Differences between TextWatcher 's onTextChanged, beforeTextChanged and afterTextChanged

Asked 7 years, 8 months ago Active 6 months ago Viewed 40k times

49

13

In my Android project, I have had to add a **TextChangedListener** (TextWatcher) to an edit text view. And there are three parts to it:

- `onTextChanged()`
- `beforeTextChanged()`
- `afterTextChanged()`

What are the differences of these three? I have had to implement a search of a table on the key listener and for my case all these three looked the same. Also they functioned the same. When I input a part of a product name, the table redraws with only those products that contain entered text in it. But I used the `afterTextChanged()` part. My code is:

```
EditProduct.addTextChangedListener(new TextWatcher() {

        @Override
        public void onTextChanged(CharSequence s, int start, int before,
                int count) {
            // TODO Auto-generated method stub

            // System.out.println("onTextChanged"+s);
        }

        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
                int after) {
            // TODO Auto-generated method stub
            // System.out.println("beforeTextChanged"+s);
        }

        @Override
        public void afterTextChanged(Editable s) {
            // TODO Auto-generated method stub
            // System.out.println("afterTextChanged"+s);

            String new_prx = s.toString();

            System.out.println(s);
            mini_productList = new ArrayList<Product>();

            // mini_productList
            int count = 0;
            if (new_prx.equals("")) {
```

```
                    count++;

                }
            }

            loadtableProducts(mini_productList);
        }
    }
});
```

So can someone give me an explanation on these three?

android textwatcher android-textwatcher

Share Improve this question

Follow

edited Aug 25 '19 at 22:05

0xCursor

2,224 4 13 30

asked Nov 29 '13 at 4:28

Samantha Withanage

3,491 10 26 56

1 You might find this thread helpful stackoverflow.com/questions/476848/... – Abhishek V Nov 29 '13 at 4:45

5 Answers

Active | Oldest | Votes

34

`onTextChanged` runs during the text changing.

`afterTextChanged` runs immediately after the text is changed.

`beforeTextChanged` runs the instant before the text is changed.

Depending on when you want to assign variables or do things, you may want to run the code the instant before the change, or the instant after.

Here is an example of this:

```
String afterTextChanged = "";
String beforeTextChanged = "";
String onTextChanged = "";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    et = (EditText)findViewById(R.id.editText);
```

```
        public void beforeTextChanged(CharSequence s, int st, int c, int a)
        {
            beforeTextChanged = et.getText().toString();
        }

        @Override
        public void afterTextChanged(Editable s)
        {
            afterTextChanged = et.getText().toString();
            Toast.makeText(Activity.this, "before: " + beforeTextChanged
                                        + '\n' + "on: " + onTextChanged
                                        + '\n' + "after: " + afterTextChanged
                        ,Toast.LENGTH_SHORT).show();
        }
    });
}
```

In this case, let's say you changed the text from "h" to "hi", the output would be:

> before: "h"
> on: "hi"
> after: "hi"

Share Improve this answer Follow edited Jul 17 '14 at 22:08

answered Nov 29 '13 at 5:04

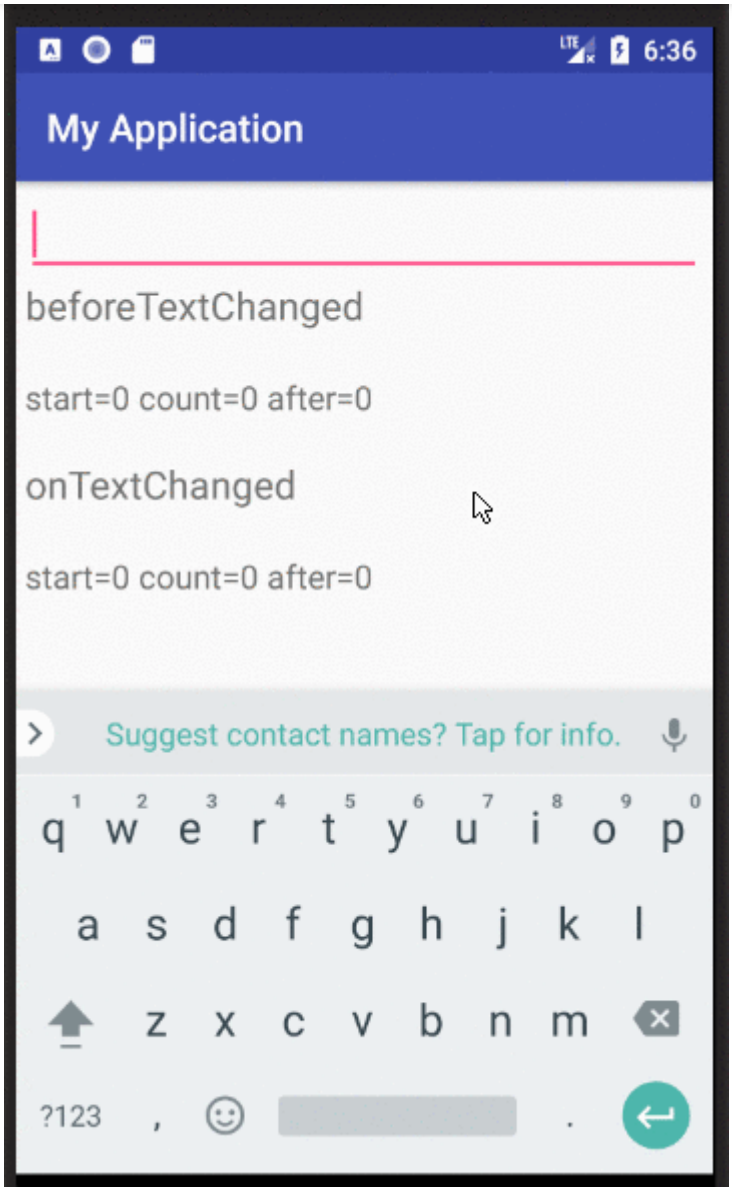
Michael Yaworski
12.5k 17 60 91

7 So, What's different between onTextChanged and afterTextChanged ? – krosshj Nov 17 '18 at 9:11

78

The parameters for `beforeTextChanged` and `onTextChanged` are a little hard to understand at first. It may be helpful to see them being used in an example. Watch the following demonstration a few times. Pay attention to the counts.

- The **red** highlight is the old text that is about to be replaced by the green text.
- The **green** highlight is the new text that just replaced the red text.



beforeTextChanged

- `start` is the start index of the red highlighted text (that is about to be deleted)
- `count` is the length of the **red** highlighted text (that is about to be deleted)
- `after` is the length of the **green** highlighted text (that is about to be added)

onTextChanged

- `start` is the start index of the green highlighted text (that just got added). This is the same as the `start` of `beforeTextChanged`.
- `before` is the length of the **red** highlighted text (that just got deleted).

- `editable` is the editable text from the EditText. You are allowed to change it here. Doing so will trigger all the `TextWatcher` events again.
- You are not given any information about what was changed. If you want to know, you can set a span in `onTextChanged` and then look up the span here.

When to use which?

If you want to observe the changes being made, use `beforeTextChanged()` or `onTextChanged()`. You are not allowed to change the `CharSequence` text in either of these methods, though.

If you want to further modify the text after it was changed, do it in `afterTextChanged()`.

Code

Here is the code if you want to play around with it yourself.

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    final static int RED_COLOR = Color.parseColor("#fb7373");
    final static int GREEN_COLOR = Color.parseColor("#40de83");

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText editText = findViewById(R.id.editText);
        final TextView tvBeforeText = findViewById(R.id.tvBeforeText);
        final TextView tvBeforeNumbers = findViewById(R.id.tvBeforeNumbers);
        final TextView tvAfterText = findViewById(R.id.tvAfterText);
        final TextView tvAfterNumbers = findViewById(R.id.tvAfterNumbers);

        editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
                SpannableString spannableString = new SpannableString(s);
                BackgroundColorSpan backgroundSpan = new
BackgroundColorSpan(RED_COLOR);
                spannableString.setSpan(backgroundSpan, start, start + count,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                tvBeforeText.setText(spannableString);
                tvBeforeNumbers.setText("start=" + start + "  count=" + count + "
after=" + after);
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count)
{
```

```
            @Override
            public void afterTextChanged(Editable s) {
                Log.i("TAG", "afterTextChanged: " + s);
            }
        });
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="beforeTextChanged" />

    <TextView
        android:id="@+id/tvBeforeText"
        android:textSize="17sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/tvBeforeNumbers"
        android:textSize="17sp"
        android:text="start=0 count=0 after=0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_marginTop="20dp"
        android:text="onTextChanged" />

    <TextView
        android:id="@+id/tvAfterText"
        android:textSize="17sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/tvAfterNumbers"
```

377k 245 1179 1224

24

Android `TextChangedListener` is one kind of trigger which is called on text change of an input field.

`TextChangedListener` has three events.

1.beforeTextChanged : This means that the characters are about to be replaced with some new text. The text is uneditable. This event is used when you need to take a look at the old text which is about to change.

2.onTextChanged: Changes have been made, some characters have just been replaced. The text is uneditable. This event is used when you need to see which characters in the text are new.

3.afterTextChanged : The same as above, except now the text is editable. This event is used when you need to see and possibly edit new text.

Share Improve this answer Follow

edited Nov 13 '17 at 1:01

Suragch
377k 245 1179 1224

answered Nov 29 '13 at 4:49

Jigar Pandya
2,039 2 13 26

what if I need to concern on changing of the text, I mean after the change. I used the same code snippet I mentioned up in afterTextChanged on each of others. It gave me the same output. – Samantha Withanage Nov 29 '13 at 5:32

I didnt got what you are trying to say! ontextchanged and aftertextchange will give same output – Jigar Pandya Nov 29 '13 at 6:04

2 When you say the text is editable/uneditable, what does "text" mean? – Jonas Sep 18 '15 at 4:46

@Jonas, it means the parameter i.e.(s) on afterTextChanged can be edited or changed to something else; while it cannot be edited or changed on the other two events(onTextChanged and beforeTextChanged)....rather, on the other two, it just informs you that something withing that "s" has changed. It also shows how much changed, when change started and ended. – pasignature Jul 7 '20 at 14:56

1

- `abstract void afterTextChanged(Editable s)`

This method is called to notify you that, somewhere within s, the text has been changed

- `abstract void onTextChanged(CharSequence s, int start, int before, int count)`

This method is called to notify you that, within s, the count characters beginning at start have just replaced old text that had length before.

You can more learn here.

Share Improve this answer Follow

edited Aug 2 '16 at 13:39

Jonik
75k 66 249 357

answered Nov 29 '13 at 4:47

Kailash Dabhi
3,359 1 25 46

0

1. **afterTextChanged** (Editable s) - This method is called when the text has been changed. Because any changes you make will cause this method to be called again recursively, you have to be watchful about performing operations here, otherwise it might lead to infinite loop.
2. **beforeTextChanged** (CharSequence s, int start, int count, int after) - This method is called to notify you that, within s, the count characters beginning at start are about to be replaced by new text with length after. It is an error to attempt to make changes to s from this callback.
3. **onTextChanged** (CharSequence s, int start, int before, int count) - This method is called to notify you that, within s, the count characters beginning at start have just replaced old text that had length before. It is an error to attempt to make changes to s from this callback.

Share Improve this answer Follow

edited Aug 2 '16 at 13:39

Jonik
75k 66 249 357

answered Nov 16 '15 at 10:00
user2666607