EditText — поле для ввода текста. Так же как и Button, является наследником TextView.

Давайте добавим EditText в наш проект и посмотрим, что получилось.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center\_horizontal"

android:orientation="vertical"

tools:context="com.test.lessons.MainActivity">

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello"

android:textSize="28sp" />

<EditText

android:id="@+id/name\_et"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<Button

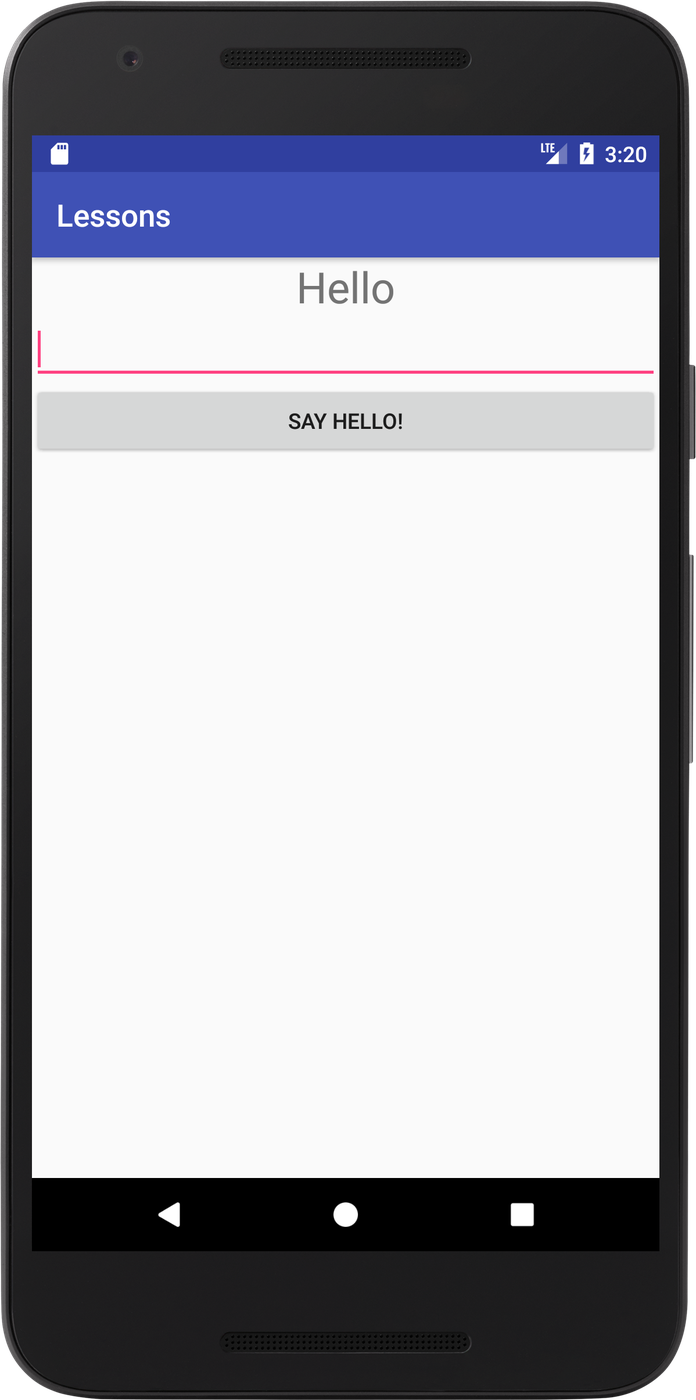
android:id="@+id/hello\_btn"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Say hello!" />

</LinearLayout>

[](https://skillberg.com/media/uploads/2017/11/17/ff547a46be174a379feac2409f456cb9.png)EditText в Android

Типы ввода

По-умолчанию, EditText принимает любой текст, введенный пользователем. Однако, представьте, что вам нужно получить от пользователя номер телефона, адрес e-mail, или пароль.

В первом случае будет логичным показать цифровую клавиатуру, во втором — клавиатуру, оптимизированную для ввода адреса, а в третьем — скрыть вводимые данные от чужих глаз.

В этом нам поможет атрибут android:inputType. Этот же атрибут есть у TextView. Он поддерживает следующие значения:

* none — поле недоступно для редактирования.
* date — дата.
* datetime — дата и время.
* time — время.
* number — целые числа.
* numberDecimal — числа с опциональной частью после запятой (например, 13,37). Может комбинироваться с number.
* numberPassword — целые числа, при этом введенные данные будут скрыты.
* numberSigned — целые числа со знаком. Может комбинироваться с number.
* phone — номер телефона.
* text — обычный текст.
* textAutoComplete — текст с возможностью автодополнения (автодополнение мы должны будем реализовать сами) . Может комбинироваться с text.
* textAutoCorrect — текст с автоисправлением. Может комбинироваться с text.
* textCapCharacters — текст с автоматическим переводом символов в верхний регистр. Может комбинироваться с text.
* textCapSentenses — текст с автоматическим переводом в верхний регистр первого символа каждого приложения. Может комбинироваться с text.
* textCapWords — текст с автоматическим переводом в верхний регистр первого символа каждого слова. Может комбинироваться с text.
* textEmailAddress — адрес e-mail.
* textEmailSubject — тема письма.
* textFilter — текст, фильтрующий какие-то другие данные.
* textImeMultiLine — говорит методу ввода поддерживать ввод многострочного текста. Может комбинироваться с text.
* textLongMessage — длинный текст.
* textMultiLine — многострочный текст. Может комбинироваться с text.
* textNoSuggestions — говорит методу ввода не предлагать подсказки при вводе. Может комбинироваться с text.
* textPassword — пароль. Введенные данные будут скрыты.
* textPersonName — имя.
* textPhonetic — текст, отображающий фонетическое произношение.
* textPostalAddress — почтовый адрес.
* textShortMessage — короткий текст.
* textUri — URL.
* textVisiblePassword — то же самое, что textPassword, но не скрывающий данные.
* textWebEditText — текст для ввода в веб-форме.
* textWebEmailAddress — адрес e-mail, который будет введен в веб-форме.
* textWebPassword — пароль, который будет введен в веб-форме.

Как видите, вариантов очень много. Но не пугайтесь — на практике вам придется использовать лишь малую часть из них, так что учить их не надо :)

Давайте посмотрим пару вариантов. Например, phone:

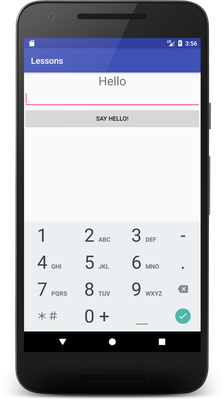
<EditText

android:id="@+id/name\_et"

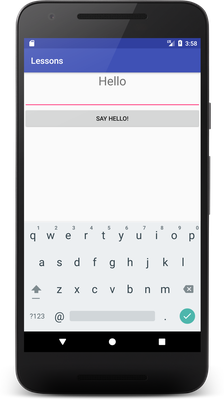
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

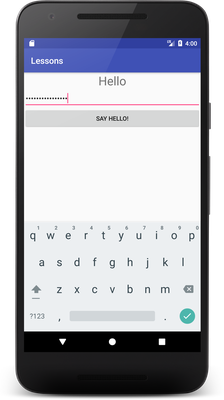
android:inputType="phone" />

[](https://skillberg.com/media/uploads/2017/11/17/aeea283b2235492b8c7304c439e85da0.png)EditText для ввода телефона

Как видите, клавиатура сменилась на цифровую с несколькими дополнительными символами. А теперь textEmailAddress:

[](https://skillberg.com/media/uploads/2017/11/17/9dd48d8cd5db41d98d12f048b0a9d575.png)EditText для ввода адреса e-mail

На клавиатуре появился символ "@". Ну и пароль:

[](https://skillberg.com/media/uploads/2017/11/17/44eed00edfa04772ae1246c869364bb9.png)EditText для ввода пароля

Сразу после ввода символ видно, но через небольшой промежуток времени он сменяется на "•".

Подсказка

Как дать пользователю понять, что мы хотим получить от него в поле? Показать подсказку! Для этого есть атрибут android:hint. Он принимает текстовый ресурс или текст.

<EditText

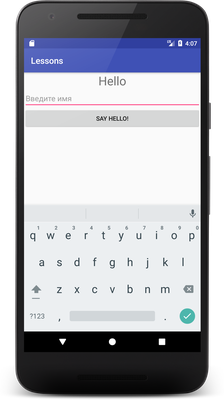
android:id="@+id/name\_et"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Введите имя"

android:inputType="textPersonName" />

[](https://skillberg.com/media/uploads/2017/11/17/56777617c3f745449e033c3eb876f342.png)Подсказка EditText в Android

Подсказка исчезает, когда пользователь что-то вводит и появляется, когда в поле пусто.

Многострочный текст

По умолчанию в EditText, как и в TextView всего одна строка. Допустим, вы хотите, чтобы было 10 строк. Для этого нужно задать атрибут android:lines.

Однако, для того, чтобы EditText стал многострочным, нужно установить значение textMultiLine в атрибут android:inputType (его можно комбинировать с другими флагами).

Например, вот так:

<EditText

android:id="@+id/name\_et"

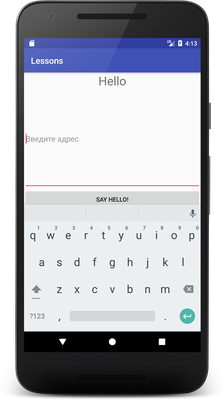
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Введите адрес"

android:inputType="textPostalAddress|textMultiLine"

android:lines="10" />

[](https://skillberg.com/media/uploads/2017/11/17/a0117d3bf23c40729cbb91ed0eaff23c.png)Многострочный EditText

Видите проблему? Текст и подсказка вводятся в середине поля. Чтобы это исправить, нужно задать атрибут android:gravity:

<EditText

android:id="@+id/name\_et"

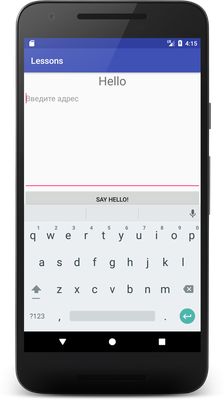
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="top|start"

android:hint="Введите адрес"

android:inputType="textPostalAddress|textMultiLine" />

[](https://skillberg.com/media/uploads/2017/11/17/38b849a607574231913fc125ed1dd25f.png)android:gravity в EditText

Отлично, теперь все так, как мы и хотели. А что, если нам надо по умолчанию показать одну строку, но дать возможность ввести несколько?

Нужно использовать атрибут android:maxLines:

<EditText

android:id="@+id/name\_et"

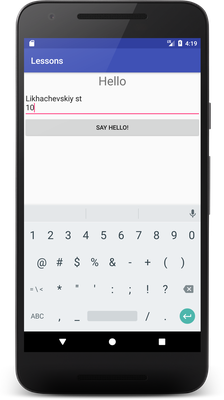
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="top|start"

android:hint="Введите адрес"

android:inputType="textPostalAddress|textMultiLine" />

[](https://skillberg.com/media/uploads/2017/11/17/768ccca8336d40ac9c91bd477a10c953.png)android:maxLines в EditText

Обратите внимание, что этот атрибут не ограничивает количество строк, которые можно ввести. Он говорит EditText, сколько строк в высоту он может занимать. Если ввести большее количество строк, то контент внутри EditText начнет скроллиться.

Получение введенного текста

Отлично, мы научились конфигурировать поле для ввода текста. Но от того, что мы показали его, толку не очень много, если мы не знаем, как получить введенные данные, не так ли?

Давайте модифицируем задачу из прошлого урока: мы просим пользователя ввести его имя, и по нажатию на кнопку "здороваемся" с ним в TextView.

Модифицируйте код лэйаута таким образом:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center\_horizontal"

android:orientation="vertical"

tools:context="com.test.lessons.MainActivity">

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello"

android:textSize="28sp" />

<EditText

android:id="@+id/name\_et"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Введите имя"

android:inputType="textPersonName" />

<Button

android:id="@+id/hello\_btn"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Say hello!" />

</LinearLayout>

Теперь пришло время для работы с Java-кодом. Для начала нужно найти наш EditText и присвоить его члену класса:

package com.test.lessons;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

private TextView textView;

private EditText editText;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textView = findViewById(R.id.hello\_tv);

Button button = findViewById(R.id.hello\_btn);

editText = findViewById(R.id.name\_et);

button.setOnClickListener(onClickListener);

}

private final View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

textView.setText("Hi!");

}

};

}

Получить текст можно, вызвав метод getText(). Обратите внимание, что этот метод возвращает не строку, а [Editable](https://developer.android.com/reference/android/text/Editable.html). Так сделано по ряду оптимизационных причин.

Чтобы получить строку, нужно вызвать метод toString(). Найдите обработчик нажатия на кнопку и модифицируйте его следующим образом:

private final View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

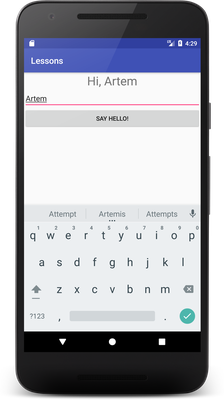
String name = editText.getText().toString();

textView.setText("Hi, " + name);

}

};

Теперь запустите приложение, введите имя и нажмите на кнопку.

[](https://skillberg.com/media/uploads/2017/11/17/488a9564be68408e84447173cfe33508.png)Получение текста из EditText

Супер!

Следим за введенным текстом

Те, кто являются тестировщиками по профессии или в душе, могут заметить, что можно не вводить имя, и нажать на кнопку.

Вряд ли это можно назвать корректным поведением приложения. Что же делать? Отслеживать пользовательский ввод! Можно показать пользователю сообщение об ошибке, если он ввел пустой текст, а можно просто деактивировать кнопку, если текст пустой.

Мы можем задать TextWatcher для EditText, и получать в нем оповещения о том, что текст был отредактирован. Делается это с помощью метода addTextChangedListener. Измените код MainActivity, чтобы он выглядел вот так:

package com.test.lessons;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

private TextView textView;

private EditText editText;

private Button button;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textView = findViewById(R.id.hello\_tv);

button = findViewById(R.id.hello\_btn);

editText = findViewById(R.id.name\_et);

editText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void afterTextChanged(Editable editable) {

}

});

button.setOnClickListener(onClickListener);

}

private final View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

String name = editText.getText().toString();

textView.setText("Hi, " + name);

}

};

}

Нам пригодится метод afterTextChanged(). Проверим в нем длину текста, и если она больше нуля, то включим кнопку, если меньше — то выключим. За доступность любого View отвечают атрибут android:enabled и метод setEnabled(), принимающие true или false.

Кстати, не забудьте добавить атрибут android:enabled со значением false для кнопки в файле XML, иначе по умолчанию она будет включена.

Измените метод afterTextChanged() следующим образом:

@Override

public void afterTextChanged(Editable editable) {

if (editable.toString().trim().length() > 0) {

button.setEnabled(true);

} else {

button.setEnabled(false);

}

}

Метод trim() нужен для того, чтобы хитрые пользователи не ввели пробел — он как раз убирает все пробелы в начале и в конце текста.

Запустите приложение, чтобы убедиться что все работает.

На этом все. Жду вас в следующем уроке!

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/LessonsAndroid/tree/lesson8)