Ciclo de vida de aplicaciones android

android

Octubre 2019

Klever Uyana Barahona

Indice

Tarea

Se trata de conocer:
LogCat
Ciclo de vida de aplicaciones android

Responder a la pregunta : ¿Donde guardarías el estado de una app para poder recuperarlo en caso que sea destruida (p.ej. por rotar el móvil y vuelta a crear?

Actividades.

¿Donde guardarías el estado de una app para poder recuperarlo en caso que sea destruida p.ej: por rotar el móvil y vuelta a crear?

Antes de realizar la actividad hay que saber que es el ciclo de vida de una APP de Android así como su importancia.

Este es un tema que no puede pasar desapercibido ya que es vital para entender como Android trata a nuestras actividades (Activities) y gestionar de manera correcta el ciclo de vida de estas. Un ciclo de vida, al igual que el ciclo vida de un animal: nace, crece, se reproduce y muere; también, es semejante en una actividad.

¿Porqué es importante esto?

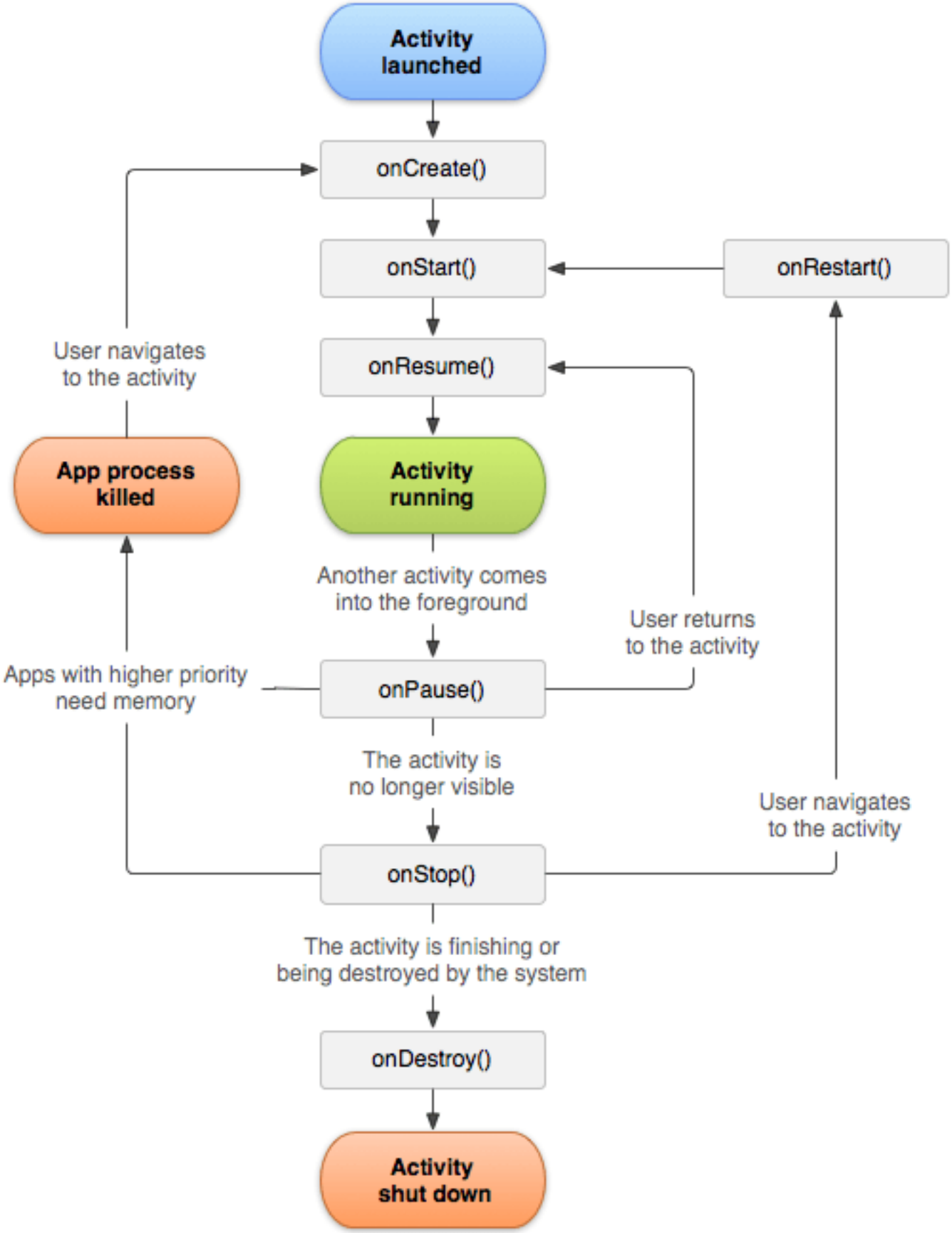
Dentro de los métodos del ciclo de vida, tu puedes declarar como se comporta tu actividad cuando el usuario abandona o vuelve a entrar en la actividad. Por ejemplo, si estás creando un reproductor de vídeo en streaming, deberías pausar el vídeo y finalizar la conexión a Internet cuando el usuario cambia a otra actividad. Cuando el usuario regrese, puedes volver a reconectar tu app a Internet y permitir al usuario iniciar el vídeo desde el punto donde lo dejó.
Este post explica la importancia de los métodos del ciclo de vida que cada instancia Activity recibe y como puedes utilizarlos para que tu actividad haga lo que el usuario espera de ella y no consuma recursos del sistema si no son necesarios.

El usuario y las actividades

En función de como el usuario navegue por las instancias de tu app, es decir entre las actividades, las cuales sufrirán transiciones entre diferentes estados en su ciclo de vida. Por ejemplo, cuando tu actividad se inicie por primera vez, se muestra en el primer plano del sistema y recibe la atención del usuario. Durante este proceso, el sistema Android llama a una serie de métodos del ciclo de vida sobre la actividad en la cual tu configuras la interfaz de usuario y otros componentes. Si el usuario realiza una acción que inicia otra actividad o cambia a otra app, el sistema llama a otro conjunto de métodos del ciclo de vida sobre tu actividad y se moverá al segundo plano (donde la actividad no se encuentra visible, pero su instancia y estado permanecen intactos).

Actividades.

El siguiente diagrama ilustra todos los métodos que existen en el ciclo de vida de una Actividad:



Actividades.

onCreate
Este método se llama al crear una Actividad. Una Actividad debe emplear este método para configurar su pantalla principal y realizar cualquier otra configuración inicial, como la creación de vistas, el enlace de datos a las listas, etc. Este método proporciona un parámetro Bundle, que es un diccionario para almacenar y transmitir información de estado y de los objetos entre las actividades.

onStart
Este método se llama cuando la Actividad está a punto de ser visible para el usuario. Las Actividades deben emplear este método si necesitan realizar cualquier tarea específica justo antes de que se hagan visibles, como por ejemplo refrescar valores actuales de los valores de las vistas dentro de la Actividad, o de puente para frame rates (una tarea común en la construcción de juegos).

onPause
Este método se llama cuando el sistema está a punto de poner la Actividad en background. Las Actividades deben emplear este método si necesitan confirmar los cambios no guardados a los datos persistentes, destruir o limpiar objetos para liberar los recursos que consumen, etc. Las implementaciones de este método deben finalizar lo antes posible, ya que ninguna Actividad sucesiva se reanudará hasta que este método devuelve un valor.
La Actividad puede ser destruida silenciosamente después de onPause(). Por lo tanto, podemos ignorar onStop() y onDestroy() y guardar los datos en onPause() (por ejemplo, puntuaciones más altas o nivel de progreso).

onResume
Este método se llama cuando la Actividad va a empezar a interactuar con el usuario después de estar en un estado de pausa. Cuando este método se llama, la Actividad se está moviendo a la parte superior de la pila y está recibiendo actividad del usuario. Las Actividades pueden emplear este método si lo necesitan para realizar cualquier tarea después de que la Activity haya aceptado la interacción del usuario.

onStop
Este método se llama cuando la Actividad ya no es visible para el usuario, ya que otra Actividad se ha reanudado o iniciado y está eclipsando a esta. Esto puede suceder porque la Actividad ha finalizado (el método de finalización fue llamado), porque el sistema está destruyendo a esta instancia de Actividad para ahorrar recursos, o porque un cambio de orientación se ha producido en el dispositivo. Se puede distinguir entre estos escenarios mediante el uso de la propiedad isFinishing. Una Actividad debe emplear este método si es necesario para realizar las tareas específicas antes de ser destruido, o si se trata de una interfaz de usuario para iniciar la reconstrucción después de un cambio de orientación.

onRestart
Este método se llama después de que una Actividad se haya detenido, antes de que se inicie de nuevo. Este método siempre es seguido por onStart. Una Activity debe pasar por onRestart si es necesario para realizar las tareas inmediatamente antes de llamar a onStart. Por ejemplo, si la actividad previamente ha sido enviada al background y onStop ha sido llamada, pero el proceso de la Actividad aún no ha sido destruido por el sistema operativo, entonces el método onRestart debe ser anulado.
Un buen ejemplo de esto sería cuando el usuario presiona el botón de inicio cuando aún había una Activity en la aplicación. El onPause, y a continuación el onStop, son llamados, pero la actividad no se destruye. Si el usuario a continuación restaurara la aplicación mediante el administrador de tareas o una aplicación similar, el método onRestart de la actividad sería llamado por el sistema operativo, durante la reactivación de actividades.

onDestroy
Este es el último método que se llama en una Actividad antes de que sea destruido. Después de llamar a este método, su Actividad se matará y será purgado de los grupos de recursos del dispositivo. El sistema operativo se destruirán permanentemente los datos de estado de una Actividad después de que este método se haya ejecutado, por lo que una Actividad debe reemplazar este método como medio final para salvar los datos del estado.

onSaveInstanceState
Este método es proporcionado por el ciclo de vida de Android para dar a una Actividad la oportunidad de guardar los datos cuando se produce un cambio, por ejemplo un cambio de orientación de pantalla.

Actividades.

Código completo

```
import androidx.annotation.NonNull;

public class MainActivity extends Activity {
    public int contador;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        android.util.Log.i(TAG, msg: "onCreate");
    if(savedInstanceState != null) {
        contador = savedInstanceState.getInt( key: "contador");
        mostrarResultado();
    }
    else
        {
        contador = 0;
        mostrarResultado();
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        android.util.Log.i(TAG, msg: "onStart");
    }
```

```
    @Override
    protected void onRestart() {
        super.onRestart();
        android.util.Log.i(TAG, msg: "onRestart");
    }

    @Override
    protected void onResume() {
        super.onResume();
        android.util.Log.i(TAG, msg: "onResume");
    }

    @Override
    protected void onPause() {
        super.onPause();
        android.util.Log.i(TAG, msg: "onPause");
    }

    @Override
    protected void onStop() {
        super.onStop();
        android.util.Log.i(TAG, msg: "onStop");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        android.util.Log.i(TAG, msg: "onDestroy");
    }
```

Actividades.

Código completo

```
    @Override
    protected void onSaveInstanceState(@NonNull Bundle outState) {
        outState.putInt("contador",contador);
        super.onSaveInstanceState(outState);
    }

    public void incrementaContador(View vista){
        contador++;
        mostrarResultado();
    }

    public void restaContador(View vista){
        contador--;
        mostrarResultado();
    }
    public void resetearContador(View vista){
        contador=0;
        mostrarResultado();
    }
    public void mostrarResultado(){
        TextView textoResultado=(TextView)findViewById(R.id.contadorPulsaciones);
        textoResultado.setText("Contador: "+contador);
    }
private static final String TAG = "CicloDeVida";
}
```

Actividades.

Conclusión

Android puede en cualquier momento pausar, parar, destruir nuestra aplicación según las necesidades del momento y nosotros debemos controlar todos estos eventos para hacer una aplicación robusta.

Respuesta a la pregunta :

¿Donde guardarías el estado de una app para poder recuperarlo en caso que sea destruida
p.ej. por rotar el móvil y vuelta a crear?

En mi caso a la hora de realizar la APP ,busque en numerosas fuentes tales como foros y material visual teles como https://www.youtube.com/watch?v=NMAJfqDOpBQ&t=498s
en donde explican la forma de guardar el estado , para su posterior uso.

Para mi aplicación tenia que guardar la información de una variable llamada contador , ya que si no lo haces la información se borra y el contador vuele a su esta original.

```
public class MainActivity extends Activity {
    public int contador;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    if(savedInstanceState != null) {
        contador = savedInstanceState.getInt( key: "contador");
        mostrarResultado();
    }
    else
        {
        contador = 0;
        mostrarResultado();
        }
    }

    @Override
    protected void onSaveInstanceState(@NonNull Bundle outState) {
        outState.putInt("contador",contador);
        super.onSaveInstanceState(outState);
    }
```

Para ello lo guarde con ayuda de onSaveIntanceState de esta manera .

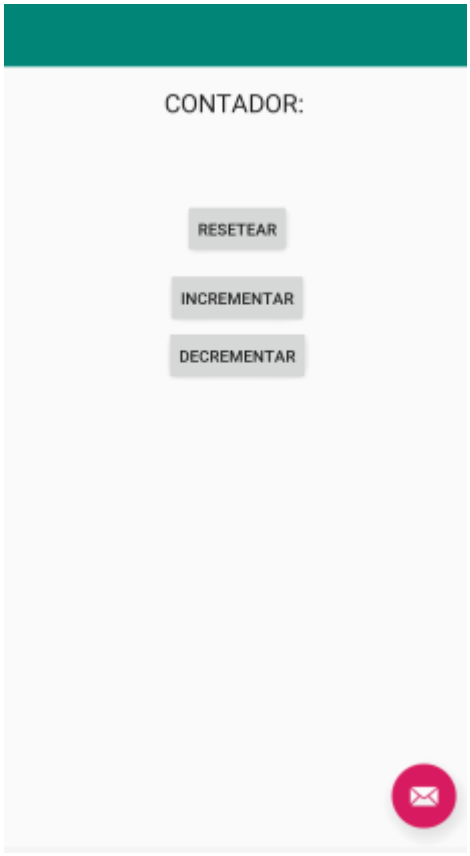
Y luego añadir en la clase MainActivity un condicional If , que comprueba si la Actividad a sido creada con anterioridad , si es así pasa por las lineas de su interior en donde recupera su valor anterior, y lo muestra, con la ayuda del metodo montrarResultado();

Esta es una forma de relalizarlo.

Actividades.

Problemas

En mi caso no he tenido complicación saldo a la hora de maquetar los botones y me aparecieran tanto de forma vertical como horizontal, ya que yo añadí dos botones mas , uno para restar , y otro para restaurar. Lo resolví y lo probé con mi móvil y todo perfecto.



Logcat vertical

```
28-428/? I/mple.proyecto_: Late-enabling -Xcheck:jr
28-428/? I/DecorView: createDecorCaptionView >> Dec
28-428/? I/CicloDeVida: onCreate
28-428/? I/CicloDeVida: onStart
28-428/? I/CicloDeVida: onResume
28-428/? W/mple.proyecto_: Accessing hidden method
28-460/? I/ConfigStore: android::hardware::configst
```

Logcat horizontal

```
proyecto_2 I/CicloDeVida: onPause
proyecto_2 I/CicloDeVida: onStop
proyecto_2 I/CicloDeVida: onDestroy
proyecto_2 W/libEGL: EGLNativeWindowType 0x7cleaf80
proyecto_2 I/DecorView: createDecorCaptionView >> De
proyecto_2 I/CicloDeVida: onCreate
proyecto_2 I/CicloDeVida: onStart
proyecto_2 I/CicloDeVida: onResume
proyecto_2 E/ViewRootImpl: sendUserActionEvent() ret
```