SocialNetwork Tutorial

Based on Chapter 11 (pages 745 - 780) from iOS 6 by Tutorials; refer to that chapter for a more detailed explanation of the steps involved in this tutorial. The steps in this tutorial start at Page 762.

Getting Started: Use the project in the *SocialNetworkTutorial* folder.

Part 1

Objective: Display a UIActivityController.

1. Add the following code to the *actionTapped:* selector of **SocialViewController.m**.

```
- (IBAction)actionTapped
{
    NSString *initalTextString = [NSString stringWithFormat:@"Fun Fact: %@", self.factTextView.text];
    UIActivityViewController *activityViewController = [[UIActivityViewController alloc] initWithActivityItems:@[initalTextString] applicationActivities:nil];
    [self presentViewController:activityViewController animated:YES completion:nil];
}
```

   What this does: This creates a **UIActivityViewController**, sets the text it will display (the post), and then presents the view controller. Note that it displays several different optional activities, which are device and settings dependent.

2. **RUN THE APP.** Pressing the action button should present the view controller that was created in the above step.

3. Add a property that is a boolean called *deviceWasShaken* to the **SocialViewController** class. For a hint, see the line of code below.

```
@property (assign, nonatomic) BOOL deviceWasShaken;
```

4. In the *motionEnded:withEvent:* selector, modify the boolean property that was created in the previous step.

```
if (motion == UIEventSubtypeMotionShake) {
    self.deviceWasShaken = YES;
    //...
}
```

5. Modify the code in the *actionTapped* selector to check to see if the deviceWasShaken or not.

```
- (IBAction)actionTapped {
    if (self.deviceWasShaken) {
        //... Code that was already here.
    } else {
        UIAlertView *alertView = [[UIAlertView alloc]
            initWithTitle:@"Shake"
            message:@"Before you can share, please shake the device
in order to get a random Fun Fact"
            delegate:nil
            cancelButtonTitle:@"Dismiss" otherButtonTitles:nil];
        [alertView show];
    }
}
```

6. **RUN THE APP.** Note that a *shake gesture* can be sent to the simulator from the **Hardware** menu or by pressing ^⌘Z (CONTROL+COMMAND+Z). If you press the action button before sending the shake gesture, the **UIAlertView** will appear; if the shake gesture has been sent, the activity view controller will appear.

7. Add the **Social.Framework** to the project, and make sure you import <*Social.Social.h*> where required. Please see the instructions starting at the bottom of page 765 for more details.

8. Add the following enumerator declaration to **SocialViewController.m**, just below the *#define* lines.

```
typedef enum SocialButtonTags {
    SocialButtonTagFacebook,
    SocialButtonTagSinaWeibo,
    SocialButtonTagTwitter
} SocialButtonTags;
```

What this does: This creates list of constants that belong to a named group. This is almost the same as writing:

```
SocialButtonTagFacebook = 0;
SocialButtonTagSinaWeibo = 1;
SocialButtonTagTwitter = 2;
```

9. Add the following code to the *socialTapped:* selector in **SocialViewController.m**. Note that the sender "tag" was already set in the project when it was opened, and can be found in the **MainStoryBoard.storyboard** file.

```
- (IBAction)socialTapped:(id)sender {
    if (self.deviceWasShaken){
        NSString *serviceType = @"";
        switch (((UIButton *)sender).tag) {
            case SocialButtonTagFacebook:
                serviceType = SLServiceTypeFacebook;
                break;
            case SocialButtonTagSinaWeibo:
                serviceType = SLServiceTypeSinaWeibo;
                break;
            case SocialButtonTagTwitter:
                serviceType = SLServiceTypeTwitter;
                break;
            default:
```

```
                    break;
            }
            if (![SLComposeViewController
isAvailableForServiceType:serviceType]) {
                    [self showUnavailableAlertForServiceType:serviceType];
            } else {
                    SLComposeViewController *composeViewController =
[SLComposeViewController
composeViewControllerForServiceType:serviceType];
                    [composeViewController
addImage:self.authorImageView.image];
                    NSString *initalTextString = [NSString
stringWithFormat:@"Fun Fact: %@", self.factTextView.text];
                    [composeViewController setInitialText:initalTextString];
                    [self presentViewController:composeViewController
animated:YES completion:nil];
            }
      } else {
            UIAlertView *alertView = [[UIAlertView alloc]
                    initWithTitle:@"Shake"
                    message:@"Before you can share, please shake the
device in order to get a random Fun Fact"
                    delegate:nil
                    cancelButtonTitle:@"Dismiss"
                    otherButtonTitles:nil];
            [alertView show];
      }
}
```

What this does: This determines which button (*sender*) was pressed by comparing it's "tag" against a predefined list (`SocialButtonTags`). It then uses that variable to initialize the **SLComposeViewController** serviceType. It then presents that view controller.

10. The selector *showUnavailableAlertForServiceType:* selector if that type of activity is not available on that device (for example, a Facebook account may not be setup on the device being used). This means we'll need to add that selector to **SocialViewController.m**.

```
-(void)showUnavailableAlertForServiceType: (NSString *)serviceType {
      NSString *serviceName = @"";
      if (serviceType == SLServiceTypeFacebook) {
            serviceName = @"Facebook";
      } else if (serviceType == SLServiceTypeSinaWeibo) {
            serviceName = @"Sina Weibo";
      } else if (serviceType == SLServiceTypeTwitter) {
            serviceName = @"Twitter";
      }
      UIAlertView *alertView = [[UIAlertView alloc]
initWithTitle:@"Account" message:[NSString stringWithFormat:@"Please go
to the device settings and add a %@ account in order to share through
that service", serviceName]  delegate:nil cancelButtonTitle:@"Dismiss"
otherButtonTitles:nil];
      [alertView show];
}
```

What this does: This simply shows an alert view with a message specific to the requested service.

11. **RUN THE APP.** Pressing one of the buttons for a specific social network will now display a SLComposeViewController with the appropriate service already initialized.