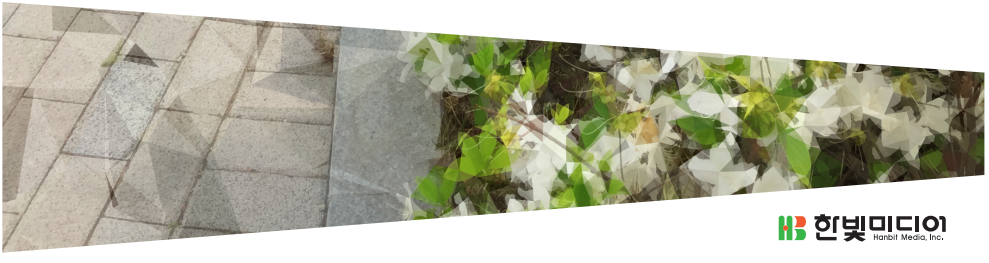
Hanbit RealTime 100

iOS 8 핵심 노트

야곰 지음



한빛미디어 Hanbit Media, Inc.

iOS 8 핵심 노트

야곰 지음

한빛미디어
Hanbit Media, Inc.

iOS 8 핵심 노트

초판발행 2015년 5월 15일

지은이 야곰 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-749-1 15000 / **정가** 12,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 정지연
디자인 표지/내지 여동일, 조판 최송실
마케팅 박상용 / **영업** 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

저자 소개

지은이_ **야곰**

yagom's blog(http://blog.yagom.net/)를 운영하는 iOS 개발 블로거이자 iOS 프로그래머로, 아이폰 개발자 커뮤니티인 맥부기(http://cafe.naver.com/mcbugi)에서 강좌를 연재 중이다. 컴퓨터 교육을 전공했으며 비전공자와 학생들에게 컴퓨터 지식을 더 쉽고 재미있게 알리는 데 관심이 많다. 2010년부터 iOS 개발을 시작해 현재까지도 계속 iOS 관련 개발에 열정을 쏟고 있다. 내일 걱정은 모레 하는 것이 좋다고 생각하며 스스로 긍정적인 마음가짐을 빼면 시체라고 말한다. 스스로 개발자라고 생각하지 않는 것을 보면 괴짜임이 틀림없다. 무엇보다 여행과 요리를 좋아한다.

스마트폰의 수요가 늘어남에 따라 사용자가 필요로 하는 기능도 많아졌습니다. 그에 발맞추어 모바일 OS도 제각기 발전을 거듭하고 있고 iOS도 사용자의 편의를 위해 점점 더 빠르게 진화해 가고 있습니다. 벌써 iOS 8.3 버전이 배포되었지만, 아직도 iOS 8에 대응하지 못한 애플리케이션이 더 많은 것 같습니다. 그만큼 이번 iOS 8은 지난 어떤 OS 업데이트보다 많은 기능이 업데이트되었습니다. 물론 그것이 OS 안정성에 영향을 미친 것 같기도 하지만 새로운 많은 기능은 사용자와 개발자들에게 큰 환영을 받았습니다.

이 책에 iOS 8에서 추가된 모든 기능을 담지는 못했지만, 공통으로 사용할 수 있는 굵직굵직한 신규 기능 위주로 정리해 보았습니다.[01] 장마다 해당 기능을 간단히 설명하고 예제를 함께 수록하였습니다. 재빠르게 변하는 OS에 발맞추어 나가는 데 작지만 큰 도움이 되면 좋겠습니다.

늦었지만 iOS 8 관련 책을 마감하였습니다. 늦어진 원고에도 불구하고 밤낮없이 집필하는 데 많은 도움을 주신 한빛미디어 여러분(특히 정지연 님)께 감사의 말씀을 드립니다. 또 어쩌다 보니 〈한빛 리얼타임〉 100호가 되어 관계자 여러분의 기대를 받게 되었는데, 몸 둘 바를 모르겠습니다. 앞으로도 〈한빛 리얼타임〉 1000호, 10000호가 계속 탄생하면 좋겠습니다. 항상 응원하고 격려해 주시는 부모님과 가족, 친구, 동료들께 감사의 말을 전합니다. 사랑합니다.

01 iOS 8에 관한 더 많은 정보를 얻으려면 https://developer.apple.com/kr/ios8/을 방문해 보세요.

대상 독자 및 참고사항

이 책은 이미 iOS 개발 경험이 있는 개발자 중에서 빠르게 iOS 8의 기능과 예제를 훑어보고자 하는 독자를 대상으로 합니다. Xcode 사용법과 Cocoa Touch 프레임워크를 이해하고 있으며, 관련 키워드들을 이해할 수 있어야 합니다. 또한, Swift 기본 문법을 알고 있으면 좋습니다. iOS를 처음 접하거나 Cocoa Touch 프레임워크에 관한 전반적인 이해가 없는 독자는 소화하기 어려울 수 있습니다.

이 책의 예제는 Xcode 6.0~6.3, iOS SDK 8.0~8.3, Swift 1.2 버전 기준으로 작성되었으며, 언어 및 iOS SDK, Xcode 버전에 따라 코드가 동작하지 않을 수도 있습니다.

이 책에서 사용한 코드는 다음에서 내려받으면 됩니다.

- https://bitbucket.org/yagom/ios8_yagom

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾기도 쉽지 않습니다. 또한, 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1 eBook First – 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 좀 더 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한, 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2 무료로 업데이트되는 전자책 전용 서비스입니다

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3 독자의 편의를 위해 DRM-Free로 제공합니다

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 어려운 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

chapter 1

애플리케이션의 확장

1.1 익스텐션 개요

익스텐션[01]은 iOS 8에서 처음으로 선보인 매우 흥미로운 세계입니다. 내 애플리케이션 안에서의 동작뿐만 아니라 다른 애플리케이션에서도 내가 직접 구현한 기능을 동작할 수 있게 되었습니다. 잘 이해되지 않는다고요? iOS에서 제공하는 특정한 기본 기능들(공유, 액션 등)로 내가 직접 구현한 기능을 사용할 수 있다는 뜻입니다.

먼저 부모 앱을 만든 후 여러 개의 익스텐션을 붙일 수 있습니다. 익스텐션은 하나의 완전체가 될 수 없지만 부모 앱을 설치하는 동시에 설치되어 iOS 전반에 걸쳐 사용할 수 있습니다. 그럼 활용도가 무궁무진한 익스텐션 자세히 알아볼까요?

1.2 공유

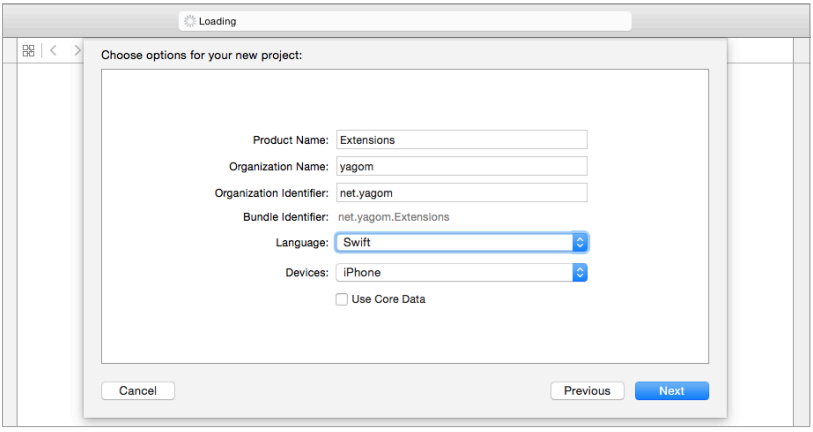
공유(Share) 익스텐션은 사용자가 특정한 유형의 자료(사진, 비디오, 웹 사이트 등)를 공유할 수 있는 방법을 제공합니다. 말로 풀어쓰니 잘 이해가 되지 않지요? 쉽게 말해 사진을 공유하려고 할 때 iOS 7까지는 iOS에서 기본으로 제공하는 옵션들(메시지로 보내기, 메일로 보내기, 페이스북 공유 등)을 통해서만 다른 사람에게 공유할 수

01 Extension : '확대', '확장' 등으로 번역할 수 있지만 원어 발음을 살려 표기합니다.

있었지만, 이제는 특정 앱에서 원하는 공유 옵션을 만들 수 있다는 뜻입니다. 새로운 SNS 앱을 개발하고 있다면 더할 나위 없이 좋은 기능입니다. 그럼 직접 만들어 봅시다. :)

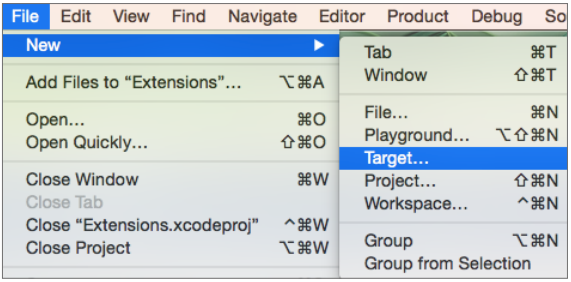
먼저 다음과 같이 'Extensions'라는 새로운 Single View Application 프로젝트를 생성합니다. 익스텐션의 몇몇 예제는 'Extensions'라는 한 프로젝트에서 진행하겠습니다.

그림 1-1 Single View Application 템플릿을 이용한 프로젝트 생성



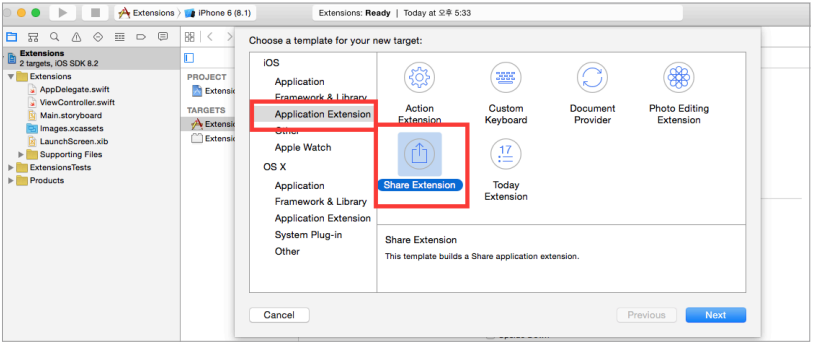
익스텐션은 보통 새로운 타깃으로 생성하는데, Xcode 메뉴에서 [File → New → Target...]을 선택합니다.

그림 1-2 새로운 타깃 생성 메뉴



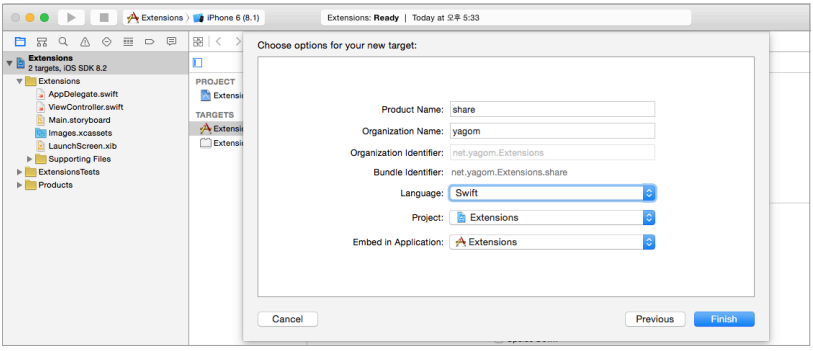
좌측 메뉴 중 [Application Extension]에서 [Share Extension] 템플릿을 선택하여 타깃을 생성합니다.

그림 1-3 Application Extension 템플릿을 이용한 타깃 생성



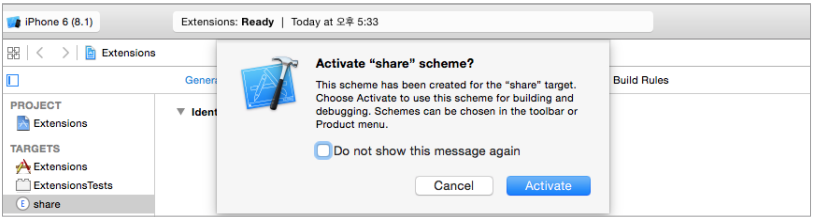
타깃 이름은 'share'로 지정합니다.

그림 1-4 새로운 타깃의 옵션 지정



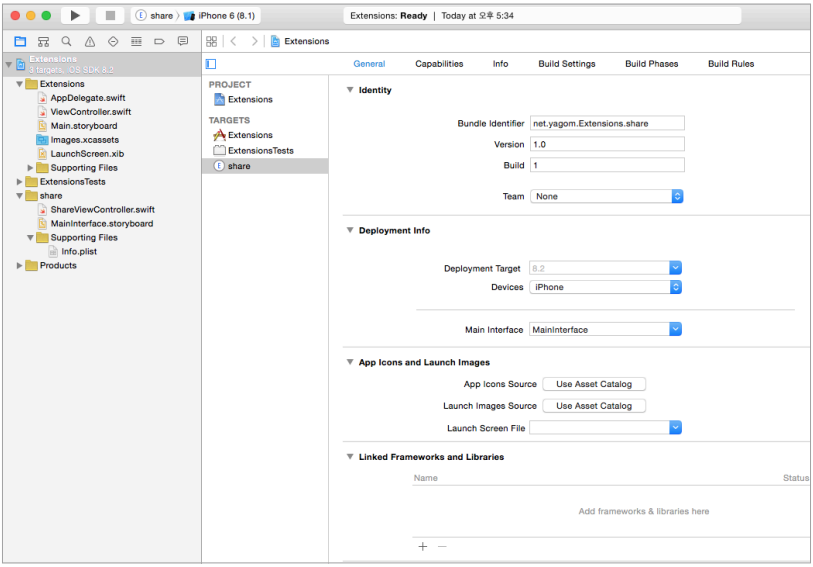
[Finish] 버튼을 누른 후 'share scheme를 활성화할 것인지' 물으면 활성화해 주세요. 어차피 실행할 때 scheme은 선택하면 그만이니까요. :)

그림 1-5 새로 생성된 타깃의 Scheme 활성화 얼럿창



이렇게 새로운 타깃이 생성되었습니다.

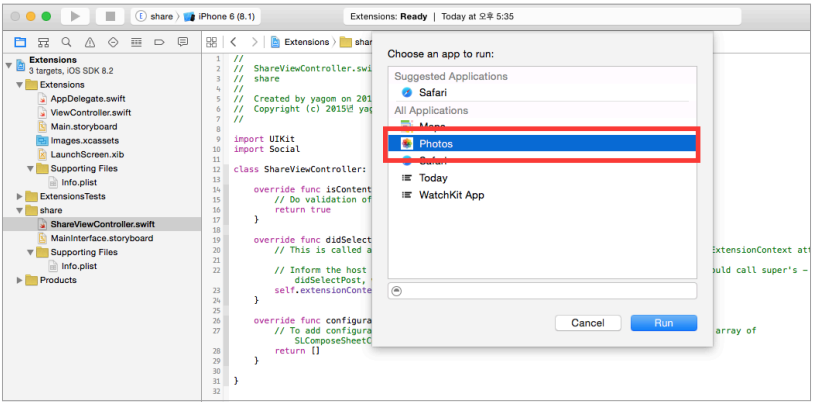
그림 1-6 새로 생성된 share 타깃



실행 Scheme이 'share'로 지정된 상태라면 단축키 [⌘+R] 또는 메뉴에서 [Product → Run]을 선택하여 무작정 실행해 봅니다. 실행하려고 하니 실행할 앱을 선택하라는 창이 나옵니다. 익스텐션은 이처럼 다른 앱에서 하나의 기능처럼 동작하게 됩니다.

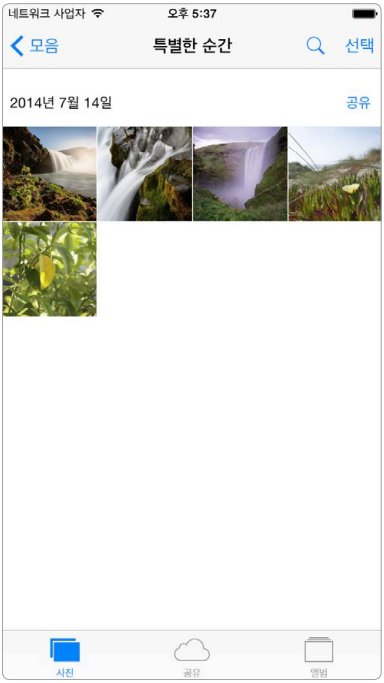
그럼 'Photos(사진첩)' 앱을 선택하여 실행해 봅시다.

그림 1-7 익스텐션 실행 옵션



기본 사진첩 앱이 실행되었습니다. 우측 위에 [선택(Select)] 버튼을 누르고 사진첩에서 마음에 드는 사진을 고른 다음 사진첩 하단의 공유 아이콘을 누릅니다.

그림 1-8 share 익스텐션을 사용할 수 있는 상태의 기본 사진첩 앱



[share]라는 아이콘이 보이면 꾹 눌러 줍니다. [share] 아이콘이 보이지 않으면 [기타(More)] 버튼을 클릭하여 [share] 아이콘을 활성화해 주세요. 새로운 공유창이 생성됩니다. [Post] 버튼을 누르면 창이 사라지지만 특별한 동작이 일어나지는 않습니다. 아직 별도의 동작을 지정하지 않았기 때문이죠.

그림 1-9 share 아이콘 활성화

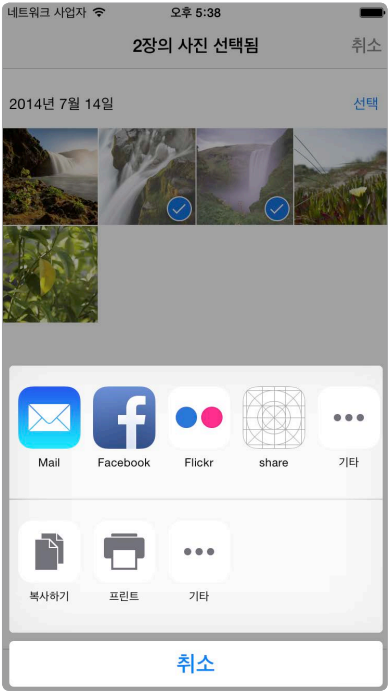
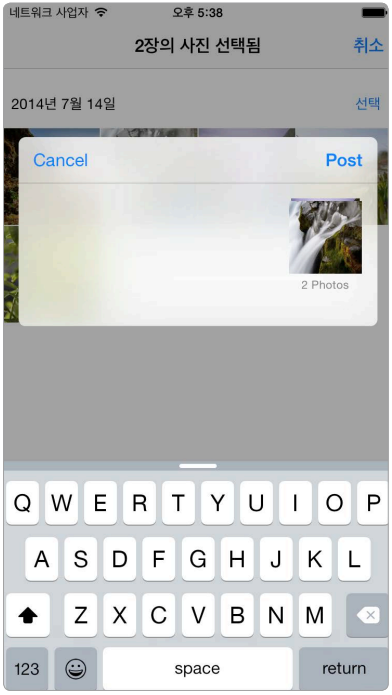


그림 1-10 공유 내용 입력화면



이제 특정 동작을 코딩하면 되겠군요. [Extensions → share → ShareViewController.swift] 파일을 열고 다음과 같이 작성합니다.

[코드 1-1] ShareViewController.swift 파일의 전체 코드

```
import UIKit
import Social

class ShareViewController: SLComposeServiceViewController {

    // 공유할 때 적절한 콘텐츠를 작성하고 있는지 체크해 허용 여부를 결정합니다.
```

```
    override func isContentValid() -> Bool {

        // 예를 들어 메시지의 길이를 체크합니다.
        // 메시지 길이를 가져와서
        let messageLength = count(self.contentText)
        // 10글자가 넘으면
        if messageLength > 10 {
            // 불허합니다(화면에서 Post 버튼이 비활성화됩니다).
            return false
        }
        // 그렇지 않다면 공유를 허용합니다.
        return true
    }
    /*
    // 이곳에 적절한 뷰를 생성하면 미리보기 화면을 직접 제작할 수 있습니다.
    override func loadPreviewView() -> UIView! {
        return nil
    }
    */

    /*
    // 취소 버튼을 선택하면 호출되는 함수입니다.
    override func didSelectCancel() {

    }
    */

    // 공유 화면에서 Post 버튼을 눌렀을 때 호출되는 함수입니다.
    override func didSelectPost() {
        // 공유를 위해 선택된 아이템들
        let inputItems: [NSExtensionItem]? = self.extensionContext?.inputItems
as? [NSExtensionItem]

        // 만약 아이템들이 nil이 아니라면
        if let items = inputItems {
            // 이곳에다 적절히 커스텀 공유 작업을 진행하는 코드를 작성합니다.
            /*
            ...
            */
        // 호스트 앱에게 작업이 완료되었음을 알려줍니다.
        self.extensionContext!.completeRequestReturningItems(items,
completionHandler: nil)
        } else {
```

```
            // 아이템이 제대로 전달되어 오지 않았다면 취소합니다.
            self.didSelectCancel()
        }
    }
    // 제어 옵션 등을 선택할 수 있는 아이템을 반환하는 함수입니다.
    override func configurationItems() -> [AnyObject]! {

        // 임의의 아이템을 만들어 봅니다.
        var item = SLComposeSheetConfigurationItem()
        // 버튼의 제목을 정해주고
        item.title = "Configure something"
        // 터치했을 때 수행할 동작을 클로저로 구현해 줍니다.
        // pushConfigurationViewController 함수 등을 참고하면 더 좋습니다.
        item.tapHandler = {

            let alert = UIAlertController(title: nil, message: "Configure",
preferredStyle: UIAlertControllerStyle.Alert)

            let okAction = UIAlertAction(title: "Okay", style:
UIAlertActionStyle.Cancel, handler: nil)

            alert.addAction(okAction)

            self.presentViewController(alert, animated: true, completion: nil)
        }
        // 아이템은 여러 개를 배열 형태로 전달합니다.
        return [item]
    }
}
```

코드를 작성한 후 다시 실행해 보면 [그림 1-11]과 같이 제어 옵션(Configure something)도 생성되어 있고, 10글자 넘게 작성했을 때에는 [Post] 버튼이 비활성화되는 것을 볼 수 있습니다. 제어 옵션에서는 사진에 효과를 주는 등 다양한 동작을 구현할 수 있습니다.

그림 1-11 제어 옵션이 추가된 공유 화면

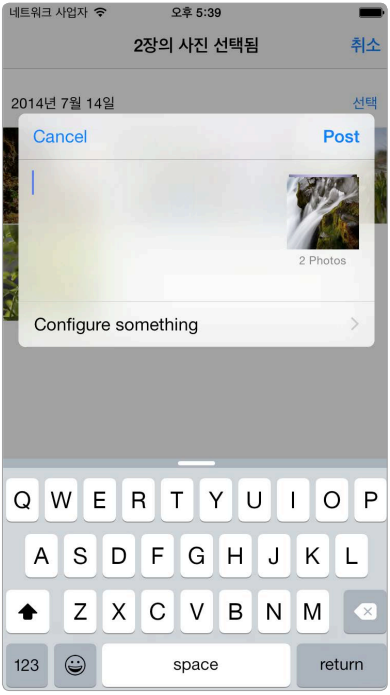
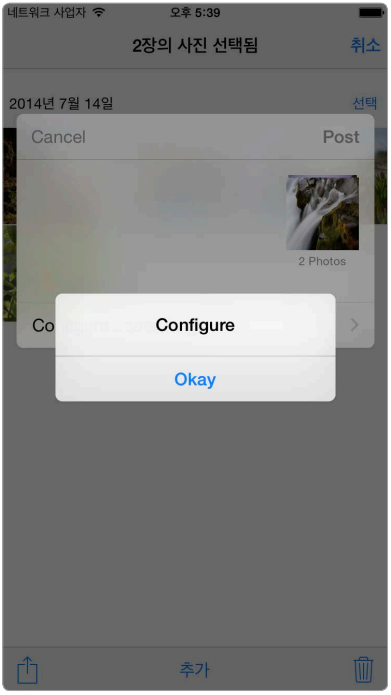


그림 1-12 제어 옵션 실행결과



더 다양한 옵션은 SLComposeServiceViewController의 구현을 살펴보거나 SLComposeServiceViewController에 관한 개발문서를 참고하면 정보를 얻을 수 있습니다. 공유하려는 데이터의 종류나 개수를 제한하려면 info.plist 파일에서 NSExtensionActivationRule[02]을 편집하면 됩니다(애플 도큐먼트 및 부록 참고).

1.3 액션

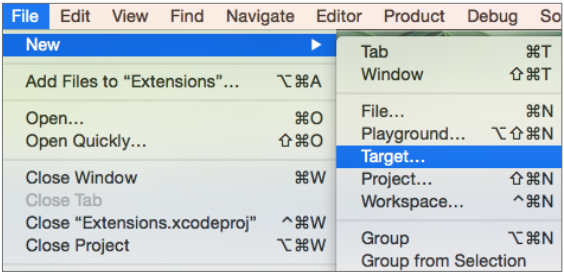
액션(Action) 익스텐션은 Activity View Controller에 특정 아이템을 넣고 실행할 때 수행할 기능들을 구현하는 익스텐션입니다. 즉, 텍스트를 Activity View

02 http://apple.co/1D6tRXL

Controller에 아이템으로 지정하고 보여주면 직접 생성한 익스텐션을 사용하여 특정 기능을 수행할 수 있습니다. 설명만으로는 이해하기 어려우니 직접 한번 해보겠습니다. `UIActivityViewController`에 텍스트를 아이템으로 지정하여 실행될 액션 익스텐션을 만들어 보겠습니다.

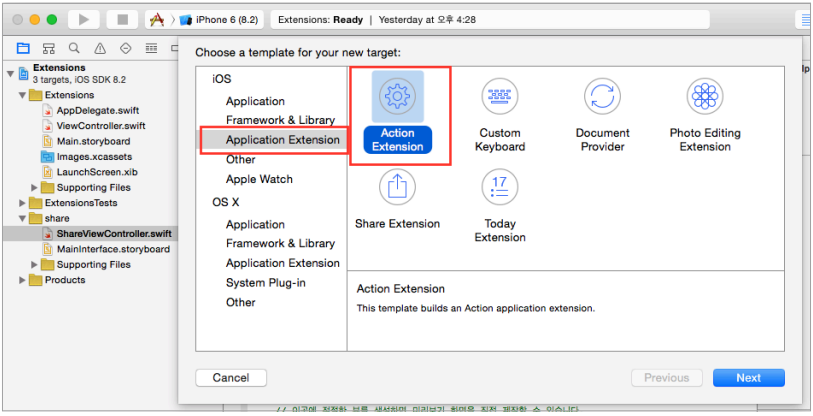
1.2 공유에서 생성한 Extensions 프로젝트에 이어서 만듭니다. 1.2 공유의 예제를 따라 해보지 않았다면 먼저 'Extensions'라는 Single View Application 프로젝트를 생성하고 액션 익스텐션을 만들기 위한 새로운 타깃을 생성합니다.

그림 1-13 새로운 타깃 생성 메뉴



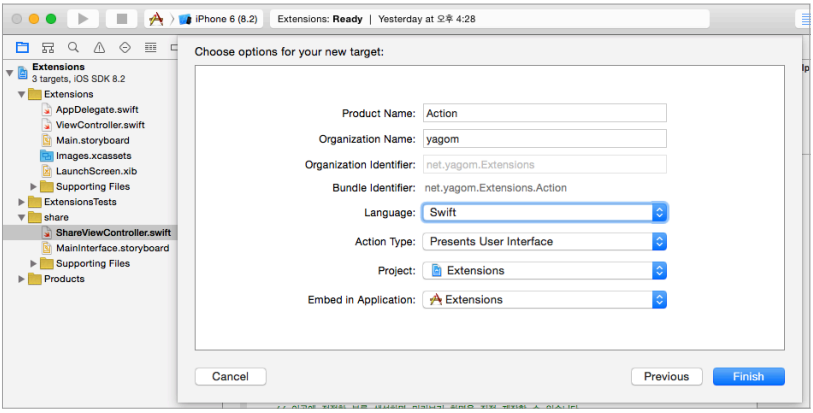
그다음 [Application Extension]에서 [Action Extension] 템플릿을 선택합니다.

그림 1-14 새로운 타깃의 템플릿 선택



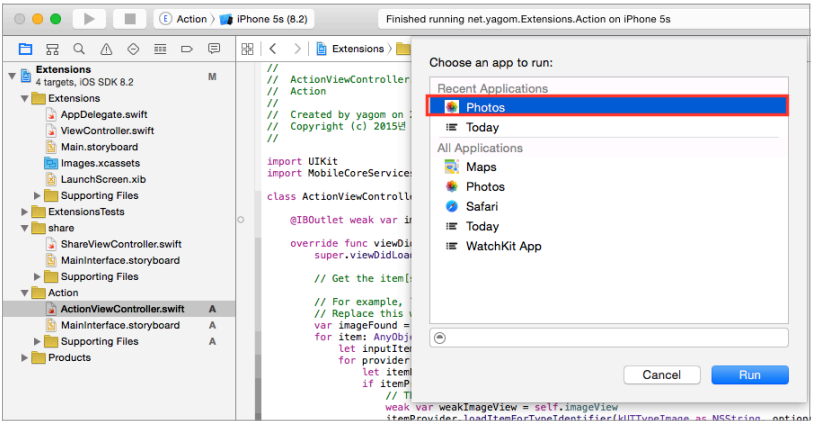
새로운 타깃의 이름은 'Action', [Action Type]은 'Presents User Interface'로 지정합니다. 이는 액션을 실행했을 때 새로 만든 UI를 보여줄지 결정하는 항목입니다.

그림 1-15 Action 익스텐션 타깃 옵션 설정



마찬가지로 Scheme을 활성화할지 물어보면 [Activate] 버튼을 눌러 활성화합니다. Scheme이 활성화되면 액션 익스텐션을 기본 사진첩 앱으로 실행합니다.

그림 1-16 액션 익스텐션 실행 옵션 중 사진첩 애플리케이션 선택



실행된 사진첩 앱에서 사진을 하나 고르고 왼쪽 아래의 액션 버튼을 누릅니다. 액티비티 컨트롤러Activity Controller가 실행되면 액티비티 메뉴 중 [Action] 아이콘을 선택합니다. [Action] 아이콘이 없다면 [More(기타)] 아이콘을 선택하여 Action 항목을 활성화해 주세요.

그림 1-17 실행된 사진첩 앱에서 사진 선택

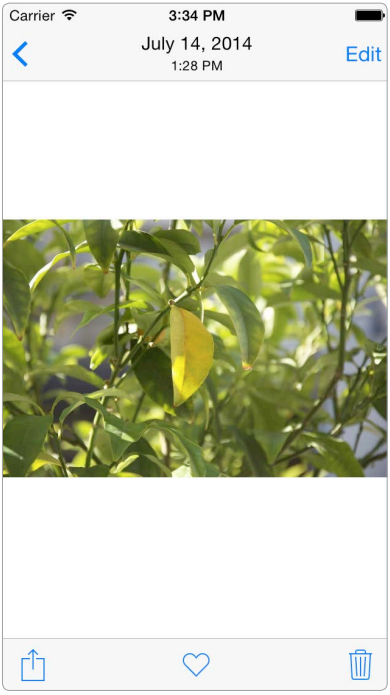
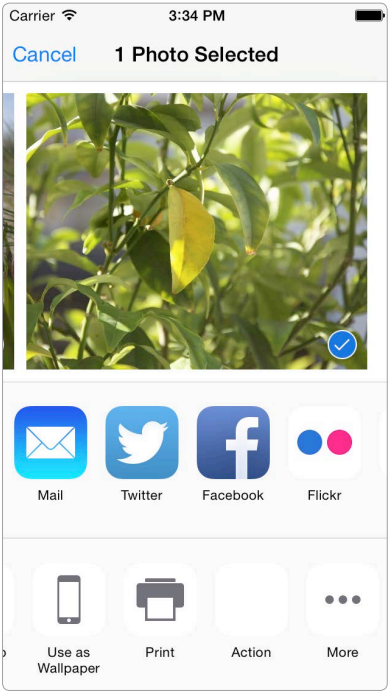
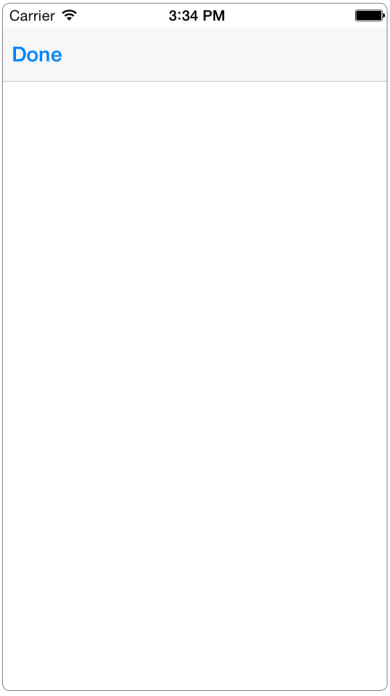


그림 1-18 액티비티 컨트롤러가 활성화된 모습



Action 익스텐션이 실행되면 빈 화면이 보입니다. 템플릿만으로도 뭔가 실행되고 있는 것 같아 좋습니다.

그림 1-19 실행된 Action 익스텐션



이제 Action을 만들어 보겠습니다. 여기서 만들려는 액션은 '읽어주기'입니다. 먼저 Action의 ActionViewController.swift 파일을 열어주세요. 그리고 다음과 같이 코드를 작성합니다.

[코드 1-2] ActionViewController.swift 파일의 전체 코드

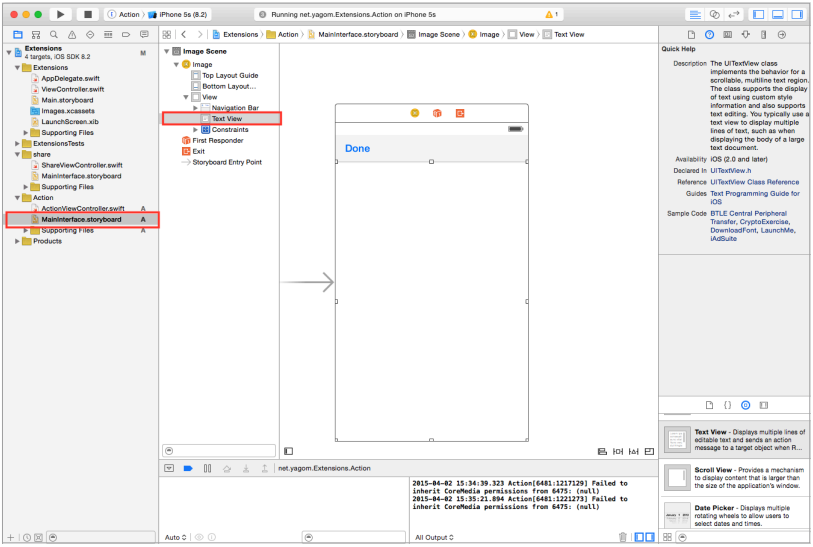
```
import UIKit
import MobileCoreServices
import AVFoundation

class ActionViewController: UIViewController {

    @IBOutlet weak var textView: UITextView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // for문을 실행할 때 텍스트를 찾았는지 확인해 줄 flag
```

```
        var textFound = false

        // 익스텐션을 실행할 때 함께 들어온 아이템들을 체크하며 순회할 for문
        for item: AnyObject in self.extensionContext!.inputItems {

            // 들어온 아이템 중 하나
            let inputItem = item as! NSExtensionItem

            // 들어온 아이템의 첨부 데이터들을 순회하는 for문
            for provider: AnyObject in inputItem.attachments! {

                // item 제공자
                let itemProvider = provider as! NSItemProvider

                // item 제공자가 텍스트를 가지고 있다면
                if itemProvider.hasItemConformingToTypeIdentifier(kUTTypePlain
Text as String) {

                    // 화면에 있는 텍스트뷰 - 튜플 안에서는 self 키워드를 사용하
여 접근하면 메모리 관리가 어려워질 수 있으므로 weak 변수를 생성합니다.
                    weak var weakTextView = self.textView

                    // item 제공자에게서 텍스트를 가져옵니다.
                    itemProvider.loadItemForTypeIdentifier(kUTTypePlainText as
String, options: nil, completionHandler: { (text, error) in

                        // 텍스트가 nil이 아니라면
                        if text != nil {

                            // Main Operation Queue에 작업을 추가합니다.
                            NSOperationQueue.mainQueue().addOperationWithBlock {

                                // 텍스트뷰가 존재한다면
                                if let textView = weakTextView {

                                // 가져온 텍스트를 세팅하고
                                    textView.text = text as? String

                                // 텍스트를 읽어주기 위한 AVSpeechSynthesizer 생성
                                    let speecher = AVSpeechSynthesizer()

                                // 텍스트를 읽어주기 위한 Utterance(발언자) 생성
                                    let utterance = AVSpeechUtterance(string:
text as! String)

                                // 최대한 느리게 읽어주세요. 가장 듣기 좋은 rate
는 0.1 정도입니다.

                                    utterance.rate = AVSpeechUtteranceMinimumSp
eechRate
```

```
                                    // 한글을 읽기 위한 언어 설정
                                    utterance.voice = AVSpeechSynthesisVoice(lan
guage: "ko-KR")

                                    // 읽어주세요.
                                    speecher.speakUtterance(utterance)
                                }
                            }
                        }
                    })

                    // 텍스트를 찾았다는 flag를 설정합니다.
                    textFound = true
                    break
                }
            }

            if (textFound) {
        // 텍스트만 취급하므로 텍스트를 발견했다면 더는 실행할 필요가 없습니다.
                break
            }
        }
    }

    @IBAction func done() {
        // 액션을 실행한 후 호스트 애플리케이션에 응답을 돌려보냅니다. item을 수
정했다면 수정한 item을 넣어주면 됩니다.
        self.extensionContext!.completeRequestReturningItems(self.
extensionContext!.inputItems, completionHandler: nil)
    }
}
```

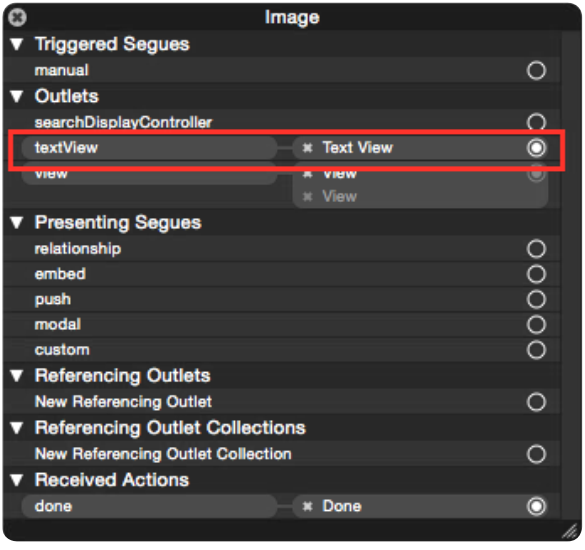
코드 작성이 끝나면 Action의 MainInterface.storyboard 파일을 열어주세요. 기존에 있던 이미지뷰를 없애고 대신 텍스트뷰를 얹어주세요.

그림 1-20 뷰컨트롤러 위에 얹어진 텍스트뷰



텍스트뷰를 IBOutlet으로 연결하고 텍스트뷰가 IBOutlet에 잘 연결되었는지 확인해 주세요.

그림 1-21 IBOutlet으로 연결된 텍스트뷰



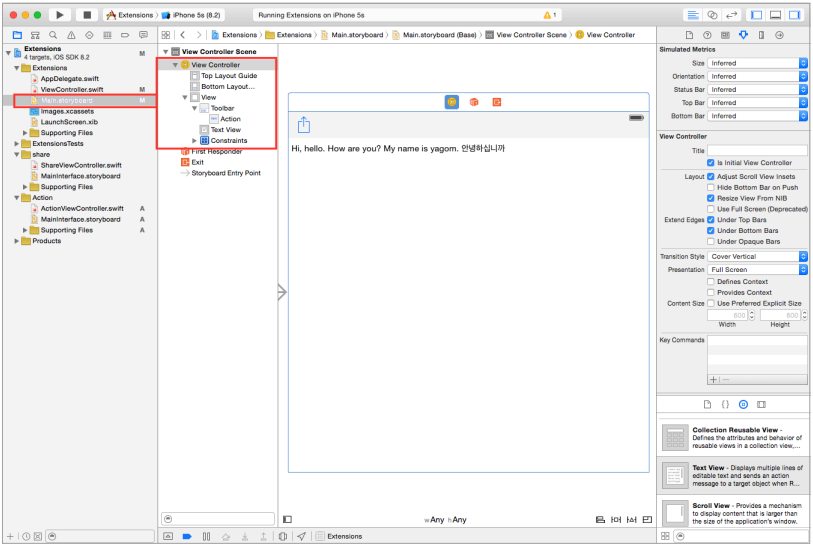
Extensions의 ViewController.swift 파일을 열고 다음과 같이 코드를 작성합니다.

[코드 1-3] ViewController.swift 파일의 전체 코드

```
import UIKit

class ViewController: UIViewController {

    // 스토리보드의 텍스트뷰와 연결된 IBOutlet
    @IBOutlet weak var textView: UITextView!

    // 스토리보드의 액션 버튼과 연결된 IBAction
    @IBAction func clickActionButton(sender: UIBarButtonItem) {

        // 텍스트뷰의 텍스트를 아이템으로 하는 액티비티 컨트롤러 생성
        let activityViewController = UIActivityViewController(activityItems:
[self.textView.text], applicationActivities: nil)

        // 액티비티 컨트롤러를 화면에 보여준다
        self.presentViewController(activityViewController, animated: true,
completion: nil)
    }
}
```

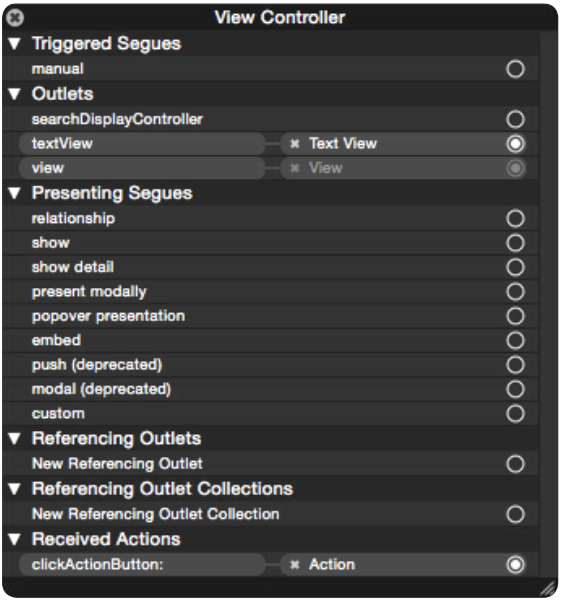
코드 작성을 마치고 Extension의 Main.storyboard 파일을 열어 툴 바를 얹고 액션 바 버튼 아이템도 얹어줍니다. 그 아래에는 텍스트뷰를 얹은 후 읽고 싶은 텍스트를 삽입합니다.

그림 1-22 ViewController의 UI 구성



텍스트뷰는 IBOutlet으로 연결하고 바 버튼 아이템은 IBAction으로 연결합니다.

그림 1-23 ViewController의 IBOutlet과 IBAction연결 확인

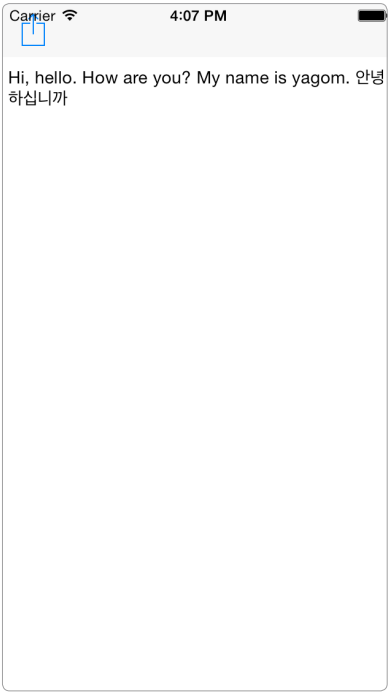


이제 실행 Scheme을 Extensions로 변경하고 애플리케이션을 실행해 보겠습니다.

그림 1-24 실행 Scheme 설정

왼쪽 상단의 액션 버튼을 선택합니다.

그림 1-25 Extensions 애플리케이션 실행 화면



액티비티 컨트롤러가 화면에 보이면 Action 메뉴를 선택하여 앞에서 구현한 Action 익스텐션 기능을 호출합니다. 익스텐션 화면이 실행되고 텍스트를 읽어주기 시작합니다. 소리가 나오지 않는다면 볼륨을 키워 보세요.

그림 1-26 액티비티 컨트롤러에서 Action 기능 호출

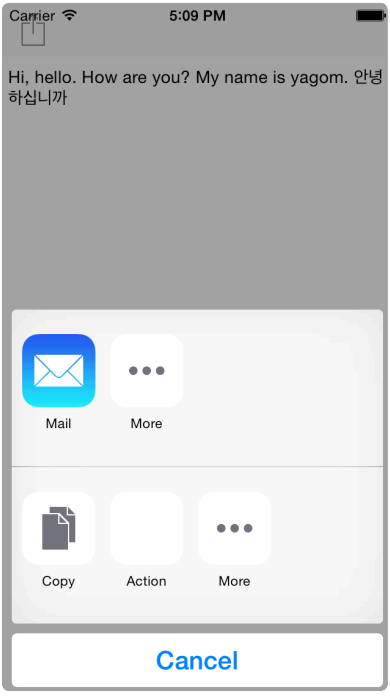
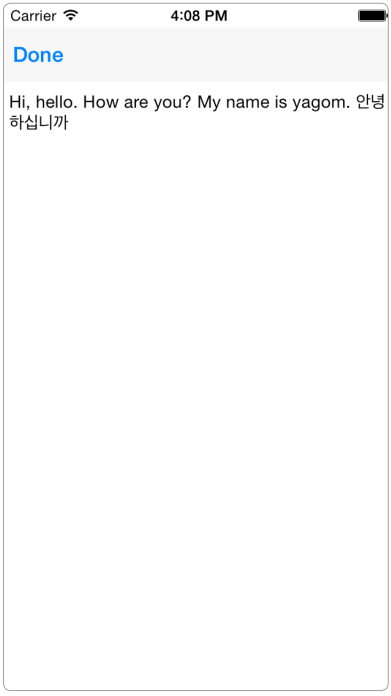


그림 1-27 Action 익스텐션이 실행된 화면



텍스트를 활용한 액션 외에도 다른 데이터 형식도 다룰 수 있으니 다른 아이디어도 적용해 보는 것은 어떨까요? :)

1.4 사진 편집

사진 편집 기능은 iOS에서 기본으로 제공하지만 이를 응용하면 자신만의 특별한 필터를 만들 수 있습니다. 사진 편집(Photo Editing) 익스텐션은 사진첩 또는 카메라 애플리케이션의 편집 모드에서 직접 구현한 필터로 필터링할 수 있는 기능으로, 사용자는 사진 애플리케이션이나 카메라 애플리케이션에서 지정한 특수효과를 손쉽게 사용할 수 있습니다.