How can I close/hide the Android soft keyboard programmatically?

Asked 15 years, 5 months ago Modified 2 months ago

Viewed 1.9m times Part of Mobile Development Collective

4365

I have an `EditText` and a `Button` in my layout.

After writing in the edit field and clicking on the `Button`, I want to hide the virtual keyboard when touching outside the keyboard. Can someone provide a simple example of how to achieve this?

MD

android android-edittext android-softkeyboard android-input-method soft-keyboard

Share Follow

edited Jun 3 at 11:43

community wiki
31 revs, 19 users 19%
Vidar Vestnes

19 What if you have only one EditText and several buttons, like check boxes and radios? The only place you need the keyboard is in the single EditText. How do you register to

know that something else was chosen/clicked in order to hide the keyboard? – AlikElzin-kilaka Jun 1, 2011 at 15:48

19 i feel stupid. I am unable to hide the keyboard on ICS. Tried all methods here and combinations of them. No way. The method to show it works, but I cant hide it no matter what windw token, hide flags, manifest settings or candles to any saints. On keyboard show I always see this: I/LatinIME(396): InputType.TYPE_NULL is specified W/LatinIME(396): Unexpected input class: inputType=0x00000000 imeOptions=0x00000000 – rupps May 15, 2013 at 13:28

5 /** * This method is used to hide soft keyboard. * @param activity */ public void hideSoftKeyboard(Activity activity) { InputMethodManager inputMethodManager = (InputMethodManager)activity.getSystemService(Activity.INPUT_METHOD_SERVICE); inputMethodManager.hideSoftInputFromWindow(activity.getCurrentFocus().getWindowToken(), 0); } – Harshal Benake Jan 13, 2014 at 13:30

this worked for me – nmxprime Jun 20, 2014 at 12:45

Need to play with InputMethodManager with the INPUT_METHOD_SERVICE to handle soft keyboard like readyandroid.wordpress.com/show-hide-android-soft-keyboard – Ready Android May 4, 2018 at 6:09

132 Answers

Sorted by: Highest score (default)

You can force Android to hide the virtual keyboard using the InputMethodManager, calling

4717

`hideSoftInputFromWindow`, passing in the token of the window containing your focused view.

```
// Check if no view has focus:
View view = this.getCurrentFocus();
if (view != null) {
    InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_MET
    imm.hideSoftInputFromWindow(view.getWindowToken(),
}
```

This will force the keyboard to be hidden in all situations. In some cases, you will want to pass in `InputMethodManager.HIDE_IMPLICIT_ONLY` as the second parameter to ensure you only hide the keyboard when the user didn't explicitly force it to appear (by holding down the menu).

Note: If you want to do this in Kotlin, use: `context?.getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager`

Kotlin Syntax

```
// Only runs if there is a view that is currently focu
this.currentFocus?.let { view ->
    val imm = getSystemService(Context.INPUT_METHOD_SE
InputMethodManager
    imm?.hideSoftInputFromWindow(view.windowToken, 0)
}
```

Share Follow edited Feb 21, 2021 at 9:44

community wiki
15 revs, 15 users 21%
Reto Meier

2 now `getSystemService()` requires a `Context` and a serviceClass `Class`. For the context I can call `requiredContext` but what about for the serviceClass? – capo11 May 26, 2021 at 8:37

1 @capo11 I tried with Application.Context.getSystemService(), so I didn't need the serviceClass, but it doesn't work – Windgate Sep 8, 2021 at 10:37

1 Works well for me inside a `Fragment` using `getActivity().getSystemService()...` – Captain Jack Sparrow Jun 29, 2022 at 14:36

2584

To help clarify this madness, I'd like to begin by apologizing on behalf of all Android users for Google's downright ridiculous treatment of the soft keyboard. The reason there are so many answers, each different, for the same simple question is that this API, like many others in Android, is horribly designed. I can think of no polite way to state it.

I want to hide the keyboard. I expect to provide Android with the following statement: `Keyboard.hide()`. The end. Thank you very much. But Android has a problem. You must use the `InputMethodManager` to hide the keyboard. OK, fine, this is Android's API to the keyboard. BUT! You are required to have a `Context` in order to get access to the IMM. Now we have a problem. I may want to hide the

keyboard from a static or utility class that has no use or need for any `Context`. or And FAR worse, the IMM requires that you specify what `View` (or even worse, what `Window`) you want to hide the keyboard FROM.

This is what makes hiding the keyboard so challenging. Dear Google: When I'm looking up the recipe for a cake, there is no `RecipeProvider` on Earth that would refuse to provide me with the recipe unless I first answer WHO the cake will be eaten by AND where it will be eaten!!

This sad story ends with the ugly truth: to hide the Android keyboard, you will be required to provide 2 forms of identification: a `Context` and either a `View` or a `Window`.

I have created a static utility method that can do the job VERY solidly, provided you call it from an `Activity`.

```
public static void hideKeyboard(Activity activity) {
    InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVIC
    //Find the currently focused view, so we can grab
from it.
    View view = activity.getCurrentFocus();
    //If no view currently has focus, create a new one
window token from it
    if (view == null) {
        view = new View(activity);
    }
    imm.hideSoftInputFromWindow(view.getWindowToken(),
}
```

Be aware that this utility method ONLY works when called from an `Activity`! The above method calls

`getCurrentFocus` of the target `Activity` to fetch the proper window token.

But suppose you want to hide the keyboard from an `EditText` hosted in a `DialogFragment`? You can't use the method above for that:

```
hideKeyboard(getActivity()); //won't work
```

This won't work because you'll be passing a reference to the `Fragment`'s host `Activity`, which will have no focused control while the `Fragment` is shown! Wow! So, for hiding the keyboard from fragments, I resort to the lower-level, more common, and uglier:

```
public static void hideKeyboardFrom(Context context, V
    InputMethodManager imm = (InputMethodManager)
context.getSystemService(Activity.INPUT_METHOD_SERVICE
    imm.hideSoftInputFromWindow(view.getWindowToken(),
}
```

Below is some additional information gleaned from more time wasted chasing this solution:

About windowSoftInputMode

There's yet another point of contention to be aware of. By default, Android will automatically assign initial focus to the first `EditText` or focusable control in your `Activity`. It naturally follows that the InputMethod (typically the soft keyboard) will respond to the focus event by showing itself. The `windowSoftInputMode` attribute in

`AndroidManifest.xml`, when set to `stateAlwaysHidden`, instructs the keyboard to ignore this automatically-assigned initial focus.

```
<activity
    android:name=".MyActivity"
    android:windowSoftInputMode="stateAlwaysHidden"/>
```

Almost unbelievably, it appears to do nothing to prevent the keyboard from opening when you touch the control (unless `focusable="false"` and/or `focusableInTouchMode="false"` are assigned to the control). Apparently, the windowSoftInputMode setting applies only to automatic focus events, not to focus events triggered by touch events.

Therefore, `stateAlwaysHidden` is VERY poorly named indeed. It should perhaps be called `ignoreInitialFocus` instead.

UPDATE: More ways to get a window token

If there is no focused view (e.g. can happen if you just changed fragments), there are other views that will supply a useful window token.

These are alternatives for the above code `if (view == null)  view = new View(activity);` These don't refer explicitly to your activity.

Inside a fragment class:

```
view = getView().getRootView().getWindowToken();
```

Given a fragment `fragment` as a parameter:

```
view = fragment.getView().getRootView().getWindowToken
```

Starting from your content body:

```
view = findViewById(android.R.id.content).getRootView(
```

UPDATE 2: Clear focus to avoid showing keyboard again if you open the app from the background

Add this line to the end of the method:

`view.clearFocus();`

Share Follow

edited Mar 12, 2021 at 2:41

community wiki
15 revs, 10 users 60%
rmirabelle

1 Why need `getRootView()` , why not `getView()` only? – ilw Dec 15, 2020 at 10:21

5 One liner: `((InputMethodManager)getContext().getSystemService(Activity.INPUT_METHOD_SERVICE)).hideSoftInput`

`FromWindow(getView().getWindowToken(), 0);`
– Martin Braun Apr 9, 2021 at 23:26

1 Recently we finally got an official, backwards compatible way to do this – gmk57 May 21, 2021 at 13:33

1 Finally there is an official way stackoverflow.com/a/67683124/4985958
– Sergey Chilingaryan May 25, 2021 at 6:59

I have one problem with this. The text inside the EditText field stays underlined as if the keyboard's suggestions were still active. – Marcell Jun 10, 2022 at 18:38

856

Also useful for hiding the soft-keyboard is:

```
getWindow().setSoftInputMode(
    WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS
);
```

This can be used to suppress the soft-keyboard until the user actually touches the editText View.

Share Follow

edited Nov 24, 2014 at 16:07

community wiki
5 revs, 4 users 50%
Garnet Ulrich

This was the only one that worked for in 2020. I have a edit text on the main activity and don't want the keyboard to come up when starting the app. – Brian M Dec 13, 2020 at 0:58

Try using InputMethodManager to handle the keyboard. Can try following this article.
androidacademic.blogspot.com/2023/02/…
– Pragnesh Ghoda シ Feb 28, 2023 at 5:24

390

I got one more solution to hide keyboard:

```
InputMethodManager imm = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
imm.toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_O
```

Here pass `HIDE_IMPLICIT_ONLY` at the position of `showFlag` and `0` at the position of `hiddenFlag`. It will forcefully close soft Keyboard.

Share Follow

edited Jun 27, 2016 at 15:26

community wiki
3 revs, 3 users 80%
Saurabh Pareek

6 You're using a hide flag in the showflags parameter. This only works because the constants use the same integers. Example using the correct flags – Alex Mar 23, 2013 at 14:35

38 @Mark: Because the method is called "toggleSoftInput", not "hideSoftInput" :) – Sver Aug 30, 2013 at 2:08

4 This doesn't work correctly. It some times shows the keyboard. – Rohaitas Tanoli Sep 3, 2021 at 11:30

171

Meier's solution works for me too. In my case, the top level of my App is a tab host and I want to hide the

keyword when switching tabs - I get the window token from the tab host View.

```
tabHost.setOnTabChangedListener(new OnTabChangeListene
    public void onTabChanged(String tabId) {
        InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(tabHost.getApplica
    }
}
```

Share Follow

edited Jun 11, 2020 at 4:18

community wiki
3 revs, 3 users 64%
mckoss

167

Please try this below code in `onCreate()`

```
EditText edtView = (EditText) findViewById(R.id.editTe
edtView.setInputType(InputType.TYPE_NULL);
```

Share Follow

edited Mar 23, 2022 at 3:09

community wiki
5 revs, 5 users 48%
Jeyavel

---

3 This method works as a means of getting around the "can't hide the soft keyboard" bug in 2.0 and 2.1 as described in

code.google.com/p/android/issues/detail?id=7115 ... the hideSoftInputFromWindow method listed above did not work when I tried it, but editView.setInputType(0) did.
– Spike Williams Apr 17, 2010 at 5:50

22 This is legit per Javadoc (not a hack) though I would rewrite the method as
`editView.setInputType(InputType.TYPE_NULL);`
– Bostone Oct 11, 2010 at 20:49

3 this works, however, it hides the android:hint. i'm using Android 1.5 – Tirtha Jan 11, 2012 at 10:32

It works, but it's also hiding the cursor. I need the cursor, but no system keyboard. – Stefan Brendle Aug 8, 2016 at 18:51

Second part of the answer was copied from another answer. Revision was rolled back. – General Grievance Mar 23, 2022 at 3:09

158

Update: I don't know why this solution is not work any more (I just tested on Android 23). Please use the solution of Saurabh Pareek instead. Here it is:

```
InputMethodManager imm = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
//Hide:
imm.toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_O
//Show
imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT,
```

Old answer:

```
//Show soft-keyboard:
getWindow().setSoftInputMode(WindowManager.LayoutParam
//hide keyboard :
```

```
getWindow().setSoftInputMode(WindowManager.LayoutParam
```

Share Follow

edited May 23, 2017 at 12:10

community wiki
3 revs
Nguyen Minh Binh

8 Where should i place this code? I've tried to paste getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN); in onCreate() but the keyboard is never hidden – user2236096 Jul 15, 2013 at 18:20

does not work, tested in radioGroup.setOnCheckedChangeListener, API 23 – syr Aug 6, 2015 at 16:50

If you look closer, InputMethodManager.HIDE_IMPLICIT_ONLY and InputMethodManager.SHOW_IMPLICIT have the same value, which is "1", so there is no difference between these calls. => not working – Palejandro Aug 18, 2016 at 10:43

if calling imm.toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_ONLY, 0); then keyboard will show on screen :) Best implementation is: github.com/ravindu1024/android-keyboardlistener Shame on Android SDK – Duna Nov 29, 2016 at 10:15

`I don't know why this solution is not work any more` - because it's ***Android***, everything will be able to change, maybe partly of bad design... We write carelessly,

then we strike out all and rewrite everything. – King King Dec 16, 2016 at 4:22

104

```
protected void hideSoftKeyboard(EditText input) {
    InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow(input.getWindowToken()
}
```

Share Follow

edited Nov 28, 2018 at 10:55

community wiki
5 revs, 5 users 43%
Sreedev R

5 This worked for me! But why did you put **input.setInputType(0)** ? I couldn't interact with the *EditTextView* when I had that line of code (It worked when I removed it). – ymerdrengene Apr 10, 2014 at 13:52

Probably `input.getContext().getSystemService(Context.INPUT_METHOD_SERVICE)` . – CoolMind Sep 19, 2018 at 14:28

I removed `input.setInputType(0);` from this code. It changed a keyboard behaviour and `inputType` for the `EditText` . – CoolMind Nov 28, 2018 at 10:56

If all the other answers here don't work for you as you would like them to, there's another way of manually

88

controlling the keyboard.

Create a function with that will manage some of the `EditText`'s properties:

```
public void setEditTextFocus(boolean isFocused) {
    searchEditText.setCursorVisible(isFocused);
    searchEditText.setFocusable(isFocused);
    searchEditText.setFocusableInTouchMode(isFocused);

    if (isFocused) {
        searchEditText.requestFocus();
    }
}
```

Then, make sure that onFocus of the `EditText` you open/close the keyboard:

```
searchEditText.setOnFocusChangeListener(new OnFocusCha
    @Override
    public void onFocusChange(View v, boolean hasFocus
        if (v == searchEditText) {
            if (hasFocus) {
                // Open keyboard
                ((InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE)
InputMethodManager.SHOW_FORCED);
            } else {
                // Close keyboard
                ((InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE)
0);
            }
        }
    }
});
```

Now, whenever you want to open the keyboard manually call:

```
setEditTextFocus(true);
```

And for closing call:

```
setEditTextFocus(false);
```

Share Follow

edited Nov 24, 2014 at 15:55

community wiki
3 revs, 2 users 81%
Rotemmiz

I got 'Cannot resolve symbol context', on 7th and 10th line of second block of code. – gimmegimme Feb 28, 2017 at 17:02

Use getContext() instead – Rotemmiz Feb 28, 2017 at 17:04

Context context = View.getContext(); – AgungCode.Com Mar 10, 2022 at 2:22

73

Saurabh Pareek has the best answer so far.

Might as well use the correct flags, though.

```
/* hide keyboard */
((InputMethodManager) getSystemService(Activity.INPUT_
    .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT,
```

```
/* show keyboard */
((InputMethodManager) getSystemService(Activity.INPUT_
    .toggleSoftInput(0, InputMethodManager.HIDE_IMPLIC
```

Example of real use

```
/* click button */
public void onClick(View view) {
  /* hide keyboard */
  ((InputMethodManager) getSystemService(Activity.INPU
      .toggleSoftInput(InputMethodManager.SHOW_IMPLICI

  /* start loader to check parameters ... */
}

/* loader finished */
public void onLoadFinished(Loader<Object> loader, Obje
    /* parameters not valid ... */

    /* show keyboard */
    ((InputMethodManager) getSystemService(Activity.IN
        .toggleSoftInput(0, InputMethodManager.HIDE_IM

    /* parameters valid ... */
}
```

Share Follow

edited May 23, 2017 at 11:55

community wiki
2 revs
Alex

---

from so searching, here I found an answer that works for me

66

```
// Show soft-keyboard:
InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
imm.toggleSoftInput(InputMethodManager.SHOW_FORCED, 0)

// Hide soft-keyboard:
getWindow().setSoftInputMode(WindowManager.LayoutParam
```

Share Follow

edited Apr 24, 2015 at 21:09

community wiki
2 revs, 2 users 80%
shontauro

64

The short answer

In your `OnClick` listener call the `onEditorAction` of the `EditText` with `IME_ACTION_DONE`

```
button.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        someEditText.onEditorAction(EditorInfo.IME_ACT
    }
});
```

The drill-down

I feel this method is better, simpler and more aligned with Android's design pattern. In the simple example above

(and usually in most of the common cases) you'll have an `EditText` that has/had focus and it also usually was the one to invoke the keyboard in the first place (it is definitely able to invoke it in many common scenarios). In that same way, **it** should be the one to release the keyboard, usually that can be done by an `ImeAction`. Just see how an `EditText` with `android:imeOptions="actionDone"` behaves, you want to achieve the same behavior by the same means.

Check this related answer

Share Follow

edited May 23, 2017 at 12:10

community wiki
3 revs
Alex.F

51

Thank God it's officially supported after 11 years.

First add dependency `implementation 'androidx.core:core-ktx:1.7.0'` to app gradle.

Then get InsetsController from ViewCompat or WindowCompat class.

Finally use hide() and show() function of InsetsController

Add support for Dialog. Available in BottomSheetDialog. @Rondev. Using a safer way to get activity instead of

directly cast from context.

```
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.ContextWrapper
import android.view.View
import androidx.core.view.ViewCompat
import androidx.core.view.WindowCompat
import androidx.core.view.WindowInsetsCompat
import androidx.fragment.app.Fragment

fun View.showKeyboard() =
ViewCompat.getWindowInsetsController(this)?.show(Windo
fun View.hideKeyboard() =
ViewCompat.getWindowInsetsController(this)?.hide(Windo

fun Dialog.showKeyboard() = window?.decorView?.showKey
fun Dialog.hideKeyboard() = window?.decorView?.hideKey

fun Context.showKeyboard() = getActivity()?.showKeyboa
fun Context.hideKeyboard() = getActivity()?.hideKeyboa

fun Fragment.showKeyboard() = activity?.showKeyboard()
fun Fragment.hideKeyboard() = activity?.hideKeyboard()

fun Activity.showKeyboard() = WindowCompat.getInsetsCo
window.decorView)?.show(WindowInsetsCompat.Type.ime())
fun Activity.hideKeyboard() = WindowCompat.getInsetsCo
window.decorView)?.hide(WindowInsetsCompat.Type.ime())

fun Context.getActivity(): Activity? {
    return when (this) {
        is Activity -> this
        is ContextWrapper -> this.baseContext.getActiv
        else -> null
    }
}
```

Old anwser below

Here is the simple [project] on github

```
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.ContextWrapper
import android.view.View
import androidx.core.view.ViewCompat
import androidx.core.view.WindowCompat
import androidx.core.view.WindowInsetsCompat
import androidx.fragment.app.Fragment

fun View.showKeyboard() =
ViewCompat.getWindowInsetsController(this)?.show(Windo
fun View.hideKeyboard() =
ViewCompat.getWindowInsetsController(this)?.hide(Windo

fun Dialog.showKeyboard() = window?.decorView?.showKey
fun Dialog.hideKeyboard() = window?.decorView?.hideKey

fun Context.showKeyboard() = getActivity()?.showKeyboa
fun Context.hideKeyboard() = getActivity()?.hideKeyboa

fun Fragment.showKeyboard() = activity?.showKeyboard()
fun Fragment.hideKeyboard() = activity?.hideKeyboard()

fun Activity.showKeyboard() = WindowCompat.getInsetsCo
window.decorView)?.show(WindowInsetsCompat.Type.ime())
fun Activity.hideKeyboard() = WindowCompat.getInsetsCo
window.decorView)?.hide(WindowInsetsCompat.Type.ime())

fun Context.getActivity(): Activity? {
    return when (this) {
        is Activity -> this
        is ContextWrapper -> this.baseContext.getActiv
        else -> null
    }
}
```

Share Follow

edited Oct 18, 2023 at 15:21

community wiki
7 revs, 4 users 58%
Sergey Chilingaryan

3 Requires API level 31 – Johann Nov 22, 2021 at 18:34

@Johann For compatibility, use WindowCompat and WindowInsetsControllerCompat. You'll need to upgrade your gradle dependency for androidx.core to at least 1.6.0-alpha03 so that there will be support on SDK < 30. – Sergey Chilingaryan Nov 23, 2021 at 7:44

@Johann take a look at the sample project github.com/sergchil/KeyboardTest – Sergey Chilingaryan Dec 9, 2021 at 15:01

2 Unfortunately I couldn't get the hide function to work inside a `BottomSheetDialogFragment` – Rondev Jan 23, 2022 at 13:46

3 getWindowInsetsController is deprecated – Muhammad Babar Feb 16, 2023 at 13:40

50

This should work:

```
public class KeyBoard {

    public static void show(Activity activity){
        InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVIC
        imm.toggleSoftInput(0, InputMethodManager.HIDE
    }

    public static void hide(Activity activity){
        InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVIC
```

```
        imm.toggleSoftInput(InputMethodManager.HIDE_IM
    }

    public static void toggle(Activity activity){
        InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVIC
        if (imm.isActive()){
            hide(activity);
        } else {
            show(activity);
        }
    }
}

KeyBoard.toggle(activity);
```

Share Follow

edited Jul 15, 2016 at 14:51

community wiki
3 revs
slinden77

@YoushaAleayoub yes it will.
`KeyBoard.toggle(fragment.getActivity())`
– slinden77 Jul 15, 2016 at 14:46

@slinden77, lol, I'm talking about your Answer... not this one you have commented. So that answer still WONT work.
– Yousha Aleayoub Jul 16, 2016 at 16:55

@YoushaAleayoub uhm yes it will. The original question doesn't mention fragments, you are the one who mentioned fragments. So my answer is perfectly valid. To use it with fragments, call the method differently from a `Fragment`, like a commented. Learn how to use methods please and then come back. You're confusing people with your silly replies
– slinden77 Jul 17, 2016 at 16:48

49

I'm using a custom keyboard to input an Hex number so I can't have the IMM keyboard show up...

In v3.2.4_r1 `setSoftInputShownOnFocus(boolean show)` was added to control weather or not to display the keyboard when a TextView gets focus, but its still hidden so reflection must be used:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEY
    try {
        Method method = TextView.class.getMethod("setS
boolean.class);
        method.invoke(mEditText, false);
    } catch (Exception e) {
        // Fallback to the second method
    }
}
```

For older versions, I got very good results (but far from perfect) with a `OnGlobalLayoutListener`, added with the aid of a `ViewTreeObserver` from my root view and then checking if the keyboard is shown like this:

```
@Override
public void onGlobalLayout() {
    Configuration config = getResources().getConfigura

    // Dont allow the default keyboard to show up
    if (config.keyboardHidden != Configuration.KEYBOAR
        InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(mRootView.getWindo
    }
}
```

This last solution may show the keyboard for a split second and messes with the selection handles.

When in the keyboard enters full screen, onGlobalLayout isn't called. To avoid that, use TextView#setImeOptions(int) or in the TextView XML declaration:

```
android:imeOptions="actionNone|actionUnspecified|flagN
```

Update: Just found what dialogs use to never show the keyboard and works in all versions:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_A
        WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_
```

Share Follow

edited May 28, 2012 at 20:48

community wiki
sergio91pt

Cool solution, however, if your front activity is not fullscreen, the keyboard is visible behind it. Also the keyboard's cursor movement aid is also still visible. And it's not skinnable.
– halxinate Mar 26, 2013 at 17:42

I second that. Of all the possible ways only the getWindow().setFlags() method works, at least on stock Android 5.1. Note that setSoftInputShownOnFocus() is now setShowSoftInputOnFocus() and no longer hidden but does not work, at least not when the user touches the field.
– olefevre Dec 20, 2015 at 19:04

38

```
public void setKeyboardVisibility(boolean show) {
    InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
    if(show){
        imm.toggleSoftInput(InputMethodManager.SHOW_FO
    }else{
        imm.hideSoftInputFromWindow(getCurrentFocus().
    }
}
```

Share Follow

answered Jun 1, 2015 at 13:36

community wiki
shobhan

37

Now, almost 12 years later, we finally have an official, backwards compatible way to do this with AndroidX Core 1.5+:

```
fun View.hideKeyboard() = ViewCompat.getWindowInsetsCo
    ?.hide(WindowInsetsCompat.Type.ime())
```

or specifically for Fragment:

```
fun Fragment.hideKeyboard() =
ViewCompat.getWindowInsetsController(requireView())
    ?.hide(WindowInsetsCompat.Type.ime())
```

Share Follow

edited Sep 21, 2022 at 15:05

community wiki
3 revs, 2 users 67%
gmk57

8 Well would you look at that! Of course, it had to involve yet another API (window insets controller vs input method manager), but hey, at least the word 'hide' is there. – rmirabelle May 21, 2021 at 16:25

1 If you want to hide the keyboard but only have a reference to the Activity, which view should you use? `window.decorView` ? Before, you would get the windowToken with `currentFocus?.windowToken` and then use `InputMethodManager.hideSoftInputFromWindow(windowToken, 0)` – Mark Aug 7, 2021 at 16:45

1 @Mark, good question! Testing shows that `window.decorView` does not work on API 25-29: `ViewCompat.getWindowInsetsController()` returns `null` . `currentFocus` has a similar problem on API 30. But you can use any view in your layout, e.g. its root view. For hiding keyboard it works, but for showing keyboard you might be better off with some `EditText` : `WindowInsetsControllerCompat.show()` uses it to request focus. – gmk57 Aug 22, 2021 at 22:30

Thanks @gmk57 So how about using `currentFocus` pre-30, and `window.decorView` 30+? – Mark Aug 25, 2021 at 11:34

1 @gmk57 good to know thanks. So it sounds like it would be better to use a View in the activity's View hierarchy (usually I have access to `SomeBinding.root`). From a `Fragment` , just use `view?.hideKeyboard()` . However, I'm not seeing any benefit over using `context.getSystemService<InputMethodManager>`

```
()?.hideSoftInputFromWindow(windowToken, 0)
```
– Mark Aug 26, 2021 at 4:52

33

I have spent more than two days working through all of the solutions posted in the thread and have found them lacking in one way or another. My exact requirement is to have a button that will with 100% reliability show or hide the on screen keyboard. When the keyboard is in its hidden state is should not re-appear, no matter what input fields the user clicks on. When it is in its visible state the keyboard should not disappear no matter what buttons the user clicks. This needs to work on Android 2.2+ all the way up to the latest devices.

You can see a working implementation of this in my app clean RPN.

After testing many of the suggested answers on a number of different phones (including froyo and gingerbread devices) it became apparent that android apps can reliably:

1. Temporarily hide the keyboard. It will re-appear again when a user focuses a new text field.
2. Show the keyboard when an activity starts and set a flag on the activity indicating that they keyboard should always be visible. This flag can only be set when an activity is initialising.
3. Mark an activity to never show or allow the use of the keyboard. This flag can only be set when an activity

is initialising.

For me, temporarily hiding the keyboard is not enough. On some devices it will re-appear as soon as a new text field is focused. As my app uses multiple text fields on one page, focusing a new text field will cause the hidden keyboard to pop back up again.

Unfortunately item 2 and 3 on the list only work reliability when an activity is being started. Once the activity has become visible you cannot permanently hide or show the keyboard. The trick is to actually restart your activity when the user presses the keyboard toggle button. In my app when the user presses on the toggle keyboard button, the following code runs:

```
private void toggleKeyboard(){

    if(keypadPager.getVisibility() == View.VISIBLE){
        Intent i = new Intent(this, MainActivity.class
        i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
        Bundle state = new Bundle();
        onSaveInstanceState(state);
        state.putBoolean(SHOW_KEYBOARD, true);
        i.putExtras(state);

        startActivity(i);
    }
    else{
        Intent i = new Intent(this, MainActivity.class
        i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
        Bundle state = new Bundle();
        onSaveInstanceState(state);
        state.putBoolean(SHOW_KEYBOARD, false);
        i.putExtras(state);

        startActivity(i);
```

```
    }
}
```

This causes the current activity to have its state saved into a Bundle, and then the activity is started, passing through an boolean which indicates if the keyboard should be shown or hidden.

Inside the onCreate method the following code is run:

```
if(bundle.getBoolean(SHOW_KEYBOARD)){
    ((InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE)).showSo
    getWindow().setSoftInputMode(LayoutParams.SOFT_INP
}
else{
    getWindow().setFlags(WindowManager.LayoutParams.FL
            WindowManager.LayoutParams.FLAG_ALT_FOCUSA
}
```

If the soft keyboard should be shown, then the InputMethodManager is told to show the keyboard and the window is instructed to make the soft input always visible. If the soft keyboard should be hidden then the WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM is set.

This approach works reliably on all devices I have tested on - from a 4 year old HTC phone running android 2.2 up to a nexus 7 running 4.2.2. The only disadvantage with this approach is you need to be careful with handling the back button. As my app essentially only has one screen (its a calculator) I can override onBackPressed() and return to the devices home screen.

Share Follow

edited Mar 6, 2018 at 7:10

community wiki
2 revs, 2 users 93%
Luke Sleeman

2 elaborate workaround, but i think it's just too much , to recreate thousands of objects just to hide the Keyboard. I dont know who designed the IMM for android, but it smells like a Windows APi. In my opinion, a good IME should have two methods: hide and show :-) – rupps May 15, 2013 at 13:43

1 Its all true, but my workaround does have one advantage - it always works! There is no other solution I could find that would always toggle the keyboard, regardless of of what fields in the UI have the focus, what the user has done to toggle and keyboard and what version of android they are running :-\ – Luke Sleeman May 20, 2013 at 11:35

Man, I'm totally desperate to hide the keyboard. Tried thousands of things and noooone works. But your workaround is too much for me, I'd have to recreate like 10 fragments, initialize services, delete a lot of WeakReferences .... you know? the GC would just throw away like 25mb :S ... Still looking for a reliable way to do it :(– rupps May 20, 2013 at 14:57

@Dmitry well it's not a hello world...it's a complex application for tablets. I refuse to totally unload it from memory just to hide a silly keyboard... Anyway I found something that works combining the thousand solutions proposed here :) – rupps Oct 8, 2013 at 16:21

31

Alternatively to this all around solution, if you wanted to close the soft keyboard **from anywhere** without having a reference to the (EditText) field that was used to open the keyboard, but still wanted to do it if the field was focused, you could use this (from an Activity):

```
if (getCurrentFocus() != null) {
    InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow(getCurrentFocus().getW
}
```

Share Follow

edited May 23, 2017 at 12:10

community wiki
4 revs, 2 users 82%
Saran

26

Thanks to this SO answer, I derived the following which, in my case, works nicely when scrolling through the the fragments of a ViewPager...

```
private void hideKeyboard() {
    // Check if no view has focus:
    View view = this.getCurrentFocus();
    if (view != null) {
        InputMethodManager inputManager = (InputMethod
this.getSystemService(Context.INPUT_METHOD_SERVICE);
        inputManager.hideSoftInputFromWindow(view.getW
InputMethodManager.HIDE_NOT_ALWAYS);
    }
}
```

```
private void showKeyboard() {
    // Check if no view has focus:
    View view = this.getCurrentFocus();
    if (view != null) {
        InputMethodManager inputManager = (InputMethod
this.getSystemService(Context.INPUT_METHOD_SERVICE);
        inputManager.showSoftInput(view, InputMethodMa
    }
}
```

Share Follow

edited May 23, 2017 at 12:34

community wiki
2 revs
ban-geoengineering

22

Above answers work for different scenario's but **If you want to hide the keyboard inside a view and struggling to get the right context try this:**

```
setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        hideSoftKeyBoardOnTabClicked(v);
    }
}

private void hideSoftKeyBoardOnTabClicked(View v) {
    if (v != null && context != null) {
        InputMethodManager imm = (InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE)
        imm.hideSoftInputFromWindow(v.getApplicationWi
InputMethodManager.HIDE_NOT_ALWAYS);
    }
}
```

and to get the context fetch it from constructor:)

```
public View/RelativeLayout/so and so (Context context,
defStyle) {
    super(context, attrs, defStyle);
    this.context = context;
    init();
}
```

Share Follow

edited Apr 24, 2015 at 21:10

community wiki
2 revs, 2 users 76%
Ash

21

If you want to close the soft keyboard during a unit or functional test, you can do so by clicking the "back button" from your test:

```
// Close the soft keyboard from a Test
getInstrumentation().sendKeyDownUpSync(KeyEvent.KEYCOD
```

I put "back button" in quotes, since the above doesn't trigger the `onBackPressed()` for the Activity in question. It just closes the keyboard.

Make sure to pause for a little while before moving on, since it takes a little while to close the back button, so subsequent clicks to Views, etc., won't be registered until after a short pause (1 second is long enough ime).

Share Follow

edited Dec 13, 2011 at 21:55

community wiki
Peter Ajtai

18

Here's how you do it in Mono for Android (AKA MonoDroid)

```
InputMethodManager imm = GetSystemService (Context.Inp
InputMethodManager;
if (imm != null)
    imm.HideSoftInputFromWindow (searchbox.WindowToken
```

Share Follow

answered May 15, 2012 at 1:36

community wiki
Ian Vink

1 What is `searchbox` in the snippet? – PCoder Nov 26, 2012 at 19:14

17

This worked for me for all the bizarre keyboard behavior

```
private boolean isKeyboardVisible() {
    Rect r = new Rect();
    //r will be populated with the coordinates of your
visible.
    mRootView.getWindowVisibleDisplayFrame(r);
```

```
    int heightDiff = mRootView.getRootView().getHeight
    return heightDiff > 100; // if more than 100 pixel
keyboard...
}

protected void showKeyboard() {
    if (isKeyboardVisible())
        return;
    InputMethodManager inputMethodManager = (InputMeth
getSystemService(Context.INPUT_METHOD_SERVICE);
    if (getCurrentFocus() == null) {
        inputMethodManager.toggleSoftInput(InputMethod
    } else {
        View view = getCurrentFocus();
        inputMethodManager.showSoftInput(view, InputMe
    }
}

protected void hideKeyboard() {
    if (!isKeyboardVisible())
        return;
    InputMethodManager inputMethodManager = (InputMeth
getSystemService(Context.INPUT_METHOD_SERVICE);
    View view = getCurrentFocus();
    if (view == null) {
        if (inputMethodManager.isAcceptingText())

inputMethodManager.toggleSoftInput(InputMethodManager.
    } else {
        if (view instanceof EditText)
            ((EditText) view).setText(((EditText) view
// reset edit text bug on some keyboards bug
        inputMethodManager.hideSoftInputFromInputMetho
InputMethodManager.HIDE_NOT_ALWAYS);
    }
}
```

Share Follow

edited Apr 24, 2015 at 21:11

community wiki
4 revs, 3 users 89%
Pinhassi

What is mRootView? – justdan0227 Nov 28, 2018 at 19:49

16

Simple and Easy to use method, just call **hideKeyboardFrom(YourActivity.this);** to hide keyboard

```
/**
 * This method is used to hide keyboard
 * @param activity
 */
public static void hideKeyboardFrom(Activity activity)
    InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Activity.INPUT_METHOD_SERVIC
    imm.hideSoftInputFromWindow(activity.getCurrentFoc
0);
}
```

Share Follow

answered Feb 21, 2017 at 6:52

community wiki
Naveed Ahmad

You didn't check whether `activity.getCurrentFocus()` was null, it could well be if the keyboard was actually not visible – Adam Burley Jun 7, 2022 at 13:06

16

Kotlin version via an extension function

Using Kotlin extension functions, it'd be so simple to show and hide the soft keyboard.

ExtensionFunctions.kt

```
import android.app.Activity
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.EditText
import androidx.fragment.app.Fragment

fun Activity.hideKeyboard(): Boolean {
    return (getSystemService(Activity.INPUT_METHOD_SER
InputMethodManager)
        .hideSoftInputFromWindow((currentFocus ?: View
}

fun Fragment.hideKeyboard(): Boolean {
    return (context?.getSystemService(Activity.INPUT_M
InputMethodManager)
        .hideSoftInputFromWindow((activity?.currentFoc
View(context)).windowToken, 0)
}

fun EditText.hideKeyboard(): Boolean {
    return (context.getSystemService(Activity.INPUT_ME
InputMethodManager)
        .hideSoftInputFromWindow(windowToken, 0)
}

fun EditText.showKeyboard(): Boolean {
    return (context.getSystemService(Activity.INPUT_ME
InputMethodManager)
        .showSoftInput(this, 0)
}
```

Usage

Now in your `Activity` or `Fragment`, `hideKeyboard()` is clearly accessible as well as calling it from an instance of `EditText` like:

```
editText.hideKeyboard()
```

Share Follow

edited Oct 18, 2023 at 15:07

community wiki
2 revs, 2 users 89%
aminography

15

Just use this optimized code in your activity:

```
if (this.getCurrentFocus() != null) {
    InputMethodManager inputManager = (InputMethodMana
this.getSystemService(Context.INPUT_METHOD_SERVICE);

inputManager.hideSoftInputFromWindow(this.getCurrentFo
InputMethodManager.HIDE_NOT_ALWAYS);
}
```

Share Follow

edited Apr 24, 2015 at 21:11

community wiki
2 revs, 2 users 75%
Hamid FzM

15

Add to your activity `android:windowSoftInputMode="stateHidden"` in Manifest file. Example:

```
<activity
            android:name=".ui.activity.MainActivity"
            android:label="@string/mainactivity"
            android:windowSoftInputMode="stateHidden"/
```

Share Follow

answered Feb 4, 2016 at 12:21

community wiki
NickUnuchek

14

For Open Keyboard :

```
InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
imm.showSoftInput(edtView, InputMethodManager.SHOW_IMP
```

For Close/Hide Keyboard :

```
 InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_MET
 imm.hideSoftInputFromWindow(edtView.getWindowToken(),
```

Share Follow

answered Feb 5, 2015 at 12:39

community wiki
Girish Patel

14

I have the case, where my `EditText` can be located also in an `AlertDialog`, so the keyboard should be closed on dismiss. The following code seems to be working anywhere:

```
public static void hideKeyboard( Activity activity ) {
    InputMethodManager imm = (InputMethodManager)activ
Context.INPUT_METHOD_SERVICE );
    View f = activity.getCurrentFocus();
    if( null != f && null != f.getWindowToken() &&
EditText.class.isAssignableFrom( f.getClass() ) )
        imm.hideSoftInputFromWindow( f.getWindowToken(
    else
        activity.getWindow().setSoftInputMode(
WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HID
}
```

Share Follow

edited Apr 24, 2015 at 21:12

community wiki
3 revs, 2 users 78%
injecteer

1 This solution is better because you have not to control which EditText pass as a parameter to hideSoftInputFromWindow() method. It works great!! – Billyjoker Feb 12, 2015 at 8:20