WatchKit In-Depth, Part 1

Session 207

Andrew Platzer WatchKit Engineer
Forest Hill WatchKit Engineer
WatchKit In-Depth, Part 1

Architecture
Resources and Data
Migration
Enhancements
Architecture

Watch App for watchOS 2
Architecture
Watch App for watchOS 2

Components
• iOS application
• watchOS WatchKit extension
• watchOS Watch application
Architecture

Watch App for watchOS 2
Architecture
Watch App for watchOS 2
Architecture
Watch App for watchOS 2
Architecture
Watch App for watchOS 2

Diagram:
- iPhone
  - iOS Application
    - Code
    - Resources
  - Watch App
    - Interface
    - Resources
Architecture
Watch App for watchOS 2
Architecture
Watch App for watchOS 2
Architecture
Watch App for watchOS 2

- iPhone
  - iOS Application
    - Code
    - Resources
  - Watch App
    - Interface
    - Resources
  - WatchKit Extension
    - Code
    - Resources

- Apple Watch
Architecture
Watch App for watchOS 2

iPhone
- iOS Application
  - Code
  - Resources
- Watch App
  - Interface
  - Resources
- WatchKit Extension
  - Code
  - Resources

Apple Watch
- Watch App
  - Interface
  - Resources
- WatchKit Extension
  - Code
  - Resources
Architecture
Watch App for watchOS 2
Architecture
Watch App for watchOS 2

Watch Kit Extension
- Code
- Resources

Watch App
- Interface
- Resources
Architecture
Watch App for watchOS 2

- Watch App
  - Interface
  - Resources
- WatchKit Extension
  - Code
  - Resources
Architecture

WatchKit App
Architecture

Interface.storyboard
Architecture

Interface.storyboard

Edit with Interface Builder
Architecture

Interface.storyboard

Edit with Interface Builder

Interface Elements

- Label
- Image
- Button
- Switch
- Slider
- Date Label
- Timer Label
- Separator
- Group
- Table
- Menus
- Map
Architecture
Interface.storyboard

Edit with Interface Builder

Interface Elements

- Label
- Image
- Button
- Switch
- Slider
- Date Label
- Picker
- Timer Label
- Separator
- Group
- Table
- Menus
- Map
- Movie
Architecture

Interface.storyboard
Architecture
Interface.storyboard
Architecture

Interface.storyboard
Architecture
WatchKit Extension

Roles
• Application
• Glance
• Notification
• Complication
Architecture
WatchKit Extension

Roles

• Application   WKInterfaceController
• Glance        WKInterfaceController
• Notification  WKUserNotificationInterfaceController
• Complication  CLKComplicationDataSource
Architecture
WatchKit Extension

Roles

• Application: WKInterfaceController
• Glance: WKInterfaceController
• Notification: WKUserNotificationInterfaceController
• Complication: CLKComplicationDataSource

Creating Complications with ClockKit
Architecture

WatchKit Extension
Architecture
WatchKit Extension

WKInterfaceController
Architecture

WatchKit Extension

WKInterfaceController

• Interface properties
Architecture
WatchKit Extension

WKInterfaceController

• Interface properties
• Menu handling
Architecture
WatchKit Extension

WKInterfaceController

• Interface properties
• Menu handling
• Controller navigation and paging
Architecture

WatchKit Extension

WKInterfaceController

- Interface properties
- Menu handling
- Controller navigation and paging
- Controller modal presentation
Architecture
WatchKit Extension

WKInterfaceController
• Interface properties
• Menu handling
• Controller navigation and paging
• Controller modal presentation
• Alert and action sheets
Architecture

WatchKit Extension

WKInterfaceController
• Interface properties
• Menu handling
• Controller navigation and paging
• Controller modal presentation
• Alert and action sheets
• System UI — text input, video, audio
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
}
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!

    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
}
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
    }
}
Resources and Data

Application and Extension
Resources

Static Resources

Multiple locations

• Watch App bundle
• WatchKit Extension bundle
Resources

Static Resources

Multiple locations

- Watch App bundle
- WatchKit Extension bundle
class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
}
Resources

Static Resources

class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!

    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
}
Resources
Static Resources

class InterfaceController: WKInterfaceController {
    @IBOutlet weak var appImage: WKInterfaceImage!
    @IBOutlet weak var extImage: WKInterfaceImage!
    override func willActivate() {
        super.willActivate()
        appImage.setImageNamed("ApplicationImage")
        extImage.setImageNamed("ExtensionImage")
    }
}
class InterfaceController: WKInterfaceController {
    ...
    appImage.setImageNamed("ApplicationImage")
    let image = UIImage(named: "ExtensionImage")
    extImage.setImage(image)
}
class InterfaceController: WKInterfaceController {

    ...

    appImage.setImageNamed("ApplicationImage")
    let image = UIImage(named: "ExtensionImage")
    extImage.setImage(image)
}
Data
Local data storage
Data

Local data storage

Document folder
- Non-purgeable
- Not restored
Data

Local data storage

Document folder
  • Non-purgeable
  • Not restored

Caches folder
  • Purgeable
func saveData(data: NSData, fileName: String) {
    let fileManager = NSFileManager.defaultManager()
    guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory,
        inDomains: .UserDomainMask).first else {
        return
    }
    let url = documentDirectory.URLByAppendingPathComponent(fileName)
    data.writeToURL(url, atomically: true)
}
func `saveData`(data: NSData, fileName: String) {
    let fileManager = NSFileManager.defaultManager()
    guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory,
        inDomains: .UserDomainMask).first else { return }
    let url = documentDirectory.URLByAppendingPathComponent(fileName)
    data.writeToURL(url, atomically: true)
}
func saveData(data: NSData, fileName: String) {
    let fileManager = NSFileManager.defaultManager()
    guard let documentDir = fileManager.URLsForDirectory(.DocumentDirectory, inDomains: .UserDomainMask).first else { return }
    let url = documentDirectory.URLByAppendingPathComponent(fileName)
    data.writeToURL(url, atomically: true)
}
Data

Media

Application

• Plays media files
• Records audio to a file
Data
Media

Application
• Plays media files
• Records audio to a file

Extension
• Downloads media files
• Reads recorded audio files
Data

Media

Must use a shared container
Data Media

Must use a shared container

Enable App Groups in Xcode

- Extension
- Application
func recordAudio(name: String) {

    let fileManager = NSFileManager.defaultManager()
    let container = fileManager.containerURLForSecurityApplicationGroupIdentifier("group.myapp")!

    let fileName = name.stringByAppendingPathExtension("mp4")!
    let audioFileURL = container.URLByAppendingPathComponent(fileName)

    self.presentAudioRecordingControllerWithOutputURL(audioFileURL, ...)
func recordAudio(name: String) {

    let fileManager = NSFileManager.defaultManager()
    let container =
        fileManager.containerURLForSecurityApplicationGroupIdentifier("group.myapp")!

    let fileName = name.stringByAppendingPathExtension("mp4")!
    let audioFileURL = container.URLByAppendingPathComponent(fileName)

    self.presentAudioRecordingControllerWithOutputURL(audioFileURL, ...
Data

Transferring data
Data

Transferring data

NSURLSession
Data

Transferring data

NSURLSession
WatchConnectivity
Data
NSURLSession
Data

NSURLSession

Direct access to internet

• http:// and https://
Data

NSURLSession

Direct access to internet
• http:// and https://

Background uploads and downloads
• Extension may not be running
• Downloaded files must be copied
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration.
            .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self,
                             delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
}
class Downloader : NSObject, URLSessionDownloadDelegate {

    lazy var session: URLSession = self.createDownloadSession()

    func createDownloadSession() -> URLSession {
        let config = NSURLSessionConfiguration
            .backgroundSessionConfigurationWithIdentifier("Downloader")
        return URLSession(configuration: config, delegate: self, delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume() 
    }
}
class Downloader : NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration
            .backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self, delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
}
class Downloader: NSObject, NSURLSessionDownloadDelegate {

    lazy var session: NSURLSession = self.createDownloadSession()

    func createDownloadSession() -> NSURLSession {
        let config = NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier("Downloader")
        return NSURLSession(configuration: config, delegate: self, delegateQueue: nil)
    }

    func download(url: NSURL) {
        let task = session.downloadTaskWithURL(url)!
        task.resume()
    }
}
func restart() {
    session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask, didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory, inDomains: .UserDomainMask).first else { return }
    let cacheURL = cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)
    do { try fileManager.copyItemAtURL(location, toURL: cacheURL) }
catch { print(error) }
}
func restart() {
    session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask,
    didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory,
        inDomains: .UserDomainMask).first else { return }
    let cacheURL = cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)
    do {
        try fileManager.copyItemAtURL(location, toURL: cacheURL)
    } catch {
        print(error)
    }
}
func restart() {
    session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask, didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory, inDomains: .UserDomainMask).first else {
        return
    }
    let cacheURL = cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)
    do {
        try fileManager.copyItemAtURL(location, toURL: cacheURL)
    } catch {
        print(error)
    }
}
func restart() {
    session = createDownloadSession()
}

func URLSession(NSURLSession, downloadTask: NSURLSessionDownloadTask, 
    didFinishDownloadingToURL location: NSURL) {
    let fileManager = NSFileManager.defaultManager()
    guard let cacheDir = fileManager.URLsForDirectory(.CachesDirectory, 
        inDomains: .UserDomainMask).first else { return }
    let cacheURL = 
        cacheDir.URLByAppendingPathComponent(location.lastPathComponent!)
    do {
        try fileManager.copyItemAtURL(location, toURL: cacheURL)
    } catch {
        print(error)
    }
}
Data
WatchConnectivity framework

Watch ↔ Phone

• Share data
• Transfer files
• Talk to counterpart
Data

WatchConnectivity framework

Watch ↔ Phone

• Share data
• Transfer files
• Talk to counterpart

Introducing Watch Connectivity
Migration

watchOS 1 ➞ watchOS 2
Migration

watchOS 1
Migration
watchOS 1

WatchKit extension for watchOS 1
Migration

watchOS 1

WatchKit extension for watchOS 1

• Uses iOS Platform and SDK
Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone
Migration

watchOS 1

WatchKit extension for watchOS 1

• Uses iOS Platform and SDK
• Runs on iPhone
• Share framework with iOS application
Migration

watchOS 1

WatchKit extension for watchOS 1

- Uses iOS Platform and SDK
- Runs on iPhone
- Share framework with iOS application
- Image caching
Migration
watchOS 1

WatchKit extension for watchOS 1
• Uses iOS Platform and SDK
• Runs on iPhone
• Share framework with iOS application
• Image caching
• openParentApplication()
Migration
watchOS 2
Migration

watchOS 2

New watchOS platform and SDK
Migration

watchOS 2

New watchOS platform and SDK

Subset of iOS frameworks available
Migration

watchOS 2

New watchOS platform and SDK
Subset of iOS frameworks available
Include project frameworks
Migration
Reusability
Migration

Reusability

Same API

• Copy code
• Copy resources
Migration

Reusability

Same API

• Copy code
• Copy resources

Improvements

• UI responsiveness
• Independent operation
• New UI elements
• Animation
Migration
Reusability

Same API
• Copy code
• Copy resources

Improvements
• UI responsiveness
• Independent operation
• New UI elements
• Animation
Migration
Controllers
Migration

Controllers

Same

- Interface controller
- Glance controller
- Notification controller
Migration
Controllers

Same

- Interface controller
- Glance controller
- Notification controller

New

- Extension delegate
- Complication data source
Migration

Xcode

Existing project

• Add watchOS Application target
Migration

Xcode

Existing project
• Add watchOS Application target

New project
• Create iOS App with WatchKit App
Migration

Xcode

Existing project
• Add watchOS Application target

New project
• Create iOS App with WatchKit App
New APIs in WatchKit for watchOS 2

Forest Hill WatchKit Engineer
WKExtensionDelegate

Callback methods for app lifecycle
WKExtensionDelegate

Callback methods for app lifecycle

iOS

UIApplicationDelegate
WKExtensionDelegate

Callback methods for app lifecycle

iOS

UIApplicationDelegate

watchOS 2

WKExtensionDelegate
WKExtensionDelegate
applicationDidFinishLaunching

Called once on launch
Perform app initialization
Setup notification observers
Warmup services

NOTE: App is not active yet
WKExtensionDelegate

applicationDidBecomeActive

Each time app becomes visually active

Activate timers

Update any state
WKExtensionDelegate
applicationWillResignActive

Only call before going to background
Prepare to be inactive
Save state
Disable running services, timers, etc
WKExtensionDelegate

App lifecycle

func applicationDidEnterLaunching()
func applicationDidBecomeActive()
func applicationWillResignActive()

Callbacks on app lifecycle only
WKExtensionDelegate

User Activity

func handleUserActivity(userInfo: [NSObject : AnyObject])

watchOS 1
    method on root WKInterfaceController

watchOS 2
    method on WKExtensionDelegate
User Activity
User Activity

*This property on WKExtension:

```swift
var rootInterfaceController: WKInterfaceController
```

will be coming in a future seed of watchOS 2
*This property on WKExtension:

```swift
var rootInterfaceController: WKInterfaceController

will be coming in a future seed of watchOS 2

func handleUserActivity(userInfo: [NSObject : AnyObject]) {
    let rootController = WKExtension.sharedExtension.rootInterfaceController
    rootController.popToRootController()
    rootController.doStuffForUserActivity(userInfo)
}
```
WKExtension
Analogous to UIApplication in iOS

Singleton
Encapsulates the running application
WKExtension
Analogous to UIApplication in iOS

Singleton
Encapsulates the running application.
In watchOS 2, we have WKExtension
WKExtension

Open URLs

WKExtension.sharedExtension().openSystemURL(systemURL)

Valid Schemes

• Phone
• SMS
• PassKit
Notifications
Remote Notifications
Remote Notifications
Remote Notifications
Notification Routing Rules

Criteria include
Notification Routing Rules

Criteria include

iPhone screen is locked
Notification Routing Rules

Criteria include

- iPhone screen is locked
- Apple Watch
  - on wrist
  - unlocked
Notifications

Notification handling when app is not active

Called on WKUserNotificationInterfaceController

```swift
func didReceiveRemoteNotification(remoteNotification: [NSObject : AnyObject],
    withCompletion: WKUserNotificationInterfaceType -> Void)
func didReceiveLocalNotification(localNotification: UILocalNotification,
    withCompletion: WKUserNotificationInterfaceType -> Void)
```
Local Notifications
Local Notifications
Local Notifications
Scheduling Local Notification

Sending message to phone
let message = ["request" : "fireLocalNotification"]
WCSession.defaultSession().sendMessage(message,
    replyHandler: nil,
    errorHandler: { error in
        print(error.localizedDescription)
    }
)
let message = 
"request" : "fireLocalNotification"

WCSession.defaultSession().sendMessage(message,
   replyHandler: nil,
   errorHandler: { error in
      print(error.localizedDescription)

   })

func session(session: WCSession, didReceiveMessage message:
[String : AnyObject]) {
   guard message["request"] as? String == "fireLocalNotification" else {
      return
   }

   let localNotification = buildLocalNotification()
   UIApplication.sharedApplication().scheduleLocalNotification(
      localNotification)

}
Andrew says
Beep! Beep!

Reply
Dismiss
Notification Actions

Launching from a notification action

```swift
func handleActionWithIdentifier(identifier: String,
                               forRemoteNotification: [NSObject : AnyObject])

func handleActionWithIdentifier(identifier: String,
                               forLocalNotification: UILocalNotification)
```
Inline notification text replies
Andrew says
Beep! Beep!

Reply
Dismiss
func suggestionsForResponseToActionWithIdentifier(identifier: String, remoteNotification: [String : AnyObject]) -> [String] {
    return ["Beep!", "Beep! Beep!"]
}
Notification Actions

Launching from a notification action with inline text input

```swift
func handleActionWithIdentifier(identifier: String,
    forRemoteNotification: [NSObject : AnyObject],
    withResponseInfo: [NSObject : AnyObject])

func handleActionWithIdentifier(identifier: String,
    forLocalNotification: UILocalNotification,
    withResponseInfo: [NSObject : AnyObject])
```
Notification Actions

Launching from a notification action with inline text input

```swift
func handleActionWithIdentifier(identifier: String,
                               forRemoteNotification: [NSObject : AnyObject],
                               withResponseInfo: [NSObject : AnyObject])

func handleActionWithIdentifier(identifier: String,
                               forLocalNotification: UILocalNotification,
                               withResponseInfo: [NSObject : AnyObject])

Inline text stored in

UIUserNotificationActionResponseTypedTextKey
```
Cancel

English

Español
Cancelar

¡Bip!

¡Bip! ¡Bip!
Notifications

Notification handling when app is active

On your WKExtensionDelegate:

```swift
func didReceiveRemoteNotification(userInfo: [NSObject : AnyObject])
func didReceiveLocalNotification(notification: UILocalNotification)
```
Modal Alerts
func presentAlertControllerWithTitle(title: String?,
message: String?,
preferredStyle: WKAlertControllerStyle,
actions: [WKAlertAction])
Modal Alert

Alert

Something happened

OK
Modal Alerts
.SideBySideButtonsAlert
Modal Alerts

ActionSheet

Nevermind

Delete Stuff?
Are you Sure?

Move Aside

Delete
Summary
Summary

New architecture in watchOS 2
Summary

New architecture in watchOS 2

WKExtensionDelegate
Summary

New architecture in watchOS 2
WKExtensionDelegate
New API
Summary

New architecture in watchOS 2
WKExtensionDelegate
New API
More to come…
More Information

Documentation
watchOS 2 Transition Guide
WatchKit Programming
developer.apple.com/library

Sample Code
Lister
WatchKit Catalogue
developer.apple.com/watchOS

Technical Support
Apple Developer Forums
Developer Technical Support

General Inquiries
Jake Behrens, watchOS
Frameworks Evangelist
behrens@apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing WatchKit for watchOS 2</td>
<td>Presidio</td>
<td>Tuesday 10:00AM</td>
</tr>
<tr>
<td>Building Watch Apps</td>
<td>Pacific Heights</td>
<td>Tuesday 4:30PM</td>
</tr>
<tr>
<td>WatchKit In-Depth, Part 2</td>
<td>Pacific Heights</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Creating Complications with ClockKit</td>
<td>Pacific Heights</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>WatchKit Tips and Tricks</td>
<td>Presidio</td>
<td>Friday 10:00AM</td>
</tr>
</tbody>
</table>