Inter-Fragment Communication Design Pattern

Fragment Fundamentals

Inter-Fragment Communication Design

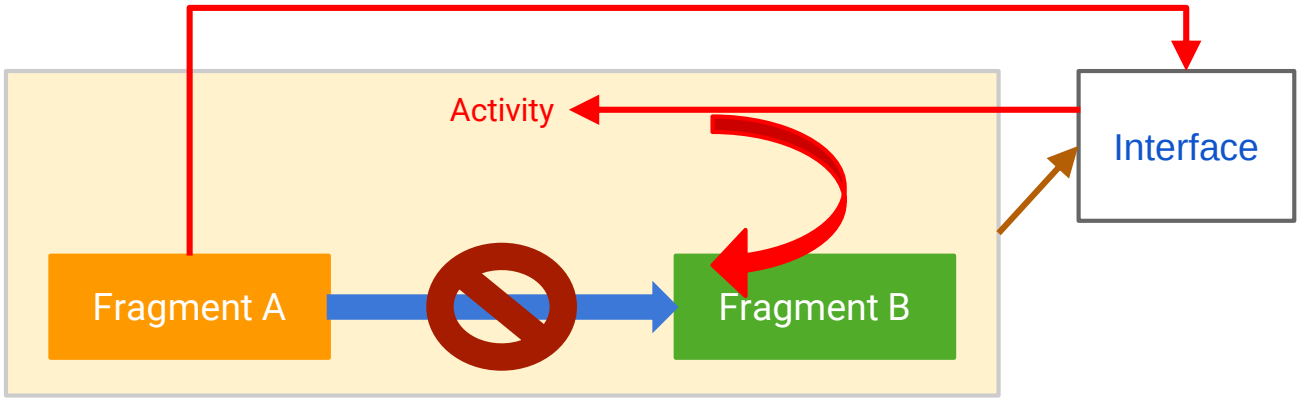
Don’t maintain direct references within each other.

Create an interface that contains the method which will act as an event carrier.

Let the Activity implement the interface.

Let Fragment A use the interface to send messages.

Use the callback method in Activity to trigger changes in Fragment B.



Fragment A

```
class FragmentA extends Fragment
{

    Communicator comm;

    public void onAttach(Activity a)
    {
        comm = (Communicator) a;
    }

    public void onClick(View v)
    {
       comm.respond("button was"
                          + " clicked");
    }




}
```

MainActivity

```
class MainActivity extends Activity
implements Communicator
{

      public void respond(String data)
      {

            FragmentB fB =
            manager.findFragmentById
            (R.id.fb);

            fB.changeData(data);


      }




}
```

Fragment B

```
class FragmentB extends Fragment
{
    public void changeData(String data)
    {


    }


}
```

```
interface Communicator
{
    void respond(String data);
}
```

Inter-Fragment Communication Design

NOTES:

Is there a simple way to do this?

YES.

- Use the onClick attribute in FragmentA for the button, define the method in the activity and call FragmentB from there. This is a BAD DESIGN however, because your FragmentB is supposed to take care of its own events rather than leaking them to the Activity increasing the level of COUPLING.
- The onClick attribute for a button looks for a method inside the Activity and NOT the Fragment, and you should DEFER from handling fragment events in the activity since it's BAD Design. Use the View.OnClickListener instead.
- For the ListFragment on the left side, use onItemClickListener to get the index of the item clicked and send it to the Activity using the same design pattern.

End of Slide