Introducing

iOS 9 Search APIs

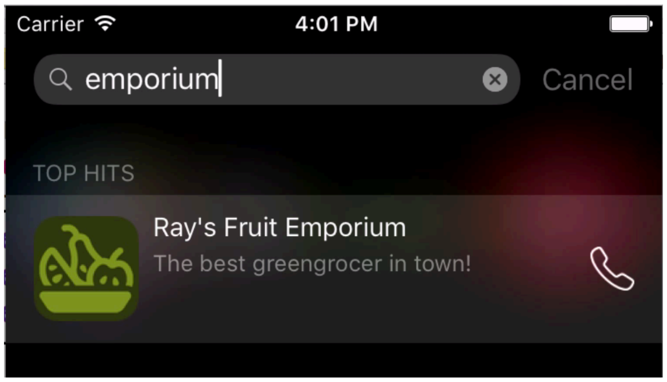
Hands-On Challenges

Introducing iOS 9 Search APIs Hands-On Challenges

Challenge 1: Indexing User Activities

In the tutorial, you discovered how to add the products in the GreenGrocer app to the Spotlight search index using user activity indexing. This is a powerful technique that allows you to search for screens within your app that don't necessarily have a data model associated with them.

In this challenge, you're going to use this same technique to add the **Store** page to the search index – allowing users to find Ray's Fruit Emporium right from Spotlight.



Hints

- Once again you'll need to prepare an `NSUserActivity` and set the appropriate properties to make it searchable.
- There is a property on `CSSearchableItemAttributeSet` that will allow the user to call the store right from the search result.

Solution

Remember that every user activity must have an ID which identifies the kind of activity that it represents. Open StoreViewController.swift and add the following at the top of file – just below the module imports:

```
import MobileCoreServices
import CoreSpotlight

let storeDetailsActivityID =
  "com.razeware.GreenGrocer.storeDetails"
```

This first adds a couple of module imports you'll need next, before defining a global constant in reverse-DNS notation that identifies the new activity you're going to create.

As you did in the tutorial, you need to create a function which creates the user activity, which you'll call from `viewDidLoad()`. Add the following extension to the bottom of the file:

```
extension StoreViewController {
  private func prepareUserActivity() -> NSUserActivity {
    let activity = NSUserActivity(activityType:
      storeDetailsActivityID)
    activity.eligibleForHandoff = true
    activity.eligibleForPublicIndexing = true
    activity.eligibleForSearch = true
    return activity
  }
}
```

This defines a private utility function that creates an NSUserActivity. You configure this user activity to be eligible for Handoff, search and public indexing. Notice that you've used the `storeDetailsActivityID` constant you defined above to create the activity.

Since `UIViewController` is a subclass of `UIResponder`, you can once again take advantage of its automatic handling of `NSUserActivity`, via the `userActivity` property. Add the following line to the end of `viewDidLoad()`:

```
userActivity = prepareUserActivity()
```

This uses the utility function you added to create a new user activity, and assigns it to the managed `userActivity` property.

Although you've now created this user activity, it doesn't contain any useful contextual info that Spotlight can use to index it. As you did in the tutorial, add the following method override in the extension you created previously:

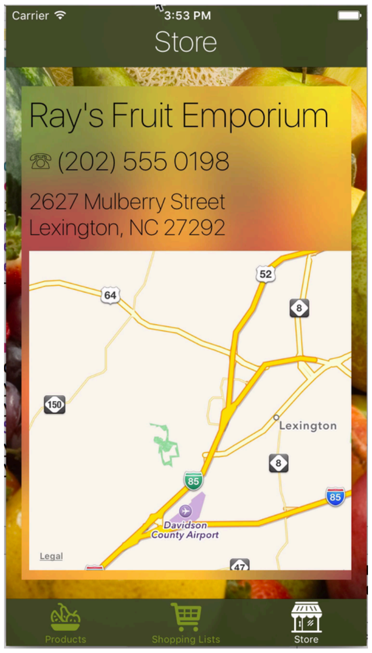
```
override func updateUserActivityState(activity: NSUserActivity) {
  // 1:
  activity.title = storeName
  activity.keywords = Set(["shopping", "fruit", "lists",
                           "greengrocer", storeName])
  // 2:
  let attributeSet = CSSearchableItemAttributeSet(itemContentType:
                                                  kUTTypeContact as String)
  attributeSet.contentDescription =
                                            "The best greengrocer in town!"
  attributeSet.phoneNumbers = [phoneNumber]
  attributeSet.longitude = coordinate.longitude
  attributeSet.latitude = coordinate.latitude
  // 3:
  attributeSet.supportsPhoneCall = true
  // 4:
  activity.contentAttributeSet = attributeSet
}
```

This overrides the `UIResponder` method `updateUserActivityState(_:)`, to provide the metadata required to make the user activity searchable:

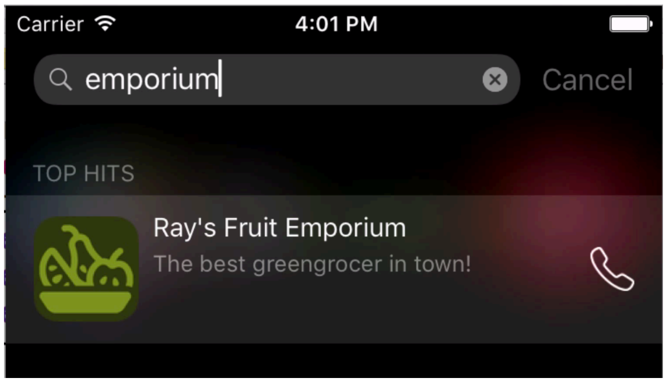
1. `NSUserActivity` has title and keywords properties which are used in indexing and displaying search results. You set these first.
2. For richer search results, you create a `CSSearchableItemAttributeSet`, and use some of the 150 properties available to annotate your user activity. `contentDescription` is used in the displayed search results, and the `phoneNumbers` property allows calling directly from the search result.
3. Setting `supportsPhoneCall` to `true` enables calling Ray's Fruit Emporium without having to open the app.
4. Finally you assign the newly created attribute set to the `contentAttributeSet` property on your user activity.

   Now you can build and run your app to check that it all works.

   Remember that the activity will only be indexed when you visit this view controller, so first, navigate to the **Store** screen:



Now you can head to the home screen, pull down the Spotlight search, and search for "**emporium**". You will see your new search result appearing:



You can also try searching for some of the other terms you specified to check that it all works as you would have expected.

If you tap the phone icon whilst running on a device, you'll be prompted to call Ray's Fruit Emporium.

Tapping the result itself will open the GreenGrocer app, but won't necessarily take you to the correct place. Don't worry – you'll discover how to tackle this problem in an upcoming video tutorial.