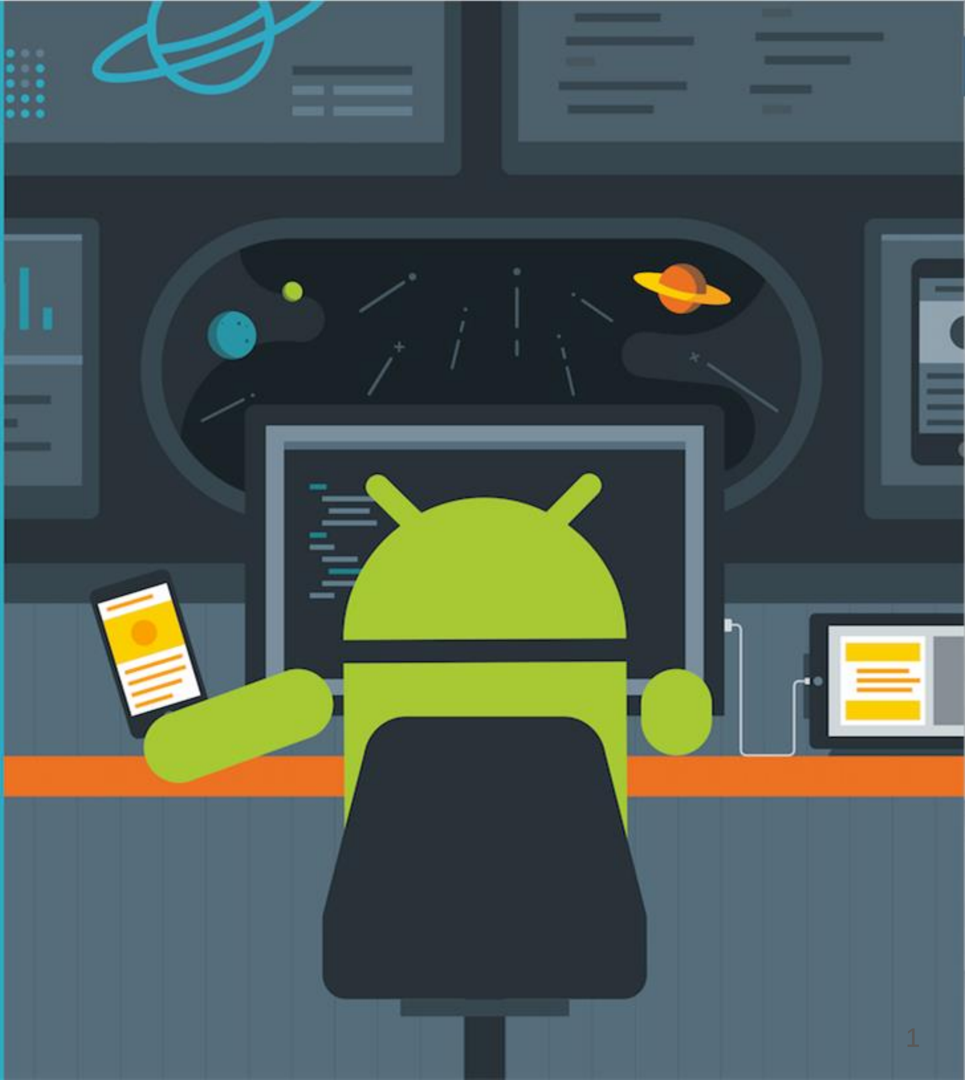
Advanced Android Development

Fragments

Lesson 1



1.2 Fragment lifecycle and communications

Using the Fragment lifecycle and communicating with an Activity

Contents

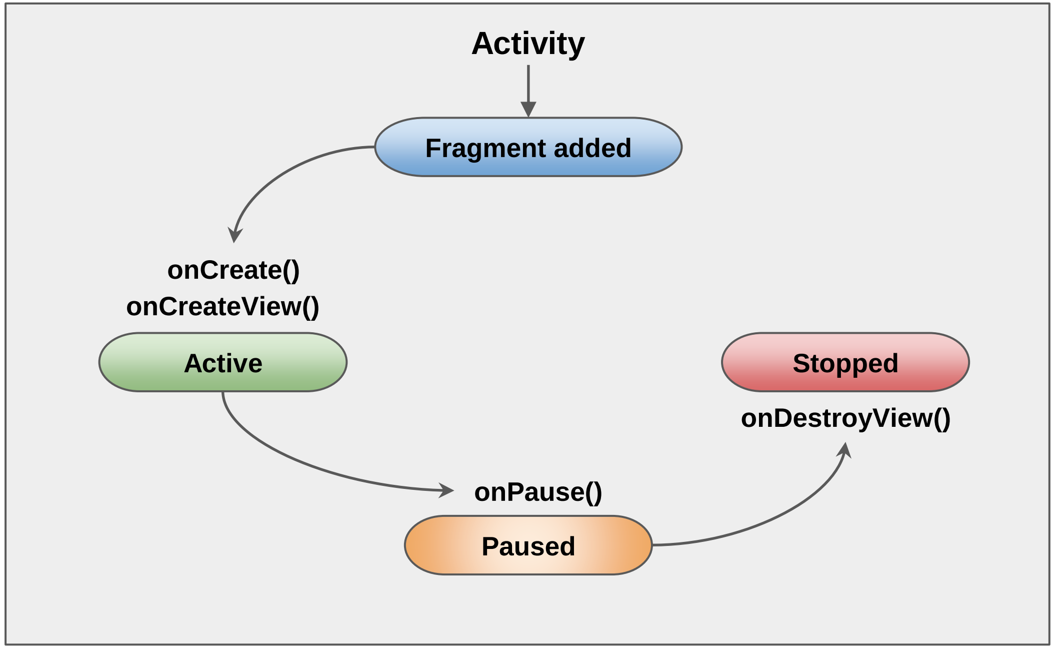
Understanding the Fragment lifecycle

The Fragment lifecycle

- `Fragment` lifecycle is similar to `Activity` lifecycle
- Lifecycle callbacks define how `Fragment` behaves in each state

Fragment lifecycle states

- `Activity` (host) adds `Fragment`
- States that a `Fragment` can be in:
    - Active (or resumed)
    - Paused
    - Stopped



How Activity affects Fragment lifecycle

Activity state	Fragment callbacks triggered	Fragment lifecycle
Created	• onAttach() • onCreate() • onCreateView() • onActivityCreated()	Fragment is added and its layout is inflated
Started	• onStart()	Fragment is active and visible
Resumed	• onResume()	Fragment is active and ready for user interaction

How Activity affects Fragment lifecycle

Activity state	Fragment callbacks triggered	Fragment lifecycle
Paused	● onPause()	Fragment is paused because Activity is paused
Stopped	● onStop()	Fragment is stopped and no longer visible
Destroyed	● onDestroyView() ● onDestroy() ● onDetach()	Fragment is destroyed

Using Fragment lifecycle callbacks

Callbacks to make Fragment active

- `onCreate()`: Initialize `Fragment` components and variables
- `onCreateView()`: Inflate the `Fragment` XML layout

Initialize the Fragment in onCreate()

Override `onCreate(Bundle savedInstanceState)`:

- System calls `onCreate()` when the `Fragment` is created
- Initialize `Fragment` components and variables (preserved if the `Fragment` is paused and resumed)
- Always include `super.onCreate(savedInstanceState)` in lifecycle callbacks

Display layout in onCreateView()

Override `onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)`:

- Inflate XML layout—required if `Fragment` has a UI
- System calls this method to make `Fragment` visible
- Must return the root `View` of `Fragment` layout or `null` if the `Fragment` does not have a UI

More Fragment lifecycle callbacks

- onAttach(): Called when Fragment is first attached to Activity
- onPause(): Called when Activity is paused
- onResume(): Called by Activity to resume a visible Fragment
- onActivityCreated(): Called when Activity onCreate() method has returned
- onDestroyView(): Called when View previously created by onCreateView() is detached from Fragment

Callback for final initialization

`onActivityCreated()`: Called when the `Activity` `onCreate()` method has returned

- Called after `onCreateView()` and before `onViewStateRestored()`
- Retrieve views or restore state
- Use `setRetainInstance()` to keep `Fragment` instance when `Activity` is recreated

Use onDestroyView

Use `onDestroyView()` to perform action after `Fragment` is no longer visible

- Called after `onStop()` and before `onDestroy()`
- `View` that was created in `onCreateView()` is destroyed
- A new `View` is created next time `Fragment` needs to be displayed

Using Fragment methods and Activity context

Use the Activity context

When `Fragment` is active or resumed:

- Use `getActivity()` to get the `Activity` that started the fragment
- Find a `View` in the `Activity` layout:

```
View listView = getActivity().findViewById(R.id.list);
```

Call methods in the Fragment

Get `Fragment` by calling [`findFragmentById()`]() on `FragmentManager`:

```
ExampleFragment fragment = (ExampleFragment)
     getFragmentManager().findFragmentById(R.id.example_fragment);
// ...
mData = fragment.getSomeData();
```

Use the back stack (1)

- Add `Fragment` to back stack: [`addToBackStack(null)`]

```
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

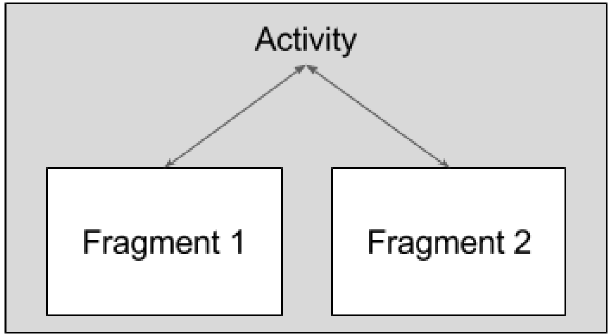
Use the back stack (2)

- Host `Activity` maintains back stack even after `Fragment` is removed
- User can navigate back to `Fragment` with Back button

Fragment communications

Activity and Fragment communications

- `Activity` can send data to `Fragment` and receive data from `Fragment`
- All `Fragment`-to-`Fragment` communication is done through host `Activity`



Send data to Fragment

In `Fragment` `newInstance()` factory method:

- Create [`Bundle`]
- Use [`setArguments(Bundle)`] to supply `Fragment` construction arguments

Fragment factory method

`newInstance()` factory method:

```
public static SimpleFragment newInstance(int choice) {
    SimpleFragment fragment = new SimpleFragment();
    Bundle arguments = new Bundle();
    arguments.putInt(CHOICE, choice);
    fragment.setArguments(arguments);
    return fragment;
}
```

Use the Fragment factory method

Include data (such as `mRadioButtonChoice`) in call to `Fragment` from `Activity`:

```
SimpleFragment fragment =
        SimpleFragment.newInstance(mRadioButtonChoice);
```

Use data from Activity in Fragment

- Before drawing Fragment View, get arguments from Bundle using getArguments()
- Use onCreate() or onCreateView()

```
if (getArguments().containsKey(CHOICE)) {
    mRadioButtonChoice = getArguments().getInt(CHOICE);
    // ...
}
```

Retrieve data from Fragment

- In `Fragment`:
  - Define interface (such as a listener) with callback method(s)
  - Override [`onAttach()`]() to retrieve interface implementation (check if the `Activity` implements the interface)
  - Call interface method to pass data as parameter
- In `Activity`:
  - All `Activity` classes using `Fragment` must implement interface
  - Use the interface callback method(s) to retrieve data

Define interface and callback methods

In `Fragment`: Define interface (such as a listener) with callback method (such as `onRadioButtonChoice()`)

```
interface OnFragmentInteractionListener {
    void onRadioButtonChoice(int choice);
}
```

Retrieve interface implementation

Retrieve interface implementation from `Activity`:

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnFragmentInteractionListener) {
        mListener = (OnFragmentInteractionListener)
context;
    } else {
        // ...
```

Call interface method to pass data

```
public void onCheckedChanged(RadioGroup group, int checkedId) {
    // ...
    switch (index) {
        case YES: // User chose "Yes."
            mListener.onRadioButtonChoice(YES);
            break;
        case NO: // User chose "No."
            mListener.onRadioButtonChoice(NO);
            break;
        // ...
    }
```

Implement interface in Activity

`Activity` must implement the interface defined in the `Fragment`:

```
public class MainActivity extends AppCompatActivity
 implements SimpleFragment.OnFragmentInteractionListener {
```

Use the callback in the Activity

`Activity` can then use `onRadioButtonChoice()` callback:

```
@Override
public void onRadioButtonChoice(int choice) {
    mRadioButtonChoice = choice;
    // Use mRadioButtonChoice in Activity
    // ...
}
```

What's next?

- Concept chapter: 1.2 Fragment lifecycle and communications
- Practical: 1.2 Communicating with a Fragment

END