Building Great Workout Apps

HealthKit APIs and best practices

Session 235

Dash Brittain iOS Software Engineer
Jorge Moriñigo iOS Software Engineer
Background Running
Background Running

Workout Lifecycle
Background Running

Workout Lifecycle

Activity Rings

Workouts in iOS 10
Background Running

Workout Lifecycle

Activity Rings

Workouts in iOS 10

Best Practices
Getting Started
HKWorkoutSession
HKWorkoutSession

Motion and calorimetry for activity type
HKWorkoutSession

Motion and calorimetry for activity type
Activity rings
HKWorkoutSession

Motion and calorimetry for activity type
Activity rings
Show app on wake
Motion and calorimetry for activity type
Activity rings
Show app on wake
Background running
Background Running
Background Running

Process sensor data
Background Running

- Process sensor data
- Live feedback
Background Running

Process sensor data
Live feedback
Quickly update UI
// Background Running
// WatchKit Extension Info.plist

<key>WKBackgroundModes</key>
<array>
  <string>workout-processing</string>
</array>
Background Running
Conserve power
Background Running

Conserve power

Limit background work
Background Running

Conserve power

Limit background work

App may be suspended
Background Running

Conserve power

Limit background work

App may be suspended

Measure background work with tools
Background Running

Conserve power

Limit background work

App may be suspended

Measure background work with tools

- CPU Report in Xcode
Background Running
Conserve power

Limit background work
App may be suspended
Measure background work with tools
  • CPU Report in Xcode
  • Time Profiler in Instruments
Background Running
Conserve power

Limit background work
App may be suspended
Measure background work with tools
• CPU Report in Xcode
• Time Profiler in Instruments
• Backtrace log
Starting a Workout
Starting a Workout

Authorization
Starting a Workout

Authorization → Workout Configuration → Start Workout Session
Authorization
Authorization

Authorization to write

• Workouts
Authorization

Authorization to write
- Workouts

Authorization to read
- Energy burned
- Distance
- Heart rate
Authorization

Authorization to write
• Workouts

Authorization to read
• Energy burned
• Distance
• Heart rate

Getting the Most Out of HealthKit
// Workout Configuration

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor
// Workout Configuration

var workoutConfiguration = HKWorkoutConfiguration()

workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor
var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor
// Start Workout Session

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
healthStore.start(workoutSession)
// Start Workout Session

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
healthStore.start(workoutSession)
// Start Workout Session

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)

healthStore.start(workoutSession)
Demo

SpeedySloth: Starting a Workout

Dash Brittain
iOS Software Engineer
Data Collection and Control

Jorge Moriñigo
iOS Software Engineer
Ending and Saving Workout

Starting Workout

Data Collection and Control
Data Collection and Control

Starting Workout → Start Observing Samples → Start Observing Events → Ending and Saving Workout
Data Collection and Control

- Starting Workout
- Start Observing Samples
- Start Observing Events
- Running
- Ending and Saving Workout
Data Collection and Control

- Starting Workout
- Start Observing Samples
- Start Observing Events
- Running
- Paused
- Ending and Saving Workout
Observing Samples
Observing Samples

Open query for each data type
Observing Samples

Open query for each data type

Update your running totals
Observing Samples

Open query for each data type
Update your running totals
Display live data
Open query for each data type
Update your running totals
Display live data
Notify user of reached goals
Observing Samples
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObjects(from: [HKDevice.local()])
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObject(from: [HKDevice.local()])

let queryPredicate = CompoundPredicate(andPredicateWithSubpredicates: [datePredicate, devicePredicate])
// Observing Samples

let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObjects(from: [HKDevice.local()])

let queryPredicate = CompoundPredicate(andPredicateWithSubpredicates: [datePredicate, devicePredicate])

let updateHandler: (HKAnchoredObjectQuery, [HKSample]?, [HKDeletedObject]?, HKQueryAnchor?, NSError?) -> Void = { query, samples, deletedObjects, queryAnchor, error in
    // Process samples
}
// Observing Samples
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObjects(from: [HKDevice.local()])

let queryPredicate = CompoundPredicate(andPredicateWithSubpredicates:[datePredicate, devicePredicate])

let updateHandler: (HKAnchoredObjectQuery, [HKSample]?, [HKDeletedObject]? , HKQueryAnchor?, NSError?) -> Void = { query, samples, deletedObjects, queryAnchor, error in
  // Process samples
}

let query = HKAnchoredObjectQuery(type: quantityType,
  predicate: queryPredicate,
  anchor: nil,
  limit: HKObjectQueryNoLimit,
  resultsHandler: updateHandler)
// Observing Samples
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil,
    options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObjects(from: [HKDevice.local()])

let queryPredicate = CompoundPredicate(andPredicateWithSubpredicates:[datePredicate,
    devicePredicate])

let updateHandler: (HKAnchoredObjectQuery, [HKSample]?, [HKDeletedObject]?,
    HKQueryAnchor?,
    NSError?) -> Void = { query, samples, deletedObjects, queryAnchor, error in
    // Process samples
}

let query = HKAnchoredObjectQuery(type: quantityType,
    predicate: queryPredicate,
    anchor: nil,
    limit: HKObjectQueryNoLimit,
    resultsHandler: updateHandler)

query.updateHandler = updateHandler
// Observing Samples
let datePredicate = HKQuery.predicateForSamples(withStart: workoutStartDate, end: nil, options: .strictStartDate)

let devicePredicate = HKQuery.predicateForObjects(from: [HKDevice.local()])

let queryPredicate = CompoundPredicate(andPredicateWithSubpredicates:[datePredicate, devicePredicate])

let updateHandler: (HKAnchoredObjectQuery, [HKSample]?, [HKDeletedObject]?, HKQueryAnchor?, NSError?) -> Void = { query, samples, deletedObjects, queryAnchor, error in
  // Process samples
}

let query = HKAnchoredObjectQuery(type: quantityType, predicate: queryPredicate, anchor: nil, limit: HKObjectQueryNoLimit, resultsHandler: updateHandler)

query.updateHandler = updateHandler

healthStore.execute(query)
Observing Samples

Notify user of reached goal
Observing Samples

Notify user of reached goal

Play haptic to alert user
Observing Samples

Notify user of reached goal

Play haptic to alert user

Update UI to reflect reached goal
Observing Samples

Notify user of reached goal

Play haptic to alert user
Update UI to reflect reached goal

WKInterfaceDevice.current().play(.notification)
Observing Events
Observing Events

Timestamps to highlight point in session
Observing Events

Timestamps to highlight point in session

Some events created by your app
Observing Events

Timestamps to highlight point in session

Some events created by your app

Some events created by HealthKit
Observing Events

Timestamps to highlight point in session
Some events created by your app
Some events created by HealthKit

```swift
protocol HKWorkoutSessionDelegate {
    func workoutSession(_ workoutSession: HKWorkoutSession, didGenerate event: HKWorkoutEvent)
}
```
Laps and Markers
Laps and Markers

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Laps and Markers

Create to store in a workout

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Laps and Markers

Create to store in a workout

Timestamps for graphs and statistics

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Laps and Markers

Create to store in a workout

Timestamps for graphs and statistics

Laps for equal distance partitions

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Laps and Markers

Create to store in a workout
Timestamps for graphs and statistics
Laps for equal distance partitions
Markers for arbitrary points

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Laps and Markers

Create to store in a workout

Timestamps for graphs and statistics

Laps for equal distance partitions

Markers for arbitrary points

HKWorkoutEventType.lap
HKWorkoutEventType.marker
Pausing and Resuming
Pausing and Resuming

Users stop their activity
Pausing and Resuming

Users stop their activity
Pause workouts to save power and space
Pausing and Resuming

Users stop their activity
Pause workouts to save power and space
Ignore data while paused
Pausing and Resuming

Users stop their activity
Pause workouts to save power and space
Ignore data while paused
HealthKit responds with pause/resume events
Pausing and Resuming

Users stop their activity

Pause workouts to save power and space

Ignore data while paused

HealthKit responds with pause/resume events

After pause event, you will stop receiving new events
class HKHealthStore {
    func pause(_ workoutSession: HKWorkoutSession)
    func resume(_ workoutSession: HKWorkoutSession)
    ...
}
class HKHealthStore {
    func pause(_ workoutSession: HKWorkoutSession)
    func resume(_ workoutSession: HKWorkoutSession)
    ...
}

HKWorkoutEventType.pause
HKWorkoutEventType.resume
Motion Events for Running Workouts

HKWorkoutEventType.motionPaused
HKWorkoutEventType.motionResumed
Motion Events for Running Workouts

Watch detects when motion pauses or resumes

HKWorkoutEventType.motionPaused
HKWorkoutEventType.motionResumed
Motion Events for Running Workouts

Watch detects when motion pauses or resumes

Stop collecting data

HKWorkoutEventType.motionPaused
HKWorkoutEventType.motionResumed
Motion Events for Running Workouts

Watch detects when motion pauses or resumes
Stop collecting data
Don’t need to manual pause

HKWorkoutEventType.motionPaused
HKWorkoutEventType.motionResumed
Motion Events for Running Workouts

Watch detects when motion pauses or resumes

Stop collecting data

Don't need to manual pause

Only for `HKWorkoutActivityTypeRunning`

`HKWorkoutEventType.motionPaused`
`HKWorkoutEventType.motionResumed`
Demo

SpeedySloth: Data Collection and Control

Jorge Moriñigo
iOS Software Engineer
Workouts in Activity
Workouts in Activity

Activity workout list
Workouts in Activity

Activity workout list
Activity Move ring
Ending and Saving
Data Collection and Control

Ending and Saving

End Workout Session
Data Collection and Control

Ending and Saving

- End Workout Session
- Save Workout
- Add Samples to Workout
Ending an HK Workout Session
Ending an HKWorkoutSession

```javascript
healthStore.end(workoutSession)
```
Ending an HKWorkoutSession

```swift
func workoutSession(_ workoutSession: HKWorkoutSession,
                 didChangeTo toState: HKWorkoutSessionState,
                from fromState: HKWorkoutSessionState,
                 date: Date) {
    if toState == .ended {
        // Workout ended, time to save
    }
}
```
let workout = HKWorkout
// Save Workout

let workout = HKWorkout(activityType: config.activityType,
let workout = HKWorkout(activityType: config.activityType,
                         start: startDate,
// Save Workout

let workout = HKWorkout(activityType: config.activityType,
                         start: startDate,
                         end: endDate,
let workout = HKWorkout(activityType: config.activityType, 
start: startDate, 
end: endDate, 
workoutEvents: events, 
)
let workout = HKWorkout(activityType: config.activityType,
                         start: startDate,
                         end: endDate,
                         workoutEvents: events,
                         totalEnergyBurned: totalEnergyBurned,
// Save Workout

let workout = HKWorkout(
    activityType: config.activityType,
    start: startDate,
    end: endDate,
    workoutEvents: events,
    totalEnergyBurned: totalEnergyBurned,
    totalDistance: totalDistance,
let workout = HKWorkout(activityType: config.activityType,
start: startDate,
end: endDate,
workoutEvents: events,
totalEnergyBurned: totalEnergyBurned,
totalDistance: totalDistance,
metadata: [HKMetadataKeyIndoorWorkout: isIndoor])
let workout = HKWorkout(activityType: config.activityType,
                        start: startDate,
                        end: endDate,
                        workoutEvents: events,
                        totalEnergyBurned: totalEnergyBurned,
                        totalDistance: totalDistance,
                        metadata: [HKMetadataKeyIndoorWorkout: isIndoor])

healthStore.save(workout) { (success, error) in
    // Workout saved
}

// Save Workout
Add Samples to Workout
Add Samples to Workout

Creates an association
Add Samples to Workout

Creates an association
Query later for graphs
Add Samples to Workout

Creates an association
Query later for graphs
Activity Move ring credit
Add Samples to Workout

Creates an association
Query later for graphs
Activity Move ring credit
Match totals on HKWorkout
Add Samples to Workout

healthStore.add(samples, to: workout) { (success, error) in
    // Samples added
}
Demo

SpeedySloth: Ending and Saving

Dash Brittain
iOS Software Engineer
Workouts in iOS 10
Workouts in iOS 10

Workout apps have parent iPhone application
Workouts in iOS 10

Workout apps have parent iPhone application
WatchConnectivity for messaging when Apple Watch app is running
Workouts in iOS 10

Workout apps have parent iPhone application
WatchConnectivity for messaging when Apple Watch app is running
Background running
Workouts in iOS 10

Workout apps have parent iPhone application
WatchConnectivity for messaging when Apple Watch app is running
Background running
Start a workout from iPhone
Workouts in iOS 10

Workout apps have parent iPhone application
WatchConnectivity for messaging when Apple Watch app is running
Background running
Start a workout from iPhone
No user intervention on Apple Watch
Workouts in iOS 10

iPhone App

HKWorkoutConfiguration
Workouts in iOS 10

iPhone App

HKWorkoutConfiguration

Apple Watch App
Workouts in iOS 10

- iPhone App: HKWorkoutConfiguration
- Apple Watch App: Start HKWorkoutSession
Workouts in iOS 10

<key>UIBackgroundModes</key>
<array>
    <string>workout-processing</string>
</array>
// Starting Workout from iPhone

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)
var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)
// Starting Workout from iPhone

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)
// Starting Workout from iPhone

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)
// Starting Workout from iPhone

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)

// Apple Watch App

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
workoutSession.delegate = self
healthStore.start(workoutSession)
// Starting Workout from iPhone

var workoutConfiguration = HKWorkoutConfiguration()
workoutConfiguration.activityType = .running
workoutConfiguration.locationType = .outdoor
If the activation state is activated and the Watch app is installed, the following code snippet sets the workout configuration for running and outdoor location.

```swift
if wcSession.activationState == .activated && wcSession.isWatchAppInstalled {
    var workoutConfiguration = HKWorkoutConfiguration()
    workoutConfiguration.activityType = .running
    workoutConfiguration.locationType = .outdoor
}
```
// Starting Workout from iPhone

// iPhone App
if wcSession.activationState == .activated && wcSession.isWatchAppInstalled {
    var workoutConfiguration = HKWorkoutConfiguration()
    workoutConfiguration.activityType = .running
    workoutConfiguration.locationType = .outdoor
}
// Starting Workout from iPhone

iPhone App

if wcSession.activationState == .activated && wcSession.isWatchAppInstalled {
   var workoutConfiguration = HKWorkoutConfiguration()
   workoutConfiguration.activityType = .running
   workoutConfiguration.locationType = .outdoor
}
// Starting Workout from iPhone

// iPhone App
if wcSession.activationState == .activated && wcSession.isWatchAppInstalled {
    var workoutConfiguration = HKWorkoutConfiguration()
    workoutConfiguration.activityType = .running
    workoutConfiguration.locationType = .outdoor

    healthStore.startWatchApp(with: workoutConfiguration) { (success, error) in ...
}
}
// Starting Workout from iPhone

// iPhone App
if wcSession.activationState == .activated && wcSession.isWatchAppInstalled {
    var workoutConfiguration = HKWorkoutConfiguration()
    workoutConfiguration.activityType = .running
    workoutConfiguration.locationType = .outdoor

    healthStore.startWatchApp(with: workoutConfiguration) { (success, error) in
        ...
    }
}
// Starting Workout from iPhone
// Starting Workout from iPhone

let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)

workoutSession.delegate = self

healthStore.start(workoutSession)
// Starting Workout from iPhone

// Apple Watch App

func handle(_ workoutConfiguration: HKWorkoutConfiguration) {
    let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
    workoutSession.delegate = self
    healthStore.start(workoutSession)
}
// Starting Workout from iPhone

// Apple Watch App

class ExtensionDelegate: WKExtensionDelegate {
  func handle(_ workoutConfiguration: HKWorkoutConfiguration) {
    let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
    workoutSession.delegate = self
    healthStore.start(workoutSession)
  }
}
// Starting Workout from iPhone

class ExtensionDelegate: WKExtensionDelegate {

    func handle(_ workoutConfiguration: HKWorkoutConfiguration) {
        let workoutSession = HKWorkoutSession(configuration: workoutConfiguration)
        workoutSession.delegate = self
        healthStore.start(workoutSession)
    }
}
Demo

SpeedySloth: iPhone App

Jorge Moriñigo
iOS Software Engineer
Best Practices
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable

• Keep session running when losing connectivity
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable

• Keep session running when losing connectivity
• Use HealthKit distance when GPS is unavailable
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable

• Keep session running when losing connectivity
• Use HealthKit distance when GPS is unavailable

User should be able to begin workout on Apple Watch or iPhone
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable
  • Keep session running when losing connectivity
  • Use HealthKit distance when GPS is unavailable

User should be able to begin workout on Apple Watch or iPhone
Display workouts from other sources
Best Practices

Ensure Apple Watch app is functional when iPhone is unreachable

• Keep session running when losing connectivity
• Use HealthKit distance when GPS is unavailable

User should be able to begin workout on Apple Watch or iPhone

Display workouts from other sources

Don’t display deleted workouts
Summary

Background running
Summary

Background running
Contribute to Activity rings
Summary

Background running
Contribute to Activity rings
Start workout from Apple Watch or iPhone
More Information

<table>
<thead>
<tr>
<th>Session</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting the Most out of HealthKit</td>
<td>Wednesday</td>
</tr>
<tr>
<td>Health and Fitness with Core Motion</td>
<td>Thursday</td>
</tr>
<tr>
<td>Introducing HealthKit</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>Designing Accessories for iOS and OS X</td>
<td>WWDC 2014</td>
</tr>
<tr>
<td>What’s New in HealthKit</td>
<td>WWDC 2015</td>
</tr>
</tbody>
</table>