKey:@"userData"];
[[NSUserDefaults standardUserDefaults] synchronize];

...

[[NSUserDefaults standardUserDefaults] stringForKey:
                                                   @"userData"];
```

NSUserDefaults

```
[[NSUserDefaults standardUserDefaults] setObject:data
                                         forKey:@"userData"];
[[NSUserDefaults standardUserDefaults] synchronize];

...

[[NSUserDefaults standardUserDefaults] stringForKey:
                                                @"userData"];
```

Data Persistence

- Raw Files
- NSPropertyListSerialization
- NSUserDefaults
- SQLite

SQLite

```
int sqlResult = sqlite3_prepare_v2(myDatabase, sql, -1, &statement, NULL);

if (sqlResult == SQLITE_OK) {

     sqlResult = sqlite3_step(statement); // Execute

    // Check the result for completion
    if(sqlResult == SQLITE_DONE) {
        ...
    }

    sqlite3_finalize(statement);
}
```

SQLite + FMDB

```
FMResultSet *s = [db executeQuery:@”SELECT * FROM myTable”];
while ([s next]) {

    //retrieve values for each record

}
```

SQLite + Friends

Gus Mueller - FMDB
https://github.com/ccgus/fmdb

Marco Arment - FCModel
https://github.com/marcoarment/FCModel

Yap Studios - Yap Database
https://github.com/yaptv/YapDatabase

Data Persistence

- Raw Files
- NSPropertyListSerialization
- NSUserDefaults
- SQLite
- NSCoding / NSKeyedArchiver

NSCoding / NSKeyedArchiver

```
// Archive
[NSKeyedArchiver archiveRootObject:books toFile:@"/path/to/archive"];

// Unarchive
[NSKeyedUnarchiver unarchiveObjectWithFile:@"/path/to/archive"];
```

```
// Archive
NSData *data = [NSKeyedArchiver archivedDataWithRootObject:books];
[[NSUserDefaults standardUserDefaults] setObject:data forKey:@"books"];

// Unarchive
NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"books"];
NSArray *books = [NSKeyedUnarchiver unarchiveObjectWithData:data];
```

NSCoding / NSKeyedArchiver

```
// Archive
[NSKeyedArchiver archiveRootObject:books toFile:@"/path/to/archive"];

// Unarchive
[NSKeyedUnarchiver unarchiveObjectWithFile:@"/path/to/archive"];
```

```
// Archive
NSData *data = [NSKeyedArchiver archivedDataWithRootObject:books];
[[NSUserDefaults standardUserDefaults] setObject:data forKey:@"books"];

// Unarchive
NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"books"];
NSArray *books = [NSKeyedUnarchiver unarchiveObjectWithData:data];
```

NSCoding / NSKeyedArchiver

```
// Archive
[NSKeyedArchiver archiveRootObject:books toFile:@"/path/to/archive"];

// Unarchive
[NSKeyedUnarchiver unarchiveObjectWithFile:@"/path/to/archive"];
```

```
// Archive
NSData *data = [NSKeyedArchiver archivedDataWithRootObject:books];
[[NSUserDefaults standardUserDefaults] setObject:data forKey:@"books"];

// Unarchive
NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"books"];
NSArray *books = [NSKeyedUnarchiver unarchiveObjectWithData:data];
```

NSCoding / NSKeyedArchiver

```
- (id)initWithCoder:(NSCoder *)decoder {

    self = [super init];
    if (!self) {
        return nil;
    }

    self.title = [decoder decodeObjectForKey:@"title"];
    ...

    return self;
}


- (void)encodeWithCoder:(NSCoder *)encoder {

    [encoder encodeObject:self.title forKey:@"title"];
    ...
}
```

NSCoding / NSKeyedArchiver

Mike Ash - Friday Q&A 2010-08-12: Implementing NSCoding
https://www.mikeash.com/pyblog/friday-qa-2010-08-12-implementing-nscoding.html

NSHipster - NSCoding / NSKeyed Archiver
http://nshipster.com/nscoding/

Mantle - Model framework for Cocoa and Cocoa Touch
https://github.com/MantleFramework/Mantle

Data Persistence

- Raw Files
- NSPropertyListSerialization
- NSUserDefaults
- SQLite
- NSCoding / NSKeyedArchiver
- Core Data

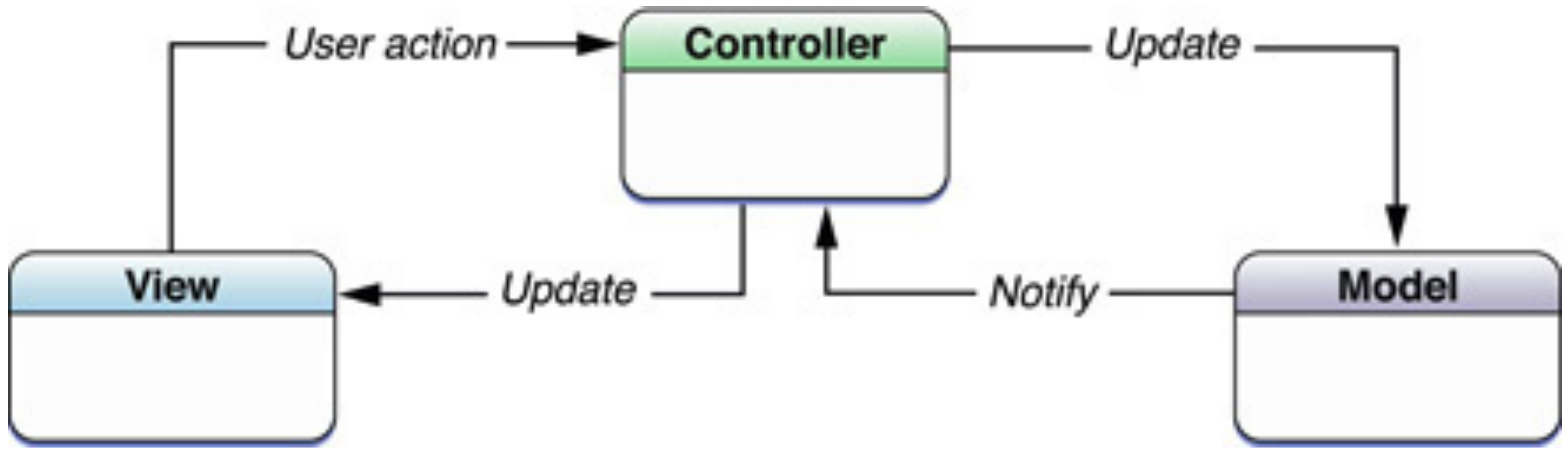
Core Data

Core Data vs NSKeyedArchiver

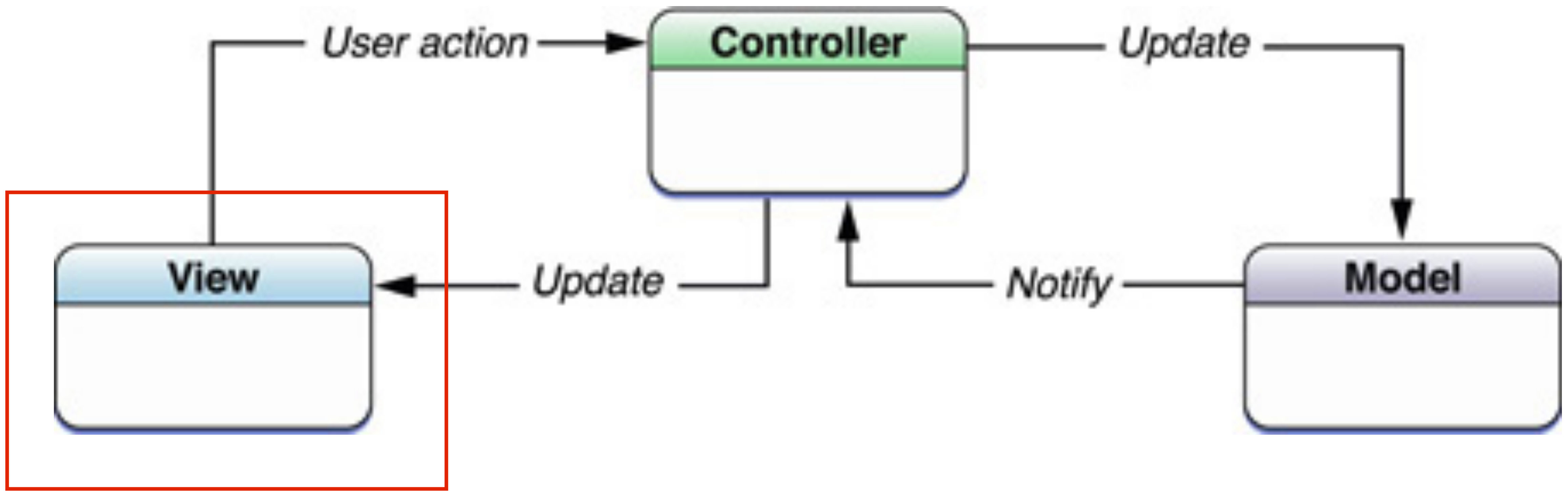
	Core Data	**NSKeyedArchiver**
Entity modeling	Yes	No
Querying	Yes	No
Speed	Fast	Slow
Serialization Format	SQLite, XML, NSData	NSData
Migrations	Automatic	Manual
Undo Manager	Automatic	Manual

NSHipster - NSCoding / NSKeyed Archiver
http://nshipster.com/nscoding/

MVC

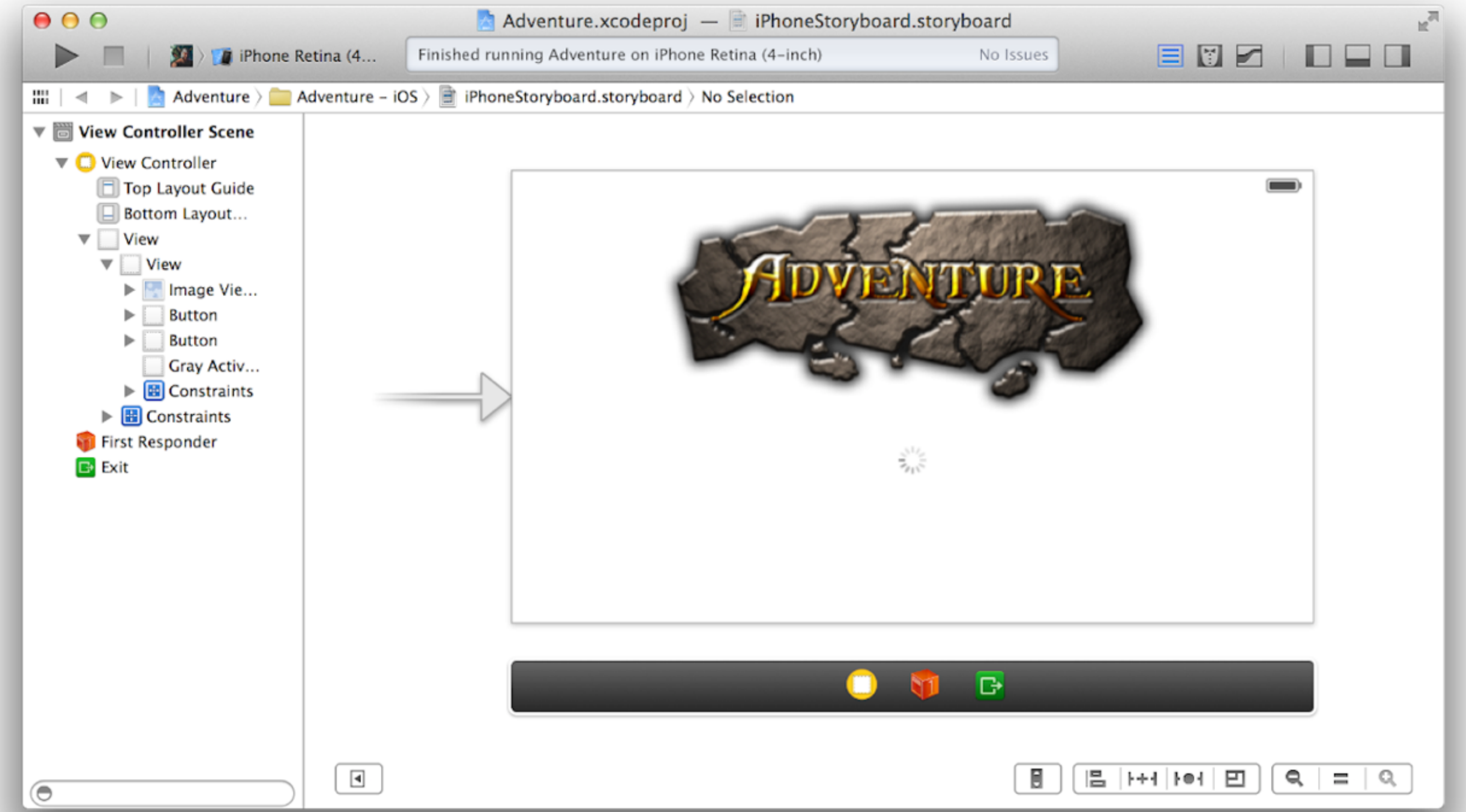


MVC

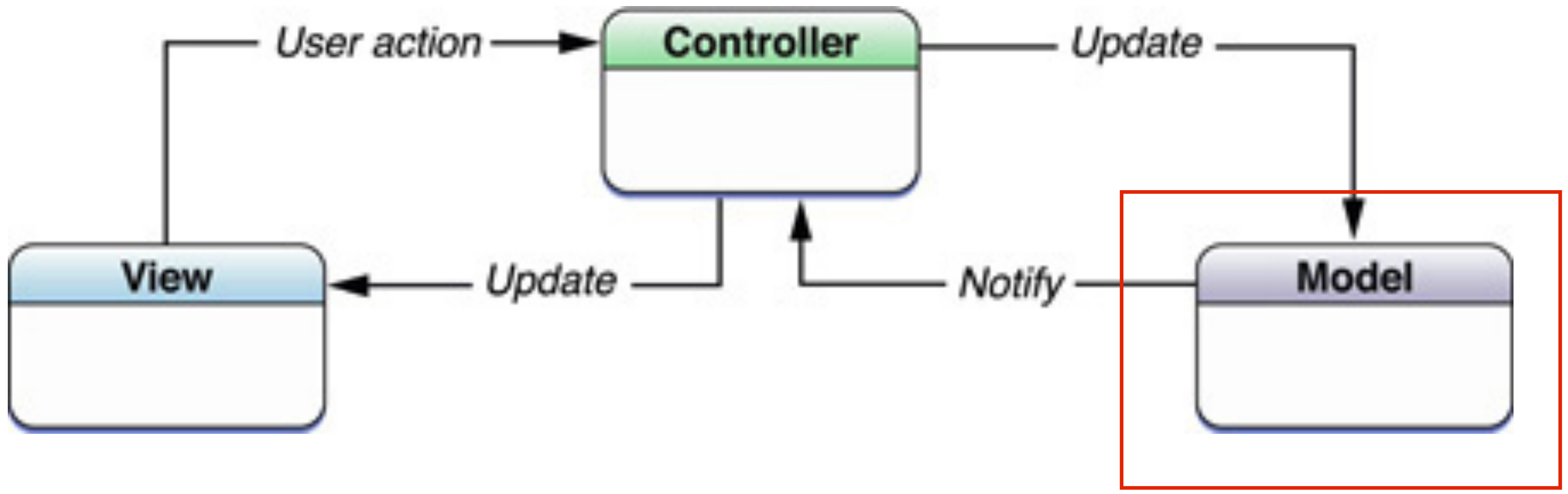


MVC

View – Interface Builder

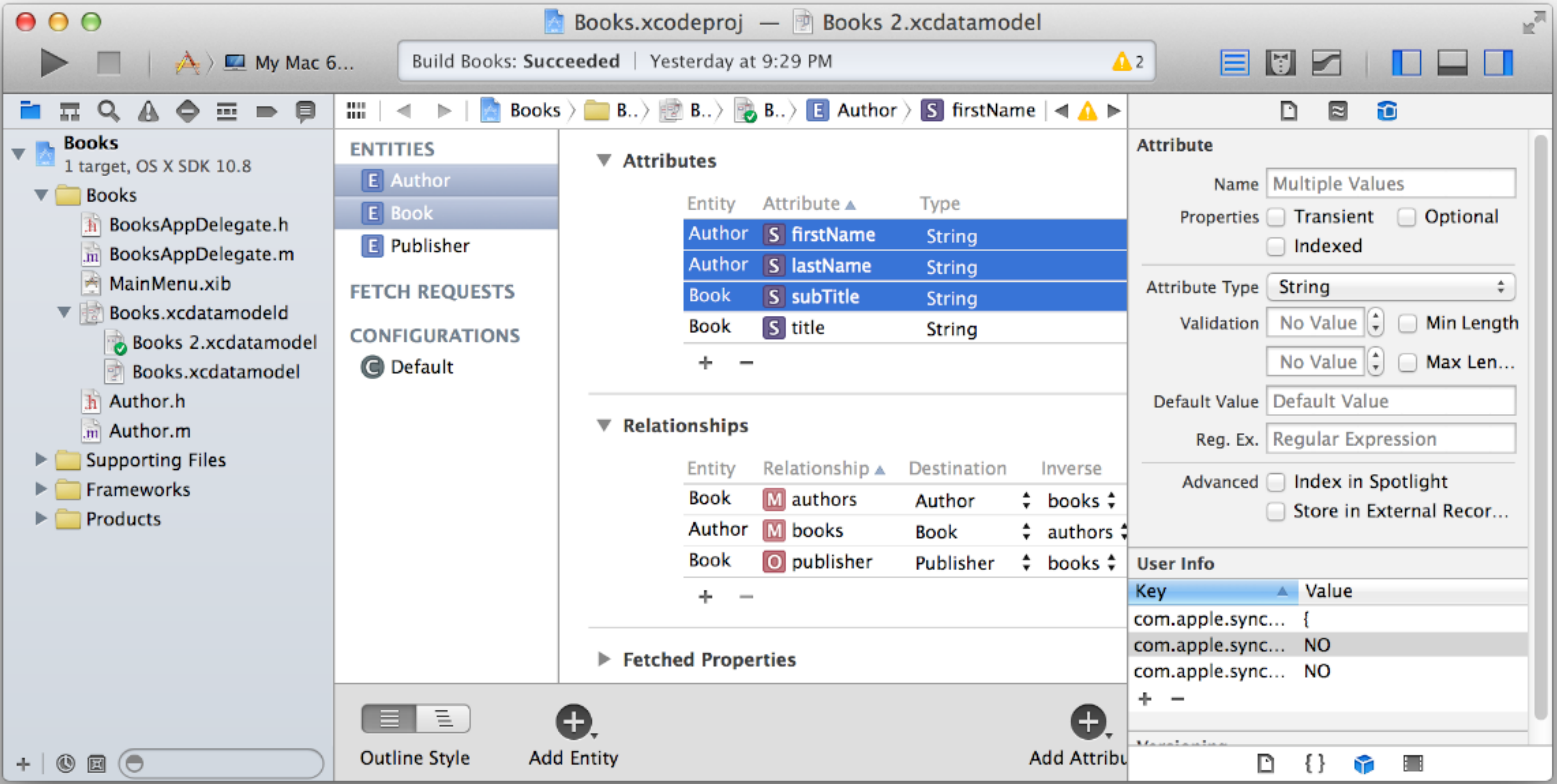


MVC



MVC

Model – Core Data Model Editor



Attributes

Entity	Attribute	Type
Author	S firstName	String
Author	S lastName	String
Book	S subTitle	String
Book	S title	String

Relationships

Entity	Relationship	Destination	Inverse
Book	M authors	Author	books
Author	M books	Book	authors
Book	O publisher	Publisher	books

Attribute

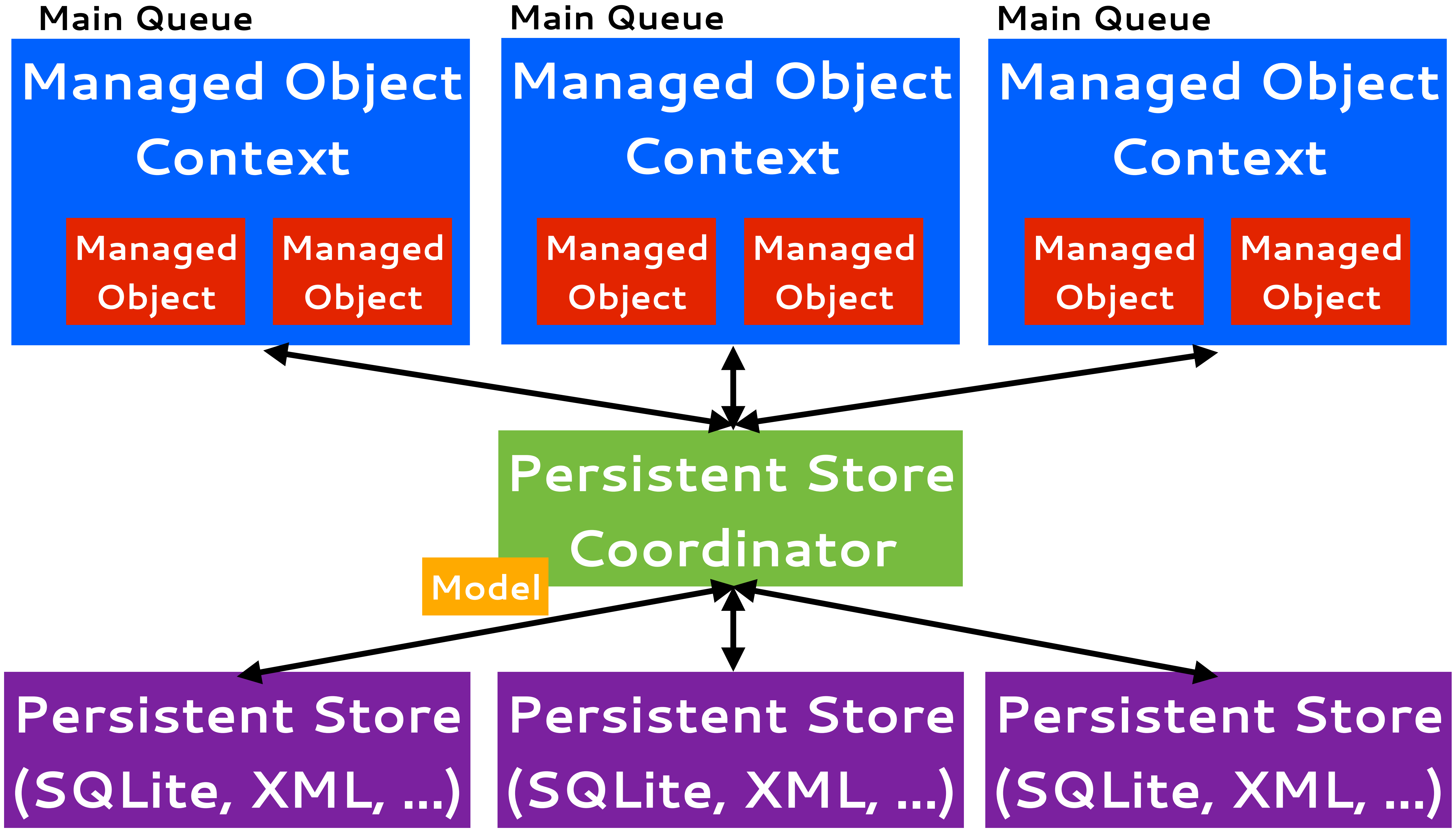
Name: Multiple Values
Properties: Transient, Optional, Indexed
Attribute Type: String
Validation: No Value, Min Length; No Value, Max Len…
Default Value: Default Value
Reg. Ex.: Regular Expression
Advanced: Index in Spotlight, Store in External Recor…

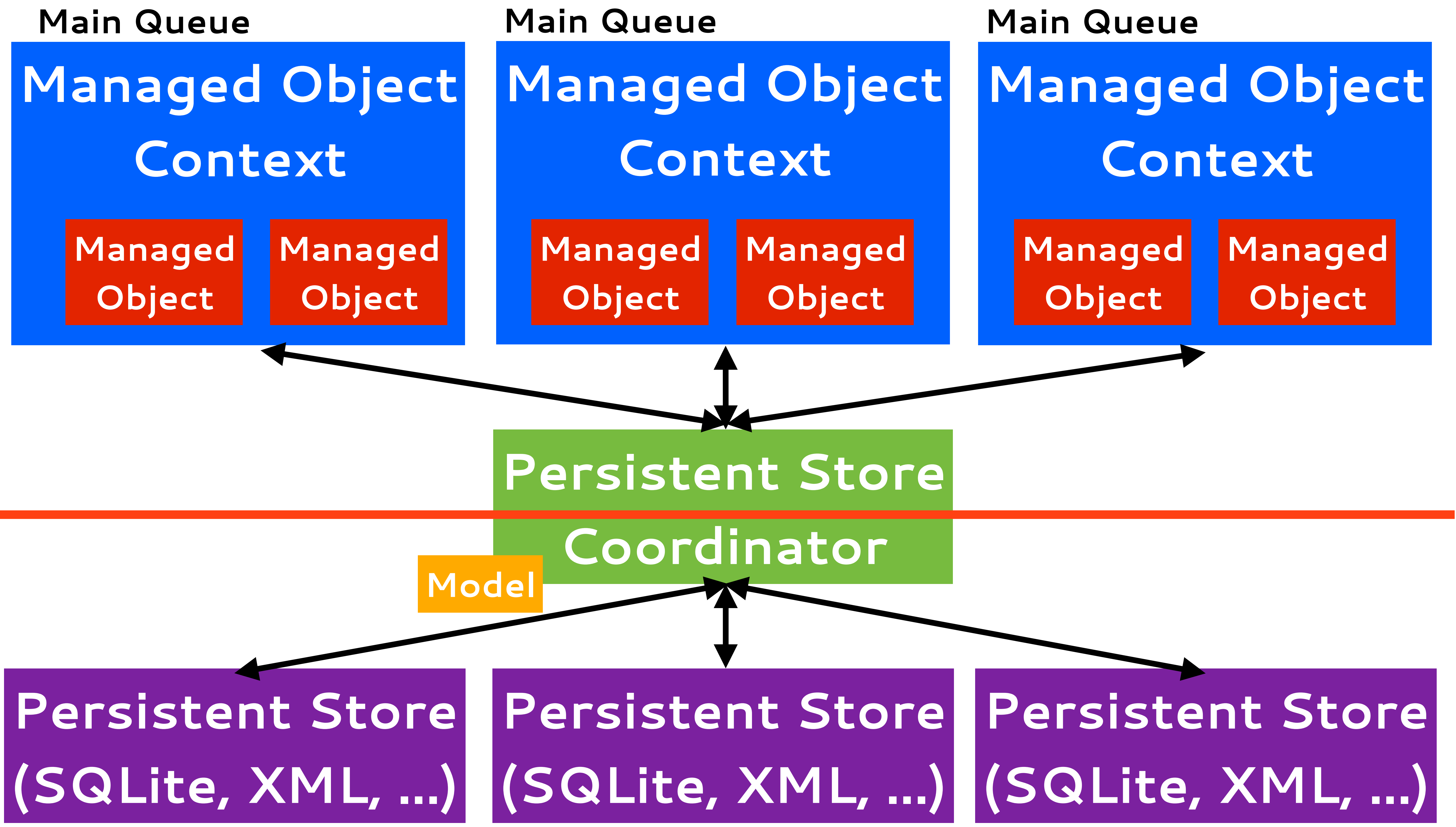
User Info

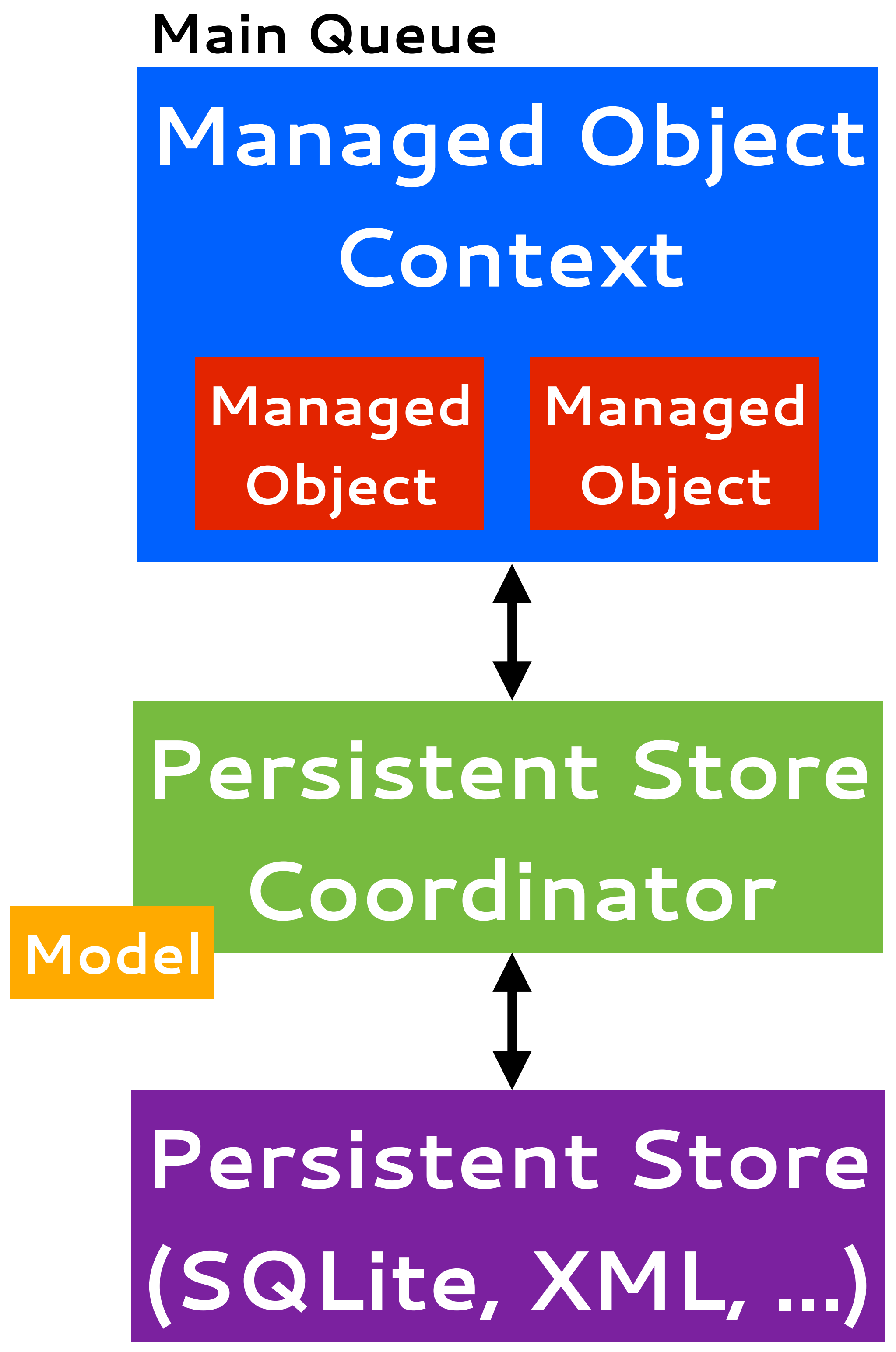
Key	Value
com.apple.sync…	{
com.apple.sync…	NO
com.apple.sync…	NO

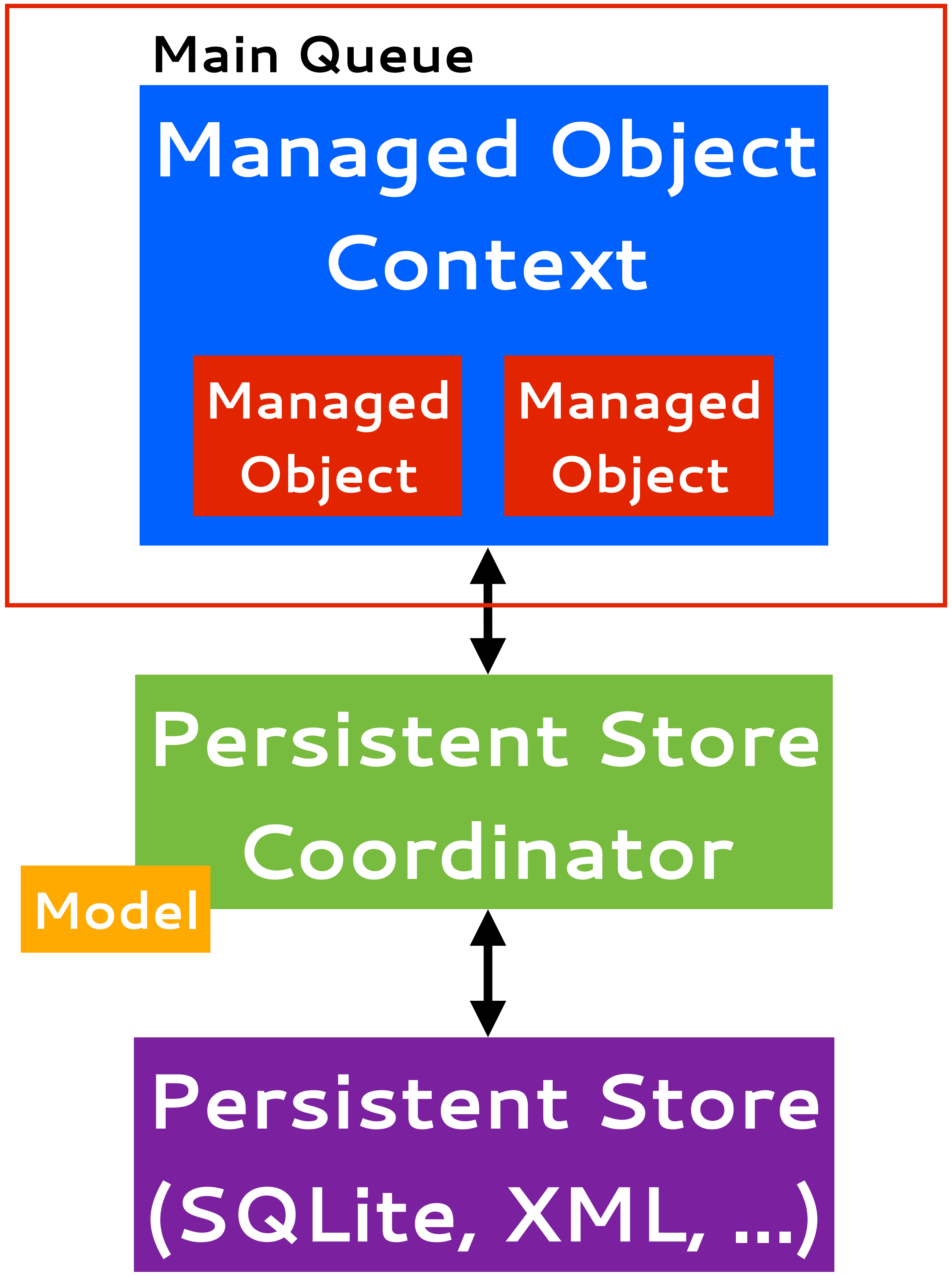
Core Data Is...

Core Data Is Not…


Main Queue
Managed Object Context
Managed Object
Managed Object
Main Queue
Managed Object Context
Managed Object
Managed Object
Main Queue
Managed Object Context
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, …)
Persistent Store (SQLite, XML, …)
Persistent Store (SQLite, XML, …)


Main Queue
Main Queue
Main Queue
Managed Object Context
Managed Object Context
Managed Object Context
Managed Object
Managed Object
Managed Object
Managed Object
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, ...)
Persistent Store (SQLite, XML, ...)
Persistent Store (SQLite, XML, ...)


Main Queue
Managed Object Context
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, …)


Main Queue
Managed Object Context
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, ...)

NSManagedObjectContext

```
self.managedObjectContext = [[NSManagedObjectContext alloc]
        initWithConcurrencyType:NSMainQueueConcurrencyType];

[self.managedObjectContext setPersistentStoreCoordinator:
                                 persistentStoreCoordinator];
```

NSManagedObjectContext

objectWithID:

executeFetchRequest:error:

countForFetchRequest:error:

deleteObject:

obtainPermanentIDsForObjects:error:

performBlock:

performBlockAndWait:

NSManagedObject

```
NSManagedObject *author;
NSString *name = [author valueForKey:@"name"];

...

[author setValue:@"John Smith" forKey:@"name"];
```

NSManagedObject

```
self.author.name = @”John Smith”;
```

NSManagedObject

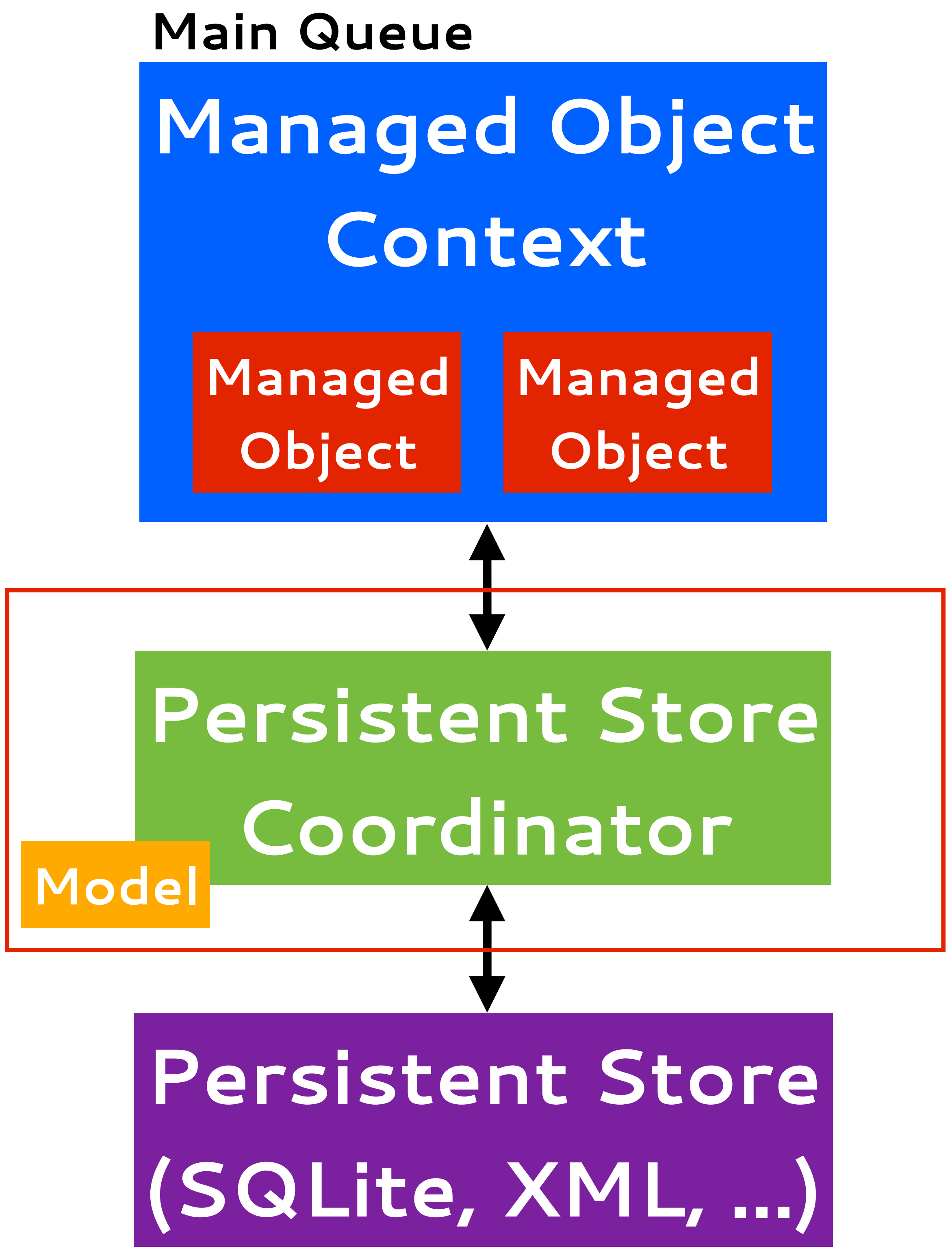
`hasChanges`

`isInserted`

`isUpdated`

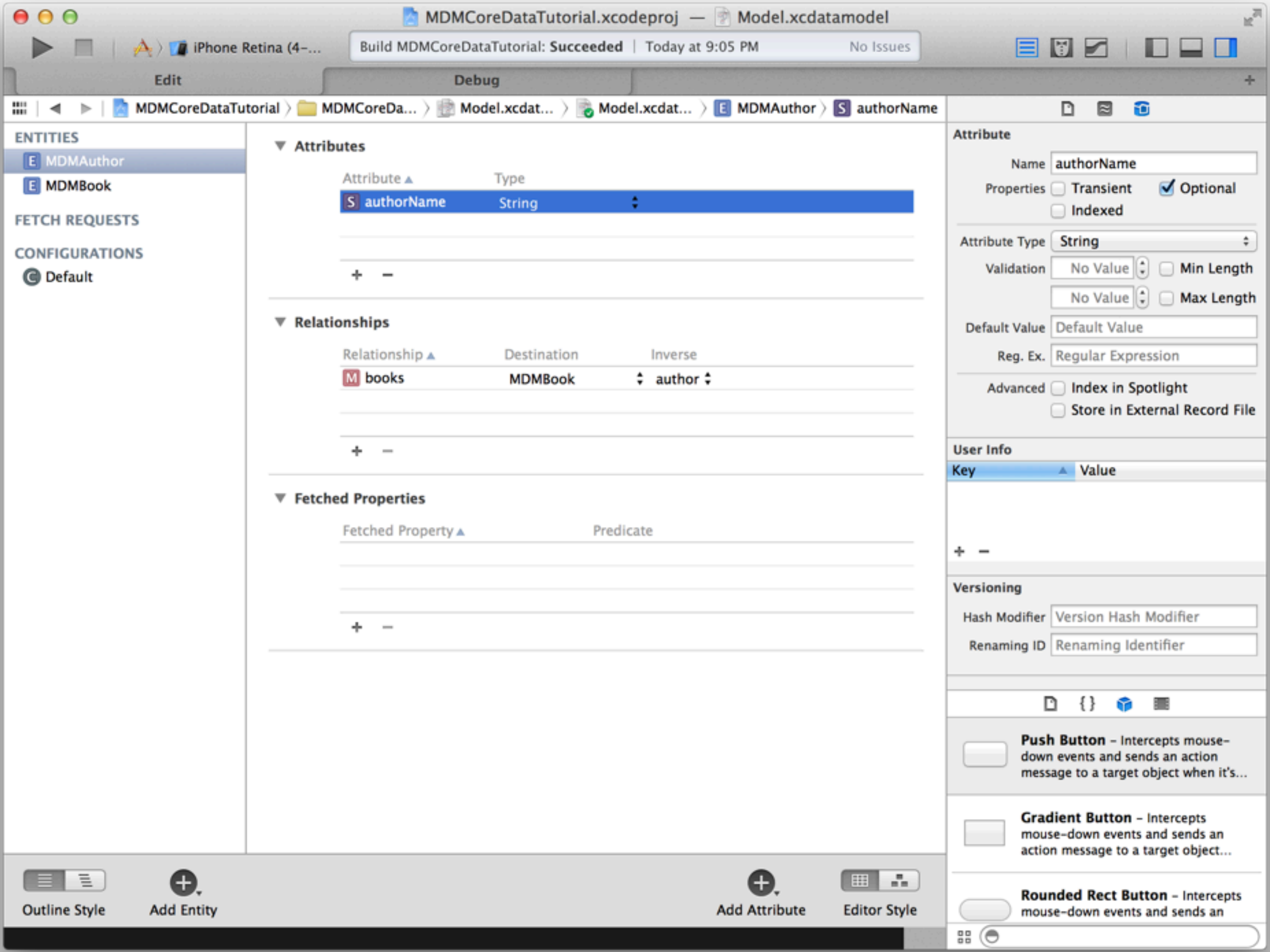
`isDeleted`

`isFault`


Main Queue
Managed Object Context
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, …)

NSManagedObjectModel

```
NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"model"
                                          withExtension:@"momd"];

NSManagedObjectModel *model = [[NSManagedObjectModel alloc]
                                initWithContentsOfURL:modelURL];
```



Attributes

Attribute	Type
authorName	String

Relationships

Relationship	Destination	Inverse
books	MDMBook	author

Fetched Properties

Fetched Property	Predicate

Attribute

Name: authorName
Properties: Transient [ ] Optional [✓] Indexed [ ]
Attribute Type: String
Validation: No Value Min Length [ ]
No Value Max Length [ ]
Default Value: Default Value
Reg. Ex.: Regular Expression
Advanced: Index in Spotlight [ ] Store in External Record File [ ]

User Info

Key	Value

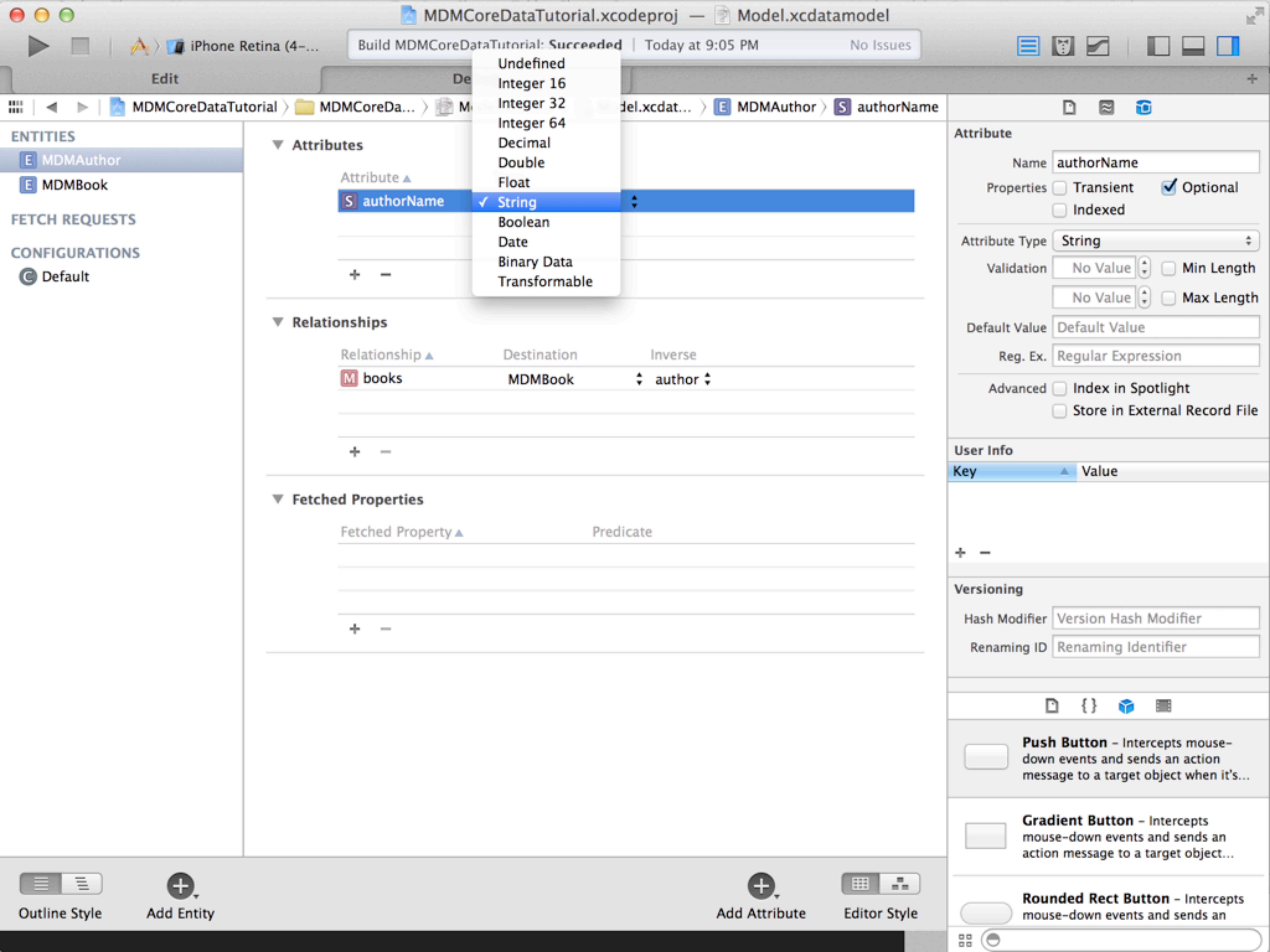
Versioning

Hash Modifier: Version Hash Modifier
Renaming ID: Renaming Identifier

Push Button – Intercepts mouse-down events and sends an action message to a target object when it's...

Gradient Button – Intercepts mouse-down events and sends an action message to a target object...

Rounded Rect Button – Intercepts mouse-down events and sends an


MDMCoreDataTutorial.xcodeproj — Model.xcdatamodel
iPhone Retina (4-...
Build MDMCoreDataTutorial: Succeeded | Today at 9:05 PM
No Issues
Edit
MDMCoreDataTutorial
MDMCoreDa...
del.xcdat...
MDMAuthor
authorName
ENTITIES
MDMAuthor
MDMBook
FETCH REQUESTS
CONFIGURATIONS
Default
Undefined
Integer 16
Integer 32
Integer 64
Decimal
Double
Float
✓ String
Boolean
Date
Binary Data
Transformable
Outline Style
Add Entity
Add Attribute
Editor Style
Push Button – Intercepts mouse-down events and sends an action message to a target object when it's...
Gradient Button – Intercepts mouse-down events and sends an action message to a target object...
Rounded Rect Button – Intercepts mouse-down events and sends an

Attributes

Attribute	Type
authorName	String

Relationships

Relationship	Destination	Inverse
books	MDMBook	author

Fetched Properties

Fetched Property	Predicate

Attribute

Name	authorName
Properties	Transient [✓ Optional] Indexed
Attribute Type	String
Validation	No Value Min Length; No Value Max Length
Default Value	Default Value
Reg. Ex.	Regular Expression
Advanced	Index in Spotlight; Store in External Record File

User Info

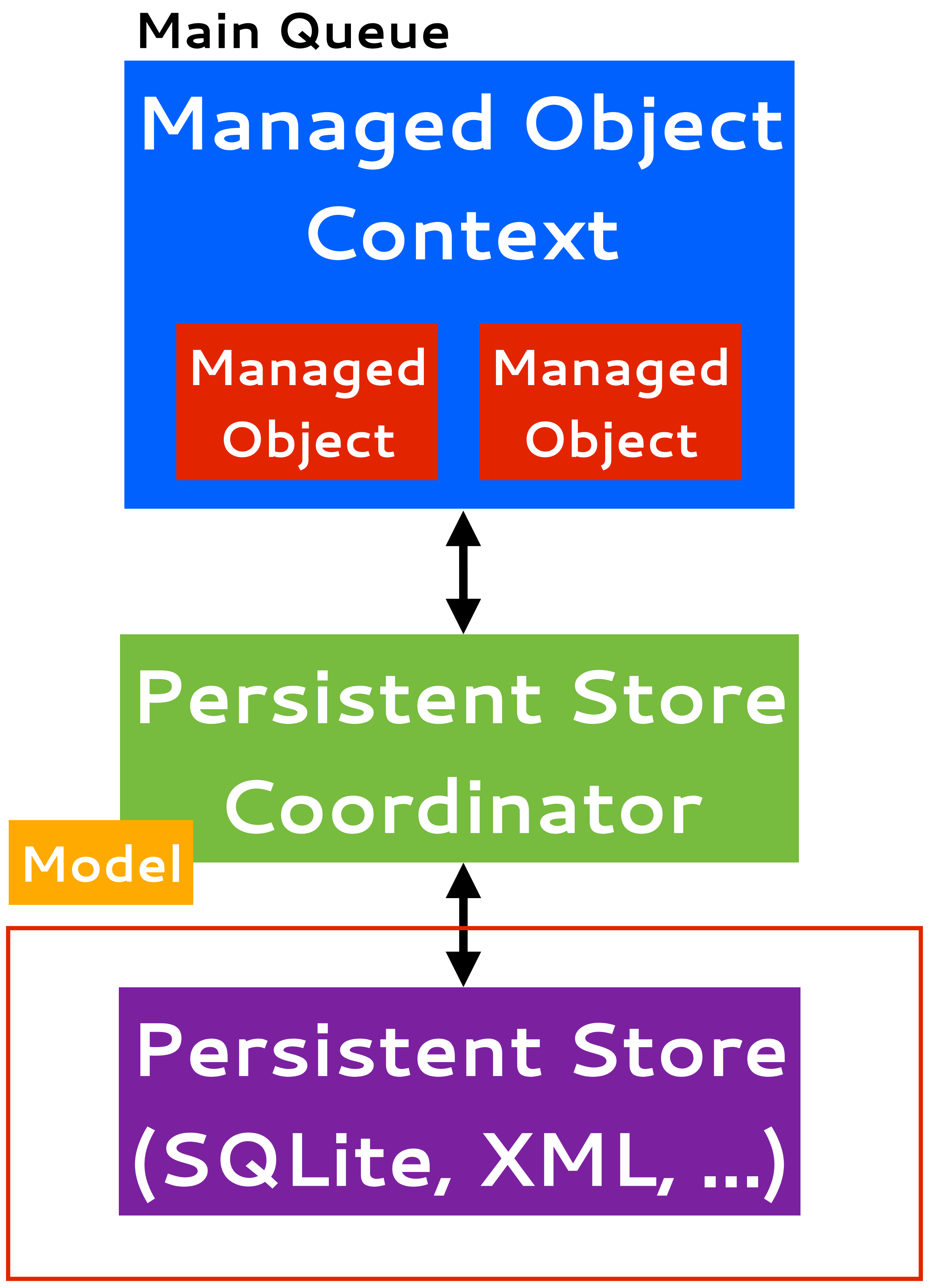
Key	Value

Versioning

Hash Modifier	Version Hash Modifier
Renaming ID	Renaming Identifier

NSPersistentStoreCoordinator

```
NSPersistentStoreCoordinator *persistentStoreCoordinator =

[[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:
                                                      self.model];
```


Main Queue
Managed Object Context
Managed Object
Managed Object
Persistent Store Coordinator
Model
Persistent Store (SQLite, XML, ...)

NSPersistentStore

```
NSError *persistentStoreError;
NSPersistentStore *persistentStore = [persistentStoreCoordinator
                    addPersistentStoreWithType:NSSQLiteStoreType
                                 configuration:nil
                                           URL:self.storeURL
                                       options:persistentStoreOptions
                                         error:&persistentStoreError];
```

Core Data Stack

1. Create data model

2. Create persistent store coordinator with model

3. Add persistent store to persistent store coordinator

4. Create managed object context and set it’s persistent store coordinator

Core Data Stack

```
- (NSManagedObjectContext *)managedObjectContext {

    if (_managedObjectContext == nil) {

        NSManagedObjectModel *mom = [NSManagedObjectModel mergedModelFromBundles:[NSBundle allBundles]];

        NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc]
                                                                initWithManagedObjectModel:mom];

        NSURL *persistentStoreURL = [[[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                                    inDomains:NSUserDomainMask] lastObject]
                                                           URLByAppendingPathComponent:@"Database.sqlite"];
        NSError *error;
        NSPersistentStore *persistentStore = [psc addPersistentStoreWithType:NSSQLiteStoreType
                                                               configuration:nil
                                                                         URL:persistentStoreURL
                                                                     options:nil
                                                                       error:&error];
        // Handle Potential Error
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithConcurrencyType:
                                                                     NSMainQueueConcurrencyType];
        [_managedObjectContext setPersistentStoreCoordinator:psc];
    }

    return _managedObjectContext;
}
```

Core Data Stack

```
- (NSManagedObjectContext *)managedObjectContext {
    if (_managedObjectContext == nil) {
        NSManagedObjectModel *mom = [NSManagedObjectModel mergedModelFromBundles:[NSBundle allBundles]];
        NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc]
                                                                      initWithManagedObjectModel:mom];

        NSURL *persistentStoreURL = [[[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                                 inDomains:NSUserDomainMask] lastObject]
                                                        URLByAppendingPathComponent:@"Database.sqlite"];
        NSError *error;
        NSPersistentStore *persistentStore = [psc addPersistentStoreWithType:NSSQLiteStoreType
                                                            configuration:nil
                                                                      URL:persistentStoreURL
                                                                  options:nil
                                                                    error:&error];
        // Handle Potential Error
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithConcurrencyType:
                                                                   NSMainQueueConcurrencyType];
        [_managedObjectContext setPersistentStoreCoordinator:psc];
    }

    return _managedObjectContext;
}
```

Core Data Stack

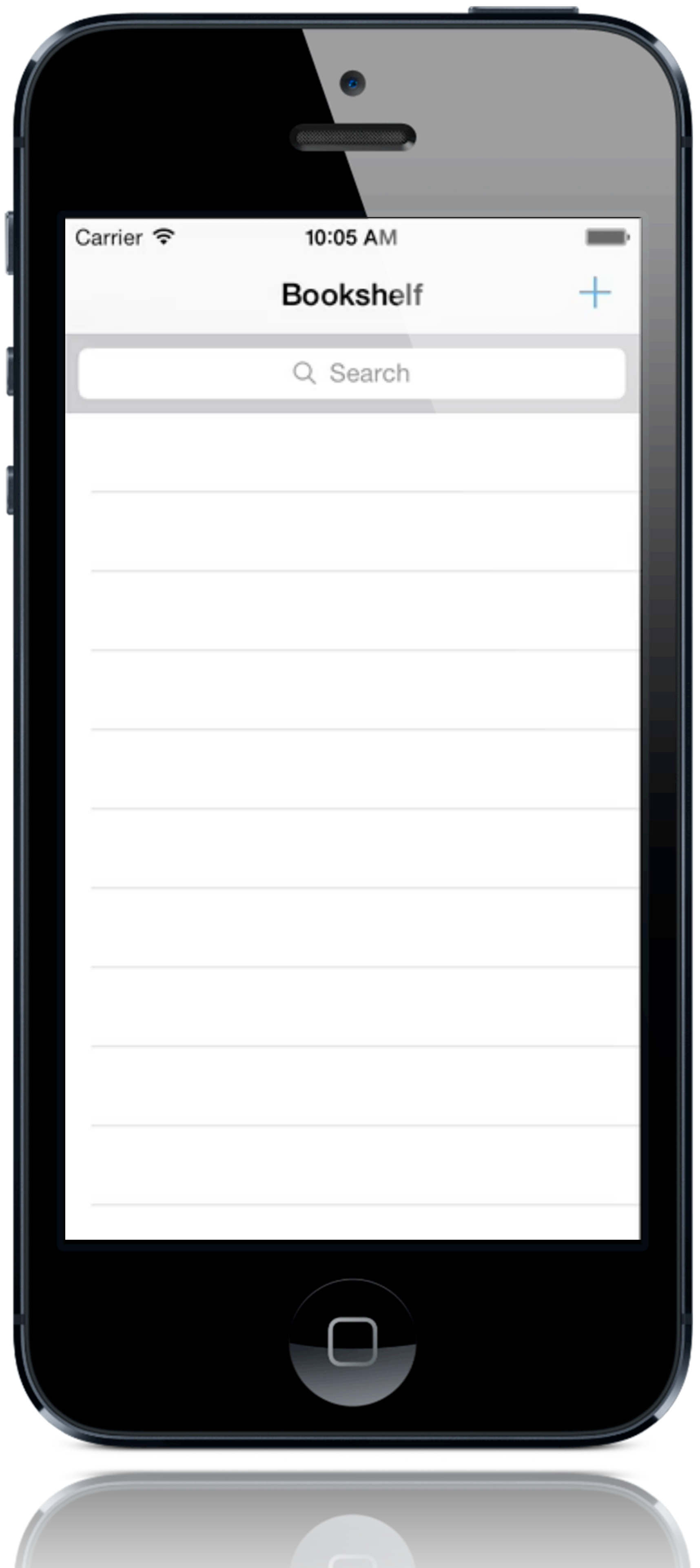
```
- (NSManagedObjectContext *)managedObjectContext {

    if (_managedObjectContext == nil) {

        NSManagedObjectModel *mom = [NSManagedObjectModel mergedModelFromBundles:[NSBundle allBundles]];

        NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc]
                                                                 initWithManagedObjectModel:mom];

        NSURL *persistentStoreURL = [[[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                                inDomains:NSUserDomainMask] lastObject]
                                                       URLByAppendingPathComponent:@"Database.sqlite"];
        NSError *error;
        NSPersistentStore *persistentStore = [psc addPersistentStoreWithType:NSSQLiteStoreType
                                                                configuration:nil
                                                                          URL:persistentStoreURL
                                                                      options:nil
                                                                        error:&error];
        // Handle Potential Error
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithConcurrencyType:
                                                                       NSMainQueueConcurrencyType];
        [_managedObjectContext setPersistentStoreCoordinator:psc];
    }

    return _managedObjectContext;
}
```

Core Data Stack

```
- (NSManagedObjectContext *)managedObjectContext {
    if (_managedObjectContext == nil) {
        NSManagedObjectModel *mom = [NSManagedObjectModel mergedModelFromBundles:[NSBundle allBundles]];

        NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc]
                                                                     initWithManagedObjectModel:mom];

        NSURL *persistentStoreURL = [[[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                                  inDomains:NSUserDomainMask] lastObject]
                                                         URLByAppendingPathComponent:@"Database.sqlite"];
        NSError *error;
        NSPersistentStore *persistentStore = [psc addPersistentStoreWithType:NSSQLiteStoreType
                                                               configuration:nil
                                                                         URL:persistentStoreURL
                                                                     options:nil
                                                                       error:&error];
        // Handle Potential Error
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithConcurrencyType:
                                                                           NSMainQueueConcurrencyType];
        [_managedObjectContext setPersistentStoreCoordinator:psc];
    }

    return _managedObjectContext;
}
```

Core Data Stack

```
- (NSManagedObjectContext *)managedObjectContext {

    if (_managedObjectContext == nil) {

        NSManagedObjectModel *mom = [NSManagedObjectModel mergedModelFromBundles:[NSBundle allBundles]];

        NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc]
                                                                  initWithManagedObjectModel:mom];

        NSURL *persistentStoreURL = [[[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                             inDomains:NSUserDomainMask] lastObject]
                                                    URLByAppendingPathComponent:@"Database.sqlite"];
        NSError *error;
        NSPersistentStore *persistentStore = [psc addPersistentStoreWithType:NSSQLiteStoreType
                                                               configuration:nil
                                                                         URL:persistentStoreURL
                                                                     options:nil
                                                                       error:&error];
        // Handle Potential Error
        _managedObjectContext = [[NSManagedObjectContext alloc] initWithConcurrencyType:
                                                                        NSMainQueueConcurrencyType];
        [_managedObjectContext setPersistentStoreCoordinator:psc];
    }

    return _managedObjectContext;
}
```


Carrier
10:05 AM
Bookshelf
+
Search

Demo

Fetch Request

Fetch Request

```
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc]
                                               initWithEntityName:@"MDMBook"];
NSSortDescriptor *sortDescriptor = [NSSortDescriptor
                                                sortDescriptorWithKey:@"title"
                                                            ascending:YES];
[fetchRequest setSortDescriptors:@[sortDescriptor]];

NSError *fetchError;
NSArray *results = [self.managedObjectContext executeFetchRequest:fetchRequest
                                                            error:&fetchError];

if (results == nil) {

    NSLog(@"Error: %@", [fetchError localizedDescription]);
} else if ([results count] > 0) {

    [self.tableDatasource addObjectsFromArray:results];
}
```

Fetch Request

```
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc]
                                                initWithEntityName:@"MDMBook"];
NSSortDescriptor *sortDescriptor = [NSSortDescriptor
                                                 sortDescriptorWithKey:@"title"
                                                             ascending:YES];
[fetchRequest setSortDescriptors:@[sortDescriptor]];

NSError *fetchError;
NSArray *results = [self.managedObjectContext executeFetchRequest:fetchRequest
                                                            error:&fetchError];

if (results == nil) {

    NSLog(@"Error: %@", [fetchError localizedDescription]);
} else if ([results count] > 0) {

    [self.tableDatasource addObjectsFromArray:results];
}
```

Sort Descriptor

```
NSSortDescriptor *sortDescriptor =
  [NSSortDescriptor sortDescriptorWithKey:@"title"
                               ascending:YES];
```

```
[fetchRequest setSortDescriptors:@[sortDescriptor]];
```

Sort Descriptor

```
NSSortDescriptor *sortDescriptor =
  [NSSortDescriptor sortDescriptorWithKey:@”title”
                                     ascending:YES];

[fetchRequest setSortDescriptors:@[sortDescriptor]];
```

Sort Descriptor

Apple Class Reference - NSSortDescriptor
https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSSortDescriptor_Class/Reference/Reference.html

NSHipster - NSSortDescriptor
http://nshipster.com/nssortdescriptor/

Predicate

```
NSPredicate *predicate =
  [NSPredicate predicateWithFormat:
    @”author.authorName == %@”, @”Charles Dickens”];
```

```
[fetchRequest setPredicate:predicate];
```

Predicate

Introduction to Predicates Programming Guide

https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Predicates/predicates.html

Fetch Request Controller

Fetch Request Controller Delegate

```
– controllerWillChangeContent:

– controller:didChangeObject:atIndexPath:forChangeType:newIndexPath:

– controller:didChangeSection:atIndex:forChangeType:

– controllerDidChangeContent:
```

Fetch Request Controller

Ash Furrow - How To Use **NSFetchedResultsController** with **UICollectionView**
http://ashfurrow.com/blog/how-to-use-nsfetchedresultscontroller-with-uicollectionview

BJ Miller - Using **NSFetchedResultsController** with an **MKMapView**
http://bjmiller.me/post/58431532849/nsfetchedresultscontroller-with-mkmapview

Demo

New NSManagedObjects

New NSManagedObjects

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

    MDMDetailViewController *detailViewController = segue.destinationViewController;
    NSEntityDescription *authorEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMAuthor"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *author = [[NSManagedObject alloc]
                                           initWithEntity:authorEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];



    NSEntityDescription *bookEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMBook"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *book = [[NSManagedObject alloc]
                                           initWithEntity:bookEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];
    [book setValue:author forKey:@"author"];



    detailViewController.book = book;
}
```

New NSManagedObjects

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

    MDMDetailViewController *detailViewController = segue.destinationViewController;
    NSEntityDescription *authorEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMAuthor"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *author = [[NSManagedObject alloc]
                                           initWithEntity:authorEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];


    NSEntityDescription *bookEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMBook"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *book = [[NSManagedObject alloc]
                                           initWithEntity:bookEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];
    [book setValue:author forKey:@"author"];


    detailViewController.book = book;
}
```

New NSManagedObjects

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

    MDMDetailViewController *detailViewController = segue.destinationViewController;
    NSEntityDescription *authorEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMAuthor"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *author = [[NSManagedObject alloc]
                                           initWithEntity:authorEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];



    NSEntityDescription *bookEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMBook"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *book = [[NSManagedObject alloc]
                                           initWithEntity:bookEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];
    [book setValue:author forKey:@"author"];



    detailViewController.book = book;
}
```

New NSManagedObjects

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

    MDMDetailViewController *detailViewController = segue.destinationViewController;
    NSEntityDescription *authorEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMAuthor"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *author = [[NSManagedObject alloc]
                                           initWithEntity:authorEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];



    NSEntityDescription *bookEntityDescription = [NSEntityDescription
                                            entityForName:@"MDMBook"
                                   inManagedObjectContext:self.managedObjectContext];
    NSManagedObject *book = [[NSManagedObject alloc]
                                           initWithEntity:bookEntityDescription
                           insertIntoManagedObjectContext:self.managedObjectContext];
    [book setValue:author forKey:@"author"];


    detailViewController.book = book;
}
```

NSManagedObjects Subclass

NSManagedObjects Subclass

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

  MDMAuthor *author = [MDMAuthor
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];


  MDMBook *book = [MDMBook
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];


  book.author = author;


  MDMDetailViewController *detailViewController = segue.destinationViewController;
  detailViewController.book = book;
}
```

NSManagedObjects Subclass

```
} else if ([segue.identifier isEqualToString:@"NewBookSegue"]) {

  MDMAuthor *author = [MDMAuthor
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];

  MDMBook *book = [MDMBook
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];

  book.author = author;

  MDMDetailViewController *detailViewController = segue.destinationViewController;
  detailViewController.book = book;
}
```

NSManagedObjects Subclass

```
} else if ([segue.identifier isEqualToString:@”NewBookSegue”]) {

  MDMAuthor *author = [MDMAuthor
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];


  MDMBook *book = [MDMBook
        MDMCoreDataAdditionsInsertNewObjectIntoContext:self.managedObjectContext];

  book.author = author;

  MDMDetailViewController *detailViewController = segue.destinationViewController;
  detailViewController.book = book;
}
```

NSManagedObjects Category

```
+ (NSString *)MDMCoreDataAdditionsEntityName {

    return NSStringFromClass(self);
}



+ (instancetype)MDMCoreDataAdditionsInsertNewObjectIntoContext:
                                                (NSManagedObjectContext *)context {

    return [NSEntityDescription insertNewObjectForEntityForName:[self
                    MDMCoreDataAdditionsEntityName] inManagedObjectContext:context];
}
```

NSManagedObjects Category

```
+ (NSString *)MDMCoreDataAdditionsEntityName {

    return NSStringFromClass(self);
}
```

```
+ (instancetype)MDMCoreDataAdditionsInsertNewObjectIntoContext:
                                                 (NSManagedObjectContext *)context {

    return [NSEntityDescription insertNewObjectForEntityForName:[self
                        MDMCoreDataAdditionsEntityName] inManagedObjectContext:context];
}
```

NSManagedObjects Category

```
+ (NSString *)MDMCoreDataAdditionsEntityName {

    return NSStringFromClass(self);
}
```

```
+ (instancetype)MDMCoreDataAdditionsInsertNewObjectIntoContext:
                                                (NSManagedObjectContext *)context {

    return [NSEntityDescription insertNewObjectForEntityForName:[self
                        MDMCoreDataAdditionsEntityName] inManagedObjectContext:context];
}
```

Demo

Deleting NSManagedObjects

Deleting NSManagedObjects

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath {

    if (editingStyle == UITableViewCellEditingStyleDelete) {

      [self.managedObjectContext deleteObject:
        [self.fetchedResultsController objectAtIndexPath:indexPath]];

      NSError *error;
      if (![self.managedObjectContext save:&error]) {
        NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
        abort();
      }
    }
}
```

Deleting NSManagedObjects

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath {

    if (editingStyle == UITableViewCellEditingStyleDelete) {

      [self.managedObjectContext deleteObject:
        [self.fetchedResultsController objectAtIndexPath:indexPath]];

      NSError *error;
      if (![self.managedObjectContext save:&error]) {
        NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
        abort();
      }
    }
}
```

Deleting NSManagedObjects

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath {

    if (editingStyle == UITableViewCellEditingStyleDelete) {

      [self.managedObjectContext deleteObject:
        [self.fetchedResultsController objectAtIndexPath:indexPath]];

      NSError *error;
      if (![self.managedObjectContext save:&error]) {
        NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
        abort();
      }
    }
}
```

Demo

Searching Core Data

Searching Core Data

```
- (void)searchForText:(NSString *)searchText {

    if (self.managedObjectContext) {

        NSPredicate *predicate = [NSPredicate predicateWithFormat:
                    @"%K BEGINSWITH[cd] %@", @"title", searchText];
        [self.searchFetchRequest setPredicate:predicate];


        NSError *fetchError = nil;
        NSArray *results = [self.managedObjectContext
          executeFetchRequest:self.searchFetchRequest error:&fetchError];


        if (results == nil) {
            NSLog(@"Error: %@", [fetchError localizedDescription]);
        } else {
            self.filteredList = results;
        }
    }
}
```

Searching Core Data

```
- (void)searchForText:(NSString *)searchText {

    if (self.managedObjectContext) {

        NSPredicate *predicate = [NSPredicate predicateWithFormat:
                    @"%K BEGINSWITH[cd] %@", @"title", searchText];
        [self.searchFetchRequest setPredicate:predicate];

        NSError *fetchError = nil;
        NSArray *results = [self.managedObjectContext
          executeFetchRequest:self.searchFetchRequest error:&fetchError];


        if (results == nil) {
            NSLog(@"Error: %@", [fetchError localizedDescription]);
        } else {
            self.filteredList = results;
        }
    }
}
```

Searching Core Data

```
- (void)searchForText:(NSString *)searchText {

    if (self.managedObjectContext) {

        NSPredicate *predicate = [NSPredicate predicateWithFormat:
                    @"%K BEGINSWITH[cd] %@", @"title", searchText];
        [self.searchFetchRequest setPredicate:predicate];


        NSError *fetchError = nil;
        NSArray *results = [self.managedObjectContext
          executeFetchRequest:self.searchFetchRequest error:&fetchError];


        if (results == nil) {
            NSLog(@"Error: %@", [fetchError localizedDescription]);
        } else {
            self.filteredList = results;
        }
    }
}
```

Core Data Errors

Core Data Errors

```
- (IBAction)saveButtonTapped:(id)sender {

    self.book.title = self.titleTextField.text;
    self.book.author.authorName = self.authorTextField.text;

    NSError *saveError;
    if ([self.book.managedObjectContext save:&saveError] == NO) {
        NSLog(@"Error: %@", [saveError localizedDescription]);
    }

    [self.navigationController popViewControllerAnimated:YES];
}
```

Core Data Errors

```
- (IBAction)saveButtonTapped:(id)sender {

    self.book.title = self.titleTextField.text;
    self.book.author.authorName = self.authorTextField.text;

    NSError *saveError;
    if ([self.book.managedObjectContext save:&saveError] == NO) {
        NSAssert(NO, @"Error: %@", [saveError localizedDescription]);
        [self showAlert];
    }

    [self.navigationController popViewControllerAnimated:YES];
}

- (void)showAlert {
...
}

- (void)alertView:(UIAlertView *)alertView
        didDismissWithButtonIndex:(NSInteger)buttonIndex {
    abort();
}
```

Core Data Errors

```
- (IBAction)saveButtonTapped:(id)sender {

    self.book.title = self.titleTextField.text;
    self.book.author.authorName = self.authorTextField.text;

    NSError *saveError;
    if ([self.book.managedObjectContext save:&saveError] == NO) {
        NSAssert(NO, @"Error: %@", [saveError localizedDescription]);
        [self showAlert];
    }

    [self.navigationController popViewControllerAnimated:YES];
}

- (void)showAlert {
...
}

- (void)alertView:(UIAlertView *)alertView
        didDismissWithButtonIndex:(NSInteger)buttonIndex {
    abort();
}
```

Core Data Errors

```
- (IBAction)saveButtonTapped:(id)sender {

    self.book.title = self.titleTextField.text;
    self.book.author.authorName = self.authorTextField.text;

    NSError *saveError;
    if ([self.book.managedObjectContext save:&saveError] == NO) {
        NSAssert(NO, @"Error: %@", [saveError localizedDescription]);
        [self showAlert];
    }

    [self.navigationController popViewControllerAnimated:YES];
}

- (void)showAlert {
...
}

- (void)alertView:(UIAlertView *)alertView
        didDismissWithButtonIndex:(NSInteger)buttonIndex {
    abort();
}
```

Core Data Errors

```
...

    _fetchedResultsController.delegate = self;
    NSError *fetchError;
    if ([_fetchedResultsController performFetch:&fetchError] == NO) {

        NSLog(@"Error: %@", [fetchError localizedDescription]);
        abort();
    }
}

return _fetchedResultsController;
```

Core Data Errors

```
...

    _fetchedResultsController.delegate = self;
    NSError *fetchError;
    if ([_fetchedResultsController performFetch:&fetchError] == NO) {

        NSAssert(NO,@"Error: %@", [fetchError localizedDescription]);
        abort();
    }
}

return _fetchedResultsController;
```

Core DataErrors

Core Data Constants Reference

https://developer.apple.com/library/mac/documentation/Cocoa/Reference/CoreDataFramework/Miscellaneous/CoreData_Constants/Reference/reference.html#//apple_ref/c/data/NSPersistentStoreSaveConflictsErrorKey

MDMCoreData

https://github.com/mmorey/MDMCoreData

MDMPersistenceController (iOS, OS X)

MDMFetchedResultsTableDataSource (iOS)

NSManagedObject+MDMCoreDataAdditions (iOS, OS X)

MDMPersistenceController

```
NSURL *storeURL = [[self applicationDocumentsDirectory]
                   URLByAppendingPathComponent:@"MDMCoreData.sqlite"];

NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"MDMCoreData"
                                              withExtension:@"momd"];

_persistenceController = [[MDMPersistenceController alloc]
                          initWithStoreURL:storeURL modelURL:modelURL];
```

MDMFetchedResultsTableDataSource

```
self.tableDataSource = [[MDMFetchedResultsTableDataSource alloc]
             initWithTableView:self.tableView
      fetchedResultsController:[self fetchedResultsController]];

self.tableDataSource.delegate = self;

self.tableDataSource.reuseIdentifier = @"Cell";

self.tableView.dataSource = self.tableDataSource;
```

NSManagedObject+MDMCoreDataAdditions

```
+ (NSString *)MDMCoreDataAdditionsEntityName {

    return NSStringFromClass(self);
}



+ (instancetype)MDMCoreDataAdditionsInsertNewObjectIntoContext:
                                                 (NSManagedObjectContext *)context {

    return [NSEntityDescription insertNewObjectForEntityForName:[self
                    MDMCoreDataAdditionsEntityName] inManagedObjectContext:context];
}
```

Instruments

Demo

Questions?

Tomorrow: High Performance Core Data

MatthewMorey.com | @xzolian