Tingle Version 4

Thor V.A.N. Olesen

February 25, 2016

1 Important design choices

I have done some design choices in regards to communication between fragments, resource encapsulation, user interface, Singleton database and MVC architecture.

Firstly, I have separated the concerns of the software functionality by using the Model-View-Controller (MVC) design pattern. The Views are composed of the XML layout files in the res layout package, the Controllers are composed of the fragments and activities in the Controller package and the Models are composed of the Thing and ThingsDB in the Model package. This has been done to to avoid overlapping between the different functionalities of the program and thus achieve an overall modular design.

Secondly, to allow the fragments, TingleFragment and ListFragment, to communicate with their host activity, TingleActivity, I have defined a listener interface in the fragments that is implemented by the activity. Also, the fragment UI components are self-contained, modular and define their own layout and behaviour to allow reuse. The fragment captures the interface implementation during its onAttach() lifecycle method and can then call the Interface methods in order to communicate with the activity. As opposed to using intents like in Tingle V3 where only activities were used. During runtime, the TingleActivity will receive events triggered from calling the interface method in the fragments (e.g. going back or showing the list of items). As a result, all Fragment-to-Fragment communication is done through the associated Activity and not directly.

Thirdly, I have followed some of the conventions outlined in the book such as field naming conventions (e.g. "mButton") and encapsulated all string resource files in a separate XML file to avoid hard coding strings in the code (easy localization). Many of the string resource files are also used to make pop-up toasts that inform the user of e.g. adding and deleting items. The Singleton pattern is also used to control access to the in-memory database with Thing resources

2 User Interface

The User Interface is composed of the many XML layout files in the res layout package and include a layout for both portrait and landscape screen orientation. In this regard, I have chosen to make different layout files based on the orientation because some of the widgets are not relevant. By way of example, I do not include a Back button in the user interface composed in landscape mode. This is based on the fact that the main page is already shown in the fragment of the left container. Also, I do not include the "See Items" button used to redirect to the list of items in the ListFragment because they are already shown in landscape mode. Thus, I have designed the user interface for supporting both portrait and landscape orientation by using different layouts and widgets.

3 Testing

I have tested the application manually by running an emulator on my computer and sometimes running through the user experience on a real Android device. Manual testing was easy to do and gave me fast qualitative information on how my app behaved in different work flows. Specifically, I tested the usability (user experience), interface (testing of buttons) and compatibility (different screen sizes and orientations). However, as the Tingle app becomes increasingly complex, I will most likely consider using automated UI tests due to the time it takes to test all important workflows and features. Also, I could have made some unit tests for the Model to clarify that items are stored correctly. However, I have chosen not to look into this due the relatively small complexity of the application so far.

4 Problems

The main challenges in the Tingle version 4 were to replace the activities by fragments, associate the fragments through interfaces and most importantly to support different layouts based on the orientation of the device. I spent a lot of time implementing the interfaces used to communicate between the fragments and the host activity and moved the layout inflation in the onCreate() methods to the onCreateView() methods used in fragments. Also, I had to make some deliberate choices about what to show on the screen based on the orientation (e.g. the back button is not shown in landscape mode). I had an issue with refreshing the list of items in landscape mode after deleting an item without changing orientation that I could not fix.