EXERCISES

Track: **Developers**
Tutorial: **Easy FHIR Persistence with FireKit for iOS**
Speaker: Ryan Baldwin

The speaker will be in the hands-on area to answer your questions!

As we learned during the tutorial, FireKit allows you to easily persist FHIR resources to a local Realm database. During the hands-on session of the Easy FHIR Persistence with FireKit for iOS exercise we will explore working with FireKit and, to a small degree, Restivus. Specifically, we will learn how to:

1. Create a new FHIR resource (Patient) and how to persist it locally
2. Edit an existing, locally persisted FHIR patient (created in step 1)
3. Create request to POST, PUT, and GET resources to/from a remote FHIR server

We'll perform the above activities by "filling in the blanks" of an example application called *FHIRPatients*, an open source application created to help developers get started with FireKit. *FHIRPatients* is a simple application which allows you to create and update Patients, upload those patients to a remote FHIR server, search for Patients on a remote FHIR server, and finally download a patient from a remote FHIR server to the iOS device.

Requirements
This lab requires *Xcode 9+*, and optionally *Carthage* (for dependency resolution). If you do not have *Carthage* and do not wish to install it (or if the Internet is flaky) talk to me and I can provide you with the lab source code with all dependencies. Familiarity with *Realm* (https://www.realm.io) is beneficial but should not be necessary.

Before You Start Coding
To get started you'll want to do the following:

1. Clone the example repository at https://github.com/ryanbaldwin/FHIRDevDays-FireKit-Lab.git
2. Follow the readme for setup.

Alternatively, talk to me and I'll provide you the source code with the required dependencies.

Note: The tutorial follows a logical sequence, where we start locally before extending functionality to integrate with a remote FHIR Server. It is recommended you follow the steps sequentially. Each part of the lab has further instructions and hints documented inside the actual Swift file you will be editing.

1. Save a Patient

The *FHIRPatients* application allows you to create and edit patients on an iOS device. In the first part of this lab you will complete the *save()* function in the *PatientModel.swift* file (located at *FHIRPatients/PatientModel.swift*).

1. Convert the in-memory patient information provided by the *PatientModel* (that is, *familyName*, *givenName*, *dateOfBirth*, *gender*, *telecoms*, and optionally *image*) to a *FireKit.Patient*, and save that *Patient* to the local Realm
   (*Remember: All Realm persistence/mutations but be performed inside a Realm write transaction block; i.e.* ***realm.write { }****)*
2. Modify the *save()* function such that it can update the patient if it already exists.

2. Upload a Patient

The FHIRPatients application allows you to upload a Patient via the super fancy upload button on the Patient Details screen. In the second part of this lab you will complete the *uploadPatient(completion:)* in the *PatientModel.swift* file.

1. In the *HttpPatient.swift* file (*/FHIRPatients/http/HttpPatient.swift*) create 2 Restivus requests for uploading a Patient: One POST request (for adding to the remote FHIR Server) and one PUT request (for updating to the remote FHIR server).
2. Complete the *uploadPatient(completion:)* function such that it uses the appropriate request (POST vs. PUT) depending on whether or not the patient has been uploaded to the server or not.
   (*Note:* For the purpose of this lab we will let the FHIR server assign the *id* of the patient, and use that as an indicator as to whether or not the Patient has been uploaded)

3. Download a Patient

The *FHIRPatients* application allows you to download a patient from a remote FHIR server. In the final part of this lab you will complete the *downloadPatient(completion:)* function in the *PatientModel.swift*.

1. In the *HttpPatient.swift* file, create a Restivus request for getting a specific Patient from the FHIR server (using the patient's *id*).
2. In the *PatientModel.swift*, complete the *downloadPatient(completion:)* request such that it downloads the patient from the FHIR server and updates the patient in the local Realm

Have fun, and remember to ask for help if you get stuck! There's a lot here and you may have questions, and I'm happy to answer!.

If you wish to view a completed example, you can reference the non-lab version of this application at https://github.com/ryanbaldwin/FHIRPatients

*This exercise will not be evaluated and no prizes are attached to any result. The exercise can only be used at HL7 FHIR DevDays 2017.