iOS Application Development

Day 3

Chris Zelenak

10/06/2010

Outline

Outline

Core Data and SQLite

1. Core Data and SQLite

NSManagedObject and beyond

NSManagedObject is the primary element of working with your core data store. It represents a single entity's worth of information in the database that backs your app. The actual data for the entity is able to be queried via the valueForKey and setValue:forKey: methods of NSManagedObject.

NSManagedObject and beyond

NSManagedObject changes are queued to an NSManagedObjectContext which is roughly equivalent to a database transaction; it provides locking, commit/rollback and undo/redo functionality for object changes.

NSManagedObject and beyond

NSEntityDescription describes the schema associated with a specific type of NSManagedObject; a database table is similar in the way that it describes the layout of its child rows.

NSManagedObject and beyond

NSManagedObjectModels are created by the programmer, and are collections of NSEntityDescriptions.

NSManagedObject and beyond

NSPersistentStore represents the actual physical location of your Core Data objects, and can be a database, an XML file, or any other other physical manifestation of data for which an NSPersistentStore has been written.

NSManagedObject and beyond

You can create your own NSManagedObjectModel subclasses that provide convenient abstractions over NSManagedObjectModel.

NSManagedObject and beyond

To get an instance of an NSManagedObject that is able to be saved to the NSPersistentStore, use:

```
NSManagedObject * newObject = [NSEntityDescription
                                insertNewObjectForEntityForName:@"EntityName"
                                inManagedObjectContext:managedObjectContext];
```

Listing 1: Getting a new NSManagedObject that will eventually be saved to an NSPersistentStore

NSManagedObject and beyond

To save changes to all current NSManagedObjects currently managed by an NSManagedObjectContext:

```
NSError * error = nil;
[managedObjectContext save:&error];
if(error){
    NSLog(@"Couldn't save objects, %@", error);
}
```

Listing 2: Saving changed objects

NSPredicate and NSFetchRequest

Actually fetching objects from the Core Data store requires you to build queries using NSPredicate or NSFetchRequest.

NSPredicate and NSFetchRequest

The syntax for NSPredicate and NSFetchRequest requests is similar to SQL, but not identical.

NSPredicate and NSFetchRequest

```
NSPredicate * searchPredicate = [NSPredicate
                        predicateWithFormat:@"(firstName = 'Chris') AND "
                                            @"(lastName BEGINSWITH 'Zel') AND "
                                            @"(age BETWEEN {%i,%i})", 20, 30];
```

Listing 3: NSPredicate example

See more..

Read more about writing NSPredicates in the "Predicate Programming Guide" in the XCode documentation.

NSPredicate and NSFetchRequest

You can also build named NSFetchRequests in the .xcdmodel for your entities.

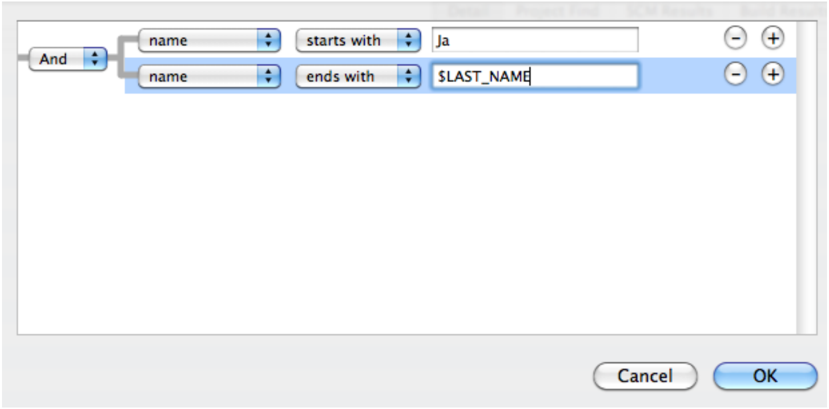

Figure: Predicate builder

NSPredicate and NSFetchRequest

```
NSFetchRequest * req = [managedObjectModel
                         fetchRequestFromTemplateWithName:@"requestByName"
                         substitutionVariables:[NSDictionary
                            dictionaryWithObjectsAndKeys:
                              @"Pwnsberry", @"LAST_NAME", nil]];
```

Listing 4: Stored NSPredicate example

Lab 6

Create a property list with the names of the students in the class

Lab 6

Create the xcdmodel that describes the entities

Lab 6

Initialize the Core Data persistence layer

Lab 6

Use NSUserDefaults to detect whether or not the app has been started before

Lab 6

Perform the import if the app has never been started before

Lab 6

Pass the user and image data on to the details controller

Outline

Networking (NSURLRequest)

2. Networking (NSURLRequest)

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURL is meant to only represent a single resource location

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURL can be allocated to represent either a filesystem location, or a web resource

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURLRequest and NSMutableURLRequest represent specific web resources that you'd like to initiate a connection to; NSURLRequest should be used for simple GET HTTP requests, while NSMutableURLRequests can be used for more complex HTTP requests.

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSMutableURLRequest allows you to set HTTP headers, the HTTP method used, and the request body.

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURLRequest and NSMutableURLRequest both by default only work with HTTP requests

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURLConnection initiates the download and returns the NSHTTPURLResponse which contains the body of the response, as well as http status code and response headers.

NSURL, NSURLRequest, NSMutableURLRequest AND NSURLConnection

NSURLConnection can send data both synchronously or asynchronously; the response information is passed back to the connection delegate via the informal NSURLConnection delegate.

Networking Lab

Create a new UIVIewController

Networking Lab

Add an NSURLConnection object

Networking Lab

Download the plist object and deserialize it

Networking Lab

Load the tableView with the new data

(ASIHttpRequest, EasyURLDownloader)

ASIHttpRequest gives you a full-featured HTTP library enhancement; cookie persistence support, enhanced HTTP auth support, S3 support and more. `http://github.com/pokeb/asi-http-request/`

(ASIHttpRequest, EasyURLDownloader)

EasyURLDownloader gives you a simple library to perform asynchronous GET downloads in the background
`http://github.com/netshade/EasyUrlDownloader`

Outline

Getting your application on the device

3. Getting your application on the device

Development certificates, Distribution certificates

Log in to the iPhone developer portal and request a developer certificate

Development certificates, Distribution certificates

Explain Key requests

Development certificates, Distribution certificates

Download and install certificate

Development certificates, Distribution certificates

Create development provisioning profile with devices

Development certificates, Distribution certificates

Assign development provisioning profile

Development certificates, Distribution certificates

AdHoc and Store based distribution reserved for Agents only

Outline

Using Instruments

4. Using Instruments

Always, always, always memory leak check

Opening up the Leaks tool and examining your application behavior

Always, always, always memory leak check

Examining specific leaks

Always, always, always memory leak check

Understanding the Leaks tool

Always, always, always activity monitor check

Using Activity Monitor to monitor your current system state

Outline

Distributing your app to others

5. Distributing your app to others

The process you need to know

Releasing your app to others in beta form

The process you need to know

What is an .ipa file

The process you need to know

How to send your .ipa file to others

The URLs you need to know

`http://developer.apple.com/iphone/` is the frontend to most Apple web services

The URLs you need to know

`http://itunesconnect.apple.com/` is the URL to access the app-release frontend and store management services provided by Apple

Outline

Questions