



Agenda Tag 2

09:00 → Einführung in iOS-Entwicklung: Delegation
iOS Schichten

Aufgabe: Erstellen einer TableView

12:00 Pause

13:00 Cocoa in a Nutshell

Aufgabe: Verwenden des NavigationControllers

17:00 Ende

- ◆ Protokolle definieren Schnittstellen, die von Klassen implementiert werden können
 - Entspricht Interfaces in Java
 - Methoden können optional sein

```
@protocol Entflammbar  
-(void)anzuenden;  
  
@optional  
-(BOOL)loeschen;  
  
@end
```

```
@interface Streichholz : NSObject<Entflammbar>  
...  
@end
```

- ◆ Muster, um Aufgaben an andere Objekte zu delegieren
- ◆ Wird an vielen Stellen im iOS eingesetzt
- ◆ Erlaubt die Integration von individuellem Code in Framework-Klassen, ohne Vererbung einzusetzen
- ◆ Wird über Protokolle realisiert
 - `@interface RSSParser : NSObject<NSXMLParserDelegate>`
- ◆ Beispiele:
 - UITableViewDelegate
 - UITableViewDataSource
 - NSXMLParserDelegate

Delegation: Verwendung

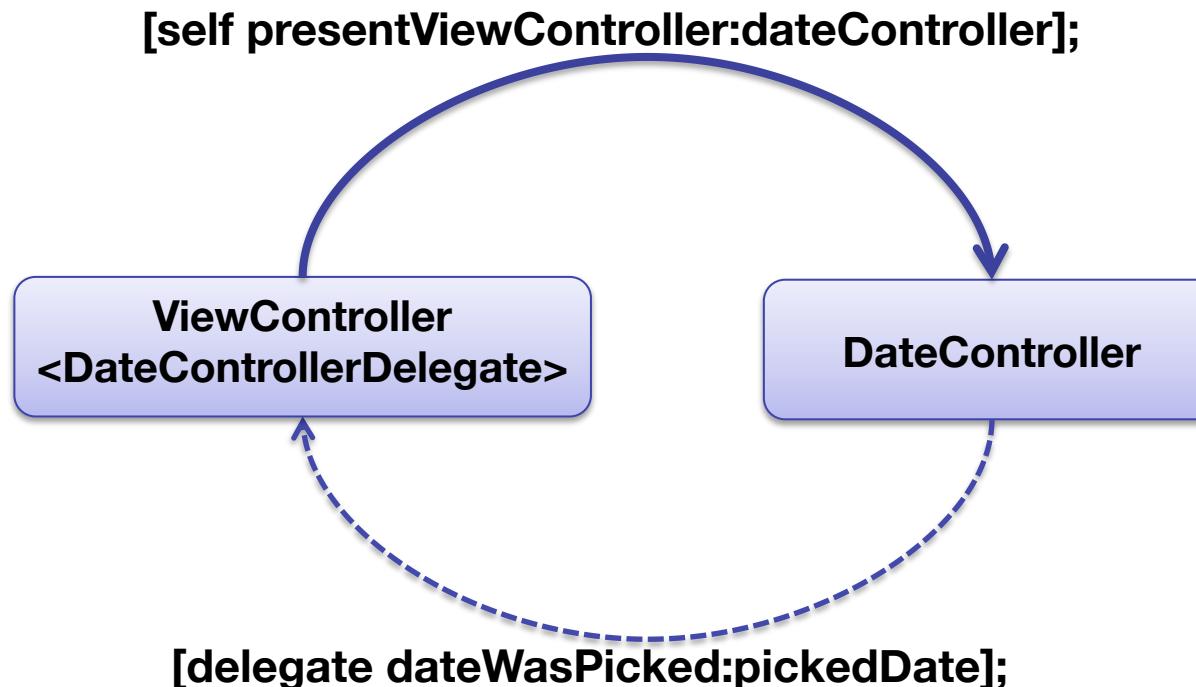
- ◆ Aufteilung von Funktionalität zwischen Klassen
- ◆ Vorgefertigte Framework-Funktionen können mit eigenen Inhalten angereichert werden
- ◆ Beben von komplexen Framework-Klassen ist nicht nötig
- ◆ Lediglich das Protokoll muss erfüllt werden
 - Viele Methoden in Protokollen sind optional
→ Fokussierung auf die benötigten Funktionen
- ◆ Oft implementieren die Controller selbst die benötigten Protokolle
 - Zum Beispiel

```
@interface UITableViewController : UIViewController
<UITableViewDelegate, UITableViewDataSource>
```

Beispiel:

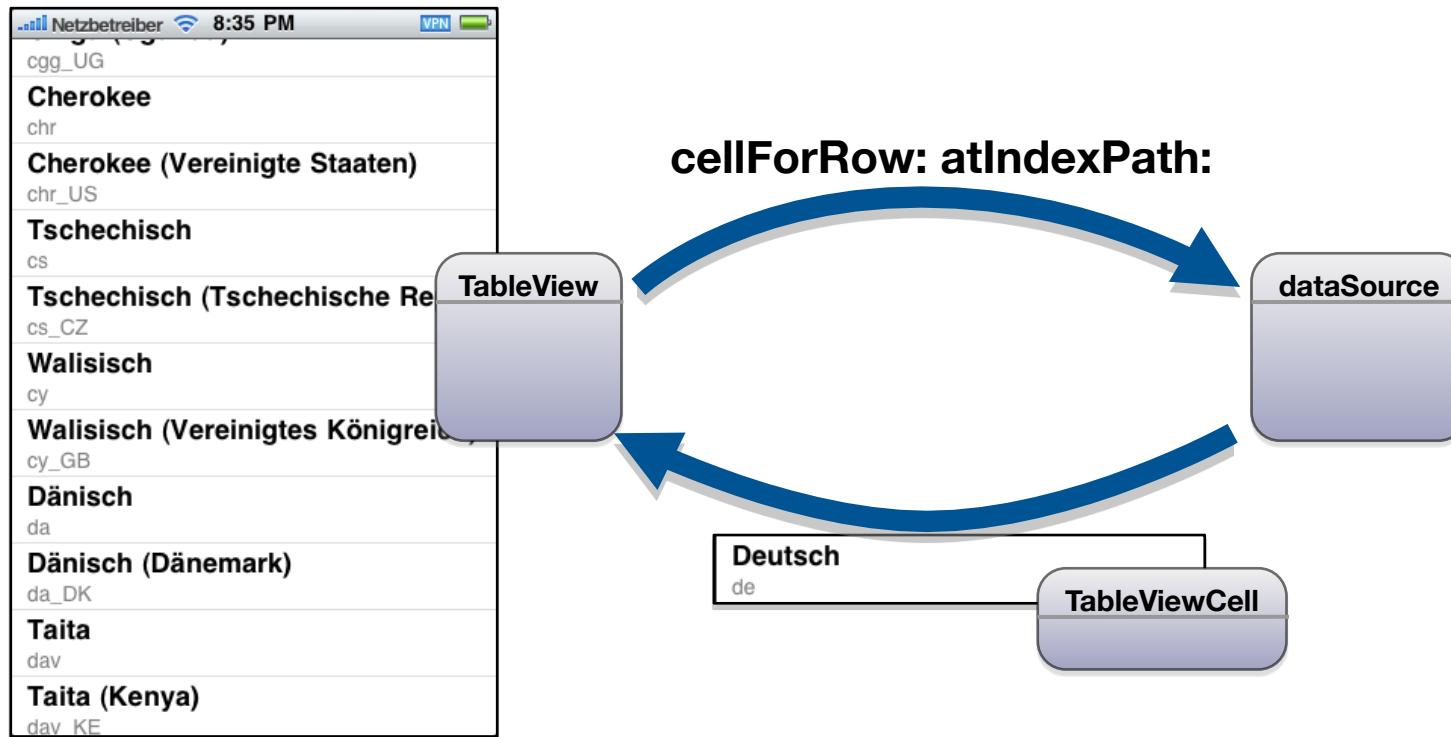
- ◆ DateController hat ein Delegate, dass das DateControllerDelegate-Protokoll implementiert

```
// DateController.h:  
@property (nonatomic, weak) id<DateControllerDelegate> delegate;
```



Beispiel: UITableView

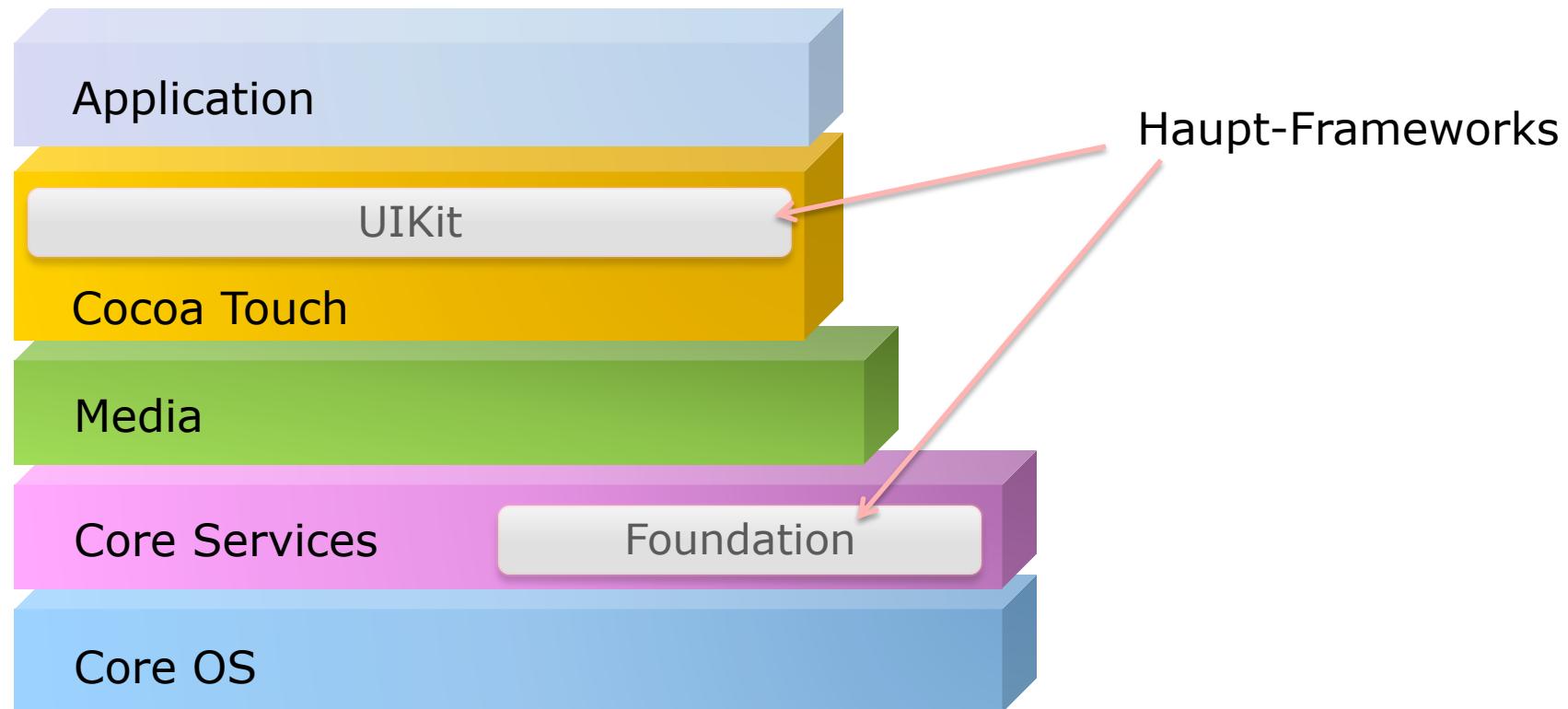
- ◆ UITableViews haben zwei Delegates:
 - ◆ delegate, um auf Aktionen des Benutzers zu reagieren
 - ◆ dataSource, um die darzustellenden TableViewCells zu erhalten





iOS Schichten

- ◆ Vieles schon dabei
 - Look and Feel
 - Bedienkonzepte
 - Animationen
 - Rotation
- ◆ Aufbau der UI mit MVC:
 - UIViews zur Darstellung
 - UIViewController zur Verknüpfung von Modell und View
 - Modell-Klassen vom Entwickler
 - Pro Bildschirmseite ein ViewController



Core OS

- ◆ Kernel
- ◆ Dateisystem
- ◆ Netzwerk
- ◆ Security
- ◆ Treiber

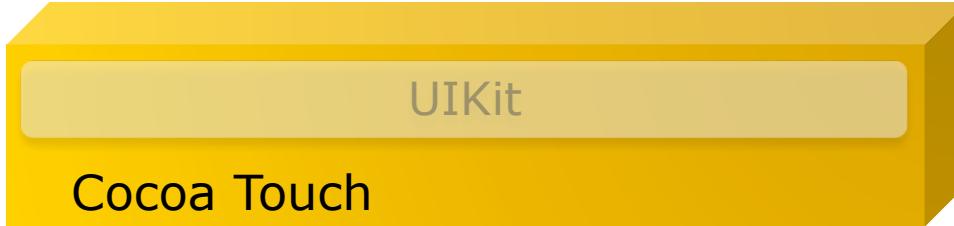
Core Services

Foundation

- ◆ Strings
- ◆ Collections
- ◆ Kontakte
- ◆ Zugriff auf: GPS, Kompass, Beschleunigungssensor, Gyroskop

Media

- ◆ Core Graphics
- ◆ Core Animation
- ◆ OpenGL ES
- ◆ Audio
- ◆ Video



- ◆ Map Kit
- ◆ iAd

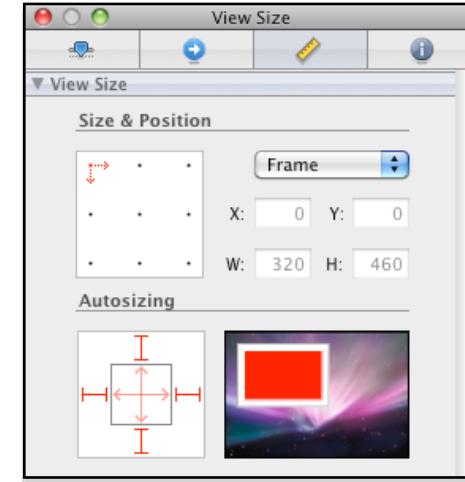


- ◆ Views
- ◆ ViewController
- ◆ Gestenbehandlung
- ◆ Interface Builder zum Erstellen von UIKit-Objekten

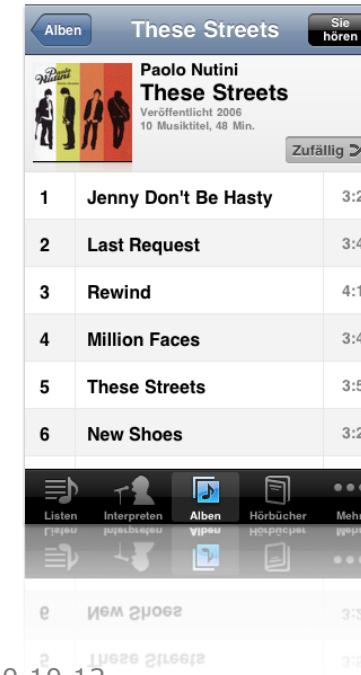
- ◆ Eine Methode im ViewController bestimmt, wie rotiert werden kann

```
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)orientation
{
    return YES;
}
```

- ◆ Views müssen sich an neue Größe anpassen:
 - Konfiguration im Interface Builder



- ◆ Vorgefertigte Controller für
 - Bedienführung in der App (Übersicht zu Details)
 - Tabs
 - Listen
 - Webseiten





UIKit: TableView

◆ Listen / Tabellen



UIKit: TableView

Styles:

plain



grouped

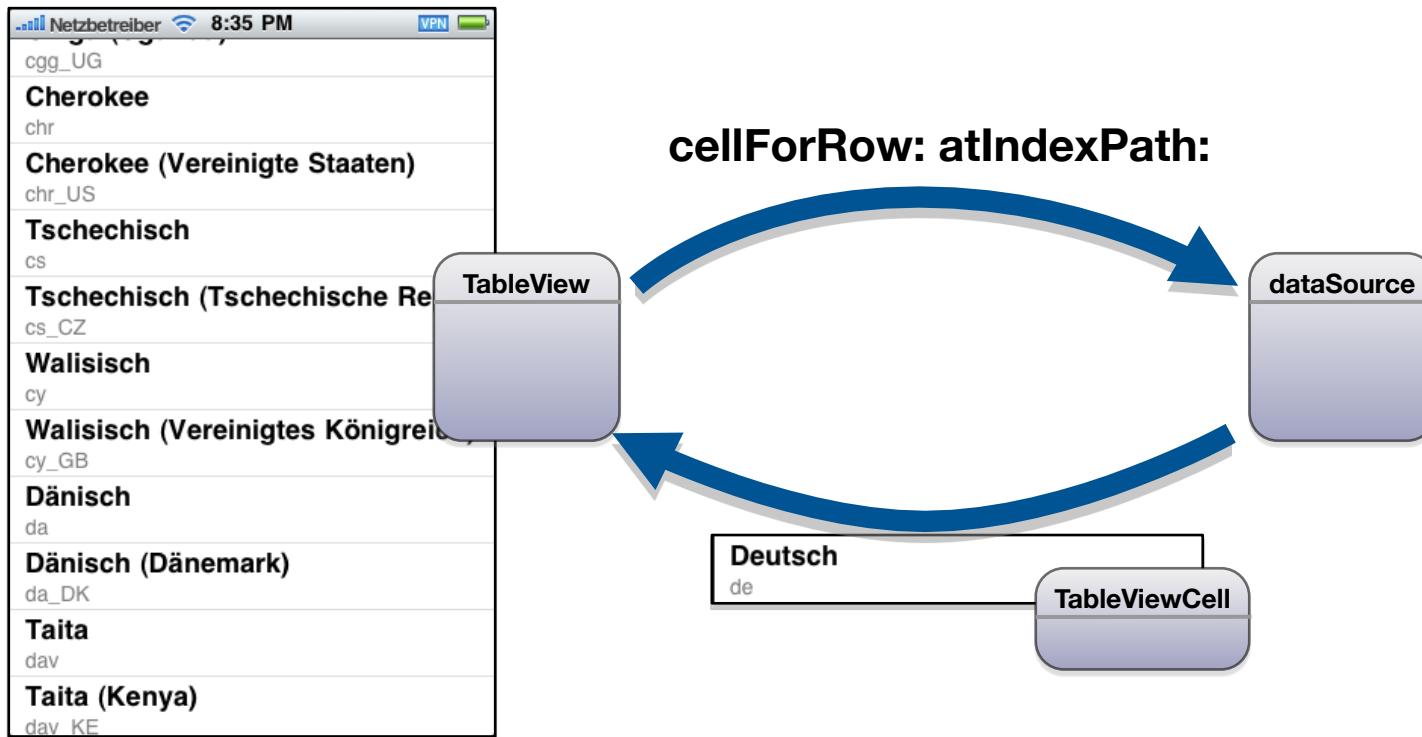


Zellen:



UIKit: UITableView

- **UITableViews haben zwei Delegates:**
 - *delegate*, um auf Aktionen des Benutzers zu reagieren
 - *dataSource*, um die darzustellenden TableViewCells zu erhalten



- **UITableViewController implementiert beide Delegate-Protokolle**

▪ **UITableViewDataSource**

```
// Liefert die Zellen für die einzelnen Tabellen-Zeilen
- (UITableViewCell*)tableView:(UITableView*)tableView
    cellForRowAtIndexPath:(NSIndexPath*)indexPath

// Anzahl der Zeilen in einem Tabellenabschnitt
- (NSInteger)tableView:(UITableView*)tableView numberOfRowsInSection:
    (NSInteger)section
```

▪ **UITableViewDelegate**

```
// Wird aufgerufen wenn eine Zeile angetippt wurde
- (void)tableView:(UITableView*)tableView
    didSelectRowAtIndexPath:(NSIndexPath*)indexPath
```



UIKit: TableView

- **UITableViewCell**
- Aus Performancegründen sollten Zellen wiederverwendet werden

```
// Gibt eine schon vorhandene Zelle für den identifier?  
- (UITableViewCell*)dequeueReusableCellWithIdentifier:(NSString*)identifier
```

- Beispiel:

```
// in Datasource-Methode tableView:cellForRow:atIndexPath:  
  
UITableViewCell* cell = [tableView dequeueReusableCellWithIdentifier:@"standard"];  
  
if(cell == nil) {  
  
    cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle  
                                reuseIdentifier:@"standard"];  
  
}  
  
...  
  
return cell;
```

◆ Touch-Erkennung

- Einzelne Finger
- Bewegung
- Loslassen

◆ Gesten

- Benachrichtigung für vordefinierte Gesten
- Tipp
- Wischen
- Verschieben
- Pinch (Zoomen)
- etc.



- ◆ Gesten können von View-Objekten erkannt werden
 - Viele Gesten sind schon vordefiniert
z.B. tippen, streichen, verschieben, rotieren
 - Einheitliche Gestenerkennung in Apps

- ◆ UIGestureRecognizer-Objekte zur Erkennung
 - Subklassen für konkrete Gesten
 - Methode zum Handling implementieren

- ◆ Ablauf einer Tippgesten-Erkennung:



◆ Recognizer beim View anmelden

```
// Nachdem der View geladen wurde
-(void) viewDidLoad {
    UIPanGestureRecognizer* panRecognizer =
    [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(schiebeGeste:)];
    [self.view addGestureRecognizer:panRecognizer];
    [panRecognizer release];
}
```

◆ Action und Target

- ◆ Bei Erkennung der Geste wird der **Action-Selector** am Objekt **target** aufgerufen.
- ◆ Recognizer übergibt sich selbst an die Action-Methode

- ◆ **Methode zum Geste-Handling implementieren**
- ◆ **Recognizer bieten jeweils Methoden zur Behandlung der Geste**
- ◆ **UIPanGestureRecognizer:**

```
- (CGPoint)translationInView:(UIView*)view  
- (CGPoint)velocityInView:(UIView*)view  
- (void)setTranslation:(CGPoint)translation inView:(UIView*)view
```

- ◆ **UIRotationGestureRecognizer:**

```
@property CGFloat rotation  
@property (readonly) CGFloat velocity
```

- ◆ **Diese Methoden und Properties können in der Action-Methode verwendet werden**

◆ Beispiel einer Action-Methode:

```
// Verschieben-Geste wurde erkannt  
-(void)schiebeGeste:(UIPanGestureRecognizer*)recognizer {  
    CGPoint trl = [recognizer translationInView:self.view];  
    CGPoint oldCenter = rectView.center;  
    rectView.center = CGPointMake(oldCenter.x + trl.x, oldCenter.y + trl.y);  
    [recognizer setTranslation:CGPointZero inView:self.view];  
}
```



◆ **Recognizer haben einen Zustand**

```
@property(nonatomic,readonly) UIGestureRecognizerState state;
```

Zustand	Verhalten
possible	Geste noch nicht erkannt, möglich
recognized	Geste erkannt (diskret)
failed	Geste nicht erkannt
began	Geste erkannt (kontinuierlich)
changed	Geste modifiziert (kontinuierlich)
ended	Geste abgeschlossen (kontinuierlich)
cancelled	Geste abgebrochen (kontinuierlich)

- ◆ Vorgefertigte Recognizer für Gesten:

Geste	Recognizer
Tippen	UITapGestureRecognizer
Pinch (Zoomen)	UIPinchGestureRecognizer
Verschieben	UIPanGestureRecognizer
Wischen	UISwipeGestureRecognizer
Rotieren	UIRotationGestureRecognizer
Pressen und halten	UILongPressGestureRecognizer

UIGestureRecognizerDelegate

- ◆ **Delegate für Gesten-Recognizer ermöglicht z.B. gleichzeitige Erkennung mehrerer Gesten**

```
// Anmeldung als Delegate, z.B. in viewDidLoad...
```

```
panRecognizer.delegate = self;
```

```
// Implementation der Delegate-Methode
```

```
- (BOOL)gestureRecognizer:(UIGestureRecognizer*)gestureRecognizer  
shouldRecognizeSimultaneouslyWithGestureRecognizer:  
(UIGestureRecognizer*)otherGestureRecognizer  
{  
    return YES;  
}
```

◆ DEMO



Aufgabe 3

Erstellen einer TableView



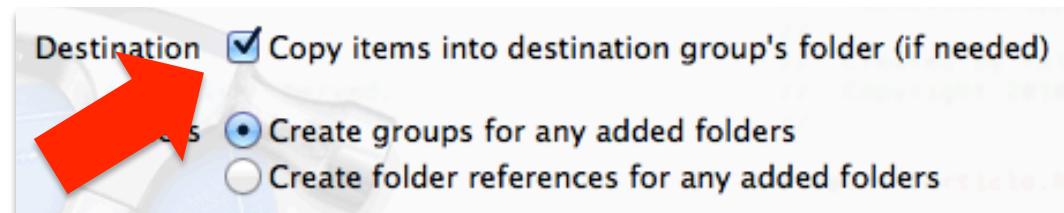
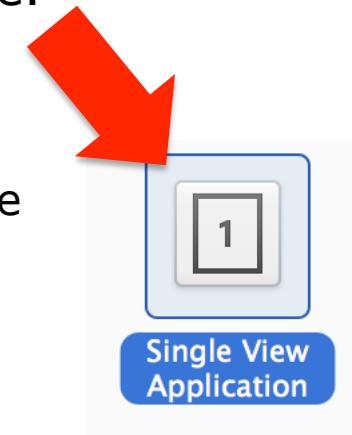
Aufgabe 3: Erstellen einer TableView

Ziel

- ◆ Eine **Tabelle/Liste** ist zu erstellen, die in ihren Zeilen „**Article**“-**Objekte** aus einem RSS-Feed darstellen kann.

Weg

- ◆ Erstelle eine neue **Single View Application** in Xcode.
- ◆ Zum Befüllen der Zellen deiner Tabelle kannst du ein Objekt der Klasse **RSSParser** verwenden.
 - Bei Übergabe einer **NSURL** an *RSSParser* wird eine Liste (NSArray) von „Article“-Objekten zurückliefert.
 - Kopiere die Dateien für die Klassen „RSSParser“, „Article“ und „RssFeed“.
 - Beispiel-Feed-Adressen sind über die Klasse „RssFeed“ zu finden.



Aufgabe 3: Erstellen einer TableView

- ◆ Baue einen **UITableView** in deine View ein.
- ◆ Dein ViewController soll **Delegate** und **DataSource** der TableView sein.
 - Dazu muss er die Protokolle **UITableViewDelegate** und **UITableViewDataSource** implementieren.
 - Verknüpfe im Interface Builder die Outlets dataSource und delegate des UITableViews mit dem Controller (File's Owner)

Aufgabe 3: Erstellen einer TableView

- ◆ Implementiere die **Methoden des dataSource** deiner TableView.
 - Die **DataSource** (ebenfalls ein Delegate) befüllt die darzustellenden Zellen mit Daten.
 - Das **Delegate** reagiert auf die Selektionen in der Tabelle
 - Jede Zelle soll Informationen zu einem „Article“-Objekt des RSS-Feed anzeigen.
 - Die Klasse RssFeed liefert dir RSS-Feeds.
 - Der *RSSParser* liefert dir die „Article“-Objekte.



Agenda Tag 2

09:00 iOS Schichten

Aufgabe: Erstellen einer TableView

12:00 Pause

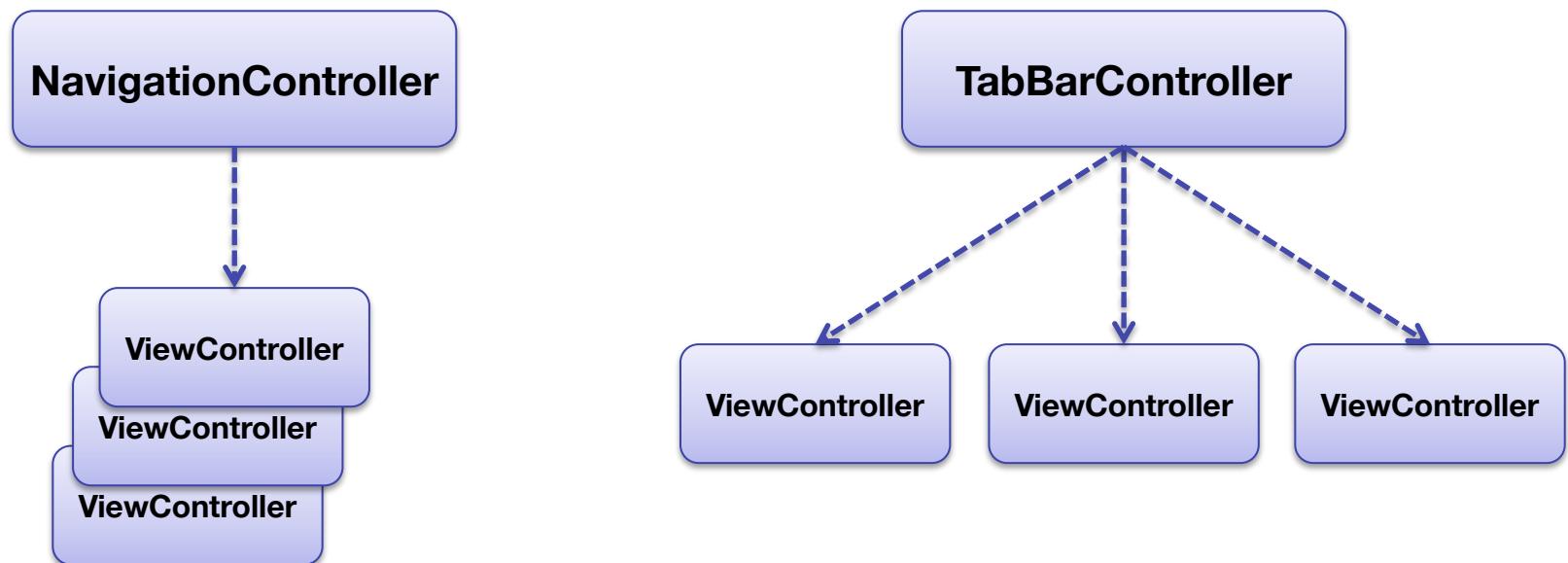
13:00 → Cocoa in a Nutshell

Aufgabe: Verwenden des NavigationControllers

17:00 Ende

UIKit: ViewController-Hierarchien

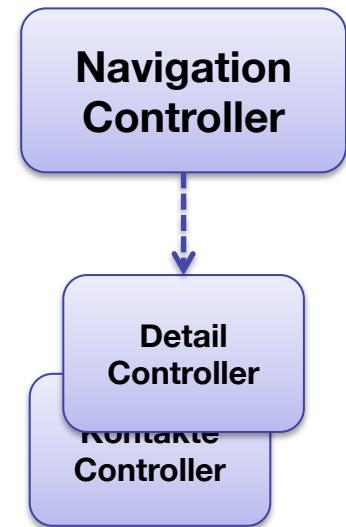
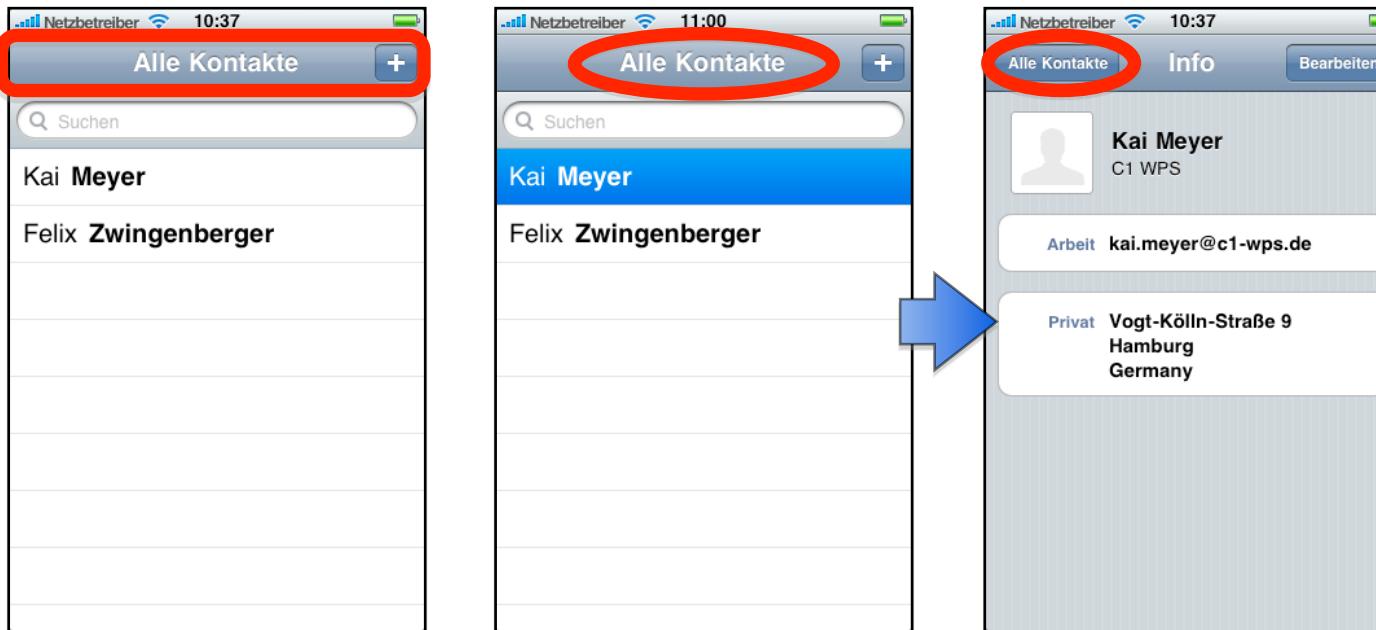
- ◆ ViewController können hierarchisch angeordnet werden
 - in speziellen Container-ViewControllern



UIKit: UINavigationController

◆ Navigation

- Konsistentes Bedienkonzept
- Allgemein -> Detailsicht
- Als Stack angeordnet



UIKit: UINavigationController

- ◆ Navigations-Bildschirme landen auf einem Stack

```
// zum Laden eines neuen ViewControllers
- (void)pushViewController:(UIViewController*)viewController animated:
    (BOOL)animated

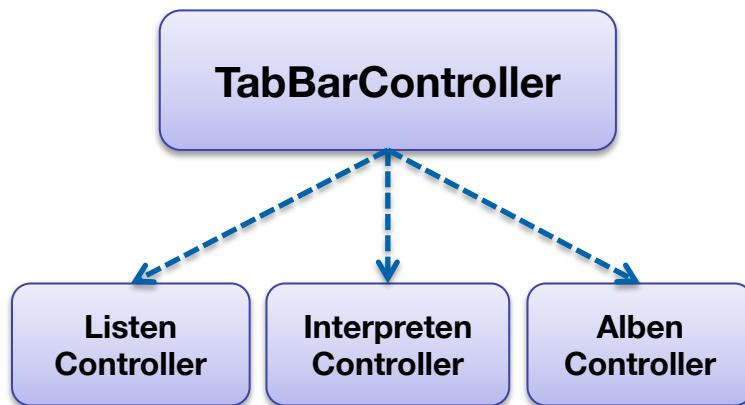
// Zum Zurücknavigieren (wird z.B. durch Button links oben aufgerufen)
- (UIViewController*)popViewControllerAnimated:(BOOL)animated
```

```
// Jeder UIViewController hat als Property den ihn
// beinhaltenden NavigationController
[self.navigationController pushViewController ...]
```

UIKit: UITabBarController

◆ Tabs

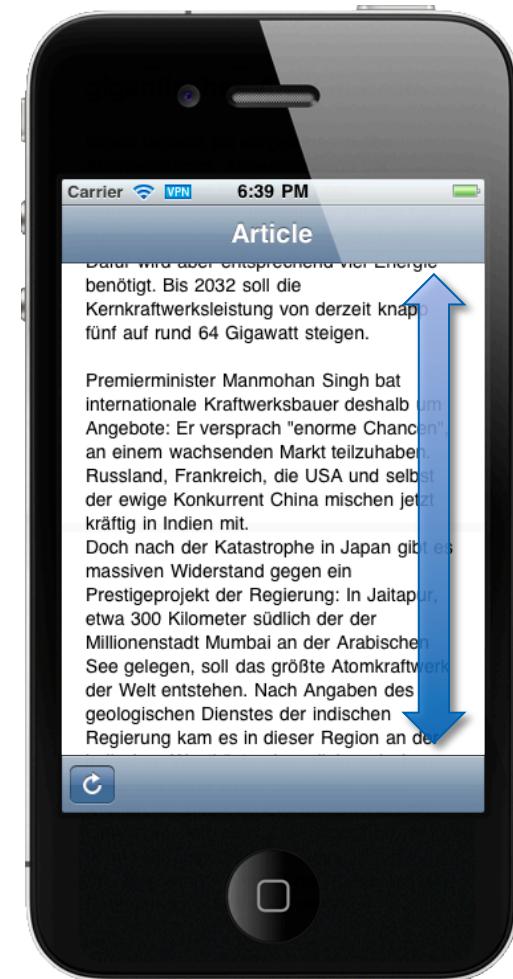
- Unabhängige Views
- Controller verwaltet Liste von Views





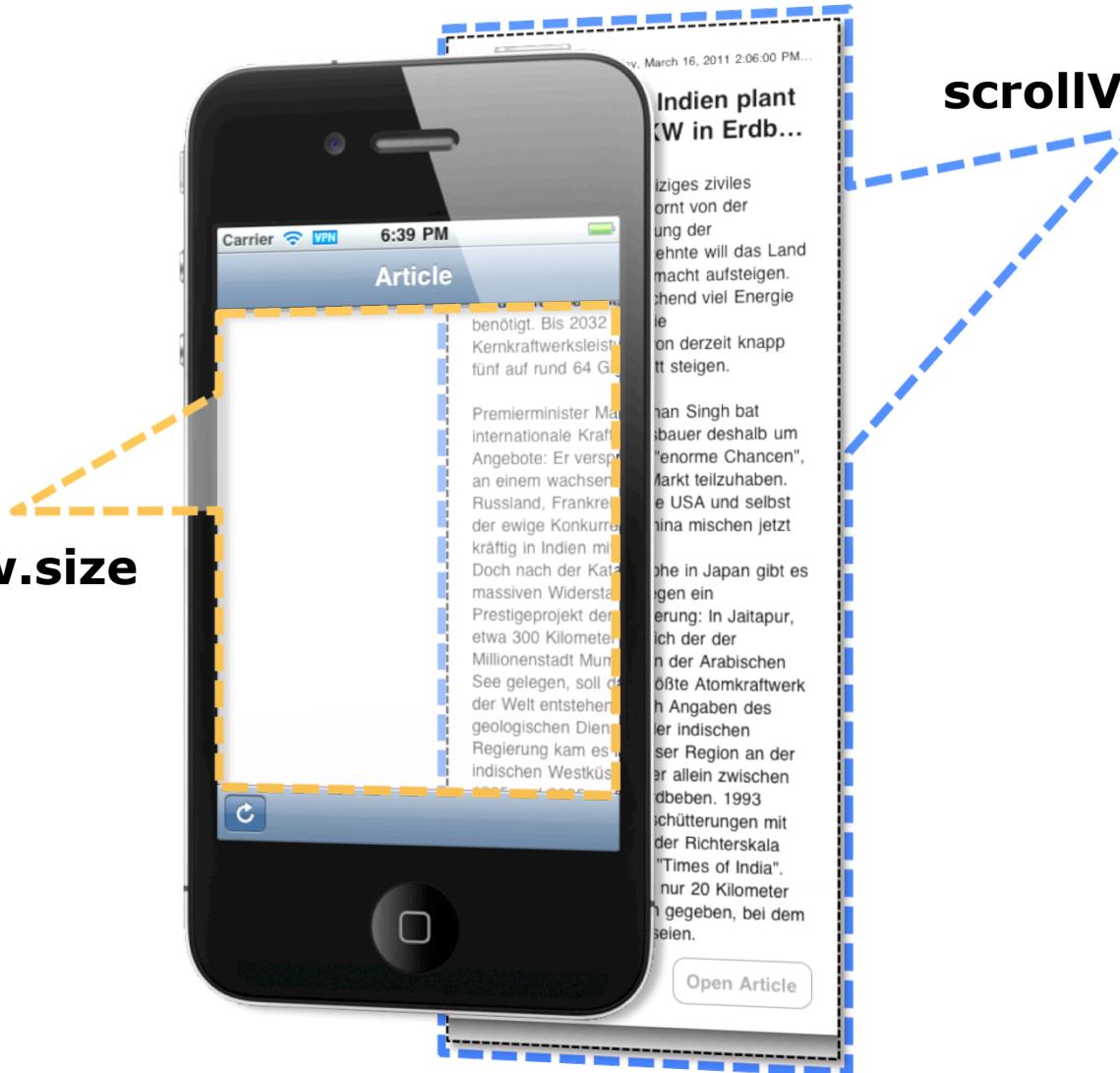
UIKit: ScrollView

- ◆ Ermöglicht das Scrollen von Views
 - wenn Views zu groß sind
 - wenn die Tastatur eingeblendet wird
- ◆ ScrollViews ermöglichen auch zoomen
 - mit Pinch-Geste





UIKit: ScrollView



scrollView.contentSize

scrollView.size

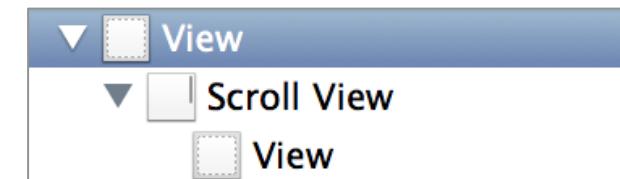
◆ Anzeigen / Verbergen der Tastatur

- wird automatisch eingeblendet, wenn ein Texteingabefeld Fokus erhält
- wird **nicht** automatisch ausgeblendet

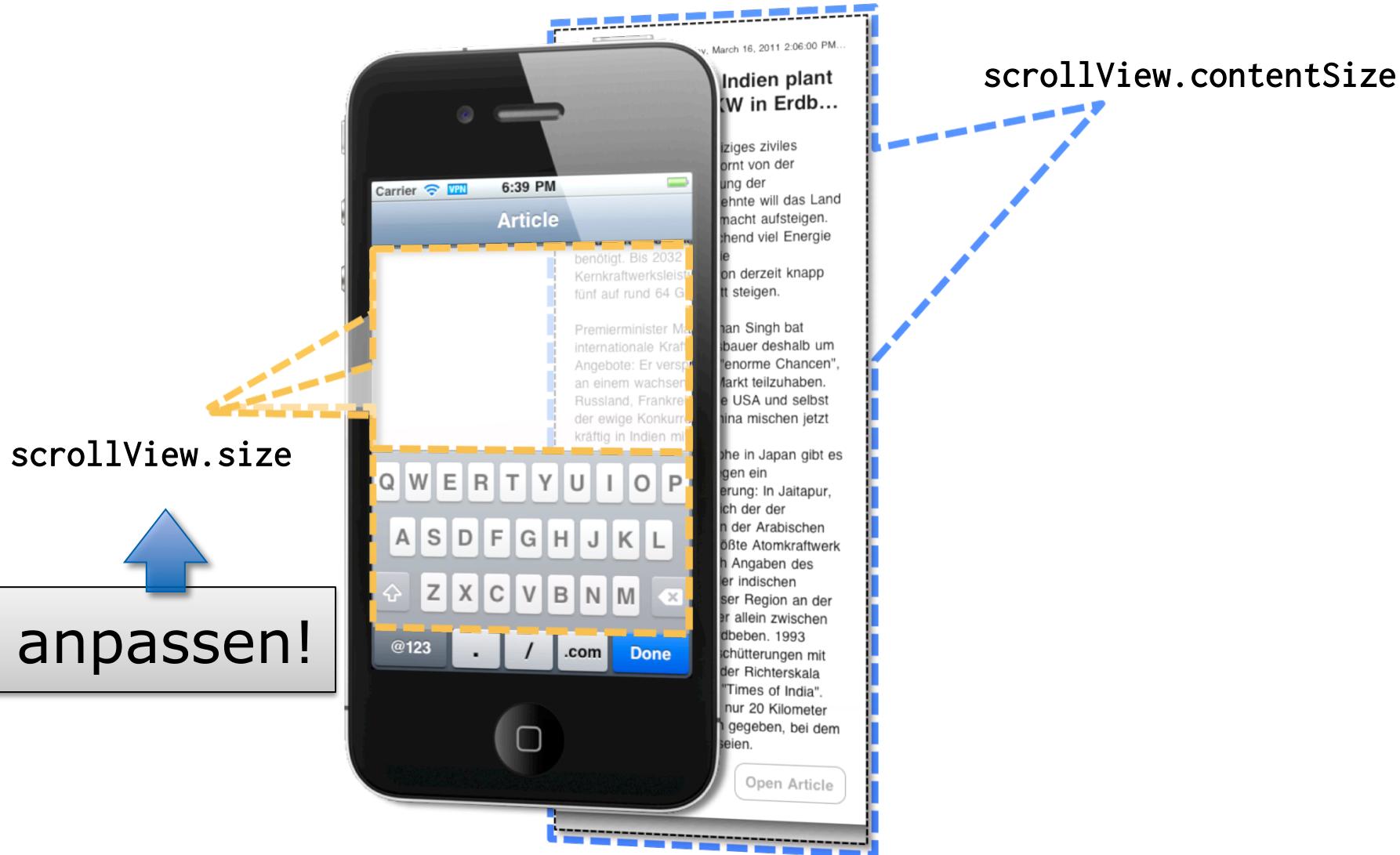
```
// Tastatur ausblenden  
-(IBAction)doneButtonPressed:(id)sender {  
    [textField resignFirstResponder];  
}
```

◆ Tastatur kann Teile der UI verdecken

- Ggf. muss die UI gescrollt werden
- Views können in UIScrollView eingebettet werden
- Anpassen der Größe des Scrollviews bei Einblendung der Tastatur



Tastatur



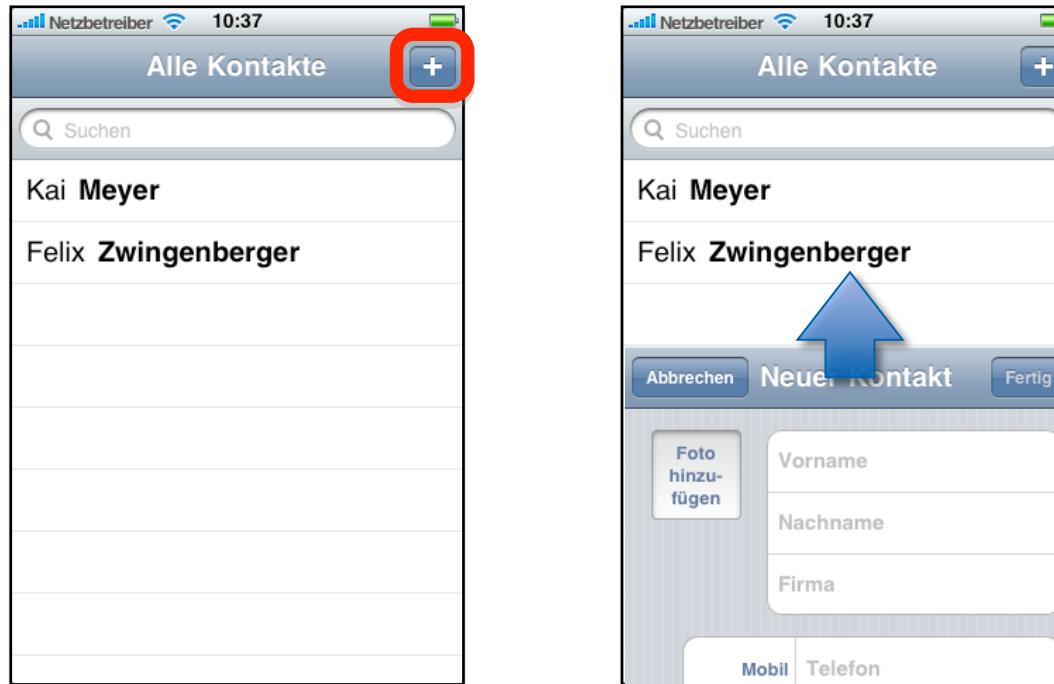
◆ Anpassen des Views

1. Anmelden für Keyboard-Notifications
2. Anpassen der ScrollView-Größe an Tastatur-Einblendung
3. Scrollen des Views (z.B. mit scrollRectToVisible:animated:)



Modale Views

- (void) presentModalViewController:(UIViewController*)modalViewController animated:(BOOL)animated
- (void) dismissModalViewControllerAnimated:(BOOL)animated





Aufgabe 4

Verwenden des NavigationControllers

Aufgabe 4: Verwenden des NavigationControllers

Ziel

- ◆ Die **App aus Aufgabe 3** ist zu **erweitern**.
 - Durch Tippen auf einen Artikel in der Liste der Artikel soll eine neue View mit **Details** des Artikels angezeigt werden.
 - ❖ Titel, Text und das Datum des Artikels sollen angezeigt werden.
 - ❖ Ein Button soll die URL zu dem Artikel in einer **WebView** öffnen (wieder ein neuer View).
 - ❖ Über Back-Buttons soll zur jeweils vorigen View **navigiert** werden können.
 - Ein Button in der TableView soll eine **ModalView** öffnen können, in der der angezeigte RSS-Feed ausgewählt werden kann.



Aufgabe 4: Verwenden des NavigationControllers

Weg

- ◆ Erzeuge in der Methode „**didFinishLaunchingWithOptions**“ deines AppDelegate einen UINavigationController.
 - Setze den UINavigationController anstelle deines UITableViewControllers als „*rootViewController*“ an dem „*window*“ deiner App.
 - Push deinen UITableViewController auf diesen UINavigationController.



Aufgabe 4: Verwenden des NavigationControllers

- ◆ Erzeuge einen **neuen UIViewController** mit XIB zum Anzeigen der Details eines Artikels - im folgenden genannt **DetailViewController**.
 - Der DetailViewController soll durch Tippen auf eine Zelle in deiner TableView zu öffnen sein.
 - Implementiere die TableView-Delegate-Methode „*didSelectRowAtIndexPath*“ in deinem Artikel-ViewController.
- ◆ Erzeuge eine Klasse **WebViewController** (mit Xib) zum öffnen der URL eines Artikels.
 - Der WebViewController ist ein ViewController mit einer **UIWebView**.
 - Der WebViewController sollte das **<UIWebViewDelegate>** Protokoll implementieren.



Aufgabe 4: Verwenden des NavigationControllers

- ◆ Füge deiner Detail-View einen **Button zum Anzeigen der WebView** hinzu.
 - **Push** zum Anzeigen der WebView **den ViewController auf den NavigationController**.
 - Teile der **UIWebView** mit, welche URL geöffnet werden soll.
 - Hinweis: Jeder ViewController besitzt automatisch eine Referenz auf den NavigationController, in den er eingebettet ist (**self.navigationController**).
- ◆ Erzeuge einen neuen ViewController zum Auswählen des RSS-Feeds. Im folgenden genannt „**FeedSelectionViewController**“.
 - Die View zur Feed-Auswahl sollte in modal angezeigt werden.



Aufgabe 4: Verwenden des NavigationControllers

- ◆ Füge der NavigationBar deines NavigationControllers ein „**UIBarButtonItem**“ hinzu (programmatisch oder per XIB). Dieser Button soll den **FeedSelectionViewController** öffnen.
 - Programmatische Einbindung:
 - ❖ Erzeuge ein neues UIBarButtonItem-Objekt
 - ❖ Setze es als „leftBarButtonItem“ dem „navigationItem“ deines TableViewControllers.
 - Schreibe eine Methode als „action“ für den Button.
 - Setze „target“ und „action“ am UIBarButtonItem.
 - ❖ **Erzeuge den FeedSelectionViewController** in der „action“.
 - ❖ Öffne in der „action“ den FeedSelectionViewController als **ModalViewController**. (siehe: „presentModalViewController“ und „dismissModalViewController“).
 - **Definiere ein Protokoll**, damit du z.B. deinen ArtikelViewController als Delegate an den FeedSelectionViewController übergeben kannst.
 - ❖ Erzeuge dazu eine neue Datei vom Typ **Objective-C Protocol**
 - ❖ Definiere im Protokoll eine Methode, um dem Delegate einen **neuen RSS-Feed zu übergeben**. Nutze diese nach der Auswahl des RSS-Feeds.
 - ❖ Das **Delegate soll den neuen RSS-Feed laden** und die ArtikelView mit „reloadData“ aktualisieren.