CallLogger Application

Written by Jong Ho Baek
icanmobile@gmail.com

CallLogger application(https://github.com/icanmobile/mobile-basic-projects.git) displays the call histories of Android phone. The structure of this application is composed of MVVM (Model-View-Viewmodel) design pattern with Android Jetpack architecture components, Dagger2, and RxJava. The target android sdk version of this application is 28 and minimum sdk version is 21 and it was tested on Galaxy S9 device.

Application components are as below.

1. Activity class

- It Includes the Injected ViewModel class object to receive the call histories using LiveData.
- It Includes a RecyclerView and Adapter to display the call histories.
- The call histories will be updated on RecyclerView using RxJava.
- Related source location: Package "com.icanmobile.calllogger.ui.main"

2. ViewModel class

- It Includes the injected DataManager class object to receive the call histories using RxJava.
- It Includes a Deque and LiveData to manage call histories.
- It Includes loadCallLogs() method to bring up to 50 call histories.
- It Includes updateCallLogs() method to bring the updated call histories only after the date and time of the last call.
- The updated call histories will be merged with previous call histories using Deque up to 50.
- Related source location: Package "com.icanmobile.calllogger.ui.main"

3. DataManager class

- This class is the bridge that connects ViewModel and data source.
- It Includes the injected CallLogManager class object to receive the call histories using callback method.
- Related source location: Package "com.icanmobile.calllogger.data"

4. CallLogManager class

- It brings the call histories from ContentResolver.
- It includes loadCallLogs() method to bring up to 50 call histories from ContentResolver.
- It includes updateCallLogs() method to load the updated call histories only after the date and time of the last call.
- Related source location: Package "com.icanmobile.calllogger.data.calllog"

5. CallLog class

- It is the model class for call history.
- Related source location: Package "com.icanmobile.calllogger.data.calllog"

Questions

1. What are the libraries or frameworks you have chosen? and why?

I chose Android Jetpack architecture components (ViewModel and LiveData), Dependency injection framework (Dagger2) and Reactive extensions (RxJava) for this application.

- Android architecture components are a collection of libraries that help you design robust, testable, and maintainable applications. For this application, I used ViewModel and LiveData components. The ViewModel class supports to save and modify call histories and LiveData updates the call histories of Activity.
- Dependency injection is basically providing the objects that an object needs (its dependencies) instead of having it construct them itself. It's a very useful technique for testing, since it allows dependencies to be mocked or stubbed out.
- Reactive Extension is reactive based and it makes threading, error handling, and cancellation easy.

Using the above frameworks and libraries, I focused on creating a scalable application.

2. If you had more time, what would you have done better?

If I had more time, I would like to add the functionalities as below.

- Implement Interfaces for activity and data source classes.
- Use data binding library to bind UI components in the layouts to data sources.
- Implement Unit Test and Android UI Test.
- If I have to support multiple activities and fragments, I would like to implement abstract classes for the activities and fragments.

3. What are some of the key architecture considerations?

My architecture considerations are as below.

- Scalable application architecture using Android Jetpack architecture components such as ViewModel and LiveData, Dependency injection, and Reactive extensions.
- Support to update the call histories right after phone call using broadcast receiver and a Deque in ViewModel class.