13 내장된 애플리케이션 접근하기

이번 장에서 다루는 내용

- ☑ 애플리케이션 안에서 이메일 보내는 방법
- ☑ 애플리케이션 안에 사파리 포함하기
- ☑ 애플리케이션 안에서 전화 거는 방법
- ☑ 애플리케이션 안에서 SMS 메시지 보내는 방법
- ☑ 카메라와 사진 라이브러리에 접근하기

아이폰 안에는 아이폰을 가장 인기있는 모바일 장치들 중에 하나로 만들어 주는 여러 개의 애플리케이션들이 내장되어 있다. 그 중에 몇 가지들이 바로 메일(Mail), 전화(Phone), 사파리(Safari), SMS, 그리고 달력(Calendar)이다. 이들 애플리케이션은 모바일 폰에 기대하는 대부분의 작업들을 수행한다. 아이폰 개발자인 여러분은 아이폰 SDK가 제공하는 다양한 API들을 이용하여 여러분의 애플리케이션에서 이들 애플리케이션을 프로그램적으로 실행할 수도 있다.

이번 장에서는 아이폰에 포함된 여러 애플리케이션들을 실행하는 방법과 여러분의 아이폰 애플리케이션과 그 애플리케이션들이 상호작용하는 방법을 배우게 될 것이다.

이메일 보내기

이메일 보내기는 아이폰 사용자들이 가장 많이 이용하는 것 중 하나이다. 아이폰에서 이메일을 보내는 것은 POP3와 IMAP, 그리고 Exchange 이메일 시스템과 Yahoo! 그리고 Gmail과 같은 대부분의 웹기반의 이메일을 지원하는 리치(rich) HTML 메일 클라이언트인 메일(Mail) 애플리케이션을 이용하여 할 수가 있다.

애플리케이션을 개발하다 보면 여러분의 아이폰 애플리케이션에서 사용자가 이메일을 보낼 수 있도록 해야 할 때가 있다. 좋은 예로는 애플리케이션에 피드백 버튼을 포함시켜서 사용자가 그 버튼을 클릭하면 바로 피드백을 보낼 수 있게 하는 것이다. 프로그램적으로 이메일을 보내기 위해서는 두 가지 방법이 있다.

- 이메일 클라이언트를 만들고 이메일 서버와 통신하는 데 필요한 모든 프로토콜을 구현한다.
- 내장된 메일(Mail) 애플리케이션을 실행하여 이메일을 전송할 수 있도록 한다.

만약에 여러분이 네트워크 통신을 매우 잘 알고 있으며 모든 이메일 프로토콜에 익숙한 게 아니라면, 가장 합리적인 선택은 두 번째 옵션인 메일(Mail) 애플리케이션으로 이메일을 전송하는 것이 될 것이다. 다음의 도전 과제는 그 방법을 보여준다[1].

도전 과제 메일 애플리케이션을 이용하여 이메일 보내기

소스 코드: 13/Emails

1. Xcode에서 View-based Application (iPhone) 프로젝트를 만들고 이름을 `Emails`라고 한다.
2. `EmailsViewController.xib` 파일을 인터페이스 빌더에서 편집할 수 있도록 더블클릭한다.
3. 뷰 윈도우에 아래의 뷰들을 생성한다(그림 13-1 참조).
   - `Label`
   - `TextField`
   - `TextView`(뷰에 포함된 샘플 텍스트는 지우도록 한다)
   - `Button`

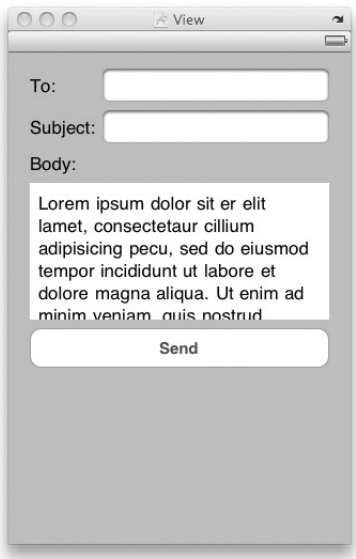

그림 13-1

[1] **옮긴이** 이 예제를 따라하기 위해서는 소스 코드를 다운받아야 한다. 이 책의 모든 소스 코드는 원출판사인 WROX에도 있지만, iOS 4로 역자가 직접 따라하고 주석을 붙인 소스 코드는 제이펍 블로그(http://jpub.tistory.com/)와 역자의 블로그(http://peterslab.tistory.com/)에서 받을 수 있다.

4. EmailsViewController.h 파일에 다음의 굵게 표시한 코드를 삽입한다.

```
#import <UIKit/UIKit.h>

@interface EmailsViewController : UIViewController {
    IBOutlet UITextField *to;
    IBOutlet UITextField *subject;
    IBOutlet UITextView  *body;
}

@property (nonatomic, retain) UITextField *to;
@property (nonatomic, retain) UITextField *subject;
@property (nonatomic, retain) UITextView  *body;

-(IBAction) btnSend: (id) sender;

@end
```

5. 인터페이스 빌더로 돌아가서, File's Owner 항목에서 텍스트 필드들과 텍스트 뷰로 Control-클릭하고 드래그 앤 드롭하여 to와 subject, 그리고 body를 각각 선택한다.

6. 버튼에서부터 Control-클릭한 상태로 드래그하여 File's Owner 항목에 드롭한 다음, btnSend:를 선택한다.

7. EmailsViewController.m 파일에 아래의 굵게 표시된 코드를 추가한다.

```
#import "EmailsViewController.h"

@implementation EmailsViewController

@synthesize to, subject, body;

- (void) sendEmailTo:(NSString *) toStr
         withSubject:(NSString *) subjectStr
            withBody:(NSString *) bodyStr {

    NSString *emailString =
         [[NSString alloc]
         initWithFormat:@"mailto:?to=%@&subject=%@&body=%@",
         [toStr
              stringByAddingPercentEscapesUsingEncoding:
              NSASCIIStringEncoding],
         [subjectStr
              stringByAddingPercentEscapesUsingEncoding:
```

```
                NSASCIIStringEncoding],
            [bodyStr
                stringByAddingPercentEscapesUsingEncoding:
                NSASCIIStringEncoding]];
    [[UIApplication sharedApplication] openURL:[NSURL URLWithString:emailString]];
    [emailString release];
}

-(IBAction) btnSend: (id) sender {
    [self sendEmailTo:to.text withSubject:subject.text withBody:body.text];
}

- (void)dealloc {
    [to release];
    [subject release];
    [body release];
    [super dealloc];
}
```

8. Command-R을 눌러 실제 아이폰에서 애플리케이션을 테스트해보자. 그림 13-2는 실제로 동작하는 애플리케이션을 보여준다. 텍스트 필드와 텍스트 뷰에 적절한 내용을 입력을 한 후, Send 버튼을 클릭하면 메일(Mail) 애플리케이션이 실행되며 애플리케이션에 입력했던 내용들이 모두 채워져 있을 것이다. 메일(Mail) 애플리케이션에서 Send 버튼을 클릭하면 이메일이 발송된다.

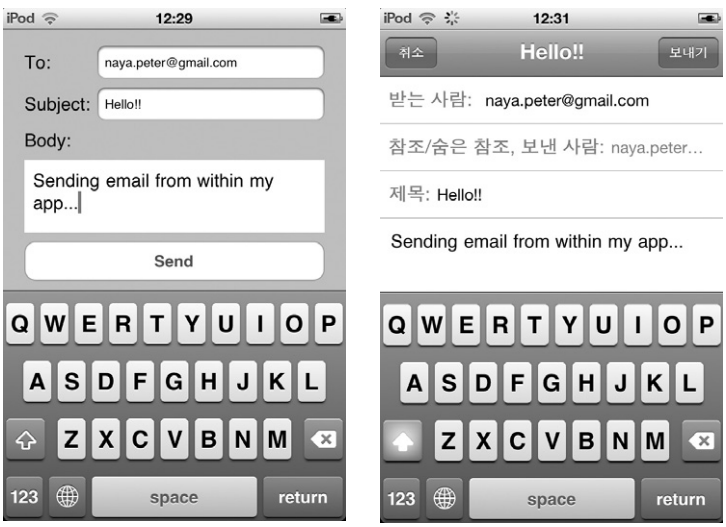

그림 13-2

동작 방법

메일(Mail) 애플리케이션을 실행하는 마법은 정의한 sendEmailTo:withSubject:withBody: 메서드에서 생성한 문자열에 있다.

```
NSString *emailString =
    [[NSString alloc] initWithFormat:@"mailto:?to=%@&subject=%@&body=%@",
        [toStr
            stringByAddingPercentEscapesUsingEncoding:
            NSASCIIStringEncoding],
        [subjectStr
            stringByAddingPercentEscapesUsingEncoding:
            NSASCIIStringEncoding],
        [bodyStr
            stringByAddingPercentEscapesUsingEncoding:
            NSASCIIStringEncoding]];
```

기본적으로 이것은 mailto: 프로토콜을 가리키는 URL 문자열이다. to와 subject, 그리고 body 같은 여러 개의 매개변수들에 문자열이 삽입되었다. 여러 가지 매개변수들을 인코딩하여 유효한 URL 문자열을 만들기 위하여 NSString 클래스의 stringByAddingPercentEscapes UsingEncoding: 메서드를 사용한다는 것에 주목하자.

메일(Mail) 애플리케이션을 실행하기 위하여 싱글턴(singleton) 애플리케이션 인스턴스를 반환하는 sharedApplication 메서드를 호출한 다음, openURL: 메서드를 사용하여 메일(Mail) 애플리케이션을 실행한다.

```
[[UIApplication sharedApplication] openURL:[NSURL URLWithString:emailString]];
```

이번 예제는 실제 기기에서만 동작한다는 것을 기억하자. 아이폰 시뮬레이터에서 테스트를 하면 동작하지 않을 것이다. 부록 A는 아이폰에서 테스트하기 위한 준비 과정을 설명하고 있다.

이 방법을 사용하는 것에 대한 단점은 Send 버튼을 터치할 때 메일(Mail) 애플리케이션을 가져오기 위하여 애플리케이션을 백그라운드로 밀어낸다는 점이다. 이메일이 전송된 다음에는 사용자가 직접 애플리케이션을 포그라운드(foreground)로 가져와야 한다는 것이다. 그렇지 않으면 나타나지 않을 것이기 때문이다. 애플리케이션 내에서 이메일을 작성하고 전송하기 위해서는 MFMailComposeViewController 클래스를 사용하면 된다. 다음의 도전 과제는 그렇게 할 수 있는 방법을 보여준다.

도전 과제 메일 애플리케이션을 이용하여 이메일 보내기

1. 앞에서 했던 프로젝트를 이용하자. `EmailsViewController.xib` 파일에 새로운 Round Rect button을 추가한다(그림 13-3 참조).

2. Xcode에서 Frameworks 폴더를 오른쪽-클릭하여 Add ⇨ Existing Frameworks...를 선택하고, `MessageUI.framework` 파일을 추가한다(그림 13-4 참조).

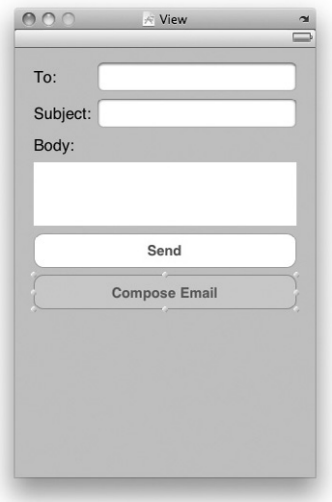

그림 13-3

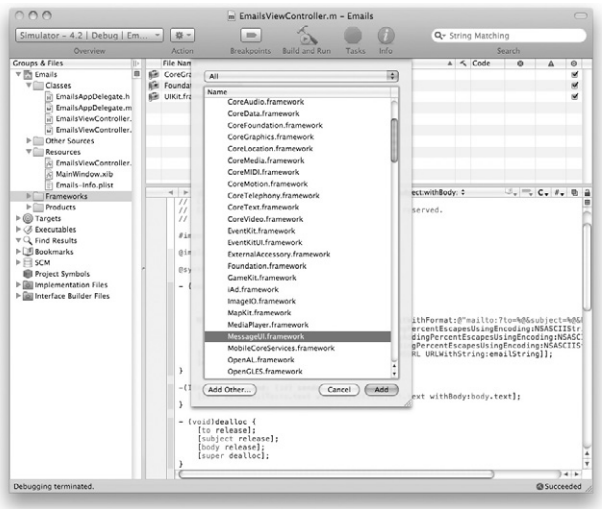

그림 13-4

3. `EmailsViewController.h` 파일에 다음의 굵게 표시된 코드를 추가한다.

```
#import <UIKit/UIKit.h>
#import <MessageUI/MFMailComposeViewController.h>

@interface EmailsViewController : UIViewController
    <MFMailComposeViewControllerDelegate> {
    IBOutlet UITextField *to;
    IBOutlet UITextField *subject;
    IBOutlet UITextView  *body;
}

@property (nonatomic, retain) UITextField *to;
@property (nonatomic, retain) UITextField *subject;
@property (nonatomic, retain) UITextView  *body;
```

```
-(IBAction) btnSend: (id) sender;
-(IBAction) btnComposeEmail: (id) sender;

@end
```

4. 인터페이스 빌더에서, Compose Email 버튼에서 Control-클릭한 상태로 드래그하여 File's Owner 항목에서 드롭하고 `btnComposeEmail:`을 선택한다.

5. `EmailsViewController.m` 파일에 다음의 굵게 표시된 코드를 추가한다.

```
#import "EmailsViewController.h"

@implementation EmailsViewController

@synthesize to, subject, body;

-(IBAction) btnComposeEmail: (id) sender {
    MFMailComposeViewController *picker =
        [[MFMailComposeViewController alloc] init];
    picker.mailComposeDelegate = self;

    [picker setSubject:@"Email subject here"];
    [picker setMessageBody:@"Email body here" isHTML:NO];
    [self presentModalViewController:picker animated:YES];
    [picker release];
}

- (void)mailComposeController:(MFMailComposeViewController*)controller
          didFinishWithResult:(MFMailComposeResult)result
                        error:(NSError*)error {
    [controller dismissModalViewControllerAnimated:YES];
}
```

6. Command-R을 눌러서 실제 아이폰에서 애플리케이션을 테스트하자. Compose Email 버튼을 누르면 앞의 도전 과제처럼 메일(Mail) 애플리케이션의 작성 화면이 나타날 것이다(그림 13-5 참조). 그러나 이전 예제와는 달리, 이메일을 전송하면 애플리케이션으로 돌아오게 된다.

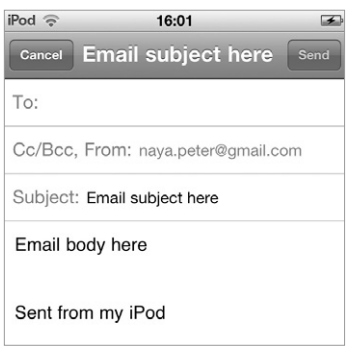

그림 13-5

동작 방법

MFMailComposeViewController 클래스는 모달하게(modally) 메시지를 작성할 수 있는 윈도우가 나타나게 하기 때문에, 현재 애플리케이션을 백그라운드로 보내지 않는다. 이것은 이메일을 전송한 후에도 계속 애플리케이션을 사용할 수 있도록 하길 원할 때 매우 유용하다.

사파리 실행하기

만약에 아이폰 애플리케이션에서 사파리(Safari) 웹 브라우저를 실행하고 싶다면, URL 문자열을 만들어서 애플리케이션 인스턴스의 openURL: 메서드를 사용하면 된다.

```
[[UIApplication sharedApplication]
    openURL:[NSURL URLWithString: @"http://www.apple.com"]];
```

위의 코드는 http://www.apple.com 페이지를 열기 위해서 사파리를 실행한다(그림 13-6 참조).

전화걸기

아이폰에서 전화를 걸기 위해서는 아래의 URL 문자열을 사용한다.

```
[[UIApplication sharedApplication]
    openURL:[NSURL URLWithString: @"tel:02-123-4567"]];
```

NOTE

위의 코드는 아이폰에서만 동작하며, 당연히 아이팟 터치에서는 동작하지 않는다. 왜냐하면 아이팟 터치는 전화 기능이 없기 때문이다. 또한 이 기능을 테스트하려면 실제 기기에서 사용해야 한다. 아이폰 시뮬레이터에서는 아무런 일도 일어나지 않을 것이다. 부록 A는 아이폰에서 테스트하기 위한 준비 과정을 설명하고 있다.

SMS 보내기

아이폰에서 SMS 메시지를 보내기 위해서는 아래의 URL 문자열을 사용한다.

```
[[UIApplication sharedApplication]
    openURL:[NSURL URLWithString: @"sms:02-123-4567"]];
```

앞의 코드는 SMS 애플리케이션을 실행시킨다(그림 13-7 참조). 현재 실행되는 애플리케이션이 백그라운드로 간다는 것에 주목하자.

그림 13-6

그림 13-7

> **NOTE**
> 앞의 절에서 언급했듯이, 위의 코드는 아이폰에서만 동작하며 아이팟 터치에서는 동작하지 않는다. 왜냐하면 아이팟 터치는 전화 기능과 문자 전송 기능이 없기 때문이다. 또한 이 기능을 테스트하려면 실제 기기에서 사용해야 한다. 아이폰 시뮬레이터에서는 아무런 일도 일어나지 않을 것이다. 부록 A는 아이폰에서 테스트하기 위한 준비 과정을 설명하고 있다.

아이폰 SDK 4에서는 애플리케이션 내에서 SMS 메시지를 직접 보낼 수도 있다. 다음의 도전 과제는 그 방법을 보여준다.

도전 과제 애플리케이션을 떠나지 않고 SMS 메시지를 보내기

1. 앞에서 했던 프로젝트를 이용하자. `EmailsViewController.h` 파일에 아래의 굵게 표시된 코드를 추가한다.

```
#import <UIKit/UIKit.h>
#import <MessageUI/MFMailComposeViewController.h>
#import <MessageUI/MFMessageComposeViewController.h>
```

```
@interface EmailsViewController : UIViewController
    <MFMailComposeViewControllerDelegate,
    MFMessageComposeViewControllerDelegate> {
    IBOutlet UITextField *to;
    IBOutlet UITextField *subject;
    IBOutlet UITextView  *body;
}

@property (nonatomic, retain) UITextField *to;
@property (nonatomic, retain) UITextField *subject;
@property (nonatomic, retain) UITextView  *body;

-(IBAction) btnSend: (id) sender;
-(IBAction) btnComposeEmail: (id) sender;
-(IBAction) btnComposeSMS: (id) sender;

@end
```

2. EmailsViewController.xib 파일을 인터페이스 빌더에서 편집할 수 있도록 열고, Round Rect button을 추가하여 이름을 Compose SMS라고 한다.

3. 그 버튼에서 Control-클릭한 상태로 드래그하여 File's Owner 항목에서 드롭하고 btnComposeSMS:를 선택한다.

4. EmailsViewController.m 파일에 아래의 굵게 표시된 코드를 추가한다.

```
-(IBAction) btnComposeSMS: (id) sender {
    MFMessageComposeViewController *picker =
        [[MFMessageComposeViewController alloc] init];
    picker.messageComposeDelegate = self;

    [picker setBody:@"This message sent from the application."];
    [self presentModalViewController:picker animated:YES];
    [picker release];
}

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

5. Command-R을 눌러서 아이폰에서 애플리케이션을 테스트하면 SMS 메시지를 작성할 수 있을 것이다(그림 13-8 참조). 메시지를 전송하면 다시 애플리케이션으로 돌아올 것이다.

그림 13-8

동작 방법

`MFMessageComposeViewController` 클래스는 아이폰 SDK에 새롭게 생긴 API들 중 하나이다. 이것은 SMS 작성 화면을 모달하게 나타내며, 현재의 애플리케이션을 백그라운드로 보내지 않는다. 이것은 SMS 메시지를 보낸 후에 현재 애플리케이션을 계속 사용하고자 할 때 매우 유용하다.

SMS 메시지 가로채기

아이폰 SDK에 가장 많이 요청되는 기능들 중 하나는 수신된 SMS 메시지를 아이폰 애플리케이션 내에서 가로채는 기능이다. 그러나 안타깝게도 현재 버전의 SDK에서는 이런 기능을 지원하지 않는다.

마찬가지로, SMS 메시지를 애플리케이션에서 직접 보낼 수도 없다. 메시지는 반드시 내장된 SMS 애플리케이션에서만 전송되어야 한다. 이것은 사용자가 알지 못한 채 악성 애플리케이션에서 SMS 메시지가 발송되는 것을 방지하기 위함이다.

카메라와 포토 라이브러리 접근하기

아이폰은 사진과 동영상 촬영 모두를 할 수 있는 카메라를 가지고 있다(아이폰 4와 아이팟 터치 4세대는 앞면과 뒷면의 2개의 카메라를 가지고 있다). 촬영된 사진들과 비디오들은 사진(Photos) 애플리케이션에 저장된다. 개발자인 여러분에게는 카메라를 조작하는 것과 사진(Photos) 애플리케이션에 저장되어 있는 사진과 비디오에 접근하는 두 가지 옵션이 있다.

- 사진 또는 비디오 촬영을 위해서 카메라를 실행할 수 있다.
- 사진 앨범에서 사용자가 사진이나 비디오를 선택할 수 있도록 사진(Photos) 애플리케이션을 실행하여 사용자가 선택한 사진이나 비디오를 애플리케이션에서 사용할 수 있다.

포토 라이브러리에 접근하기

모든 아이폰과 아이팟 터치는 모든 사진과 비디오를 저장하는 사진(Photos) 애플리케이션을 가지고 있다. 아이폰 SDK를 사용하여 사용자가 사진(Photos) 애플리케이션에서 사진이나 비디오를 선택할 수 있는 UI를 프로그램적으로 보여주기 위한 `UIImagePickerController` 클래스를 사용할 수 있다. 다음의 도전 과제는 여러분의 애플리케이션에서 이 작업을 하는 방법을 보여준다.

도전 과제 포토 라이브러리에 있는 사진 접근하기

소스 코드: 13/PhotoLibrary

1. Xcode에서 View-based Application (iPhone) 프로젝트를 만들고 이름을 `PhotoLibrary`라고 한다.
2. `PhotoLibraryViewController.xib` 파일을 인터페이스 빌더에서 편집할 수 있도록 더블클릭한다.
3. 뷰 윈도우에 다음의 뷰들을 생성한다(그림 13-9 참조).
   - Round Rect Button
   - ImageView
4. ImageView에 대한 Attributes Inspector에서 Mode 속성을 Aspect Fit으로 설정한다(그림 13-10 참조).
5. `PhotoLibraryViewController.h` 파일에 다음의 굵게 표시된 코드를 추가한다.

그림 13-9

```
#import <UIKit/UIKit.h>

@interface PhotoLibraryViewController :
UIViewController
    <UINavigationControllerDelegate,
    UIImagePickerControllerDelegate> {

    IBOutlet UIImageView *imageView;
    UIImagePickerController *imagePicker;
}
```

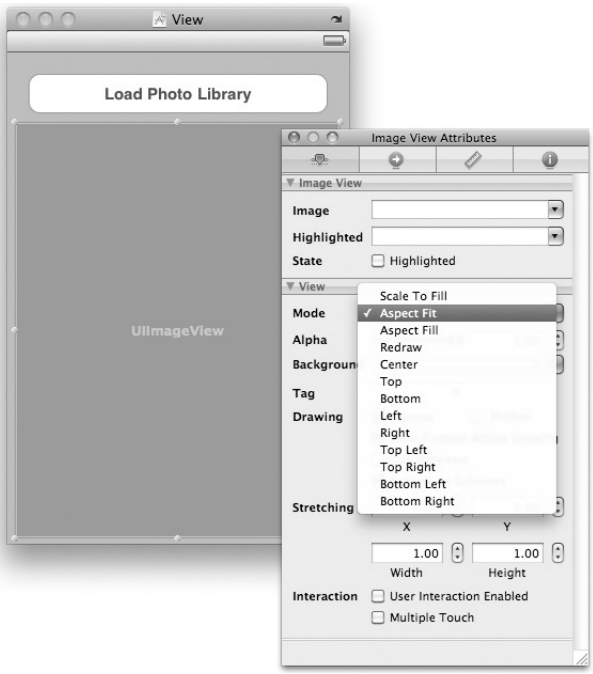

그림 13-10

```
@property (nonatomic, retain) UIImageView *imageView;

-(IBAction) btnClicked: (id) sender;

@end
```

6. 인터페이스 빌더로 돌아가서, File's Owner 항목을 Control-클릭한 상태로 드래그하여 imageView에서 드롭하고 `ImageView`를 선택한다.

7. 버튼을 Control-클릭한 상태로 드래그하여 File's Owner 항목에서 드롭하고 `btnClicked:`를 선택한다.

8. `PhotoLibraryViewController.m` 파일에 다음의 굵게 표시된 코드를 추가한다.

```
#import "PhotoLibraryViewController.h"

@implementation PhotoLibraryViewController

@synthesize imageView;
```

```
- (void)viewDidLoad {
    imagePicker = [[UIImagePickerController alloc] init];
    [super viewDidLoad];
}

- (IBAction) btnClicked: (id) sender {
    imagePicker.delegate = self;
    imagePicker.sourceType =
        UIImagePickerControllerSourceTypePhotoLibrary;

    //---이미지 픽커 표시---
    [self presentModalViewController:imagePicker animated:YES];
}

- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
    UIImage *image;
    NSURL *mediaUrl;
    mediaUrl = (NSURL *)[info valueForKey:
                         UIImagePickerControllerMediaURL];

    if (mediaUrl == nil) {
        image = (UIImage *) [info
                             valueForKey:UIImagePickerControllerEditedImage];

         if (image == nil) {
             //---선택된 원본 이미지---
             image = (UIImage *)
             [info valueForKey:UIImagePickerControllerOriginalImage];

             //---그 이미지 표시하기---
             imageView.image = image;
        }
        else { //---편집된 선택된 이미지---
            //---이미지에 적용된 크롭(crop)을 위한 사각형 얻기---
            CGRect rect =
                [[info valueForKey:UIImagePickerControllerCropRect]
                  CGRectValue];

            //---그 이미지 표시하기---
            imageView.image = image;
        }
    }
    else {
        //---선택된 비디오---
        //---나중에 구현한다---
```

```
    }
    //---이미지 픽커 감추기---
    [picker dismissModalViewControllerAnimated:YES];
}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
    //---사용자가 이미지/비디오를 선택하지 않으며, 이미지 픽커를 숨긴다---
    [picker dismissModalViewControllerAnimated:YES];
}

- (void)dealloc {
    [imageView release];
    [imagePicker release];
    [super dealloc];
}
```

9. Command-R을 눌러서 아이폰에서 애플리케이션을 테스트한다.

10. 애플리케이션이 로드되면 Load Photo Library 버튼을 누른다. 아이폰에 Photo Albums가 나타난다. 특정 앨범을 선택하고(그림 13-11 참조), 사진을 선택한다. 선택된 사진이 ImageView에 나타나게 된다(그림 13-12 참조).

그림 13-11

그림 13-12

동작 방법

아이폰에 있는 사진과 비디오를 선택하고 가져오는 UI를 제공하는 포토 라이브러리(Photo Library)에 대한 접근은 UIImagePickerController 클래스에 의해 제공된다. 여러분이 해야 할 일은 이 클래스에 대한 인스턴스를 생성하고 UIImagePickerControllerDelegate 프로토콜을 따르는 델리게이트를 제공하는 것이다. 또한 그 델리게이트는 UINavigationControllerDelegate 프로토콜도 따라야 한다. 왜냐하면 UIImagePickerController 클래스는 사용자가 포토 라이브러리에서 사진을 선택할 수 있도록 하는 내비게이션 컨트롤러를 사용하기 때문이다. 그러므로 PhotoLibraryViewController.h에 이 프로토콜들을 먼저 지정해야 한다.

```
@interface PhotoLibraryViewController : UIViewController
    <UINavigationControllerDelegate,
    UIImagePickerControllerDelegate> {
. . .
```

Load Photo Library 버튼을 클릭하면 UIImagePickerController 클래스에 의해 표시되는 픽커 인터페이스의 종류를 설정한 다음, 그것을 모달하게 표시한다.

```
- (IBAction) btnClicked: (id) sender {
    imagePicker.delegate = self;

    imagePicker.sourceType =
        UIImagePickerControllerSourceTypePhotoLibrary;

    //---이미지 픽커 표시---
    [self presentModalViewController:imagePicker animated:YES];
}
```

만일 사용자가 사진을 선택했을 때 편집할 수 있도록 하길 원한다면, 다음의 코드를 추가하면 된다.

```
    imagePicker.allowsEditing = YES;
```

디폴트로, 소스 타입은 항상 UIImagePickerControllerSourceTypePhotoLibrary이지만, 아래에 있는 것들 중 하나로 바꿀 수도 있다.

- UIImagePickerControllerSourceTypeCamera – 카메라에서 직접 사진을 찍는다.
- UIImagePickerControllerSourceTypeSavedPhotosAlbum – Photo Albums 애플리케이션으로 바로 간다.

사용자가 사진/비디오를 선택하면, 선택된 미디어의 종류를 체크하여 그에 맞는 적절한 처리를 하게 되는 imagePickerController:didFinishPickingMediaWithInfo: 이벤트가 실행된다.

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {

    UIImage *image;
    NSURL *mediaUrl;
    mediaUrl = (NSURL *)[info valueForKey:UIImagePickerControllerMediaURL];

    if (mediaUrl == nil) {
        image = (UIImage *) [info valueForKey:UIImagePickerControllerEditedImage];

        if (image == nil) {
            //---선택된 원본 이미지---
            image = (UIImage *)
            [info valueForKey:UIImagePickerControllerOriginalImage];

            //---그 이미지 표시하기---
            imageView.image = image;
        }
        else { //---편집된 선택된 이미지---
            //---이미지에 적용된 크롭(crop)을 위한 사각형 얻기---
            CGRect rect =
                [[info valueForKey:UIImagePickerControllerCropRect]
                    CGRectValue];

            //---그 이미지 표시하기---
            imageView.image = image;
        }
    }
    else {
        //---선택된 비디오---
        //---나중에 구현한다---
    }

    //---이미지 픽커 감추기---
    [picker dismissModalViewControllerAnimated:YES];
}
```

사용자에 의해 선택된 미디어 종류는 info: 매개변수에 포함된다. 적절한 미디어 종류를 추출하기 위하여 valueForKey: 메서드를 사용한 다음, 미디어 종류에 맞도록 형변환을 한다.

```
    mediaUrl = (NSURL *)[info valueForKey:UIImagePickerControllerMediaURL];
```

사용자가 선택을 취소한다면 `imagePickerControllerDidCancel:` 이벤트가 실행된다. 이번 예제에서는 이미지 픽커를 해제한다.

```
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
    //---사용자가 이미지/비디오를 선택하지 않으며 이미지 픽커를 숨긴다---
    [picker dismissModalViewControllerAnimated:YES];
}
```

위의 예제에서 주목해야 할 것은 사용자가 포토 라이브러리(Photo Library)에서 사진만 선택할 수 있다는 것이다. 사용자가 비디오를 선택하고 싶다면 어떻게 해야 할까? 다음의 도전 과제는 사용자가 포토 라이브러리에서 비디오를 선택할 수 있게 하며, 이 애플리케이션에서 재생할 수 있도록 하는 방법을 보여준다.

도전 과제 포토 라이브러리에 있는 비디오 접근하기

1. 앞에서 한 프로젝트를 이용하자. Xcode 프로젝트의 `Frameworks` 폴더에 `MediaPlayer.framework`와 `MobileCoreServices.framework` 파일을 추가한다(그림 13-13 참조).

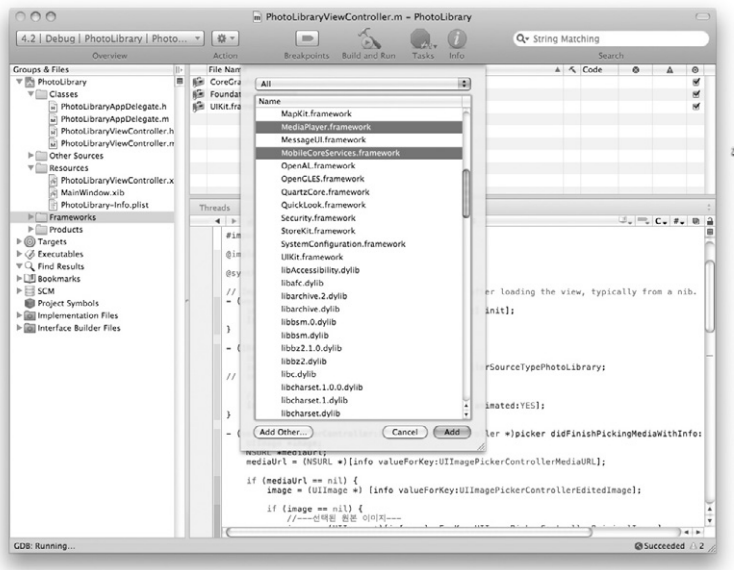

그림 13-13

2. PhotoLibraryViewController.h 파일에 다음의 굵게 표시된 코드를 삽입한다.

```
#import <UIKit/UIKit.h>
#import <MediaPlayer/MediaPlayer.h>
#import <MobileCoreServices/MobileCoreServices.h>

@interface PhotoLibraryViewController : UIViewController
    <UINavigationControllerDelegate, UIImagePickerControllerDelegate> {

    IBOutlet UIImageView *imageView;
    UIImagePickerController *imagePicker;
}

@property (nonatomic, retain) UIImageView *imageView;

-(IBAction) btnClicked: (id) sender;

@end
```

3. PhotoLibraryViewController.m 파일에 다음의 굵게 표시된 코드를 삽입한다.

```
- (IBAction) btnClicked: (id) sender {
    imagePicker.delegate = self;

    imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

    NSArray *mediaTypes =
        [NSArray arrayWithObjects:kUTTypeImage, kUTTypeMovie, nil];
    imagePicker.mediaTypes = mediaTypes;

    //---이미지 픽커 표시---
    [self presentModalViewController:imagePicker animated:YES];
}

- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
    UIImage *image;
    NSURL *mediaUrl;
    mediaUrl = (NSURL *)[info valueForKey:UIImagePickerControllerMediaURL];
    if (mediaUrl == nil) {
        image = (UIImage *) [info valueForKey:UIImagePickerControllerEditedImage];
        if (image == nil) {
            //---선택된 원본 이미지---
            image = (UIImage *)
```

```
            [info valueForKey:UIImagePickerControllerOriginalImage];

            //---그 이미지 표시하기---
            imageView.image = image;
        }
        else { //---편집된 선택된 이미지---
            //---이미지에 적용된 크롭(crop)을 위한 사각형 얻기---
            CGRect rect =
                [[info valueForKey:UIImagePickerControllerCropRect]
                       CGRectValue];

            //---그 이미지 표시하기---
            imageView.image = image;
        }
    }
    else {
        //---선택된 비디오---
        MPMoviePlayerController *player =
            [[MPMoviePlayerController alloc]
                initWithContentURL:mediaUrl];

        [[NSNotificationCenter defaultCenter]
            addObserver:self
               selector:@selector(movieFinishedCallback:)
                   name:MPMoviePlayerPlaybackDidFinishNotification
                 object:player];

        //---부분 화면에서 재생---
        player.view.frame = CGRectMake(0, 0, 320, 460);
        [self.view addSubview:player.view];

        [player play];
    }

    //---이미지 픽커 감추기---
    [picker dismissModalViewControllerAnimated:YES];
}

- (void) movieFinishedCallback:(NSNotification*) aNotification {
    MPMoviePlayerController *player = [aNotification object];
    [[NSNotificationCenter defaultCenter]
        removeObserver:self
                  name:MPMoviePlayerPlaybackDidFinishNotification
                object:player];

    [player.view removeFromSuperview];
```

```
    [player autorelease];
}
```

4. Command-R을 눌러서 아이폰에서 애플리케이션에서 테스트하자. 비디오를 선택한 다음에 Choose 버튼을 클릭한다(그림 13-14 참조). 선택된 비디오는 압축된 다음에 애플리케이션에서 재생될 것이다.

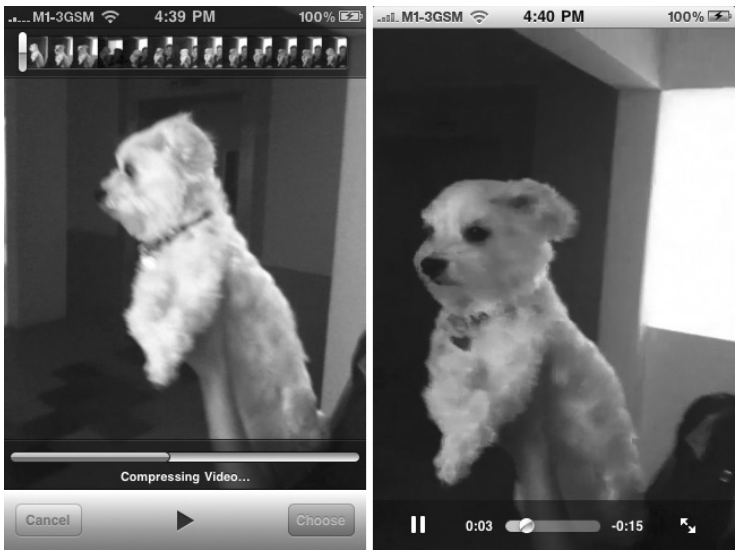

그림 3-14

동작 방법

UIImagePickerController 객체는 디폴트로 이미지를 표시한다. 비디오를 선택하고 표시할 수 있도록 하기 위해서는 UIImagePickerController 객체에 mediaTypes 속성을 설정해야 한다. 이 속성은 표시하길 원하는 미디어 종류들을 포함하는 NSArray 객체를 취한다. 여기서는 kUTTypeImage와 kUTTypeMovie로 미디어 종류를 설정한다.

```
imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

NSArray *mediaTypes =
    [NSArray arrayWithObjects:kUTTypeImage, kUTTypeMovie, nil];
imagePicker.mediaTypes = mediaTypes;

//---이미지 픽커 표시---
[self presentModalViewController:imagePicker animated:YES];
```

비디오가 선택되면 MPMoviePlayerController 클래스를 사용하여 재생하게 될 것이다.

```
    if (mediaUrl == nil) {
        //...
        //...
    }
    else {
        //---선택된 비디오---
        MPMoviePlayerController *player =
            [[MPMoviePlayerController alloc]
                initWithContentURL:mediaUrl];

        [[NSNotificationCenter defaultCenter]
            addObserver:self
               selector:@selector(movieFinishedCallback:)
                   name:MPMoviePlayerPlaybackDidFinishNotification
                 object:player];

        //---부분 화면에서 재생---
        player.view.frame = CGRectMake(0, 0, 320, 460);
        [self.view addSubview:player.view];

        [player play];
    }

    //---이미지 픽커 감추기---
    [picker dismissModalViewControllerAnimated:YES];
}
```

비디오 재생이 종료되면 뷰 윈도우에서 비디오를 제거한다.

```
- (void) movieFinishedCallback:(NSNotification*) aNotification {
    MPMoviePlayerController *player = [aNotification object];
    [[NSNotificationCenter defaultCenter]
        removeObserver:self
                  name:MPMoviePlayerPlaybackDidFinishNotification
                object:player];

    [player.view removeFromSuperview];
    [player autorelease];
}
```

카메라 접근하기

포토 라이브러리(Photo Library)에 접근하는 것 말고도, 아이폰의 카메라도 접근할 수 있다. 하드웨어 접근이 다음 장의 주제이지만, UIImagePickerController 클래스를 사용하여 카메라에 접근할 수 있으므로 그 방법을 여기서 살펴보도록 하자.

다음의 도전 과제에서는 이전 절에서 생성한 프로젝트를 수정한다. 많은 코드를 수정하지 않을 것이다. 왜냐하면 작성했던 대부분의 코드를 그대로 적용할 것이기 때문이다.

도전 과제 카메라 활성화하기

1. 앞 절에서 생성했던 프로젝트를 이용하자. PhotoLibraryViewController.m 파일에서 이미지 픽커의 소스 타입을 카메라로 변경한다(굵게 표시된 코드).

```
- (IBAction) btnClicked: (id) sender {
    imagePicker.delegate = self;

    //---아래는 주석처리---
    /*
    imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    NSArray *mediaTypes =
        [NSArray arrayWithObjects:kUTTypeImage, kUTTypeMovie, nil];
    imagePicker.mediaTypes = mediaTypes;
    */

    //---카메라 실행---
    imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
    NSArray *mediaTypes =
        [NSArray arrayWithObjects:kUTTypeImage, kUTTypeMovie, nil];

    imagePicker.mediaTypes = mediaTypes;

    imagePicker.cameraCaptureMode = UIImagePickerControllerCameraCaptureModeVideo;
    imagePicker.allowsEditing = YES;

    //---이미지 픽커 표시---
    [self presentModalViewController:imagePicker animated:YES];
}
```

2. PhotoLibraryViewController.m 파일에 다음의 메서드 두 개를 정의한다.

```
- (NSString *) filePath: (NSString *) fileName {
    NSArray *paths =
        NSSearchPathForDirectoriesInDomains(
            NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDir = [paths objectAtIndex:0];
    return [documentsDir stringByAppendingPathComponent:fileName];
}

- (void) saveImage{
    //---이미지 뷰에서 데이터를 가져옴---
    NSData *imageData =
    [NSData dataWithData:UIImagePNGRepresentation(imageView.image)];

    //---그 데이터를 파일에 기록---
    [imageData writeToFile:[self filePath:@"MyPicture.png"] atomically:YES];
}
```

3. 다음의 굵게 표시된 코드를 삽입한다.

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
    UIImage *image;
    NSURL *mediaUrl;
    mediaUrl = (NSURL *)[info valueForKey:UIImagePickerControllerMediaURL];

    if (mediaUrl == nil) {
        image = (UIImage *)
        [info valueForKey:UIImagePickerControllerEditedImage];

        if (image == nil) {
            //---선택된 원본 이미지---
            image = (UIImage *)
            [info valueForKey:UIImagePickerControllerOriginalImage];

            //---그 이미지 표시하기---
            imageView.image = image;

            //---캡처한 이미지 저장---
            [self saveImage];
        }
        else { //---편집된 선택된 이미지---

            //---이미지에 적용된 크롭(crop)을 위한 사각형 얻기---
            CGRect rect =
                [[info valueForKey:UIImagePickerControllerCropRect]
```

```
                    CGRectValue];

            //---그 이미지 표시하기---
            imageView.image = image;

            //---캡처한 이미지 저장---
            [self saveImage];
        }
    }
    else {
        //---선택된 비디오---
        MPMoviePlayerController *player =
            [[MPMoviePlayerController alloc]
                initWithContentURL:mediaUrl];

        [[NSNotificationCenter defaultCenter]
            addObserver:self
               selector:@selector(movieFinishedCallback:)
                   name:MPMoviePlayerPlaybackDidFinishNotification
                 object:player];

        //---부분 화면에서 재생---
        player.view.frame = CGRectMake(0, 0, 320, 460);
        [self.view addSubview:player.view];

        [player play];
    }

    //---이미지 픽커 감추기---
    [picker dismissModalViewControllerAnimated:YES];
}
```

4. Command-R을 눌러서 실제 아이폰에서 애플리케이션을 테스트해보자.

5. Load Photo Library 버튼을 터치하면 아이폰의 카메라를 이용하여 사진과 비디오를 찍을 수 있게 된다. 사진을 찍으면(그림 13-15 참조) 사진은 애플리케이션의 Documents 폴더에 저장된다. 만약에 비디오를 찍으면 아이폰의 미디어 플레이어를 사용하여 재생될 것이다(그림 13-16 참조).

그림 13-15

그림 13-16

동작 방법

이번 예제에서는 이미지 픽커의 소스 타입을 카메라로 수정하였다.

```
imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
```

카메라로 사진을 촬영하면 그 사진은 다시 `imagePickerController:didFinishPickingMediaWithInfo:` 메서드로 전달되며, 이미지뷰에 표시된다. 아이폰의 어떤 위치에 그 이미지를

직접 저장하는 것은 여러분의 몫이다. 이 예제에서는 filePath: 메서드에서 애플리케이션의 Documents 폴더에 사진을 저장하도록 정의했다.

```
- (NSString *) filePath: (NSString *) fileName {
    NSArray *paths =
        NSSearchPathForDirectoriesInDomains(
            NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDir = [paths objectAtIndex:0];
    return [documentsDir stringByAppendingPathComponent:fileName];
}
```

saveImage: 메서드는 이미지뷰에서 이미지 데이터를 추출한 다음, MyPicture.png라는 이름의 파일로 데이터를 저장하기 위해서 filePath: 메서드를 호출한다.

```
- (void) saveImage{
    //---이미지 뷰에서 데이터를 가져옴---
    NSData *imageData =
    [NSData dataWithData:UIImagePNGRepresentation(imageView.image)];

    //---그 데이터를 파일에 기록---
    [imageData writeToFile:[self filePath:@"MyPicture.png"] atomically:YES];
}
```

비디오 녹화를 위해 아이폰의 카메라로 캡처된 비디오는 기기에 저장되고 URL을 반환했고, 그 비디오를 다시 재생하기 위해서는 MediaPlayer 프레임워크를 사용하는 MPMoviePlayerController를 사용한다.

NOTE

기본적으로 아이폰 4는 UIImagePickerController 클래스를 사용하면 뒷쪽 카메라가 항상 활성화된다. 만약에 뒷쪽 카메라 대신 앞쪽 카메라를 활성화하고 싶다면, UIImagePickerController 클래스의 cameraDevice 속성을 UIImagePickerControllerCameraDeviceRear(디폴트), 또는 UIImagePickerControllerCameraDeviceFront으로 설정하면 된다.

NOTE

부록 A는 아이폰에서 테스트하는 방법을 설명하고 있다.

요약

이번 장에서는 아이폰 애플리케이션에 내장된 다양한 애플리케이션들을 쉽게 통합하는 방법을 배웠다. 특히 URL을 이용하여 SMS와 메일(Mail), 사파리(Safari), 그리고 전화걸기(Phone) 방법을 배웠다. 또한 애플리케이션을 떠나지 않고 SMS와 이메일 메시지를 보내는 방법도 배웠으며, 아이폰 SDK가 제공하는 클래스들을 사용하여 포토 라이브러리(Photo Library) 애플리케이션에 접근하는 방법을 배웠다.

연습 문제

1. SMS와 메일(Mail), 사파리(Safari), 그리고 전화(Phone) 애플리케이션을 실행하기 위한 URL은 각각 무엇인가?
2. 아이폰에서 Image Picker UI를 실행하기 위한 클래스는 무엇인가?
3. 아이폰에서 Mail Composer UI를 실행하기 위한 클래스는 무엇인가?
4. 아이폰에서 Message Composer UI를 실행하기 위한 클래스는 무엇인가?

연습문제에 대한 답은 부록 E를 참조하자.

✖ 이번 장에서 배운 내용

주제	주요 내용
애플리케이션 내에서 이메일 전송	`NSString *emailString = @"mailto:?to=user@email.com&subject=Subject&body=Body", [[UIApplication sharedApplication] openURL:[NSURL URLWithString:emailString]];`
사파리(Safari) 실행	`[[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"http://www.apple.com"]];`
전화(Phone) 실행	`[[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"tel:0123456789"]];`
SMS 실행	`[[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"sms:0123456789"]];`
포토 라이브러리 접근	`UIImagePickerController` 클래스를 사용하고 뷰 컨트롤러가 `UINavigationControllerDelegate` 프로토콜을 따르도록 한다.
메일 작성(Mail Composer) UI 실행	`MFMailComposeViewController` 클래스를 사용한다.
메시지 작성(Message Composer) UI 실행	`MFMessageComposeViewController` 클래스를 사용한다.