

# NETWORKING WITH URLSession



**HANDS-ON CHALLENGES**

## Networking with URLSession

Audrey Tam

Copyright ©2017 Razeware LLC.

### Notice of Rights

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written permission of the copyright owner.

### Notice of Liability

This challenge and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use of other dealing in the software.

### Trademarks

All trademarks and registered trademarks appearing in this book are the property of their own respective owners.

# Challenge #8: Refresh Auth Token After Expiry

By Audrey Tam

Open the StormpathNotes app in the Challenge Starter folder. This is **loosely** based on the app described in the [Stormpath tutorial](#), but doesn't use their authentication framework. This version retains the three view controllers, but bypasses the Login view controller's text fields, and doesn't use the Register view controller, or implement the Notes view controller's Logout button.

## Login to Stormpath Notes

- Open **LoginViewController.swift**, and locate the `basicAuth` line in `login(_:)`. Register for a Stormpath Notes account (Page 2 of the demo playground), and replace the user and password strings here, with what you used to register.

**Note:** Please don't use eugeneivanov's details, or other people doing the demo won't see what they should.

`basicAuth` is a `URLCredential`, so it's stored in the app's keychain. `login(_:)` also sets the `username` and `password` properties of `apiManager`, because its `sendLoginRequest(completion:)` method doesn't use the `URLCredential`.

`sendLoginRequest(completion:)` contains the same code as the demo playground, but now when it saves the tokens and token type, it uses [Jason Rendel's KeychainWrapper](#) to store them in the app's keychain. This means you can exit the app, then run it later, with the same tokens.

## GET Notes Task

In **APIManager.swift**, locate `getNotes(getCompletion:)`. Again, this is similar to the demo code — I left the handler code there, commented out — but now the session has a custom delegate. The delegate methods are just the same `URLSessionDataDelegate` methods we used for the download task in HalfTunes: `didReceive response` and `didReceive data`.

didReceive data does the usual conversion of data into notes.

If didReceive response receives status code 401 unauthorized, it calls sendRefreshRequest(task:).

Locate sendRefreshRequest(task:): it creates and sends a refresh request, receives and stores new auth tokens, then creates a new request from the unauthorized request, replacing the access token with the new token.

There are three TODOs for you to do:

- To create refreshRequest.httpBody with the refresh token, enter these two lines:

```
let requestBody = ["grant_type": "refresh_token",  
    "refresh_token": refreshToken!] as JSONDictionary  
refreshRequest.httpBody = try? JSONSerialization.data(  
    withJSONObject: requestBody, options: [])
```

- In the data task's handler, to create a new request from the task's current request, add this line:

```
var request = task.currentRequest!
```

- On the next line, to replace the old access token, add this line:

```
request.setValue("\(self.tokenType!) \(self.accessToken!)",  
    forHTTPHeaderField: "authorization")
```

## Test Refresh Request

Build and run the app. Add some notes. Then wait at least an hour: you can stop the app in Xcode, but don't delete it from the simulator, or reset the simulator.

After an hour or more, the auth token expires. In the storyboard, **move the start arrow** so the initial controller is the navigation controller of NotesViewController. This bypasses the login screen, and runs the GET notes task right away.

Taking care to use the same simulator, build and run the app. In the debug console, you should see:

```
did receive response: send refresh request  
did receive response: OK  
<your notes>
```

Build and run again, and this time the GET task succeeds:

```
did receive response: OK  
<your notes>
```