tuts+

Advertisement

CODE > ANDROID SDK

What Is the Android Activity Lifecycle?

by Chinedu Izuchukwu 13 Sep 2017

Difficulty: Beginner Length: Medium Languages: English

Android SDK | Mobile Development | Mobile App | Java

In my previous post, you learned that Intents let us send messages from one Android component to another. Well, one very important kind of component is an Activity.

Activities are a fundamental part of Android app development. And it's impossible to understand Activities without also understanding their lifecycles. In this post, you'll learn all about the Activity lifecycle.

ANDROID SDK

What Are Android Intents?

Chinedu Izuchukwu

Activity Lifecycle

An Activity is a single screen in Android. It is like a window in a desktop app, or a Frame in a Java program. An Activity allows you place all your UI components or widgets together on the screen.

It's important to understand that an Activity has a lifecycle: that is to say that it can be in one of several different states, depending on what is happening with the app and with the user interaction.

Lifecycle Methods

Let's look more closely at the lifecycle of an Android Activity. Each time the Activity state changes, one of the following lifecycle methods will be called on the Activity class.

`onCreate()`: This is called when the Activity is first initialized. You need to implement this method in order to do any initialization specific to your Activity.

`onStart()`: This is called the first time that the Activity is about to become visible to the user, as the Activity prepares to come to the foreground become interactive. Once this callback finishes, the `onResume()` method will be called.

`onResume()`: When the Activity goes into this state, it begins to interacts with the user. The Activity continues in this state till something happen to take focus from the app or Activity (such as an incoming call). When this happens, the `onPause()` method will be called.

`onPause()`: This method is used to pause operations that should not happen when the Activity is in paused state. A call to this method indicates that the user is leaving the app. For example, in a music player app, an incoming call will cause the app to transition into a paused state. This should mute or pause the currently playing music. When the user returns to the app, the `onResume()` method will be called.

`onStop()`: This method is called when the Activity is no longer visible in the app. It can happen, for example, when another Activity has been loaded and is taking the full screen of the device. When this method is called, the Activity is said to be in a stopped state. In this state, the system either calls the `onRestart()` to bring back interactivity with Activity. Or it calls the `onDestroy()` method to destroy the Activity.

`onDestroy()`: This gets called before the Activity is destroyed. The system calls this method when a user terminates the Activity, or because the system is temporarily destroying the process that contains the Activity to save space. Be sure to free up any resources your Activity has created in this method, or else your app will have a memory leak!

`onRestart()`: This gets called when an Activity restarts after it had been stopped.

Starting an Activity

Most user interactions with an app cause the active Activity to be changed. So an app transitions between Activities many times during its lifetime.

It's necessary to link Activities together when one Activity needs to start another Activity. To start an Activity, you either use `startActivity()` or `startActivityForResult()`. You have to pass an Intent in either case.

Starting an Activity With No Expected Result

`startActivity()` is used if the newly started Activity does not need to return a result.

The following code snippet shows how to start another Activity using this method:

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

You can also perform actions such as passing data from one Activity to another. In this case, your current Activity (the calling Activity) wants to pass data a target Activity. This is where Intents come in handy. To learn about using Intents to start an Activity, check out my previous article.

Starting an Activity With a Result

`startActivityForResult()` is used to start another Activity and expects to get data back from the newly started Activity. In other words, use this when you want to get a result from the target Activity back to the calling Activity, e.g. if the target Activity is collecting some user information in a modal dialog.

You receive the result from the Activity in the `onActivityResult(int requestCode, int resultCode, Intent data)` method. The result will be returned as an Intent.

Example of Starting an Activity

Here is an example to show how starting an Activity works.

First, you create your `MainActivity` with your `onCreate()` method, a layout file, and a request code.

```
public class MainActivity extends Activity {

    // Unique request code for each use case
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

In your `onCreate()` method, you'll create a new instance of an intent to start your second Activity.

When you're ready to start that Activity, say in response to a button click, you'll call `startActivityForResult()`, which will pass the newly created intent and the request code.

```
// Create a new instance of Intent to start SecondActivity
final Intent intent = new Intent(this, SecondActivity.class);

// This starts SecondActivity with the request code
startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
```

Still, in your `MainActivity`, you need to handle Activity result events. You do this by implementing the `onActivityResult()` method. This is how you will receive the result from the other Activity.

Here's how it should look:

```
// onActivityResult only get called
// when the other Activity previously started using startActivityForResult
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // First we need to check if the requestCode matches the one we used.
    if(requestCode == REQUEST_CODE_EXAMPLE) {

        // The resultCode is set by the SecondActivity
        // By convention RESULT_OK means that whatever
        // SecondActivity did was executed successfully
        if(resultCode == Activity.RESULT_OK) {
            // Get the result from the returned Intent
            final String result = data.getStringExtra(SecondActivity.EXTRA_DATA);

            // Use the data - in this case, display it in a Toast.
            Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
        } else {
            // setResult wasn't successfully executed by SecondActivity
            // Due to some error or flow of control. No data to retrieve.
        }
    }
}
```

Now go ahead and create your `SecondActivity`. It should look something like the code below.

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_detail);

      final Button button = (Button) findViewById(R.id.button);
      // When this button is clicked we want to return a result
      button.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              // Create a new Intent object as container for the result
              final Intent data = new Intent();

              // Add the required data to be returned to the MainActivity
              data.putExtra(EXTRA_DATA, "Some interesting data!");

              // Set the resultCode as Activity.RESULT_OK to
              // indicate a success and attach the Intent
              // which contains our result data
              setResult(Activity.RESULT_OK, data);

              // With finish() we close the SecondActivity to
              // return back to MainActivity
              finish();
          }
      });
  }
```

Terminating an Activity

Before an Activity terminates, the corresponding lifecycle methods will be called.

The `onPause()` method should stop all listeners and UI updates. The `onStop()` method should save the application data. Finally, the `onDestroy()` method will free up any resources that the Activity has allocated.

When the user switches back to an app that has been terminated by the system, the `onResume()` method is called. Based on saved data, it can re-register listeners and trigger UI updates.

Activity Instance State

An Activity needs a way to keep valuable state and user data that it has obtained. These data might be obtained from user input or created while the Activity was not on-screen.

For example, a change of device orientation can cause an Activity to be destroyed and recreated. In such a scenario, you need to make sure to save all Activity state before it is destroyed and reload it again when it is recreated. Otherwise, any data your Activity has at that time can be completely lost.

To save Activity state, you can override the `onSaveInstanceState()` method. This method is passed a `Bundle` object as a parameter. A bundle can contain strings, primitive data types, or objects. In this method, simply add any important state data to the bundle. This bundle will be returned to the Activity later so you can restore the Activity state.

To extract the saved state from the bundle and restore it, implement the `onRestoreInstanceState()` method. This callback is invoked between the `onStart()` and the `onResume()` lifecycle methods.

We will look deeper into Activity instance state in a future article.

Conclusion

After following this post, you'll have a good understanding of how an Activity lifecycle works. And you've learned that there are two ways to start an Activity, as well as getting some pointers to how instance state is handled in the Activity lifecycle.

Thanks for reading, and while you're here, check out some of our other posts on coding Android apps.

Chinedu Izuchukwu
Lagos, Nigeria

I enjoy writing code and passing knowledge. When not doing either of them, I watch movies.

Translate this post

Powered by native

0 Comments Tuts+ Hub Login

Recommend Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy

Advertisement

QUICK LINKS - Explore popular categories

ENVATO TUTS+ +

JOIN OUR COMMUNITY +

HELP +

tuts+

25,695	1,117	22,209
Tutorials	Courses	Translations

Envato.com Our products Careers Sitemap

Follow Envato Tuts+