ANDROID DEVELOPMENT

FRAGMENT

Tan Cher Wah (isstcw@nus.edu.sg)

Agenda

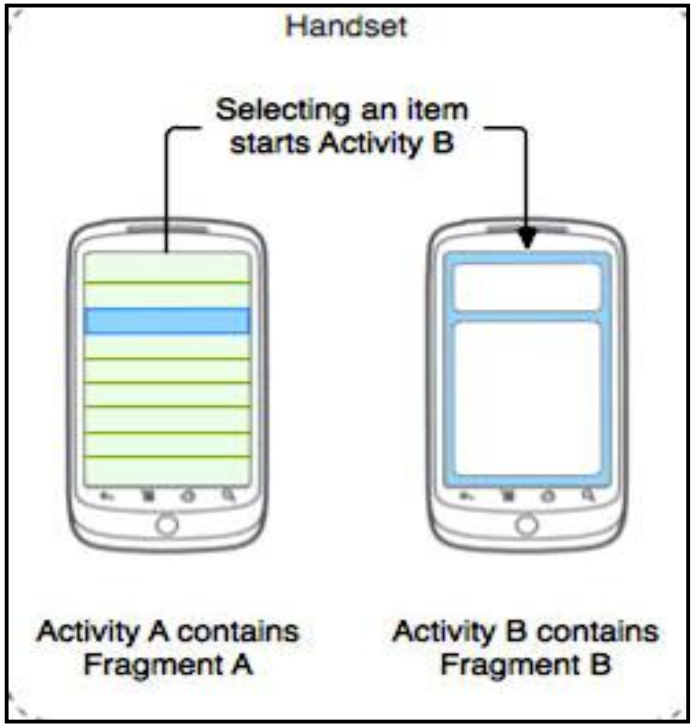
- What is a Fragment?
- Fragment’s Lifecycle (within an Activity)
- Sample views using fragments

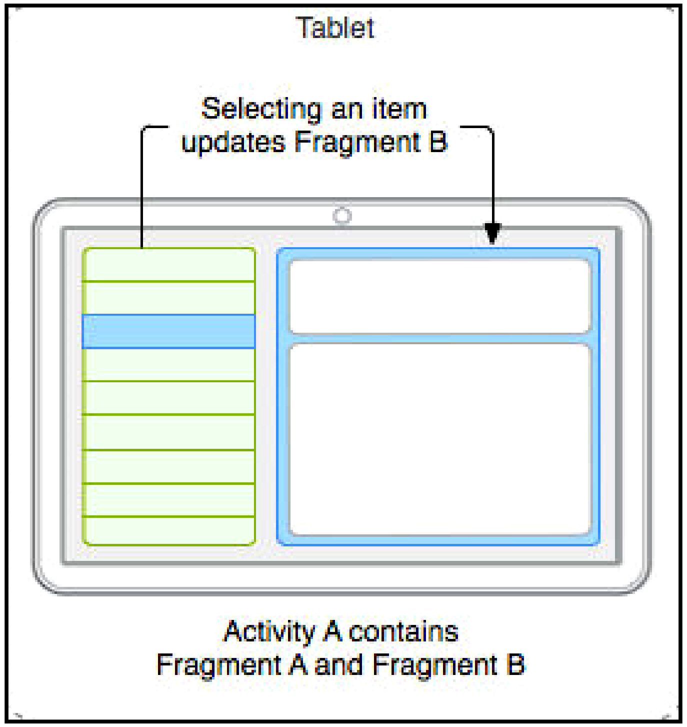
What is a Fragment?

- A modular and reusable component
- Runs inline within a normal Activity
- Has its own lifecycle (separate from the Activity it runs in)
- Usage
  - Define in layout file using <fragment> tag
  - Instantiate and insert into layout dynamically

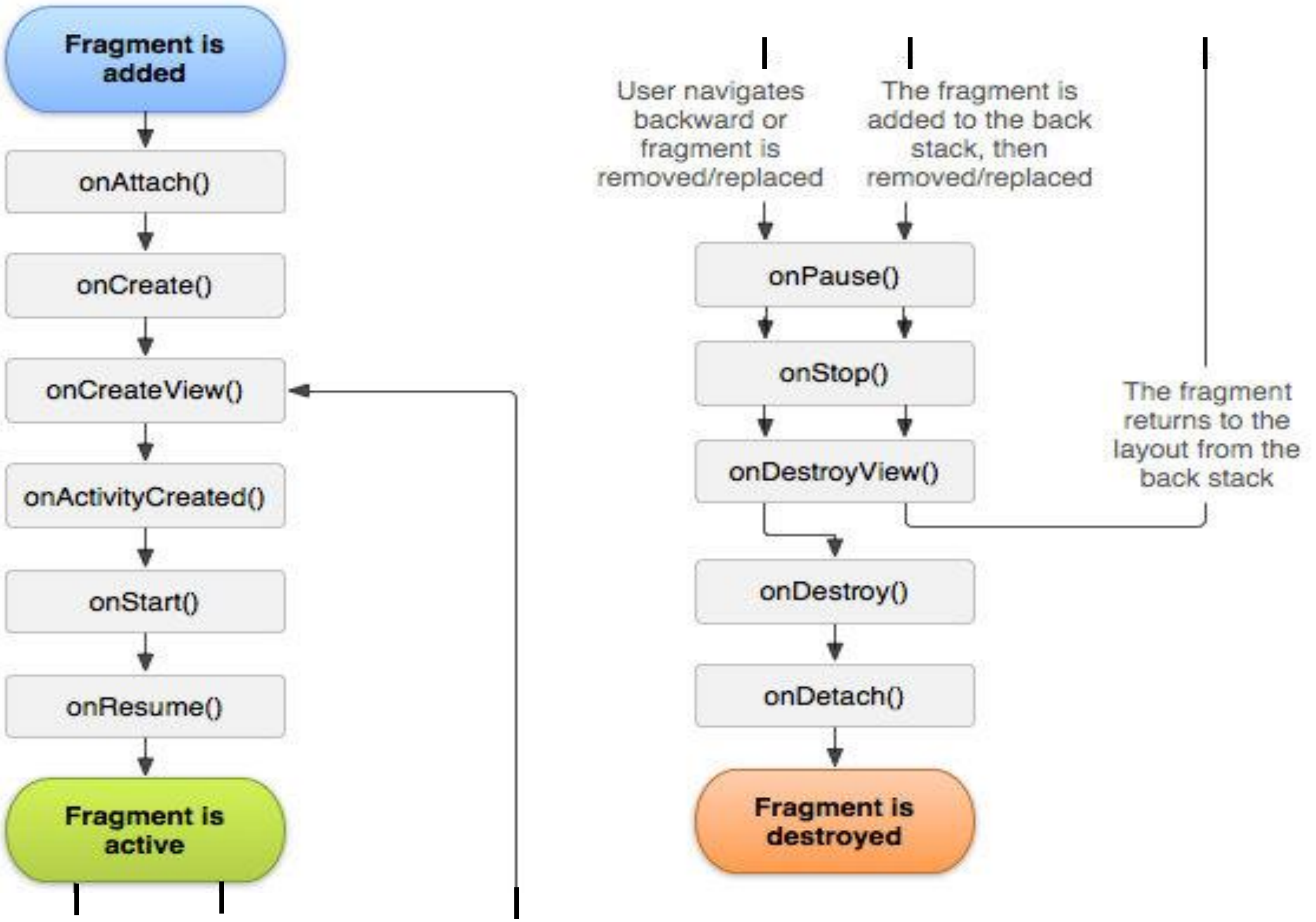
Why Fragments?

- Tablets allows more **Views** due to their larger screens
- Reuse our **Views** in Handsets and Tablets


Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B

Fragment Life-cycle



Fragment Life-cycle

onAttach	The fragment instance is associated with an activity instance.The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.
onCreate	Fragment is created. The onCreate()method is called after the onCreate()method of the activity but before the onCreateView() method of the fragment.
onCreateView	The fragment instance creates its view hierarchy. In the onCreateView() method the fragment creates its user interface. Here you can inflate a layout via the inflate() method call of the Inflator object passed as a parameter to this method. In this method you should not interactive with the activity, the activity is not yet fully initialized.

Fragment Life-cycle

onActivityCreated	At this point, view can be accessed with the findViewById()method. In this method you can instantiate objects which require a Context object.
onStart	Called when fragment becomes visible.
onResume	Fragment becomes active.
onPause	Fragment is visible but becomes not active anymore, e.g., if another activity is animating on top of the activity which contains the fragment.
onStop	Fragment becomes not visible.
OnDestroyView	Called when the host Activity has stopped, or the Activity has removed the Fragment (detached)
OnDestroy	Not guaranteed to be called by the Android platform. Discretion of the Android System.

1. Statically insert fragment into activity

- res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/listView1"
        android:name="com.example.fragmentexample.ListFrag"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</android.support.constraint.ConstraintLayout>
```

Layout for Fragment body

- res/layout/list_frag.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Similar layout resource

- res/layout/row.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/nameView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/addrView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Begins from an activity

- Fragments must exist within an Activity

```
public class MainActivity extends AppCompatActivity {
    ArrayList<Customer> customers = new ArrayList<>();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        customers.add(new Customer("Tan", "Sentosa Cove"));
        customers.add(new Customer("Wong", "Upper Thomson"));

        setContentView(R.layout.activity_main);
    }

    public ArrayList<Customer> getCustomers() {
        return customers;
    }
}
```

```
class Customer extends HashMap<String, Object> {
    Customer(String name, String address) {
        this.put("name", name);
        this.put("address", address);
    }
}
```

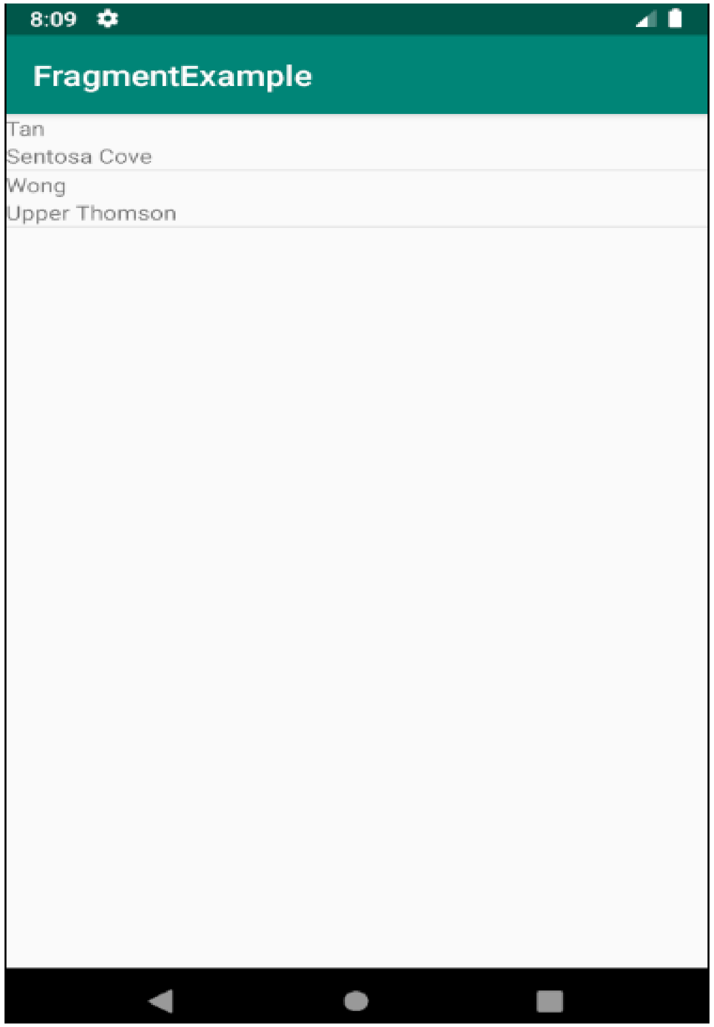
Creating a Fragment

- Inherits Fragment class
- Inflates fragment’s layout to get view

```
public class ListFrag extends Fragment {
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        ArrayList<Customer> customers = ((MainActivity)getActivity()).getCustomers();

        View v = inflater.inflate(R.layout.list_frag, container, false);
        ListView list = v.findViewById(R.id.listView1);

        list.setAdapter(new SimpleAdapter(getActivity(), customers,
                R.layout.row, new String[]{ "name", "address" },
                new int[]{R.id.nameView, R.id.addrView}));

        return v;
    }
}
```

End Result

- Our fragment within the Main Activity



Tan
Sentosa Cove
Wong
Upper Thomson

Making our fragment more reusable

- Implements an interface defined in our Fragment

```java
public class MainActivity extends AppCompatActivity implements ListFrag.IListFrag {
    ArrayList<Customer> customers = new ArrayList<>();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        customers.add(new Customer("Tan", "Sentosa Cove"));
        customers.add(new Customer("Wong", "Upper Thomson"));

        setContentView(R.layout.activity_main);
    }

    @Override
    public void onAttachFragment(Fragment fragment) {
        if (fragment instanceof ListFrag) {
            ListFrag frag = (ListFrag) fragment;
            frag.setParent(this);
        }
    }

    public ArrayList<Customer> getCustomers() {
        return customers;
    }
}
```

Making our fragment more reusable

- Get a “callback” pointer to the host
- Call the implemented getCustomers() found within the host

```
public class ListFrag extends Fragment {
    IListFrag callback;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        ArrayList<Customer> customers = callback.getCustomers();

        View v = inflater.inflate(R.layout.list_frag, container, false);
        ListView list = v.findViewById(R.id.listView1);

        list.setAdapter(new SimpleAdapter(getActivity(), customers,
                R.layout.row, new String[]{ "name", "address" },
                new int[]{R.id.nameView, R.id.addrView}));

        return v;
    }

    public interface IListFrag {
        public ArrayList<Customer> getCustomers();
    }

    public void setParent(IListFrag callback) {
        this.callback = callback;
    }
}
```

Dynamically insert a fragment

- Within an activity, use FragmentManager to add/replace fragments

```java
@Override
public void onStart()
{
    super.onStart();

    Bundle bundle = new Bundle();
    bundle.putString("meta", "x101");
    bundle.putSerializable("customers", customers);

    Fragment frag = new ListFrag();
    frag.setArguments(bundle);

    FragmentManager fm = getSupportFragmentManager();
    FragmentTransaction trans = fm.beginTransaction();
    trans.replace(R.id.frag1, frag);
    trans.commit();
}
```

To retrieve data passed to Fragment

- Use getArguments() to retrieve data from caller

```
public class ListFrag extends Fragment {

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
            ViewGroup container, Bundle savedInstanceState)
    {
        super.onCreateView(inflater, container, savedInstanceState);

        String meta = null;
        List<Customer> customers = null;

        Bundle bundle = getArguments();
        if (bundle != null)
        {
            meta = bundle.getString("meta");
            customers = (ArrayList<Customer>)bundle.getSerializable("customers");
        }

        View v = inflater.inflate(R.layout.list_frag, container, false);
        ListView list = v.findViewById(R.id.listView1);

        list.setAdapter(new SimpleAdapter(getActivity(), customers,
                R.layout.row, new String[]{ "name", "address" },
                new int[]{R.id.nameView, R.id.addrView}));

        return v;
    }
}
```

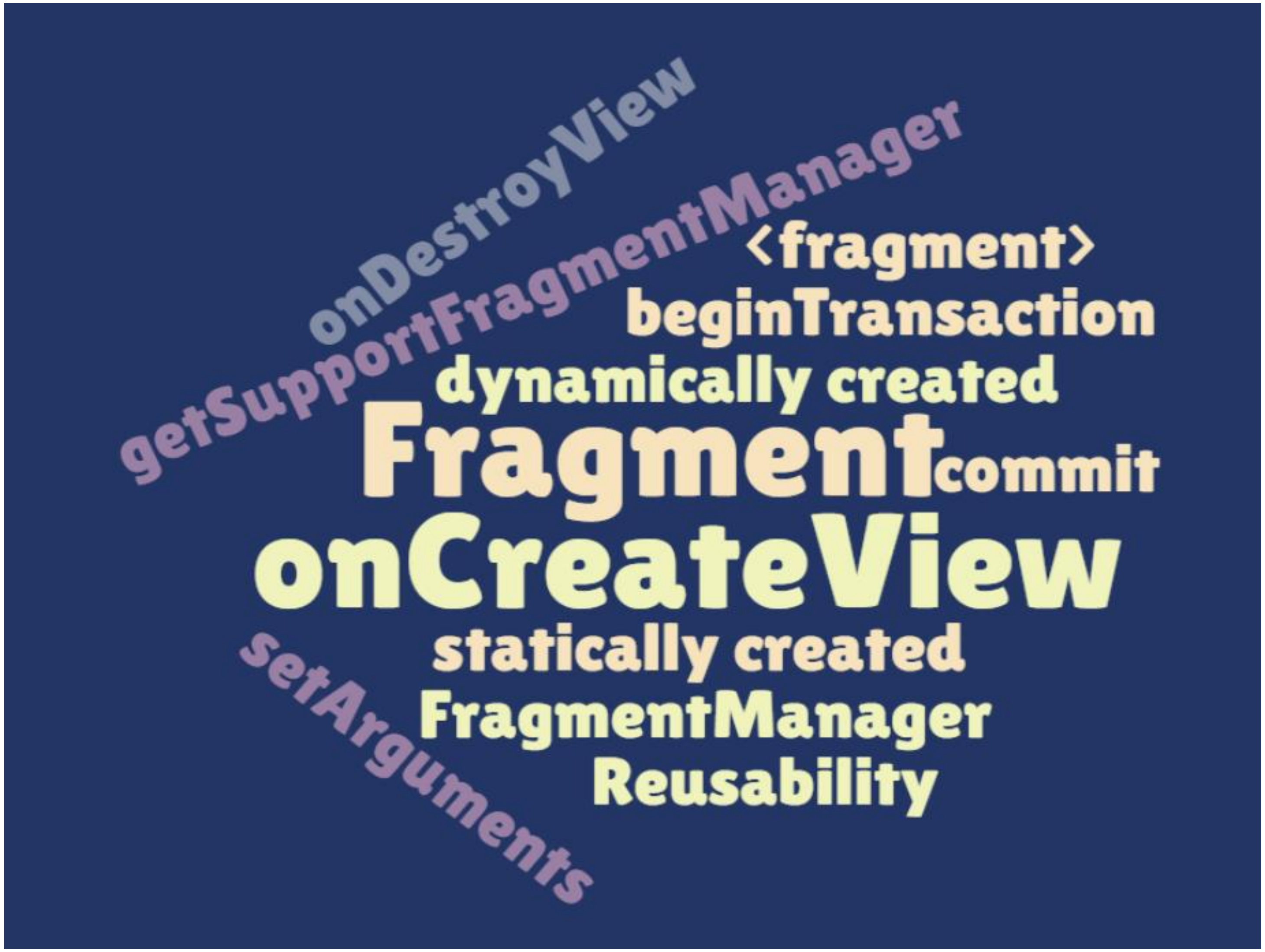
Practical Notes

- Import your Fragment library from
  - **android.support.v4.app.Fragment**
  - **NOT** ~~android.app.Fragment~~
- To access Fragment Manager, use
  - **getSupportFragmentManager**
  - **NOT** ~~getFragmentManager~~
- The **android.support** library allows you to target devices with older runtimes.
- You can use either **FragmentActivity** or **AppCompatActivity** to host your fragments

References

- Android’s developer notes on Fragments - https://developer.android.com/guide/components/fragments.html
- Creating and using Fragments - https://guides.codepath.com/android/creating-and-using-fragments

Active Recall


onDestroyView
getSupportFragmentManager
<fragment>
beginTransaction
dynamically created
Fragment
commit
onCreateView
statically created
FragmentManager
Reusability
setArguments