Ciclo de Vida de las Actividades

Qué alegría estar juntos de nuevo para seguir conociendo el desarrollo de aplicaciones en Android. En esta oportunidad nos centraremos en el **Ciclo de Vida de las Actividades**.

Una buena comprensión del ciclo de vida de la Actividad es vital para garantizar que nuestra aplicación proporciona una experiencia de usuario perfecta y administra adecuadamente sus recursos. Las aplicaciones de Android no controlan su propio ciclo de vida; por el contrario el **Android Runtime** gestiona el proceso de cada aplicación y, por extensión, el de cada actividad dentro de ella. Aunque el Runtime maneja la finalización y la administración del proceso de una Actividad, el estado de la Actividad ayuda a determinar la prioridad de nuestra aplicación. La prioridad de la aplicación, a su vez, permite al Runtime decidir cuáles aplicaciones terminar primero, así como las actividades que se ejecutan dentro de ella.

Conceptos del Ciclo de Vida de la Actividad

Para navegar por las transiciones entre las etapas del ciclo de vida de la actividad, la **clase Activity** proporciona un **conjunto básico de seis controladores de eventos (callbacks) : onCreate (), onStart (), onResume (), onPause (), onStop () y onDestroy ()**. El sistema invoca cada una de estos callbacks cuando una actividad entra en un nuevo estado.

El siguiente video presenta una representación visual de este paradigma.

“VER VIDEO EN EL ENLACE ADJUNTO A ESTA SECCION”

Cuando el usuario comienza a abandonar la actividad, el sistema llama a métodos para desmantelar la actividad. En algunos casos, este desmantelamiento es sólo parcial; la actividad aún reside en la memoria (como cuando el usuario cambia a otra aplicación) y aún puede volver a primer plano. Si el usuario vuelve a esa actividad, la actividad se reanuda desde donde la dejó. Con algunas excepciones, las aplicaciones tienen restringido el inicio de actividades cuando se ejecutan en segundo plano.

La probabilidad del sistema de matar un proceso determinado, junto con las actividades en él, depende del estado de la actividad en ese momento. El estado de actividad y la expulsión de la memoria proporcionan más información sobre la relación entre el estado y la vulnerabilidad a la expulsión.

Probablemente no necesitemos implementar todos los métodos del ciclo de vida en nuestra aplicación. Sin embargo, es importante comprender cada uno e

implementar aquellos que garanticen que nuestra aplicación se comporta como los usuarios esperan.

Pilas de Actividades y la Lista de las Menos Utilizadas Recientemente (LRU, Least Recently Used)

El **estado de cada actividad está determinado por su posición en la pila de actividades (o "back stack")**, una colección tipo último en entrar, primero en salir (LIFO, Last In First Out) de todas las actividades que se ejecutan actualmente. Cuando se inicia una nueva Actividad, se activa y se mueve a la parte superior de la pila. Si el usuario navega hacia atrás con el botón Atrás, o si la Actividad en primer plano está cerrada, la siguiente Actividad hacia abajo en la pila se mueve hacia arriba y se activa. La siguiente figura ilustra este proceso.

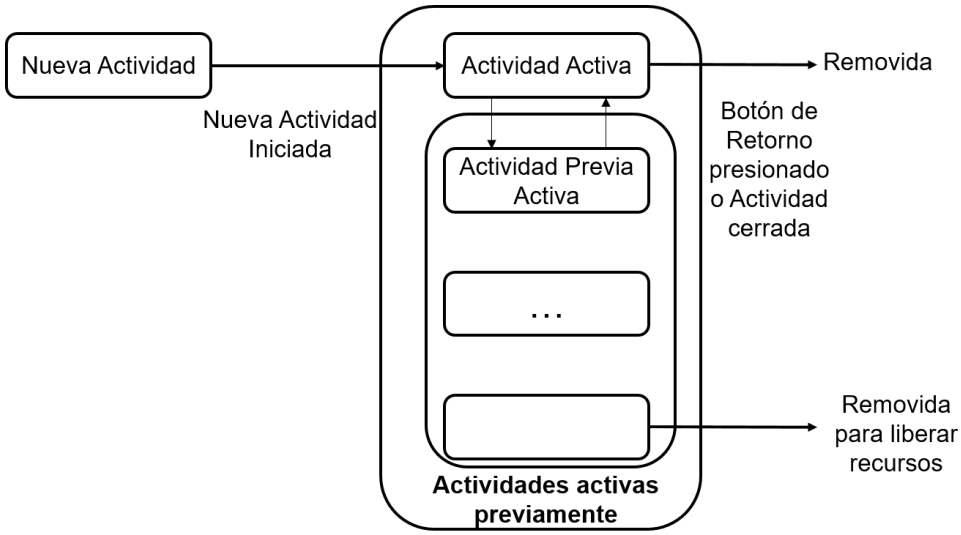

Figura 1: Activity Back Stack

La prioridad de una aplicación está influenciada por su Actividad de mayor prioridad. Cuando el administrador de memoria de Android decide qué aplicación finalizar para liberar recursos, utiliza esta pila de Actividad para determinar la prioridad de las aplicaciones. Cuando ninguna de las Actividades de una aplicación está visible, la aplicación en sí se mueve a la lista de uso menos reciente (LRU), que se utiliza para determinar el orden en que las aplicaciones finalizarán para liberar recursos como se describió anteriormente.

Estados de la Actividad

A lo largo del ciclo de vida de una aplicación, sus Actividades entran y salen de la pila de Actividades, como se muestra en la figura 1. A medida que lo hacen, pasan a través de cuatro estados posibles:

- **Activo:** cuando una actividad está en la parte superior de la pila, es la actividad visible, enfocada y en primer plano, que recibe la entrada del usuario. Android intentará mantenerla viva a toda costa, eliminando las aplicaciones que poseen Actividades más abajo en la pila según sea necesario, para garantizar que esta Actividad tenga los recursos que necesita. Cuando se active otra Actividad, esta se pondrá en pausa, y cuando ya no sea visible, se detendrá, como se describe en los siguientes puntos.
- **En pausa:** en algunos casos, nuestra actividad será visible pero no tendrá foco; en este punto está en pausa. También se puede alcanzar este estado en el que nuestra aplicación se está utilizando en un entorno de múltiples ventanas, donde pueden verse múltiples aplicaciones, pero sólo la Actividad con la que el usuario interactuó por última vez se considera activa. Del mismo modo, si nuestra Actividad es una Actividad transparente o que no tiene la interfaz completa activa y en primer plano, permanecerá en estado de pausa. Cuando está en pausa, una actividad se trata como si estuviera activa; sin embargo, no recibe eventos de entrada del usuario. En casos extremos, Android eliminará una actividad pausada para recuperar recursos para la actividad activa. Cuando una actividad se oscurece por completo, se detiene; todas las actividades pasan por el estado de pausa antes de detenerse.
- **Detenido:** cuando una actividad no está visible, se "detiene". La actividad permanecerá en la memoria y retendrá toda la información del estado; sin embargo, ahora es un candidato probable para la terminación, en el caso de que el sistema requiera memoria. Cuando una actividad está en estado detenido, es importante guardar los datos y el estado actual de la interfaz de usuario, además de detener cualquier operación no crítica. Una vez que una Actividad ha salido o cerrado, se vuelve inactiva.
- **Inactivo:** después de que una Actividad ha sido eliminada, así como antes de que se inicie (launched), está inactiva. Las actividades inactivas se han eliminado de la pila de actividades y deben reiniciarse antes de que puedan mostrarse y utilizarse.

Las transiciones de estado ocurren a través de acciones del usuario y del sistema, lo que significa que nuestra aplicación no tiene control sobre cuándo suceden. Del mismo modo, la finalización de la aplicación es manejada por el administrador de memoria de Android, que comenzará cerrando las aplicaciones que contienen actividades inactivas, seguidas por las que se detienen. En casos extremos, eliminará las que están en pausa.

Para garantizar una experiencia de usuario perfecta, las transiciones entre estados deben ser invisibles para el usuario. No debería haber diferencia en una Actividad que pasa de un estado en pausa, detenido o inactivo a activo, por lo que **es importante guardar todo el estado de la interfaz de usuario (IU)** y conservar todos los datos cuando se detiene una Actividad. Es una buena práctica realizar operaciones de persistencia de estado que consuman mucho tiempo (como transacciones de bases de datos o transferencias de red) cuando la Actividad pasa al estado detenido (dentro del controlador **onStop**), en lugar de su transición al estado de pausa (dentro de **onPause**).

Las actividades pueden hacer una transición entre los estados activos y en pausa con frecuencia y rapidez (particularmente cuando se usan en un entorno de múltiples ventanas), por lo que es importante que esta transición se ejecute lo más rápido posible. Una vez que una Actividad se vuelve activa, debería restaurar los valores guardados. Del mismo modo, aparte de los cambios en la prioridad de la Actividad, las transiciones entre los estados activo, en pausa y detenido tienen poco impacto directo en la Actividad misma. Depende de nosotros usar estas señales para pausar y detener nuestras Actividades en consecuencia, y estar preparados para la terminación en cualquier momento.

Comprender los Tiempos de Vida de la Actividad

Para garantizar que las actividades puedan reaccionar a los cambios de estado, **Android proporciona una serie de controladores de eventos que se activan como una transición de actividad a través de sus vidas completas, visibles y activas**. La figura 1 resume estas vidas en términos de los estados de actividad descritos en la sección anterior. Durante la vida útil completa de una Actividad, entre la creación y la destrucción, pasa por una o más iteraciones de las vidas activas y visibles. Cada transición activa los manejadores de métodos descritos anteriormente. Las siguientes secciones proporcionan una mirada más cercana a cada una de estas vidas y los eventos que los agrupan.

Entendiendo el Ciclo de Vida de las Actividades

Para garantizar que las actividades puedan reaccionar a los cambios de estado, **Android proporciona una serie de controladores de eventos (callbacks)** que se activan cuando una actividad transiciona a través de las diferentes etapas del ciclo de vida: tiempo de vida completo, visible y activo (full, visible and active Lifetimes).

Tiempo de Vida Completo

El Tiempo de Vida Completo de nuestra Actividad se produce entre la primera llamada a onCreate y cuando se destruye. No es raro que el proceso de una Actividad finalice, sin que se llame al controlador onDestroy correspondiente. Usamos el método onCreate para inicializar su Actividad: inflar la interfaz de usuario, obtener referencias a Fragmentos, asignar referencias a variables de clase, vincular datos a controles e iniciar Servicios. Si la actividad se finalizó inesperadamente por el Runtime, el **método onCreate** pasa un objeto **Bundle** que contiene el estado guardado en la última llamada a **onSaveInstanceState**. Debemos usar este paquete para restaurar la interfaz de usuario a su estado anterior, ya sea dentro del método **onCreate u onRestoreInstanceState**.

Sobreescribimos (override) onDestroy para limpiar los recursos creados en **onCreate** y asegurarnos de que todas las conexiones externas, como enlaces de red o de base de datos, estén cerradas. Como parte de las pautas

de Android para escribir código eficiente, se recomienda evitar la creación repetida de objetos a corto plazo. La rápida creación y destrucción de objetos obliga a la recolección de basura adicional, un proceso que puede tener un impacto negativo directo en la experiencia del usuario. Si nuestra Actividad crea el mismo conjunto de objetos regularmente, debemos considerar crearlos en el método onCreate, ya que se llama sólo una vez durante la vida útil de la Actividad.

Tiempo de Vida Visible

El Tiempo de Vida Visible de una Actividad está limitado entre las llamadas a onStart y onStop. Entre estas llamadas, nuestra actividad será visible para el usuario, aunque puede no tener el foco y puede estar parcialmente oculta. Es probable que las actividades atraviesen varias veces el tiempo de vida visible durante toda su vida, a medida que se mueven entre el primer plano y el fondo.

Desde Android 3.0 Honeycomb (API Nivel 11), podemos asumir con seguridad que se llamará a onStop antes de que finalice el proceso de la aplicación . El método **onStop** debe usarse para pausar o detener animaciones, subprocesos, escuchas de sensores, búsquedas de GPS, temporizadores, servicios u otros procesos que se usan exclusivamente para actualizar la interfaz de usuario. Hay poco valor en consumir recursos (como memoria, ciclos de CPU o ancho de banda de red) para actualizar la IU, cuando no es visible. Use el método **onStart** para reanudar o reiniciar estos procesos, cuando la IU vuelva a estar visible. El método onRestart se llama inmediatamente antes de todos menos la primera llamada a onStart. Lo podemos usar para implementar el procesamiento especial que desea que se realice sólo cuando la Actividad se reinicie dentro de su tiempo de vida útil completo. Los métodos **onStart / onStop** también deben usarse para registrar y anular el registro de los receptores de difusión utilizados exclusivamente para actualizar la IU.

Tiempo de Vida Activo

El Tiempo de Vida Activo comienza con una llamada a onResume y termina con una llamada correspondiente a onPause. Una actividad activa está en primer plano y recibe eventos de entrada del usuario. Es probable que nuestra Actividad pase por muchos tiempos de vida activo antes de ser destruida, ya que este finalizará cuando se muestra una nueva Actividad, el dispositivo se suspende o la Actividad pierde el foco. Intentemos mantener el código en los métodos **onPause** y **onResume** para que se ejecuten de forma rápida y liviana para garantizar que nuestra aplicación siga respondiendo cuando se mueva dentro y fuera del primer plano. Podemos suponer con seguridad que durante la vida activa se llamará a **onPause** antes de que finalice el proceso. Si el sistema determina que es posible que se deba reanudar este estado de Actividad, inmediatamente antes de onPause se realiza una llamada a **onSaveInstanceState**. Este método brinda la oportunidad de guardar el estado de la IU de la actividad en un paquete que se

puede pasar a los métodos **onCreate** y **onRestoreInstanceState**.
Usemos **onSaveInstanceState** para guardar el estado de la interfaz de usuario y así asegurarnos de que la actividad puede presentar la misma interfaz de usuario la próxima vez que se active. No se llamará al controlador **onSaveInstanceState**, si el sistema determina que el estado actual no se reanudará, por ejemplo, si la Actividad se cierra presionando el botón Atrás. Desde Android 3.0 Honeycomb (API Nivel 11), la finalización del controlador onStop marca el punto más allá del cual una Actividad puede ser eliminada sin previo aviso. Esto le permite mover todas las operaciones que requieren mucho tiempo para guardar el estado en **onStop**, manteniendo nuestro onPause liviano y enfocado en suspender las operaciones intensivas de memoria o CPU mientras la Actividad no está activa. Dependiendo de la arquitectura de nuestra aplicación, esto puede incluir la suspensión de subprocesos, procesos o receptores de difusión mientras su actividad no esté en primer plano. El método **onResume** correspondiente también debe ser ligero. No es necesario volver a cargar el estado de la interfaz de usuario aquí, porque esto debería ser manejado por los métodos **onCreate** y **onRestoreInstanceState** según sea necesario.
Use **onResume** para revertir las acciones realizadas dentro de **onPause**, como asignar recursos liberados, inicializar o registrar componentes eliminados o no registrados, y reanudar cualquier comportamiento suspendido.

Esperamos que hayas disfrutado este encuentro de aprendizaje con el ciclo de vida de las Actividades.