Practice:

Add a ViewModel to Dessert Clicker

1. Before you begin

Introduction

Dessert Clicker works with inline state and data. In this exercise, you will remove the inline state, data, and logic out of the MainActivity and move it to a `ViewModel`.

Abstracting app logic away from the view and into a ViewModel is a modern practice for Android development. This practice offers the following benefits:

- The code becomes more readable for other developers.
- The code becomes more testable.
- Multiple developers can work simultaneously on an app without interfering with other developers' work.

The solution code is available at the end, but try to solve the exercises before you check the answers. Consider the solution as one way to implement the app.

Prerequisites

- The Android Basics in Compose coursework through the ViewModel and State in Compose codelab

What you'll need

- A computer with internet access and Android Studio installed
- The solution code for the Dessert Clicker app

What you'll build

In these practice problems, you will improve the architecture of the Dessert Clicker app by adding a `ViewModel` to handle the data and app logic.

The practice problems are split into sections where you will complete the following steps individually:

- Update and add the necessary dependencies.
- Create a `ViewModel` class.

2. Download the starter code

Starter code URL:

https://github.com/google-developer-training/basic-android-kotlin-compose-training-dessert-clicker

Branch name with starter code: main

1. In Android Studio, open `the basic-android-kotlin-compose-training-dessert-clicker` folder.
2. Open the Dessert Clicker app code in Android Studio.

3. Set up dependencies

1. Add the following variable to your project `build.gradle` file:

```
buildscript {
   ext {
       ...
       lifecycle_version = '2.5.1'
   }
}
```

2. Add the following dependency to the `app/build.gradle` file:

```
dependencies {
    ...implementation "androidx.lifecycle:lifecycle-viewmodel-
compose:$lifecycle_version"
}
```

4. Create a UI state class

Currently, the `DessertClickerApp()` composable in the `MainActivity` contains the data and state that drive the UI.

Create a data class that holds all the necessary data for the UI. The data within this class replaces the data that the `DessertClickerApp()` composable currently manages.

5. Create a ViewModel

Create a `ViewModel` class using the Jetpack ViewModel component. You use the ViewModel to manage the UI state.

6. Relocate the app logic and data to the ViewModel

Relocate the logic from the `MainActivity` to the ViewModel and make the UI state data accessible using the UI state class you created. Delete all the data and state management logic from `MainActivity`.

Try to approach this task on your own. If necessary, refer to the ViewModel and State in Compose codelab for guidance.

7. Call the ViewModel

Use the data and methods that the ViewModel provides to drive the UI in the `MainActivity`.

8. Solution code

Starter code URL:

https://github.com/google-developer-training/basic-android-kotlin-compose-training-dessert-clicker

Branch name with starter code: viewmodel