Camera, Events, and Contacts

Dr. Sarah Abraham
Camera and Photo Library
Using the Camera and Photos

- UIImagePickerController handles access to camera device, camera roll and photo library
- Photos and videos can be taken within application
- Existing photos and videos can be presented to the user
- Configure UIImagePickerController object to determine functionality in app
Using UIImagePickerController

1. Check for access to camera/camera roll/photo library within app
   ✤ use authorizationStatus(for:) method

2. Create an instance of UIImagePickerController

3. Set attributes:
   ✤ sourceType sets image picker source (Camera, SavedPhotosAlbum, PhotoLibrary)
   ✤ mediaType sets image (kUTTypeImage) and video (kUTTypeMovie) types
   ✤ allowsEditing allows changes to image before returning it to the app
Camera Example

let imagePicker = UIImagePickerController()

imagePicker.delegate = self

imagePicker.sourceType = .PhotoLibrary

imagePicker.mediaTypes = [kUTTypeImage as NSString]

imagePicker.allowsEditing = false

self.presentViewController(imagePicker, animated: true, completion: nil)

Note: requires testing on a device
ImagePickerController Controller

- Presents controller based on requested source type
  - UIImagePickerControllerSourceType.Camera
  - UIImagePickerControllerSourceType.PhotoLibrary
  - UIImagePickerControllerSourceType.SavedPhotoAlbums
- Camera provides access to camera as the source
- PhotoLibrary provides access to all photos available on the device including iCloud libraries
- SavedPhotoAlbums provides access to local images on the device
UIImagePickerControllerDelegate

- Provides notifications between image picker and application
  - User takes a picture/records a video
  - User selects something from the camera roll/photo library
  - User cancels selection operation
- Must implement these methods to extract media from image picker
  - didFinishPickingMediaWithInfo provides a dictionary with media (UIImage) and associated data
  - imagePickerControllerDidCancel notifies delegate that the user cancelled the pick operation
AVCaptureSession

- More customizable framework for photo and video capture
- Manages capture and output to media
Calendar and Events

Meeting
Office
All-day
Starts 16 August 2013 13:45
Ends 14:45
Using the App Calendar

- Event Kit provides access to a user’s calendar events and reminders
  - Events and reminders stored within Event Store database on a device
- Event Kit provides functionality for:
  - Getting a list of calendars
  - Getting attributes of a calendar
  - Creating/deleting a calendar
  - Creating/modifying/deleting an event
Using Event Kit

1. import EventKit

2. Create an instance of EventStore

3. Use EventStore object to verify app has permission to access events using authorizationStatus(for:)
   ✤ Must handle cases if app does not have access

4. Read/write calendars and events within EventStore
EventStore

- EKEventStore provides access to calendar and reminder list APIs
  - Must be used to access and modify calendars/reminders
  - let eventStore = EKEventStore()

- Calendars returned as EKCalendar objects
  - eventStore.calendarsForEntityType(EKEntity Type.Event)
Creating Calendars

✧ Create an EKCalendar object

✧ Set its attributes (title and a valid source)

✧ EKSource must be retrieved fromEKEventStore

✧ After saving, store the key associated with that calendar

✧ Allows for easy retrieval/removal of the calendar
let newCalendar = EKCalendar(for: .Event, eventStore: eventStore)

newCalendar.title = "Calendar Name"

newCalendar.source = eventStore.sources.filter {
    (source: EKSource) -> Bool in
    source.sourceType.rawValue == EKSourceType.Local.rawValue
} .first!

do {
    try eventStore.saveCalendar(newCalendar, commit: true)

    NSUserDefaults.standardUserDefaults().setObject(newCalendar.calendarIdentifier, forKey: "kCalendarNameIdentifier")
}
catch { /* Exception handling here */}
EKSourceTypes

Control the source properties of events

- local
- exchange
- calDav (iCloud)
- mobileMe
- subscribed
- birthday
Creating Events

1. Find calendar to add the event to
2. Create `EKEvent` object
3. Set attributes
4. Save

🔗 <https://www.andrewcbancroft.com/2016/06/02/creating-calendar-events-with-event-kit-and-swift/>
Event Authorization

- App must be authorized by the device user to access EventStore
  - EKAAuthorizationStatus.Authorized

- Check for authorization access:
  - let status = EKEventStore.authorizationStatusForEntityType(EKEntityType.Event)

- Potential status returns:
  - .NotDetermined
  - .Authorized
  - .Restricted
  - .Denied
Handling User Permissions

- User can change access status at any time
- Include access status check within `viewWillAppeare` to ensure authorization is up-to-date
- If status is `.NotDetermined`, call `requestAccessToEntityType`

```swift
eventStore.requestAccessToEntityType(EKEntityType.Event, completion: {
    (accessGranted: Bool, error: NSError?) in
    if accessGranted == true {
        /* Perform operations with EventStore */
    } else {
        /* Request the user for access */
    }
})
```
# Contacts

<table>
<thead>
<tr>
<th>Groups</th>
<th>All Contacts</th>
<th>Edit</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>John Appleseed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kate Bell</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anna Haro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniel Higgins Jr.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David Taylor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hank M. Zakroff</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>John Appleseed</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mobile</td>
<td>(888) 555-5512</td>
</tr>
<tr>
<td>home</td>
<td>(888) 555-1212</td>
</tr>
<tr>
<td>work</td>
<td><a href="mailto:John-Appleseed@mac.com">John-Appleseed@mac.com</a></td>
</tr>
<tr>
<td></td>
<td>3494 Kuhl Avenue</td>
</tr>
<tr>
<td></td>
<td>Atlanta GA 30303</td>
</tr>
<tr>
<td></td>
<td>USA</td>
</tr>
<tr>
<td></td>
<td>home</td>
</tr>
</tbody>
</table>
Using Contact Information

✦ Contacts Frameworks provides access to user’s contact information
  ✦ Set of classes that access contact data located in the Contacts Store database on device
  ✦ Searches local database as well as iCloud account (if connected)
  ✦ Presents unified contacts list of all data across all databases
Using Contacts Framework

1. import Contacts

2. Create an instance of CNContactStore

3. Use CNContactStore object to verify app has permission to access events
   ✤ Must handle cases if app does not have access

4. Read/write contacts within Contact Store
CNContactStore

- Represents Contacts database programmatically
- Manages all communication between an app and the Contacts database
- Provides methods for authorization and fetching, saving, and updating records
- Contacts accessed as CNContact and CNMutableContact objects
Contacts Authorization

- `CNAuthorizationStatus` has similar functionality to `EKAuthorizationStatus`

```swift
let authorizationStatus = CNContactStore.authorizationStatus(for: CNEntityType)
```

- Same status returns: (.Authorized, .Restricted, .Denied, .NotDetermined)

- Proceed if app is authorized, seek permission if authorization is not determined
Efficient Retrieval

• Access `CNContactStore` on a background thread to avoid slowdown of UI
• Fetch partial results
  • Prevents expensive searches across all possible contact sources
• Predicates filter returned results
  • `let predicate = CNContact.predicateForContactsMatchingName()`
• Metadata representing contact properties to limit search
let predicate = 
CNContact.predicateForContactsMatchingName(
[String-to-match])

let keys = [CNContactGivenNameKey, CNContactFamilyNameKey]

var contacts = [CNContact]()

let contactsStore = AppDelegate.getAppDelegate().contactStore

do {

    contacts = try
    contactsStore.unifiedContactsMatchingPredicate(
predicate, keysToFetch: keys)

    if contacts.count == 0 {

        /* No contacts found */

    }

} catch {

    /* Couldn’t fetch contacts */

}
Displaying and Selecting

- **CNContactPickerController** for selecting a contact
- **CNContactViewController** for displaying a contact
<table>
<thead>
<tr>
<th>Groups</th>
<th>All Contacts</th>
<th>Cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>John Appleseed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kate Bell</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anna Haro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniel Higgins Jr.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David Taylor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Johnny Appleseed</th>
</tr>
</thead>
<tbody>
<tr>
<td>He knows his trees</td>
</tr>
<tr>
<td>woods</td>
</tr>
<tr>
<td>(555) 123-4567</td>
</tr>
</tbody>
</table>

Create New Contact

Add to Existing Contact