MODULE: Deep Dive into iOS App Development
SESSION [4]
SESSION TITLE: [Sending Emails and SMS, and Dialing a Phone]

Sources:

1. [Professional iOS Programming] [Chapter 11][9781118661130]
2. [Wrox Beginning iPhone SDK Programming with objective C] [Chapter 16][ISBN No]

MODULE Objectives

At the end of this module, you will be able to:

- Send email and text messages from your applications
- Dial phone numbers from your applications

Session Objectives

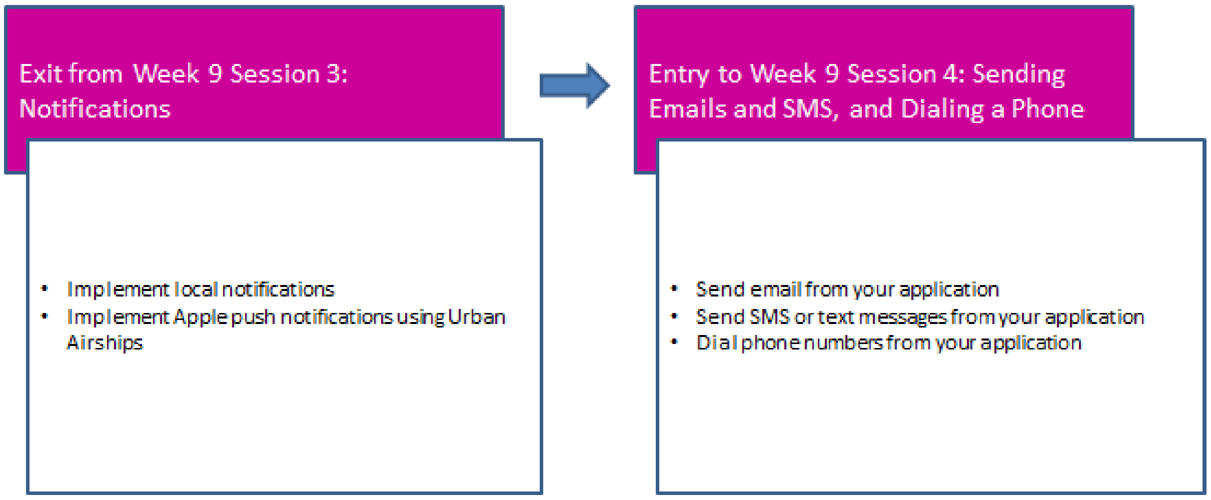
At the end of this session, you will be able to:

- Send email from your application
- Send SMS or text messages from your application
- Dial phone numbers from your application

<H1> Introduction

The iPhone comes with a number of built-in applications that make it one of the most popular mobile devices of all time. Some of these applications are **Contacts**, **Mail**, **Phone**, **Safari**, **Messages**, and **Calendar**. These applications perform most of the tasks you would expect of a mobile phone. As an iPhone developer, you can also programmatically invoke these applications from within your application using the various application programming interfaces (APIs) provided by the iPhone software development kit (SDK).

Suppose you get a notification through an email or a message and you want to respond back through a text message or by making a call. This session describes how to send an email and a text message and dial a phone number through your application.



Exit from Week 9 Session 3: Notifications

- Implement local notifications
- Implement Apple push notifications using Urban Airships

Entry to Week 9 Session 4: Sending Emails and SMS, and Dialing a Phone

- Send email from your application
- Send SMS or text messages from your application
- Dial phone numbers from your application

<H1> [Sending Email]

Source: [Professional iOS Programming] [11] [355]

[Wrox Beginning iPhone SDK Programming with objective C][Chapter 16][Page 355]

Sending emails is one of the many tasks performed by the iPhone users. This task is accomplished using the built-in **Mail** application in the iPhone. The Mail application is a rich HTML mail client that supports POP3, IMAP, Exchange email systems, and most web-based emails such as Yahoo! and Gmail.

There are times when you need to send an email in your iPhone application. A good example is embedding a feedback button in your application that users can click to email and send feedback to you directly.

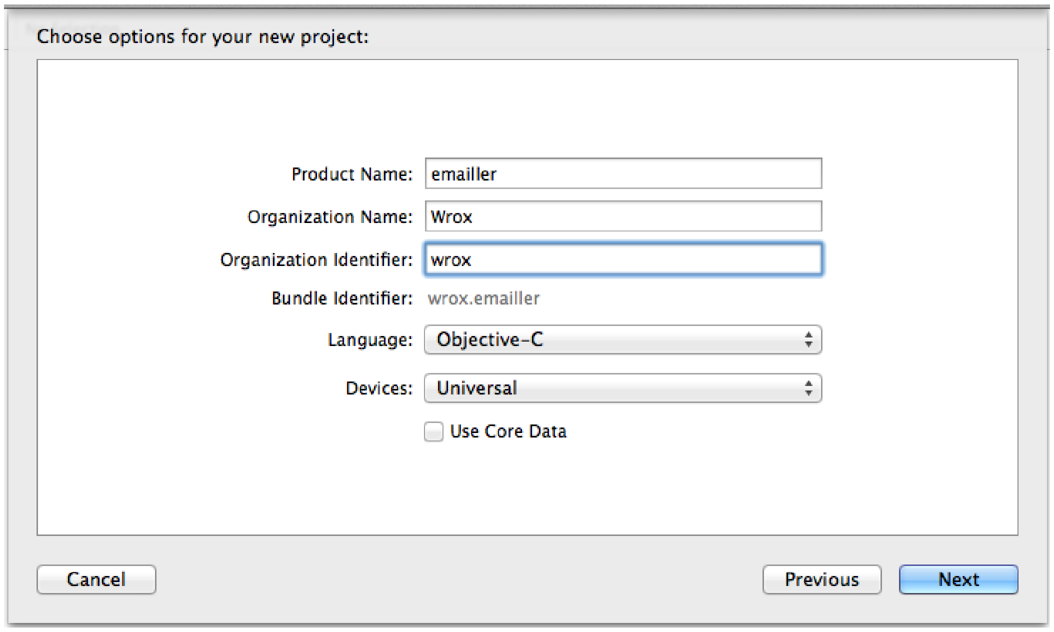
You have two ways to send emails programmatically:

- Build your own email client and implement all the necessary protocols necessary to communicate with an email server
- Invoke the built-in Mail application and ask it to send the email for you

Unless you are well versed in network communications and familiar with all the email protocols, your most logical choice is to go for option two—invoke the Mail application to do the job.

In this session, you will create a simple application that enables you to present the user interface element for writing an email message and that contains the logic for sending the message to a recipient.

Start **Xcode** and create a new project using the **Single View Application** template. Name your project as **emailler** using the options shown in **Figure 1**.



Product Name: emailler
Organization Name: Wrox
Organization Identifier: wrox
Bundle Identifier: wrox.emailler
Language: Objective-C
Devices: Universal
Use Core Data

Figure 1: Create a New Project, emailler, in Xcode

To send email messages, you need to add the `MessageUI` framework. This framework provides the View Controllers for interfacing emails and text messages in your application without actually leaving the application.

<H2> [Composing an Email]

Professional iOS Programming] [Chapter 11][Page 356]

You can compose an email using the `MFMailComposeViewController` method that you create, initialize, configure, and present as follows:

```
- (IBAction)sendEmail:(UIButton *)sender {
      [self.subjectField resignFirstResponder];
      if ([MFMailComposeViewController canSendMail])
         {
            MFMailComposeViewController *mailer =
            [[MFMailComposeViewController alloc] init];
            mailer.mailComposeDelegate = self;
            [mailer setSubject:self.subjectField.text];
            NSArray *toRecipients = [NSArray arrayWithObjects:
                                @"email@domain.com", nil];
            [mailer setToRecipients:toRecipients];
            [mailer setMessageBody:@"" isHTML:YES];
            [self presentViewController:mailer animated:YES completion:nil];
            }
      else
            {
            UIAlertView *alert = [[UIAlertView alloc]
            initWithTitle:@"Sorry"
            message:@"Your device doesn't support sending email."
                        delegate:nil

                        cancelButtonTitle:@"OK"
                        otherButtonTitles:nil];
            [alert show];
            }
}
```

In the above code, the `setToRecipients:`, `setCcRecipients:`, and `setBccRecipients:` methods accept an `NSArray` of email addresses.

You can use the `setMessageBody:isHTML:` method to set predefined text in the message body before the `ViewController` is presented. The `isHTML` boolean defines whether the message body itself is HTML or plain text.

You initiate the actual sending of the email by tapping the Send button of the `MFMailComposeViewController`. There is also a Cancel button that can be tapped by the user. In any case, it is important for you to implement the `mailComposeController:didFinishWithResult:error:` delegate method so you can take the appropriate action depending on the result of the send or cancel action in the `MFMailComposeViewcontroller`, as shown in the following code:

```
- (void)mailComposeController:(MFMailComposeViewController*)controller

didFinishWithResult:(MFMailComposeResult)result
error:(NSError*)error
{
      switch (result)
      {
      case MFMailComposeResultCancelled:
      // you cancelled the operation and no email message was queued.
      break;
      case MFMailComposeResultSaved:
      //Mail saved: you saved the email message in the drafts folder.
      break;
      case MFMailComposeResultSent:
      {
      // the email message is queued in the outbox. It is ready to send. ;

      }
      break;
      case MFMailComposeResultFailed:
      // the email message was not saved or queued, possibly due to an error.
      break;
      default:

      break;
      }
      // Remove the mail view
      [self dismissViewControllerAnimated:YES completion:nil];
}
```

In **Swift** the `MessageUI.framework`, which comes from Objective-C, is used to add and send email and SMS. Select the **Project -> Build Phase -> Link Binary With Libraries** option. Then, click the **+** button and add the `MessageUI.framework`.

After you have added the `MessageUI.framework` to your project, you have to import it in the `ViewController.`**`Swift`** file as follows:

```
import MessageUI
```

Next, you need to add an appropriate delegate to your project to handle user requests. You can add the following delegates into the `ViewController.`**`Swift`** file:

- `UINavigationControllerDelegate`
- `MFMessageComposeViewControllerDelegate`
- `MFMailComposerViewControllerDelegate`

These delegates will be added as follows:

```
class   ViewController:   UIViewController,   UINavigationControllerDelegate,
MFMessageComposeViewControllerDelegate, MFMailComposeViewControllerDelegate
```

The syntax to send email in **Swift** language will be as follows:

```
@IBAction func sendEmail(sender: UIButton) {
      var mailer: MFMailComposeViewController= MFMailComposeViewController()
      if (MFMailComposeViewController.canSendText())
      {
      mailer.mailComposeDelegate = self
            mailer.setSubject: selfsubjectField.text
            mailer.setToRecipients(["receipient@domain.com"])
            mailer.setMessageBody(" ", isHTML: true)
      }
```

<H3> [Working with Attachments]

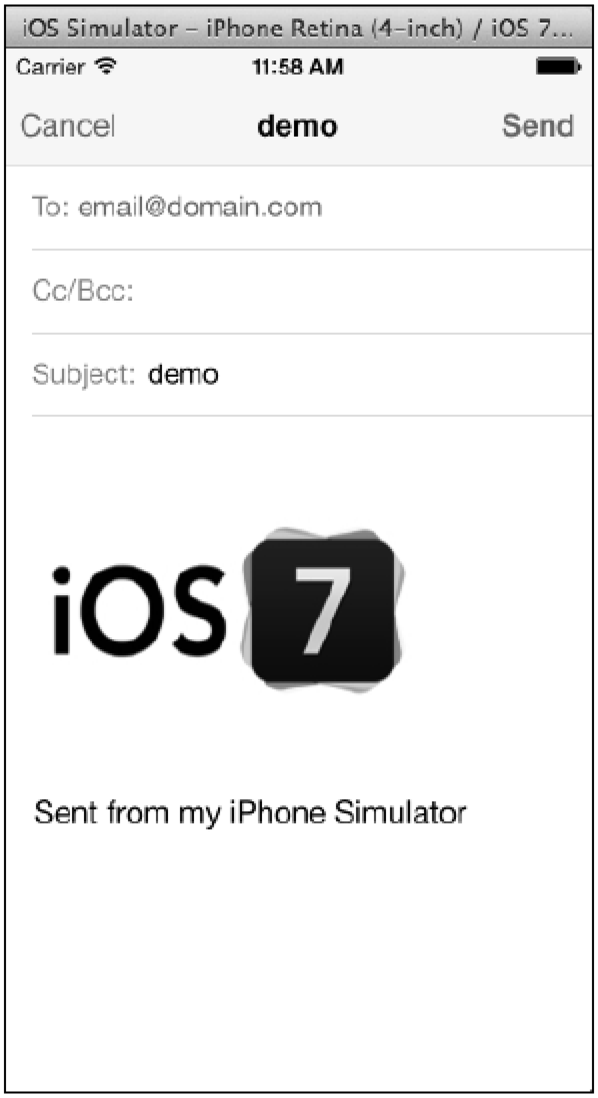
Professional iOS Programming] [Chapter 11][Page 358]

The `MFMailComposeViewController` has the delegate method `addAttachmentData:mimeType:fileName:` that you can use to add attachment data to the message you are composing.

To attach an image that is located in the bundle of your application, you can add the following code in the `sendEmail:` method:

```
//sample add some image
UIImage* imgToAttach = [UIImageimageNamed:@"banner-ios.png"];
NSData *data = UIImageJPEGRepresentation(imgToAttach, 1.0);
[maileraddAttachmentData:datamimeType:@"image/jpg"
fileName:@"banner-ios.png"];
```

If you launch the email application, you can enter a recipient(s) and a subject and then tap the **Send** button. This will present the `ViewController` of the email application. This `ViewController` is actually the configured `MFMailComposeViewController`, and it will display like the one shown in **Figure 2**.



To: email@domain.com

Cc/Bcc:

Subject: demo

Sent from my iPhone Simulator

Figure 2: Invoke Email within Application

Technical Stuff

The accelerometer used on the iPhone and iPod Touch gives a maximum reading of about +/- 2.3G with a resolution of about 0.018 g.

Quick Tip

The `MFMailComposerViewController` requires an SMTP account to have been set up on the phone. If no SMTP account has been set up, the email cannot be sent.

Big Picture

When you have enabled `MFMailComposerViewController` in your application, you are able to send an email from your app and prepopulate the fields like the recipient, subject, or body text. This can be enabled to integrate various activities in an application, such as ticket reservation app, game app, and online shopping app.

Additional Knowhow

If you want to invoke the Safari web browser on your iPhone, you can use a URL string and then use the `openURL:` method of the application instance, as follows:

```
//Objective-C syntax to invoke application
      [[UIApplicationsharedApplication]
      openURL:[NSURL URLWithString: @"http://www.apple.com"]];
//Swift Syntax to invoke application
      optional  func  application(_application:  UIApplication,  openURL  url:
NSURL,  sourceApplication  sourceApplication:String,  annotation  annotation:
AnyObject?)->Bool
```

The preceding code snippet invokes Safari to open the web site *www.apple.com*.

<H1> [Sending SMS (Text messages)]

--

Source: [Professional iOS Programming] [11] [359]

Sending SMS or text messages on an iOS device is an action that requires user interaction and user authorization. Unlike an Android device, where you can send a text message completely from a code without the user of the device even knowing it has been sent, an iOS device has a more robust security layer in place.

Just as sending email, SMS also needs the `MessageUI.framework` in both Objective-C and **Swift**. You also need to add the `MFMessageComposeViewController` delegate method for sending an SMS.

<H2> [Verifying if SMS Is Available]

[Professional iOS Programming] [Chapter 11][Page 359]

The `MFMessageComposeViewController` delegate is used to send an SMS. Therefore, you need to import the `<MessageUI/MFMessageComposeViewController.h>` header definition after you have added the `MessageUI.framework` to your project.

The `MFMessageComposeViewController` has a static method named `canSendText`.

- If the `canSendText` method returns a BOOL value, then the device is capable of sending a text message.
- If the return value is no, then the device is not capable of sending a text message. So you can inform the user that this function is not available.

<H3> [Composing a Text Message]

Professional iOS Programming] [Chapter 11][Page 359]

Suppose you have created a user interface with a `UITextField` or a `UITextView` in which you capture the user's message that needs to be sent. Now, you can perform the following operations:

1. Create and initialize an instance of the `MFMessageComposeViewController` method.
2. Set the body property of your controller's instance.
3. Set an `NSArray` of recipients, where a recipient is a mobile number.
4. Set the `messageComposeDelegate` property.
5. Present the created `ViewController` modally.

You can implement the `sendSMSToNumber:` method as shown below:

```
- (void)sendSMSToNumber:(NSString *)mobileNumber
{
      MFMessageComposeViewController *controller =
      [[MFMessageComposeViewController alloc] init];
      if([MFMessageComposeViewController canSendText])
      {
      controller.body = NSLocalizedString(@"SMSMESSAGE", nil);
      controller.recipients = [NSArray arrayWithObjects:mobileNumber, nil];
      controller.messageComposeDelegate = self;
      [self presentModalViewController:controller animated:YES];
      }
}
```

Now, the user has two options in the `ViewController` dialog box, cancel or send the message. Therefore, you need to implement the delegate method `messageComposeViewController:didFinishWithResult:`. After dismissing the modal `ViewController`, you perform the code required for canceling or sending the message, depending on your application's requirement.

You can implement the `messageComposeViewController:didFinishWithResult:` method as shown below:

```
-(void)messageComposeViewController:(MFMessageComposeViewController
*)controller
didFinishWithResult:(MessageComposeResult)result
{
      [self dismissModalViewControllerAnimated:YES];

      if (result == MessageComposeResultCancelled)
      {
            //cancelled
      }
      else if (result == MessageComposeResultSent)
      {
            //sent
      }
      else
      {
            //failed
      }
}
```

The syntax to implement the `MFMessagingComposeViewController` method in **Swift** is as follows:

```
class                    ViewController:                    UIViewController,
MFMessageComposeViewControllerDelegate {
// Define function for Message compose
      func ComposeMessage() {
      if MFMessageComposeViewController.canSendText(){
            let sms = MFMessageComposeViewController()
            sms.messageComposeDelegate = self
            sms.recipients = [“123456” ,” 1209876”]
            sms.body = “Hello This Is Swift”
            self.presentViewController(sms, animated: true, completion: nil)
            }
      else{
            println(“Message App not available”)
             }
      }
      func                            messageComposeViewController(controller:
MFMessageComposeViewController!,          didFinishWithResult          result:
MessageComposeResult) {
      self.dismissViewControllerAnimated(true, completion: nil)
      }
}
```

Quick Tip

If you want to send a message from an application, then you need to enable notifications in your device settings.

Real Life Connect

In some situations you may want to programmatically send SMS or text messages in your iOS applications. For example, your application is trying to monitor the current location of a user and automatically sending a SMS message to a predefined phone number. This will be very helpful if you were trying to build an application just to ensure someone's safety, like children or someone elder at home.

<H2> [Dialing a Phone Number]

Professional iOS Programming] [Chapter 11][Page 360]

Your application might have an About or Contact view that displays contact information about you or your customer, including a phone number. When an iPhone user views a phone number in an application, he/she expects that he/she can just click on it and the number will be dialed. That's the whole concept of a smartphone.

<H3> [Verifying Dialing Capability]

Professional iOS Programming] [Chapter 11][Page 360]

Not every iOS device, however, is capable of dialing a phone number. Therefore, before implementing the dial code, it is necessary to verify whether the device is capable of dialing a number to avoid an application crash.

Dialing a phone number consists of simply opening a URL on the device, which starts with the URL identifier `tel:// or telprompt://`. These are built-in URL schemes that work exactly like the custom URL schemes you used in the previous session.

To verify whether a device has the capability of dialing a phone number, you can use the following code snippet:

```
If([[UIApplicationsharedApplication]          canOpenURL:          [NSURL
URLWithString:@"tel://"]]){
// device can dial
}
else{
//device can't dial
}
```

<H2> [Invoking the Phone]

To make a phone call using the iPhone's dialer, you use the `openURL:` method of the `UIApplication` class and pass in the URL identifier with the phone number as follows:

```
NSString* phoneNumber = @"+15558888";
[[UIApplicationsharedApplication] openURL:
[NSURL URLWithString:[NSStringstringWithFormat:@"tel://%@",phoneNumber]]]];
```

The preceding statement invokes the dialer of the iPhone using the phone number specified.

If your application predefines the phone number to be dialed after performing an action like clicking a button, make sure you use the international notation to guarantee that the number can be dialed from any user's device.

A special URL scheme is available that will dial the phone number but will return to your application once the call is terminated. For this purpose, use the following code to start dialing. It will also present a `UIAlertView` before the dialing starts:

```
NSString* phoneNumber = @"+15558888";
[[UIApplicationsharedApplication] openURL:
[NSURL
URLWithString:[NSStringstringWithFormat:@"telprompt://%@",phoneNumber]]]];
```

Real Life Connect

Dialing a phone number programmatically is one of the most advantageous features in an application. Dialing a phone number is just like calling a URL in your application. When a user touches the number, your mobile phone will ask whether you want to dial this number, save this number, or send a text message to this number. This feature works in your iOS phone using the **tel scheme native app**. This feature will also help you to protect your app from crashing.

Additional Knowhow

Emailing and messaging is an old-fashioned and clunky concept. Now users can easily exchange files with other iOS devices with the help of AirDrop. By using AirDrop, users can quickly share photos, contacts, videos, and other application files using any app that supports the Share button integration.

----
examination, you will be required to dial phone numbers from your application.

Lab Connect
During the lab hours of this session, you will be able to create an application for sending email and SMS, and dialing a phone.

Cheat Sheet

- You can use the following code to send email from within your application:

```
NSString*emailString=
@”mailto:?to=USER@EMAIL.COM & subject=SUBJECT & body=BODY
OF EMAIL”;
[[UIApplication sharedApplication]
openURL:[NSURL
URLWithString:emailString]];
```

- You can use the following code to invoke SMS from within your application:

```
[[UIApplication sharedApplication]
openURL:[NSURL
URLWithString:@”sms:1234567890”]];
```

- You can use the following code to invoke Phone from your application:

```
[[UIApplication sharedApplication]
openURL:[NSURL
URLWithString:@”tel:1234567890”]];
```