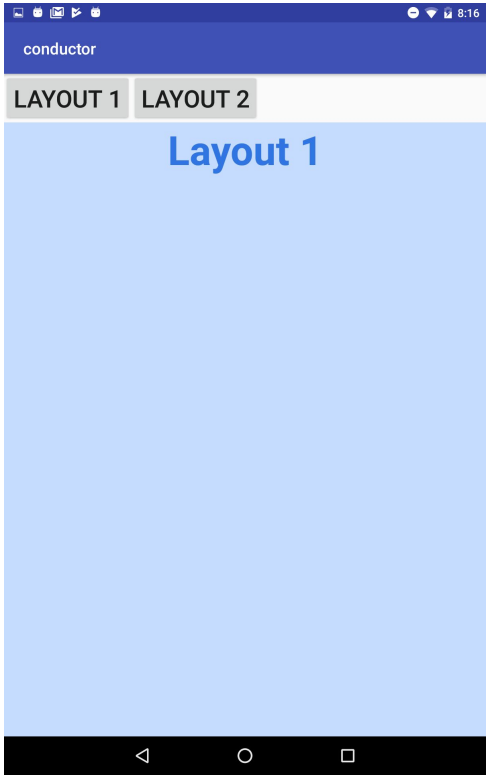
Android - Conductor

- Conductor is a view-handling framework which serves as an alternative to Fragments.
- History
  - Developed by Square because of difficulty working with Fragments.
  - Square programmers complained that Fragments:
    - had complicated lifecycles.
    - were difficult to test and debug.
    - were error-prone due to asynchronous transactions.
  - Read Advocating Against Android Fragments, written by Square programmers.
  - In response, they developed Conductor, which includes features such as simpler lifecycles, easy integration, simple backstack handling, and built-in transition handling.

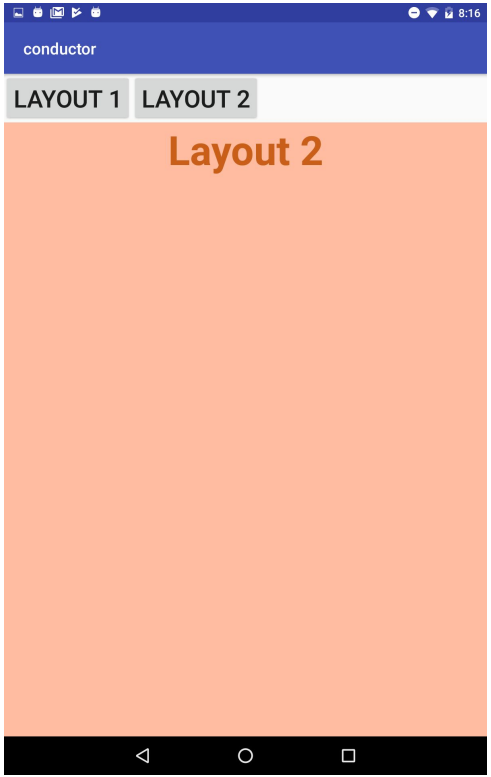
- Conductor’s main components:
  - **Controller**: Their job is equivalent to that of Fragments, except they feature simpler lifecycles. They are View-wrapper classes that make a layout visible to the user.
  - **Router**: It is responsible for handling the Controller transactions, or in other words, it initiates the process that brings forward new Controllers so that the user can interact with new layouts. The Router also handles the backstack.
  - **ControllerChangeHandler**: This is responsible for the actual switching of Controllers. When a Router completes a transaction, it delegates the job of bringing the Controller to the foreground to the ControllerChangeHandler component. This component can be customized to have different animations during the transition.
  - **ControllerTransaction**: This defines data about the transaction.
- For simple implementations, only a Router and a Controller are necessary to take advantage of this framework, which will be demonstrated in this tutorial.

How to use Conductor (as seen in the example)

- The example code makes use of only a Router and two Controllers.
- In install, insert the Conductor dependency in your app's **build.gradle** file. Use the version specified on this page.
- To initialize the router, we call `Conductor.attachRouter()` , in MainActivity's `onCreate()` callback.
- For each layout, we'll need a subclass of Controller to expand that layout. In the example code, we use **Layout1Controller** and **Layout2Controller** to expand **controller_1_layout.xml** and **controller_2_layout.xml** respectively
- We initialize the Router object and then call `setRoot()` , an instance method of Router, in MainActivity's `onCreate()` callback to push the first Controller to the Router.
- To push any sequential Controllers while adding onto the backstack, we call `pushController()` , an instance method of Router.
- We override MainActivity's onBackPressed() callback to ignore the default action when the back button is pressed if the Router object has a backstack available to it.



Pressing **Layout 1** button pushes Layout1Controller to the router



Pressing **Layout 2** button pushes Layout2Controller to the router

References

- Advocating Against Android Fragments
- Conductor github

Exercise

Add a button that makes a third layout appear.

Fill that layout with any content you would like.