KMM Transition Notes

FEBRUARY 21, 2021

Chorus Enterprise

Authored by: PAGE TYLER

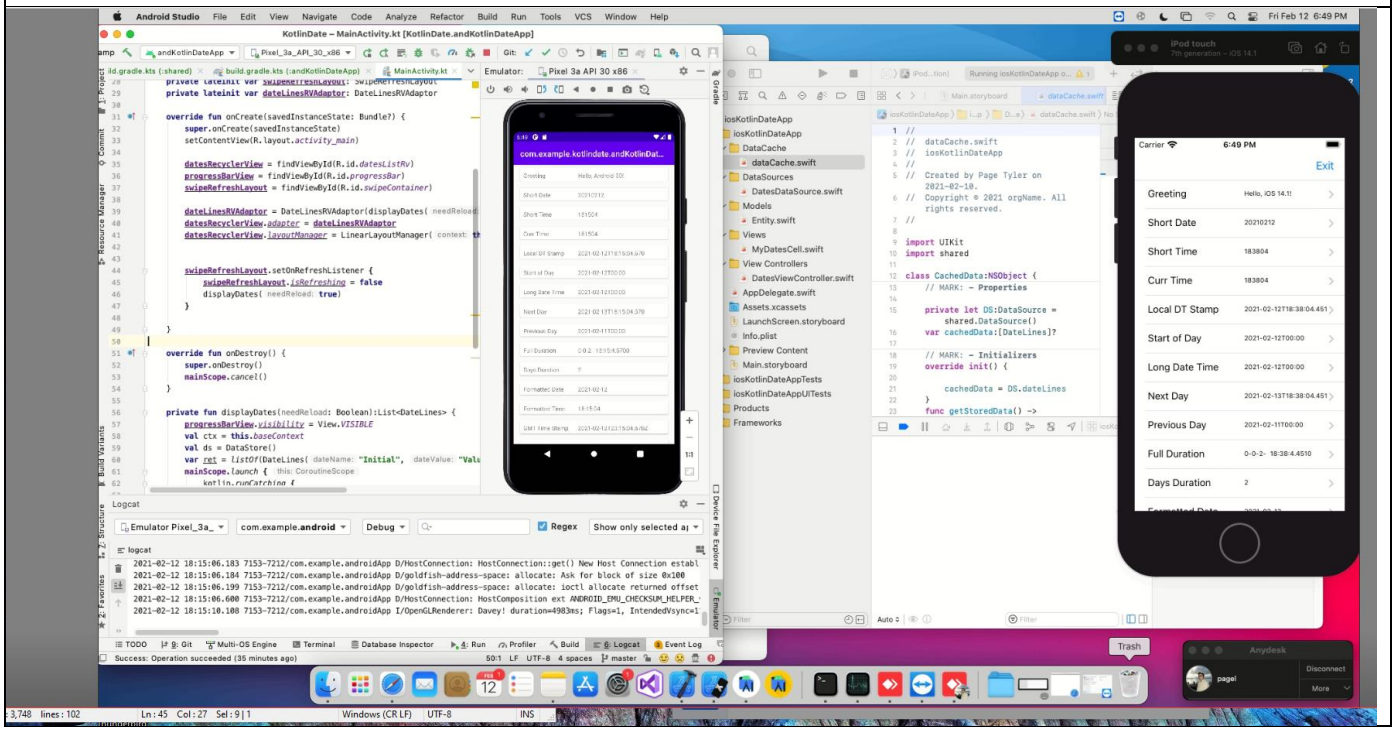
Overview

Making the Journey to the KMM environment

When I made the decision to move my application to the Kotlin KMM environment I had know idea of what the iOS environment would bring. What I found in XCode and Swift was an environment was very similar to what I was use to in Android Studio. This document is presented as an overview of the process I used to make this transition.

The application that is referred to in this document is the KotlinDateTime example program. I used this small sample program to first address the number one issue that keep my program from being a multiple platform program and that was date processing. I also used it to introduce myself to the new environments of Swift and XCode. Please remember I am going into these new tools with know knowledge at all so if you have better suggestions on anything I present here then please add then to this post. I purposefully used the simplest Android Kotlin code in this example. I am trying to make it easily readable for someone coming the other way iOS to Android.

Here is what I have found



Preparing for KMM

Before you get started here is a list of things to consider

1. **Getting an iOS system,**
    a. **Purchasing a new or used iOS system:**
        i. **Based on everything I have read getting an old or used system can be very problematic. What I read was that a lot of issues can arise from old iOS and Apple hardware. These can cause development issues and slow processing.**
        ii. **Buying a new iOS system is the best option if you can afford it. This was not an option for my budget. This is a very expensive option.**
        iii. **Installing a VM version, I spent a week trying to get this to work. Then once getting it up and running I was faced with the inability to install anything on the system because it was not a valid installation of said system. Not being an iOS systems developer, I gave up at this point.**
        iv. **Getting a cloud version of iOS (Recommended). This is the option that I found most helpful. It took the job of learning how to setup an iOS system out of my hands and put it in the hands of an expert. This left me with the task that I new how to do well.**
    b. **Getting a cloud version of iOS Options available,**
        i. **Flow (www.flow.swiss) This was way too expensive but if you can afford it this may be a faster solution for your development effort.**
        ii. **Rent the hardware in the cloud at (www.macminivault.com) this is the middle of the road option. It cost more than I was willing to pay but it is a modestly priced option. It is more than likely faster and more available then the one I selected.**
        iii. **Rent a Dedicated Mac Virtual Machine (Recommended) (https://xcodeclub.com/). This is the option I selected. The price is very reasonable and the speed is acceptable. It is slow on start**

up but once it is up and you have your programs running the speed is quite acceptable.

1. If you go this way and you plan to use Android Studio please ask to have your system enabled for VM.
2. You will need to download and install Android Studio.
3. If you are moving files in from another system you will need to place them in a subdirectory where you will need to perform a Linux [chmod -R 777 *.*] command on them via the terminal. You will do this remove them from read only mode and allow write access.

2. Learning how to use the iOS system
    a. Keyboard differences
        i. If you are connecting from a windows system here are a few tips;
            1. Copy and Paste are controlled by the windows key. This is still the hardest one to get used to.
            2. If you don't know the key code there is an icon or menu item to match what you are use to.
            3. The same comment go for the iOS user coming to the android system.
        ii. Connecting from a Linux system things should be very familiar as iOS rides on top of Linux.
    b. File system navigation differences
        i. iOS uses the Finder window to traverse the file system. I find it to be very limited in relationship to File Explorer but there are some neat tricks at the bottom of the right click pop-up menu.
        ii. If you know Linux script then you will be able to open any directory in a terminal window. There you will be able issue any script command.
    c. It would be good to have an Apple account going forward. You are going to need one eventually to install your app on an iOS system. So, you might as well request one now. You can do it by requesting an iTunes account.

Preparing Android Studio

Download and install Android Studio

1. **Download Minimum Android Studio 4.1.2 or better on your iOS system from this link Get android studio**
2. **Install it to your iOS system**
3. **Open it and install the Kotlin Multiplatform Mobile (KMM) Plug-in**
4. **Install all other plugins that your application may need**

You are now ready to develop in the iOS environment

1. **As for Kotlin and Java nothing has really changed. Some menu items have moved around to conform to iOS standards.**
2. **While it is the idea of JetBrains to have 1 development IDE that is just not a reality yet. You can run your app in both the Emulator and the Simulator from the A-Studio IDE it is not a development environment for the iOS part yet.**
3. **If you are OK with doing Swift code in a plain text editor then this may work for you. I was not comfortable with that.**
4. **I suggest that you open the iosKotlinDateApp.xcodeproj in the XCode IDE.**
5. **Here you will be able to edit, test, and debug your iOS app.**
6. **You will see I had to set the system back to default Storyboard behavior to eliminate the programmatic override that comes with the KMM template.**

I hope this helps anyone who is trying this for themselves. Please add any updates or comments to the discussion board where we can all learn together.