Article

Supporting universal links in your app

Prepare your app to respond to an incoming universal link.

Overview

When a user activates a universal link, the system launches your app and sends it an `NSUserActivity` object. Query this object to find out how your app launched and to decide what action to take.

To support universal links in your app:

1. Create a two-way association between your app and your website and specify the URLs that your app handles. See Supporting associated domains.
2. Update your app delegate to respond when it receives an `NSUserActivity` object with the `activityType` set to `NSUserActivityTypeBrowsingWeb`.

Warning

Universal links offer a potential attack vector into your app, so make sure to validate all URL parameters and discard any malformed URLs. In addition, limit the available actions to those that don't risk the user's data. For example, don't allow universal links to directly delete content or access sensitive information about the user. When testing your URL-handling code, make sure your test cases include improperly formatted URLs.

Update your app delegate to respond to a universal link

When the system opens your app as the result of a universal link, your app receives an `NSUserActivity` object with an `activityType` value of `NSUserActivityTypeBrowsingWeb`. The activity object's `webpageURL` property contains the HTTP or HTTPS URL that the user accesses. Use `NSURLComponents` APIs to extract the components of the URL. See the examples that follow.

This example code shows how to handle a universal link in macOS:

```
func application(_ application: NSApplication,
                 continue userActivity: NSUserActivity,
                 restorationHandler: @escaping ([NSUserActivityRestoring]) -> Void) -> Bool
{
    // Get URL components from the incoming user activity.
    guard userActivity.activityType == NSUserActivityTypeBrowsingWeb,
        let incomingURL = userActivity.webpageURL,
        let components = NSURLComponents(url: incomingURL, resolvingAgainstBaseURL: true) else {
        return false
    }

    // Check for specific URL components that you need.
    guard let path = components.path,
    let params = components.queryItems else {
        return false
    }
    print("path = \(path)")

    if let albumName = params.first(where: { $0.name == "albumname" } )?.value,
        let photoIndex = params.first(where: { $0.name == "index" })?.value {
        print("album = \(albumName)")
        print("photoIndex = \(photoIndex)")
        return true

    } else {
        print("Either album name or photo index missing")
        return false
    }
}
```

This example code shows how to handle a universal link in iOS and tvOS:

```
func application(_ application: UIApplication,
                 continue userActivity: NSUserActivity,
                 restorationHandler: @escaping ([UIUserActivityRestoring]?) -> Void) -> Bool
{
    // Get URL components from the incoming user activity.
    guard userActivity.activityType == NSUserActivityTypeBrowsingWeb,
        let incomingURL = userActivity.webpageURL,
        let components = NSURLComponents(url: incomingURL, resolvingAgainstBaseURL: true) else {
        return false
    }

    // Check for specific URL components that you need.
    guard let path = components.path,
    let params = components.queryItems else {
        return false
    }
    print("path = \(path)")

    if let albumName = params.first(where: { $0.name == "albumname" } )?.value,
        let photoIndex = params.first(where: { $0.name == "index" })?.value {

        print("album = \(albumName)")
        print("photoIndex = \(photoIndex)")
        return true

    } else {
        print("Either album name or photo index missing")
        return false
    }
}
```

If your app has opted into Scenes, and your app is not running, the system delivers the universal link to the `scene(_:willConnectTo:options:)` delegate method after launch, and to `scene(_:continue:)` when the universal link is tapped while your app is running or suspended in memory.

```
func scene(_ scene: UIScene, willConnectTo
           session: UISceneSession,
           options connectionOptions: UIScene.ConnectionOptions) {

    // Get URL components from the incoming user activity.
    guard let userActivity = connectionOptions.userActivities.first,
        userActivity.activityType == NSUserActivityTypeBrowsingWeb,
        let incomingURL = userActivity.webpageURL,
        let components = NSURLComponents(url: incomingURL, resolvingAgainstBaseURL: true) else {
        return
    }

    // Check for specific URL components that you need.
    guard let path = components.path,
        let params = components.queryItems else {
        return
    }
    print("path = \(path)")

    if let albumName = params.first(where: { $0.name == "albumname" })?.value,
        let photoIndex = params.first(where: { $0.name == "index" })?.value {

        print("album = \(albumName)")
        print("photoIndex = \(photoIndex)")
    } else {
        print("Either album name or photo index missing")
    }
}
```

This example code shows how to handle a universal link in watchOS:

```
func handle(_ userActivity: NSUserActivity)
{
    // Get URL components from the incoming user activity.
    guard userActivity.activityType == NSUserActivityTypeBrowsingWeb,
        let incomingURL = userActivity.webpageURL,
        let components = NSURLComponents(url: incomingURL, resolvingAgainstBaseURL: true) else { retur

    // Check for specific URL components that you need.
    guard let path = components.path,
    let params = components.queryItems else { return }
    print("path = \(path)")

    if let albumName = params.first(where: { $0.name == "albumname" } )?.value,
        let photoIndex = params.first(where: { $0.name == "index" })?.value {
        print("album = \(albumName)")
        print("photoIndex = \(photoIndex)")
    } else {
        print("Either album name or photo index missing")
    }
}
```

Note

In watchOS, a Safari-like interface is available for apps such as Messages and Mail. For other apps, when the user clicks a universal link that points to content in a companion app that the user doesn't have installed, the system notifies the user to view the URL on their iPhone.