StackMob

Building an AirBnB Clone Workshop

360iDev 2013

Introduction

We are creating an Apartment Sharing app based on the popular service AirBnB. Of course, AirBnB is much more than a set of features. I find it’s more fun to explore technology using a real use case for developing an app.

Features

Our starting point is a fully baked UI for our app. This allows us to focus on the interesting part of adding user creation, login, logout, fetching our apartments, uploading new apartment data, triggering an email and adding offline caching.

What you’ll need for the workshop

A Mac computer running XCode version 4.6.3 or newer.
Your own StackMob Account (free to create)
Amazon Web Services account for S3

Agenda

StackMob Overview
Core Data Basics
StackMob SDK Overview
Hands on Lab

Advance Prep

You should have received an email and completed the advance preparation for this workshop, if you have not, please complete this section now. If you have any questions, please ask us.

Sign Up for StackMob

If you don't have a StackMob account, you can signup at http://bit.ly/bnbclone. During the signup process your first app will be created for you.

You'll be taken to the Getting Started page, go ahead and click link "skip and go directly to Dashboard".

Go to S3 Module on StackMob

Once you are in the StackMob Dashboard, press the "G" on your keyboard. A search window will open. Type in S3 and you'll see "Module : S3 File Storage", hit the enter key. Now let's get our Amazon Web Services credentials.

Get your AWS Credentials

Open a new browser window and go to http://aws.amazon.com and login. Select **My Account / Console** in the upper right and select **Security Credentials**. You'll be prompted to login with your Amazon username/password. You may get a popup, click **Continue to Security Credentials**.

On the left, select **Users**. Then click **Create New Users**. Enter a name for your user and click **Create**. You will have the option to **Download Credentials**. Please download them to your desktop for future reference.

Close the popup window and you should see your new user in a list. Select your new user. The window at the bottom will appear with tabs. Select the **Permissions** tab. Click **Attach user Policy**. Under Select Policy Template scroll down and select **Amazon S3 Full Access**. (we will discuss attaching policies for specific buckets during the workshop). Click **Apply Policy**.

Great! Now you've got your user on AWS setup. Let's create a bucket.

Create your S3 Bucket

In the upper left, select **Services**, and from the menu choose **S3**. Click **Create New Bucket** and give your bucket a name and select **US Standard for region.**

Add Credentials and Bucket Name to StackMob S3 Module.

Open the user credentials CSV file you downloaded. Copy and paste the **Access ID** and

Access Key into StackMob Dashboard. Enter the name of your **S3 bucket** to StackMob Dashboard.   Click the **Submit** button, you'll get a popup window select, **Yes, Set my bucket to public read**. You should see a blue success message.

Code Download for the Workshop

We have the code available on Github for download
https://github.com/stackmob/ApartmentShare/tree/stackmob-removed

Section 1

Schema Configuration - Hands on

We saw Matt demonstrate schema inference to dynamically create our schemas. Let's see how we create schemas manually.

From the dashboard select **Schema Configuration.** Click **Create New Schema**. Enter **apartment** as the name for your schema.

Click **Add Field** and create the following fields.

- apartment_type : string
- location : string
- room_count : int
- price : float
- photo : binary

Scroll down and set your Schema Permissions.

- create : allow to any logged in user
- read : open
- update : allow to sm_owner
- delete : allow to sm_owner

Click **Save Schema**.

Every app created on StackMob has a User schema automatically created. Let's modify our User schema to add an email field.

From the dashboard select **Schema Configuration.** Click **Edit** next to the User Schema.

Click **Add Field** and create the following field. We will use this email field later to contact the apartment owner.

- email : string

Click **Save Schema**.

Scroll down and let's change the field used for the forgot password email.
Click **Save Schema**.

Section 2

Add your API Key - Hands On

You'll find your API keys on the home screen of the StackMob Dashboard (http://dashboard.stackmob.com). Clicking the HOME button on the left menu will take you to the dashboard home.

Copy the Development Public API key.

With the API Key we can initialize the StackMob SDK in our app. Open the ApartmentShare project in XCode.

Open: Classes > AppDelegate >AppDelegate.m.

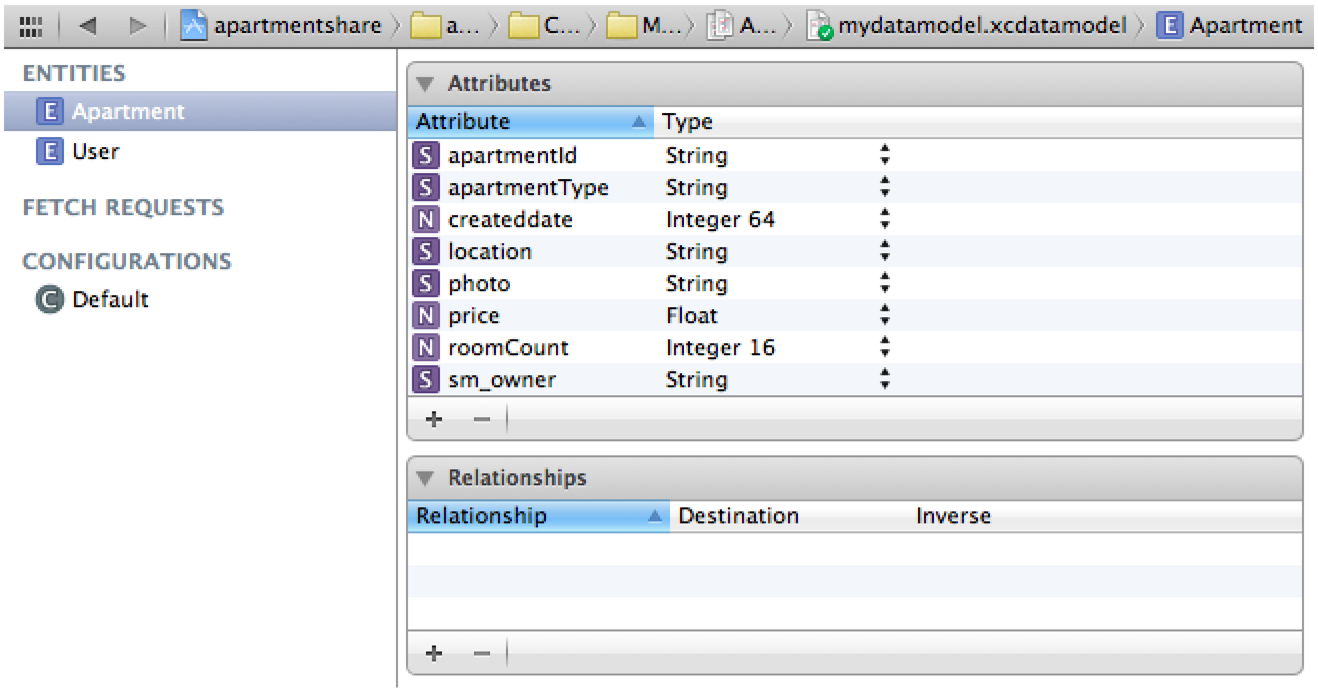
You'll see we've defined a variable **PUBLIC_KEY**. Paste your API key there.

Build and Run your Project

Section 3

Core Data Model - Walkthrough

We've already included your Core Data model, so let's take a look at it along with the NSManagedObject classes based on the entities.



Attribute	Type
apartmentId	String
apartmentType	String
createddate	Integer 64
location	String
photo	String
price	Float
roomCount	Integer 16
sm_owner	String

Relationship	Destination	Inverse

Section 4

User Creation - Hands on

We'll need to create a user account, so we can login to our app. We do this through Core Data using our User model.

OPEN: Classes > Views > Register > ADVRegisterViewController.h

Add the following property

```
@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
```

OPEN: Classes > Views > Register > ADVRegisterViewController.m

Add the following imports

```
#import "User.h"
#import "StackMob.h"
```

Init the managedObjectContext inside the **viewDidLoad** method

```
self.managedObjectContext = [[[SMClient defaultClient] coreDataStore] contextForCurrentThread];
```

Inside the **signUpPressed** method, create a new User managed object model in the current context and save the context.

```
User *newUser = [[User alloc] initIntoManagedObjectContext:self.managedObjectContext];

[newUser setUsername:self.userRegisterTextField.text];
[newUser setEmail:self.emailRegisterTextField.text ];
[newUser setPassword:self.passwordRegisterTextField.text];

[self.managedObjectContext saveOnSuccess:^{
   [self.navigationController popViewControllerAnimated:YES];
} onFailure:^(NSError *error) {
   NSLog(@"Error : %@", error);
}];
```

Build and Run your Project

Section 5

User Login - Hands on

Now that we have a user account, let's login to our app. We do this through the StackMob client.

OPEN: Classes > Views > Login > ADVLoginViewController.m

Add the following imports

```
#import "StackMob.h"
```

In the **loginPressed** method, use the StackMob Client to login our user, in the success block we'll confirm they are logged in and perform a segue to the Apartment List View.

```
[[SMClient defaultClient] loginWithUsername:self.userTextField.text password:self.passwordTextField.text
onSuccess:^(NSDictionary *results) {

        [self performSegueWithIdentifier:@"list" sender:self];

    } onFailure:^(NSError *error) {
        NSLog(@"Error : %@", error);
}];
```

Build and Run your Project

Section 6

User Logout - Hands on

Now that you are logged in, you want to logout of our app. We do this through the StackMob client.

OPEN: Classes > Views > ApartmentListVIew > ADVApartmentListViewController.m

Add the following imports

```
#import "StackMob.h"
```

In the **loginLogoutPressed** method, use the StackMob Client to logout our user, in the success block we'll confirm they are logged in and perform a segue to the Login View.

```
 [[SMClient defaultClient] logoutOnSuccess:^(NSDictionary *result) {
        NSLog(@"Success, you are logged out");

        [self.navigationController popViewControllerAnimated:YES];

    } onFailure:^(NSError *error) {
        NSLog(@"Logout Fail: %@",error);
}];
```

In the **getLogText** method, add the following line to display the appropriate text on the login/logout button:

```
NSString* logText = [[SMClient defaultClient] isLoggedIn] ? @"Log Out" : @"Log In";
```

Section 7

Add UserVoice - Hands on

UserVoice helps you manage customer support for you app with feedback forms and an embedded knowledge base.

UserVoice is a StackMob partner. We've added the UserVoice SDK to the project already. Let's add the module through StackMob Marketplace.

Go to the StackMob Marketplace at http://marketplace.stackmob.com. Select User Voice and click the **install** button. After you successful install, click **view it in your dashboard**.

To get our init code, click on the **Settings** button. This will sign you into the UserVoice dashboard. In the upper right, click on **Settings > Mobile**.

Copy the configuration code

```
UVConfig *config = [UVConfig configWithSite:@"YOUR_USERVOICE_URL"
                                     andKey:@"YOUR_KEY"
                                  andSecret:@"YOUR_SECRET"];

[UserVoice presentUserVoiceInterfaceForParentViewController:self andConfig:config];
```

OPEN: Classes > Views > Login > ADVLoginViewController.m

Paste the code in **showHelp** method.

Build and Run your Project

Section 8

Apartment List - Hands on

We need to fetch the apartments, in order to display hem. We do this through Core Data using a NSFetchRequest.

OPEN: Classes > Views > ApartmentList > ADVApartmentListViewController.h

Add the following property

```
@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
```

OPEN: Classes > Views > ApartmentList > ADVApartmentListViewController.m

Init the managedObjectContext inside the **viewDidLoad** method

```
self.managedObjectContext = [[[SMClient defaultClient] coreDataStore] contextForCurrentThread];
```

Inside the **getAllApartments** method, init a NSFetchRequest based on the Apartment managed object model, then execute the fetch request.

```
NSFetchRequest *fetch = [[NSFetchRequest alloc] initWithEntityName:@"Apartment"];
[self.managedObjectContext executeFetchRequest:fetch onSuccess:^(NSArray *results) {
    self.apartments = results;
    [self.apartmentTableView reloadData];
    [MBProgressHUD hideHUDForView:self.view animated:YES];
} onFailure:^(NSError *error) {
    [MBProgressHUD hideHUDForView:self.view animated:YES];
    NSLog(@"Error : %@", [error localizedDescription]);
}];
```

Section 9

Apartment Upload - Hands on

We need to upload an apartment, in order to view a list of them. We do this through Core Data using our Apartment model.

OPEN: Classes > Views > UploadView > ADVUploadImageViewController.h

Add the following property

```
@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
```

OPEN: Classes > Views > UploadView > ADVUploadImageViewController.m

Add the following imports

```
#import "StackMob.h"
#import "Apartment.h"
```

Init the managedObjectContext inside the **viewDidLoad** method

```
self.managedObjectContext = [[[SMClient defaultClient] coreDataStore] contextForCurrentThread];
```

Inside the **uploadDataToServer** method, create a new Apartment managed object model in the current context and save the context.

```
Apartment *newApartment = (Apartment *)[NSEntityDescription
insertNewObjectForEntityForName:@"Apartment" inManagedObjectContext:self.managedObjectContext];

[newApartment assignObjectId];

NSData *imageData = UIImageJPEGRepresentation(self.uploadImageView.image, 0.4);

NSString *picData = [SMBinaryDataConversion stringForBinaryData:imageData name:@"apartment.jpg"
contentType:@"image/jpg"];

NSString* apartmentType = self.apartmentTypeControl.selectedSegmentIndex == 0 ? @"House" : @"Flat";

NSNumber* price = [NSNumber numberWithFloat:[self.priceTextField.text floatValue]];
NSNumber* roomCount = [NSNumber numberWithFloat:self.roomsSlider.value];

[newApartment setPhoto:picData];
[newApartment setLocation:self.locationTextField.text];
[newApartment setRoomCount:roomCount];
[newApartment setPrice:price];
[newApartment setApartmentType:apartmentType];
```

```
[self.managedObjectContext saveOnSuccess:^{
    [MBProgressHUD hideHUDForView:self.view animated:YES];
     NSLog(@"Successful upload");
    [self.navigationController popViewControllerAnimated:YES];

} onFailure:^(NSError *error) {
    [MBProgressHUD hideHUDForView:self.view animated:YES];
    NSLog(@"Error: %@", [error localizedDescription]);
}];
```

Section 10

SendGrid Email - Hands on

SendGrid offers a email services via their API. As a StackMob partner, we've found SendGrid very popular with app developers. Installing the SendGrid Module on StackMob allows you to send email through StackMob's SDK. Let's add the module through the StackMob Marketplace.

Go to the StackMob Marketplace at http://marketplace.stackmob.com. Select SendGrid and click the **install** button. After you successful install, click **view it in your dashboard**.

We already have a contact form. All we need to do is add the code to send an email to the owner of the apartment from the currently logged in user.

We need to save the email of the currently logged in user:

OPEN: Classes > Views > LoginView > ADVLoginViewController.m

Go to the **logInPressed** method and in the success block of the login call add:

```
// Save email for contacting owner
[[NSUserDefaults standardUserDefaults] setObject:[results objectForKey:@"email"]
forKey:@"ContactOwnerEmailKey"];
```

OPEN: Classes > Views > ContactOwnerView > ADVContactOwnerViewController.m

Inside the **sendButton** method, get the senderEmail from NSUserDefaults, create a Custom Code request for sendgrid. Next, create an NSDictionary with your email parameters, serialize to JSON and set as the request body. Perform the custom code request and handle the success or error blocks.

```
NSString *senderEmail = [[NSUserDefaults standardUserDefaults]
objectForKey:@"ContactOwnerEmailKey"];

SMCustomCodeRequest *request = [[SMCustomCodeRequest alloc]
                               initPostRequestWithMethod:@"sendgrid/email"
                               body:nil];

NSArray *usernames = [[NSArray alloc]
                     initWithObjects:self.apartmentOwner, nil];

//convert object to data
NSDictionary *dic = [[NSDictionary alloc]
                    initWithObjectsAndKeys:
                         usernames, @"usernames",
                         @"Inquiry On Apartment", @"subject",
                         self.message.text, @"html",
                         senderEmail, @"from",
                         nil];

NSError* error = nil;
NSData* jsonData = [NSJSONSerialization
                        dataWithJSONObject:dic
                        options:0 error:&error];

[request setRequestBody:[[NSString alloc]
                             initWithData:jsonData
                             encoding:NSUTF8StringEncoding]];

[[[SMClient defaultClient] dataStore] performCustomCodeRequest:request onSuccess:^(NSURLRequest
*request, NSHTTPURLResponse *response, id JSON) {

    [MBProgressHUD hideHUDForView:self.view animated:YES];

    UIAlertView *successAlertView = [[UIAlertView alloc] initWithTitle:@"Success" message:@"Email sent!"
        delegate:self cancelButtonTitle:@"Ok" otherButtonTitles:nil, nil];

    [successAlertView show];
 } onFailure:^(NSURLRequest *request, NSHTTPURLResponse *response, NSError *error, id JSON){
         [MBProgressHUD hideHUDForView:self.view animated:YES];
         NSLog(@"Error: %@", [error localizedDescription]);
 }];
```

Build and Run your Project

Section 11

Forgot Password Template - Walk through

Now that we have the SendGrid Module, we can customize our forgot email template.

https://dashboard.stackmob.com/module/forgotpassword

For those who want to embed a link in the email and point to a hosted web page where the user can reset their password, we've written up a blog post on the topic.

https://blog.stackmob.com/2013/06/build-a-better-forgot-password-email/

Section 12

Offline Sync

Using Core Data with StackMob is not a requirement. You can use the datastore API for all the work you've done today. But, using Core Data with StackMob offers the added benefit of online/offline caching and data synchronization.

OPEN: Classes > AppDelegate > AppDelegate.m

Change the following line of code

```
SM_CACHE_ENABLED = YES;
```

and **uncomment** the block of code inside the **didFinishLaunchingWithOptions** method.

OPEN: Classes > Views > ApartmentList > ApartmentListViewController.m

Inside the **cellForRowAtIndexPath** method and after this line NSString *picString = [apartment valueForKey:@"photo"]; Add the IF statement

```
if ([SMBinaryDataConversion stringContainsURL:picString]) {
```

Before the return cell add the ELSE statement.

```
} else {
        UIImage *image = [UIImage imageWithData:[SMBinaryDataConversion dataForString:picString]];
        cell.apartmentImageView.image = image;
        [self.apartmentImages setObject:image forKey:indexPath];
 }
```

Build and Run your Project