Finden Sie Ihren Weg

A

In diesem Kapitel ...

- Nutzen Sie das Map Framework
- Legen Sie die Position und die Zoom-Stufe einer Karte fest
- Heben Sie wichtige Punkte auf der Karte hervor
- Finden Sie den aktuellen Standort des iPhones heraus

Bisher habe ich mich bei der Funktionalität darauf konzentriert, etwas *im Rahmen eines Kontextes* möglich zu machen, zum Beispiel, wie man vom Flughafen zum Hotel kommt, Geld wechselt, das richtige Trinkgeld gibt und so weiter. Aber es gibt noch eine andere Seite des Kontextes: Das iPhone kann Informationen *über den Kontext* bereitstellen. Das Wetter ist ein Beispiel, eine Karte ein anderes.

Mit der Weiterentwicklung des iPhones in den letzten Jahren haben sowohl Apple als auch die App-Entwickler die standortbasierten Dienste deutlich ausgebaut. Diese Art von Diensten ermöglicht es Ihnen, einem Anwender zu zeigen, wo er sich gerade befindet, und ihn mit Personen in der Nähe zu verbinden.

Persönlich bin ich ein großer Fan solcher Dienste und wenn Ihre Anwender sie nutzen können, schlage ich vor, sie auch in Ihre App einzubauen.

Auf dem iPhone geht es bei standortbasierten Diensten um zwei verschiedene Aspekte:

- ✔ **Mapping:** Durch das Map Kit Framework können Sie eine Karte einfach anzeigen und Anmerkungen erstellen – genauso wie bei der Karten-App.
- ✔ **Core Location:** Core Location bietet über sein Framework eine Reihe von Diensten an, mit denen der Standort und (über den eingebauten Kompass) die Richtung des Anwenders ermittelt werden kann. (Die Richtung ist ab dem iPhone 3GS verfügbar.)

In diesem Kapitel zeige ich Ihnen, wie Sie Karten in Ihre App einbinden. In Kapitel B erkläre ich dann Core Location und seine Anwendung – auch im Hintergrund. Damit können Sie dem Benutzer über lokale Benachrichtigungen Updates schicken.

Die Karte der Fantasie

Da die Leute mittlerweile wissen, was für Möglichkeiten sie mit dem iPhone haben, wäre es für potenzielle Anwender von MobileTravel411 nicht nachvollziehbar, wenn ich keine Karte einbinden würde. Für viele Reisende ist eine aufgeklappte Karte ein sicherer Hinweis auf einen Touristen (abgesehen von einem aufgeschlagenen Reiseführer). In diesem Kapitel zeige ich Ihnen, wie Sie auf dem iPhone eine Karte von so gut wie jedem Eckchen der Welt anzeigen

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

und sogar Ihre Position dort lokalisieren können. Wie schon erwähnt, eröffnet das Wissen um den eigenen Standort ganz neue Anwendungsmöglichkeiten und es hebt das iPhone auch von einem Desktop-Rechner ab.

Aus diesem Grund nutze ich in diesem Kapitel häufiger die Schritt-für-Schritt-Anleitungen, die ich auch bei ReturnMeTo verwendet habe. Das Einbinden von Karten in Ihre App ist ein wichtiges, seit dem iPhone 3.0 SDK vorhandenes Feature und ich möchte, dass Sie wirklich verstanden haben, wie Sie vorgehen. Wollen Sie mich also begleiten, können Sie mit dem Projekt iPhoneTravel411 Chapter 16 auf meiner Website (`www.nealgoldstein.com`) beginnen. Die Version, die am Ende dieses Kapitels steht, finden Sie dann in iPhoneTravel411 Chapter 17.

Ach, und übrigens: Sie werden feststellen, dass die Arbeit mit Karten auf dem iPhone erstaunlich viel Spaß macht, weil Apple es so einfach gestaltet. Sie haben das in Kapitel 16 gesehen, wo Sie schon eine Karte angezeigt haben (sogar mit der Möglichkeit, sie zu verschieben und zu zoomen), indem Sie einfach einen View Controller und eine NIB-Datei gebaut haben.

In diesem Kapitel ergänzen Sie die Karte um einige Dinge. Ich zeige Ihnen, wie Sie in der Karte einen gewünschten Bereich anzeigen (zum Beispiel Heathrow Airport oder London), wie Sie Anmerkungen hinzufügen (diese hübschen Stecknadeln, über die man eine Beschreibung des Ortes erhalten kann) und sogar wie Sie den aktuellen Standort des Anwenders anzeigen. (Sie können sogar den aktuellen Standort in einen Adressbuch-Eintrag umwandeln, aber darauf gehe ich hier nicht ein.)

Abbildung a.1 zeigt eine schönere Karte als die einfache Map View für den Weg vom Flughafen nach London.

Abbildung a.1: Karte Heathrow – London

Eines der tollen neuen Features im iPhone 3.0 SDK (und neuer) ist ein Framework: `MapKit`. Wie in Kapitel 16 beschrieben, erhalten Sie durch `MapKit` die Möglichkeit, eine einfache Karte anzuzeigen und mit wenig Aufwand damit zu arbeiten.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Die Karte sieht aus wie die in der mitgelieferten Karten-App, damit man sich gleich in einer vertrauten Umgebung wiederfindet.

MKMapView

Die Basis aller Kartendarstellungen auf dem iPhone ist die `MKMapView`. Dabei handelt es sich um eine Subklasse von `UIView`, die Sie direkt nutzen können. Sie verwenden sie zum Anzeigen von Kartendaten und zum Anpassen des Inhalts aus Ihrer App heraus. Sie können die Karte für gegebene Koordinaten anzeigen, die Größe des Bereichs definieren und eigene Anmerkungen festlegen.

Sie haben das `MapKit`-Framework in Kapitel 16 zum Projekt hinzugefügt.

Erstellen Sie eine Map View, können Sie die Region festlegen, die beim Start angezeigt werden soll. Dazu setzen Sie die *Region*-Eigenschaft der Karte. Eine Region wird dabei durch einen Mittelpunkt sowie einen horizontalen und einen vertikalen Abstand definiert. Die letzten beiden Werte sind die *Spanne*. Sie legt fest, wie viel man auf der Karte sieht, und sie führt zu einer Zoom-Stufe. Je kleiner die Spanne ist, desto größer ist der Zoom.

Die Map View erkennt auch die Standard-Gesten für Karten.

- ✔ scrollen
- ✔ auf- und zuziehen
- ✔ Doppelt tippen zum Hereinzoomen
- ✔ Doppelt mit zwei Fingern tippen zum Herauszoomen (das haben Sie vielleicht noch gar nicht gewusst)

Sie können auch mit einer Eigenschaft den Kartentyp definieren (Normal, Satellit oder Hybrid).

Da `MapKit` ganz neu entwickelt worden war, konnten gleich die Grenzen des iPhones berücksichtigt werden. Im Ergebnis wird die Performance auf dem iPhone optimiert, indem Daten gepuffert, der Speicher verwaltet und ein nahtloser Wechsel zwischen verschiedenen Verbindungsvarianten ermöglicht wird (zum Beispiel von 3G nach WLAN).

Die Kartendaten selbst sind von Google und man braucht eine Netzwerkverbindung, um sie holen zu können. Da das `MapKit`-Framework Google-Services für die Daten nutzt, sind Sie an die Verwendungsrichtlinien der Google Maps/Google Earth API gebunden.

Sie sollten zwar keine Unterklasse von `MKMapView` bilden, aber Sie können das Verhalten einer Map View beeinflussen, indem Sie ein Delegate-Objekt bereitstellen. Dabei kann es sich um ein beliebiges Objekt in Ihrer App handeln, solange es dem Protokoll `MKMapViewDelegate` folgt. Sie können dazu ganz einfach das Modell nutzen, das Sie in Kapitel 16 entwickelt haben.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

(Ich habe das durchaus ernst gemeint, als ich schrieb, Sie seien jetzt mit dem langweiligen Anlegen von Dateien fertig!)

Die Karte aufhübschen

Es ist zwar interessant, eine Karte der USA zu haben, aber nicht sehr hilfreich, wenn Sie nach London fliegen wollen. Die folgenden Abschnitte zeigen Ihnen, was Sie zu tun haben, um die Karte nützlicher zu machen – zum Beispiel, sie um Heathrow oder London herum zu zentrieren.

In Kapitel 16 haben Sie eine View-Controller-Klasse (`MapController`) mit den passenden Outlets und eine NIB-Datei erzeugt, die ein Objekt vom Typ `MKMapView` erstellt und alle Outlets setzt, wenn der Anwender in der Haupt-View KARTE wählt. Eventuell haben Sie auch schon eine Initialisierungsmethode eingerichtet.

Landscape-Modus und aktuelle Position hinzufügen

Um gleich mit einem richtigen Kracher zu beginnen, wäre es sehr hilfreich, die Karte im Landscape-Modus anzeigen lassen zu können.

Fügen Sie in Xcode in die Datei `MapController.m` die folgende Methode ein:

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
          (UIInterfaceOrientation)toInterfaceOrientation {

  return YES;
}
```

Das ist alles. Jetzt wird die Karte auch im Querformat angezeigt und `MapKit` kümmert sich um alles!

Wie ist es mit dem Anzeigen der aktuellen Position? Genauso einfach!

Sie haben in Kapitel 16 zwar schon das `MapKit`-Framework hinzugefügt, aber Sie müssen dem `MapController` immer noch mitteilen, dass er es zu nutzen hat. Fügen Sie in `MapController.h` folgende Zeile ein:

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

In der Datei `MapController.m` entfernen Sie die Kommentarzeichen um die Methode `viewDidLoad` und fügen den fett gedruckten Code ein.

```
- (void)viewDidLoad {
  [super viewDidLoad];

  mapView.showUserLocation = YES;
}
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

`showUserLocation` ist eine Eigenschaft von `MKMapView`, durch die die Map View weiß, ob der Standort des Benutzers angezeigt werden soll. Setzen Sie sie auf `YES`, erhalten Sie den gleichen blauen, pulsierenden Punkt, den Sie auch von der eingebauten Karten-App kennen.

Kompilieren Sie die App in ihrem aktuellen Zustand und lassen Sie sie dann laufen, erhalten Sie ein Ergebnis wie in Abbildung a.2 – eine Karte der USA im Querformat mit einem blauen Punkt, der den aktuellen Standort des Telefons darstellt. (Es gibt eventuell eine kleine Verzögerung, bis das iPhone Ihren Standort bestimmen kann.) Um den Landscape-Modus zu erhalten, müssen Sie das Telefon natürlich drehen, im Simulator-Menü HARDWARE|ROTATE RIGHT (oder ROTATE LEFT) wählen oder ⌘+→ oder ← drücken.

Abbildung a.2: Anzeige einer Karte im Querformat mit dem Benutzerstandort

Sehen Sie die aktuelle Position nicht, sollten Sie kontrollieren, ob Sie das Outlet `mapView` in der NIB-Datei mit der Map View verbunden haben. Das sollte eigentlich schon in Kapitel 16 geschehen sein.

Sie erhalten den aktuellen Standort, wenn Sie die App *auf dem iPhone* laufen lassen. Im Simulator ist der Standort immer der der Apple-Zentrale. Tippen Sie auf den blauen Punkt, erhalten Sie eine *Anmerkung* (Annotation). Sie werden noch erfahren, wie Sie diesen Text anpassen und sogar die aktuelle Adresse anzeigen lassen können.

Die Region

Diese Karte ist zwar hübsch, aber für unsere Zwecke immer noch nicht sehr nützlich.

Wie ich schon am Anfang des Kapitels erwähnt habe, sollten Sie idealerweise nach der Landung in Heathrow eine Karte sehen, bei der Heathrow in der Mitte ist – anders als bei der USA-Karte. Das lässt sich aber leicht einrichten.

Schauen wir uns zunächst an, wie man die Karte zentriert.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Fügen Sie in `MapController.m` den folgenden Code ein:

```
- (void)updateRegionLatitude:(float) latitude
        longitude:(float) longitude
        latitudeDelta:(float) latitudeDelta
        longitudeDelta:(float) longitudeDelta {
  MKCoordinateRegion region;
  region.center.latitude = latitude;
  region.center.longitude = longitude;
  region.span.latitudeDelta = latitudeDelta;
  region.span.longitudeDelta = longitudeDelta;
  [mapView setRegion:region animated:NO];
}
```

Jetzt fehlt noch die Deklaration der Methode in `MapController.h`:

```
- (void)updateRegionLatitude:(float)
     latitude longitude:(float)
     longitude latitudeDelta:(float)
     latitudeDelta longitudeDelta:(float) longitudeDelta;
```

Mit der Definition der *Region* legen Sie die Mitte der Karte und die Zoom-Stufe fest. Das erreichen Sie einfach durch die Anweisung

```
[mapView setRegion:region animated:NO];
```

Eine *Region* ist eine Eigenschaft einer Map View, in der vier Werte enthalten sind (siehe Abbildung a.3).

1. `region.center.latitude` legt die geografische Breite des Mittelpunktes der Karte fest.
2. `region.center.longitude` legt die geografische Länge des Mittelpunktes der Karte fest.

   Setzen Sie diese Werte zum Beispiel auf

   ```
   region.center.latitude = 51.471184;
   region.center.longitude = -0.452542;
   ```

   liegt das Zentrum der Karte auf dem Flughafen Heathrow.
3. `region.span.latitudeDelta` legt den Abstand von Norden nach Süden (in Grad) fest, der auf der Karte angezeigt werden soll. Ein Grad geografischer Breite entspricht ungefähr 111 km. Ein `region.span.latitudeDelta` von 0.0036 gibt einen Nord-Süd-Abstand auf der Karte von etwa 400 m vor. Breitengrade nördlich des Äquators haben positive Werte, südlich des Äquators sind sie negativ.
4. `region.span.longitudeDelta` legt den Ost-West-Abstand (in Grad) fest, der auf der Karte angezeigt werden soll. Leider ist der Abstand zwischen zwei Grad-Linien stark von der geografischen Breite abhängig. So entspricht ein Grad geografischer Länge am Äquator etwa 111 km, während dieser Wert zu den Polen hin auf null schrumpft. Längengrade östlich des Nullmeridians (der Nullmeridian verläuft nach internationalen Vereinbarungen

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

durch das Royal Observatory in Greenwich im Osten Londons) haben positive Werte, Längengrade westlich davon negative Werte.

Die Span-Werte sorgen für einen impliziten Zoom der Karte, aber die tatsächlich angezeigte Region muss nicht unbedingt dem angegebenen Span entsprechen, da die Karte die Zoom-Stufe wählt, die die Region am besten darstellt. Das kann auch dazu führen, dass sich beim Ändern des Mittelpunktes die Zoom-Stufe ändert, weil die Abstände, die durch die Span-Werte definiert werden, für verschiedene Längen- und Breitengrade unterschiedlich sein können. Um das zu beeinflussen, haben die klugen Jungs bei Apple eine Eigenschaft mit eingebaut, mit der Sie dafür sorgen können, dass sich die Mittelpunkt-Koordinaten ändern lassen, ohne dass sich die Zoom-Stufe anpasst.

```
@property (nonatomic) CLLocationCoordinate2D
          centerCoordinate
```

Ändern Sie diese Eigenschaft, wird die Karte um die neuen Koordinaten zentriert und die Span-Werte werden so angepasst, dass die Zoom-Stufe bestehen bleiben kann.

Diesen Typ `CLLocationCoordinate2D` werden Sie auch sonst häufiger nutzen, daher möchte ich noch mehr über ihn erzählen. Er ist eine Struktur, die eine geografische Koordinate im WGS-84-Referenzsystem enthält (das Referenzsystem, das vom Global Positioning System genutzt wird).

```
typedef struct {
  CLLocationDegrees latitude;
  CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

- ✔ `latitude` ist die geografische Breite in Grad.
- ✔ `longitude` ist die geografische Länge in Grad.

Um die Karte um Heathrow zu zentrieren, schicken Sie die Nachricht `updateRegionLatitude:latitude:longitude:latitudeDelta:longitudeDelta` (den gerade eingegebenen Code), wenn die View geladen wurde – also in `viewDidLoad`. Sie haben dort schon ein bisschen Code zum Anzeigen des aktuellen Standorts, fügen Sie also nur den fett gedruckten Code hinzu.

```
- (void)viewDidLoad {

  [super viewDidLoad];
  mapView.showsUserLocation = YES;
  CLLocationCoordinate2D initialCoordinate =
                        [destination initialCoordinate];
  [self updateRegionLatitude:
       initialCoordinate.latitude
       longitude: initialCoordinate.longitude
       latitudeDelta:.2 longitudeDelta:.2];
  self.title = [destination mapTitle];
}
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

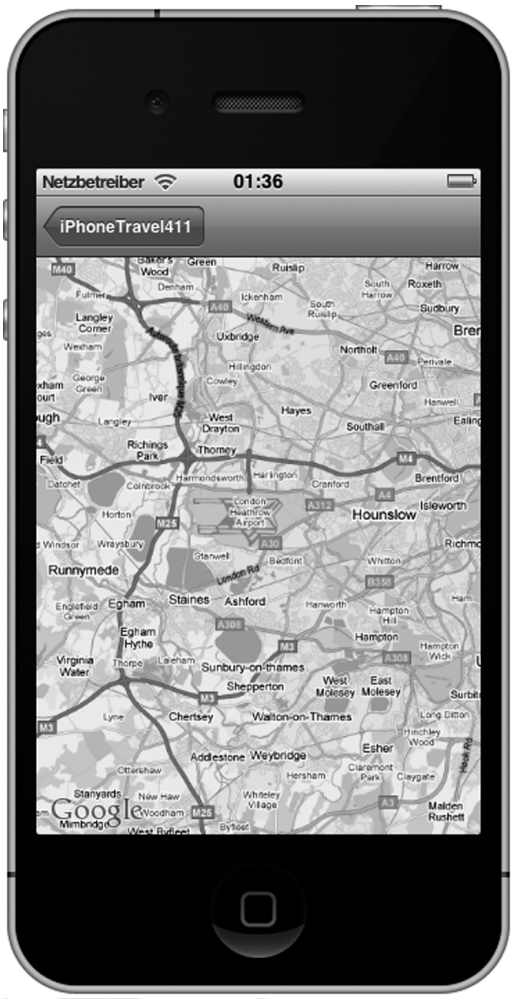

Abbildung a.3: Wie Regionen genutzt werden

Das passiert im Code:

1. Die Nachricht `initialCoordinate` wird an das Objekt `Destination` geschickt (denken Sie an Ihr Modell aus Kapitel 16), um die initialen Koordinaten zu erhalten, die Sie anzeigen wollen. Das Modell erhält ein wenig mehr Funktionalität, so das Festlegen dieser Position. Der Anwender hat die Position vielleicht beim Planen seiner Reise angefragt (darauf gehe ich hier im Buch aber nicht ein, sondern ich überlasse es dem geneigten Leser als Übung) oder es ist eine Standard-Position, für die Sie sich beim Schreiben des Codes entschieden haben (zum Beispiel ein Flughafen, der für das Ziel angegeben ist).
2. Der Titel der Karte wird durch die Nachricht `mapTitle` an das `Destination`-Objekt gesetzt – noch mehr Verantwortung für das Modell.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Damit das alles funktioniert, müssen Sie noch Code in `Destination.m` einfügen. Der folgende Code gibt die geografische Breite und Länge für Heathrow zurück:

```
- (CLLocationCoordinate2D)initialCoordinate {

  CLLocationCoordinate2D startCoordinate;
  startCoordinate.latitude=51.471184;
  startCoordinate.longitude=-0.452542;
  return startCoordinate;
}
- (NSString*) mapTitle{

  return @" Karte";
}
```

Sie müssen `MapKit` mit in `Destination` aufnehmen, fügen Sie daher folgende Zeile in `Destination.h` ein:

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

Zudem benötigen Sie dort auch noch folgende Zeilen (direkt nach den geschweiften Klammern):

```
- (CLLocationCoordinate2D)initialCoordinate;
- (NSString*)mapTitle;
```

Kompilieren und bauen Sie Ihr Projekt, sollte das Ergebnis wie in Abbildung a.4 aussehen.

Tippen Sie auf den Punkt, der in Abbildung a.2 zu sehen war, erhalten Sie ein Fensterchen mit einer *Anmerkung* Aktueller Standort. Sie können auch eigene Anmerkungen vornehmen, was Thema des nächsten Abschnitts ist.

Abbildung a.4: Die Region legt fest, was Sie in der Karte sehen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Standort-Änderungen verfolgen

Sie können auch die Änderungen des Benutzerstandorts verfolgen, indem Sie Key-Value Observing nutzen, um die Karte entsprechend der Benutzerbewegung zu verschieben. Ich gehe auf das Key-Value Observing nicht weiter ein, sondern zeige nur den Code. Er ähnelt der Registrierung auf `UIKeyboardWillShowNotification` von ReturnMeTo in Kapitel 8.

Zunächst fügen Sie den fett gedruckten Code in `MapController.m` in `viewDidLoad:` ein, um einen Observer zu erhalten, der bei Änderungen eines Wertes aufgerufen wird – hier `userLocation`.

```
- (void)viewDidLoad {
  [super viewDidLoad];
  mapView.showsUserLocation = YES;
  CLLocationCoordinate2D initialCoordinate = [map
                                    initialCoordinate];
  [self updateRegionLatitude: initialCoordinate.latitude
    longitude:initialCoordinate.longitude
    latitudeDelta:.2
    longitudeDelta:.2];

  self.title = [destination mapTitle];
  [mapView.userLocation addObserver:self
    forKeyPath:@"location"
    options:0
    context:NULL];
}
```

Damit wird die Nachricht `observeValueForKeyPath::` an den Observer geschickt (`self` oder `RootViewController`). Um die Methode in `Destination.m` zu implementieren, geben Sie folgenden Code ein:

```
- (void)observeValueForKeyPath:(NSString *) keyPath
                ofObject:(id) object
                change:(NSDictionary *) change
                context:(void *) context {

  NSLog (@"Location changed");
}
```

In dieser Methode liefert das Feld `keyPath` die `mapView.userLocation.location` zurück, die Sie für die aktuelle Position verwenden können. Im Beispiel zeige ich einfach eine Meldung in der Debugger Console an, aber Sie können auch die Karte neu zentrieren, wenn sich der Anwender eine gewisse Strecke bewegt hat.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Vorsicht Technik: Das ist nicht genau der gleiche Standort, den Sie vom `CLLocationManager` bekommen würden – er ist für die Karte optimiert, während `CLLocationManager` (den Sie in Kapitel B nutzen werden) den »echten« Benutzer-Standort liefert.

Natürlich müssen Sie das auf einem echten iPhone laufen lassen, damit sich die Position auch ändert.

Anmerkungen

Die Klasse `MKMapView` ermöglicht es auch, die Karte mit Anmerkungen zu versehen, um eigene Informationen hinzuzufügen. Solch eine Anmerkung besteht aus zwei Teilen – der Anmerkung selbst mit den eigentlichen Daten sowie die Anmerkungs-View, die diese Daten anzeigt.

Die passende Anmerkung

Eine Anmerkung spielt eine ähnliche Rolle wie das in Kapitel 16 erzeugte Dictionary für die Texte der Felder einer Tabellen-View. Beide dienen als Modell für die entsprechende View, wobei ein View Controller beides verbindet.

Anmerkungs-Objekte sind beliebige Objekte, die dem Protokoll `MKAnnotation` folgen. Meist handelt es sich um bestehende Klassen in Ihrem Anwendungs-Modell. Ein Anmerkungs-Objekt muss die Position (Koordinaten) der Karte und den anzuzeigenden Text für das Fensterchen kennen. Durch das Protokoll `MKAnnotation` muss eine Klasse die Eigenschaft `coordinate` implementieren. In diesem Fall ist es für die Objekte `Airport` und `City` sinnvoll, als Anmerkungs-Objekt zu dienen. Denn diese Modell-Objekte wissen schon, welchen Flughafen oder welche Stadt sie repräsentieren. Dann können sie sich auch um die Koordinaten und die Texte kümmern.

Die folgenden Schritte sind dafür notwendig:

1. **Fügen Sie die Import-Anweisung für `MapKit` in die Dateien `Airport.h` und `City.h` ein.**

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

2. **Lassen Sie die Klassen `City` und `Airport` das Protokoll `MKAnnotation` übernehmen.**

   ```
   @interface City : NSObject <MKAnnotation> {
   @interface Airport : NSObject <MKAnnotation> {
   ```

3. **Fügen Sie die folgenden Instanzvariablen sowohl in `Airport.h` als auch in `City.h` ein.**

   ```
   CLLocationCoordinate2D coordinate;
   ```

4. **Fügen Sie die folgende Eigenschaft und Methode sowohl in `Airport.h` als auch in `City.h` ein.**

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
@property (nonatomic) CLLocationCoordinate2D
        coordinate;
- (NSString*) title;
```

Das Protokoll `MKAnnotation` erfordert eine Eigenschaft `property` – die Methode `title` ist optional.

5. **Fügen Sie eine Synthesize-Anweisung sowohl in `Airport.h` als auch in `City.h` ein.**

```
@synthesize coordinate;
```

6. **Implementieren Sie die Methode `title` in `Airport`, indem Sie in `Airport.m` folgenden Code einfügen.**

```
- (NSString*)title {

  return airportName;
}
```

`Airport` stellt den Namen des Flughafens für das Anmerkungsfenster bereit.

7. **Implementieren Sie die Methode `title` in `City`, indem Sie in `City.m` folgenden Code einfügen.**

```
- (NSString*)title {

  return cityName;
}
```

`City` stellt den Namen der Stadt für das Anmerkungsfenster bereit.

8. **Fügen Sie den fett gedruckten Code in die Methode `initWithName::` in `Airport.m` ein, um die Koordinaten für Heathrow bereitzustellen.**

```
- (id)initWithName:(NSString*) name airportID:(int)
                                        theAirport{

  if ((self = [super init])) {
    airportName = name;
    coordinate.latitude = 51.471184;
    coordinate.longitude= -0.452542;
  }
  return self; }
```

`Airport` erhält die geografische Breite und Länge von Heathrow in der Eigenschaft `coordinate`, die dann von der Map View zum Platzieren der Anmerkung genutzt wird.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

9. **Fügen Sie den fett gedruckten Code in die Methode `initWithCity:` in `City.m` ein, um die Koordinaten für London bereitzustellen.**

```
- (id)initWithCity:(NSString*) name {
  if ((self = [super init])) {
    self.cityName = name;
    coordinate.latitude = 51.500153;
    coordinate.longitude= -0.126236;
  }
  return self;
}
```

`City` erhält die geografische Breite und Länge von London in der Eigenschaft `coordinate`, die dann von der Map View zum Platzieren der Anmerkung genutzt wird.

10. **Fügen Sie den fett gedruckten Code in `Destination.m` ein.**

```
@synthesize annotations;

- (id)initWithName:(NSString*) theDestination {
  if ((self = [super init])) {
    destinationName = theDestination;
    airport = [[Airport alloc] initWithName:
          NSLocalizedString(@"Heathrow", @"Heathrow")
                                        airportID:1];
    currency = [[Currency alloc] initWithCurrency:
        NSLocalizedString(@"pound", @"pound currency")
                                    currencyID: @"GBP"];
    city = [[City alloc] initWithCity:
             NSLocalizedString(@"London", @"London")];
    weather = [[Weather alloc] initWithCity:
             NSLocalizedString(@"London", @"London")];
    annotations = [[NSMutableArray alloc]
                                    initWithCapacity:4];
    [annotations addObject:airport];
    [annotations addObject:city];
  }
  return self;
}
```

Das Objekt `Destination` erzeugt ein Array mit Anmerkungs-Objekten. (Wie Sie diese verwenden, zeige ich Ihnen gleich.)

11. **Fügen Sie die Instanzvariable für das Array `annotations` in `Destination.h` ein und machen Sie es zu einer Eigenschaft.**

```
NSMutableArray *annotations;
...
@property (nonatomic, retain)
                        NSMutableArray *annotations;
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

So weit, so gut. Sie haben zwei Objekte `City` und `Airport`, die das Protokoll `MKAnnotation` befolgen, eine Eigenschaft `coordinate` deklarieren und eine Methode `title` implementieren. Das Objekt `Destination` erzeugt dann ein Array mit diesen Anmerkungen. Jetzt müssen wir das Array nur noch an die Map View schicken, um die Anmerkungen auch anzeigen zu können. Das entspricht in etwa dem Anzeigen der Buttons in der Toolbar am unteren Rand der Airport View in Kapitel 16.

Anmerkungen anzeigen

Das Anzeigen der Anmerkungen ist einfach. Sie müssen nur die fett gedruckte Codezeile in `MapController.m` einfügen.

```
- (void)viewDidLoad {
  [super viewDidLoad];
  mapView.showsUserLocation = YES;
  CLLocationCoordinate2D initialCoordinate = [destination
           initialCoordinate];
  [self updateRegionLatitude: initialCoordinate.latitude
           longitude:initialCoordinate.longitude
           latitudeDelta:.2 longitudeDelta:.2];
  self.title = [destination mapTitle];

  [mapView.userLocation addObserver:self
                        forKeyPath:@"location" options:0
                        context:NULL];
  [mapView addAnnotations: destination.annotations];
}
```

Der `MapController` verschickt die Nachricht `addAnnotations:` an die Map View und übergibt dabei ein Array mit Objekten, die dem Protokoll `MKAnnotation` entsprechen – jede besitzt also eine Eigenschaft `coordinate` und eine optionale Methode `title` (und eventuell noch `subtitle`), wenn Sie etwas im Anmerkungsfensterchen anzeigen wollen.

Die Map View zeigt die Anmerkungen auf dem Bildschirm an, indem sie ihrem Delegate die Nachricht `mapView:viewForAnnotation:` schickt. Diese Nachricht wird für jedes Anmerkungs-Objekt im Array versendet. An dieser Stelle können Sie eine eigene View erzeugen oder `nil` für die Standard-View zurückgeben. (Implementieren Sie diese Delegate-Methode nicht – was Sie hier nicht tun –, wird ebenfalls die Standard-View genutzt.)

Das Erstellen eigener Anmerkungs-Views geht über den Rahmen dieses Buches hinaus (auch wenn ich Ihnen noch verraten kann, dass es am einfachsten ist, die Image-Eigenschaft einer Anmerkungs-View zu setzen). Hier reicht uns aber auch die normale View. Es wird ein Pin an den angegebenen Koordinaten (aus dem Delegate `City` und `Airport`) gezeigt. Tippt der Anwender den Pin an, werden Titel und Subtitel angezeigt, sofern die Methoden `title` beziehungsweise `subtitle` im Delegate implementiert sind.

Diese Standard-View zeigt rote Pins auf der Karte an. Später werde ich Ihnen zeigen, wie Sie eine eingebaute Anmerkungs-View nutzen können, die ein bisschen flexibler ist. Sie können

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

dann eine Pin-Farbe auswählen, die Pins so animieren, dass sie auf die Karte fallen, und ein paar Einstellungen so ändern, dass sich die Pins verschieben lassen.

Sie können der Anmerkungs-View noch andere Dinge hinzufügen, ohne eine eigene View bauen zu müssen – so zum Beispiel einen Disclosure-Button (den mit dem weißen Pfeil auf blauem Grund, den Sie auch aus der Tabellen-View kennen) oder einen Info-Button (wie er in vielen kleineren Apps zu sehen ist). Das dürfen Sie selbst gerne einmal ausprobieren, wenn Sie wollen.

Kompilieren und bauen Sie Ihr Projekt, sehen Sie eine der Anmerkungen, die Sie gerade hinzugefügt haben (siehe Abbildung a.5).

Abbildung a.5: Eine Anmerkung

Mehrere Anmerkungen anzeigen

Die Karte sieht jetzt zwar schon richtig gut aus, aber in bestimmten Situationen (zum Beispiel, wenn Sie gerade am Flughafen gelandet sind) wäre es viel besser, Sie würden auf der Karte gleichzeitig Heathrow und London sehen (und die entsprechenden Anmerkungen), ohne die Region im Code festlegen zu müssen. Das ist gar nicht so schwierig. Fügen Sie in `MapController.m` den Code aus Listing a.1 ein und ergänzen Sie die Deklaration in `MapController.h`.

```
- (MKCoordinateRegion)
              regionForAnnotationGroup: (NSArray*) group {

  double maxLonWest= 0;
  double minLonEast = 180;
  double maxLatNorth = 0;
  double minLatSouth = 180;
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
  for (City* location in group) {
    if (fabs(location.coordinate.longitude ) >
          fabs(maxLonWest))
      maxLonWest = location.coordinate.longitude;
    if (fabs(location.coordinate.longitude ) <
          fabs(minLonEast))
      minLonEast = location.coordinate.longitude;
    if (fabs(location.coordinate.latitude ) >
          fabs(maxLatNorth))
      maxLatNorth = location.coordinate.latitude;
    if (fabs(location.coordinate.latitude ) <
          fabs(minLatSouth))
      minLatSouth = location.coordinate.latitude;
  }

  double centerLatitide =
        maxLatNorth - (((maxLatNorth) - (minLatSouth))/2);
  double centerLongitude =
           maxLonWest - (((maxLonWest) - (minLonEast))/2);

  MKCoordinateRegion region;
  region.center.latitude = centerLatitide;
  region.center.longitude = centerLongitude;
  region.span.latitudeDelta =
                    fabs(maxLatNorth - minLatSouth)+.005;
  if (fabs(maxLatNorth - minLatSouth) <= .005)
    region.span.latitudeDelta = .01;
  region.span.longitudeDelta =
                      fabs(maxLonWest - minLonEast)+.005;
  if (fabs(maxLonWest - minLonEast) <= .005)
    region.span.longitudeDelta = .01;

  return region;
}
```

Listing a.1: `regionForAnnotationGroup`

Dieser Code berechnet die Region, die beide Anmerkungen enthält. Das geht jetzt zwar über den Rahmen dieses Buches hinaus, aber es ist praktisch, zu wissen, wie es funktioniert. Ich werde daher nur zusammenfassen, was der Code macht. Die einzelnen Schritte überlasse ich Ihnen.

Beide Anmerkungen sollen auf dem Bildschirm zu sehen sein, daher liest der Code jede Anmerkung ein und bestimmt die nördlichste und südlichste geografische Breite sowie die westlichste und östlichste geografische Länge. Der Trick ist dabei, die Funktion `fabs` für den absoluten Wert einer Zahl zu nutzen, da Längen- und Breitengrad auch negativ sein können. Danach muss man nur noch die Mittelwerte bilden, um `center` zu setzen und aus dem Abstand

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

von nördlichster und südlichster Breite beziehungsweise östlichster und westlichster Länge die `span` zu berechnen. (Ich habe die Spanne zudem noch um 0,005 vergrößert, damit die Pins nicht direkt am Rand liegen.) Zudem wollte ich nicht, dass `span` kleiner als 0,005 wird. In diesem Fall setze ich den Wert auf 0,01.

Jetzt müssen wir nur noch die Methode `viewDidLoad` von `MapController` so anpassen, dass sie statt `initialCoordinate` und `updateRegionLatitude::` von `Destination` diese Methode verwendet.

Dazu ergänzen Sie in `viewDidLoad` den fett gedruckten Code und entfernen den durchgestrichenen in Listing a.2.

```
- (void)viewDidLoad {

  [super viewDidLoad];
  mapView.showsUserLocation = YES;
// ~~CLLocationCoordinate2D initialCoordinate =~~
// ~~[destination initialCoordinate];~~
// ~~[self updateRegionLatitude: initialCoordinate.latitude~~
            ~~longitude: initialCoordinate.longitude~~
// ~~latitudeDelta:.2 longitudeDelta:.2];~~
  self.title = [destination mapTitle];
  [mapView.userLocation addObserver:self
           forKeyPath:@"location" options:0 context:NULL];
  ~~[mapView addAnnotations:~~
                          ~~[destination createAnnotations]];~~
  NSArray* annotations = [destination createAnnotations];
  [mapView addAnnotations:annotations];
  [mapView setRegion:[self regionForAnnotationGroup:
           annotations] animated:NO];
}
```

Listing a.2: `viewDidLoad` *zeigt nun die Anmerkungen immer an.*

Sie müssen auch noch `City.h` importieren. Die Ergebnisse sehen Sie in Abbildung a.6.

Jetzt können Sie noch `initialCoordinate` in `Destination.h` und `.m` sowie `updateRegion Latitude` in `MapController.h` und `.m` löschen.

Nachdem Sie nun die Region berechnen lassen, statt sie fest zu verdrahten, sehen Sie sich plötzlich einem weiteren Problem gegenüber: Die Karte soll im Hoch- wie im Querformat angezeigt werden. Öffnet sich die App aber im Querformat, wird die richtige Region dafür berechnet. Drehen Sie das iPhone dann ins Hochformat, sind die Anmerkungen nicht mehr zu sehen.

Genau der richtige Zeitpunkt, um `didRotateFromInterfaceOrientation` vorzustellen. Diese Nachricht wird an den `MapController` geschickt, wenn sich die Ausrichtung des Geräts ändert. Damit das beschriebene Problem verschwindet, müssen Sie glücklicherweise nur den Code aus Listing a.3 in `MapController.m` einbinden.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung a.6: Eine bessere Möglichkeit, einen Bereich zu berechnen

```
- (void)didRotateFromInterfaceOrientation:
        (UIInterfaceOrientation)fromInterfaceOrientation {

  [mapView setRegion:
   [self regionForAnnotationGroup:destination.annotations]
                                              animated:NO];
}
```

Listing a.3: Auf eine Änderung der Ausrichtung Rücksicht nehmen

Die Region muss neu berechnet werden, aber das macht `regionForAnnotationGroup:animated:` ja schon für Sie.

Den aktuellen Standort finden

Sie können zwar die Karte immer so verschieben, dass Sie den Standort des Anwenders sehen, aber das ist recht nervig, wenn Sie nicht gerade in oder um London herum programmieren. Um zumindest diese Bürde von Ihren Schultern zu nehmen, will ich Ihnen zeigen, wie einfach Sie einen Button in der Navigationsleiste hinzufügen können, mit dem Sie zu Ihrem aktuellen Standort und dann zurück zur Karten-Region springen.

1. **Fügen Sie den folgenden Code in der Methode `viewDidLoad` von `MapController` ein, um den Button hinzuzufügen.**

   Da sich dort mittlerweile einiges an Code findet, hier nur das Hinzuzufügende.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
UIBarButtonItem *locateButton =
   [[UIBarButtonItem alloc] initWithTitle: @"Standort"
   style:UIBarButtonItemStylePlain target:self
   action:@selector(goToLocation:)];
self.navigationItem.rightBarButtonItem = locateButton;
[locateButton release];
```

Wenn Ihnen das bekannt vorkommt, liegt es daran, dass Sie schon in Kapitel 16 einen Button eingefügt haben. Tippt der Anwender auf den Button, wird die Nachricht `goTo Location:` (`action:@selector(goToLocation:)`) an den `MapController` (`target:self`) gesendet.

2. Fügen Sie die Methode `goToLocation:` in `MapController.m` ein.

```
- (IBAction)goToLocation:(id)sender{
  MKUserLocation *annotation = mapView.userLocation;
  CLLocation *location = annotation.location;
  if (nil == location)
    return;
  CLLocationDistance distance =
              MAX(4*location.horizontalAccuracy,500);
  MKCoordinateRegion region =
           MKCoordinateRegionMakeWithDistance
           (location.coordinate, distance, distance);
  [mapView setRegion:region animated:NO];

  self.navigationItem.rightBarButtonItem.action =
                                  @selector(goToTrip:);
  self.navigationItem.rightBarButtonItem.title =
                                           @"Karte"
}
```

Tippt der Anwender auf den Button STANDORT, wird zunächst geprüft, ob die aktuelle Position überhaupt verfügbar ist. (Denken Sie daran, dass es nach dem Start der App ein paar Sekunden dauern kann, bis der Standort gefunden wurde.) Wenn nicht, kehren Sie einfach zurück. (Sie könnten natürlich auch eine Warnung ausgeben und den Anwender bitten, es in zehn Sekunden oder so erneut zu versuchen.)

Ist der Standort gefunden, berechnen Sie die Spanne für die Region, die Sie anzeigen wollen. In diesem Fall soll die Spanne vier Mal so groß sein wie `horizontalAccuracy` des Geräts (aber nicht weniger als 1000 Meter).

```
CLLocationDistance distance =
            MAX(4*location.horizontalAccuracy,1000);
```

`horizontalAccuracy` ist ein Unsicherheitsradius, der von der Genauigkeit des Geräts abhängt. Irgendwo in diesem Kreis befindet sich der Anwender.

Dann rufen Sie die Funktion `MKCoordinateRegionMakeWithDistance` auf, die aus den angegebenen Koordinaten und dem Abstand eine neue `MKCoordinateRegion` erstellt. `distance` und `distance` entsprechen `latitudinalMeters` und `longitudinalMeters`.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Soll die Spanne nicht geändert werden, können Sie auch einfach die Eigenschaft `centerCoordinate` der Map View auf die `userLocation` setzen. Wie weiter vorn in *Die Region* beschrieben, wird damit die Region neu zentriert, die Spanne aber beibehalten.

3. **Ändern Sie den Titel des Buttons in KARTE und den Selektor in `(goToTrip:)`, womit der Button beim nächsten Antippen die Nachricht `goToTrip:` verschickt. Daher brauchen Sie noch folgenden Code:**

```
- (IBAction) goToTrip:(id)sender{

  [mapView setRegion:[self regionForAnnotationGroup:
        destination.annotations] animated:NO];
  self.navigationItem.rightBarButtonItem.title =
                                          @"Standort";
  self.navigationItem.rightBarButtonItem.action =
                              @selector(goToLocation:);
}
```

Es gibt hier allerdings ein Problem. In Listing a.3 setze ich die Region beim Drehen des Geräts ins Querformat mit der Methode `regionForAnnotationGroup:`. Hier wird aber die Region um Heathrow und London angezeigt.

Ich will also beim Anzeigen des Benutzer-Standorts nicht diese Region anpassen, sondern der View Controller soll seinen Standard-Vorgaben folgen. Auch wenn Sie eine Instanzvariable nutzen können, um sich zu merken, was gerade angezeigt wird, gehe ich hier den einfachen Weg (siehe Listing a.4) und prüfe, wie der Button gerade beschriftet ist.

```
- (void)didRotateFromInterfaceOrientation:
        (UIInterfaceOrientation)fromInterfaceOrientation {

  if (![self.navigationItem.rightBarButtonItem.title
                              isEqualToString: @"Karte"])
    [mapView setRegion:
    [self regionForAnnotationGroup:destination.annotations]
                                          animated:NO];
}
```

Listing a.4: Auf das Drehen des Geräts reagieren

Tippen Sie nun auf den Button STANDORT, erhalten Sie die Anzeige aus Abbildung a.7. (Ah, Infinite Loop. Apples trautes Heim.)

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung a.7: Den aktuellen Standort anzeigen

Geokodierung

Es ist zwar gut und schön, zu sehen, wo man sich gerade befindet, aber ich würde schon gerne die genaue Adresse wissen. (Dann könnte ich auch Code schreiben, der die aktuelle Adresse in einen Adressbuch-Eintrag umwandelt, aber diese Aufgabe möchte ich gerne Ihnen überlassen.)

Die »Umwandlung« einer Koordinate auf einer Karte in eine Adresse nennt man *umgekehrte Geokodierung*, was netterweise vom `MapKit` angeboten wird. Die normale *Geocodierung* – also das Umwandeln einer Adresse in eine Koordinate – wird durch das `MapKit` nicht unterstützt, allerdings gibt es eine ganze Reihe kommerzieller und kostenloser Dienste, die das anbieten.

Der Standort muss nicht ganz exakt bekannt sein – Sie erinnern sich bestimmt noch an `horizontalAccuracy` im Abschnitt *Den aktuellen Standort finden*. Da sich mein Büro zum Beispiel ganz nah an der Grundstücksgrenze befindet, ist mein Standort manchmal der meines Nachbarn.

Ein Einbau der umgekehrten Geokodierung in iPhoneTravel411 ermöglicht es Ihnen, die Adresse der aktuellen Position anzuzeigen. Dazu sind folgende Schritte notwendig:

1. **Importieren Sie das Reverse Geocoder Framework in `MapController.h` und lassen Sie `MapController` das Protokoll `MKReverseGeocoderDelegate` übernehmen.**

```
#import <MapKit/MKReverseGeocoder.h>

@interface MapController : UIViewController
      <MKMapViewDelegate, MKReverseGeocoderDelegate> {
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

2. **Fügen Sie eine Instanzvariable für die Referenz auf das Geocoder-Objekt ein.**

```
MKReverseGeocoder *reverseGeocoder;
```

Sie werden diese Referenz später nutzen, um `MKReverseGeocoder` nach dem Ermitteln der aktuellen Adresse wieder freizugeben.

3. **Fügen Sie die Methoden `reverseGeocoder:didFindPlacemark:` und `reverseGeocoder:didFailWithError:` in `MapController.m` ein.**

```
- (void)reverseGeocoder:(MKReverseGeocoder *) geocoder
          didFindPlacemark:(MKPlacemark *) placemark {

  NSMutableString* addressString;
  if ([selectedAnnotation
               isKindOfClass:[MKUserLocation class]]){
    if (placemark.subThoroughfare!= NULL) {
      addressString = [[NSMutableString alloc]
           initWithString: placemark.subThoroughfare];
      [addressString appendString: @" "];
      [addressString appendString:
                               placemark.thoroughfare];
      mapView.userLocation.subtitle =
                                     placemark.locality;
    }
    else {
      addressString = [[NSMutableString alloc]
                  initWithString: placemark.locality];
      mapView.userLocation.subtitle =
                          placemark.administrativeArea;
    }
    selectedAnnotation.title = addressString;
    [addressString release];
  }
}


- (void)reverseGeocoder:(MKReverseGeocoder *) geocoder
        didFailWithError:(NSError *) error{

  NSLog(@"Reverse Geocoder Errored");
}
```

Die Nachricht `reverseGeocoder:didFindPlacemark:` wird an das Delegate geschickt, wenn das Objekt `MKReverseGeocoder` erfolgreich eine *Placemark*-Information für eine Koordinate erhalten hat. Ein `MKPlacemark`-Objekt enthält Ortsinformationen für eine gegebene geografische Breite und Länge. Placemark-Daten besitzen unter anderem Eigenschaften zum Land, zur Stadt und zur Straße/Hausnummer. (Es gibt noch eine Reihe weiterer Daten, die Sie sich einmal anschauen können.)

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

- `country`: Landesname
- `administrativeArea`: Untergeordnete Einheit, zum Beispiel der Bundesstaat in den USA
- `locality`: Stadt
- `thoroughfare`: Straße
- `subThoroughfare`: Zusätzliche Informationen, zum Beispiel die Hausnummer
- `postalCode`: Postleitzahl

In dieser Implementierung setzen Sie den Text der Anmerkung (`userLocation`) auf einen selbst erstellten String, bestehend aus `subThoroughfare` und `thoroughfare` (also Straße und Hausnummer). Dem Subtitel wird die Eigenschaft `locality` (die Stadt) zugewiesen.

Ich habe hier ein wenig Fehlerhandling betrieben. Bei

```
Placemark.subThoroughfare != NULL
```

gehe ich davon aus, dass es keine Straßenangabe gibt. Dann setze ich den Titel auf die Stadt und den Subtitel auf den Bundesstaat.

Beachten Sie auch, dass ich prüfe, ob es sich bei der Anmerkung um ein `MKUserLocation`-Objekt handelt.

```
if ([selectedAnnotation
                isKindOfClass:[MKUserLocation class]]){
```

Dabei prüfe ich die Instanzvariable `selectedAnnotation`, die ich in diesem Abschnitt definiert und gesetzt habe.

Sie fragen sich vielleicht, warum ich so viel Zeit mit der Geokodierung verbringe, denn bisher geschieht sie nur für den Standort des Benutzers. Aber wie Sie gleich sehen werden, werde ich auch den Standort ermitteln, wenn der Anwender den Pin auf der Karte verschiebt.

Ein Placemark ist ebenfalls eine Anmerkung, die das Protokoll `MKAnnotation` umsetzt. Dessen Eigenschaften und Methoden beinhalten die Koordinaten des Placemarks und andere Informationen. Da es sich um Anmerkungen handelt, können Sie sie direkt der Map View hinzufügen.

Die Nachricht `reverseGeocoder:didFailWithError:` wird an das Delegate geschickt, wenn der `MKReverseGeocoder` für die angegebenen Koordinaten keine Placemark-Informationen ermitteln konnte. (Das ist eine Methode, die für das Protokoll `MKReverseGeocoderDelegate` erforderlich ist.)

Um die umgekehrt geokodierten Informationen zu erhalten, brauchen Sie natürlich ein Objekt vom Typ `MKReverseGeocoder`. Machen Sie `MapController` zu seinem Delegate, schicken Sie dem Geocoder eine Nachricht `start` und geben Sie ihn wieder frei, wenn Sie fertig sind.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

1. **Machen Sie den `MapController` zu einem Delegate von `MKReverseGeocoder`, indem Sie den fett gedruckten Code in `MapController.h` einfügen.**

```
@interface MapController : UIViewController
      <MKMapViewDelegate, MKReverseGeocoderDelegate> {
```

2. **Allokieren und starten Sie den Reverse Geocoder und fügen Sie in der Methode `goTo Location:` den `MapController` als sein Delegate mit dem fett gedruckten Code ein.**

```
- (IBAction)goToLocation:(id) sender{

  MKUserLocation *annotation = mapView.userLocation;
  CLLocation *location = annotation.location;
  if (nil == location)
    return;
  CLLocationDistance distance =
              MAX(4*location.horizontalAccuracy, 500);
  MKCoordinateRegion region =
             MKCoordinateRegionMakeWithDistance
            (location.coordinate, distance, distance);
  [mapView setRegion:region animated:NO];
  self.navigationItem.rightBarButtonItem.action =
                                  @selector(goToTrip:);
  self.navigationItem.rightBarButtonItem.title =
                                           @"Karte";
  selectedAnnotation = mapView.userLocation;
  reverseGeocoder = [[MKReverseGeocoder alloc]
              initWithCoordinate:location.coordinate];
  reverseGeocoder.delegate = self;
  [reverseGeocoder start];
}
```

Beachten Sie, wie der `MKReverseGeocoder` mit den Koordinaten der aktuellen Position initialisiert wird. Sie brauchen auch noch eine neue Instanzvariable `MKUserLocation *selectAnnotation`, die ich im Geokodierer nutze und hier zuweise.

```
selectedAnnotation = mapView.userLocation;
```

3. **Geben Sie den `MKReverseGeocoder` frei, indem Sie den fett gedruckten Code in `goToTrip:` hinzufügen.**

```
- (IBAction)goToTrip:(id) sender{

  [reverseGeocoder release];
  [mapView setRegion:[self regionForAnnotationGroup:
        destination.annotations] animated:NO];
  self.navigationItem.rightBarButtonItem.title =
                                          @"Standort";
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
    self.navigationItem.rightBarButtonItem.action =
                                @selector(goToLocation:);
}
```

Sie geben den `MKReverseGeocoder` in dieser Methode frei, denn obwohl Sie ihn in der Methode `goToLocation:` gestartet haben, liefert er seine Informationen dort nicht zurück. Er arbeitet asynchron – wenn er die Informationen ermitteln konnte oder aufgibt, schickt er die Nachricht `reverseGeocoder:didFindPlacemark:` beziehungsweise `reverseGeocoder:didFailWithError:`. Kehren Sie zur ursprünglichen Map View zurück, ist es Ihnen aber egal, ob er noch erfolgreich sein wird oder nicht, daher können Sie `MKReverseGeocoder` freigeben.

Abbildung a.8 zeigt das Ergebnis Ihrer Abenteuer im Land des Geokodierens.

Abbildung a.8: Umgekehrtes Geokodieren

Aber wenn ich doch gar nicht nach London will?

Wie Sie sicherlich bemerkt haben, sind alle Koordinaten und Standorte in iPhoneTravel411 hart kodiert. Aber in Ihren eigenen Apps wollen Sie dem Anwender sicherlich die Kontrolle darüber geben, was angezeigt wird.

In dieser App könnte man argumentieren, dass das kein wirkliches Problem ist – schließlich handelt es sich um einen Reiseführer für London. Aber in Wirklichkeit ist London groß und der Anwender will das Ziel vielleicht etwas genauer auswählen.

Sie könnten zwar einen modalen Dialog erstellen, der es dem Anwender erlaubt, eine Position auf der Karte hinzuzufügen oder zu ändern, aber das geht über den Rahmen dieses Buches hinaus. Ich zeige Ihnen aber, wie Sie eine verschiebbare Anmerkung erstellen – in diesem Fall die für die `City` – ‚damit der Anwender sein Ziel genau dorthin verschieben kann, wo er hinmöchte.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Das Vorgehen ist recht einfach. Sie setzen nur eine Eigenschaft der Anmerkung. Dazu brauchen Sie natürlich Zugriff auf die Anmerkungs-View. Statt also die Standard-View zu verwenden, wie Sie es bisher getan haben, zeige ich Ihnen, wie Sie eine vom SDK bereitgestellte Anmerkungs-View nutzen – `MKPinAnnotationView`. Zusätzlich werden Sie die Farbe des Pins verändern und ihn auf die Karte »fallen« lassen können.

Fügen Sie einfach den Code aus Listing a.5 in `MapController.m` ein. `mapView:viewForAnnotation:` ist eine Delegate-Methode von `MKMapView`, die automatisch aufgerufen wird, bevor die Map View die Anmerkung anzeigt. Dort haben Sie die Chance, die View entsprechend anzupassen.

```
- (MKAnnotationView *)mapView:(MKMapView *)aMapView
          viewForAnnotation:(id <MKAnnotation>)annotation {

  if ([annotation isKindOfClass:[MKUserLocation class]])
    return nil;
  MKPinAnnotationView* pinView = (MKPinAnnotationView*)
    [mapView dequeueReusableAnnotationViewWithIdentifier:
                              @"CustomPinAnnotationView"];
  if (!pinView) {
    pinView = [[[MKPinAnnotationView alloc]
          initWithAnnotation:annotation
          reuseIdentifier:@"CustomPinAnnotation"]
                                          autorelease];
    if ([annotation isKindOfClass:[City class]]) {
      pinView.pinColor = MKPinAnnotationColorRed;
      pinView.draggable =YES;
    }
    else
      pinView.pinColor = MKPinAnnotationColorGreen;
    pinView.animatesDrop = YES;
    pinView.canShowCallout = YES;
  }
  else
    pinView.annotation = annotation;
  return pinView;
}
```

Listing a.5: `mapView:viewForAnnotation:`

Zunächst prüfe ich, ob die Anmerkung eine `MKUserLocation` ist, und nutze dann die eingebaute View.

```
if ([annotation isKindOfClass:[MKUserLocation class]])
  return nil;
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Dann prüfe ich, ob es eine vorhandene View gibt, die ich nutzen kann. Das ist der gleiche Mechanismus, mit dem ich Tabellen-Felder in Kapitel 15 wiederverwende. Ist keine View verfügbar, erzeuge ich eine und initialisiere sie mit der Anmerkung.

```
if (!pinView) {
  pinView = [[[MKPinAnnotationView alloc]
    initWithAnnotation:annotation
    reuseIdentifier:@"CustomPinAnnotation"] autorelease];
```

Ist die Anmerkung ein Objekt vom Typ `City`, setze ich die Farbe des Pins auf Rot – das ist meine definierte Farbe für ein Ziel. Zudem sorge ich dafür, dass der Pin verschiebbar ist. Dazu setze ich die Eigenschaft `draggable` auf `YES` (der Standardwert ist `NO`), allerdings muss das Anmerkungs-Objekt dann noch die Methode `setCoordinate:` implementieren. Schauen Sie sich den Code für das `City`-Modell auf meiner Website (`www.nealgoldstein.com`) an, sehen Sie, dass ich die Koordinaten als Eigenschaft definiert und per `@synthesize` »implementiert« habe.

Ist es kein `City`-Objekt – sondern zum Beispiel ein Flughafen –, setze ich die Pin-Farbe auf Grün. Ich will nicht, dass diese Pins verschiebbar sind, daher kümmere ich mich auch nicht um die Eigenschaft `draggable`.

Dieser Code ist dafür notwendig:

```
if ([annotation isKindOfClass:[City class]]) {
  pinView.pinColor = MKPinAnnotationColorRed;
  pinView.draggable =YES;
}
else
  pinView.pinColor = MKPinAnnotationColorGreen;
```

Schließlich sorge ich dafür, dass der Pin elegant auf die Karte fällt und er in einem Fensterchen Text anzeigen kann. Beides setze ich über die Pin-View.

```
pinView.animateDrop = YES;
pinView.canShowCallout = YES;
```

Handelte es sich um eine wiederverwendbare Pin-View, weise ich die Anmerkung zu. Immer gebe ich danach die Pin-View zurück.

```
else
  pinView.annotation = annotation;
return pinView;
```

Bewegt der Anwender den Pin, wäre es natürlich schön, auch Titel und Subtitel an das neue Ziel anzupassen. Glücklicherweise gibt es eine weitere Delegate-Methode, die mir die Chance dazu gibt – `mapView:annotationView:didChangeDragState:fromOldState:`.

Listing a.6 zeigt den Code, den Sie in `mapController.m` einfügen müssen, um die Änderungen von Titel und Subtitel zu erreichen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
- (void)mapView:(MKMapView *)mapView annotationView:
   (MKAnnotationView *)annotationView
   didChangeDragState:(MKAnnotationViewDragState)newState
        fromOldState:(MKAnnotationViewDragState)oldState {

  if (newState == MKAnnotationViewDragStateEnding) {
    reverseGeocoder = [[MKReverseGeocoder alloc]
           initWithCoordinate:
                    annotationView.annotation.coordinate];
    selectedAnnotation = annotationView.annotation;
    reverseGeocoder.delegate = self;
    [reverseGeocoder start];
  }
}
```

Listing a.6: `mapView:annotationView:didChangeDragState:fromOldState:`

Die Nachricht `mapView:annotationView:didChangeDragState:fromOldState:` wird fortlaufend an das Delegate geschickt, während der Anwender die Anmerkung hin und her zieht. Man wird zwar kontinuierlich informiert, aber eigentlich ist es mir egal, wohin der Anwender gerade zieht, bis er den Finger vom Bildschirm genommen hat.

```
If (newState == MKAnnotationViewDragStateEnding) {
```

Nach dem Ziehen merke ich mir dann doch die Anmerkung in `selectedAnnotation` und erzeuge einen Geocoder, um etwas über die neue Position herauszufinden.

```
selectedAnnotation = annotationView.annotation;
reverseGeocoder.delegate = self;
[reverseGeocoder start];
```

Wie Sie sich erinnern werden, aktualisieren Sie in `reverseGeocoder:didFindPlacemark:` Titel und Subtitel der `MKUserLocation`-Anmerkung durch die Placemark-Informationen. Ich muss jetzt nur noch den fett gedruckten Code in Listing a.7 einfügen, um eine `City`-Anmerkung zu aktualisieren.

```
- (void)reverseGeocoder:(MKReverseGeocoder *) geocoder
          didFindPlacemark:(MKPlacemark *) placemark {
  NSMutableString* addressString;
  if ([selectedAnnotation
              isKindOfClass:[MKUserLocation class]]){
    if (placemark.subThoroughfare!= NULL) {
      addressString = [[NSMutableString alloc]
           initWithString: placemark.subThoroughfare];
      [addressString appendString: @" "];
      [addressString appendString:
                              placemark.thoroughfare];
      mapView.userLocation.subtitle =
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
                                    placemark.locality;
    }
    else {
      addressString = [[NSMutableString alloc]
                  initWithString: placemark.locality];
      mapView.userLocation.subtitle =
                          placemark.administrativeArea;
    }
    selectedAnnotation.title = addressString;
  }
  else {
    if (placemark.locality!= NULL)
      addressString = [[NSMutableString alloc]
                  initWithString: placemark.locality];
    else
      addressString = [[NSMutableString alloc]
        initWithString: placemark.administrativeArea];
      [(City*)selectedAnnotation
                            setCityName:addressString];
    }
  [addressString release];
// }
}
```

Listing a.7: `reverseGeocoder:didFindPlacemark:` *ergänzen*

Um den Namen der Stadt zu aktualisieren, verschicke ich die Nachricht `setCityName:`, die von der `@synthesize`-Anweisung erzeugt wurde.

Hier könnte ich natürlich auch die Punkt-Notation verwenden, aber durch Verwenden der Methode wird das Casten einfacher.

Wie Sie in Abbildung a.9 sehen, habe ich den Pin nach Kensington gezogen, wo ich plane, mich mit meinem Freund zu treffen.

Was kommt jetzt?

Sie haben schon eine ziemlich robuste App. Aber ich will mir ja nicht nachsagen lassen, ich ließe Sie im Regen stehen. Daher zeige ich Ihnen noch mehr standortbasierte Features – in Kapitel B, das Sie sich ebenfalls von unserer Webseite `http://www.wiley-vch.de/publish/dt/books/ISBN978-3-527-70729-4` herunterladen können.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung a.9: Ein Treffen mit Freunden

Standort, Standort, Standort

B

In diesem Kapitel …

- Nutzen Sie das Core Location Framework
- Finden Sie den aktuellen Standort des iPhones heraus
- Lassen Sie die App im Hintergrund laufen
- Nutzen Sie den Significant-Change Location Service
- Beobachten Sie Regions-Änderungen

Wenn ich in Heathrow lande, bin ich mit Sicherheit müde und reizbar. Ich will einfach nur in mein Hotel, mich duschen und ein Nickerchen halten. Was mich da noch wachhält, ist das Wissen um die Entfernung zum Hotel. Ich fände es sogar interessant, zu sehen, wie sich der Abstand zu meinem Hotelbett verringert. Da ich den Paddington Express nutzen werde, muss ich auch noch ein Taxi oder die U-Bahn nehmen. Nutze ich ein Taxi, wäre es nett, zu sehen, wie ich mich nähere, statt eine Stadtrundfahrt zu machen.

Außerdem möchte ich gerne, dass mich iPhoneTravel411 auch dann mit Informationen versorgt, wenn es im Hintergrund läuft, damit ich währenddessen andere Dinge mit meinem iPhone anstellen kann (zum Beispiel das coole Spiel zu spielen, das ich vorher noch entwickelt hatte, oder meine E-Mails abzurufen).

Das ist alles im iOS 4 möglich. In diesem Kapitel erläutere ich, wie man die Hintergrundverarbeitung, Core Location und lokale Benachrichtigungen nutzt, um meine Wünsche zu erfüllen. Mit iOS 4 bieten sich Ihnen eine Reihe von Alternativen rund um Ihren Standort, die sich unterschiedlich stark auf die Akku-Laufzeit auswirken. Ich möchte Ihnen diese Alternativen vorstellen. Ich zeige Ihnen sogar ein Feature in iOS 4, mit dem Sie Ihre Benachrichtigungen auf eine bestimmte Region einschränken können.

Das ist alles ziemlich leicht zu erreichen, aber die Regeln, an die Sie sich bei einem Ausführen Ihrer App im Hintergrund halten müssen – und die Auswahlmöglichkeiten, die Sie haben –, zwingen Sie dazu, auch auf die Details zu achten.

Ich will mit Core Location beginnen und Ihnen dann zeigen, wie Sie Ihre App im Hintergrund laufen lassen können.

Sich des Standorts bewusst sein

Es gibt eine Klasse von Anwendungen (zum Thema Reisen und soziale Netzwerke), bei denen das Wissen um den geografischen Standort des Anwenders die Qualität der angebotenen Informationen verbessern kann. Und schließlich geht es bei mobilen Geräten wie dem iPhone ja auch um Mobilität. standortbezogene Dienste unterstützen das, indem sie Informationen darüber liefern, wo sich das Gerät gerade befindet.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

In Kapitel A habe ich erklärt, dass die Location Services aus zwei Teilen bestehen: Kartendarstellung (siehe das letzte Kapitel) und Core Location (das wird in diesem Kapitel erklärt).

Aber zunächst ein bisschen Hintergrundinformationen.

Das iPhone besitzt drei Technologien, um den Standort zu ermitteln:

1. Funkzellen
2. WLAN
3. GPS

Diese Technologien sind nach Genauigkeit und Stromverbrauch geordnet, wobei die Erkennung über Funkzellen am ungenauesten und am stromsparendsten ist.

Die Standortermittlung über Funkzellen kann den Benutzer auch ohne eine Internetverbindung lokalisieren und hat eine Genauigkeit zwischen 0,5 und 50 km. Je mehr Zellen es gibt, desto genauer ist die Erkennung – daher funktioniert das am besten in Ballungsräumen. Diese Erkennung lässt sich in allen Geräten mit einer Mobilfunkverbindung nutzen.

WLAN funktioniert für alle Geräte. Es basiert darauf, dass die Position von WLAN-Hotspots bekannt ist, zudem gibt es Caching-Techniken, die genutzt werden, wenn keine Internetverbindung vorhanden ist. Die Genauigkeit liegt häufig im Bereich von 100 Metern.

Schließlich gibt es die GPS-Erkennung, die auf den GPS-Satelliten basiert. Dazu gehören noch weitere Techniken, die die Genauigkeit verbessern können. Sie hängt von einer Reihe von Faktoren ab, Sie können aber von etwa acht Metern ausgehen.

Auch wenn der Anwender das Roaming abgeschaltet hat, kann er die Location Services weiterhin nutzen.

Unabhängig von der verwendeten Technologie brauchen Sie nur mit einer einzigen API zu arbeiten – mit den Klassen des Core Location Frameworks. Sie müssen sich also im Allgemeinen keine Gedanken darüber machen, welche Technologie(n) das Gerät verwendet.

Core Location stellt drei verschiedene Dienste bereit, die Sie nutzen können, um den aktuellen Standort des Geräts zu erkennen und zu verfolgen:

- ✔ **Die Standard Location Services:** Dieser Dienst ist sehr bequem nutzbar, zudem ist es der genaueste Dienst, der allerdings auch am meisten Strom verbraucht. Er wird in allen Versionen des iPhone OS unterstützt.
- ✔ **Die Significant-Change Location Services:** Dieser Dienst kam mit iOS 4 hinzu. Er nutzt die Positionsdaten von Funkzellen, um Sie über größere Änderungen der Position zu informieren. (Ich gehe später noch darauf ein, was das bedeutet.) Er ist weniger genau, braucht aber auch weniger Strom als der Standard Location Service.
- ✔ **Region Monitoring:** Auf dem iPhone 4 (und neueren Geräten) können Sie mit diesem Dienst darauf achten, ob der Anwender einen selbstdefinierten Bereich betritt oder verlässt.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Um die Features des Core Location Frameworks zu nutzen, müssen Sie `CoreLocation.framework` zu Ihrem Xcode-Projekt hinzufügen (so wie Sie es mit `MapKit.framework` gemacht haben) und eine Anweisung `#import <CoreLocation/CoreLocation.h>` in allen Dateien aufnehmen, die die Core-Location-Klassen verwenden. Ihr `iPhoneTravel411AppDelegate` wird zudem ein Core Location Delegate werden müssen, indem es das Protokoll `CLLocationManagerDelegate` umsetzt.

Ich lasse Sie die ganze Standort-Arbeit in `iPhoneTravel411AppDelegate` machen, daher müssen Sie die fett gedruckten Anweisungen in `iPhoneTravel411AppDelegate.h` einfügen:

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

@interface iPhoneTravel411AppDelegate : NSObject
      <UIApplicationDelegate, CLLocationManagerDelegate> {
```

Sie machen die ganze Arbeit in `iPhoneTravel411AppDelegate`, damit Sie selbst Core-Location-Dienste in Ihrer App anbieten können. Ich hätte zugegebenermaßen auch eine eigene Klasse dafür anlegen und die Funktionalität darin kapseln können. Da ich Core Location in meinen eigenen Apps recht ausgiebig nutze, habe ich eine Klasse erstellt, die ich dort verwende.

Das Einholen von Standort-Daten kostet viel Strom. Wie viel? Nun, darauf komme ich noch im Abschnitt *Den Significant-Change Location Service nutzen, um den Akku zu schonen* weiter hinten.

Den aktuellen Standort des Anwenders über die Standard Location Services ermitteln

Zunächst müssen Sie herausbekommen, ob die Location Services verfügbar sind. Auch wenn jedes iOS-Gerät Location Services unterstützt, gibt es immer noch Situationen, in denen sie nicht nutzbar sind, zum Beispiel:

- ✔ Der Anwender kann die Location Services in der Einstellungs-App deaktivieren.
- ✔ Der Anwender kann die Verwendung von Location Services für eine bestimmte App ablehnen.
- ✔ Das Gerät befindet sich im Flugzeug-Modus.
- ✔ Bei Geräten ohne GPS gibt es eventuell keine Funkzellen oder WLAN-Hotspots in der Nähe.

Sie müssen die Möglichkeiten in Ihrer App berücksichtigen. Ich erkläre Ihnen hier einiges, um den Start zu erleichtern, aber für eine nahtlose Benutzerfreundlichkeit müssen Sie noch mehr investieren.

Zeit für einen Überblick: Um iPhoneTravel411 mit Location Services zu ergänzen, müssen Sie drei Dinge tun:

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

1. Den Location Manager starten
2. Auf Standort-Änderungen reagieren
3. Fehler abfangen

Die nächsten Abschnitte gehen auf die entsprechenden Details ein.

Den Location Manager starten

Als Erstes starten Sie die Location Services. Das müssen Sie an zwei Stellen machen – `application:didFinishLaunchingWithOptions` und `setDefaults:`. Fügen Sie den fett gedruckten Code in Listing b.1 und Listing b.2 in die entsprechenden Methoden in `iPhoneTravel411AppDelegate.m` ein.

```
- (BOOL)application:(UIApplication *)
   application didFinishLaunchingWithOptions:
                            (NSDictionary *)launchOptions {

  [[NSNotificationCenter defaultCenter] addObserver:self
     selector:@selector(setDefaults:)
     name:NSUserDefaultsDidChangeNotification object:nil];

  if (![[NSUserDefaults standardUserDefaults]
        objectForKey:kUseStoredDataPreference]) {
    useStoredData = NO;
    monitorLocationChanges = YES;
    [self startLocationUpdates:monitorLocationChanges];
  }
  else
    [self setDefaults:nil];
...
}
```

Listing b.1: Den Location Manager im App-Delegate starten

```
- (void)setDefaults:(NSNotification*)notification {

  useStoredData = [[NSUserDefaults standardUserDefaults]
          boolForKey:kUseStoredDataPreference];
  monitorLocationChanges = [[NSUserDefaults
          standardUserDefaults]
          boolForKey:kMonitorLocationPreference];
  [self startLocationUpdates:monitorLocationChanges];
}
```

Listing b.2: Auf die Änderung von Einstellungen reagieren

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Nun zu `setDefaults`. Wie Sie sich vielleicht aus Kapitel 16 erinnern, prüfen Sie beim Starten der App, ob Einstellungen vorhanden sind. Wenn nicht, schicken Sie sich selbst die Nachricht `startLocationUpdates:` mit dem Argument `YES` (dem Standardwert).

In diesem Fall haben Sie eine Benachrichtigung erhalten, dass sich die Einstellungen geändert haben (oder dass die App gestartet wurde und Einstellungen vorhanden sind), und Sie übergeben diesen Status an `startLocationUpdates:`.

Es scheint, in `startLocationUpdates:` spielt die Musik, also wollen wir es uns genauer anschauen. Nachdem Sie das Core Location Framework und die Import-Anweisungen hinzugefügt haben, tragen Sie noch den Code aus Listing b.3 in `iPhoneTravel411AppDelegate.m` nach.

```
- (void)startLocationUpdates:(BOOL) startUpdates {

  if (startUpdates) {
    if (nil == locationManager)
      locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.purpose = @"iPhoneTravel411 prüft,
        wie weit Sie vom Ziel entfernt sind.";
    locationManager.desiredAccuracy =
                          kCLLocationAccuracyKilometer;
    locationManager.distanceFilter = 500;
    [locationManager startUpdatingLocation];
  }
  else {
    [locationManager stopUpdatingLocation];
  }
}
```

Listing b.3: Starten des Standard Service

Wenn Sie sich den Code anschauen, sehen Sie, dass ich zunächst prüfe, ob ich den Location Manager starten oder beenden soll.

```
if (startUpdates) {
```

Ich prüfe hier nicht, ob die Location Services überhaupt zur Verfügung stehen. Dazu würde ich die Nachricht `locationServicesEnabled` an die Klasse `CLLocationManager` schicken. (In iPhone OS 3.x und älter prüfen Sie stattdessen der Wert der Eigenschaft `locationServicesEnabled`.) Sind sie nicht verfügbar, versuchen Sie aber trotzdem, sie zu starten, fragt das System den Anwender, ob die Location Services wieder aktiviert werden sollen. Das kann den Anwender zwar nerven, wenn er die Dienste bewusst abgeschaltet hat. Andererseits wird es so für den Anwender leichter, sie wieder zu aktivieren.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Nun prüfe ich, ob schon eine Instanz vorhanden ist. Wenn nicht, erzeuge ich eine.

```
if (nil == locationManager)
     locationManager = [[CLLocationManager alloc] init];
```

Als Nächstes weise ich mich selbst als Delegate zu. Das Delegate empfängt dann die Standort-Aktualisierungen (was ich Ihnen als Nächstes zeige).

```
locationManager.delegate = self;
```

Um höflich zu sein, erkläre ich dem Anwender, warum ich seine Position verfolgen möchte. Dieser Zusatztext wird bei der Frage mit angezeigt, ob es in Ordnung ist, dass die App den Standort ermittelt. Sie sehen diese Nachricht in Abbildung b.1. (Allerdings erhalten Sie sie nur auf dem Gerät.)

```
locationManager.purpose = @"iPhoneTravel411 prüft,
    wie weit Sie vom Ziel entfernt sind.";
```

Nun setze ich die gewünschte Genauigkeit und einen Abstandsfilter:

```
locationManager.desiredAccuracy =
                              kCLLocationAccuracyKilometer;
locationManager.distanceFilter = 500;
```

Für die Genauigkeit lassen sich verschiedene Konstanten nutzen. Die genaueste ist `kCLLocationAccuracyBestForNavigation`, die zusätzliche Sensordaten auswertet. Diese Genauigkeit wird aber nur für Apps gebraucht, die jederzeit exakte Positionsinformationen benötigen, und sie sollte nur genutzt werden, wenn das Gerät an einer Stromquelle hängt. Wenn das nicht der Fall ist, sollten Sie `kCLLocationAccuracyBest` nutzen.

Auch wenn der Empfänger sein Bestes gibt, die gewünschte Genauigkeit zu erreichen, kann er sie nicht garantieren.

Nutzen Sie die Genauigkeit, die für Ihre App sinnvoll ist. Je mehr Genauigkeit Sie brauchen, desto mehr Strom wird auch benötigt und desto kürzer ist die Akkulaufzeit.

Auch wenn Sie sehr genaue Daten anfordern, kann die erste Ermittlung der Position eine schlechtere Genauigkeit besitzen. Location Services versucht, die erste Standort-Bestimmung so schnell wie möglich an die App zu liefern und dann Aktualisierungen zu liefern, wenn bessere Daten ermittelt werden.

Der Abstandfilter legt einen minimalen Abstand (gemessen in Metern) fest, den sich ein Gerät bewegen muss, bevor ein neues Update-Ereignis erzeugt wird. Achten Sie auch auf diesen Wert. Spielen Sie mit ihm in Ihrer App herum und setzen ihn auf einen zu kleinen Wert, werden Sie bald sehen, wie nervig zu häufige Updates sein können.

Wollen Sie bei jedem einzelnen Schritt des Anwenders informiert werden, nutzen Sie `kCLDistanceFilterNone`. Das ist auch der Standardwert.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Beide Eigenschaften werden nur von den Standard Location Services verwendet, nicht von den Significant-Change Location Services (auf die ich gleich noch komme).

Schließlich weisen Sie den Location Manager noch an, mit dem Ermitteln der Position zu beginnen:

```
[locationManager startUpdatingLocation];
```

Ist der übergebene Wert `NO`, schalten Sie den Location Manager ab:

```
[locationManager stopUpdatingLocation];
```

Sie müssen in `iPhoneTravel411AppDelegate.h` auch noch die folgenden beiden Zeilen einfügen, um die Instanzvariable und eine Methode zu deklarieren. Die erste Zeile sollte innerhalb der geschweiften Klammern bei den anderen Instanzvariablen stehen, die zweite Zeile bei den Deklarationen von Methoden und Eigenschaften – zwischen der schließenden geschweiften Klammer und `@end`.

```
  CLLocationManager *locationManager;
- (void)startLocationUpdates: (BOOL) startUpdates;
```

Standort-Daten von einem Service erhalten

Der Location Manager schickt Nachrichten an Ihr Delegate, wenn er aktuellere Informationen erhält. Ob Sie nun den Standard Location Service oder den Significant-Change Location Service nutzen (den ich Ihnen gleich noch zeige) – die Ereignisse empfangen Sie auf die gleiche Art und Weise. Immer wenn ein neues Ereignis zur Verfügung steht, schickt der Location Manager die Nachricht `locationManager:didUpdateToLocation:fromLocation:` an sein Delegate. Gibt es beim Empfangen eines Ereignisses einen Fehler, schickt der Location Manager stattdessen die Nachricht `locationManager:didFailWithError:`.

Sie können auch Richtungsinformationen von einem Gerät bekommen, das einen Kompass eingebaut hat, indem Sie die Nachricht `startUpdatingHeading` an den Location Manager schicken. Richtungs-Ereignisse werden über die Methode `locationManager:didUpdateHeading:` an Ihr Delegate geliefert. Gibt es hier einen Fehler, ruft der Location Manager die Methode `locationManager:didFailWithError:` für Ihr Delegate auf.

Listing b.4 zeigt die Delegate-Methode `locationManager:didUpdateToLocation:fromLocation:`. Da der Location Manager manchmal zwischengespeicherte Ereignisse liefert, schlägt Apple vor, dass Sie auch den Zeitstempel von Standort-Ereignissen überprüfen. In diesem Code passiert das. Ist ein Ereignis älter als 15 Sekunden, wird es verworfen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung b.1: Warum Sie den Anwender »verfolgen« wollen

```
- (void)locationManager:(CLLocationManager *)manager
           didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation {

  NSDate* eventDate = newLocation.timestamp;
  NSTimeInterval howRecent = [eventDate
           timeIntervalSinceNow];
  if (abs(howRecent) < 15.0)
  {
    NSLog(@"latitude %+.6f, longitude %+.6f ",
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
            newLocation.coordinate.latitude,
            newLocation.coordinate.longitude);
    }
}
```

Listing b.4: Ein eintreffendes Standort-Ereignis verarbeiten

Um die Zeit zu ermitteln, zu der der Standort bestimmt wurde, greifen Sie auf die Eigenschaft `timestamp` des `CLLocation`-Objekts zu.

```
NSDate* eventDate = newLocation.timestamp;
```

Um herauszufinden, wie alt dieses Ereignis ist (in Sekunden), schicken Sie die Nachricht `timeIntervalSinceNow` an das `NSDate`-Objekt.

```
howRecent = [eventDate timeIntervalSinceNow];
```

Ich habe nichts über die Klasse `NSDate` erzählt, die Datums- und Uhrzeit-Informationen liefert. Das geht über den Rahmen dieses Buches hinaus, aber Sie sollten sie sich auf jeden Fall genauer anschauen.

Nachdem Sie den Code hinzugefügt und kompiliert haben, erhalten Sie alle möglichen netten Nachrichten im Log, zum Beispiel:

```
2010-06-27 07:36:37.156 iPhoneTravel411[94443:207]
                latitude +37.331650, longitude -122.030704
```

Natürlich hilft die Ausgabe des Standorts in der Logdatei dem Anwender nicht sehr viel. Sie werden ihm eine Benachrichtigung schicken wollen, was ich Ihnen auch im Abschnitt *Lokale Benachrichtigungen* zeigen werde. Aber zunächst …

Fehlerbehandlung

Gibt es beim Empfangen eines Ereignisses einen Fehler, schickt der Location Manager die Nachricht `locationManager:didFailWithError:` an sein Delegate.

Um auch damit klarzukommen, fügen Sie den Code aus Listing b.5 in `iPhoneTravel411AppDelegate.m` ein.

```
- (void)locationManager:(CLLocationManager *)manager
       didFailWithError:(NSError *)error {

  NSLog(@"Location error %@, %@", error, @"Fill in
         the reason here");
}
```

Listing b.5: Fehler

Ich gebe hier einfach die Fehlermeldung in der Logdatei aus, Sie wollen vielleicht mehr damit machen (zum Beispiel den Anwender informieren).

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

`error` ist ein Objekt, das den Grund für die fehlerhafte Standort- oder Richtungs-Bestimmung enthält. (Sie werden feststellen, dass die meisten System-Serviceklassen Fehler auf diese Art und Weise melden.)

Mögliche Fehler sind unter anderem:

- ✔ `kCLErrorLocationUnknown`, wenn der Location Service den Standort nicht sauber ermitteln konnte
- ✔ `kCLErrorDenied`, wenn der Anwender den Zugriff der App auf den Location Service verweigert

Lokale Benachrichtigungen

Okay, jetzt haben Sie Ihre Standort-Information, aber was wollen Sie damit machen?

Sie als Entwickler haben natürlich kein Problem damit, Stunden damit zu verbringen, die Standort-Änderungen im Log zu verfolgen. (Lachen Sie nicht – ich habe mein iPhone tatsächlich mit meinem MacBook Pro verbunden und bin damit herumgefahren, um sicherzustellen, dass alles funktioniert.) Aber Ihre Anwender brauchen dann doch eine etwas benutzerfreundlichere Darstellung der Aktualisierungen. Das können Sie über lokale Benachrichtigungen erreichen.

Lokale Benachrichtigungen kann man nicht nur nutzen, wenn die App im Vordergrund läuft, sondern auch im Hintergrund. Also muss ich Ihnen erklären, wie das jeweils funktioniert. Ich fange mit dem Vordergrund-Szenario an und zeige im nächsten Abschnitt, wie Sie durch das Multitasking in iOS 4 Ihre App Standort-Ereignisse verarbeiten und Benachrichtigungen veröffentlichen lassen können, während sie im Hintergrund läuft.

Listing b.6 zeigt Ihnen erst einmal, wie Sie eine lokale Benachrichtigung anzeigen.

```
- (void)locationChangeNotify {

  UILocalNotification *note
                    = [[UILocalNotification alloc] init];
  RootViewController *theRootViewController =
   [navigationController.viewControllers objectAtIndex:0];
  CLLocationCoordinate2D cityCoordinate =
   [theRootViewController.destination returnCityLocation];
  CLLocation *cityLocation =
        [[CLLocation alloc] initWithLatitude:
                      cityCoordinate.latitude
                      longitude:cityCoordinate.longitude];
  NSString *message = [[NSString alloc]initWithFormat:
    @" Sie sind jetzt %.1f km von %@ entfernt",
    [locationManager.location
    distanceFromLocation:cityLocation]/1000.0,
    [theRootViewController.destination returnCityName]];
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
  note.alertBody= message;
  [message release];
  [[UIApplication sharedApplication]
                          presentLocalNotificationNow:note];
  [note release];
  [cityLocation release];
}
```

Listing b.6: Die Benachrichtigung erzeugen

Beachten Sie, dass ich hier als Erstes ein `UILocalNotification`-Objekt allokiere und initialisiere.

```
UILocalNotification *note
                    = [[UILocalNotification alloc] init];
```

Sie können für die Benachrichtigung eine Reihe von Eigenschaften setzen, so zum Beispiel das `fireDate` – Datum und Uhrzeit, zu der das Betriebssystem die Benachrichtigung ausgeben soll. Hat `fireDate` den Wert `nil` (der Standardwert), wird die Benachrichtigung sofort ausgegeben.

Auch können Sie ein Symbol und sogar einen Klang auswählen sowie eigene Daten an die Benachrichtigung hängen, indem Sie die Eigenschaft `userInfo` nutzen. Das mache ich alles nicht, aber es ist gut zu wissen, dass es möglich ist.

Kommen wir wieder auf Listing b.6 zurück. Ich hole mir den Standort des `City`-Objekts (der sich zuerst in London befand, aber Sie haben es ja in Kapitel A verschiebbar gemacht) und schicke ihn an die Nachricht `returnCityLocation`. Das sollte für Sie jetzt nichts Spannendes mehr sein.

Interessant ist aber, wie ich den `RootViewController` finde. Das `iPhoneTravel411AppDelegate` hat keine Instanzvariable, die auf den `RootViewController` zeigt. Vielleicht überlegen Sie gerade, wie man eine erzeugen könnte, aber es gibt eine einfachere Möglichkeit. `iPhoneTravel411AppDelegate` ***hat*** nämlich eine Instanzvariable, die auf den `NavigationController` zeigt. Wie ich in Kapitel 14 beschrieben habe, verwaltet der Navigation Controller einen Stack mit View Controllern – einen für jede angezeigte View –, der mit dem Root View Controller beginnt.

Dann erstelle ich ein `CLLocation`-Objekt, das ich bald benötige, und weise es `cityCoordinate` zu.

```
RootViewController *theRootViewController =
   [navigationController.viewControllers objectAtIndex:0];
CLLocationCoordinate2D cityCoordinate =
   [theRootViewController.destination returnCityLocation];
CLLocation *cityLocation =
   [[CLLocation alloc] initWithLatitude:
      cityCoordinate.latitude
      longitude:cityCoordinate.longitude];
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Nun erstelle ich die Nachricht `note.alertBody`, in der beschrieben wird, wie weit ich von der Stadt entfernt bin. Um das herauszufinden, schicke ich eine Nachricht `distanceFromLocation` an das Objekt `locationManager.location` (mit dem aktuellen Standort) mit dem `CLLocation`-Objekt `cityLocation` als Argument.

`distanceFromLocation:` ist eine meiner Lieblingsmethoden, da sie den Abstand zwischen zwei beliebigen `CLLocation`-Objekten in Metern zurückgibt. (Darum habe ich ein `CLLocation`-Objekt erzeugt, indem ich die `CLLocationCoordinate2D` nutzte, das `City` zurückgab.) Damit spare ich mir eine Menge Rechenaufwand.

```
NSString *message = [[NSString alloc]initWithFormat:
    @" Sie sind jetzt %.1f km von %@ entfernt",
    [locationManager.location
    distanceFromLocation:cityLocation]/1000,
    [[theRootViewController destination] returnCityName]];
  note.alertBody= message;
  [message release];
```

Ich weise nun das OS an, die lokale Benachrichtigung zu veröffentlichen. Dazu verschicke ich die Nachricht `scheduleLocalNotification:` von `UIApplication`. Wie schon erwähnt, nutzt die App die Eigenschaft `fireDate` des Objekts `UILocalNotification`, um den Zeitpunkt des Bekanntmachens festzulegen. Da ich dort `nil` angegeben habe, wird die Nachricht sofort angezeigt. Ich hätte auch `presentLocalNotificationNow:` nutzen können, um die Nachricht sofort auszugeben.

Schließlich müssen Sie auch noch die Deklaration in `iPhoneTravel411AppDelegate.h` ergänzen.

Um die lokalen Benachrichtigungen zu sehen, müssen Sie sich natürlich selbst die Nachricht `locationChangeNotify` senden und dann etwas damit anstellen. Sie schicken sich die Nachricht selbst, wenn Sie eine Standort-Aktualisierung von `locationManager: didUpdateToLocation:fromLocation:` erhalten, die Sie gerade hinzugefügt haben. Damit das funktioniert, fügen Sie den fett gedruckten Code aus Listing b.7 hinzu.

```
- (void)locationManager:(CLLocationManager *)manager
           didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation {

  NSDate* eventDate = newLocation.timestamp;
  NSTimeInterval howRecent = [eventDate timeIntervalSinceNow];
  if (abs(howRecent) < 15.0) {
    NSLog(@"latitude %+.6f, longitude %+.6f ",
          newLocation.coordinate.latitude,
          newLocation.coordinate.longitude);
  }
  [self locationChangeNotify];
}
```

Listing b.7: Eine Benachrichtigung anfordern

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Das letzte Puzzleteil ist das Empfangen der Nachricht und ihre Verarbeitung, um den Anwender zu informieren. Die Verarbeitung wird von einer anderen Delegate-Methode `application:didReceiveLocalNotification:` vorgenommen. Listing b.8 zeigt die Details.

```
- (void)application:(UIApplication *)application
         didReceiveLocalNotification:
                     (UILocalNotification *)notification {

  if ([UIApplication sharedApplication].applicationState
                              == UIApplicationStateActive) {
  UIAlertView *alertView = [[UIAlertView alloc]
    initWithTitle:@"Sie sind unterwegs"
    message:notification.alertBody delegate:nil
    cancelButtonTitle:@" OK" otherButtonTitles:nil];
  [alertView show];
  [alertView release];
  }
}
```

Listing b.8: Die Benachrichtigungen verarbeiten, wenn die App im Vordergrund abläuft

In Listing b.8 sehen Sie, dass das App-Delegate die Nachricht `application:didReceiveLocalNotification:` erhält, wenn die Anwendung im Vordergrund läuft. (Nicht ganz überraschend erhält das Delegate die Nachricht `application:didReceiveRemoteNotification` bei Remote-Benachrichtigungen.) Dabei wird das Benachrichtigungsobjekt übergeben.

In `application:didReceiveLocalNotification:` können Sie alles Mögliche mit den Benachrichtigungen anstellen, auf das ich hier im Buch nicht eingehen werde.

In diesem Fall prüfe ich nur, ob die App aktiv ist– also im Vordergrund läuft –, um dann einen Alert auszugeben:

```
if ([UIApplication sharedApplication].applicationState
                              == UIApplicationStateActive) {
  UIAlertView *alertView = [[UIAlertView alloc]
    initWithTitle:@"Sie sind unterwegs"
    message:notification.alertBody delegate:nil
    cancelButtonTitle:@" OK" otherButtonTitles:nil];
  [alertView show];
  [alertView release];
}
```

Denken Sie daran, dass die App auch im Hintergrund laufen kann. Dann sollten Sie keinen Alert oder andere Benutzeroberflächen-Objekte erstellen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Wenn alles läuft, sollten Sie mehr oder weniger so etwas wie in Abbildung b.2 erhalten. »Mehr oder weniger«, weil ich das Ziel vorher nach Harrow verschoben hatte, was Sie auch im Alert zu sehen bekommen.

Ist die App gerade nicht aktiv, überlassen Sie es dem Betriebssystem, den Alert auszugeben (siehe Abbildung b.3). Einen Hinweis ausgeben zu können, während die App nicht angezeigt wird, verdanken wir den Multitasking-Fähigkeiten des iPhones. Darum kümmere ich mich als Nächstes.

Schließlich müssen Sie noch `returnCityName` und `returnCityLocation` in `Destination.m` und ihre Deklarationen in `Destination.h` einfügen. Vergessen Sie auch nicht, in `iPhoneTravel411AppDelegate.m` die Zeile `#import Destination.h` einzufügen.

```
- (CLLocationCoordinate2D) returnCityLocation {

  return city.coordinate;
}

- (NSString*) returnCityName {

  return city.cityName;
}
```

Im Hintergrund laufen

Ich finde es natürlich klasse, immer über meine Entfernung zum Ziel informiert zu werden, aber ich habe iPhoneTravel411 nicht immer im Vordergrund laufen. Einer der tollen Vorteile des Multitasking im iOS 4 auf dem iPhone 3GS und neuer ist, dass ich bestimmte Ereignisse auch im Hintergrund verarbeiten kann. Noch besser ist, dass Standort-Aktualisierungen dazugehören.

Soll die App Standort-Aktualisierungen auch im Hintergrund erhalten, stehen Ihnen verschiedene Wege zur Verfügung:

- ✔ Sie können den Standard Location Service weiterhin nutzen. Das werde ich in diesem Abschnitt vorstellen.
- ✔ Sie können auch den Significant-Location Change Service nutzen. Damit wird Ihre App regelmäßig geweckt, um neue Ereignisse zu verarbeiten.

  Ich werde im Abschnitt *Den Significant-Change Location Service nutzen, um den Akku zu schonen* noch erklären, wann Sie diesen Weg gehen sollten.

Eine App sollte Location Services im Hintergrund nur anfordern, wenn das wirklich notwendig ist. Es liegt an Ihnen, zu entscheiden, ob Sie die Services für eine besonders hohe Benutzerfreundlichkeit brauchen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung b.2: Weit, weit weg

In Kapitel 2 habe ich beschrieben, wie (bestimmte) Apps in den Hintergrund verschoben werden können, wenn der Anwender etwas tut:

- ✔ Die App wird durch einen eingehenden Telefonanruf, eine SMS oder eine Benachrichtigung durch den Kalender unterbrochen.
- ✔ Der Anwender drückt auf den Sleep/Wake-Knopf.
- ✔ Der Anwender drückt auf die Home-Taste oder das System startet eine andere App und die aktuelle App wird in den Hintergrund verschoben.

Anwendungen, die mit dem iPhone SDK 4.0 oder neuer gebaut wurden (und unter dem iOS 4.0 oder neuer laufen), werden in diesen Situationen nicht mehr länger beendet. Stattdessen

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung b.3: Ein System-Alert

werden sie in den Hintergrund verschoben. Die meisten Apps werden zwar kurz nach diesem Verschieben auch schlafen gelegt, aber Apps, die dort weiterarbeiten müssen, können eines der folgenden Dinge tun:

1. Eine begrenzte Zeitdauer anfordern, um eine Aufgabe zu beenden.
2. Deklarieren, dass sie bestimmte Dienste unterstützen, die im Hintergrund ablaufen können müssen.
3. Lokale Benachrichtigungen verwenden, um den Anwender zu bestimmten Zeiten zu informieren – egal, ob die App läuft oder nicht.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

In diesem Abschnitt werde ich Option 2 behandeln, im Abschnitt *Den Significant-Change Location Service nutzen, um den Akku zu schonen* kümmere ich mich um Option 3.

Applikationen, die im Hintergrund laufen, können unter bestimmten Bedingungen immer noch beendet werden (wie zum Beispiel, wenn zu wenig Speicher zur Verfügung steht). Sie müssen also darauf vorbereitet sein, dass die App jederzeit abgebrochen werden kann. Daher müssen viele der Aufgaben, die beim Beenden zu erledigen sind, schon beim Verschieben in den Hintergrund ausgeführt werden.

Nicht alle iOS-basierten Geräte unterstützen eine Verarbeitung im Hintergrund. Läuft auf einem Gerät kein iOS 4 (oder neuer) oder unterstützt die Hardware kein Multitasking, behandelt das System das Beenden einer App anders. Wollen Sie wissen, wie Sie eine App erstellen, die auf älteren Betriebssystemen läuft, schauen Sie auf meiner Website `www.nealgoldstein.com` vorbei.

Die unterstützten Hintergrund-Aufgaben deklarieren

Soll die App im Hintergrund laufen können, müssen Sie dem Betriebssystem *mitteilen*, dass Sie das vorhaben. Dazu nehmen Sie den Schlüssel `UIBackgroundModes` in der Datei `Info.plist` der Anwendung mit auf. Dieser Schlüssel legt fest, welche Hintergrund-Aufgaben Ihre App unterstützt. Er kann die folgenden Werte besitzen:

- ✔ **Audio:** Die App spielt Klänge im Hintergrund ab.
- ✔ **Location:** Die App verarbeitet Standort-Ereignisse im Hintergrund.
- ✔ **VoIP:** Die App bietet dem Anwender die Möglichkeit, Voice-over-IP-Telefonate zu führen.

Wie richte ich iPhoneTravel411 nun so ein, dass es Location Services im Hintergrund unterstützt?

In der Liste GROUPS & FILES im Projektfenster wählen Sie `iPhoneTravel411-info.plist` aus und markieren dann den letzten Eintrag im Editor-Bereich (siehe Abbildung b.4).

Das funktioniert ähnlich wie die Datei `root.plist`, mit der Sie in Kapitel 15 gearbeitet haben, um Einstellungen festzulegen.

Wählen Sie nun das +-Symbol auf der rechten Seite aus und scrollen Sie herunter bis zum Eintrag REQUIRED BACKGROUND MODES. Klappen Sie den Eintrag auf und wählen Sie APP REGISTERS FOR LOCATION UPDATES. Das Ergebnis sehen Sie in Abbildung b.5.

Mehr ist nicht zu tun. Ab jetzt erhalten Sie auch Standort-Ereignisse, wenn Ihre App im Hintergrund läuft.

Wie schon erwähnt, gibt es allerdings auch eine andere Möglichkeit, regelmäßig Aktualisierungen zu erhalten. Sie können den Significant-Change Location Service verwenden. Sie erhalten dann zwar keine regelmäßigen Aktualisierungen im Hintergrund, aber Ihre App wird aktiviert (oder sogar neu gestartet), um eine Änderung zu verarbeiten.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

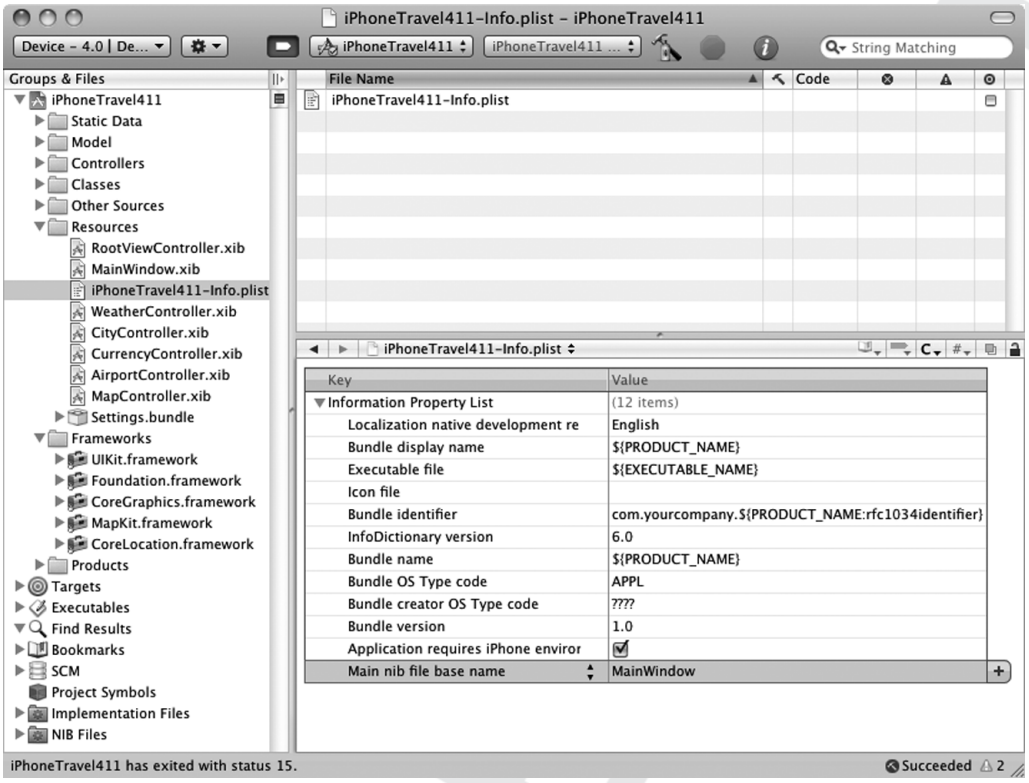

Abbildung b.4: `Info.plist` *für Ihre App*

Den Significant-Change Location Service nutzen, um den Akku zu schonen

Wie Sie sich vielleicht schon angesichts meines Herumreitens auf der Akkulaufzeit gedacht habe, ist das Einholen von Standort-Informationen sehr stromintensiv. Lassen Sie den Standard Location Service laufen, kostet das viel Akkulaufzeit. Die meisten Apps benötigen gar keine dauerhaft laufenden Location Services, daher ist es am einfachsten, sie abzuschalten. Sie können sie ja jederzeit wieder starten.

Sie sollten zudem die Parameter des Standard Location Service so einstellen, dass seine Auswirkungen auf die Akkulaufzeit so gering wie möglich sind. Nutzen Sie die niedrigsten Genauigkeits-Einstellungen, die für Ihre App sinnvoll sind. Je höher die Genauigkeit sein soll, desto mehr Strom wird verbraucht. Sofern Ihre App nicht wirklich auf ein paar Meter genau wissen muss, wo sich Ihr Anwender befindet, nehmen Sie für die Eigenschaft `desiredAccuracy` nicht die Werte `kCLLocationAccuracyBest` oder `kCLLocationAccuracyNearestTenMeters`. Und denken Sie daran, dass eine Wahl von `kCLLocationAccuracyThreeKilometers` den Location Service nicht davon abhält, bessere Daten zurückzugeben. Core Location kann meist

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

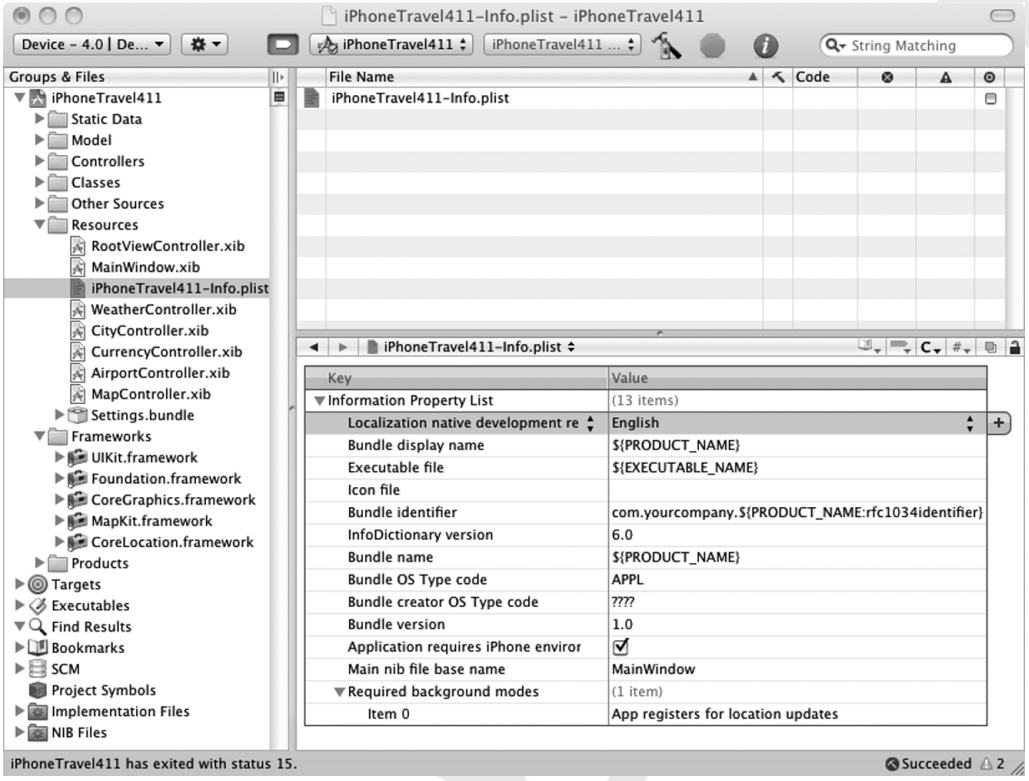

Abbildung b.5: Jetzt kann die App im Hintergrund laufen.

schon eine Genauigkeit von 100 Metern oder besser liefern, wenn es nur WLAN- oder Funkzellen-Signale verwendet.

Sie können auch den Significant-Change Location Service nutzen, der im nächsten Abschnitt behandelt wird. Er benötigt viel weniger Strom, da er nur die Änderungen der Funkzellen beobachtet, aber Sie brauchen dementsprechend auch eine Mobilfunkempfangsmöglichkeit. Wenn Sie die haben, können Sie aber die Location Services nutzen und gleichzeitig Strom sparen. Sie werden das in Apps einsetzen, die die Position des Anwenders verfolgen, dabei aber den Akku nicht gleich leersaugen. Zudem kann der Significant-Change Location Service Ihre schlafen gelegte App aufwecken oder sie überhaupt erst starten, wenn ein Standort-Ereignis eintrifft – Sie müssen die App gar nicht unbedingt im Hintergrund laufen lassen.

Neben all den Möglichkeiten, die Sie als Entwickler haben, um die Akkulaufzeit nicht zu sehr zu verringern, sollten Sie auch dem Anwender Einflussnahme gewähren. Das ist der Grund, warum ich Sie die Einstellung ABSTAND ÜBERWACHEN in Kapitel 15 habe hinzufügen lassen. So kann der Benutzer entscheiden, ob ihm das Feature wichtig ist oder ob er lieber eine längere Akkulaufzeit hätte.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Den Significant-Change Service nutzen

Sie nutzen den Significant-Change Location Service fast genauso wie den Standard Location Service. Änderungen werden an die gleiche Delegate-Methode `locationManager:didUpdateToLocation:fromLocation:` geschickt – hier werden aber nur deutliche Änderungen vermerkt. Fehler werden genauso wie beim Standard Service an `locationManager:didFailWithError:` geschickt. Wenn Sie in dieser App den Significant-Change Location Service nutzen, bleibt der Code in diesen beiden Methoden identisch. Der einzige Unterschied ist, dass Sie dem Location Manager nicht die Methoden `startUpdatingLocation` und `stopUpdatingLocation` senden, sondern `startMonitoringSignificantLocationChanges` und `stopMonitoringSignificantLocationChanges`, wie das auch im fett gedruckten Code in Listing b.9 zu sehen ist. Wollen Sie den Significant-Change Location Service nutzen, löschen Sie den durchgestrichenen Code und fügen den fett gedruckten ein.

```
- (void)startLocationUpdates:(BOOL) startUpdates {
  if (startUpdates ) {
    if (nil == locationManager)
      locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.purpose = @"iPhoneTravel411 möchte
          berechnen, wie weit Sie von Ihrem Ziel
          entfernt sind.";
    locationManager.desiredAccuracy =
          kCLLocationAccuracyKilometer;
    locationManager.distanceFilter = 500;
    ~~[locationManager startUpdatingLocation];~~
    [locationManager
                startMonitoringSignificantLocationChanges];
  }
  else {
    ~~[locationManager stopUpdatingLocation];~~
    locationManager
                 stopMonitoringSignificantLocationChanges];
    NSLog(@"Monitoring stopped");
  }
}
```

Listing b.9: Der Significant-Change Location Service

Sie müssen sich dann auch nicht um das Registrieren Ihrer App für Standort-Aktualisierungen in den Required Background Modes kümmern, das ich im Abschnitt *Die unterstützten Hintergrund-Aufgaben deklarieren* weiter oben beschrieben habe – Ihre App muss nämlich gar nicht im Hintergrund laufen. Sie wird im Zweifel aufgeweckt und kann dann im Hintergrund die Nachricht `locationManager:didUpdateToLocation:fromLocation:` verarbeiten. Wurde Ihre App beendet, wird sie im Hintergrund wieder gestartet, um das Ereignis zu nutzen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Region Monitoring

Auf dem iPhone 4 und neuer können Apps das Region Monitoring nutzen, um informiert zu werden, wenn der Anwender bestimmte geografische Grenzen überschreitet. Sie können dieses Feature nutzen, um Meldungen zu erzeugen, wenn der Anwender einen Bereich betritt oder verlässt. So kann die App zum Beispiel Leuten wie mir eine Nachricht zukommen lassen, nochmals zu schauen, ob sie ihren Pass, die Flugtickets und so weiter dabei haben, bevor sie allzu weit von zu Hause entfernt sind.

Das können Sie zwar auch selbst programmieren – was ich auch in einigen meiner Apps getan habe –, aber mit den Location Services ist es viel einfacher.

Die Implementierung entspricht weitgehend der für den Significant-Change Location Service. Die Region, die mit Ihrer App verbunden ist, wird dauerhaft überwacht, auch wenn die App selbst nicht läuft. Wird ein »Grenzübertritt« beobachtet, während die App nicht läuft, wird sie im Hintergrund gestartet, damit sie das Ereignis verarbeiten kann. (Ich zeige Ihnen das gleich noch.) Ist die App schlafen gelegt, wird sie aufgeweckt und hat eine gewisse Zeitspanne zur Verfügung, um das Ereignis zu verarbeiten.

Listing b.10 zeigt, wie Sie das Region Monitoring in iPhoneTravel411 implementieren können.

```
- (BOOL)startRegionMonitoring {
  if (![CLLocationManager regionMonitoringAvailable] ||
      ![CLLocationManager regionMonitoringEnabled] )
    return NO;

  CLLocationCoordinate2D home;
  home.latitude = +37.441655;
  home.longitude = -122.143282;
  CLRegion* region = [[CLRegion alloc]
                      initCircularRegionWithCenter:home
                      radius:100.0 identifier:@"home"];
  if (locationManager == nil)
    locationManager = [[CLLocationManager alloc] init];
  [locationManager startMonitoringForRegion:region
              desiredAccuracy:kCLLocationAccuracyBest];

  [region release];
  return YES;
}

- (void)locationManager:(CLLocationManager *)manager
           didEnterRegion:(CLRegion *)region {

  [self leavingHomeNotify];
}
```


Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

```
- (void)locationManager:(CLLocationManager *)manager
          didExitRegion:(CLRegion *)region {

  [self leavingHomeNotify];
}

- (void)locationManager:(CLLocationManager *)manager
          monitoringDidFailForRegion:(CLRegion *)
          regionwithError:(NSError *)error {
  NSLog(@"Location error %@, %@", error, @"Fill in
         the reason here");
}

- (void)leavingHomeNotify {

  UILocalNotification *note =
  [[UILocalNotification alloc] init];
  note.alertBody= @"Vergessen Sie nicht Ihr iPhone,
                    Ihre Tickets und den Pass";
  [[UIApplication sharedApplication]
              presentLocalNotificationNow:note];
  [note release];
}
```

Listing b.10: Region Monitoring

Ich werde nicht auf jede Codezeile eingehen, aber ich will auf ein paar Dinge hinweisen.

Bevor ich irgendwelche Regionen überwache, prüfe ich, ob das Region Monitoring überhaupt auf dem aktuellen Gerät unterstützt wird.

```
if (![CLLocationManager regionMonitoringAvailable] ||
    ![CLLocationManager regionMonitoringEnabled] )
  return NO;
```

Es gibt einige Gründe, warum das Region Monitoring nicht verfügbar ist:

- ✔ Das Gerät hat gar nicht die Hardware, die das Region Monitoring unterstützt.
- ✔ Der Anwender hat die Location Services abgeschaltet.
- ✔ Das Gerät befindet sich im Flugzeug-Modus.

Es kann sich durchaus lohnen, herauszufinden, ob der Service überhaupt verfügbar ist. Denn wenn nicht, könnten Sie diese Funktion in der App ganz ausblenden oder eine Software-Version davon selbst implementieren, indem Sie Ihre eigene Region definieren und dann regelmäßig prüfen, ob sich der Standort verändert hat und ob sich der Anwender daher nicht mehr in der Region befindet (was allerdings deutlich mehr Strom kostet).

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Will ich eine Region überwachen, muss ich sie zuerst definieren und im System registrieren. Regionen werden mit der Klasse `CLRegion` definiert, die zurzeit nur das Erstellen kreisförmiger Regionen unterstützt. Eine solche Region besteht aus der Definition eines geografischen Bereichs und einer eindeutigen (String-)Kennung, die Sie brauchen, um die Region später wiederzufinden. Ich definiere den Mittelpunkt der Region mit den Koordinaten meines Zuhauses, zudem lege ich einen Radius von 100 Metern fest.

```
CLLocationCoordinate2D home;
home.latitude = +37.441655;
home.longitude = -122.143282;
CLRegion* region = [[CLRegion alloc]
                    initCircularRegionWithCenter:home
                    radius:100.0 identifier:@"home"];
if (locationManager == nil)
  locationManager = [[CLLocationManager alloc] init];
[locationManager startMonitoringForRegion:region
            desiredAccuracy:kCLLocationAccuracyBest];
```

Regionen sind gemeinsam genutzte System-Ressourcen und es gibt eine Obergrenze für die Gesamtanzahl. Versuchen Sie, eine Region zu registrieren, obwohl kein Platz mehr zur Verfügung steht, schickt der Location Manager dem Delegate die Nachricht `locationManager:monitoringDidFailForRegion:withError:` und den Fehlercode `kCLErrorRegionMonitoringFailure`.

Die Überwachung einer Region beginnt zwar sofort, aber nur das Überschreiten einer Grenze erzeugt ein Ereignis. Befindet sich der Anwender schon in der Region, erhält er kein Ereignis, bis er sie verlässt.

Bei solch einem Grenzübertritt schickt der Location Manager eine Delegate-Nachricht:

```
- (void)locationManager:(CLLocationManager *)manager
           didEnterRegion:(CLRegion *)region {

  // Hier könnten Sie etwas Kluges sagen
}

- (void)locationManager:(CLLocationManager *)manager
           didExitRegion:(CLRegion *)region {

  [self leavingHomeNotify];
}
```

(Sie würden `locationManager:didEnterRegion:` hier natürlich nicht nutzen. Ich zeige es nur, damit Sie beide Varianten kennen.)

Läuft Ihre App bereits, gehen diese Ereignisse direkt an die Delegates aller aktuellen Location-Manager-Objekte. Läuft Ihre App nicht, wird sie im Hintergrund gestartet, damit sie darauf reagieren kann.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung b.6 zeigt, was passiert, wenn ich mein Haus verlasse und die in Listing b.10 definierte Region verlasse.

Abbildung b.6: Verlassen Sie Ihr Haus nicht ohne die wichtigen Dinge.

Ihre App wird gestartet

Damit komme ich zum (wirklich) Letzten, das Sie über Multitasking wissen müssen.

Ist die App beendet worden und verarbeitet sie eigentlich Ereignisse im Hintergrund oder nutzt die Significant-Change oder Region-Monitoring Services, wird sie vom System automatisch gestartet, wenn es neue Standort-Daten gibt.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Fragen Sie sich, wofür all die Optionen in der Methode `application:didFinishLaunchingWithOptions:` nützlich sind? Nun, jetzt erfahren Sie es. Wird Ihre App vom System aus einem bestimmten Grund gestartet, enthält das Dictionary `launchOptions` Daten dazu. Sie finden darin Schlüssel für eine Reihe von Ereignissen, wobei ich meine Erläuterungen auf die folgenden beiden beschränken werde:

- ✔ **Die App wurde vom Anwender als Reaktion auf eine lokale Benachrichtigung gestartet.** Der Schlüssel für dieses Ereignis ist `UIApplicationLaunchOptionsLocalNotificationKey`.
- ✔ **Die App verfolgt Standort-Änderungen im Hintergrund, wurde beendet und jetzt wieder neu gestartet.** Der Schlüssel dazu ist `UIApplicationLaunchOptionsLocationKey`.

Wie Sie sehen, muss iPhoneTravel411 gar nicht alle Start-Optionen kennen, um Standort-Änderungen oder lokale Benachrichtigungen zu berücksichtigen. Aber es kann durchaus einmal vorkommen, dass Ihre App so etwas wissen muss. Listing b.11 zeigt, wie Sie solch einen Start als Reaktion auf eine lokale Benachrichtigung erkennen.

```
UILocalNotification *locationNotification = [launchOptions
          objectForKey:UIApplicationLaunchOptionsLocation
          Key];
if (locationNotification) {
  UILocalNotification *note = [[UILocalNotification
          alloc] init];
  note.alertBody= @"iPhoneTravel441 wurde gestartet, um
          Sie an Ihre Tickets und den Pass zu
          erinnern";
  [[UIApplication sharedApplication]
          presentLocalNotificationNow:note];
  [note release];
}
```

Listing b.11: Start nach einer lokalen Benachrichtigung

Abbildung b.7 zeigt das Ergebnis.

Das Vorhandensein von Location Services als Voraussetzung für einen Start der App

Baut Ihre App auf dem Vorhandensein von nutzbaren Location Services auf, sollten Sie den Schlüssel `UIRequiredDeviceCapabilities` in der Datei `Info.plist` Ihrer App aufnehmen. Damit wird festgelegt, dass die Location Services vorhanden sein müssen, um Ihre App laufen lassen zu können. Der App Store nutzt diese Information, um Anwender davon abzuhalten, Apps auf Geräte herunterzuladen, die die entsprechenden Features nicht besitzen.

Der Wert von `UIRequiredDeviceCapabilities` ist ein String-Array mit den Features, die Ihre App benötigt. Die folgenden beiden Strings sind für Location Services relevant:

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Abbildung b.7: Start als Reaktion auf ein Ereignis

- ✔ `location-services`: Brauchen Sie allgemein Zugriff auf Location Services, nehmen Sie diesen String mit auf.
- ✔ `gps`: Nehmen Sie diesen String mit auf, wenn Ihre App eine Standort-Genauigkeit benötigt, die nur GPS-Hardware bieten kann.

Nutzt Ihre App Location Services, kann aber auch wunderbar ohne sie laufen, stellen Sie keine solchen Anforderungen über `UIRequiredDeviveCapabilities`.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4

Was kommt jetzt?

Wir haben hier das Ende unserer Tour durch die iPhone-Softwareentwicklung erreicht. Sie sollten jetzt in der Lage sein, selbst Programme zu entwickeln – wenn Sie nicht schon längst angefangen haben.

Das Entwickeln von iPhone-Apps ist eine der interessantesten Aktivitäten, die ich seit langer Zeit erleben durfte. Ich hoffe, dass auch Sie das so erleben.

Verlassen Sie mich aber nicht ganz. Schauen Sie regelmäßig auf meiner Website `www.nealgoldstein.com` vorbei. Dort finden Sie komplette Xcode-Projekte für ReturnMeTo und iPhoneTravel411. Sie können mir dort auch Fragen stellen und wenn mehr als ein paar Leute bei einem Thema Probleme haben, werde ich dazu etwas schreiben. Auch werde ich immer wieder einmal aktuelle Texte dort veröffentlichen, so zum Beispiel Einblicke von einem App-Entwickler zum Verkaufen von Apps, zu neuen Features von Apple oder zu Richtlinien-Änderungen, von denen Sie wissen sollten. Der letzte Punkt ist besonders wichtig, weil die Richtlinien von Apple durchaus variieren können.

Auch werde ich dort meine Meinung dazu schreiben, wohin sich mobile Anwendungen allgemein bewegen, welche Trends in der Branche entstehen und was der nächste große Knaller werden könnte.

Schließlich: Haben Sie Spaß! Ich hoffe, ich kann einmal eine Ihrer Anwendungen aus dem App Store herunterladen.

Bonuskapitel zu iPhone Apps Entwicklung für Dummies, ISBN 978-3-527-70729-4