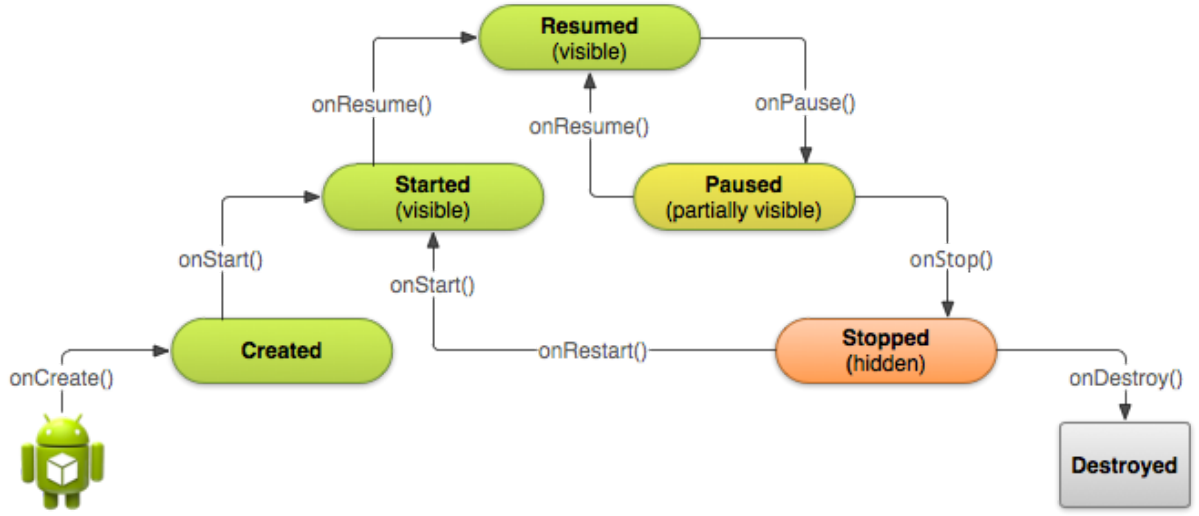
Ciclo de Vida de uma Activity

Conceito

Uma Activity representa uma tela com interface gráfica capaz de promover algum tipo de interação com o usuário. Uma aplicação Android pode ser composta de diversas activities para fornecer um conjunto de funcionalidades para o usuário.

Compreender o funcionamento e as transições é importante para o desenvolvimento e para a escolha da sua lógica. Por exemplo, se estiver trabalhando com um aplicativo de vídeo, você pode querer pausar o vídeo se o dispositivo receber uma ligação ou se outro aplicativo for aberto simultaneamente. Dentro desse ciclo de vida, é possível declarar como a *activity* se comporta quando o usuário deixa ou retorna a *activity*.

A *Activity* possui um ciclo de vida específico. Quando um usuário abre o aplicativo, troca de tela, deixa o aplicativo em segundo plano, a *activity* passa por uma série de estados descritos abaixo.



Callbacks no Ciclo de Vida

Sempre que uma *activity* muda de estado, o Android faz a chamada de um método (callback) correspondente.

onCreate()

Quando seu aplicativo é aberto, o Android sabe qual a página inicial que deve ser criada, uma vez que está definida no *AndroidManifest.xml.* Com isso, o método onCreate() dessa página é invocado.

É nesse momento que ocorre a definição de qual layout sua página utilizará e você também pode inicializar as variáveis que serão utilizadas, popular uma lista, definir uma thread, etc.

É importante destacar que esse método deve ser declarado obrigatoriamente. Caso não seja declarado o Android Studio acusará um erro e não será possível rodar a aplicação.

onStart()

Esse método faz com que a activity seja visível para o usuário, enquanto o aplicativo prepara para a atividade entrar em funcionamento e se tornar interativa.

O método onStart() é completado rapidamente. Assim que esse callback é terminado, a activity entra no modo R*esumido*.

onResume()

Quando a *activity* entrar no estado *Resumido*, ela entra em funcionamento e está pronta para interação com o usuário. Assim que o aplicativo entra nesse estado, permanecerá assim até que algo aconteça e tire o foco do aplicativo. Por exemplo, uma chamada telefônica, um alarme ou a tela do aplicativo sendo bloqueada ou desligada.

Se a activity retorna ao estado Resumido, o sistema chama onResume() novamente e não onStart(), uma vez que ela já foi iniciada e já está em memória. Por esse motivo, você deve implementar onResume() para reinicializar componentes que descartados no onPause().

Esse método é chamada todas as vezes que uma activity se torna visível para o usuário, inclusive a primeira vez após a chamada do método onCreate().

onPause()

Quando qualquer um dos eventos descritos no método onResume ocorre, o Android faz a chamada do callback onPause(). Esse método é chamado como o primeiro indicativo de que o usuário está saindo da *activity*.

Esse método deve ser usado para pausar animações e músicas que não devem continuar quando sua tela está pausada.

Por exemplo, se seu aplicativo utiliza a câmera, o método onPause() é um ótimo local para desalocar o recurso da câmera para ser alocada posteriormente no método onResume().

A execução desse método é muito breve e não necessariamente dispõe de muito tempo para performar ações de salvar dados. Por esse motivo, esse método não deve ser usado para salvar dados da aplicação, chamadas a APIs ou banco de dados. Tais ações devem ser feitas no método onStop() descrito abaixo.

onStop()

Quando sua activity não está mais visível para o usuário, porém ainda permanece alocada na memória, ela entra no modo *Parado* e o sistema invoca o callback onStop().

É nesse método que todos os recursos alocados na criação devem ser descartados. Você deve usar o método quando desejar realizar alguma operação que use relativamente a CPU, como por exemplo, salvar dados no banco de dados ou uma chamada a uma API.

A partir do estado *Parado*, a atividade volta a interagir com o usuário ou a atividade é finalizada e desaparece. Se a atividade voltar, o sistema invoca onRestart(). Se a atividade terminar de ser executada, o sistema chamará onDestroy().

onDestroy()

Método chamado antes da atividade ser destruída. Esta é a chamada final que a activity recebe. Este método libera todos os recursos alocados que não tenham sido descartados previamente durante os outros métodos.