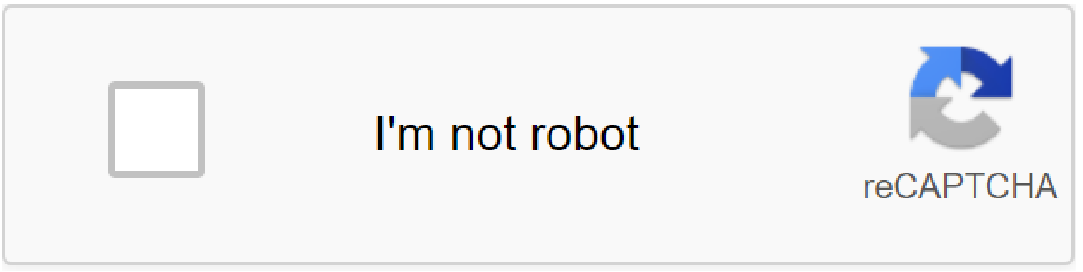
Android finishaffinity vs finish



Continue

Public class activities are expanded by ContextThemeWrapper implements LayoutInflater.Factory2, Window.Callback, KeyEvent.Callback, View.OnCreateContextMenuListener, ComponentCallbacks2 Activity is one, purposeful thing that the user can do. Almost all activities interact with the user, so the Activity class cares about creating a window for you where you can place your user interface with setContentView (View). While actions are often presented to the user as full-screen windows, they can also be used in other ways: like floating windows (through a theme with the R.attr.windowIsFloating set), Multi-Window mode, or built-in other windows. There are two methods that will be implemented in almost all Activity subclasses: onCreate (Bundle) is the place where you initiate your activities. Most importantly, here you're usually referred to as setContentView (int) with a layout of the resource that defines your user interface, and using findViewById (int) to get widgets in this user interface that you need to interact with the software. onPause is the place where you are dealing with a user who suspends active activity. Any changes made by the user must be made at this point (usually in ContentProvider holding the data). In this state, the action is still visible on the screen. To be used with Context.startActivity, all activity classes must have an appropriate declaration in AndroidManifest.xml of their package. Topics covered here: Activity class is an important part of the overall lifecycle of an application, and the way you run and act so much is a fundamental part of the platform application model. For more information about the structure of the Android app and how the actions are behaved, see the Application Fundamentals and Tasks and Back Stack guide. You can also find a detailed discussion on how to create actions in the Action Developer Guide. Fragments of the FragmentActivity subclass can use the Fragment class to better modalize the code, create more complex user interfaces for larger screens, and scale their applications between small and large screens. For more information about using fragments, read the Fragments developer's guide. Lifecycle activities in the system are managed as activity stacks. When a new action starts, it usually fits at the top of the current stack and becomes the current action - the previous action always stays below it in the stack, and will not come to the fore again until the new action comes out. One or more stacks of activity can be seen on the screen. The action essentially has four states: if the action is in the foreground (at the highest position of the top stack), it is active or working. This is usually an activity that the user is currently interacting with. If the action has lost focus, but is still presented to the user, it is not yet known. Visible. This is possible if the new non-full-size or transparent activity focuses on the top of your activity, the other action has a higher position in the multi-window mode, or the action itself is not focused in the current window mode. Such activity is fully alive (it retains all information about the condition and member and remains attached to the window managers). If the action is completely hidden by another action, it is stopped or hidden. It still saves all information about the state and member, however, it is no longer visible to the user, so its window is hidden, and it will often be killed by the system when memory is needed elsewhere. The system can eliminate activity from memory, either by asking it to finish, or simply by killing its process, making it destroyed. When it is displayed again to the user, it must be fully restarted and restored to its former state. The following chart shows important ways of state of action. Square rectangles represent callback methods that can be implemented to perform operations when the action moves between states. Colored ovals are the main states of activity can be in. there are three key cycles you may be interested in monitoring in your activities: the entire duration of the action takes place between the first onCreate call (Bundle) to one final call onDestroy (). The activity will do all the global state settings in onCreate and free up all remaining resources in onDestroy. For example, if a thread is running in the background to download data from the network, it can create that thread into onCreate, and then stop the flow in onDestroy. The apparent expiration date occurs between the onStart call before the appropriate onStop call. During this time, the user can see the action on the screen, although it may not be in the foreground and interact with the user. Between these two methods, you can support the resources needed to show activity to the user. For example, you can register BroadcastReceiver in onStart to monitor changes that affect your user interface, and unregistered it in onStop when the user no longer sees what you're displaying. OnStart and onStop methods can be called several times, as the action becomes visible and hidden to the user. The foreground life of the action takes place between the onResume call and the appropriate onPause call. During this time, the action is visible, active and interacting with the user. The action can often go between renewed and suspended states - for example, when the device goes to sleep, when the result of the action is delivered, when a new intent is delivered - so the code in these methods should be fairly light. the lifecycle of the action is determined by the following activity methods. These are all hooks that can be overridden to drink the appropriate work when changes the state. All actions will be done onCreate (Bundle) for initial setting; many of them will also implement onPause to make changes to the data and prepare for a pause in user experience, and onStop () to process is no longer visible on the screen. You should always call up your super class when implementing these methods. Public class activities expand ApplicationContext - protected void onCreate (saved BundleInstanceState); Protected void onStart protected void onRestart (); protected void onResume protected void onPause Protected void onStop Protected void onDestroy In general, the movement on the life cycle of action looks like this: Method Description Killable? Further a day, The Site is called when the action is first created. Here you have to do all your usual static settings: create views, link data to lists, etc. Always followed onStart. No onStart () onRestart () Called after your activity has been stopped, before it has been launched again. Always followed onStart () No onStart () is called when the activity becomes visible to the user. Then on Resume if the action comes to the fore, or onStop if it becomes hidden. No onResume () or onStop () onResume () Called when the action starts to interact with the user. At the moment your activity is at the top of the activity stack, with custom input going towards it. Always followed by onPause. No onPause () onPause () Called when the action loses the foreground state, is no longer focused or before moving into a stopped/hidden or destroyed state. Activity is still visible to the user, so it's a good idea to keep it visually active and continue updating the user interface. Implementation of this method should be very quick, as the next action will not be resumed until this method returns. This is followed by either onResume if the action returns back to the front, or onStop if it becomes invisible to the user. It's Build.VERSION_CODES. HONEYCOMB onResume () or onStop () onStop () Is called when the action is no longer visible to the user. This can happen either because new activity is being launched on top, existing is being brought to this, or this one is being destroyed. This is commonly used to stop animation and update the user interface, etc. then it should either onRestart () if this action returns to interact with the user, or onDestroy () if this activity goes away. Yes onRestart () onDestroy () The final call you receive before your activity is destroyed. This can happen either because the action is terminated (someone called Activity'finish on it) or because the system temporarily destroys that instance of action to save space. You can distinguish between the two Activity'isFinishing. Yes, don't pay attention to the Killable column in the table above - for those methods that are labeled as lethal, after this method returns the hosting process, activities can be killed by the system at any time without running another line of code. Because of this, you should use onPause to record any permanent data (such as user edits) for storage. In addition, onSaveInstanceState (android.os.Bundle) is called before placing the action in such a background state, which allows you to keep any dynamic state of the instance in your activity in this kit, which will later be received in onCreate (Bundle) if the action is to be created. For more information on how the process's lifecycle is tied to the activities it conducts, visit the Life Cycle Process section. Note that it is important to keep the data permanent in onPause () instead of onSaveInstanceState (Bundle) because the latter is not part of the lifecycle callbacks, so it will not be named in any situation as described in its documentation. Keep in mind that this semantics will change a little between application targeting platforms, starting with Build.VERSION_CODES. HONEYCOMB vs. those that are focused on previous platforms. Starting with Honeycomb, the app is not in a killer state until it is onStop.com. This affects when onSaveInstanceState (android.os.Bundle) can be called (this can be safely called after onPause)) and allows the app to safely wait until onStop () to maintain a permanent state. For app targeting platforms, starting with Build.VERSION_CODES. P onSaveInstanceState (android.os.Bundle) will always be called after onStop, so the app can safely execute fragment transactions in onStop and will be able to maintain a permanent state later. For those methods that are not labeled as lethal, the process of action will not be killed by the system, starting from the time when the method is called and continues after its return. Thus the activity is in a killable state, for example, between after onStop () to the start onResume(). Keep in mind that under extreme memory pressure, the system can kill the application process at any time. Configuration changes If the configuration of the device (defined by the configuration class) changes, then everything that displays the user interface needs to be updated to fit that configuration. Because activity is the primary mechanism of user interaction, it includes special support to handle configuration changes. If you don't specify otherwise, a change in configuration (such as changing the orientation of the screen, language, input devices, etc.) will result in current activity destroyed, if necessary, during the normal lifecycle of onPause (), onStop () and onDestroy(). If the action was in the foreground or visible user, once onDestroy () is called in this case, then a new instance of activity will be created, with any savedfaved previous instance was created from onSaveInstanceState (Bundle). This is because any app resource, including layout files, can change depending on the configuration value. Thus, the only safe way to handle a configuration change is to re-extract all resources, including layouts, drawings, and strings. Because actions need to know how to save their state and re-create yourself from this state, this is a convenient way to restart the action with a new configuration. In some special cases, you can bypass the restart of the action based on one or more types of configuration changes. This is done with android:configChanges attribute in his manifesto. For any type of configuration changes that you say you're working there, you'll get a call to onConfigurationChanged (Configuration) of your current action instead of rebooting. However, if the configuration change is due to the fact that you are not processing, the action will still be restarted and onConfigurationChanged (Configuration) will not be called. StartActivity is used to start a new action that will be placed at the top of the activity stack. One argument is required, an intention that describes the action that needs to be performed. Sometimes you want to get the result back from the activity when it ends. For example, you can start an action that allows the user to select the person on the contact list; when it ends, it returns the person who was chosen. To do this, you call startActivityForResult (Intention, Int) version with the second integrator option, the call identification. The result will return through your onActivityResult (int, int, Intent) method. When the action comes out, it can trigger setResult (int) to return the data back to its parent. It should always deliver a result code that can be standard results RESULT_CANCELED, RESULT_OK or any user values, starting with RESULT_FIRST_USER. In addition, he may additionally return the intention containing any additional data he or she wants. All this information appears back on Activity.onActivityResult parents, along with the integrator ID he originally provided. If a child fails for any reason (such as a failure), the parent's actions will result in a RESULT_CANCELED code. MyActivity community class expands activity ... static final int PICK_CONTACT_REQUEST No 0; public boolean onKeyDown (int keyCode, KeyEvent event) - if (keyCode - KeyEvent.KEYCODE_DPAD_CENTER) / When the user center clicks, let them choose the contact. (new intention (Intent.ACTION_PICK, new Uri (content)), PICK_CONTACT_REQUEST); The return is true. That's right. requestCode, int resultCode, Intent Data if (requestCode - PICK_CONTACT_REQUEST) - if (resultCode - RESULT_OK) - Contact was chosen. Here we just display it / for the user. startActivity (new intention (Intent.ACTION_VIEW, data)) Maintaining a permanent state typically exists two types of permanent status that will deal with the action: general document-like data (usually stored in the S'Lite database using a content provider) and an internal state, such as user preferences. For the content provider data, we suggest using a custom model to edit on-site. That is, any edits that the user makes are actually made immediately, without requiring an additional confirmation step. Support for this model usually consists of two rules: when a new document is created, you immediately enter a database or a backup database file. For example, if a user decides to write a new email, a new entry for that email is created as soon as they start entering the data, so if they go to any other action after that point that email will now appear in the draft list. When the onPause method is called, it must vouch for the backup content provider or submit any changes made by the user. This ensures that these changes will be visible to any other activity that is about to start. You're probably want to commit your data even more aggressively at key moments during the lifecycle of your activity: for example, before the start of a new activity, before the end of your own activity, when the user switches between input fields, etc. This model is designed to prevent data loss when the user moves between activities, and allows the system to safely kill actions (because system resources are needed somewhere else) at any time after it has been stopped (or suspended on the platform until Build.VERSION_CODES. Please note that this means that the user who is pressed back from your activity does not mean cancellation - it means that the action with the current content saved. The cancellation of edits in action should be made through another mechanism, such as a clear return or cancellation option. For more information about content providers, see the content package. This is a key aspect of how different actions cause and disseminate data among themselves. The Activity class also provides an API to manage the internal standing state associated with the action. This can be used, for example, to memorize the user's preferred original display in the calendar (day or week view) or homepage default in a web browser. The steady state of activity is controlled by getPreferences (int), which allows you to obtain and change a set of pairs of names/values associated with the action. You can use the preferences shared between multiple components of the application (activity, recipients, services, vendors) to use Context-getSharedPreferences to extract a preference object stored under a specific name. (Please note that you can't share customization data in app packages, so you'll need a content provider.) Here's an excerpt from the calendar action that keeps the user's preferred view mode in its persistent settings: calendarActivity's public class expands the activity... static final int DAY_VIEW_MODE No 0; static final int WEEK_VIEW_MODE No 1; Private CommonRections mPrefs; Private int mCurViewMode; protected void onCreate (Bundle savedInstanceState) - super.onCreate SharedPreferences mPrefs - getSharedPreferences (); mCurViewMode - mPrefs.getInt (view_mode, DAY_VIEW_MODE); - protected void on The Pause () - super.onPause(); SharedPreferences.Editor ed - mPrefs.edit(); ed.putInt (view_mode, mCurViewMode); ed.commit(); Permissions Can start a certain action can be applied when it is advertised in the tag of the manifesto. In doing so, other applications will have to declare the appropriate element in their manifest to be able to begin this activity. When you start an Action, you can establish the intention FLAG_GRANT_READ_URI_PERMISSION and/or intent FLAG_GRANT_WRITE_URI_PERMISSION intention. This will provide Activity access to specific URI's intention. Access will remain until the action is complete (it will remain in the process of hosting the killed and other temporary destruction). In Build.VERSION_CODES. GINGERBREAD, if the Action has already been created and the new intention is delivered to OnNewIntent (android.content.Intent), any newly granted URI permissions will be added to the existing ones it has. For more information on permits and security in general, please visit Security and Permits. The Lifecycle Android Process system tries to keep the application process around as much as possible, but will eventually need to remove old processes when the memory runs out. As described in Lifecycle Activity, the decision on which process to remove is closely related to the state of the user's interaction with it. In general, there are four states in which the process can be based on the actions working in it listed here in order of importance. The system will kill less important processes (the latter) before resorting to killing more important processes (first). The most important is the foreground action (activity at the top of the screen with which the user is currently interacting). Its process will only be killed as a last resort if it uses more memory than is available on the device. Typically, at this point the device has reached a state of memory paging, so it is necessary in order to keep the user interface responsive. Visible action (activities that are visible to the user, but not in qlt;/use-permission) for example, one sitting at a dialogue in the foreground or next to other multi-window activities is considered extremely important and will not be killed if it is not required for the foreground time. Background activity (an activity that is not visible to the user and has been stopped) is no longer critical, so the system can safely kill its process to restore memory to other front or visible processes. If its process is to be killed when the user returns to activity (making it visible on the screen again), his onCreate (Bundle) method will be called with the savedInstanceState it previously delivered to onSaveInstanceState (Bundle), so that it can restart itself in the same state as the user last left it. A empty process is a process in which no action or other application components (such as Service or BroadcastReceiver) are carried out. They kill very quickly by the system as the memory becomes low. For this reason, any background operation you perform outside of the action must be performed in the context of a BroadcastReceiver or Service action to make sure the system knows that it needs to keep your process around. Sometimes Activity may need to develop a long-term operation that exists independently of the lifecycle itself. An example would be a camera app that lets you upload an image to a website. The download can take a long time, and the app should allow the user to leave the app while it is running. To do this, your activity must start the service in which the download takes place. This allows the system to correctly prioritize the process (considering it more important than other non-visible applications) at download time, regardless of whether the original action has been suspended, stopped, or completed. From the android.content.Context String class ACCESSIBILITY_SERVICE Use with getSystemService (java.lang.String) to get AccessibilityManager to provide user feedback for user interface events through registered event listeners. The line ACCOUNT_SERVICE used with getSystemService (java.lang.String) to obtain AccountManager to get intentions at the time of your choice. The line ACTIVITY_SERVICE used with getSystemService (java.lang.String) to obtain ActivityManager to interact with the global state of the system. The line ALARM_SERVICE used with getSystemService (java.lang.String) to obtain AlarmManager to get the intentions of the time of your choice. The line APPWIDGET_SERVICE used with getSystemService (java.lang.String) to get AppWidgetManager to access AppWidgets. The line APP_OPS_SERVICE used with getSystemService (java.lang.String) to get AppOpsManager for app operations on your device. The line AUDIO_SERVICE used with getSystemService (java.lang.String) to obtain AudioManager to handle volume, volume, modes and audio routing. The line BATTERY_SERVICE used with getSystemService (java.lang.String) to obtain BatteryManager to control battery condition. int BIND_ABOVE_CLIENT Flag for bindService (Intention, ServiceConnection, int): indicates that the customer's application linking this service considers the service to be more important than the app itself. Int BIND_ADJUST_WITH_ACTIVITY Flag for bindService (Intention, ServiceConnection, int): If the action binding allows you to put the value of the target service process depending on whether the activity is visible to the user, regardless of whether another flag is being used to reduce the amount that the total value of the customer process is used to impact it. int BIND_ALLOW_OOM_MANAGEMENT flag for bindService (Intention, ServiceConnection, int): Let the process of hosting a related service go through its normal memory management. int BIND_AUTO_CREATE flag for bindService (Intention, ServiceConnection, int): automatically create a service as long as there is a binding. int BIND_DEBUG_UNBIND flag for bindService (Intention, ServiceConnection, Int): Include debugging help for mismatched calls on unbind. int BIND_EXTERNAL_SERVICE flag for bindService (Intention, ServiceConnection, int): The associated service is an isolated foreign service. int BIND_IMPORTANT Flag for bindService (Intention, ServiceConnection, Int): This service is very important for the customer, so it should be brought to the front level of the process when the customer. Int BIND_INCLUDE_CAPABILITIES Flag for bindService: If the attachment from the application has certain features due to the foreground state, such as the activity or foreground service, this flag will allow the associated application to get the same features if it has the necessary permissions. int BIND_NOT_FOREGROUND Flag for bindService (Intention, ServiceConnection, int): don't let this link bring the target service process to the forefront of priority planning. int BIND_NOT_PERCEPTIBLE Flag for bindService (Intention, ServiceConnection, int): If the binding is from an application that is visible or visible to the user, lower the value of the target service below a noticeable level. int BIND_WAIVE_PRIORITY Flag for bindService (Intention, ServiceConnection, Int): do not affect the planning or priority of memory management of the target service hosting process. The line BIOMETRIC_SERVICE used with getSystemService (java.lang.String) to obtain BiometricManager to handle biometric and PIN/pattern/password authentication. The line BLOB_STORE_SERVICE used with getSystemService (java.lang.String) to obtain BlobStoreManager to promote and access drop data from a drop drop drop from the store drop supported by the system. Line BLUETOOTH_SERVICE to use with getSystemService to get BluetoothManager to use Bluetooth. Using CAMERA_SERVICE with to get CameraManager to interact with camera devices. The line CAPTIONING_SERVICE used with getSystemService (java.lang.String) to obtain captioningManager to obtain property subtitles and listen to changes in subtitle preferences. The line CARRIER_CONFIG_SERVICE used with getSystemService (java.lang.String) to obtain CarrierConfigManager to read the carrier configuration values. The line CLIPBOARD_SERVICE used with getSystemService (java.lang.String) to obtain ClipboardManager to access and modify the contents of the global clipboard. Line COMPANION_DEVICE_SERVICE Use getsystemService (java.lang.String) to obtain CompanionDeviceManager to control related String devices CONNECTIVITY_DIAGNOSTICS_SERVICE Use with getSystemService (java.lang.String) to obtain ConnectivityDiagnosticsManager to perform network connectivity diagnostics, as well as to obtain information about network connectivity from the system. The line CONNECTIVITY_SERVICE used with getSystemService (java.lang.String) to obtain ConnectivityManager to handle network connections management. The line CONSUMER_IR_SERVICE used with getSystemService (java.lang.String) to obtain ConsumerIrManager to transmit infrared signals from the device. int CONTEXT_IGNORE_SECURITY the flag for use with createPackageContext (String, int): ignore any security restrictions on the requested context, allowing it to always load. int CONTEXT_INCLUDE_CODE flag for use with createPackageContext (String, int): turn on the application code with context. int CONTEXT_RESTRICTED flag for use with createPackageContext (String, int): Limited context can disable certain features. The line CROSS_PROFILE_APPS_SERVICE used with getSystemService (java.lang.String) to obtain CrossProfileApps for cross-profile operations. The line DEVICE_POLICY_SERVICE used with getSystemService (java.lang.String) to obtain DevicePolicyManager to work with global device policy management. The line DISPLAY_SERVICE used with getSystemService (java.lang.String) to obtain DisplayManager to interact with display devices. The line DOWNLOAD_SERVICE used with getSystemService (java.lang.String) to obtain DownloadManager to request HTTP downloads. The line DROPBOX_SERVICE used with getSystemService (java.lang.String) to obtain a DropBoxManager copy to record diagnostic journals. The line EUICC_SERVICE used with getSystemService (java.lang.String) to obtain EuiccManager to control the eUICC device (embedded SIM). The line FILE_INTEGRITY_SERVICE used with getSystemService (java.lang.String) to obtain FileIntegrityManager. The line FINGERPRINT_SERVICE used with getSystemService (java.lang.String) to obtain to process fingerprint control. The line HARDWARE_PROPERTIES_SERVICE used with getSystemService (java.lang.String) to obtain to access the hardware property service. The line INPUT_METHOD_SERVICE used with getSystemService (java.lang.String) to obtain InputMethodManager to access input methods. The line INPUT_SERVICE used with getSystemService (java.lang.String) to obtain InputManager to interact with input devices. The line IPSEC_SERVICE used with getSystemService (java.lang.String) to obtain IpSecManager to encrypt outlets or networks with IPSec. The line JOB_SCHEDULER_SERVICE used with getSystemService (java.lang.String) to obtain a JobScheduler copy to manage random background tasks. The line KEYGUARD_SERVICE used with getSystemService (java.lang.String) to get KeyguardManager to control the keyboard. The line LAUNCHER_APPS_SERVICE used with getSystemService (java.lang.String) to obtain LauncherApps to request and monitor the launch of apps in user profiles. The line LAYOUT_INFLATER_SERVICE used with getSystemService (java.lang.String) to obtain LayoutInflater to inflate layout resources in this context. The line LOCATION_SERVICE used with getSystemService (java.lang.String) to get LocationManager to manage location updates. The line MEDIA_PROJECTION_SERVICE used with getSystemService (java.lang.String) to obtain a MediaProjectionManager copy to manage media projection sessions. The line MEDIA_ROUTER_SERVICE used with getSystemService (class) to obtain MediaRouter for media routing management and management. The line MEDIA_SESSION_SERVICE used with getSystemService (java.lang.String) to obtain MediaSessionManager to manage media sessions. The line MIDI_SERVICE used with getSystemService (java.lang.String) to obtain MidiManager to access the MIDI service. int MODE_APPEND file: for use with openFileOutput (String, int), if the file already exists, then write the data to the end of the existing file instead of erasing it. int MODE_ENABLE_WRITE_AHEAD_LOGGING database open flag: when the set opens with the record forward registration enabled by default. int MODE_MULTI_PROCESS This constant has been deprecated in API level 23. MODE_MULTI_PROCESS doesn't work reliably in some Android versions, and also doesn't provide any mechanism to agree on simultaneous changes between processes. Apps shouldn't try to use it. Instead, they should take a clear approach to managing data between processes such as ContentProvider. int MODE_NO_LOCALIZED_COLLATORS database with an open flag: when you set up, the database opens without the support of localized callouters. int MODE_PRIVATE Creation File: A default mode where the file you create can only be accessed to the call app (or all apps sharing the same ID int MODE_WORLD_READABLE This constant has been deprecated in API level 17. Creating the world's readable files is very dangerous and can cause security holes in the This is strongly discouraged; instead, applications should use a more formal interaction mechanism, such as ContentProvider, BroadcastReceiver and Service. There is no guarantee that this access mode will remain in the file, for example, when it is backed up and restored. int MODE_WORLD_WRITEABLE This constant has been deprecated in API level 17. Creating files that are narrated in the world is very dangerous and can cause security holes in applications. This is strongly discouraged; instead, applications should use a more formal interaction mechanism, such as ContentProvider, BroadcastReceiver and Service. There is no guarantee that this access mode will remain in the file, for example, when it is backed up and restored. The line NETWORK_STATS_SERVICE used with getSystemService (java.lang.String) to obtain NetworkStatsManager to request network usage statistics. The line NFC_SERVICE used with getSystemService (java.lang.String) to obtain NfcManager for NFC use. The line NOTIFICATION_SERVICE used with getSystemService (java.lang.String) to get NotificationManager to inform the user about background events. Line NSD_SERVICE Use with getSystemService (java.lang.String) to get NsdManager to handle The String Network Detection Service POWER_SERVICE Use with getSystemService (java.lang.String) to get PowerManager to manage power, including wake-up locks that allow you to keep the device while you're running long tasks. String PRINT_SERVICE PrintManager for printing and managing printers and printing tasks. Int RECEIVER_VISIBLE_TO_INSTANT_APPS Flag for RegisterReceiver (BroadcastReceiver, IntentFilter): The recipient can receive broadcasts from Instant Applications. The RESTRICTIONS_SERVICE Use getSystemService (java.lang.String) to get restrictions to get application restrictions and request permissions for limited operations. The line ROLE_SERVICE used with getSystemService (java.lang.String) to get RoleManager to manage roles. The line SEARCH_SERVICE used with getSystemService (java.lang.String) to obtain SearchManager to process searches. The line SENSOR_SERVICE used with getSystemService (java.lang.String) to obtain SensorManager to access the sensors. The line SHORTCUT_SERVICE used with getSystemService (java.lang.String) to obtain ShortcutManager to access the launcher label service. The line STORAGE_SERVICE used with getSystemService (java.lang.String) to obtain StorageManager to access system storage features. The line STORAGE_STATS_SERVICE used with getSystemService (java.lang.String) to obtain StorageStatsManager to access system storage statistics. String with getSystemService (java.lang.String) to obtain SystemHealthManager to access the health system (battery, power, memory, etc.) metrics. Using TELECOM_SERVICE with to obtain TelecomManager to manage the telecommunications functions of the device. The line TELEPHONY_IMS_SERVICE used with getSystemService (java.lang.String) to obtain ImsManager. The line TELEPHONY_SERVICE used with getSystemService (java.lang.String) to obtain The PhoneManager to control the device's telephony functions. The line TELEPHONY_SUBSCRIPTION_SERVICE used with getSystemService (java.lang.String) to obtain SubscriptionManager to handle the device's phone subscription management. The line TEXT_CLASSIFICATION_SERVICE used with getSystemService (java.lang.String) to obtain TextClassificationManager for text classification services. The line TEXT_SERVICES_MANAGER_SERVICE used with getSystemService (java.lang.String) to obtain TextServicesManager to access text services. The line TV_INPUT_SERVICE used with getSystemService (java.lang.String) to obtain TvInputManager to interact with TV inputs on the device. The line UI_MODE_SERVICE used with getSystemService (java.lang.String) to obtain UiModeManager to manage user interface modes. The line USAGE_STATS_SERVICE used with getSystemService (java.lang.String) to obtain UsageStatsManager to request device usage statistics. String USB_SERVICE use getSystemService (java.lang.String) to get UsbManager to access USB DEVICES (as a USB host) and to control the behavior of that device as a USB device. The line USER_SERVICE used with getSystemService (java.lang.String) to get UserManager to control users on devices that support multiple users. The line VIBRATOR_SERVICE used with getSystemService (java.lang.String) to produce a vibrator to interact with vibration equipment. The line VPN_MANAGEMENT_SERVICE used with getSystemService (java.lang.String) to obtain a VpnManager to manage profiles for the built-in VPN platform. The line WALLPAPER_SERVICE used with getSystemService (java.lang.String) to get com.android.server.WallpaperService to access wallpaper. The line WIFI_AWARE_SERVICE used with getSystemService (java.lang.String) to obtain WifiAwareManager to handle Wi-Fi Aware management. The line WIFI_P2P_SERVICE used with getSystemService (java.lang.String) to obtain WifiP2pManager to handle the management of Wi-Fi peer connections. The line WIFI_RTT_RANGING_SERVICE used with getSystemService (java.lang.String) to get Wi-FiRttManager for a range of Wi-Fi devices. The line WIFI_SERVICE used with getSystemService (java.lang.String) to obtain WifiManager to handle Wi-Fi access management. The line WINDOW_SERVICE used with getSystemService (java.lang.String) to get WindowManager to access the window manager protected static final int FOCUSED_STATE_SET invalid addContentView (View View, ViewGroup.LayoutParams params) Add an additional view of the content to the activity. Void closeContextMenu closes recently opened context menu, if you look. the void closeOptionsMenu () Progammatically closes the options menu. PendingIntent createPendingResult (int requestCode, Intent Data, int Flags) Create a new pendingIntent object that you can pass on to others for their use to send results back to onActivityResult (int, Int, Intention) callback. The final gap of dismissDialog (int ID) This method has been deprecated in API level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. The final void to dismissKeyboardShortcutsHelper () Dismiss the keyboard screen shortcuts. boolean dispatchGenericMotionEvent (MotionEvent ev) Called to handle common traffic events. boolean dispatchKeyEvent (KeyEvent Event) is designed to handle key events. boolean dispatchKeyShortcutEvent (KeyEvent event) is designed to handle a key label event. boolean dispatchPopulateAccessibilityEvent (AccessibilityEvent Event) is designed to process the population of AccessibilityEvents. boolean dispatchTouchEvent (MotionEvent ev) Is called to handle touchscreen events. boolean dispatchTrackballEvent (MotionEvent ev) Called to handle trackball events. void dump (Line Prefix, FileDescriptor fd, PrintWriter writer, String'args) Print activity state in a given stream. boolean enterPictureInPictureMode (PictureInPictureParams params) puts the action in picture-in-picture mode if possible in the current state of the system. the invalid enterPictureInPictureMode puts the action in picture mode, if possible in the current state of the system. T findViewById (int id) Finds a view that has been identified by android:id XML, which has been processed in onCreate (Bundle). The invalid finish of the Action (int requestCode) Force completes another action that you previously started with startActivityForResult (Intention, Int). This method has been removed at API 30. Instead, use finishActivity (int). Finish this activity, as well as all the actions directly underneath it in the current task, which have the same affinity. In effect, The Landing Completion () cancels the transition of entering the action scene and launches causing Activity to reverse its exit transition. Call this when your activity is completed and should be closed and the task should be completely removed as part of the completion of the root activity of the task. The void of finishFromChild (Child Activity) This method has been deprecated in the API level 30. Use the finish () instead. ActionBar getActionBar () Get a link to ActionBar of this the final getApplication app () Return the app that owns this activity. ComponentName getCallingActivity () Bring back the action name that caused this to be the view The getCallingPackage line () Return the name of the package that caused this action. If this activity breaks down because it cannot handle the modified configuration parameter (and thus its onConfigurationChanged (android.content.res.Configuration) method is not called, then you can use this method to detect a set of changes that occurred during the destruction process. ComponentName getComponentName () Returns the full name of the component of this action. Scene getContentScene () Get a scene representing the current contents of this window. TransitionManager getContentTransitionManager () Get TransitionManager responsible for default transitions in this window. Browse getCurrentFocus () Window.getCurrentFocus () on the window of this activity to bring back the now focused view. FragmentManager getFragmentManager () This method has been wilted in API level 28. Use FragmentActivity.getSupportFragmentManager (the intent of getIntent( ) To return the intention that started this activity. GetLastNonConfigurationInstance () Retrieving copy non-configuration data that has previously been returned to onRetainNonConfigurationInstance. LayoutInflater getLayoutInflater () Convenience to call Window.getLayoutInflater (). LoaderManager getLoaderManager () This method has been decreced in THEIs level 28. Use FragmentActivity.getSupportLoaderManager () The getLocalClassName line returns the class name for this action with the prefix package removed. int getMaxNumPictureInPictureActions () Return the number of actions that will be displayed in the user interface with the picture in the picture when the user interacts with the activity currently in picture mode. The final MediaController getMediaController () Receives a controller that should receive a media key and volume of events while this activity is in the foreground. MenuInflater getMenuInflater () Returns MenuInflater with this context. The final action of getParent () Return of parental activity if this view is a built-in child. The intention is to get ParentActivityIntent () Get the intention that will trigger the explicit target action specified by the logical parent of this action. SharedPreferences getPreferences (int mode) Remove the SharedPreferences facility to access preferences that are private for this activity. Uri getReferrer () Return information about who launched this activity. int getRequestedOrientation () Return of the current requested action orientation. The final SearchEvent getSearchEvent () During onSearchRequested () callback, this feature will return SearchEvent, which has called a callback if it exists. GetSystemService (String name) Return the pen to the service level by name. int getTaskId (the return of the task ID in which this activity is located. final CharSequence getTitle () final int getTitleColor () VoiceInteractor VoiceInteractor Remove the active VoiceInteractor that the user goes through to interact with this activity. The final Int getVolumeControlStream () Receives the proposed audio stream, the volume of which must be changed by the hardware control of the volume. GetWindow window () Check out the current window for action. WindowManager getWindowManager () Remove the window manager to show custom windows. boolean hasWindowFocus () Returns true if the main window of this activity now has a focus box. Invalid OptionsMenu () Announce that the options menu has changed, so must be recreated. boolean isActivityTransitionRunning () Returns to see if there are any activity transitions currently working on this activity. boolean isChangingConfigurations () Check to see if this action is in the process of destruction in order to be recreated with a new configuration. Is this activity embedded inside another activity? boolean isDestroyed () Returns true if the final onDestroy () call was made to the activity, so this instance is now dead. boolean isFinishing () Check whether this action is in the process of finishing, either because you called the finish () on it or someone else asked that it finish. boolean isImmersive () Bit indicating that this activity is immersive and should not be interrupted by notifications if possible. boolean isInMultiWindowMode () Returns correctly if the action is currently in multi-window mode. boolean isInPictureInPictureMode () Returns correctly if the action is currently in picture-in-picture mode. Boolean isLocalVoiceInteractionSupported, if you support the voice interaction service currently included, the return of voice interaction for use in action. boolean isTaskRoot () Return whether this action is the root of the task. Boolean isVoiceInteraction () Check to see if this action works as part of a voice interaction with the user. boolean isVoiceInteractionRoot () As isVoiceInteraction, but returns true only if it is also the root of voice interaction. The final Courser was driven by Ever (Uri Uri, String Projection, String Choice, String selectionArgs, String sortOrder) This method was humiliating in the API level 15. Instead, use CursorLoader. boolean moveTaskToBack (boolean nonRoot) Will move a task containing this activity into the back of the activity stack. Boolean navigateUpTo (Intent upIntent) Move from this activity to the activity specified by upIntent, ending this activity in the process. boolean navigateUpToFromChild (Child Activity, UpIntent Intention) This method has been deprecated in API level 30. Instead, use (android.content.Intent). The void in ActionModeStarted (ActionMode Mode) notifies you of the running mode. void onActivityReenter (int Intent Data) Is called when the action you've started with the activity transition exposes this action through a return activity transition, giving you the resultCode and any additional data from it. Emptiness onAttachFragment (fragment fragment) This method has been deprecated in api level 28. Use FragmentActivity.onAttachFragment (android.support.v4.app.Fragment) void onAttachedToWindow () Called when the main window associated with the action was attached to the window manager. invalid onBackPressed () Called when the action is detected to click on the user's back key. The void onConfigurationChanged (Configuration newConfig) is caused by the system when the device configuration changes during the time of operation. invalid onContentChanged () This hook is called whenever the view of the contents of the screen changes (due to a call in Window'setContentView (View, android.view.ViewGroup.LayoutParams) or Window'addContentView (View, android.viewGroup.LayoutParams)). boolean onContextItemSelected (MenuItem) This hook is called whenever an item is selected in the context of the menu. OnContextMenuClosed (Menu Menu) This hook is called whenever the context menu closes (either by the user, cancelling the menu with the Back/Menu button, or when selecting an item). Emptiness onCreate (Bundle savedInstanceState, PersistableBundle persistentState) Same as onCreate (android.os.Bundle), but called for those actions created with the attribute R.attr.persistableMode set for persistAcrossReboots. The void oncreateContextMenu (ContextMenu, View v, ContextMenu.ContextMenuInfo menuInfo) is called when the contextual menu for the view is about to be shown. CharSequence onCreateDescription () Create a new description for this activity. Void onCreateNavigateUpTaskStack (TaskStackBuilder builder) Identify a synthetic stack of tasks that will be generated while navigating Up from another task. Boolean onCreateOptionsMenu (Menu Menu) Initiate the contents of the standard Activity options menu. boolean onCreatePanelMenu (int featureId, menu menu) Default implementation Window.Callback.onCreatePanelMenu (int, Menu) for events. SeeCreatePanelView (int featureId) by default of Window.Callback.onCreatePanelView (int) for action. boolean onCreateThumbnail (Bitmap outBitmap, Canvas Canvas) This method has been deprecated in API level 28. The method does nothing and will be removed in the future. View OnCreateView (View Parent, String Name, Context context, AttributeSet attrs) Standard implementation layoutInflater.Factory2.onCreateView (View, Row, Context, AttributeSet) used to inflate with LayoutInflater returned to Context.getSystemService (class). View onCreateView (String name, Context, AttributeSet attrs) Standard implementation (String, Context, AttributeSet) used in inflating with LayoutInflater returned returned The void onDetachedFromWindow () Is called when the main window associated with the action has been separated from the window manager. The void on The Event Can't Draw during the period when their windows are animated in. boolean onGenericMotionEvent (MotionEvent Event) Is called when a general motion event has not been processed by any of the views within the action. Void onGetDirectActions (CancelSeny cancel, zgt; callback) Returns a list of direct actions supported by the app. boolean onKeyDown (int keyCode, KeyEvent event) is called when the key has been pressed and not processed by any of the performances inside the event. boolean onKeyLongPress (int keyCode, KeyEvent event) Implementation of KeyEvent.Callback-onKeyLongPress (int, KeyEvent): always returns false (can't handle the event). KeyEvent Event) By default, KeyEvent.Callback-onKeyMultiple (int, int, KeyEvent): always returns false (does not process the event). boolean onKeyShortcut (int keyCode, KeyEvent event) Is called when a key label event is not handled by any of the performance submissions. boolean onKeyUp KeyEvent event) Called when the key was released and did not process any of the submissions inside the activity. The void on The Call To Tell the Voice has led to the launch session's voice interaction. either because it was requested through a call to stop LocalVoiceInteraction () or because it was cancelled by the user. The void onLowMemory is called when the common system is running low in memory, and actively working processes should trim their memory usage. boolean onMenuItemSelected (int featureId, menuItem) Default implementation Window.Callback.onMenuItemSelected (int, MenuItem) for events. boolean onMenuOpened (int featureId, menu menu) Is called when the panel menu is opened by the user. Void on MultiWindowModeChanged (boolean isInMultiWindowMode) This method has been deprecated in API level 26. Instead, use onMultiWindowModeChanged (boolean, android.content.res.Configuration). The void on MultiWindowModeChanged (boolean isInMultiWindowMode, Configuration newConfig) is caused by the system when the action changes from full-screen mode to multi-window and visa-versa mode. boolean onNavigateUp () This method is called whenever the user chooses to navigate in the activity hierarchy of the application from the action bar. boolean onNavigateUpFromChild (Child Activity) This method has been deprecated in the API level 30. Use onNavigateUp instead. boolean onOptionsItemSelected (MenuItem) This hook is called whenever an item is selected in the options menu. emptiness onOptionsMenuClosedThis hook is called whenever the options menu closes (either by user, cancelling the back/menu menu or selecting the item). Void onPanelClosed (int featureId, menu menu) Implementation by default Window.Callback.onPanelClosed (int, Menu) for events. Void onPerformDirectAction (String ActionId, Kit Arguments, CancellationSignedSenyal, Consumer ResultListener) This is called to perform actions previously defined by the app. The void onPictureInPictureModeChanged (boolean isInPictureInPictureMode, newConfig configuration) Is called by the system when activity changes in image mode in the picture. Instead, use onPictureInPictureModeChanged (boolean, android.content.res.configuration). boolean onPictureInPictureRequested () This method is called the system in various cases where the image in image mode should be entered with support. Emptiness onPostCreate (Bundle savedInstanceState, PersistableBundle persistentState) Is the same as onPostCreate (android.os.Bundle), but is called for actions created with the attribute R.attr.persistableMode set for persistAcrossReboots. Void onPrepareNavigateUpTaskStack (TaskStackBuilder Builder) Prepare a synthetic stack of tasks that will be created while navigating up from another task. boolean onPrepareOptionsMenu (Menu menu) Preparing a standard menu of screen options for display. boolean onPreparePanel (int featureId, View View, Menu Menu) By default implementation window.Callback.onPreparePanel (int, View, Menu) for events. This is called when a user asks for help to provide links to content related to current activity. Void onProvideAssistData (data bundle) This is called when the user requests help to build the full intention ACTION_ASSIST Intention with the entire context of the current application. Void onProvideKeyboardShortcuts (KeyboardShortcutGroup Data List, Menu Menu, Int deviceId) Is called when requesting keyboard shortcuts for the current window. Uri onProvideReferrer () Redefine to generate the desired reference for the content currently displayed application. void onRequestPermissionsResult (int requestCode, String) permissions, int grantResults) Callback as a result of a permit request. The emptiness onRestoreInstanceState (Bundle savedInstanceState, PersistableBundle persistentState) Is the same as on RestoreInstanceState (android.os.Bundle), but is designed to be an activity created with the attribute R.attr.persistableModableMode, tuned to the tenacrossReboots. The object onRetainNonConfigurationInstance is called by the system as part of the destruction of the action due to a configuration change when it is known that the new instance will be &lt;/Bundle&gt; &lt;/Bundle&gt; for a new configuration. The void onSaveInstanceState (Bundle outState, PersistableBundle outPersistentState) Is the same as onaveInstanceState (Bundle), but is designed to be an activity created with the attribute of R.attr.persistableMode set for persistAcrossReboots. boolean onSearchRequested (SearchEvent searchEvent) This hook is called when the user signals a desire to start a search. boolean onSearchRequested () Is called when a user signals a desire to start a search. This method was at the API level 29. since Build.VERSION_CODES. P onSaveInstanceState is called after onStop (), so this hint is not accurate anymore: you should consider your condition not saved between onStart and onStop callback inclusive. The void onTopResumedActivityChanged (boolean isTop ResumActivity) is called when activity receives or loses the top resumed position in the system. boolean onTouchEvent (MotionEvent Event) Is called when the touchscreen event has not been processed by any of the views under it. boolean onTrackballEvent (MotionEvent Event) is called when the trackball has been moved and is not processed by any of the species within the event. onTrimMemory (int level) Is called when the operating system has determined that this is a good time for the process of trimming the necessary memory from its process. OnUserInteraction emptiness is called whenever a key, touch or trackball event is sent into action. The void onVisibleBehindCanceled () This method was wilted in THEIs 26. The functionality of this method is no longer supported as Build.VERSION_CODES. O and will be removed in a future release. Void onWindowAttributesChanged (WindowManager.LayoutParams params) This is called whenever the current window attributes change. The void onWindowFocusChanged (boolean hasFocus) is called when the current activity window acquires or loses focus. ActionMode onWindowStartingActionMode (ActionMode.Callback callback, int type) is called in running mode for this window. ActionMode onWindowStartingActionMode (ActionMode.Callback callback) gives the action the ability to control the user interface for the mode of action requested by the system. Invalid OpenContextMenu (View View) Software opens the context menu for a specific view. Invalid openOptionsMenu () Software opens the options menu. Invalid overridePendingTransition (int enterAnim, int exitAnim) Call immediately after one of the startActivity flavors (android.content.Intent) or finish () to indicate a clear transition animation to perform the next. The void puts off the transition when activity started with ActivityOptions.makeSceneTransitionAnimation (Activity, android.util.Pair). recreate (the) reason for this action to be recreated with a new instance. Invalid RegisterActivitycycleCallbacks (Application.ActivityLifecycleCallbacks Callbacks Callbacks) Registration Registration instance that receives lifecycle callbacks just for this Action. The invalid registerForContextMenu (View View) Registers the contextual menu that will be shown for this view (several views can show contextual menus). boolean releaseInstance ask that a local copy of the application of this action be released to release its memory. This method was removed at API level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Invalid reportFullyDrawn () Tell the system that your app is now fully drawn, for diagnostic and optimization purposes. DragAndDropPermissions requestsDragAndDropPermissions (DragEvent event) to create a DragAndDropPermissions facility associated with this activity and control permission to access URIs content associated with DragEvent. The final invalid request Formissions (String Permits, Int requestCode) requests permissions that will be granted to this application. The final invalid requestShowKeyboardShortcuts () Request the keyboard shortcuts the screen to show. boolean requestVisibleBehind (boolean visible) This method was faded at API level 26. The functionality of this method is no longer supported as Build.VERSION_CODES. O and will be removed in a future release. The final boolean query IsWindowFeature (int featureId) to include advanced window features. T requireViewById (int id) Finds a view that has been identified by the android:id XML attribute, which has been processed in onCreate (Bundle), or throws IllegalArgumentException if the ID is invalid or there is no corresponding representation in the hierarchy. the final invalid runOnUiThread (Runnable Action) runs this action on the user interface stream. Void SetActionBar (Toolbar) Install a toolbar to act as an action bar for this Activity window. Void setContenttransitionManager (TransitionManager tm) Install TransitionManager for use for the default transition in this window. ViewGroup.LayoutParams params) Set the action content on an explicit view. Void SetContentView (View View) Set the action content on an explicit view. The final set of default voidsDefaultKeyMode (int mode) Select the processing of the default keys for this activity. Void kitEnterSharedElementCallback (SharedElementCallback callback) When activityOptions.makeSceneTransitionAnimation (Activity, android.view, String) has been used to launch activities, the callback will be called to handle common items on the running Invalid setExitSharedElementCallback (SharedElementCallback callback) When ActivityOptions.makeSceneTransitionAnimation (Activity, android.view.View, Line) was used to run up activities, a callback will be called in to handle common items at the start of the activity. final void setFeatureDrawable (int featureId, qlt;/T extends View) Drawable) Convenience to call Window.setFeatureDrawable (int, Drawable). The final void setFeatureDrawableAlpha (int featureId, int alpha) Convenience to call Window.setFeatureDrawableAlpha (int, int). The final void setFeatureDrawableResource (int featureId, int resId) Convenience to call Window.setFeatureDrawableResource (int, int). The final void setFeatureDrawableUri (int featureId, Uri uri) Convenience to call Window.setFeatureDrawableUri (int, Uri). The invalid set ofFinishOnTouchOutside (boolean finish) determines whether this activity is finished when you touch the outside window. Unemanable set Immersive (boolean i) Adjust the current setting of immersive mode. The invalid setInheritShowWhenLocked (boolean inheritShowWhenLocked) determines whether to show this action on top of the lock screen whenever the lock screen is up, and this action has a different action behind it with a set of showWhenLock attributes. Emptiness setIntent (Intention newIntent) Change of Intent returns getIntent . The invalid setLocusContext (LocusId locusId, bundle bundle) sets LocusId for this activity. The final void set OfMediaController (MediaController controller) installs MediaController to send media keys and volume changes. The invalid setPictureInPictureParams (PictureInPictureParams params) updates the activity properties of the picture in the picture or installs it for use later at the entrancePictureInPictureMode () is called. This method has been wilted in API 24. No longer supported, starting with API 21. The final void setProgressBarIndeterminate (boolean indefinite) This method has been deprecated in api level 24. No longer supported, starting with API 21. The final void setProgressBarIndeterminateVisibility (boolean visible) This method was wilted in the API level 24. No longer supported, starting with API 21. The final void setProgressBarVisibility (boolean visible) This method has been deprecated in API level 24. No longer supported, starting with API 21. invalid setRequestedOrientation (int requestorientation) Change the desired orientation of this activity. The final set of voidsResult (int resultCode, data of intent) Call on this issue to establish the result, as a result of which your activity will return to the subscriber. The final void set ofResult (int resultCode) Call on this to establish the result that your activity will return to your subscriber. The final set of SecondaryProgress (int secondaryProgress) voids this method has been removed at API 24. No longer supported, starting with API 21. invalid setShowWhenLocked (boolean showWhenLocked) determines whether to show an action Top of the lock screen whenever the lock screen and activity resumes. The invalid SetTaskDescription (ActivityManager.TaskDescription taskDescription) establishes information describing the task with this action for a presentation within the user interface of the Recents system. Invalid setTheme (int resid) Set a basic theme for this context. invalid setTitle (CharSequence Name) Change the name associated activities. invalid setTitle (int titleId) Name change associated with this activity. This method was highlighted at API 21. Instead, use action bar styles. boolean setTranslucent (semi-transparent) Activity Conversion, which is particularly with R.attr.windowIsTranslucent or R.attr.windowIsFloating attribute, to full-screen opaque activity, or convert it from opaque back into translucent. The invalid TurnScreenOn set (boolean turnScreenOn) determines whether to turn on the screen when the action resumes. The invalid setvisible (boolean visible) Control whether the main window of this activity is visible. The final set of The FreeumeControlStream (int streamType) offers audio streams, the volume of which must be changed by the hardware volume control. invalid setVrModeEnabled (boolean enabled, ComponentName askedComponent) to turn on or off virtual reality (VR) mode for this activity. boolean shouldShowRequestPermissionRationale (String permit) gets whether you should show the user interface with justification before asking for permission. boolean shouldUpRecreateTask (TargetIntent Intention) Returns correctly if the application has to recreate the task when navigating 'up' from this activity using targetIntent. boolean showAssist (Bundle args) Ask to show the user the current assistant. The final boolean showDialog (int id, Bundle args) This method has been deprecated in api level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. The final void of showDialog (int id) This method has been deprecated in THEO level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Invalid showLockTaskEscapeMessage () Shows the user that the system has identified a message in order to tell the user how to get out of the lock problem mode. ActionMode StartActionMode (ActionMode.Callback callback, int type) Start this type mode. ActionMode StartActionMode (ActionMode.Callback callback) Run ActionMode-TYPE_PRIMARY. Invalid startActivities (intentions, kit options) Start a new action. Invalid startActivities (intentions) Same as startActivities (android.content.Intent, android.os.Bundle) without these options. Invalid startActivity (intention intention) Same as startActivity (android.content.Intent, android.os.Bundle) without specified options. invalid startActivity (intention, kit options) Start a new action. invalid startActivityForResult Intentions, Int requestCode) Same as call startActivityForResult (android.content.Intent, int, android.os.Bundle) with no options. invalid startActivityForResult (Intention, Int requestCode, Options Kit) Running up for which you would like a result when it is finished. invalid startActivityFromChild (Child Activity, Intention Intention This method has been wilted at API 30. Use androidx.fragment.app.FragmentActivity-startActivityFromFragment (androidx.fragment.app.Fragment,Intent,int) invalid startActivityFromChild (Child Activity, Intention, Int requestCode, Options Kit) This method has been faded in API level 30. Use androidx.fragment.app.FragmentActivity-startActivityFromFragment (androidx.fragment.app.Fragment,Intent,int,Bundle) invalid startActivityFromFragment (snippet, intent, int requestCode, kit options) This method has been deprecated in API level 28. Use androidx.fragment.app.FragmentActivity-startActivityFromFragment (androidx.fragment.app.Fragment,Intent,int,Bundle) invalid startActivityFromFragment (fragment, intent, int requestCode) This method was degraded at API 28. Use androidx.fragment.app.FragmentActivity-startActivityFromFragment (androidx.fragment.app.Fragment,Intent,int) boolean startActivityIfNeeded (Intention, Int requestCode, Options Kit) Special variation to start activities only if a new instance of action is needed to handle this intent. boolean startActivityIfNeededed (Intention, Int requestCode) Same as call startActivityIfNeeded (android.content.Intent, int, android.os.Bundle) without options. Void startIntentSender (IntentionSender, Intention fillInintent, Int flagsMask, IntValu flagses, Int extraFlags) Same as call startIntentSender (android.content.IntentSender, android.content.Intent, int, int, int, int, android.os.Bundle) with no options. invalid startIntentSender (IntentionSender, Intention fillInIntent, Int flagsMask, IntValu flagses, Int extraFlags, Options Kit) How to startActivity (android.content.Intent, android.os.Bundle), but taking IntentSender to start; For more information, see startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int, android.os.Bundle). invalid startIntentSenderForResult (IntentionSender Intentions, Int requestCode, Int flagsInintent, Int flagsMask, Int flagsValues, Int extraFlags) Same as call startIntentSenderForResult (android.content.IntentSender, int, intent.content., Intent. int, int, int, int. invalid startIntentSenderForResult (IntentionSender Intention, Int requestCode, Int flagsIntent, Int flagsValues, Int flagsValues, Int extraFlags, Bundle options) As startActivityForultRes (android.content.Intent, Int) , IntentionSender, Int requestCode, Intention fillInIntent, Int flagsMask, Int flagsValues, Int extraFlags, Options Kit) This has been deprecated in API level 30. Instead, use startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int, android.os.Bundle). invalid startIntentSenderFromChild (Child activity, activity, intention, Int requestCode, Intention fillInIntent, Int flagsMask, Int flagsValues, Int extraFlags) This method has been deprecated in API level 30. Instead, use startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int). The invalid startLocalVoiceInteraction (Bundle privateOptions) starts a local voice session. Invalid startLockTask () Request to put this action in a mode in which the user is blocked by a limited set of applications. this method has been wilted in the API level 15. Use the new CursorLoader class with LoaderManager instead; It's also available on older platforms through the Android compatibility package. boolean startNextMatchingActivity (intention, kit options) A special version of the start-up, for use when replacing other components of activity. boolean startNextMatchingActivity (intention intention) Same as call startNextMatchingActivity (android.content.Intent, android.os.Bundle) with no options. invalid startPostponedEnterTransition () Beginning of deferred transitions after the postponementBusiness () was called. invalid startSearch (String initialquery, boolean selectInitial-keri, Bundle appSearchData, boolean globalSearch) This hook is designed to launch a search engine interface. Invalid stopLocalVoiceInteraction (android.os.Bundle). invalid stopLockTask () Stop the current task from blocking. invalid stopManagingCursor (Cursor c) This method was wilted in the api level 15. Use the new CursorLoader class with LoaderManager instead; It's also available on older platforms through the Android compatibility package. invalid takeKeyEvents (boolean get) Request that key events come to this activity. invalid triggerSearch (String query, Bundle appSearchData) Similar to startSearch (String, boolean, boolean), but actually launches a search query after calling a search dialogue. Invalid unregistered ActivityLifecycleCallbacks (Application.ActivityLifecycleCallbacks callbacks callbacks callbacks) Unregistered Application.ActivityLifecycleCallbacks, previously registered in the ActivityLifecycleCallbacks (Application.ActivityLifecycleCallbacks). Invalid unregisteredForContextMenu (View View) Prevents the context menu that should be shown for this view. Invalid attachBaseContext (Context newBase) Set the basic context for this ContextWrapper. Void onActivityResult (int requestCode, int resultCode, Intent Data) Is called in action that you giving you the requestCode from which you launched it, the resultCode it returned, and any additional data from it. Void onApplyThemeResource (Theme. theme, int resid, boolean first) Called setTheme (Resources.Theme) and getTheme () apply thematic resource to the current subject of the theme. Void on ChildTitle Displaced (Operation ChildActivity, CharSequence Title) invalid savedInstanceState) Called when the action is launched. Dialog onCreateDialog (int id) This method has been wilted at API 15. The old version without arguments onCreateDialog (int, android.os.Bundle). Dialog onCreateDialog (int id, Bundle args) This method has been wilted at API 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Void on Destroy, do any final clean-up before the action is destroyed. Void on NewIntent (Intention Intention) This is called for actions that set launchMode on a singleTop in their package, or if a customer used the Flag of Intent FLAG_ACTIVITY_SINGLE_TOP when calling startActivity (Intention). OnPause emptiness is called as part of the lifecycle of the action, when the user no longer actively interacts with the action, but it is still visible on the screen. The emptiness onPostCreate (Bundle savedInstanceState) is called when the launch activity is completed (after onStart () and onRestoreInstanceState (Bundle) have been called). onPostResume is called when the resumption of activity is completed (after onResume() has been called. emptiness onPrepareDialog (int id, Dialogue Dialogue, Bundle args) This method has been deprecated in api level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Emptiness onPrepareDialog (int id, Dialogue Dialogue) This method has been deprecated in API level 15. The old version without arguments onPrepareDialog (int, android.app.Dialog, android.os.Bundle). The void on Restart is called after onStop () when the current action is re-displayed to the user (the user has returned to it). OnRestoreInstanceState (Bundle savedInstanceState) This method is called after onStart, when the action is re-initiated from the previously saved state given here in the savedInstanceState. onResume void is called after onRestoreInstanceState (Bundle), onRestart (), or onPause, for your activities to start interacting with the user. The void onSaveInstanceState (Bundle outState) is called to extract the state in each instance from the action before being killed, so that the state can be restored in onCreate (Bundle) or onRestoreInstanceState (Bundle, inhabited by this method, will be transferred to both). onStart emptiness is called after onCreate (Bundle) - or after onRestart, when the action has been stopped, but now is displayed again to the user. onStop emptiness is called when you are no longer visible to the user. The void onTitleChanged (Name CharSequence, int color) void onUserLeaveHint is called as part of the activity lifecycle when the action is about to fade into the background as a result of user choice. Of android.content.ContextWrapper boolean bindIsolatedService (Service of Intent, int flags, flags, instanceName, Artist, ServiceConnection conn) Variation bindService (Intention, ServiceConnection, Int), which in a particular case of isolated services allows the subscriber to generate multiple instances of the service from a single component of the declaration. boolean bindService (Intention Service, int Flags, Artist, ServiceConnection conn) Same as bindService (android.content.Intent, android.content.ServiceConnection, int) with the executor to manage ServiceConnection callbacks. boolean bindService (Intention Service, ServiceConnection conn, Int Flags) Connect to the app service, creating it if necessary. Int checkCallingOrSelfPermission (String Permit) Determine whether the IPC call process has been granted or you have been granted a certain permit. Int checkCallingOrSelfUriPermission (Uri uri, int modeFlags) Determine whether the IPC call process has been granted permission to access a specific URI. int checkCallingPermission (String Resolution) Determine whether the IPC call process you are processing has received some permission. Int checkCallingUriPermission (Uri uri, int modeFlags) Determine whether there was permission to access a particular URI call process and user ID. Determine whether this permission is allowed for a particular process and the user ID that runs in the system. Determine if you have been granted a given permission. Int checkUriPermission (Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags) Check both Uri and normal resolution. int checkUriPermission (Uri uri, int pid, int uid, int modeFlags) Determine whether permission has been granted to access a particular URI. invalid clearWallpaper () This method is deprecated. Use WallpaperManager.clear instead. This method requires time from the caller Manifest.permission.SET_WALLPAPER. Context createAttributionContext (String attributionTag) Bring back a new context object for the current context, but attribute to another tag. Context of the creationContext Configuration (Configuration overrideConfiguration) Return a new context object for the current context, but whose resources are adjusted according to this configuration. The context of creatingDeviceProtectedStorageContext () Return of a new Context object for current context, but whose API storage is supported by a secure device. The context of creating The DisplayContext (Display Display) Bring back a new context for the current context, but whose resources are adjusted according to the metrics of this display. Context of The PackageContext (String packageName, int flags) Return of the new Context object for app's name. The context of The WindowContext (type int, kit options) creates a context for Window. The databaseList line returns an array of lines, naming private databases associated with this Context application package. boolean deleteDatabase (String name) Remove the existing private database of S'LiteData associated with the application package of this context. boolean deleteFile (String name) Delete this private file associated with this Context app package. boolean deleteSharedPreferences (String name) Delete an existing common preference file. Invalid ForcedCallingOrSelfPermission (String resolution, line message) If neither you nor the IPC call process you are processing has been granted a specific permission to quit SecurityException. Invalid enforceCallingOrSelfUriPermission (Uri uri, int modeFlags, String message) If the process of calling IPC or you have not been granted permission to access a particular URI, drop SecurityException. invalid enforceCallingPermission (Line Resolution, Line Message) If the IPC call process you are processing has not been granted a specific permission to quit SecurityException. invalid enforceCallingUriPermission (Uri uri, Int modeFlags, String message) If the call process and user ID were not granted permission to access a particular URI, quit SecurityException. invalid enforcePermission (Line Resolution, Int pid, int uid, String message) If this permission is not allowed for a particular process and the user ID working in the system, quit SecurityException. Void enforceUriPermission (Uri uri, ReadPermission line, writePermission line, Int pid, Int uid, Int modeFlags, String Message) Provide both Uri and Normal Resolution. invalid enforceUriPermission (Uri uri, int pid, int uid, int modeFlags, String message) If a particular process and user ID have not been granted permission to access a particular URI, drop SecurityException. String's fileList returns an array of lines, naming personal files associated with the context package. Context getApplicationContext () Return the context of a single, global application object of the current process. ApplicationInfo getApplicationInfo () Return full app information for this context package. AssetManager getAssets returns an AssetManager copy for the app package. The getBaseContext context returns the absolute path to a specific app cache catalog in the file system. ClassLoader getClassLoader returns a class loader that you can use to get classes in this package. The getCodeCacheDir file returns the absolute path to a specific app cache directory on a file system designed to store cached code. ContentResolver getContentResolver () Return of the ContentResolver instance for the app package. File () Returns the absolute path to the directory in the file system, where all the personal files belonging to this application are stored. File File name) Returns the absolute path to the file system where the database created with openOrCreateDatabase (String, int, S'LiteDatabase) is stored. CursorFactory). Get a getDir (String name, int mode) file by creating a new directory if necessary, in which the app can place its own user data files. Displaying getDisplay () Get the display this context is associated with. file getExternalCacheDir () Returns the absolute path to the app catalog on the primary shared/external storage device where the app can place the cache files that it has. The getExternalCacheDirs file returns absolute paths to app directories on all shared/external storage devices where the app can place the cache files it has. The getExternalFilesDir file (string type) returns the absolute path to the catalog on the primary general/external storage device, where the app can place the persistent files that it has. The getExternalFilesDirs file (line type) returns absolute paths to app catalogs on all shared/external storage devices where the app can place the persistent files that it has. File ' getExternalMediaDirs () This method is ediciable. These directories still exist and are scanned, but developers are encouraged to go to inserting content into the MediaStore collection directly, as any app can bring new media to MediaStore without any permission, starting with Build.VERSION_CODES. The getFileStream (String name) file returns the absolute path on the file system where the file created with openFileOutput (String, int) is stored. The getFilesDir file returns the absolute path to the directory in the file system, where files created with openFileOutput (String, int) are stored. The artist receivesMainExecutor () Return the artist who will perform the covered tasks on the main thread associated with this context. Looper getMainLooper () Return Looper for the main flow of the current process. The getNoBackupFilesDir file returns the absolute path to the catalog on a file system similar to getFilesDir(File getObbDir () Return the primary general/external storage directory where the OBB files of this app (if any) can be found. The getObbDirs file returns absolute paths to app directories on all shared/external storage devices where you can find OBB app files (if any). Line getPackageCodePath () Bring back the full path to the basic Android package of this context. PackageManager get ThePackageManager (The Return of PackageManager Copy to find global information about the package. The getPackageName line () Return the name of the package of this app. Line getPackageResourcePath () Bring the full path to the basic Android package of this context. GetResources Resources () Returns copy resources for Application. SharedPreferences getSharedPreferences (String name, int mode) Extracting and holding withholding from the 'name' preference files, returning SharedPreferences, with which you can get and change its values. GetSystemService (String name) Return the pen to the system-level service by name. The getSystemServiceName line is called a system-level service that is represented by the specified class. Resources.Theme getTheme () Return of the subject object related to this context. Drawable getWallpaper () This method is a deprecation. Use WallpaperManager.get instead. Int getWallpaperDesiredMinimumHeight () This method is a deprection. Instead, use WallpaperManager.getDesiredMinimumHeight. int getWallpaperDesiredMinimumWidth () This method is decretal. Instead, use WallpaperManager.getDesiredMinimumWidth. An invalid UriPermission grant (String toPackage, Uri uri, int modeFlags) grants permission to access a particular Uri package, regardless of whether the package has a general permission to access Uri content provider. boolean isDeviceProtectedStorage indicates whether API messages are supported by storage of this context by a secure device. boolean is limited () indicates whether this context is limited. boolean moveDatabaseFrom (Context sourceContext, String name) Move an existing database file from this source storage context into this context. boolean moveSharedPreferencesFrom (Context sourceContext, String name) Move the existing common preference file from this background of storing source data into this context. FileInputStream openFileInput (String name) Open a private file associated with a package of applications of this context for reading. FileOutputStream openFileOutput (String name, Int mode) Open a private file associated with the application package of this context for writing. Open site S'LiteDatabase OpenOrCreateDatabase (String name, int mode, S'LiteDatabase factory. CursorFactory, DatabaseError HandlerHandler bug) opens a new private database of S'LiteData associated with the application package of this context. Open site S'LiteDatabase openOrCreateDatabase (String name, int mode, S'LiteDatabase plant. CursorFactory) opens a new private database of S'LiteData associated with the context application package. Drawable peekWallpaper () This method is deprecated. Instead, use WallpaperManager.peek. The intention of registerReceiver (BroadcastReceiver receiver, IntentFilter filter) register BroadcastReceiver to run the main activity stream. Intention registerReceiver (Receiver BroadcastReceiver, IntentFilter filter, int flags) Registration to receive intentional broadcasting, with the receiver additionally exposed to Instant Apps. Intention registerReceiver (BroadcastReceiver Receiver, IntentFilter Filter, Line BroadcastPermission, Handler Planner, Int Register to receive broadcast intentions, to run in context. Intention registerReceiver (BroadcastReceiver Receiver, IntentFilter filter, String BroadcastPermission, Handler Planner) Registration Registration intentions of transmissions to run in the context of the planner. Emptiness removeStickyBroadcast (Intention intention) This method is deprecated. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. invalid removeStickyBroadcastAsUser (intention, UserHandle user) This method is invalid. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. invalid cancellation of UriPermission (Uri uri, int modeFlags) Remove all permissions to access a specific Uri content provider that was previously added with the UriPermission (String, Uri, int) grant or any other mechanism. invalid revokeUriPermission (String targetPackage, Uri uri, int modeFlags) Remove permissions to access a specific Uri content provider that was previously added using grantUriPermission (String, Uri, int) for a specific target package. Invalid sendBroadcast (Intention, Line receiverPermission) Broadcasting of this intent to all interested BroadcastReceivers, allowing the optional required permission to be enforced. invalid sendBroadcast (Intention Intention) Broadcasting of this intent to all interested BroadcastReceivers. Invalid sendingBroadcastAsUser (intention, userHandle user) version sendBroadcast (android.content.Intent), allowing you to specify the user of the broadcast will be sent. invalid sendBroadcastAsUser (Intention, UserHandle User, ReceiverPermission Line) Version sendBroadcast (android.content.Intent, java.lang.String) that allows you to specify the user's broadcast will be sent. Void send OrderBroadcast (Intention, Line receiverPermission, String ReceiverAppOp, BroadcastReceiver resultReceiver, Handler Planner, int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.string, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.String, android.os. void sendOrderBroadadcast (Intention, Int. Line of initialData, Bundle initialExtras, Bundle options) invalid sendOrderedBroadcast (Intention, Line receiverPermission, BroadcastReceiver resultReceiver, Scheduler Handler, scheduler, that allows you to get the data back from the broadcast. Void sendOrderedBroadcast (Intention, Line receiverPermission) Broadcasting this intent to all interested BroadcastReceivers, supplying them one at a time to the more preferred receivers to consume the broadcast before it is delivered to less preferred receivers. Void DispatchOnBroadcastAsUser (Intention, UserHandle User, String ReceiverPermission, BroadcastReceiver resultReceiver, Handler Planner, Int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.string, invalid sendStickyBroadcast (Intention) This method is deprecated. to report that something has changed, with a different mechanism for applications to get the current value when needed. Invalid DispatchStickyBroadcastAsUser (Intention Intention, UserHandle User) This method is deprecated. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Invalid dispatchStickyOrderedBroadcast (Intention Intention, BroadcastReceiver resultReceiver, Handler planner, Int initialCode, String initialData, Bundle initialExtras) This method is invalid.. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Invalid dispatchStickyOrderedBroadcastAsUser (Intention Intention, UserHandle User, BroadcastReceiver resultReceiver, Handler Planner, Int initialCode, String initialData, initial BundleExtras) This method is deprecated. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for to get the current value when you need it. Invalid setTheme (int resid) Set a basic theme for this context. Void setWallpaper (Bitmap bitmap) This method is deprecated. Instead, use WallpaperManager.set. This method requires time from the caller Manifest.permission.SET_WALLPAPER. Void Void data) This method is deprecated. Instead, use WallpaperManager.set. This method requires time from the caller Manifest.permission.SET_WALLPAPER. Invalid startActivities (intentions, kit options) Launch several new actions. Invalid startActivities (intentions) Same as startActivities (android.content.Intent, android.os.Bundle) without these options. Invalid startActivity (intention intention) Same as startActivity (android.content.Intent, android.os.Bundle) without specified options. invalid startActivity (intention, kit options) Start a new action. ComponentName startForegroundService (Intention Service) is similar to startService (android.content.Intent), but with the implicit promise that the service will call startForeground (int, android.app.Notification) as soon as it starts working. boolean startInstrument (ComponentName className, ProfileFile Line, Bundle Arguments) Start performing the tooling class. Void startIntentSender (Intention, Intention fillInIntent, Int flagsMask, Int flagsValues, Int extraFlags) Same, that startIntentSender (android.content.IntentSender, android.content.Intent, int, int, int, int, int, int, int. invalid startIntentSender (Intention, fillIntentIntent, Int flagsMask, Int flagsValues, Int extraFlags, Bundle options) How to startActivity ComponentName StartService (Intention Service) Request to launch this application service. boolean stopService (Name of Intent) Request to stop this application service. invalid unbindService (ServiceConnection conn) Disconnection from the application service. invalid unregisteredReceiver (BroadcastReceiver) Unregistered previously registered BroadcastReceiver. For a service previously associated with bindService (Intention, ServiceConnection, int) or related method, change the way the service is managed in relation to other processes. From the class android.content.Context boolean bindIsolatedService (Service of Intention, int flags, Line instanceName, Artist, ServiceConnection conn) VariationService bind (Intention, ServiceConnection, Int), which, in a particular case of isolated services, allows the subscriber to generate multiple copies of the service from one component of the declaration. boolean bindService (Intention Service, int Flags, Artist, ServiceConnection conn) Same as bindService (android.content.Intent, android.content.ServiceConnection, int) with the executor to manage ServiceConnection callbacks. abstract boolean bindService (Intention Service, ServiceConnection conn, int flags) Connect to the service creating it if necessary. boolean bindServiceAsUser (ServiceConnection conn, int flags, UserHandle user) contacted the service in this user in the same way as bindService (android.content.Intent, android.content.ServiceConnection, int). Abstract Check IntCallingOrSelfPermission (Line Resolution) Determine whether the IPC call process has been granted or you have been granted a certain permit. Abstract Check IntCallingOrSelfUriPermission (Uri uri, int modeFlags) Determine whether the IPC call process has been granted permission to access a particular URI. Abstract Check IntCallingPermission (String permission) Determine whether the IPC call process you are processing has been defined by resolution. Abstract Check IntCallingUriPermission (Uri uri, int modeFlags) Determine whether there was permission to access a particular URI call process and user ID. Determine whether this permission is allowed for a particular process and the user ID that runs in the system. Abstract Check IntSelfPermission (Line Resolution) Determine whether you have been granted a certain permit. Abstract check IntUriPermission (Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags) Check both Uri and normal resolution. abstract int checkUriPermission (Uri uri, int pid, int uid, int modeFlags) Determine whether permission has been granted to access a particular URI. clearWallpaper's abstract void () This method was introduced in the API level 15. Use WallpaperManager.clear instead. This method requires time from the caller Manifest.permission.SET_WALLPAPER. Context createAttributionContext (String attributionTag) Bring back a new context object for the current context, but attribute to another tag. abstract context creates ConfigurationContext (OverrideConfiguration Configuration) Returning a new context object for the current context, but whose resources are adjusted according to that configuration. Abstract Context to createContextForSplit (String splitName) Bring back a new context object for a given split name. the abstract context of creatingDeviceProtectedStorageContext (the return of a new context object for the current context, but whose API storage is supported by a secure device. abstract context createsDisplayContext (Display) To return a new context object to the current context, but whose resources are adjusted according to the metrics of the display. Abstract Context createPackageContext (String packageName, Int Flags) Bring back a new context object for the app name. The context of The WindowContext (type int, kit options) creates context for a non-activity window. abstract databaseList returns an array of lines, naming private databases associated with this suite of applications abstract boolean deleteDatabase (String name) Remove the existing private base of S'LiteData related to this app package. abstract boolean deleteFile (String name) Delete this private file associated with this Context application package. abstract boolean deleteSharePreferences (String name) Delete an existing file of common preferences. If neither you nor the IPC call process you work from has been granted a certain permission, drop SecurityException. If you don't have permission to access a particular URI, leave SecurityException if you don't have permission to access a specific URI. Abstract Space enforceCallingPermission (String resolution, line message) If the process of calling IPC you are processing has not been granted a specific permission to quit SecurityException. If the abstract void that applies CallingUriPermission (Uri uri, int modeFlags, String message) If the call process and user ID have not been granted permission to access a particular URI, drop SecurityException. Abstract void ensures compliance With ThePermission (Resolution of the Line, Int pid, int uid, String message) If this permission is not allowed for a particular process and the user ID running in the system, drop SecurityException. abstract void to forceUriPermission (Uri uri, string readmission, writePermission line, int pid, int uid, int modeFlags, String message) To enforce both Uri and normal permission. Abstract void to forceUriPermission (Uri uri, int pid, int uid, int modeFlags, String message) If a particular process and user ID have not been granted permission to access a particular URI, drop SecurityException. string's fileList returns an array of lines, naming personal files associated with the application package of that context. the abstract context of getApplicationContext () The return of the context of a single, global object to the current process application. Abstract ApplicationInfo getApplicationInfo () Return full application information for this context package. Abstract AssetManager getAssets () Returns a copy of AssetManager for the app package. The getAttributionTag line can be used in complex applications of logically separate parts of the application. the getCacheDir abstract file returns the absolute path to a specific application cache catalog on the file system. Abstract ClassLoader getClassLoader ,- Return class loader that you can use to get classes in this package. the getCodeCacheDir abstract file returns the absolute path to a specific app cache catalog in a file system designed to store cached code. final int getColor (int id) returns the color associated with a certain resource and stylized for the current theme. The final ColorStateList getColorStateList (int id) returns a list of color status associated with a specific resource ID and stylized as a current theme. Abstract Abstract getContentResolver () Return of the ContentResolver instance for the app package. the getDataDir abstract file returns the absolute path to the catalog in the file system, where all the personal files belonging to this application are stored. the abstract getDatabasePath (String name) file returns the absolute path in the file system where the database created with openOrCreateDatabase (String, int, S'LiteDatabase) is stored. CursorFactory). We remove the getDir (String name, int mode) file by creating a new directory where the app can place its own user data files if necessary. GetDisplay () Get the display this context is associated with. Final Drawable getDrawable (int id) returns a drawing object associated with a specific resource id and stylized as a current theme. the getExternalCacheDir abstract file returns an absolute path to a specific application directory on the primary shared/external storage device, where the app can place the cache files it owns. the getExternalCacheDirs abstract file returns absolute paths to app catalogs on all shared/external storage devices where the app can place the cache files it has. the getExternalFilesDir (line type) abstract file returns the absolute path to the catalog on the primary shared/external storage device, where the app can place the persistent files that it has. the getExternalFilesDirs (Line Type) file returns absolute paths to application directories on all shared/external storage devices where the app can place the persistent files that it has. This method was adopted at API 30. These directories still exist and are scanned, but developers are encouraged to move to insert content into the MediaStore collection directly, as any app can bring new media to MediaStore without any permissions, starting at Build.VERSION_CODES. the abstract getFileStreamPath (String name) returns the absolute path on the file system where the file created with openFileOutput (String, int) is stored. the abstract getFilesDir file returns the absolute path to the catalog on the file system, where files created with openFileOutput (String, int) are stored. The artist receivesMainExecutor () Return the artist who will perform the covered tasks on the main thread associated with this context. Abstract Looper getMainLooper () Bring back Looper for the main flow of the current process. the abstract file getNoBackupFilesDir () Returns the absolute path to the catalog on a file system similar to getFilesDir(). abstract file getObbDir () Return the basic general/external storage directory where the OBB files of this application (if any) can be found. getObbDirs abstract file absolute paths to app directories on all shared/external storage devices where you can find OBB app files (if any). There are). getOpPackageName () Return the name of the package that should be used to call AppOpsManager out of this context, so checking the uid app manager ops will work with the name. the abstract line getPackageCodePath () Bring back the full path to the basic Android package of this context. Abstract PackageManager getPackageManager (The Return of PackageManager copy to find global information about the package. the abstract line getPackageName () Return the name of the package of this application. the abstract line getPackageResourcePath () Bring the full path to the basic Android package of this context. the abstract resources getResources () Returns an example of resources for the application package. abstract SharedPreferences getSharedPreferences (String name, int mode) Extract and hold the contents of the 'name' preference file, returning SharedPreferences through which you can get and change its values. final line getString (int resId, object... formatArgs) Returns a localized formatted line from the default line table of the application package package, replacing the format arguments set out in Formatter and String.format (String, Object...). The final string getString (int resId) returns a localized line from the default application package table. The ultimate serviceClass is to return the pen to the system level of service by class. getSystemService (String name) Return the pen to the system level of the service by name. the abstract line getSystemServiceName (Class serviceClass) Gets the name of the system-level service, which is represented by the specified zlt;?? It's a class. The final CharSequence getText (int resId) Return of the localized, stylized CharSequence from the default application package line table. Use WallpaperManager.get instead. Abstract int getWallpaperDesiredMinimumHeight () This method was humiliating in the API level 15. Instead, use WallpaperManager.getDesiredMinimumHeight. Abstract int getWallpaperDesiredMinimumWidth () This method was humiliating at API level 15. Instead, use WallpaperManager.getDesiredMinimumWidth. The abstract invalid grant OfUriPermission (String toPackage, Uri uri, int modeFlags) grants permission to access a particular Uri package, regardless of whether the package has a general permission to access Uri content provider. abstract boolean isDeviceProtectedStorage indicates whether API messages are supported by storage of this context by a secure device. boolean is limited () indicates whether this context is limited. abstract boolean moveDatabaseFrom (Context sourceContext, String name) Move an existing database file from this source storage context into this context. abstract boolean (Контекст sourceContext, Название строки) Переместить&lt;/T&gt; &lt;/T&gt; &lt;/T&gt; file of general preferences from this background of storing the original data in this context. The ultimate TypedArray getStyledAttributes (AttributeSet set, int' attrs) Get style attribute information in this context of the theme. Ultimate TypedArray getStyledAttributes (AttributeSet set, int' attrs, IntStyle defAttr, int defStyleRes) Get style attribute information in this context of the theme. The ultimate TypedArray getStyledAttributes (int resid, int's attrs) Get style attribute information in this context of the theme. The ultimate TypedArray getStyledAttributes (int' attrs) Get style attribute information in this context of the theme. FileInputStream OpenFileInput (String name) Open a private file associated with the application package of this context for reading. Abstract FileOutputStream OpenFileOutput (String name, int mode) Open a private file associated with the application package of this context for writing. S'LiteDatabase openOrCreateDatabase (String name, int mode, S'LiteDatabase. CursorFactory factory, DatabaseErrorHandler errorHandler) Discover a new private database of S'LiteData associated with the context application package. S'LiteDatabase openOrCreateDatabase (String name, int mode, S'LiteDatabase factory. CursorFactory) opens a new private database of S'LiteData associated with the application package of this context. Abstract Drawable peekWallpaper () This method has been deprecated in API level 15. Instead, use WallpaperManager.peek. Invalid RegisterComponentCallbacks (ComponentCallbacks callbacks callbacks callbacks) Add a new ComponentCallbacks to the basic context application, which will be called at the same time as ComponentCallbacks practices and other components are called. Abstract Register of IntentionReceiver (BroadcastReceiver receiver, IntentFilter filter) Register BroadcastReceiver to run the main activity stream. Abstract Register of IntentReceiver (BroadcastReceiver receiver, IntentFilter filter, int flags) Registration to receive gear intentions, with the receiver optionally exposed to Instant Apps. Abstract Register of IntentionReceiver (Receiver BroadcastReceiver, IntentFilter Filter, Line BroadcastPermission, Handler Planner, Int Flags) Registration to receive intentional broadcasting, to run in context. Abstract Register of IntentReceiver (BroadcastReceiver Receiver, IntentFilter Filter, BroadcastPermission Line, Handler Planner) Registration to receive intention gear, to run in the context of the planner. abstract void removeStickyBroadcast (Intention Intention) This method has been deprecated in API level 21. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get the current value when needed. abstract emptiness of emptiness intention, userHandle user) This method has been deprecated in API level 21. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Abstract Void cancels UriPermission (Uri uri, int modeFlags) Remove all permissions to access a specific Uri content provider that was previously added with a UriPermission (String, Uri, int) grant or any other mechanism. Abstract Void cancelsUriPermission (String toPackage, Uri uri, int modeFlags) Remove permissions to access a specific Uri content provider that was previously added using grantUriPermission (String, Uri, int) for a specific target package. Abstract void sendBroadcast (Intention, Line receiverPermission) Broadcasting this intent to all interested BroadcastReceivers, allowing the optional required permission to be executed. abstract void sendBroadcast (intention intention) Broadcasting this intention to all interested BroadcastReceivers. abstract emptiness of sendingBroadcastAsUser (intention, user UserHandle) version sendBroadcast (android.content.Intent), which allows you to specify the user the broadcast will be sent. abstract void of sendingBroadcastAsUser (intention, UserHandle user, receiverPermission line) version sendBroadcast (android.content.Intent, java.lang.String), allowing you to specify the user's broadcast will be sent. Invalid sendBroadcastWithMultiplePermissions (Intention, Line ReceiverPermission) Broadcast this intent to all interested BroadcastReceivers, allowing an array of required permissions to be applied. Void to send OrderBroadcast (Intention, Line receiverPermission, String ReceiverAppOp, BroadcastReceiver resultReceiver, Handler, int initialCode, String initialData, initial BundleExtras) Version sendOrderedBroadcast (android.content.intent, java.lang.String, android.content.broadcastReceiver, android.content.com.au Abstract Void sendOrderedBroadcast (Intention intention, Line receiverPermission, BroadcastReceiver resultReceiver, Handler, Int. InitialCode, String initialData, initial BundleExtras) Version sendBroadcast (android.content.Intent), which allows you to receive data back from the broadcast. by supplying them by to allow the more preferred receivers to consume the broadcast before it is delivered to less preferred receivers. abstract emptiness to sendThe orderBroadcastAsUser (intention, (intention, User, String ReceiverPermission, BroadcastReceiver resultReceiver, Handler Planner, Int InitialCode, String InitialData, Bundle initialExtras) SendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.string. abstract void of sendingStickyBroadcast (intention intention) This method has been deprecated in API level 21. Sticky broadcasts should not be used. to report that something has changed, with a different mechanism for applications to get the current value when needed. Abstract void sending StickyBroadcastAsUser (intention, userHandle) This method has been hung in API level 21. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Abstract void dispatch OfStickyOrderedBroadcast (Intention Intention, BroadcastReceiver resultReceiver, Handler Planner, Int InitialCode, String initialData, Bundle initialExtras) This method has been deprecated in API level 21. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Abstract void sending StickyOrderedBroadcastAsUser (Intention, User Handle User User, BroadcastReceiver resultReceiver, Handler Planner, IntCode initialCode, String initialData, Bundle initialExtras) This method has been deprecated in API level 21. Sticky broadcasts should not be used. They provide no security (anyone can access them), no protection (anyone can change them), and many other problems. The recommended template is to use a non-sticky broadcast to report that something has changed, with a different mechanism for applications to get current value when needed. Set a basic theme for this context. setWallpaper (Bitmap bitmap) Abstract Void This method was uvemental in API 15. Instead, use WallpaperManager.set. This method requires time from the caller This method has been deprecated in API level 15. Instead, use WallpaperManager.set. This method requires time from the caller Manifest.permission.SET_WALLPAPER. abstract void startActivities (Intention) (Intention) A set of options) Launching several new events. abstract void startActivities (intentions) Same as startActivities (android.content.Intent, android.os.Bundle) without these options. abstract invalid startActivity (intention intention) Same as startActivity (android.content.Intent, android.os.Bundle) without specified options. abstract invalid startActivity (intention intention, kit options) Launch a new action. Abstract ComponentName StartForegroundService (Intention Service) is similar to startService (android.content.Intent), but with the implicit promise that the Service will call startForeground (int, android.app.Notification) as soon as it starts working. Abstract Boulean StartInstrumentation (ComponentName className, ProfileFile Line, Bundle Arguments) Start performing the instrumentalization class. Abstract Voids StartIntentSender (IntentionSender Intention, Intention fillInIntent, Int flagsMask, Int flagsValues, Int extraFlags) Same thing that startInmentSender (android.content.IntentSender, android.content.Intent, int, int, int, int. Int flagsValues, Int extraFlags, Bundle options) How to startActivity (android.content.Intent, android.os.Bundle), but taking IntentSender to begin with. unregisteredComponentCallbacks (ComponentCallbacks callbacks callbacks) Remove the ComponentCallbacks, which was previously registered with registerComponentCallbacks (android.content.ComponentCallbacks). abstract void of unregisteredReceiver (BroadcastReceiver receiver) Unregistered previously registered BroadcastReceiver. For a service previously associated with bindService (Intention, ServiceConnection, int) or related method, change the way the service is managed in relation to other processes. From the java.lang.Object Object class, the clone creates and returns a copy of this object. boolean (Object obj) indicates whether any other object is equal to this. invalid completion () Is called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. The final class of the getClass returns the time class of the subject. int hashCode () Returns the hash code value to the object. the final invalid to notify () will wake up one thread that is waiting on the monitor of this object. Line toString Returns the view of the object line. Final Expectation of Void (Long Time Out, Int Nanos) Causes the thread wait until another thread triggers a notify () method or notifyAll () method for that object, or any other thread interrupts the current thread, or a certain amount of real time has passed. The final expectation of emptiness (long time) triggers the wait for the current thread until another notification method () or notifyAll method has been triggered for that object, or a certain amount of time has passed. the final expectation of emptiness () causes the current thread to wait until another thread triggers the notification method () or the notifyAll method for that object. From the interface android.view.Window.Callback abstract drilling controlGenerationMotionEvent (MotionEvent event) Called to handle general traffic events. The abstract boolean dispatchKeyEvent (KeyEvent event) is designed to handle key events. the abstract boolean dispatchKeyShortcutEvent (KeyEvent event) is designed to handle a key label event. The Abstract Drilling Control RoomEvent (AccessibilityEvent) is designed to process the population of AccessibilityEvents. The abstract boolean dispatchTouchEvent (MotionEvent event) is designed to handle touchscreen events. The abstract boolean dispatchTrackballEvent (MotionEvent Event) is designed to handle trackball events. The abstract void on ActionMode Is called when the mode of action is complete. Abstract void on ActionModeStarted (ActionMode mode) Is called in running mode. The abstract void on The CaptionWindow is called when the window was attached to the window manager. The abstract void onContentChanged is called whenever the view of the screen content changes (due to a call to Window'setContentView (View, android.view.ViewGroup.LayoutParams) or Window'addContentView (View, android.viewGroup.LayoutParams).) abstract boolean onCreatePanelMenu (int featureId, menu menu) to initiate the contents of the menu for the 'featureId' panel. an abstract view ofCreatePanelView (int featureId) Instant view to display in the 'featureId' panel. The abstract void on DetachedFromWindow is called when the window was separated from the window manager. abstract boolean onMenuItemSelected (int featureId, MenuItem) is called when selecting a panel menu item by the user. abstract boolean onMenuOpened (int featureId, menu menu) is called when the panel menu is opened by the user. The abstract void on ThePanelClosed (int featureId, menu menu) is triggered when the panel closes. The default void onPointerCaptureChanged (boolean hasCapture) is called when the pointer captures or turns off for the current window. abstract boolean onPreparePanel (int featureId, View view, menu menu) Prepare the panel for display. default void onProvideKeyboardShortcuts (List Menu menu, Int deviceId) Is called when keyboard shortcuts are requested for current keyboardShortcutGroup windows. abstract boolean onSearchRequested () Is called when a user signals a desire to start a zlt;/KeyboardShortcutGroup abstract boolean onSearchRequested (SearchEvent searchEvent) Is called when the user signals a desire to start a search. abstract void onWindowAttributesChanged (WindowManager.LayoutParams attrs) This is called whenever current window attributes change. Abstract void onWindowFocusChanged (boolean hasFocus) This hook is called whenever the focus of the window changes. Abstract ActionMode onWindowStartingActionMode (ActionMode.Callback callback) Is called in running mode for this window. Abstract ActionMode onWindowStartingActionMode (ActionMode.Callback callback, int type) is called in running mode for this window. From the interface android.view.KeyEvent.Callback abstract boolean onKeyDown (int keyCode, KeyEvent event) is called when the event with the return key occurred. abstract boolean onKeyLongPress (int keyCode, KeyEvent event) is called when a long press occurs. abstract boolean onKeyMultiple (int keyCode, int count, KeyEvent event) Is called when the user's interaction with analog control, such as a trackball throw, generates simulated down/up events for the same key several times in a row. abstract boolean onKeyUp (int keyCode, KeyEvent event) is triggered when the key event comes up. Public static final int DEFAULT_KEYS_SHORTCUT used with the DefaultKeyMode (int) set to perform menu shortcuts when handling default keys. That is, the user does not need to hold the menu key to perform menu shortcuts. See also: Permanent Value: 2 (0x00000002) public static finale int RESULT_CANCELED the result of the activities of the Standard: Operation Cancelled. Permanent value: 0 (0x00000000) public static RESULT_FIRST_USER int to start user activity results. Permanent value: 1 (0x00000001) public static final int RESULT_OK standard performance result: the operation was successful. Permanent value: -1 (0xffffffff) Fields protected static final int FOCUSED_STATE_SET Public Designers Public Methods Public Invalid addContentView (View View, ViewGroup.LayoutParams params) Add additional content representation to the activity. Added after any existing activity - existing views are not removed. View settings: The content you want to display. params ViewGroup.LayoutParams: Layout settings for presentation. The program closes the last open context menu if it shows. public emptiness closeOptionsMenu () Progammatically closes the options menu. If the options menu is already closed, this method does nothing. Public PendingIntent createPendingResult (int requestCode, Intent Data, int flags) Create a new object that you can pass on to others for their use to send the results back to onActivityResult (int, Int, Intention) callback. The created object will be either one shot (becomes invalid after sending the result back), or multiple (allows you to send through it any number of results). results). requestCode int: A private query code for the sender that will be linked to the result data when it is returned. The sender cannot change this value, allowing you to determine incoming results. Data Intention: Default data for delivery as a result that can be changed by the sender. This value cannot be zero. Flags int: may be PendingIntent-FLAG_ONE_SHOT, PendingIntent-FLAG_NO_CREATE, PendingIntent-FLAG_CANCEL_CURRENT, PendingIntent-FLAG_UPDATE_CURRENT, or any of the flags supported by Intent'fillIn to control which uncertain parts of the intent that may be delivered when the actual dispatch occurs. The value is 0 or a combination of PendingIntent.FLAG_ONE_SHOT, PendingIntent.FLAG_NO_CREATE, PendingIntent.FLAG_CANCEL_CURRENT, PendingIntent.FLAG_UPDATE_CURRENT, PendingIntent.FLAG_IMMUTABLE, Intent.FILL_IN_ACTION, Intent.FILL_IN_DATA, Intent.FILL_IN_CATEGORIES, Intent.FILL_IN_COMPONENT, Intent.FILL_IN_PACKAGE, Intent.FILL_IN_SOURCE_BOUNDS, Intent.FILL_IN_SELECTOR, and Intent.FILL_IN_CLIP_DATA public final space. The public drilling control room MotionEvent ev is designed to handle general traffic events. You can override this to intercept all common traffic events before they are sent to the window. Be sure to call this implementation a common movement event that should be handled normally. Ev MotionEvent Options: A Common Motion Event. Returns boolean boolean Return is true if this event was consumed. The public boolean dispatchKeyEvent (KeyEvent event) is designed to handle key events. You can override this to intercept all the key events before they are sent to the box. Be sure to call this implementation key events that need to be handled as normal. KeyEvent Event Options: Key Event. Returns boolean boolean Return is true if this event was consumed. The keyShortcutEvent public boolean dispatch is designed to handle a key label event. You can override this to intercept all key label events before they are sent to the box. Don't forget to call this implementation key label events that need to be handled normally. KeyEvent Event Options: Key Label Event. Returns boo Truelean if this event was used. Public Drilling DispatchPopabilityEvent (AccessibilityEvent) is designed to handle the population of AccessibilityEvents. AccessibilityEvent Event Options: Event. Returns boolean boolean Return is true if the population event was completed. Public boolean dispatchTouchEvent (MotionEvent ev) is designed to handle sensory events You can override this to intercept all touchscreen events before they are sent to the window. Be sure to call this implementation a touchscreen event that should be handled normally. Ev MotionEvent Options: Touch Screen Event. Returns boolean boolean return true if it's Consumed. Public boolean dispatchTrackballEvent (MotionEvent ev) Called to handle trackball events. You can override this to intercept all trackball events before they are sent to the box. Be sure to name this implementation for trackball events that need to be handled normally. Ev MotionEvent Options: Trackball Event. Returns boolean boolean Return is true if this event was consumed. public dump of voids (Line Prefix, FileDescriptor fd, PrintWriter writer, String'args) Print activity state in a given stream. This is caused if you run the adb shell dumpsys activity_component_name.' String Prefix Options: A Wish-wish prefix on each line of output. This value cannot be zero. fd FileDescriptor: The unprocessor of files to which the landfill is sent. This value can be zero. Writer PrintWriter: PrintWriter, to which you have to reset your state. It will be closed to you after you return. This value cannot be zero. args String: additional arguments to request a landfill. This value can be zero. public T findViewById (int id) finds a view that has been identified by the android:id XML attribute that has been processed in onCreate (Bundle). Note: In most cases - depending on compiler support - the view is automatically discarded as the target class. If the target class type is not limited, you may need a clear throw. ID Int Options: ID to search For Returns T View with this ID, if found, or zero otherwise See also: View.findViewById (int)requireViewById (int) public finish void () Call on this issue when your activity is done and should be closed. ActivityResult extends back to those who launched you through onActivityResult. The public void of the finishActivity (int requestCode) Force completes another action that you previously started with startActivityForResult (Intention, Int). Code int query options: The action request code you gave to start ActivityForResult. If several actions are initiated with this query code, all of them will be completed. Added to the API level 1 Deprecated in the API level 30 public void ActivityFromChild (Action of the Child, int requestCode) This method has been deprecated in API level 30. Instead, use finishActivity (int). This is caused when the activity of the child of this one causes its finishActivity. Child activity options: activity, making a call. This value cannot be zero. requestCode int: Request code that was used to run the action. Public void finishAffinity () Complete this activity, as well as all the actions directly under it in the current task that same affinity. This is usually used when an application can be run on a different task (e.g. from ACTION_VIEW type of content it understands), and the user has used up navigation to move from the current task to their own task. In this case, if the user is activity_component_name/activity_component_name in any other second application actions, they all must be removed from the original task as part of the switch task. Please note that this finish does not allow you to deliver results to the previous action, and an exception will be cast if you try to do so. Public void finishAndRemoveTask () Call on this issue when your activity is completed and should be closed, and the task should be completely removed as part of the completion of the root activity of the task. Added to the API level 1 Deprecated in api level 30 public void finishFromChild (Child Action) This method has been deprecated in API level 30. Use the finish () instead. This is called when a child's activity calls it a trim method. Implementing by default simply causes the completion () of this action (parent) ending the entire group. Child activity options: activity, making a call. Public ActionBar getActionBar () Get a link to ActionBar of this activity. Returns ActionBar ActionBar action, or zero if it doesn't have one. Public ComponentName getCallingActivity () Return the name of the action that caused this action. This is the one to whom the data will be sent to setResult. You can use this information to verify that the recipient is allowed to receive the data. Note: if the call activity does not expect a result (i.e. it does not use the startActivityForResult (Intention, Int) form, which includes the request code), the call package will be zero. Returns ComponentName ComponentAin action that will get your response, or zero if not. the public line getCallingPackage () Return the name of the package that caused this action. This is the one to whom the data will be sent to setResult. You can use this information to verify that the recipient is allowed to receive the data. Note: if the call activity does not expect a result (i.e. it does not use the startActivityForResult (Intention, Int) form, which includes the request code), the call package will be zero. Note: up to Build.VERSION_CODES. JELLY_BEAN_MR2, the result of this method was unstable. If the process of hosting a call package no longer works, it will return zero instead of the proper package name. You can use getCallingActivity () and get the package name out of it instead. Returns the action package line that will receive your response, or zero if not. If this activity breaks down because it can't handle the configuration option changeable (and thus its onConfigurationChanged method (android.content.res.Configuration) is not called, then you can use this method to detect a set of changes that occurred during the destruction process. Please note that there is no guarantee that they will be accurate (other changes could Anytime), so you have to Use this as an optimization tip. Returns int Returns a few field configuration settings that change as defined by the configuration class. Returns ComponentName Returns the full name of the component for this public line activity getLocalClassName () Returns the class name for this action with the prefix package removed. This is the default name used to read and write settings. Returns the line Local class name. This value cannot be zero. Public int getMaxNumPictureInPictureActions () Return the number of actions that will be displayed in the user interface with the picture in the picture when the user interacts with the activity currently in picture mode. This number can change if the global configuration changes (for example, if the device is connected to an external display), but there will always be more than three. Public Intent getParentActivityIntent () Get an intention that will trigger the explicit target action specified by the logical parent of this activity. The logical parent is named in the app's manifesto as parentActivityName. Activity subclasses can override this method to change the Intent returned by super.getParentActivityIntent() or fully implement another parental intent extraction mechanism. Returns the intention of a new intention focused on a particular parent of this activity, or zero if there is no valid parent. Uri getReferer () Return of information about who started this activity. If the Intention Startup contains Intent.EXTRA_REFERRER that will be returned as is; otherwise, if it is known, the URI_ANDROID_APP_SCHEME the URI reference containing the name of the package that started the Intention will be returned. This can lead to zero if the reference can't be identified - it's not clearly specified, and it's not known which application package was involved. If called during processing onNewIntent (Intention), this feature returns the referee who introduced that new intention to the activity. Otherwise, it always returns the referee of the original intent. Please note that this is not a security feature - you can't trust the information of the referral, the apps can fake it. public int getRequestedOrientation () Return the current requested orientation of the action. This will be either the orientation requested in the component manifesto or the last requested orientation provided for the requestedOrientation (int) set. Returns int Returns the Constant Orientation Used in ActivityInfo-screenOrientation. Meaning: android.content.pm.ActivityInfo.SCREEN_ORIENTATION_UNSET, ActivityInfo.SCREEN_ORIENTATION_UNSPECIFIED, ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_USER, ActivityInfo.SCREEN_ORIENTATION_SENSOR, ActivityInfo.SCREEN_ORIENTATION_NOSENSOR, ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_REVERSE_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_REVERSE_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR, ActivityInfo.SCREEN_ORIENTATION_USER_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_USER_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_FULL_USER, or ActivityInfo.SCREEN_ORIENTATION_LOCKED public Object getSystemService (String name) return the pen to the system-level service by name. The class of the returned object varies depending on the name requested. Names are currently available: WINDOW_SERVICE (window) Top-level window manager in which you can place custom windows. The returned object is WindowManager. Should only be derived from visual context, such as Activity or Context created with createWindowContext (int, android.os.Bundle), which are adjusted to the configuration and visual boundaries of the area on the screen. LAYOUT_INFLATER_SERVICE (layout_inflater) LayoutInflater to inflate layout resources in this context. Should only be derived from visual context, such as Activity or Context created with createWindowContext (int, android.os.Bundle), which are adjusted to the configuration and visual boundaries of the area on the screen. activity ActivityManager ACTIVITY_SERVICE to interact with the global state of the system's activity. WALLPAPER_SERVICE (wallpaper) WallpaperService to access wallpaper in this context. Should only be derived from visual context, such as Activity or Context created with createWindowContext (int, android.os.Bundle), which are adjusted to the configuration and visual boundaries of the area on the screen. POWER_SERVICE (power) PowerManager to control power management. ALARM_SERVICE (anxiety) AlarmManager to get intentions during your choice. NOTIFICATION_SERVICE (notice) NoticeMenager to inform the user about background events. KEYGUARD_SERVICE (key-guard) KeyguardManager to control the keyboard. LOCATION_SERVICE (location) LocationMan for location control (such as GPS) updates. SEARCH_SERVICE (search) SearchManager to process the search. VIBRATOR_SERVICE (vibrator) vibrator to interact with vibrator equipment. CONNECTIVITY_SERVICE (connection) by ConnectivityManager to manage network connections. IPSEC_SERVICE (ipsec) IpSecManager to control IPSec on outlets and networks. WIFI_SERVICE Wi-Fi To control Wi-Fi. When released before NYC, it should only be obtained from the context of the application, not from any other derivative context, to avoid memory leaks during the call process. WIFI_AWARE_SERVICE WifiAwareManager to control Wi-Fi Aware detection and connectivity. WIFI_P2P_SERVICE (wifip2p) для уравлениит Wi-Fi Wi-Fi Link. INPUT_METHOD_SERVICE (input_method) InputMethodManager to manage input methods. UI_MODE_SERVICE (uimode) UiModeManager to manage user interface modes. DOWNLOAD_SERVICE (download) DownloadManager to request BATTERY_SERVICE (battery) Battery JOB_SCHEDULER_SERVICE (taskmanager) JobScheduler to manage planned NETWORK_STATS_SERVICE tasks (netstats) NetworkStatsManager to request network usage statistics. HARDWARE_PROPERTIES_SERVICE (hardware_properties) Manager Hardware offer to access hardware properties. Note: System services obtained through this API may be closely related to the context in which they are received. Typically, do not divide service objects between different contexts (activities, applications, services, vendors, etc.) Note: Instant applications for which PackageManager-isInstantApp () returns correctly, do not have access to the following system services: DEVICE_POLICY_SERVICE, FINGERPRINT_SERVICE, KEYGUARD_SERVICE, SHORTCUT_SERVICE, USB_SERVICE, WALLPAPER_SERVICE, WIFI_P2P_SERVICE, WIFI_SERVICE, WIFI_AWARE_SERVICE. For these services, this method will return zero. Normally, if you're working as an instant application, you should always check whether the result of this method is zero. Note: When implementing this method, keep in mind that new services can be added on new Android releases, so if you're looking for only explicit names mentioned above, make sure to return zero when you don't recognize the name - if you throw a RuntimeException exception instead, you app can break down on new Android releases. Line Name Options: The Value of Context.POWER_SERVICE, Context.WINDOW_SERVICE, Context.LAYOUT_INFLATER_SERVICE, Context.ACCOUNT_SERVICE, Context.ACTIVITY_SERVICE, Context.ALARM_SERVICE, Context.NOTIFICATION_SERVICE, Context.ACCESSIBILITY_SERVICE, Context.CAPTIONING_SERVICE, Context.KEYGUARD_SERVICE, Context.LOCATION_SERVICE, Context.SEARCH_SERVICE, Context.SENSOR_SERVICE, android.content.Context.SENSOR_PRIVACY_SERVICE, Context.STORAGE_SERVICE, Context.STORAGE_STATS_SERVICE, Context.WALLPAPER_SERVICE, android.content.Context.TIME_ZONE_RULES_MANAGER_SERVICE, Context.VIBRATOR_SERVICE, Context.CONNECTIVITY_SERVICE, Context.IPSEC_SERVICE, Context.VPN_MANAGEMENT_SERVICE, android.content.Context.TEST_NETWORK_SERVICE Context.NETWORK_STATS_SERVICE Context.NETWORK_STATS_SERVICE, Context.NETWORK_STATS_SERVICE, Context.NETWORK_STATS_SERVICE Context.NETWORK_STATS_SERVICE , Context.WIFI_SERVICE, Context.WIFI_AWARE_SERVICE, Context.WIFI_P2P_SERVICE, android.content.Context.WIFI_SCANNING_SERVICE, Context.WIFI_RTT_RANGING_SERVICE, Context.NSD_SERVICE, Context.AUDIO_SERVICE, android.content.Context.AUTH_SERVICE, Context.FINGERPRINT_SERVICE, Context.BIOMETRIC_SERVICE, Context.MEDIA_ROUTER_SERVICE, Context.TELEPHONY_SERVICE, Context.TELEPHONY_SUBSCRIPTION_SERVICE, Context.EUICC_SERVICE, Context.TELECOM_SERVICE Context.TELECOM_SERVICE Context.INPUT_METHOD_SERVICE, Context.TEXT_SERVICES_MANAGER_SERVICE, Context.TEXT_CLASSIFICATION_SERVICE, Context.APPWIDGET_SERVICE, android.content.Context.ROLLBACK_SERVICE, Context.DROPBOX_SERVICE, Context.DEVICE_POLICY_SERVICE, Context.UI_MODE_SERVICE, Context.DOWNLOAD_SERVICE, Context.NFC_SERVICE Context.BLUETOOTH_SERVICE, Context.USB_SERVICE, Context.LAUNCHER_APPS_SERVICE, Context.INPUT_SERVICE, Context.DISPLAY_SERVICE, Context.USER_SERVICE, Context.RESTRICTIONS_SERVICE, Context.APP_OPS_SERVICE, Context.ROLE_SERVICE, Context.CAMERA_SERVICE, Context.PRINT_SERVICE, Context.CONSUMER_IR_SERVICE, Context.TV_INPUT_SERVICE, Context.USAGE_STATS_SERVICE, Context.MEDIA_SESSION_SERVICE, Context.BATTERY_SERVICE, Context.JOB_SCHEDULER_SERVICE, Context.MEDIA_PROJECTION_SERVICE, Context.MIDI_SERVICE, android.content.Context.RADIO_SERVICE, Context.HARDWARE_PROPERTIES_SERVICE, Context.SHORTCUT_SERVICE, Context.SYSTEM_HEALTH_SERVICE, Context.COMPANION_DEVICE_SERVICE, Context.CROSS_PROFILE_APPS_SERVICE, android.content.Context.PERMISSION_SERVICE or android.content.Context.LIGHTS_SERVICE. Returns a Service object or zero if the name doesn't exist. Public int getTaskId () Return of the task ID in which this activity is located. This identifier will remain the same for the duration of the term. Returns the Int task ID, an opaque integer. The public final int getTitleColor () public final int getVolumeControlStream () receives the proposed audio stream, the volume of which must be changed by the hardware control of the volume. Returns int Proposed type of audio stream, the volume of which must be changed by the hardware volume control. See also: setVolumeControlStream (int) public window getWindow () Get the current window for activities. This can be used to directly access parts of the window API that are not available through Activity/Screen. Returns the Window Window Current Window, or zero if the action is not visual. Public boolean isActivityTransitionRunning () Returns to see if there are any activity transitions currently working on this activity. The true return value may indicate that the entry or exit transition is working, including whether the action background is animated as part of that transition. Returns boolean is true if the transition is currently running on this activity, false otherwise. Public boolean isChangingConfigurations () Check to see if this activity is in the process of destruction in order to be recreated with a new configuration. This is often used in onStop to determine whether the state should be cleaned or be transferred to the next instance of activity through onRetainNonConfigurationInstance. Returns boolean If the action is ripped off in order to be recreated with Configuration, returns correctly; still coming back false. public final boolean isChild () Is this activity activity inside another activity? Public boolean isDestroyed () Returns true if the final onDestroy () call was made on operation, so this copy is now dead. Public boolean isFinishing () Check whether this activity is in the process of finishing, either because you called the finish () on it or someone else asked him to finish. This is often used in onPause () to determine whether the activity is just a pause or a complete finish. Returns boolean If the activity ends, returns correctly; still coming back false. public boolean isInMultiWindowMode () Returns correctly if the action is currently in multi-window mode. Returns boo Truelean if the action is in multi-window mode. See also: R.attr.resizeableActivity public boolean isInPictureInPictureMode () Returns true if the activity is currently in image mode in the picture. Returns boo Truelean if the action is in picture-in-picture mode. See also: R.attr.supportsPictureInPicture public boolean isLocalVoiceInteractionSupported () Queries, whether the voice service currently supports the return of voice interaction for use in action. It's really only for the duration of the action. Returns boolean whether the current voice service supports the local interaction of the voice of the public boolean isTaskRoot () Back whether this activity is the root of the task. Root is the first action in the task. Returns boolean True if it's root activity, otherwise false. Public boolean isVoiceInteraction () Check to see if this action works as part of a voice interaction with the user. If this is true, it should perform its interaction with the user through VoiceInteractor returns getVoiceInteractor(). public boolean isVoiceInteractionRoot () As isVoiceInteraction (), but only returns true if it is also the root of voice interaction. That is, it returns correctly if this action was directly initiated by the voice interaction service as the beginning of voice interaction. Otherwise, for example, if it was running another activity while under voice interaction, returns false. If the launchMode is one, it forces the activity to start in a new task separate from the running. Thus, there is no longer a relationship between them, and isVoiceInteractionRoot () return is false in this case. Added to API level 1 Deprecated in API level 15 public final course guided by Kerry (Uri Uri, String projection, line selection, String selectionArgs, String sortOrder) This method has been highlighted in API level 15. Instead, use CursorLoader. Wrap around ContentResolver-query (android.net.Uri , String, String, String, string), which gives the resulting Cursor a startManagingCursor (Cursor) challenge, so the activity will manage your life cycle for you. If you're focused on Build.VERSION_CODES. HONEYCOMB or later, consider instead using LoaderManager LoaderManager available via getLoaderManager. Warning: Don't call Cursor-close to the cursor obtained with this method because the action will do it for you at the appropriate time. However, if you call stopManagingCursor (Cursor) on a cursor from a managed query, the system will not automatically close the cursor and, in this case, you should call Cursor'close. Uri Uri Options: URI Content Provider for Query. projection line: List of columns to return. Line of choice: SDL clause WHERE. selectionArgs String: Arguments to choose, if any? Returns Tosor Kursor, which was returned by request. Public boolean moveTaskToBack (boolean nonRoot) Will move the task containing this activity to the back of the activity stack. The order of action within the task remains the same. NonRoot boolean options: If false, this only works if the action is the root of the task; If this is true, it will work for any activity in the task. Returns boolean If the task has been moved (or it was already on the back) the truth returns and then falsely. Public boolean navigateUpTo (Intent upIntent) Move from this activity to the activity specified by upIntent, ending this activity in the process. If the activity specified by upIntent already exists in the task history, this action and all other actions specified in the history stack will be completed. If this action does not appear in the story stack, it completes each activity in the task until the root activity of the task is achieved, leading to behavior in the home application. This can be useful in applications with a complex navigation hierarchy, where action can be achieved through non-canonical parenthood. This method should be used to navigate from the same task as the destination. If up navigation should cross tasks in some cases, see shouldUpRecreateTask (android.content.Intent). Options upIntent Intention: The intention representing the target destination for pre-navigation returns boolean is true if before navigation successfully reached the activity specified upIntent and upIntent was delivered to it. false if a copy of the said activity cannot be found and this activity was simply completed normally. Added to API level 16 Deprecated in API level 30 public boolean navigateUpToFromChild (Child Activity, Intention upIntent) This method has been deprecated in API level 30. Instead, use navigateUpTo (android.content.Intent). This is called when child activity of this calls it the navigateUpTo (Intention) method. Implementing by default simply triggers navigateUpTo (upIntent) on this action (parent). Child activity options: activity, making a call. upIntent the intention representing the target destination for pre-navigation returns boolean is true if before navigation has successfully reached specified upIntent and upIntent was delivered to it. false if a copy of the said activity cannot be found and this activity was simply completed normally. Public void on ActionModeFinished (ActionMode mode) notifies the activity that the mode has completed. Activity subclasses that prevail over this method should be called the implementation of the superclass. If you override this method, you should call before the superclass is implemented. ActionMode options: an action mode that has just ended. Public void in ActionModeStarted (ActionMode mode) notifies you of the running mode. Activity subclasses that prevail over this method should be called the implementation of the superclass. If you override this method, you should call before the superclass is implemented. ActionMode Mode Options: A New Mode of Action. The public void on TheActivityReenter (int resultCode, Intent Data) is triggered when the action you've started with the activity transition exposes this action through a return activity transition, giving you the resultcode and any additional data from it. This method will only be called if the action sets the result code, which is not RESULT_CANCELED, and supports activity transitions from window-FEATURE_ACTIVITY_TRANSITIONS. The purpose of this feature is to allow the action to send a hint about its condition so that this basic action can be prepared for disclosure. Calling to this method does not guarantee that the so-called Action has or will be coming out soon. This only indicates that he will expose this activity window and he has some data to pass to prepare it. ResultCode int options: an integrative result code returned by the child's activity through their setResult.) Data Intention: An intention that can return the result data to the subscriber (various data may be attached to additional intentions). Public emptiness onBackPressed () Is called when the action detected the user's push on the back key. Implementing the default just completes the current action, but you can override it to do whatever you want. public void onConfigurationChanged (Configuration newConfig) caused by the system when the device configuration changes while working. Please note that this will only be called if you choose the configurations you would like to handle with the R.attr.configChanges attribute in your manifest. If there is any configuration change that is not selected to message this attribute, instead of reporting it, the system will stop and restart the action (so that it will be launched with a new configuration). As this feature was called, the Resource object will be updated to return resource values new configuration. New Configuration OptionsConfig: A new device configuration. This value cannot be zero. public boolean onContextItemSelected (item MenuItem) This hook is called whenever in the context of the menu is chosen. The default implementation simply returns false to have normal processing (call a Runnable item or send a message to its handler as needed). You can use this method for any items you would like to do without these other objects. Use MenuItem-getMenuInfo for more information from View, which added this menu item. Derivative classes should be called to the base class to process the default menu. MenuItem option options: Selected element of the context menu. This value cannot be zero. Returns boolean boolean Return false to allow normal processing of the context menu to continue, the truth is to consume it here. public void onContextMenuClosed (menu menu) This hook is called whenever the contextual menu closes (either by user, cancelling the menu with the Back/Menu button, or when selecting an item). Menu Menu: Contextual Menu That Closes. This value cannot be zero. Public void onCreateContextMenu (ContextMenu, View v, ContextMenu.ContextMenuInfo menuInfo) Is called when the contextual menu for the presentation is about to be shown. Unlike onCreateOptionsMenu (android.view.Menu), this will be called every time the contextual menu is shown and must be filled for submission (or an item inside the view for adapterView subclasses, this can be found in menuInfo). Use onContextItemSelected (android.view.MenuItem) to find out when the item was selected. It is not safe to hold the context menu once this method is returned. ContextMenu Menu Options: Context Menu, Which Is Built by v View: The View for which the contextual menu Info ContextMenu.ContextMenuInfo is built: Additional information about the item for which the contextual menu should be shown. This information will vary depending on class v. public CharSequence onCreateDescription () Create a new description for this activity. This method is called before the action stops and can, if desired, return to the user some text description of its current state. The default implementation returns zero, resulting in the inheritance of the description from the previous action. If all actions are null and void, the top activity tag will usually be used as a description. Returns CharSequence Description of what the user is doing. It should be short and sweet (just a few words). See also: onSaveInstanceState (Bundle)onStop () public void onCreateNavigateUpTaskStack (TaskStackBuilder builder) Identify a synthetic stack of tasks that will be generated during up navigation from another task. Implementing this method by default adds the chain of this activity, as stated in the manifesto, to the Supplied by TaskStackBuilder. Apps can override this method to build the task stack differently. it's different. The method will be called the default implementation onNavigateUp if shouldUpRecreateTask (android.content.Intent) returns correctly when delivered with the intention of returning getParentActivityIntent. Apps that want to provide additional intent settings to the parent stack determined by the manifest should be redefined byPrepareNavigateUpTaskStack (android.app.TaskStackBuilder). Builder Options TaskStackBuilder: Empty TaskStackBuilder - The app should add intent, presenting the desired task of stacking the public boolean onCreateOptionsMenu (Menu Menu) to initiate the contents of the standard menu of activity options. You have to put your menu items on the menu. This is only called once, the first time you see the options menu. To update the menu every time it is displayed, see PrepareOptionsMenu (Menu). The default implementation fills the menu with the standard menu elements of the system. They are placed in the Menu'CATEGORY_SYSTEM group to be properly ordered with menu items defined by the app. Derivative classes should always run until the basic implementation. You can safely hold onto the menu (and any elements created from it) by making changes at will until the next time onCreateOptionsMenu is called. When you add items to the menu, you can implement the Activity onOptionsItemSelected method to handle them there. Menu Options: Menu options in which you place your items. Returns boolean You have to go back true for the menu to be displayed; If you come back false, it will not be shown. See also: onPrepareOptionsMenu (Menu)onOptionsItemSelected (MenuItem) public presentation onCreatePanelView (int featureId) Default Window.callback.onCreatePanelView (int) for action. This just returns zero so that all under-window panels will have the default menu behavior. FeatureId int options: Which panel is created. Returns the view view to the top-level view for the seat in the panel. Added to API level 1 Deprecated in API level 28 public boolean onCreateThumbnail (Bitmap outBitmap, canvas canvas) This method has been unified in API level 28. The method does nothing and will be removed in the future. The outBitmap Bitmap canvas canvas canvas of public viewing onCreateView (String name, context context, AttributeSet attrs) Standard implementation layoutInflater.factory.onCreateView (Line, Context, AttributeSet) is used when inflating with LayoutInflater returns Context.getSystemService (class). This implementation does nothing for the pre-Build.VERSION_CODES. HONEYCOMB APPS. New apps should be used onCreateView (android.view.View, java.lang.String, android.content.Context, android.util.AttributeSet). Line Name Options: This value can't Zero. Context Context: This value cannot be invalidated. attributeSet: This value can't be zero. Returns View This value may be invalid. public emptiness emptiness () Actions cannot draw during the period when their windows are animated. In order to know when it is safe to start drawing they can override this method, which will be called when the animation input is complete. Public boolean onGenericMotionEvent (MotionEvent event) Is called when the general event of the movement has not been processed by any of the representations within the event. Common motion events describe the movements of the joystick, mouse hovers, track pads touches, scrolling wheels and other input events. MotionEvent-getSource () motion event determines the class input that was received. Implementation of this method should examine the bits in the source before handling the event. The following example of the code shows how this is done. Common traffic events with the original class InputDevice.SOURCE_CLASS_POINTER which are delivered to the view under the pointer. All other common traffic events are delivered to a focused view. You can find in the example how to handle this event in view'onGenericMotionEvent (MotionEvent). MotionEvent Event Options: A common motion event is processed. Returns boolean Return is true if you consumed an event false if you don't. The default implementation always returns false. Public Void onGetDirectActions (CancelSignary CancellationSenary, qlt; qlt; directAction) returns a list of direct actions supported by the app. You should return a list of activities that can be performed in the current context that is in the current state of the application. If you're taking action that can be done with app changes, you should report it by calling VoiceInteractor-DirectActionsChanged. To get voice interacting you need to call getVoiceInteractor, which will return, not zero only if there is a current voice interaction session. You also find when voice interacting is no longer valid because the voice interaction session that supports is over by calling VoiceInteractor'registerOnDestroyedCallback (Artist, Runnable). This method will only be called after onStart () is being called and before onStop () is being called. You must pass the call back now supported by direct actions that may not be invalid or contain zero items. You should return the action list as soon as possible to ensure that the consumer, such as the assistant, will be as responsive as possible, which will improve the user experience of your app. Cancellation of the optionsSy cancellationSenya: a signal about the cancellation of the operation. This value cannot be zero. Consumer CallBack: Call back to send a list of actions. The list of actions cannot contain zero items. You to call it on any topic. This value cannot be zero. A public boolean onKeyDown (int keyCode, KeyEvent event) is called when the key has been pressed down and not handled by any of the submissions inside the activity. For example, key presses while cursor/DirectAction TextView will not initiate an event (unless it's navigating another object) because TextView handles its own key press. If a focused view didn't want this event, this method is called. The default implementation takes care of KeyEvent-KEYCODE_BACK, causing onBackPressed, although the behavior changes depending on the application compatibility mode: for Build.VERSION_CODES. ECLAIR or later apps, it will set up a send to call KeyUp (int, KeyEvent), where the action will run; For earlier applications, it will perform the action immediately in on-down as these versions of the platform behaved. Other additional default key processing can be done if setDefaultKeyMode (int) is configured. KeyCode int options: Event.getKeyCode KeyEvent: Description of a key event. Returns boolean Return is true to prevent this event from spreading further, or false to indicate that you have not handled this event and it should continue to spread. See also: onKeyUp (int, KeyEvent)KeyEvent public boolean onKeyLongPress (int keyCode, KeyEvent event) default implementation KeyEvent.Callback-onKeyLongPress (int, KeyEvent): always returns false (not coping with the event). To get a callback, you must return from onKeyDown for the current flow of events. KeyCode int options: Event.getKeyCode KeyEvent: Description of a key event. Returns boolean If you coped with the event, come back true. If you want the event to be handled by the next recipient, come back falsely. See also: KeyEvent.Callback.onKeyLongPress (int, KeyEvent) public boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) By default implementation KeyEvent.Callback-onKeyMultiple (int, int, KeyEvent): always returns false (not coping with the event). KeyCode int options: Event.getKeyCode repeatCount int: Number of couples returned to event.getRepeatCount . KeyEvent: Description of a key event. Returns boolean If you coped with the event, come back true. If you want the event to be handled by the next recipient, come back falsely. a public boolean onKeyShortcut (int keyCode, KeyEvent event) is called when a key label event is not handled by any of the views in the Action. Redefine this method to implement global key labels for Action. Key labels can also be implemented by customizing menu items in MenuItem'setShortcut (char,

char). KeyCode int options: Event.getKeyCode KeyEvent: Description of a key event. Returns boolean True, if the key label has been processed. public boolean onKeyUp (int keyCode, KeyEvent event) is called when the key has been released and processed by none of the representations within the activity. So, for example, the key press while the cursor is inside TextView will not cause an event (unless it is navigating to another other because TextView handles its own key press. The default implementation handles KEYCODE_BACK to stop the action and return. KeyCode int options: Event.getKeyCode KeyEvent: Description of a key event. Returns boolean Return is true to prevent this event from spreading further, or false to indicate that you have not handled this event and it should continue to spread. See also: onKeyDown (int, KeyEvent)KeyEvent Public Void onLocalVoiceInteractionStopped () Callback to indicate that local voice interaction has ceased either because it was asked through a call to stopLocalVoiceInteraction () or because it was canceled by the user. The previously acquired VoiceInteractor is no longer valid after that. This is called when a common system is running low in memory, and actively working processes should trim their memory usage. Although the exact point at which this will be called is not defined, it is usually when the entire background process has been killed. That is, before we reach the point of killing the hosting processes and the foreground of the user interface that we would avoid killing. You have to implement this method to release any caches or other unnecessary resources that you may keep on. The system will collect garbage for you after returning from this method. Preferably, you should implement ComponentCallbacks2'onTrimMemory from ComponentCallbacks2 to gradually unload resources based on different levels of memory requirements. This API is available for the API level 14 and above, so you should only use this on theLowMemory method as a backup for older versions that can be treated the same as ComponentCallbacks2'onTrimMemory with ComponentCallbacks2'TRIM_MEMORY_COMPLETE level. public boolean onMenuItemSelected (int featureId, menuItem) Default implementation window.Callback.onMenuItemSelected (int, MenuItem) for events. This requires a new onOptionsItemSelected (MenuItem) method for the Window.FEATURE_OPTIONS_PANEL panel, so activity subclasses don't have to deal with function codes. FeatureId int options: The panel in which the menu is located. MenuItem: This value cannot be zero. Returns boolean boolean Return is true to finish processing the selection, or false to perform normal menu processing (call it Runnable or send a message for its target Handler). public boolean onMenuOpened (int featureId, menu menu) is called when the panel menu is opened by the user. This can also be caused when the menu changes from one type to another (for example, from the icon menu to the extended menu). FeatureId int options: Panel, menu The menu is on. Menu: This value cannot be zero. Returns boolean Implementation by default returns correctly. public void onMultiWindowModeChanged (boolean isInMultiWindowMode, Configuration Called the system, when the action changes from full-screen mode to multi-window and visa-versa mode. This method provides the same configuration that will be sent in the next call onConfigurationChanged (android.content.res.Configuration) after the action enters this mode. InMultiWindowMode boolean options: True if the action is in multi-window mode. New ConfigurationConfig: A new action configuration with a state. See also: R.attr.resizeableActivity public boolean onNavigateUp () This method is called whenever the user chooses to navigate up the hierarchy of the application activity from the action bar. If the ActivityName parent attribute has been listed in the manifest for this action or the action is a pseudonym to it, the default Up navigation will be processed automatically. If any activity on the parent chain requires additional intent arguments, the Action subclass must override the onPrepareNavigateUpTaskStack (android.app.taskStackBuilder) method to file these arguments. For more information about navigation in the app, see the tasks and stack back from the developer's guide and navigation from the design guide. See The TaskStackBuilder class and getParentActivityIntent activity methods, shouldUpRecreateTask (android.content.Intent) and navigateUpTo (android.content.Intent) for help with custom up navigation. The AppNavigation sample app in Android SDK is also available for reference. Returns boolean is true if Up Navigation is completed successfully and this action was finished, false otherwise. Added to the API level 16 Deprecated in API level 30 public boolean onNavigateUpFromChild (child activity) This method has been deprecated in API level 30. Use onNavigateUp instead. This is called when the child activity of this one tries to navigate up. Implementing by default simply calls onNavigateUp (parent) to this activity. Child activity options: activity, making a call. Public boolean onOptionsItemSelected (menuItem) This hook is called whenever an item in the menu of your options is selected. The default implementation simply returns false to have normal processing (call a Runnable item or send a message to its handler as needed). You can use this method for any items you would like to do without these other objects. Derivative classes should be called to the base class to process the default menu. MenuItem option options: Selected menu item. This value cannot be zero. Returns boolean boolean Return false to allow normal menu processing to continue, true to consume it here. See also: onCreateOptionsMenu (Menu) public void (Menu menu) This hook is called whenever the options menu closes (either by user, canceling the menu with the back/menu button, or when selecting the item). Goods). Menu menu: Options menu, as shown last time or first initiated onCreateOptionsMenu .). The public void onPictureInPictureModeChanged (boolean isInPictureInPictureMode, Configuration newConfig) is caused by the system when the activity changes in picture-in-picture mode and from it. This method provides the same configuration that will be sent in the next call onConfigurationChanged (android.content.res.Configuration) after the action enters this mode. InPictureInPictureMode boolean options: True if the action is in picture mode. New ConfigurationConfig: A new action configuration with a state. See also: R.attr.supportsPictureInPicture public boolean onPictureInPictureRequested () This method is called the system in various cases where the image in the image mode should be entered with support. It's up to the app developer to choose whether to call enterPictureInPictureMode (android.app.PictureInPictureParams) at this time. For example, the system will call this method when the action goes into the background mode, so the app developer may want to switch the action to PIP mode. Returns boolean is true if the activity has received this callback no matter if it acts on it or not. If this is not the case, the framework assumes that the app has not been updated to use this callback, and in turn will send an out-of-date callback onUserLeaveHint in order for the app to enter picture-in-picture mode. Public void atPrepareNavigateUpTaskStack (TaskStackBuilder Builder) Preparing a synthetic stack of tasks that will be created while navigating up from another task. This method gets TaskStackBuilder with a built-in series of intentions created by onCreateNavigateUpTaskStack (android.app.TaskStackBuilder). If any additional data needs to be added to these intentions before starting a new task, the application should override this method and add that data here. Options builder TaskStackBuilder: TaskStackBuilder, which was inhabited by intentions onCreateNavigateUpTaskStack. Public boolean onPrepareOptionsMenu (Menu menu) Prepare a standard menu of screen options for display. This is called right before the menu is displayed, every time it is displayed. You can use this method to effectively turn on/off items or change content dynamically. The default implementation updates the menu elements of the system, depending on the state of action. Derivative classes should always take place before the basic class is implemented. Menu menu: Options menu, as shown in the last or first-time initialized onCreateOptionsMenu. Returns boolean You have to go back true for the menu to be displayed; If you come back it will not be shown. See also: onCreateOptionsMenu (Menu) public boolean onPreparePanel (int featureId, View view, menu menu) Default implementation Window.Callback.onPreparePanel (int, View, Menu) for events. It's This. until the new onPrepareOptionsMenu (Menu) method for the Window.FEATURE_OPTIONS_PANEL panel, so activity subclasses don't have to deal with function codes. FeatureId int options: The panel that appears. View: This value may be invalid. Menu Menu: This value can't be zero. Returns boolean boolean You have to go back true for the panel that will show up; If you come back false, it will not be shown. Public void on ProvideAssistContent (AssistContent outContent) Is called when a user requests help to provide links to content related to current activities. Before being called, outContent Intent is filled with the basic intent of the activity (intention returns getIntent). Additional Intent services are devoid of any types that are not valid for PersistableBundle or non-currency Parcelables, while Flags Intention FLAG_GRANT_WRITE_URI_PERMISSION and Intention FLAG_GRANT_PERSISTABLE_URI_PERMISSION cleared of intent. Custom implementation can adjust content intentions to better reflect the context of top-level activity, and fill it with ClipData with additional content of interest to the user at present. For example, an image gallery app that has launched into action that allows the user to flick through the images should change the intention to refer to the current image they are looking for; Such an app when showing a list of photos should add ClipData, which has links to all the photos that are currently visible on the screen. OutContent AssistContent Options: Return Assistance Content. Public void onProvideKeyboardShortcuts (keyboardShortcutGroup list; data, menu menu, int deviceId) is called when requesting keyboard shortcuts for the current window. Data options list: List of data to fill with shortcuts. Menu: Current menu, which can be zero. This value can be zero. deviceId int: AN ID for a connected device that should be shorted. Public Uri onProvideReferrer () Override to create the desired reference for the content that is currently displayed by the app. Implementation by default returns zero, that is, the reference will be just an android application: from the name of the package of this action. Bring back the non-zero Uri to have that delivered as an intention and EXTRA_REFERRER any action started with it. Added to the API level 1 Deprecated in the API level 15 public object onRetainNonConfigurationInstance () is called by the system as part of the destruction of activity due to a configuration change, when it is known that the new instance will be immediately created for the new configuration. You can return any object you like here, including the action instance itself, you can later receive by calling getLastNonConfigurationInstance in the new action instance. If you're focused on Build.VERSION_CODES. HONEYCOMB or later, consider instead using the zlt;/KeyboardShortcutGroup/lt;/KeyboardShortcutGroup with Fragment-setReteinstans (bulean). This feature is called solely as optimization, and you don't have to rely on it being called. When called, a number of safeguards will be made to help optimize the configuration switching: the feature will be able to switch between onStop () and onDestroy. A new instance of action will always be immediately created after this one onDestroy is called. In particular, no messages will be sent during this time (when the returned object has no action to be associated with). The object you return to here will always be available in the getLastNonConfigurationInstance method of the next instance of the action described there. These safeguards are designed so that the action can be used to extend the vast state from the old to the new instance of the action, from downloaded bit cards to network connections and even active flow running. Please note that you should not distribute data that may vary depending on the configuration, including any data downloaded from resources such as strings, layouts, or drawables. Guarantee not processing messages when moving on to the next activity makes it easier to use active objects. For example, if an saved state is AsyncTask, you are guaranteed that its callback functions (e.g. AsyncTask.onPostExecute (Result) will not be called from the call here until you follow the next onCreate (android.os.Bundle) feature. (Note, however, that of course there is no such guarantee for AsyncTask.doInBackground (Params...) as it works in a separate thread.) Note: In most cases, you should use the Fragment-setRetainInstance (boolean) fragment instead; It's also available on older platforms through Android support libraries. This hook is called when the user signals a desire to start a search to the next instance of the public boolean onSearchRequested (SearchEvent searchEvent) action. You can use this feature as an easy way to run a search user interface in response to a menu item, search button, or other widgets in your activity. If not overworked, call this feature the same as the startSearch (null,false, null,false) call, which triggers a search for the current activity listed in its manifest, see SearchManager. You can override this feature to force a global search, for example, in response to a dedicated search key, or completely block the search (just by returning the false one). Note: When you start in configuration-UI_MODE_TYPE_TELEVISION or Configuration-UI_MODE_TYPE_WATCH, the default implementation changes to simply return the false, and you should your own custom implementation if you want to support the search. SearchEvent SearchEvent: SearchEvent that signaled this search. This value can be zero. Returns boolean Returns True If Search Is Launched, and False If Activity Activity don't respond to a search. The default implementation always returns correctly, except when the UI_MODE_TYPE_TELEVISION where it returns false. Public boolean onSearchRequested () Is called when the user signals about the desire to start a search. Returns boolean true if the search is launched, false if the activity refuses (blocks) See also: onSearchRequested (SearchEvent) public void onTopResumedActivityChanged (boolean isTopResumedActivity) Is called when the activity receives or loses the top renewed position in the system. Starting with Build.VERSION_CODES. Multiple actions can be resumed simultaneously in multi-window modes and multiple displays. This callback should be used instead of onResume as a sign that the action may try to open exclusive access devices such as the camera. It will always be delivered after the resumption of action and before its pause. In some cases, it may be missed and the activity may go straight from onResume () to onPause without getting the top state resumed. Options is a ResumedActivity boolean: true, if this top resumed activity in the system, false otherwise. A call with this as true will always be accompanied by another with a false. See also: onResume ()onPause ()onWindowFocusChanged (boolean) public boolean onTouchEvent (MotionEvent event) Called when the touch screen event was not processed by any of the views under it. It is most useful for handling sensory events that occur outside window boundaries where there is no representation to get it. MotionEvent Event Options: The touchscreen event is being processed. Returns boolean Return is true if you consumed an event false if you don't. The default implementation always returns false. Public boolean onTrackballEvent (MotionEvent event) Is called when the trackball has been moved and not handled by any of the activities within. So, for example, if the trackball moves while the focus is on the button, you get a call here because the buttons usually do nothing with trackball events. The call here occurs before the trackball movements are converted to key DPAD events, which are then sent back to the view hierarchy, and will be processed at the point for things like focus navigation. MotionEvent Event Options: Trackball event is processed. Returns boolean Return is true if you consumed an event false if you don't. The default implementation always returns false. Public void on The Associated Press is called whenever a key, touch, or trackball event is sent to an activity. Implement this method if you want to know that the user has interacted with the device in some during your activity works. This callback and onUserLeaveHint are designed to help you manage your status bar notifications wisely; specifically, to assist activities determine the correct time to cancel the notice. All the challenges to your activity activities The callback will be accompanied by calls to onUserInteraction. This ensures that your activity will be told about relevant user actions such as pulling down the notification bar and touching the item there. Note that this callback will be called for a touch action that starts with a touch gesture but cannot be triggered for the sensory and sensory actions that follow. See also: Added to API level 21 Deprecated in API level 26 public void onVisibleBehindCanceled () This method has been deprecated in API level 26. The functionality of this method is no longer supported as Build.VERSION_CODES. O and will be removed in a future release. It is called when the translucent activity over this action becomes opaque or other actions are triggered. The actions that override this method should cause super.onVisibleBehindCanceled () or SuperNotCalledException to be thrown. When this method is called, the action has 500 msec to release any resources it can use by being visible in the background. If the action does not return from this method in 500 msec, the system will destroy the action and kill the process to recover resources for another process. Otherwise onStop () will be called upon return. If you override this method, you should call before the superclass is implemented. See also: please RequestVisibleBehind (boolean) public void onWindowAttributesChanged (WindowManager.LayoutParams params) It is called whenever the current window attributes change. Options params WindowManager.LayoutParams Public Void onWindowFocusChanged (boolean hasFocus) Is called when the current window of activity gains or loses focus. This is the best indicator of whether this action is an entity with which the user actively interacts. Implementing by default clears the key tracking state, so you should always call it. Please note that this provides information about the state of global focus that is managed regardless of the lifecycle of your activity. So while focus changes tend to have something to do with life cycle changes (activities that are stopped usually won't get a focus box), you don't have to rely on any particular order between callbacks here and in other life cycle methods such as onResume. However, as a rule, foreground activity will have the focus of the window... unless it has displayed other dialogues or pop-ups that take the attention of input, in which case the activity itself will not have a focus when other windows have it. Similarly, the system may display system-level windows (such as a status notification panel or system alert) that will temporarily be at the entrance to the windows without stopping the foreground. Starting with Build.VERSION_CODES. There may be several resumed actions in multi-window mode at the same time, so the resumed state does not guarantee the focus of the windows, even if there are no overlays above. If the intention is intent to know when the action is most active, the one with which the user interacted last among all activities, but not including inactive windows such as conversations and pop-ups, then should use onTopResumedActivityChanged (boolean). On the platform version up to Build.VERSION_CODES. In, onResume is the best indicator. Options have a Focus boolean: Does the window of this activity have a focus. Public ActionMode onWindowStartingActionMode (ActionMode.Callback callback, int type) is called in running mode for this window. Gives the callback the ability to handle the mode of action in its own unique and beautiful way. If this method returns the zero system can choose a way to present the mode or choose not to start the mode at all. Returns ActionMode This value may be invalid. Public ActionMode onWindowStartingActionMode (ActionMode.Callback callback callback) gives the action the ability to control the user interface for the mode of action requested by the system. Note: If you're looking for a callback notification that the activity mode is running, see AtactionModeStarted (android.view.ActionMode). ActionMode.Callback Callback Call: A callback that needs to control the new ActionMode Action Mode, or zero if the activity does not want to provide special processing for that mode of action. (This will be handled by the system.) Public emptiness openOptionsMenu () Software opens the menu of options. If the options menu is already open, this method does nothing. Public void overridePendingTransition (int enterAnim, int exitAnim) Call immediately after one of the flavors startActivity (android.content.content)) or finish () to indicate a clear transition animation to perform the next. In Build.VERSION_CODES. JELLY_BEAN alternative to this start is to deliver the desired animation information through the ActivityOptions package for startActivity (android.content.Intent, android.os.Bundle) or related features. This allows you to specify a user's animation even when you start an action from outside the context of the current top activity. EnterAnim int: Animation resource ID for use for incoming actions. Use 0 without animation. exitAnim int: Animated Resource ID for use for outgoing activities. Use 0 without animation. public void to recreate () the reason for this action to be recreated with a new instance. This essentially leads to the same thread as when you create the Action due to a configuration change - the current instance will go through its lifecycle on OnDestroy and the new instance created after it. public invalid (Application.ActivityLifecycleCallbacks callbacks callbacks callbacks) Register a copy of Application.ActivityLifecycleCallbacks that receives lifecycle callbacks only for this activity. For anyone The callbacks registered here will always occur in these callbacks. This means: If multiple callbacks are registered here, They receive events in the first (up application.ActivityLifecycleCallbacks'onActivityPost Suspend, the last of the order. : Call a instance to register This value cannot be zero. Ask a local copy of the activity app to release its memory. This requires the action to be destroyed, but does not complete the action - a new instance of the action will later be re-created if necessary because the user is returning to it. Returns boolean Returns true if the activity was able that it began the process of destroying its current instance; returns false if for any reason it can't be done: it is being seen by the user, it is already being destroyed, it is being finished, it has not yet retained its state, etc. public report on the void Of LyDrawn () Tell the system that your application is now fully drawn, for diagnostic and optimization purposes. The system can set up optimization to prioritize the work that happens before reportFullyDrawn is called to improve the launch of the application. Distorting the launch window by calling reportLyDrawn too late or too early can reduce the performance of apps and startups. It is also used to help time the app launch the tool, so that the app can report when it is fully in use; without it, the only thing the system itself can determine is the point at which the action window is first drawn and displayed. To take part in measuring the launch time of the application, you should always call this method after the first launch (when onCreate (android.os.Bundle) is called), at the point where you have fully drawn your user interface and are populated by all the important data. You can safely call this method anytime after the first run, in which case it will simply be ignored. If this method is called before the action window is first drawn and displayed as measured by the system, the time reported here will be moved to the time measured by the system. public final void of the requestPermissions (Resolution String, Int requestCode) permissions to be granted to this app. These permissions must be requested in your manifest, they should not be granted to your app, and they must have a dangerous level of protection, regardless of whether they are declared by a platform or a third-party app. The usual PermissionInfo.PROTECTION_NORMAL permissions are during installation, if requested in the manifest. Signature permissions PermissionInfo.PROTECTION_SIGNATURE that are granted at the time of installation if requested in the manifest, and your app's signature corresponds to the signature of the app declaring permissions. The call should be made before calling this API to check if the system recommends showing the rationale for the user interface before asking for permission. If your app doesn't have the permissions requested, the user will be provided with the user interface to accept them. Once a user has accepted or declined the requested permissions, you will receive a callback to onRequestPermissionsResult (int, java.lang.String, int), informing you whether permissions have been issued or not. Please note that requesting permission does not guarantee that it will be granted and your application should be able to operate without that permission. This method can start an action that allows the user to choose which permissions to provide and which to reject. Therefore, you should be prepared for the fact that your activities may be suspended and resumed. In addition, granting some permissions may require a reboot of the app. In this case, the system recreates the activity stack before delivering the result to RequestPermissionsResult (int, java.lang.String, int). When checking if you have a permit, you should use ContextWrapper.checkSelfPermission (java.lang.String). Calling this API to get permissions already granted to your app will show the user the user interface to decide whether the app can still hold those permissions. This can be useful if the way permission-protected data is used changes significantly. You can't ask for permission if your activity isn't true, because in this case the action won't receive results callbacks, including on RequestPermissionsResult (int, java.lang.String, int). The RuntimePermissions example demonstrates how to use this method to request permissions while running. Throws IllegalArgumentException if requestCode is negative. Added to API level 21 Deprecated in API level 26 public boolean requestVisibleBehind (boolean visible) This method has been deprecated in API level 26. The functionality of this method is no longer supported as Build.VERSION_CODES. O and will be removed in a future release. Actions that want to remain visible beyond the translucent activity above them should trigger this method at any time between the beginning of onResume and the return from onPause. If this call is successful, the action will remain visible after the onPause call and will be allowed to continue playing media in Mode. This call is reset every time it's brought to the front. That is, every time onResume () called activity will be considered not asked visible behind. So if you want this activity to still be visible in the you should call this method again. Only full-screen opaque actions can make this call. That is, this challenge is nop for dialogue and translucent action. In all circumstances, the action must stop the replay and release of resources before or within the onVisibleBehindCanceled call or if the call is returned false. The false will be returned anytime this method is called between the return onPause and the next onResume call. Options visible boolean: it is true to notify the system that the activity wants to be visible for other translucent actions, false to indicate the opposite. Resources should be released when false data is transmitted to this method. Returns boolean as a result of visibiity. If this is true, the activity will remain visible outside onPause if the next action is translucent or not full-screen. If this is false, the action may not expect other translucent actions to be visible, and media resources should be stopped. A false return can occur instead of a call toVisibleBehindCanceled, so the return value must be verified. See also: onVisibleBehindCanceled () public final boolean requestWindowFeature (int featureId) Include advanced window features. It's a convenience to call getWindow. FeatureId int options: the desired function, as defined in the window. The boolean Returns returns correctly if the requested feature is supported and is now on. See also: Window.requestFeature (int) public finale T requireViewById (int id) Finds a view that has been identified by the android attribute XML, which has been processed in onCreate (Bundle), or throws IllegalArgumentException if the ID is invalid, or there is no corresponding representation in the hierarchy. Note: In most cases - depending on compiler support - the view is automatically discarded as the target class. If the target class type is not limited, you may need a clear throw. ID int options: The ID to search for Returns T view with this ID This value cannot be zero. See also: View.requireViewById (int)findViewById (int) public final runOnUiThread (Runnable Action) launches this action on the user interface stream. If the current thread is a user interface stream, the action is immediate. If the current thread is not a user interface stream, the action should be placed in the UI stream event queue. Runnable Action Options: Run-up action on the UI stream of public void setActionBar (toolbar) Set a toolbar to perform in action panel for this activity window. When you dial to non-zero value, the getActionBar method returns an ActionBar object that can be used to control the dashboard as if it were a traditional window decor action bar. The toolbar menu will be filled with activity options menus, and the navigation button will be connected to the standard home menu selection action. To use the toolbar in the Action Action window the app should not request the Window FEATURE_ACTION_BAR window function. Options toolbar: A toolbar installed as an Activity action bar, or zero to clean it This value can be zero. The public void set by FinishOnTouchOutside determines whether this activity is completed when you touch the outside window. Options to finish boolean public void setInheritShowWhenLocked (boolean inheritShowWhenLocked) indicates whether this action should be shown at the top of the lock screen whenever the lockscreen is up, and this action has another action behind it with the showWhenLock attribute set. That is, this action can only be seen on the lock screen, if there is another action with the attribute showWhenLock, visible at the same time on the lock screen. An example of how to use this is permission dialogues, which should only be visible on the lock screen if their requesting activity is also visible. This value can be established as an obvious attribute using an android. R.attr'inheritShowWhenLocked. Options inheritShowWhenLocked boolean: it is true to show activity at the top of the lock screen when this activity has another action behind it with a showWhenLock set of attributes; otherwise false. See also: setShowWhenLocked (boolean)R.attr.inheritShowWhenLocked public void setLocusContext (LocusId locusId, bundle bundle) sets LocusId for this activity. The locus ID helps identify different instances of the same Activity class. For example, a locus ID based on a particular conversation can be installed in an app chat. The system can then use this locus ID along with the content of the application to provide rating signals on different user interface surfaces, including sharing, notifications, shortcuts, and so on. It is recommended that you install the same locus ID in the label locus ID with setLocusId so that the system can recognize the appropriate rating signals linking the activity locus ID to the corresponding label. LocusId LocusId options: a unique, stable identifier that identifies this instance of activity from others. This can be associated with a label using setLocusId with the same line locus id. This value can be zero. Kit: A set of extras or upgraded as part of this locus context. This can help provide additional metadata, such as URLs, and conversation participants that are specific to the context of the activity. This value can be zero. See also: ContentCaptureManagerContentCaptureContext Public Final Space SetMediaController (MediaController Controller) MediaController to send media keys and volume changes. The controller will be tied to the window of this Action. Media key and volume events received during the foreground will be redirected to the controller and used to call traffic control or adjust volume. This can be used instead or in addition to setVolumeControlStream (int) to influence a particular session instead of a particular thread. Flow. there is no guarantee that the hardware volume controls will always change the volume of the session (for example, if the call continues, the volume of the flow may be changed instead). Use null as a controller to reset back to the default. MediaController Controller: A session controller that should receive media keys and volume changes. Public void setPictureInPictureParams (PictureInPictureParams params) updates the activity properties of the picture in the picture or sets it for use later at the entrancePictureInPictureMode () is called. PictureInPictureParams Parama Options: New Settings for Image in Picture. This value cannot be zero. Added to the API level 1 Deprecated in API level 24 of the public final void setProgress (int progress) This method has been deprecated in API level 24. No longer supported, starting with API 21. Sets progress for the bars to progress in the title. In order for the progress bar to be shown, the function must be requested through requestWindowFeature (int). Int Progress Options: Progress for Bar Progress. Valid ranges from 0 to 10,000 (both inclusive). If 10,000 is given, the progress of the bar will be completely filled and will disappear. Added to API level 1 Deprecated in API level 24 public final void setProgressBarDeterminate (boolean uncertain) This method has been deprecated in API level 24. No longer supported, starting with API 21. Sets whether to determine the horizontal bar of progress in the title (the circular is always uncertain). In order for the progress bar to be shown, the function must be requested through requestWindowFeature (int). Options uncertain boolean: Should the horizontal bar progress be defined. Added to API level 1 Deprecated in API level 24 public final void setProgressBarIndeterminateVisibility (boolean visible) This method has been deprecated in API level 24. No longer supported, starting with API 21. It establishes the appearance of an uncertain bar of progress in the title. In order for the progress bar to be shown, the function must be requested through requestWindowFeature (int). Options visible boolean: Should show the progress of the bars in the title. Added to API level 1 Deprecated in API level 24 public final void setProgressBarVisibility (boolean visible) This method has been deprecated in API level 24. No longer supported, starting with API 21. It establishes the appearance of the bar of progress in the title. In order for the progress bar to be shown, the function must be requested through requestWindowFeature (int). Options visible boolean: Should show the progress of the bars in the title. Public set of emptinessRequestedOrientation (int requestedOrientation) Changing the desired orientation of this activity. If the action is currently in the foreground, or otherwise affecting the orientation of the screen, the screen will be immediately changed (perhaps resulting in the action being restarted). restarted). this will be used in the next visible action. Options requested Orientation int: Orientation is constant, used in ActivityInfo-screenOrientation. The value of android.content.pm.ActivityInfo.SCREEN_ORIENTATION_UNSET, ActivityInfo.SCREEN_ORIENTATION_UNSPECIFIED, ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_USER, ActivityInfo.SCREEN_ORIENTATION_BEHIND, ActivityInfo.SCREEN_ORIENTATION_SENSOR, ActivityInfo.SCREEN_ORIENTATION_NOSENSOR, ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_REVERSE_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_REVERSE_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR, ActivityInfo.SCREEN_ORIENTATION_USER_LANDSCAPE, ActivityInfo.SCREEN_ORIENTATION_USER_PORTRAIT, ActivityInfo.SCREEN_ORIENTATION_FULL_USER, or ActivityInfo.SCREEN_ORIENTATION_LOCKED Added to API Level 1 Decatpreed in API This method was onched at 24 API. No longer supported, starting with API 21. Sets secondary progress for the bar of progress in the title. This progress is made between the main progress (setProgress (int) and the background. It can be ideal for media scenarios such as displaying the progress of buffering, while the default progress shows the progress of playback. whether to show the action on top of the lock screen whenever the lock screen and activity resumes. Normally, the action goes into a stopped state if it is running during the lock screen, but with this flag set, the action will remain in a renewed state visible at the top of the lock screen. This value can be set as an explicit attribute using R.attr.showWhenLocked. Options show where only a boolean is fired: it's true to show activity at the top of the lock screen; otherwise false. See also: setTurnScreenOn (boolean)R.attr.turnScreenOnR.attr.showWhenLocked public void setTitle (name CharSequence) Name change associated with this activity. If this is a top-level action, the name of the window will change. If it's a built-in action, the parent can do whatever they want with it. CharSequence's name parameters of the public invalid setTitle (int titleId) Name change related to this activity. If this is a top-level action, the name of the window will change. If built-in action, the parent can do whatever he wants with it. Added to API Level 1 Deprecated in API Level 21 public void setTitleColor (int textColor) textColor) the method has been wilted in API level 21. Instead, use action bar styles. Change the color of the title associated with this action. This method is used to start with the Level 11 API and is replaced by action bar styles. For information about Action Bar style, read the Action Bar developer's guide. public boolean setTranslucent (Galilee translucent) Conversion activity, which, in particular, with R.attr.windowIsTranslucent or R.attr.windowIsFloating attribute, to full-screen opaque activity, or convert it from opaque back into translucent. The parameters are translucent boolean: truly convert from opaque to translucent. false transformation from translucent to opaque. Returns boolean result installation transparency. The return is true if the set is successful, false otherwise. setTurnScreenOn (boolean turnScreenOn) determines whether to turn on the screen when the action resumes. Typically, the action goes into a stopped state if it's running during the screen that's being turned off, but with this set of flags, the action will cause the screen to turn on if the action is visible and resumed because of the upcoming screen. The screen will not be turned on unless the action is visible after the screen is turned on. This flag is commonly used in conjunction with the R.attr.showWhenLocked flag to make sure the action is visible after the screen is turned on when the screen locks up. In addition, if this flag is installed and the action causes KeyguardManager-requestDismissKeyguard (Activity, KeyguardManager.KeyguardDismissCallback), the screen will be turned on. TurnScreenOn boolean options: turnScreenon boolean: turn screen correctly; otherwise false. Public emptiness set byvisible (boolean visible) Control of whether the main window of this activity is visible. This is only for a special case of action that is not going to show the user interface itself, but can't just finish up onResume () because it has to wait for the linking service or such. Installing this to a false one prevents your user interface from being shown during that time. The default for this is taken from the R.attr.windowNoDisplay action theme attribute. The parameters of the visible boolean public final void set VolumeControlStream (int streamType) offers an audio stream, the volume of which must be changed by the hardware volume control. The proposed audio stream will be tied to the window of this activity. The volume requests received during the foreground will affect this thread. There is no guarantee that hardware volume controls will always change the volume of that thread (for example, if the call continues, the flow can be changed instead). To reset back by default, use USE_DEFAULT_STREAM_TYPE. StreamType int options: The type of audio stream that needs to be changed by volume hardware. public emptiness setVrModeEnabled (boolean included, included, Turn on or off virtual reality (VR) mode for this activity. VR mode is a hint for the Android system to go into mode optimized for VR applications, while this activity has the user's attention. It is recommended that apps additionally announce R.attr.enableVrMode in their manifesto to ensure smooth activity transitions when switching between VR activities. If the requested VrListenerService component is not available, VR mode will not be launched. Developers can handle this case as follows: String servicePackage - com.whatever.app; String serviceClass - com.whatever.app.MyVrListenerService; The name of the VrListenerService component to run. ComponentBing Component - new ComponentName (servicePakage, serviceClass); Try and installVrModeEnabled (admittedly, myComponentName); - Catch (PackageManager.NameNotFoundException e) - List of installed - qlt;applicationInfo'gt; getPackageManager (.getInstalledApplications (0); boolean isInstalled - false; for (AppInfo app : installed) - if (app.packageName.equals (servicePackage)) Let the user turn it on. startActivity (new intention (Settings.ACTION_VR_LISTENER_SETTINGS)) The package is not installed yet. Send the intention to download this. sentIntentToLaunchAppStore (Package service); Options included boolean: true to turn this mode on. requestedCoponent component name: The name of the component to be used as VrListenerService during the VR mode. This value cannot be zero. Public boolean shouldUpRecreateTask (targetIntent intention) Returns correctly if the app has to recreate the task when navigating 'up' from this activity using targetIntent. If this method returns false, the app can trivially trigger navigateUpTo (android.content.Intent) using the same parameters to properly navigate. If this method returns false, the application must synthesize a new stack of tasks using TaskStackBuilder or other similar navigation mechanism. The options targetIntent Intent: The intention representing the target destination for pre-navigation Returns boolean is true if the navigation up should recreate a new stack of tasks, false if the same task should be used to assign a public boolean showAssist (Bundle args) Ask the current assistant to show the user. This only works if the call is the current foreground action. It's the same as calling VoiceInteractionService.showSession and asking for all possible context. The receiver will always see VoiceInteractionSession.SHOW_SOURCE_APPLICATION set. Returns boolean Returns true, if the assistant was successfully called, still false. For example, false returned if the subscriber is not the current top activity. Added to API level 8 Deprecated in API level 15 public final boolean showDialog (Int ID, Bundle args) It's zlt;/ApplicationInfo'gt; has wilted in API level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Show the dialogue that is driven by this action. The call to onCreateDialog (int, android.os.Bundle) will be made with the same ID for the first time it is called for this ID. The dialogue will then be automatically saved and restored. If you're focused on Build.VERSION_CODES. HONEYCOMB or later, consider instead using DialogFragment instead. Every time the dialogue is shown, onPrepareDialog (int, android.app.Dialog, android.os.Bundle) will be done to provide an opportunity to do any timely preparation. the public void of showLockTaskEscapeMessage () shows the user that the system has identified a message in order to tell the user how to get out of the lock problem mode. The task containing this action must be in lock mode during that call to display the message. Public void startActivities (intentions, kit options) Launch a new action. You won't get any information about when the action comes out. This implementation overlaps the base version by providing information about the activities that run. Because of this additional information, the flag FLAG_ACTIVITY_NEW_TASK launch is not required; if you don't specify, a new action will be added to the caller's task. This method throws ActivityNotFoundException if no activity has been found to run this intent. Options of Intention Intention: Intentions to Begin. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Throws android.content.ActivityNotFoundException See also: startActivities (Intention)startActivityForResult (Intention, Int) public void startActivity (intention, set options) Launch of new activities. You won't get any information about when the action comes out. This implementation overlaps the base version by providing information about the activities that run. Because of this additional information, the flag FLAG_ACTIVITY_NEW_TASK launch is not required; if you don't specify, a new action will be added to the caller's task. This method throws ActivityNotFoundException if no activity has been found to run this intent. Options intention Intention: Intent to start. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Throws android.content.ActivityNotFoundException See also: startActivity (Intention, Int) Public Void startActivityForResult (Intention Intention, Int requestCode, Options Kit) Running an activity for which you would like a result when it has it When this activity comes out, your onActivityResult method will be called with requestCode data. Using a negative Code request is the same as a startActivity call (Intention) (activity is not started as a subactivity). Please note that this method should only be used with Intent protocols that are determined to return the result. In other protocols (such as Intent ACTION_MAIN or Intent-ACTION_VIEW) you can't get a result when you expect. For example, if the action you're running is using Intent-FLAG_ACTIVITY_NEW_TASK, it won't work in your task, and thus you'll get the cancellation result right away. As a special case, if you call startActivityForResult () with a requestcode zgt; x 0 during the initial onCreate (Bundle savedInstanceState)/onResume() of your activity, your window will not be displayed until the result is returned from the activity. This is to avoid visible flickering when redirected to another action. This method throws ActivityNotFoundException if no activity has been found to run this intent. Options intention Intention: Intent to start. requestCode int: If you are 0, this code will be returned to onActivityResult when you leave the action. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Throws android.content.ActivityNotFoundException See also: Added to API level 16 Deprecated in API level 30 public void startActivityFromChild (Action of the Child, Intention Intention, Int requestCode, Bundle Options) This method has been deprecated in API level 30. Use androidx.fragment.app.FragmentActivity-startActivityFromFragment (androidx.fragment.app.Fragment,Intent,int,Bundle) It's called when child activity calls it startActivity (Intention) or startActivityForResult (Intention, Int) method. This method throws ActivityNotFoundException if no activity has been found to run this intent. Child activity options: activity, making a call. This value cannot be zero. Intention intention: intention to start. requestCode int: Response request code. 0 if the answer is not requested. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Throws android.content.ActivityNotFoundException See also: startActivity (Intention)startActivityForResult (Intention, Int) public boolean startActivityIfNeeded (Intention, Int requestCode, Options Set) Variation to start an activity only if a new instance of activity is needed to handle that intent. In other words, it's the same as startActivityForResult (android.content.Intent, Int), except if you use the Intent-FLAG_ACTIVITY_SINGLE_TOP flag, or singleTask or singleTop launchMode, and activity activity Intentions are handled just like your current running activity, then a new instance is required. In this case, instead of the normal behavior of the call on NewIntent (Intention), this feature will return and you can handle the intention yourself. This function can only be called from top-level activities; if it is called out of the child's action, an exception to the execution time will be thrown. Options intention Intention: Intent to start. This value cannot be zero. requestCode int: If it is 0, this code will be returned to onActivityResult () when the action is released, as described in startActivityForResult (Intention, Int). Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Returns boolean If new action was launched then truly returned; otherwise the false returns and you have to handle the intentions yourself. See also: StartActivity (Intent)startActivityForResult (Intention, Int) public void startIntentSender (Intention, Intention fillIntent, Int flagsMask, Int flagsValues, int extraFlags) Same, that call startIntentSender (android.content.IntentSender, android.content.Intent, int, int. Int. Intent Options Intent: IntentionSender for Launch. that you would like to change. flagsValues int: Desirable values for any bits set in the flagsMask extraFlags int: Always tuned to 0. Throws IntentSender.SendIntentException public void startIntentSender (Intent, intention fillIntent, int flagsMask, int flagsValues, int extraFlags, complete options) How to startActivity (android.content.Intent, android.os.Bundle), but taking IntentSender to begin with; For more information, see startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, android.os.Bundle). IntentSender Intent Options: IntentSender to Launch. fillInIntent Intention: If not zero, it will be provided as an intent parameter to IntentSender'sendIntent. This value can be zero. flagsMask int: Intentional flags in the original IntentSender that you would like to change. flagsValues int: Desirable values for any bits set in the flagsMask extraFlags int: Always tuned to 0. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). If have also been provided by IntentSender, the options given here will override any conflict with those that have been provided by IntentSender. Sends IntentionsSender.SendIntentException Public Void startIntentSenderForResult (IntentionSender, Int requestCode, Intention fillIntent, Int Int int flagsValues, int extraFlags) Same as to call startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int, int, android.os.Bundle) with no options. IntentSender Intent Options: IntentSender to Launch. requestCode int: If you are 0, this code will be returned to onActivityResult when you leave the action. fillInIntent Intention: If not zero, it will be provided as an intent parameter to IntentSender'sendIntent. This value can be zero. flagsMask int: Intentional flags in the original IntentSender that you would like to change. flagsValues int: Desirable values for any bits set in the flagsMask extraFlags int: Always tuned to 0. Throws IntentSender.SendIntentException Public Void startIntentSenderForResult (IntentionSender, Int requestCode, Intention fillInintent, Int flagsMask, Int flagsValues, Int extraFlags, Kit Options) How to startActivityForResult (android.content.Intent, int), but allows you to use If IntentSender for activity, this activity will be started, as if you called a regular startActivityForResult (android.content.Intent, int) here; otherwise, the associated action will be performed (such as sending a broadcast) as if you called IntentSender'sendIntent on it. requestCode int: If you are 0, this code will be returned to onActivityResult when you leave the action. fillInIntent Intention: If not zero, it will be provided as an intent parameter to IntentSender'sendIntent. This value can be zero. flagsMask int: Intentional flags in the original IntentSender that you would like to change. flagsValues int: Desirable values for any bits set in the flagsMask extraFlags int: Always tuned to 0. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). If options were also provided by IntentSender, the options given here will override any conflict with those that have been provided by IntentSender. Throws IntentSender.SendIntentException Added to API Level 16 Deprecated in API Level 30 Public Void startIntentSenderFromChild (Child Activity, IntentSender Intentions, Int RequestCode, Intention fillIntent, Int flagsMask, Int flagsValues, Int extraFlags, Bundle variants) This method has been deprecated in API level 30. Instead, use startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int, android.os.Bundle). Like startActivityFromChild (android.app.Activity, android.content.Intent, int), but taking IntentSender; For more information, see (android.content.IntentSender, int, android.content.Intent, int, int, int). Параметры ребенка Намерение деятельности Намерение НамерениеSenderCode Int fillInIntent Намерение флагиМаска Int flagsValues Int extraFlags Int Int Set: This value may be invalid. Throws IntentSender.SendIntentException Added to API Level 5 Deprecated in API Level 30 Public Void startIntentSenderFromChild (Child Activity, IntentSender Intentions, Int requestCode, Intention fillInintent, Int flagsMask, Int flagsValues, Int extraFlags) This method has been deprecated in THE API 30. Instead, use startIntentSenderForResult (android.content.IntentSender, int, android.content.Intent, int, int, int). Same as startIntentSenderFromChild (android.app.Activity, android.content.IntentSender, int, android.content.Intent, int, int, int, android.os.Bundle) with no options. IntentSender Intent Options: IntentSender requestCode int fillInIntent Intent flagsMask int flagsValues int extraFlags int Throws IntentSender.SendIntentException Public Invalid StartLocalVoiceInteraction (Bundle privateOptions) launches a local voice session. When ready, onLocalVoiceInteractionStarted is called. You can transfer a package of private options to a registered voice interaction service. PrivateOptions Bundle Options: A set of private arguments to the current voice interaction service is a public startLockTask space () Request to put this activity in mode where the user is locked on a limited set of applications. If DevicePolicyManager-isLockTaskPermitted (String) returns according to this component, the current task will be launched directly into LockTask mode. While LockTask mode is active, only applications on the White DevicePolicyManager-setLockTaskPackages (ComponentName, String) can be launched. The user will not be able to leave this mode until this action causes stopLockTask. Calling this method while the device is already in LockTask mode has no effect. Otherwise, the current task will be put into screen pinning mode. In this case, the system will prompt the user to talk to the user to allow the use of this mode. The user can exit at any time through the instructions shown on the request dialogue. The stopLockTask call will also stop this mode. Note: This method can only be called when the action is in the foreground. That is, between onResume and onPause. See also: stopLockTask ()R.attr.lockTaskMode Added to API Level 1 Deprecated in API Level 15 Public Void StartManagingCursor (Cursor c) This method has been deprecated in API level 15. Use the new CursorLoader class with LoaderManager instead; It's also available on older platforms through the Android compatibility package. This method allows activities to take care of the lifecycle management of this Cursor for you based on the lifecycle of the action. That is, when the action is stopped, it will automatically be Cursor'deactivate on this cursor, and when it is later restarted, it will call Cursor'requery for you. When the action is destroyed, all guided Cursors will be automatically closed. If you're focused on Build.VERSION_CODES. HONEYCOMB or or or consider instead using LoaderManager instead, available through getLoaderManager. Warning: Don't call Cursor'close () on the cursor received from the controlled Overy (Uri, String, String, String, String) because the action will do it for you at the appropriate time. However, if you call stopManagingCursor (Cursor) on a cursor from a managed query, the system will not automatically close the cursor and, in this case, you should call Cursor'close. Options c Courser: A cursor that needs to be managed. Public boolean startNextMatchingActivity (intention, kit options) Special version of launch action, for use when replacing other activity components. You can use this to pass the Intention on to the next Action, which can handle it. You usually call it onCreate (Bundle) with the intention of being returned to getIntent. Intent Options: Intent to direct to the next activity. For correct behavior, it must be just like the intention that started its own activity; The only changes you can make are the extras inside it. This value cannot be zero. Options Kit: Additional options for how activities should be started. Read more about this at context.startActivity (intention, kit) Context.startActivity (intention, kit). This value can be zero. Returns boolean Returns Boolean pointing out whether there was another action to begin: true if there was the next action to start, false if not. In general, if the truth comes back you will want to call finish () on yourself. Public boolean startNextMatchingActivity (intention intention) Same as call startNextMatchingActivity (android.content.Intent, android.os.Bundle) with no options. Intent Options: Intent to direct to the next activity. For correct behavior, it must be just like the intention that started its own activity; The only changes you can make are the extras inside it. This value cannot be zero. Returns boolean Returns Boolean pointing out whether there was another action to begin: true if there was the next action to start, false if not. In general, if the truth comes back you will want to call finish () on yourself. Public invalid startPostponedEnterTransition () The beginning of deferred transitions after the postponementBusiness () was called. If you've been called by the project, you should call startPostponedEnterTransition to get your picture started. public void startSearch (String initialquery, boolean selectInitial,Bundle appSearchData, boolean globalSearch) This hook is designed to launch a search user It is usually called from onSearchRequested, either directly from Activity.onSearchRequested or from an override version in any given activity. If your goal is to simply activate the search, it is preferable to callSearchRequested, which may have been overridden elsewhere in your activity. If your goal is to give specific specifics such as contextual data, it is preferable to override toSearchRequested, so that any subscribers to it will benefit from the override. Note: When you run in UI_MODE_TYPE_WATCH, the use of this API is not supported. Initial line options: Any non-zero non-empty line will be inserted as pre-entered text in the search query box. This value can be zero. selectInitial-Kueri boolean: If true, the initial request will be pre-selected, meaning that any further input will replace it. This is useful for cases where the entire pre-formed request is inserted. If this is not true, the selection point will be placed at the end of the inserted query. This is useful when the inserted query is the text that the user has typed in, and the user will expect to be able to continue typing. This option only makes sense if the starting line is a non-empty line. appSearchData Bundle: The app can insert a specific app context here to improve the quality or specifics of your own search. This data will be returned with the intention of SEARCH (s). Null, if no additional data is required. This value can be zero. globalSearch boolean: If it's a lie, it will only run a search that has been specifically defined by the application (which is usually defined as a local search). If you don't define the default search in your current app or action, a global search will be launched. If this is true, it will always be launching a global (such as web) search platform instead. See also: SearchManageronSearchRequested () Added to API Level 1 Deprecated in API Level 15 public void stopManagingCursor (Cursor c) This method has been deprecated in API level 15. Use the new CursorLoader class with LoaderManager instead; It's also available on older platforms through the Android compatibility package. Given the cursor, which was previously given for startManagingCursor (Cursor), stop controlling the activity of this cursor. Warning: After calling this method on the cursor from a managed query, the system will not automatically close the cursor and you should call Cursor'close. Options c Kursor: The cursor that was administered. See also: StartManagingCursor (Cursor) Public Void TakeKeyEvents (boolean get) Request that key events come to this activity. Use this if your activity has no focus views, but activity still wants the ability to handle key events. See also: Window.takeKeyEvents (boolean) public empty triggerSearch (String Query, Bundle appSearchData) Similar to startSearch (String, boolean, Bundle, boolean), but actually launches a search query after calling a search dialogue. Made available for purposes Request line options: Request to start. If the request is empty, it will be ignored. appSearchData Bundle: The app can insert a specific app context here to improve the quality or specifics of your own search. This data will be returned with SEARCH SEARCH Null, if no additional data is required. This value can be zero. Secure Void Protection Methods attachBaseContext (Context newBase) Set the basic context for this ContextWrapper. All calls will then be delegated to the basic context. Throws IllegalStateException if the basic context is already set. NewBase Context options: A new basic context for this wrapper. The protected void on ActivityResult (int requestCode, int resultCode, intent data) is called in the action you launched, giving you the requestCode from which you launched it, the resultCode it returned, and any additional data from it. ResultCode will be used RESULT_CANCELED if the action explicitly returned it, did not return any result or crashed during its operation. The action can never get a result in a renewed state. You can count on the Resume () called after this method, although not necessarily immediately after. If the action has been resumed, it will be suspended and the result will be delivered, and then onResume(. If the action has not been resumed, then the result will be delivered, with onResume () called some time later when the activity becomes active again. This method is never called if your activity is not true. Code Int Query Options: The integer query code is originally supplied to startActivityForResult, which allows you to determine who came from. resultCode int: The result code integer returned by the child's activity through its setResult(). Data Intention: An intention that can return the result data to the subscriber (various data may be attached to additional intentions). Protected void onApplyThemeResource (Theme Resources.Theme, int resid, boolean first) Called setTheme (Resources.Theme) and getTheme () to apply the thematic resource to the current subject of the Topic. Can be redefined to change the default behavior (simple). This method will not be called in multiple threads at the same time. Theme Options Resources.Theme: The theme changes inhabit Int: a style resource applied to the theme of the first boolean: true, if this is the first time the style is applied to the topic of protected void onCreate (Bundle savedInstanceState) Is called when the activity begins. This is where most initializations should go: call setContentView (int) to inflate the user interface of activity, using findViewById (int) for software interaction with widgets in the user interface, calling managed avery (android.net.Uri, java.lang.String, java.lang.String, java.lang.String) to obtain data. You can call finish () from this feature, in which case onDestroy () will be immediately called after onCreate (Bundle) without any rest of life (onStart(), onResume (), onPause(etc.) performing. Derivative classes should trigger before the super class of this method is implemented. If they don't, the exception will be This method should be called from Looper-getMainLooper () of your application. If you override this method, you should call before the superclass is implemented. Options saved Instate Bundle: If the action is re-initiated after previously closed, this kit contains data that it recently provided in onSaveInstanceState (Bundle). Note: Otherwise it is zero. This value can be zero. Added to the API level 8 Deprecated in API level 15 protected by Dialog onCreateDialog (Int ID, Bundle args) This method has been deprecated in API level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Call back to create conversations that are managed (preserved and restored) for you by activity. The default implementation requires onCreateDialog (int) for compatibility. If you're focused on Build.VERSION_CODES. HONEYCOMB or later, consider instead using DialogFragment instead. If you use showDialog (int), the action will trigger up this method for the first time and hang on to it after that. Any dialogue created by this method will be automatically saved and restored to you, including when shown. If you want the action to manage the retention and recovery of conversations for you, you must override this method and handle any identifiers that are transmitted to show Dialog (int). If you want the opportunity to prepare a dialogue before his testimony, override onPrepareDialog (int, android.app.Dialog, android.os.Bundle). Brings Back Dialogue Dialogue. If you return null, the dialogue will not be created. Protect the void on Destroy, do any final clean-up before the action is destroyed. This can happen either because the action is being completed (someone is called a finish), or because the system temporarily destroys that instance of action to save space. You can distinguish between these two scenarios using the isFinishing method. Note: don't expect this method to be called a data saving location! For example, if an action edits data in a content provider, these edits should be made either on The Show or onAveInstanceState (Bundle) and not here. This method is usually implemented to free up resources such as action-related threads, so the shattered action doesn't leave such things around while the rest of its application is still working. There are situations where the system will just kill the hosting activity process without naming this method (or any other) in it, so it should not be used to do things that are designed to stay around after the process goes away. Derivative classes should trigger before the super class of this method is implemented. If they will not come true, an exception will be thrown. If you override this method, you should call before the superclass is implemented. See also: also: Protected Void on NewIntent (Intention Intention) This is called for actions that set launchMode on a singleTop in their package, or if a customer used the Intent-FLAG_ACTIVITY_SINGLE_TOP flag when calling startActivity. In any case, when the action is re-launched at the top of the activity stack instead of the new instance running the action, onNewIntent () will be called to an existing instance with the intention that was used to re-run it. The action can never get a new intention in a renewed state. You can count on the fact that after this method will be called, although not necessarily immediately after the completion of this callback. If the action was resumed, it will be suspended and new intentions will be delivered, and then onResume .. If the action has not been resumed, then a new intention can be delivered immediately, with onResume called some time later when the activity becomes active again. Please note that getIntent still returns the original intent. You can use setIntent (Intention) to update it before this new intention. Options of intent: A new intention that has been launched for the activity. See also: getIntent ()setIntent (Intent)onResume () protected void onPause () Is called as part of the activity lifecycle when the user no longer actively interacts with the action, but it is still visible on the screen. Analogue onResume(). When Action B is launched before Action A, this callback will be called to A. B will not be created until A's onPause returns, so don't do anything long here. This callback is mainly used to maintain any permanent state that edits the action to present the user with an edit on the spot model and to make sure that nothing is lost if there are not enough resources to start a new activity without one killing it. It's also a good place to stop things that consume a noticeable amount of CPU in order to move on to the next action as quickly as possible. On the platform version up to Build.VERSION_CODES. It's also a good place to try to close exclusive access devices or free up access to monochrome resources. Starting with Build.VERSION_CODES. There may be several resumed activities in the system at the same time, so onTopResumedActivityChanged (boolean) should be used for this purpose. If the action is running from above, you usually receive the next call on onStop after the next activity has been resumed and shown above). However, in some cases there will be a direct call back to Resume () does not pass through the stopped state. In some cases, the action may also rest in state when it is in multi-chord mode, still visible to the user. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call in superclass. See also: onResume ()onSaveInstanceState (Bundle)onStop () protected void onPostCreate (Bundle savedInstanceState) Called when the launch of the activity is completed (after onStart() and onRestoreInstanceState (Bundle) have been called). Apps generally don't implement this method; it's designed for system classes to finally initiate after the application code is launched. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call before the superclass is implemented. Options saved Instate Bundle: If the action is re-initiated after previously closed, this kit contains data that it recently provided in onSaveInstanceState (Bundle). Note: Otherwise it is zero. This value can be zero. See also: Protected void onPostResume is called when the resumption of activity is completed (after onResume)) has been called). Apps generally don't implement this method; it's designed for system classes to be finalized after the application summary code is launched. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call before the superclass is implemented. Added to API level 8 Deprecated in API level 15 protected voids onPrepareDialog (Int ID, Dialogue Dialogue, Bundle args) This method has been deprecated in API level 15. Instead, use the new DialogFragment class with FragmentManager; It's also available on older platforms through the Android compatibility package. Provides an opportunity to prepare a controlled dialogue before his testimony. The default implementation requires onPrepareDialog (int, android.app.Dialog) for compatibility. Override this if you need to update a managed conversation based on the state of the app every time it is displayed. For example, a time builder's dialogue might want to be updated with the current time. You have to call before the superclass. The default implementation will set this action as the owner's action in Dialog. protected void onRestart () Called after onStop () when the current action is re-displayed to the user (the user has returned to it). It will be followed by onStart and then onResume. For actions that use untreated Cursor objects (instead of creating them through a controlled avery (android.net.Uri, java.lang.String, java.lang.String), this is usually the place where the cursor should be reburied (because you deactivated it in onStop).). Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call up Superclass. Seeing See onStop ()onStart ()onResume () protected void onResume () Called after onRestoreInstanceState (Bundle), onRestart (), or onPause (), for your activities to start interacting with the user. This is an indicator that the activity has become active and ready to receive input. It is on top of the activity stack and is visible to the user. On the platform version up to Build.VERSION_CODES. It's also a good place to try to open devices with exclusive access or access monochrome resources. Starting with Build.VERSION_CODES. There may be several resumed activities in the system at the same time, so onTopResumedActivityChanged (boolean) should be used for this purpose. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call before the superclass is implemented. The protected void onaveInstanceState (Bundle outState) is called to extract the state in each instance from the action before being killed, so that the state can be restored to onCreate (Bundle) or onRestoreInstanceState (Bundle, inhabited by this method, will be transferred to both). This method is called before the action can be killed, so that when it returns some time in the future, it can regain its condition. For example, if Action B is launched before Action A and at some point Action A is killed to restore resources, Action A will be able to maintain the current state of the user interface with this method so that when the user returns to activity A, the user interface state can be restored via onCreate (Bundle) or onRestoreInstanceState (Bundle). Don't confuse this method with activity lifecycle callbacks such as onPause, which is always called when the user is no longer actively interacting with activity, or onStop, which is called when activity becomes invisible. One example of when onPause () and onStop () is called rather than this method is when the user goes from activity B to activity: there is no need to call onSaveInstanceState (Bundle) on B, because this particular instance will never be restored, so the system avoids calling it. An example where onPause () is called rather than onSaveInstanceState (Bundle) is when Activity B is launched before Operation A: the system can avoid calling onSaveInstanceState (Bundle) to activity A if it has not been killed during the life of B, as the state of the user interface A will remain intact. The default implementation takes care of most of the state of the user interface in each instance for you by calling View.onSaveInstanceState () on each view in the hierarchy that has and retaining the current focused view ID (all of which are restored as a result of the default onRestoreInstanceState (Bundle) implementation. If you override this method to save additional information that is not captured by each individual species, you might want to before realizing the default, otherwise be prepared to keep the entire state of each view yourself. If called, this method will occur after onStop () for application targeting platforms starting at Build.VERSION_CODES.P. For applications focused on earlier versions of the platform, this method will occur before onStop () and there is no guarantee whether it will happen before or after onPause.). OutState Bundle: A set in which to place a saved state. This value cannot be zero. See also: onCreate (Bundle)onRestoreInstanceState (Bundle)onPause () protected void onStart () Called after onCreate (Bundle) - or after onRestart ( ) when the action was stopped, but now again displayed to the user. It is usually followed by onResume. This is a good place to start drawing visuals, running animations, etc. you can trigger a finish () from this function, in which case onStop () will be immediately called after onStart () without life cycle transitions between them (onResume(), onPause () etc.) execution. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call before the superclass is implemented. See also: onCreate (Bundle)onStop ()onResume () protected void onStop () Called when you are no longer visible to the user. You'll then get either onRestart or nothing, depending on the user's later activity. It's a good place to stop the refreshing user interface, running animations and other visual stuff. Derivative classes should trigger before the super class of this method is implemented. If they do not, the exception will be thrown. If you override this method, you should call before the superclass is implemented. See also: onRestart ()onResume ()onSaveInstanceState (Bundle)onDestroy () protected void onTitleChanged (name CharSequence, int color) CharSequence color int protected void onUserLeInavet () Is called as part of the life cycle of activity when the action is about to go into the background mode of the user's choice. For example, when a user presses home, a UserLeaveHint will be called, but when an incoming phone call triggers an activity in the call that will be automatically brought to the foreground, onUserLeaveHint () will not be called for the activity to be interrupted. Where it is called, this method is called right before the onPause action is called. This callback and onUserInteraction are designed to help you manage status bar notifications wisely; specifically, to assist activities determine the correct time to cancel the notice. See onUserInteraction ()Intent.FLAG_ACTIVITY_NO_USER_ACTION onUserInteraction()Intent.FLAG_ACTIVITY_NO_USER_ACTION onUserInteraction()Intent.FLAG_ACTIVITY_NO_USER_ACTION