HANSER

Leseprobe

Jan Tittel, Jochen Baumann

Apps für iOS entwickeln

Am Beispiel einer realen App

ISBN (Buch): 978-3-446-43192-8

ISBN (E-Book): 978-3-446-43314-4

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-43192-8

sowie im Buchhandel.

5 Eigene Klassen und Subklassen erstellen

In Kapitel 4 haben wir mit der Entwicklung einer umfangreichen Beispiel-App begonnen und uns dabei den folgenden Themen gewidmet:

- Benutzeroberflächen unter Verwendung des Storyboards erstellen
- Zwischen mehreren Ansichten einer App mithilfe von Übergängen navigieren

Die Beispieldateien zu diesem Kapitel finden Sie unter *www.downloads.hanser.de* im Unterordner *scytenotes 0.2*.

In diesem Kapitel werden wir die App weiter ausbauen, wobei die folgenden Themen der App-Entwicklung für iOS berücksichtigt werden:

- Erstellen von eigenen Klassen
- Erstellen von eigenen Subklassen für View Controller
- Erstellen von Unwind Segues und Codes zum Verlassen von Views

5.1 Eigene Klassen erstellen

Die Beispiel-App soll, wie bereits bekannt, Notizen in Form von Text, Bild und Audio verwalten können. Hierzu erstellen wir zunächst eine eigene allgemeine Klasse für Notizen-Objekte:

1. Wählen Sie aus dem Untermenü *New* des Menüs *File* den Befehl File aus.
2. Es öffnet sich das Fenster wie in Bild 5.1.

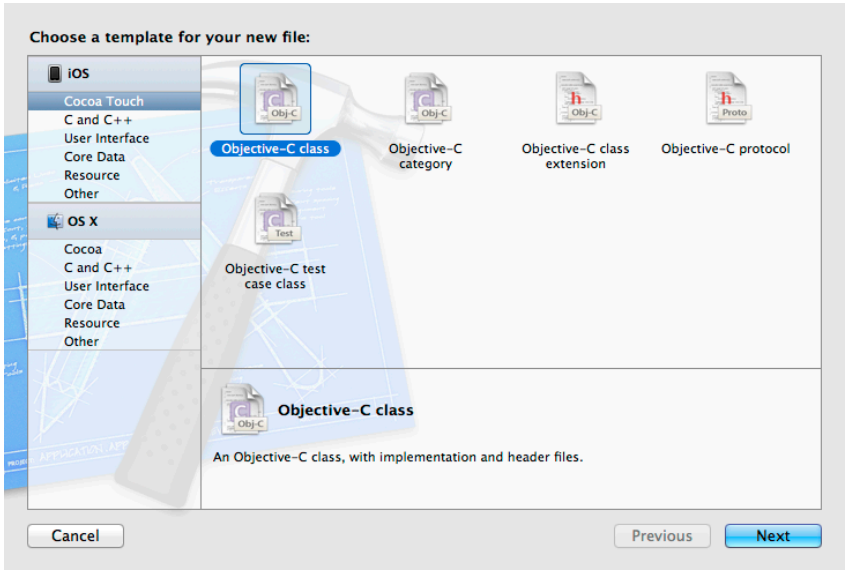

Bild 5.1 Erstellen einer neuen Objective-C-Klasse

3. Wählen Sie aus dem Bereich *iOS* den Punkt *Cocoa Touch* aus und dann die Vorlage *Objective-C class*.

In iOS sind die einzelnen Bibliotheken in den Frameworks von *Cocoa Touch* gekapselt, da Objective-C selbst keine Funktionen bereitstellt, um beispielsweise auf den Kalender zuzugreifen. Diese Funktionen stammen aus *Cocoa Touch*.

4. Bestätigen Sie die Auswahl mit NEXT, worauf das Fenster wie in Bild 5.2 angezeigt wird.

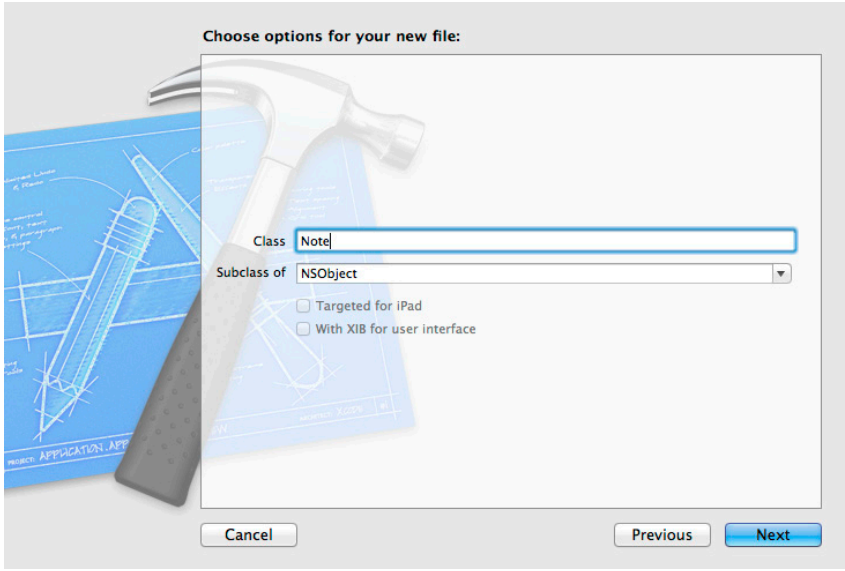

Bild 5.2 Name der Klasse und Superklasse angeben

5. Geben Sie in das Feld *Class* den Namen der Klasse ein. Dies ist in unserem Fall der Name *Note*. Losgelöste eigene Klassen werden in der Regel vom *NSObject* abgeleitet, sodass dies auch die korrekte Auswahl für die *Subclass of* ist.
6. Bestätigen Sie die Auswahl mit NEXT.
7. Geben Sie im letzten Fenster den Speicherort der neuen Klasse an. Dies ist meist das Verzeichnis des Projekts, wie in Bild 5.3 zu sehen.

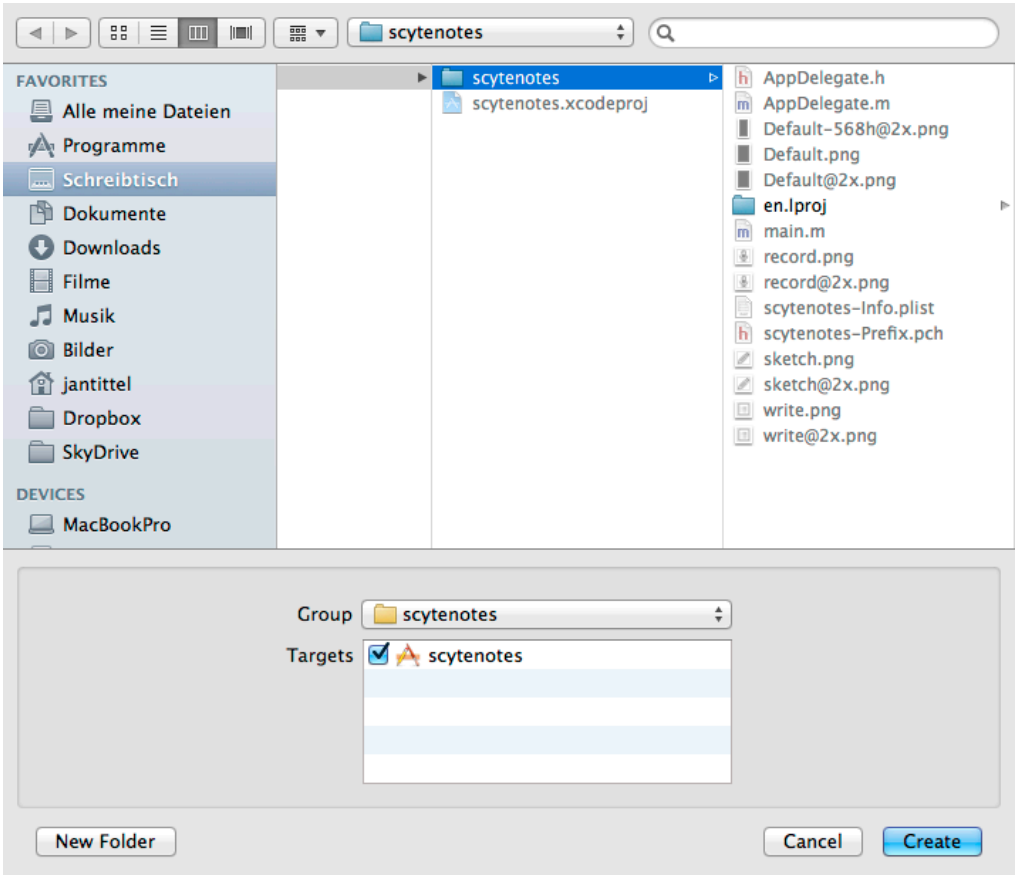

Bild 5.3 Speicherort und Projektzugehörigkeit einer Klasse angeben

8. Erstellen Sie die neue Klasse mit CREATE.

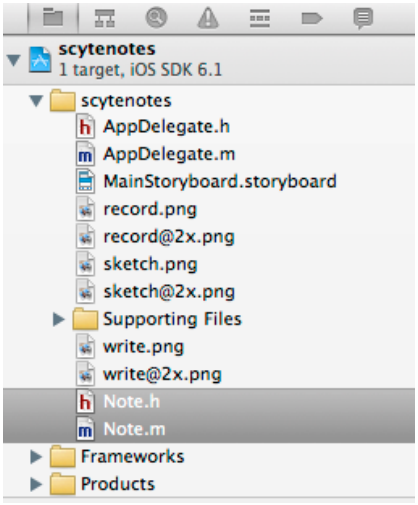

Bild 5.4
Dateien einer neuen Klasse im Project Navigator

Im *Project Navigator* in Bild 5.4 sehen Sie, dass das Projekt nun über die beiden neuen Dateien *Note.h* und *Note.m* verfügt. Fügen Sie den Dateien jetzt den benötigten Code wie folgt hinzu:

1. Öffnen Sie die Datei *Note.h* im Editor und fügen Sie den Code aus Listing 5.1 hinzu.
2. Öffnen Sie anschließend die Datei *Note.m* im Editor und fügen Sie den Code aus Listing 5.2 hinzu.

Listing 5.1 Schnittstelle Note.h der Klasse Note

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

@interface Note : NSObject

@property (strong, nonatomic) NSString *filePath;
@property (strong, nonatomic) NSDate *fileDate;
@property (strong, nonatomic) NSString *subject;

- (id)initWithFilePath:(NSString *)filePathNew fileDate:(NSDate
*)fileDateNew subject:(NSString *)subjectNew;

@end
```

Listing 5.2 Implementierung von Note.m der Klasse Note

```
#import "Note.h"

@implementation Note

@synthesize filePath;
@synthesize fileDate;
@synthesize subject;

- (id)initWithFilePath:(NSString *)filePathNew fileDate:(NSDate
*)fileDateNew subject:(NSString *)subjectNew
{
```

```
    self = [super init];
    if(self)
    {
        self.filePath = filePathNew;
        self.fileDate = fileDateNew;
        self.subject = subjectNew;
    }
    return self;
}

@end
```

Die Klasse *Note* verfügt lediglich über die drei Eigenschaften *filePath* vom Typ *NSString*, *fileDate* vom Typ *NSDate* und *subject* vom Typ *NSString*. Des Weiteren wurde eine Initialisierungsmethode implementiert, mit der diese drei Eigenschaften bei der Erstellung eines neuen Objekts gleich initialisiert werden können.

Wie Sie sehen, ist die Klasse *Note* sehr simpel aufgebaut, da sie allgemein gehalten ist, um allen Notizen in Form von Text, Bild und Audio gerecht zu werden. Darüber hinaus werden die Notizen später als Dateien abgelegt, sodass sich der Typ der Notiz aus der Dateiendung ergeben wird. Konkretere Informationen sind innerhalb der Klasse daher nicht notwendig und auch nicht beabsichtigt.

Das Datum der Datei ist stets aus den Dateieigenschaften auslesbar sowie auch der Betreff der Notiz, welcher dem Dateinamen ohne Dateiendung entsprechen wird. Diese beiden Eigenschaften sind in der Klasse lediglich vorhanden, damit die Informationen beim Start einmalig ausgelesen und festgehalten werden können und kein laufender Zugriff auf das Dateisystem notwendig ist.

Da es jede Datei in einem Dateisystem unter Einbeziehung des Dateipfades nur einmal geben kann, handelt es sich bei der Eigenschaft *filePath* gleichzeitig um einen eindeutigen Wert, über den später auch die Zuordnungen für Erinnerungen in einer Datenbank vorgenommen werden.

5.2 Eigene Subklassen erstellen

In Kapitel 4 haben wir bereits das Storyboard, bestehend aus mehreren View Controllern, erstellt. Damit einem View Controller eine eigene Logik hinzugefügt werden kann, muss diesem eine eigene Klasse zugeordnet werden, die von der jeweils entsprechenden Klasse aus dem Framework abgeleitet wird.

5.2.1 Subklassen für View Controller erstellen

Gehen Sie wie folgt vor, um eine eigene Klasse für den View Controller, der später die Copyright-Infos enthalten soll, zu erstellen:

1. Wählen Sie aus dem Untermenü *New* des Menüs *File* den Befehl FILE aus.

2. Wählen Sie aus dem Bereich *iOS* den Punkt *Cocoa Touch* aus und dann die Vorlage *Objective-C class*.
3. Bestätigen Sie die Auswahl mit NEXT.
4. Geben Sie in das Feld *Class* den Namen der Klasse ein. Dies ist in unserem Fall der Name *AboutViewController*. Für *Subclass of* geben Sie die allgemeine übergeordnete Klasse *UIViewController* an oder wählen den entsprechenden Eintrag aus der Liste aus. Ihre Angaben sollten denen aus Bild 5.5 entsprechen.

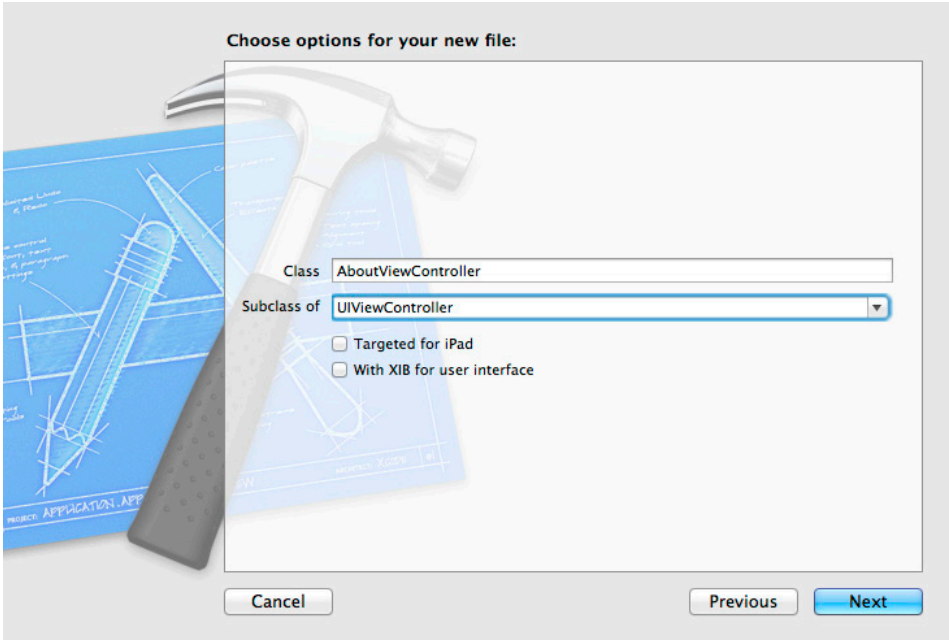

Bild 5.5 Eigene Subklasse für einen View Controller erstellen

5. Bestätigen Sie die Auswahl mit NEXT.
6. Geben Sie im letzten Fenster den Speicherort der neuen Klasse an, und erstellen Sie die neue Klasse mit CREATE.

Wir werden im weiteren Verlauf eigene Klassen für alle View Controller im Storyboard benötigen, außer für die Navigation Controller. Erstellen Sie auf die nun bekannte Weise die folgenden Klassen, deren Superklasse für die Angabe *Subclass of* stets in Klammern mit angegeben ist:

- *WriteMasterViewController (UITableViewController)*
- *SketchMasterViewController (UITableViewController)*
- *RecordMasterViewController (UITableViewController)*
- *WriteDetailViewController (UIViewController)*
- *SketchDetailViewController (UIViewController)*
- *RecordDetailViewController (UIViewController)*
- *ScyteNotesTabBarController (UITabBarController)*

Wenn Sie sich die Implementierung der generierten Klassen ansehen, sehen Sie, dass in den vom *UIViewController* und *UITabBarController* abgeleiteten Klassen jeweils eine *init*-Methode, die Methode *viewDidLoad* und die Methode *didReceiveMemoryWarning*, vorhanden sind. In den vom *UITableViewController* abgeleiteten Klassen sind zusätzlich einige weitere Methoden enthalten, über die eine *Table View* gesteuert wird.

5.2.2 Subklasse einem View Controller zuordnen

Nachdem Sie alle benötigten Klassen erstellt haben, ordnen Sie diese nun, wie nachfolgend am Beispiel der Klasse *AboutViewController* beschrieben, den View Controllern aus dem Storyboard zu:

1. Markieren Sie im Storyboard oben links den einzelnen *View Controller*, der nicht mit einer *Table View* verbunden ist, wie in Bild 5.6 zu sehen.

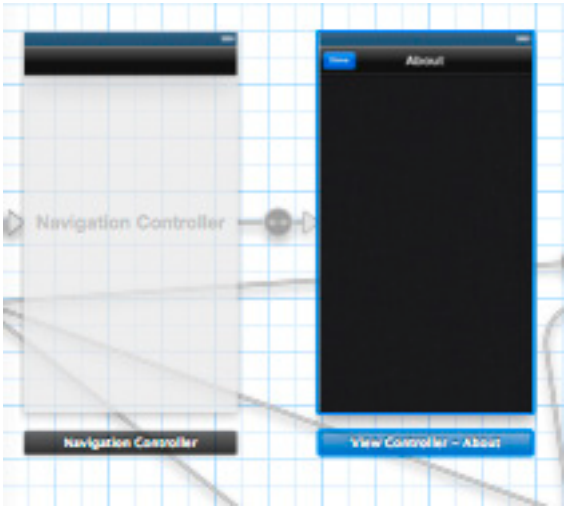

Bild 5.6
View Controller im Storyboard korrekt auswählen

2. Achten Sie darauf, dass Sie tatsächlich den gesamten *View Controller* markiert haben und nicht nur die darin enthaltene *View*. Arbeiten Sie gegebenenfalls mit der *Document Outline*.
3. Öffnen Sie nun in den Utilities den *Identity inspector*, wie in Bild 5.7 dargestellt.

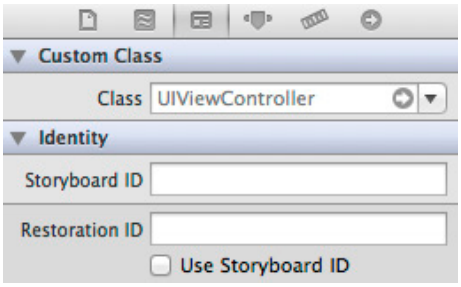

Bild 5.7
Identity inspector ohne zugewiesene eigene Klasse

4. Wählen Sie unter *Custom Class* aus der Drop-&-Down-Liste den Eintrag *AboutViewController* aus, oder geben Sie den Wert von Hand ein. Der *Identity inspector* sollte nun wie in Bild 5.8 aussehen.

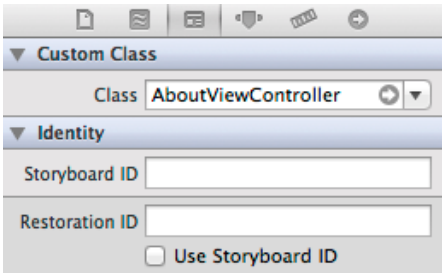

Bild 5.8
Identity inspector mit zugewiesener eigener Klasse

5. Sofern das Storyboard nicht in der Originalansichtsgröße angezeigt wird oder der *View Controller* nicht aktuell ausgewählt ist, wird nun, wie in Bild 5.9 zu sehen, auch der Name der Klasse unterhalb des *View Controllers* angezeigt, wodurch die *View Controller* im Storyboard nun auch eindeutig voneinander zu unterscheiden sind.

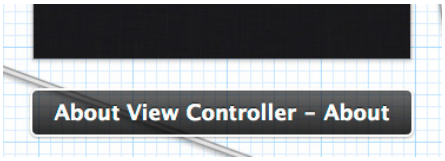

Bild 5.9
Name der Klasse im Storyboard

Ordnen Sie die zuvor erstellten Subklassen jetzt den übrigen *View Controllern* zu. Entsprechend dem Namen in der *Tab-Bar* des jeweiligen *Navigation Controllers* werden den damit verbundenen *Table Views* die Master-Klassen zugewiesen. Den mit den *Table Views* verbundenen *View Controllern* werden die Detail-Klassen zugewiesen. Ein Beispiel sehen Sie in Bild 5.10. Die einzige Tab-Bar-Controller-Klasse wird selbstverständlich dem *Tab Bar Controller* zugewiesen.

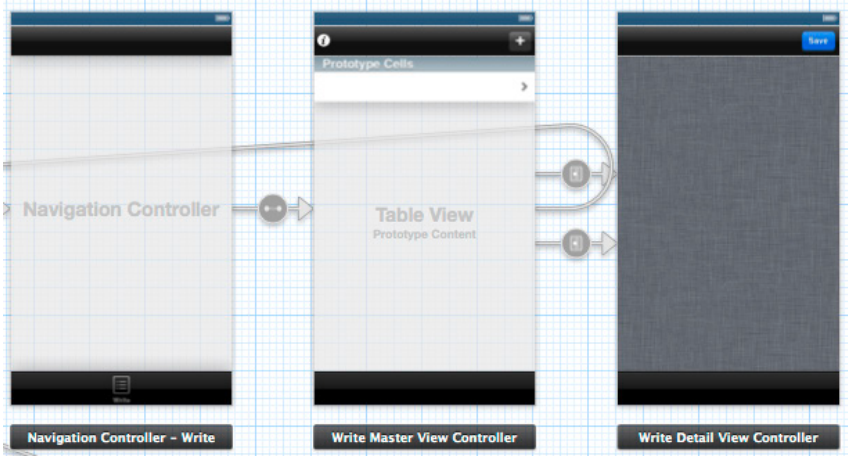

Bild 5.10 Zuordnung der Master- und Detail-Klassen im Storyboard

5.3 Aus Views zum Ursprung zurückkehren

Wie bereits im vorherigen Kapitel erwähnt, mussten wir den View Controllern erst eigene Klassen zuordnen, damit wir eine Möglichkeit zum Hinzufügen des benötigten Codes haben, um die View wieder zu schließen und zur vorherigen View zurückkehren zu können. Nachdem diese Klassen jetzt vorhanden sind, kann der entsprechende Code hinzugefügt werden, um die Navigation zu vervollständigen.

5.3.1 Unwind Segues erstellen

Mit den *Unwind Segues* steht seit iOS 6 eine Möglichkeit zur Verfügung, um, wie im Fall unserer Beispiel-App, komfortabel aus einer *Detail View* heraus dieselbe zu schließen und dabei eine Methode in der Klasse der *Master View* auszuführen. Dieses Verhalten implementieren wir wie folgt:

1. Fügen Sie der Schnittstelle *WriteMasterViewController.h* die Definition der Methode *saveWrite:* wie in Listing 5.3 hinzu.

Listing 5.3 Schnittstelle WriteMasterViewController.h

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

@interface WriteMasterViewController : UITableViewController

- (IBAction)saveWrite:(UIStoryboardSegue *)segue;

@end
```

2. Wiederholen Sie den vorherigen Schritt für die Schnittstellen *SketchMasterViewController.h* und *RecordMasterViewController.h* und nennen Sie die Methoden entsprechend *saveSketch:* und *saveRecord:*
3. Fügen Sie der Implementierung *WriteMasterViewController.m* die Implementierung der zu diesem Zeitpunkt noch leeren Methode *saveWrite:* am Ende der Datei vor dem `@end` wie in Listing 5.4 hinzu.

Listing 5.4 Implementierung der Methode saveWrite:

```
- (IBAction)saveWrite:(UIStoryboardSegue *)segue
{
    //Anweisungen folgen später...
}

@end
```

4. Wiederholen Sie den vorherigen Schritt sinngemäß für die Implementierung der Methoden in den Klassen *SketchMasterViewController* und *RecordMasterViewController*.
5. Erstellen Sie im Storyboard für den *Write Detail View Controller* einen *Unwind Segue*, indem Sie die Maus bei gedrückt gehaltener *ctrl*-Taste von dem *Save*-Button aus ziehen

und auf dem grünen *Exit*-Symbol in der unteren Leiste wieder loslassen, wie in Bild 5.11 dargestellt.

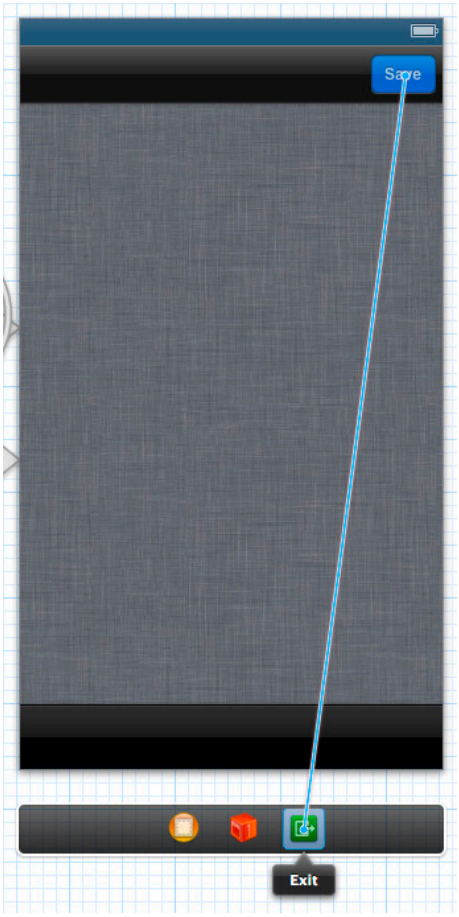

Bild 5.11
Unwind Segue im Storyboard erstellen

6. Wählen Sie in dem sich öffnenden Fenster, wie in Bild 5.12, die Methode *saveWrite:* aus.

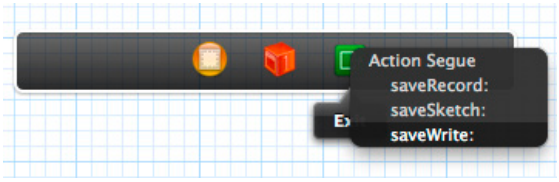

Bild 5.12
Methode für Unwind Segue auswählen

7. Wiederholen Sie den vorherigen Schritt sinngemäß für den *Sketch Detail View Controller* und den *Record Detail View Controller*.

Wenn Sie die App nun im Simulator testen und über das Plus-Symbol eine der Detailansichten aufrufen, können Sie diese nun auch über den *Save*-Button wie vorgesehen schließen.

5.3.2 Eine View per Code schließen

Den *AboutViewController* schließen wir nicht mit einem *Unwind Segue*. Hierfür implementieren wir stattdessen eine klassische Close-Methode wie folgt:

1. Nehmen Sie die folgende Methodendefinition in die Datei *AboutViewController.h* auf.

```
- (IBAction)done:(id)sender;
```

2. Fügen Sie der Datei *AboutViewController.m* die entsprechende Implementierung aus Listing 5.5 hinzu. Mit dem Aufruf der Methode *dismissViewControllerAnimated:completion:* wird die aktuelle View (*self*) unter Berücksichtigung einer zuvor definierten Animation wieder geschlossen.

Listing 5.5 Implementierung der Methode done:

```
- (IBAction)done:(id)sender
{
    [self dismissViewControllerAnimated:YES completion:nil];
}

@end
```

3. Wählen Sie den *AboutViewController* im Storyboard aus und aktivieren Sie den *Connections inspector*.
4. Verbinden Sie die Methode *done:* im *Connections inspector* mit dem *Done*-Button im *AboutViewController* per Drag & Drop, wie in Bild 5.13 zu sehen.

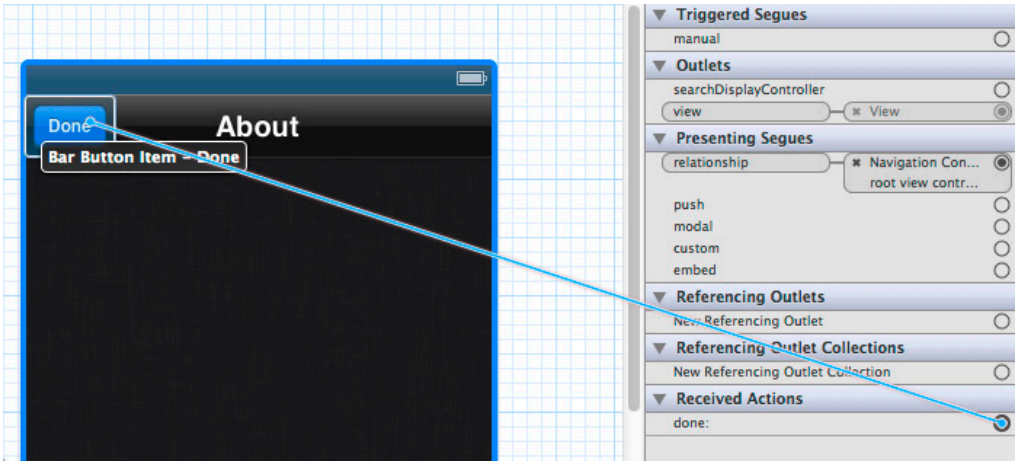

Bild 5.13 Methode mit Button im Storyboard verbinden

Damit lässt sich nun auch der *AboutViewController* schließen, womit die Navigation vollständig implementiert ist.

5.4 Zwischenstand der App (Version 0.2)

In diesem Kapitel haben wir die Klasse *Note* für Notizen erstellt und den View Controllern eigene Klassen hinzugefügt. Damit war es möglich, die noch fehlende Implementierung für die Navigation nachzuholen.

In den folgenden Kapiteln wird weniger mit dem Storyboard beziehungsweise dem Interface Builder gearbeitet, da ein wesentlicher Teil der grafischen Entwicklung hiermit abgeschlossen ist. Gleichzeitig haben wir durch die eigenen Klassen die notwendigen Vorbereitungen getroffen, damit die App mit umfangreicher Logik bestückt werden kann.