CS 329E – Elements of Mobile Computing

Dr. William C. Bulko

Notifications

Notifications

Notifications are used to provide information to the user based on time or location.

There are several kinds:

- *Basic* (*local*) notifications are simple, and use the *observe* method.
- *Remote* notifications are sent from another device
- *Scheduled local* notifications are sent at a designated time
  - The app determines the schedule (i.e., when the notification should be sent)
  - The system handles the actual delivery.

Notifications

Notifications are an implementation of the *observer software design pattern*. It follows the same general idea as event-driven and MVC patterns.

- An object maintains a list of *observers* and notifies them when the event they're registered to receive occurs.

- Notifications are used to implement distributed event-handling.

Basic (Local) Notifications

- A way to notify other parts of your application that something of interest has happened.
- Make a call to add an *observer* to a property you want to track. Then you implement notification handler code to handle changes to that property.
- Notification handling is *synchronous*.
  - All observers are executed, in turn.
- You can pass data to the notification handlers.

Basic notifications are implemented using KVO: *Key-Value Observing.*

KVO

Key Value Observing (KVO) is a mechanism that allows objects to be notified of changes to specified properties of other objects.

- KVO works with any Swift class, as long as the class inherits from the NSObject class.
- You add an observer for any property you want monitored.
- You can later remove the observer in `deinit`.

Setting up a property to monitor

Define the property you want to observe in the class.

```
class Hero: NSObject {
   @objc dynamic var name = "Peter Parker"
}

let spiderMan = Hero()
```

- The class must inherit from NSObject.
- KVO is implemented in Objective-C. Any properties you wish to observe need to have `@objc dynamic` in front to ensure access is dynamically dispatched by Objective-C runtime.

Observers

Create an *observer* using the method `observe`. It takes three arguments:

- A keypath: ”\”, followed by the class of the object, followed by the name of the property you want to observe.
- A set of options for behaviors you want to observe.

`.new`	issue notification using new property value
`.old`	issue notification using old property value
`.initial`	issue notification at create time
`.prior`	issue one notification before the change and another one after

- A closure specifying what you want to do when the behavior occurs.

Remote Notifications

- Remote notifications are generated by something outside your application
- They are sent through APNS (Apple Push Notification Server)
- Much more complicated than local notifications
- When remote notifications are received by your device iOS displays them in a little pop-down
- If you touch a remote notification iOS will launch your app (if it isn’t already running) or bring it to the foreground (if it’s in the background) and hand off the notification to you
- You need an Apple Developer ID, so we aren’t going to cover this in any detail.

Scheduled Local Notifications

To create a scheduled local notification, you use the `UILocalNotification` class combined with an `NSDate` object configured with the date and time that the notification is to be triggered.

- Properties may also be set to specify the text to be displayed to the user, an optional repeat interval, messages and sounds, etc.