

**КАЛУЖСКИЙ ФИЛИАЛ  
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО  
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»**



**Факультет** «Информатика и управление»

**Кафедра** "Программное обеспечение ЭВМ, информационные технологии"

## **Виджеты**

**Калуга**

# Виджеты

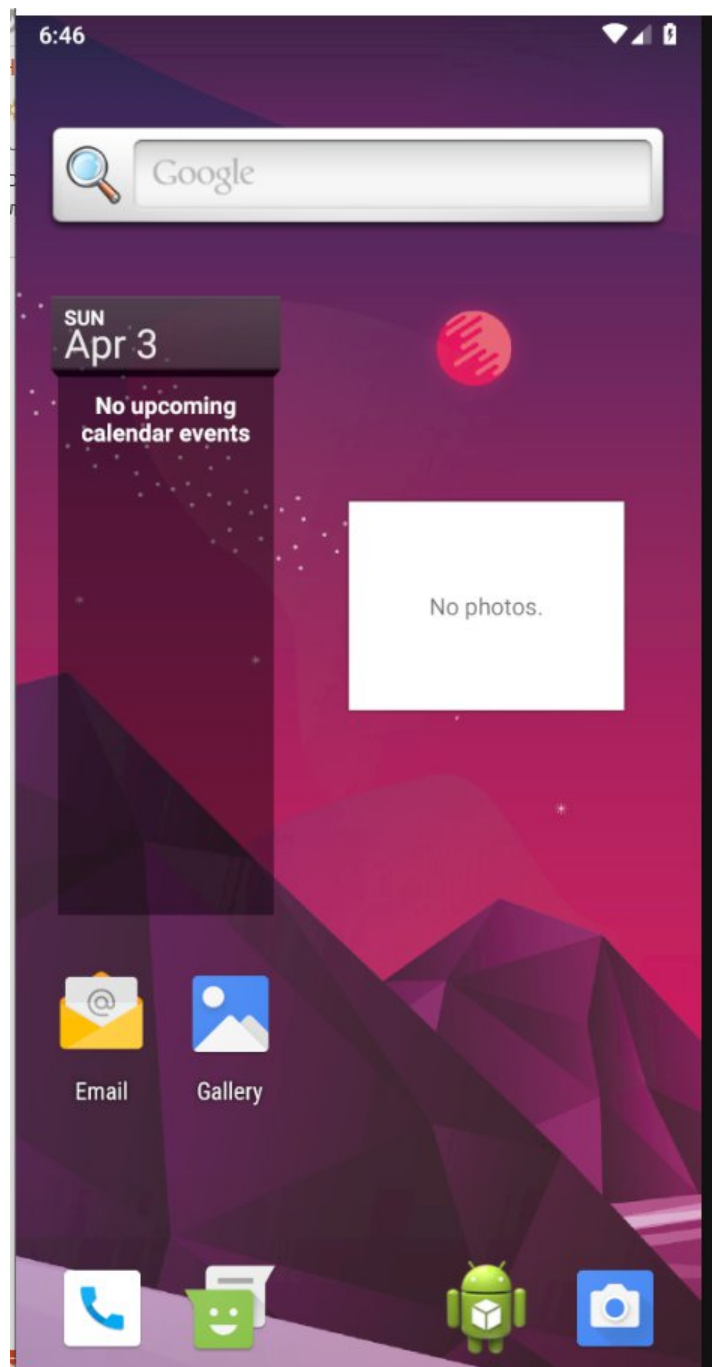
Виджет – это небольшая программа, или часть программы, которая располагается на рабочем столе устройства и предназначена для отображения информации, управления оборудованием устройства и при этом может запускать другую программу, частью которой он является. Несколько виджетов предоставляет сам Android — это аналоговые часы, пульт управления проигрыванием музыки и виджет, показывающий картинки.

Множество разработчиков создали интересные виджеты с помощью которых можно отображать такие сведения, как загрузка процессора, состояние батареи, информацию о текущей погоде и прочем. Есть виджеты, с помощью которых можно быстро включить или выключить GPS, Wi-Fi, Bluetooth, динамики и управлять другим оборудованием Android устройства. Есть такие виджеты, как погодные информеры, которые отображают на экране информацию о текущей погоде и прогнозе погоды, и которые могут вызывать погодное приложение, частью которого они являются.

Виджеты могут иметь различный размер – от минимального размера 1x1, в стиле обычного ярлыка, до полноэкранного.

Таким образом, виджет это такая программа, которая «живет» на экране вашего планшета или телефона и позволяет вам управлять им, получать необходимую информацию и прочее.

В данной лекции будет изучено, как создавать свои собственные виджеты.



**Чтобы создать простейший виджет нам понадобятся три детали:**

**1) Layout-файл.** - В нем мы формируем внешний вид виджета. Все аналогично layout-файлам для Activity и фрагментов, только набор доступных компонентов здесь ограничен следующим списком:

FrameLayout  
LinearLayout  
RelativeLayout  
GridLayout  
AnalogClock  
Button  
Chronometer  
ImageButton  
ImageView  
ProgressBar  
TextView  
ViewFlipper  
ListView  
GridView  
StackView  
AdapterViewFlipper

**2) XML-файл с метаданными** - В нем задаются различные характеристики виджета:

- layout-файл , чтобы виджет знал, как он будет выглядеть
- размер виджета, чтобы виджет знал, сколько места он должен занять на экране
- интервал обновления, чтобы система знала, как часто ей надо будет обновлять виджет

### 3) Класс, наследующий **AppWidgetProvider**.

В этом классе нам надо будет реализовать Lifecycle методы виджета.

#### **onEnabled**

вызывается системой при создании первого экземпляра виджета

#### **onUpdate**

вызывается при обновлении виджета. На вход, кроме контекста, метод получает объект **AppWidgetManager** и список ID экземпляров виджетов, которые обновляются. Именно этот метод обычно содержит код, который обновляет содержимое виджета. Для этого нам нужен будет **AppWidgetManager**, который мы получаем на вход.

#### **onDelete**

вызывается при удалении каждого экземпляра виджета. На вход, кроме контекста, метод получает список ID экземпляров виджетов, которые удаляются.

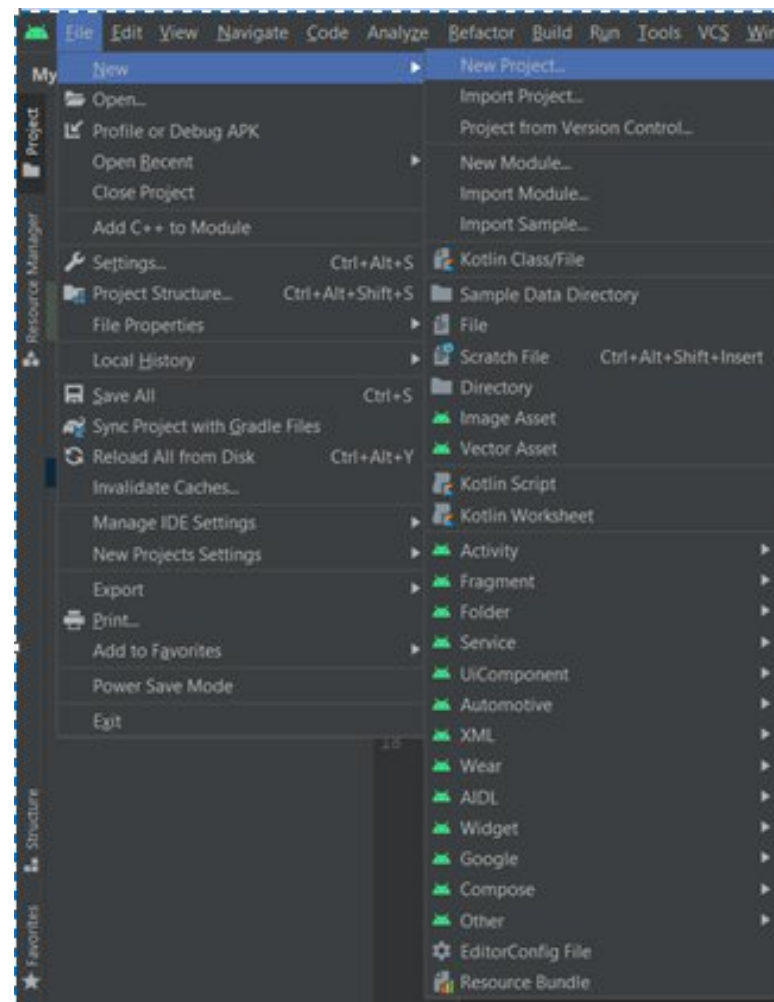
#### **onDisabled**

вызывается при удалении последнего экземпляра виджета.

# Виджет «Hello, Android»

Создадим простейший виджет, который отображает фразу «Hello, Android from Widget!».

Выберите команду меню **File ► New ► New Project...**, чтобы открыть диалоговое окно **Create New Project**.



Templates

Phone and Tablet

Wear OS

Android TV

Automotive



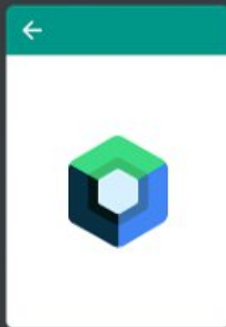
No Activity



Basic Activity



Bottom Navigation Activity



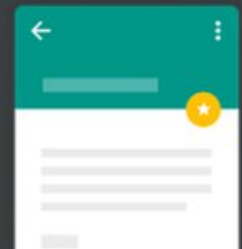
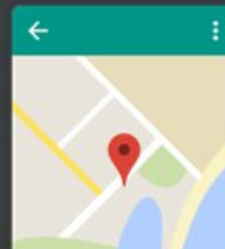
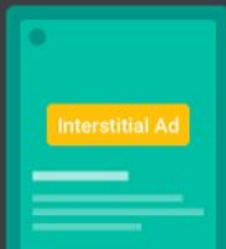
Empty Compose Activity



Empty Activity



Fullscreen Activity



Previous

Next


Cancel


Finish

**Basic Activity**

Creates a new basic activity with the Navigation component

Name	<input type="text" value="My Application"/>
Package name	<input type="text" value="com.example.myapplication"/>
Save location	<input type="text" value="C:\Android\MyApplication5"/> 
Language	<div>Java ▼</div>
Minimum SDK	<div>API 28: Android 9.0 (Pie) ▼</div>

 Your app will run on approximately **69,0%** of devices.  
[Help me choose](#)

☐ Use legacy android.support libraries 

Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

[Previous](#)[Next](#)[Cancel](#)[Finish](#)



## Отредактируем res/values/strings.xml

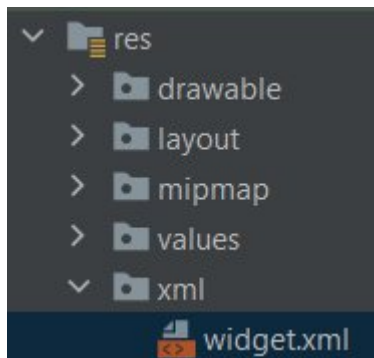
```
<resources>
    <string name="app_name">Wid3</string>
    <string name="action_settings">Settings</string>
    <string name="hello">Hello, Android from Widget!</string>
    <string name="widget_name">WidgetAnd</string>
</resources>
```

Макет определяется в **res/layout/activity\_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#6600ff00"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:text="@string/hello"
        android:textColor="@android:color/black"
        android:textSize="18sp" />
</LinearLayout>
```

## Определим метафайл виджета в **res/xml/widget.xml**



```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="146dip"
    android:minHeight="72dip"
    android:updatePeriodMillis="1800000"
    android:initialLayout="@layout/activity_main"
/>
```

Этот код определяет минимальный размер виджета, как часто его следует обновлять и ссылку на его стартовый макет.

**initialLayout** указывает layout-файл для виджета

Атрибуты **minHeight** и **minWidth** содержат минимальные размеры виджета по высоте и ширине.

Есть определенный алгоритм расчета этих цифр. Как вы наверняка замечали, при размещении виджета, экран делится на ячейки, и виджет занимает одну или несколько из этих ячеек по ширине и высоте. Чтобы конвертировать ячейки в dp, используется формула  $70 * n - 30$ , где  $n$  — это количество ячеек. Т.е. если мы, например, хотим, чтобы наш виджет занимал 2 ячейки в ширину и 1 в высоту, мы высчитываем ширину  $= 70 * 2 - 30 = 110$  и высоту  $= 70 * 1 - 30 = 40$ . Эти полученные значения и будем использовать в атрибутах **minWidth** и **minHeight**.

Атрибут **updatePeriodMillis** содержит количество миллисекунд. Это интервал обновления виджета.

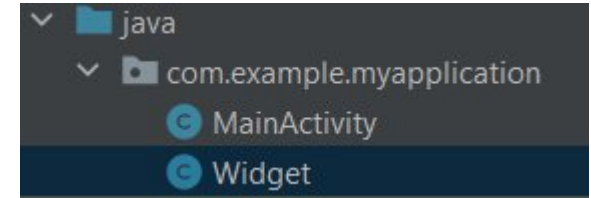
## Создадим класс наследующий AppWidgetProvider Widget.java

//Java

```
package com.example.user.wid3;
import java.util.Arrays;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.util.Log;

public class Widget extends AppWidgetProvider {

    final String LOG_TAG = "myLogs";
    @Override
    public void onEnabled(Context context) {
        super.onEnabled(context);
        Log.d(LOG_TAG, "onEnabled");
    }
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
        Log.d(LOG_TAG, "onUpdate " + Arrays.toString(appWidgetIds));
    }
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
        Log.d(LOG_TAG, "onDeleted " + Arrays.toString(appWidgetIds));
    }
    @Override
    public void onDisabled(Context context) {
        super.onDisabled(context);
        Log.d(LOG_TAG, "onDisabled");
    }
}
```



**//Kotlin**

```
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider

class Widget : AppWidgetProvider() {
    val LOG_TAG = "myLogs"
    override fun onEnabled(context: Context?) {
        super.onEnabled(context)
        Log.d(LOG_TAG, "onEnabled")
    }
    override fun onUpdate(
        context: Context?, appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray?
    ) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        Log.d(LOG_TAG, "onUpdate " + Arrays.toString(appWidgetIds))
    }
    override fun onDeleted(context: Context?, appWidgetIds:
IntArray?) {
        super.onDeleted(context, appWidgetIds)
        Log.d(LOG_TAG, "onDeleted " +
Arrays.toString(appWidgetIds))
    }
    override fun onDisabled(context: Context?) {
        super.onDisabled(context)
        Log.d(LOG_TAG, "onDisabled")
    }
}
```

Теперь можно отредактировать файл манифеста - **AndroidManifest.xml**  
Добавим туда класс **Widget.java** как **Receiver**

- укажите для него свои **label**

- настройте для него фильтр

  - с **action** = ***android.appwidget.action.APPWIDGET\_UPDATE***

  - (ACTION\_APPWIDGET\_UPDATE – это единственный action, который необходимо прописать явно. Остальные события AppWidgetManager сам доставит до AppWidgetProvider-наследника)*

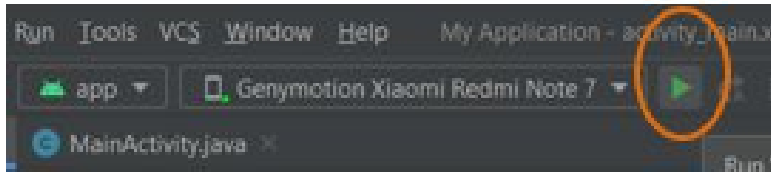
- добавьте метаданные с именем ***android.appwidget.provider*** и указанием файла метаданных **xml/widget.xml** в качестве ресурса.

В итоге файл манифеста **AndroidManifest.xml** будет выглядеть так :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.user.wid3"
    android:versionCode="1"
    android:versionName="1.0" >
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name">
        <receiver
            android:name=".Widget"
            android:label="@string/widget_name" >
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
            <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/widget" />
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />
</manifest>
```

# Запуск приложения

Проект запускается как обычно.

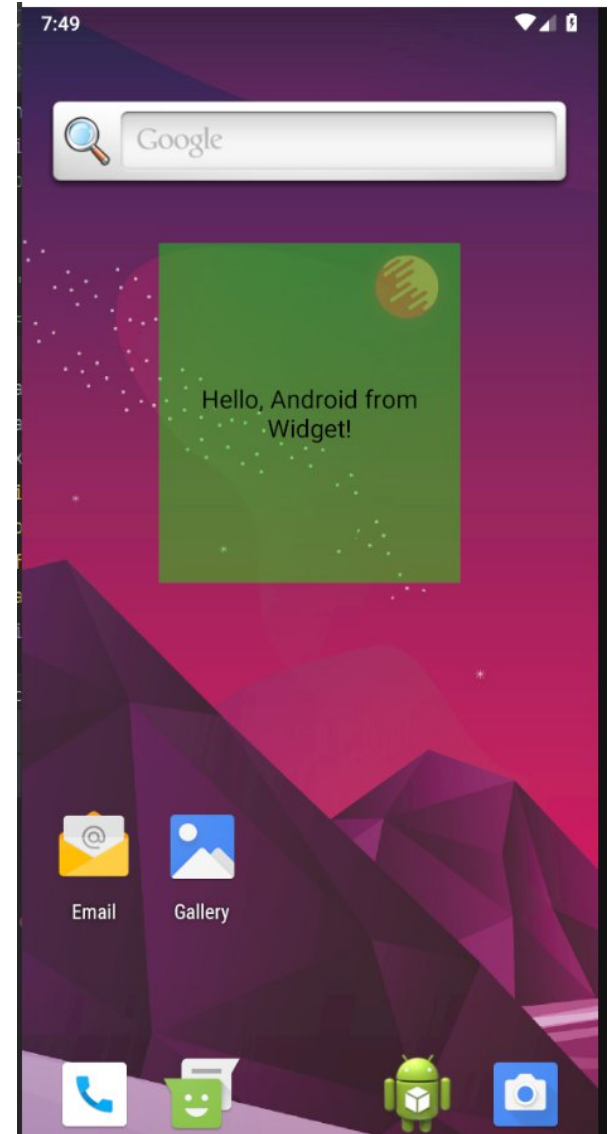
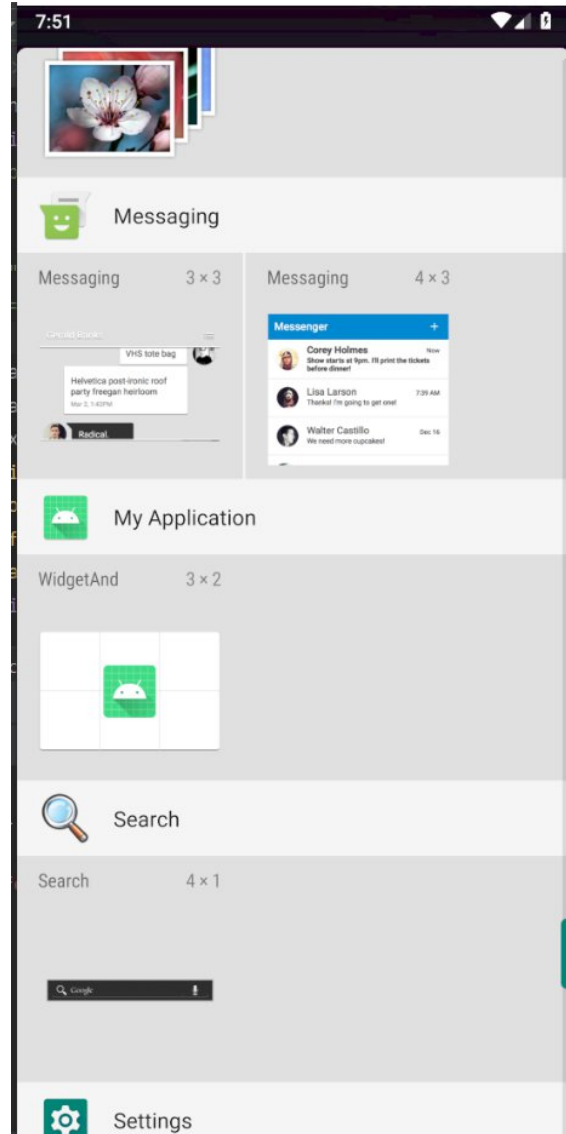
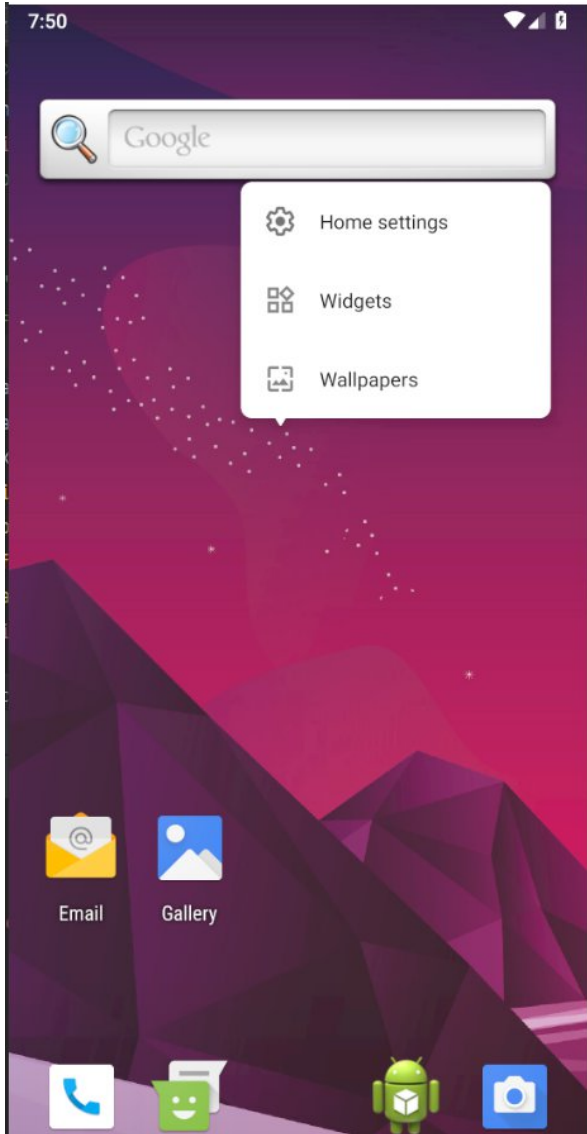


К сожалению, вы не увидите никаких сообщений, которые бы могли как-то отобразить, что ваш Android Studio компилирует и устанавливает виджет на ваш эмулятор или на устройство.

Чтобы увидеть новый виджет, откройте контекстное меню домашнего экрана: нажмите и удерживайте палец (или мышь) на домашнем экране. Появится меню с перечислением всех видов объектов, которые вы можете добавить.

Выберите **Widgets** из меню, затем выберите виджет, названный Widget . В итоге ваш виджет должен появиться на экране

# Работа приложения





# RemoteViews

Класс `RemoteViews` предназначен для описания и управления иерархиями Представлений, которые принадлежат к процессу другого приложения. Это позволяет изменять свойства или вызывать методы, принадлежащие Представлению, которое выступает частью другого приложения.

Например, Представления внутри виджетов работают в отдельном процессе (как правило, это домашний экран), поэтому `RemoteViews` может использоваться для изменения пользовательского интерфейса виджета из Приемника намерений, работающего внутри вашего приложения.

На вход `RemoteViews` принимает имя пакета нашего приложения и ID layout-файла виджета. Теперь `RemoteViews` знает view-структуру нашего виджета. `RemoteViews` имеет несколько методов работы с view, где мы указываем ID нужного нам view-компонента и значение, которое хотим передать. Из названия этих методов понятно, что они делают например `setTextViewText`. По названию понятно, что этот метод вставит текст в `TextView`. Мы вызываем его и передаем ID нашего `TextView` (из layout-файла виджета) и текст, который хотим в него поместить. Система потом найдет в виджете view с указанным ID и вызовет для него метод `setText` с указанным текстом.

Но таких явных методов немного. Они созданы просто для удобства и являются оболочками общих методов, которые позволяют вызвать любой метод view.

В названии общего метода содержится тип данных, которые вы хотите передать. А на вход методу кроме ID view и значения, необходимо будет указать имя метода.

# Отображение даты WidDate

Усовершенствуем немного наш виджет. Пусть он показывает дату (День недели и число). Для этого заполним **WidDate.java**.

**//Java**

```
public class WidDate extends AppWidgetProvider {
    private SimpleDateFormat formatter = new SimpleDateFormat("EEEEEEEEEE\nd MMM
yyy" );

    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {
        String now = formatter.format(new Date());
        RemoteViews updateViews = new RemoteViews(
            context.getPackageName(), R.layout.activity_main);
        updateViews.setTextViewText(R.id.text, now);
        appWidgetManager.updateAppWidget(appWidgetIds, updateViews);
        super.onUpdate(context, appWidgetManager, appWidgetIds);
    }
    public void onEnabled(Context context) { super.onEnabled(context); }
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
    }
    public void onDisabled(Context context) { super.onDisabled(context); }
}
```

## //Kotlin

```
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.widget.RemoteViews
import sun.invoke.util.VerifyAccess.getPackageName

class WidDate : AppWidgetProvider() {
    private val formatter: SimpleDateFormat = SimpleDateFormat("EEEEEEEEEE\nd MMM yyyy")
    override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager,
        appWidgetIds: IntArray?) {
        val now: String = formatter.format(Date())
        val updateViews = RemoteViews(context.getPackageName(), R.layout.activity_main)
        updateViews.setTextViewText(R.id.text, now)
        appWidgetManager.updateAppWidget(appWidgetIds, updateViews)
        super.onUpdate(context, appWidgetManager, appWidgetIds)
    }

    override fun onEnabled(context: Context?) {
        super.onEnabled(context)
    }

    override fun onDeleted(context: Context?, appWidgetIds: IntArray?) {
        super.onDeleted(context, appWidgetIds)
    }

    override fun onDisabled(context: Context?) {
        super.onDisabled(context)
    }
}
```

Когда бы ни поступило намерение **APPWIDGET\_UPDATE**, Android вызывает метод **onUpdate()**.

**//Java**

```
public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    String now = formatter.format(new Date());
    RemoteViews updateViews = new RemoteViews(context.getPackageName(), R.layout.activity_main);
    updateViews.setTextViewText(R.id.text, now);
    appWidgetManager.updateAppWidget(appWidgetIds, updateViews);
    super.onUpdate(context, appWidgetManager, appWidgetIds);
}
```

**//Kotlin**

```
override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager, appWidgetIds: IntArray?) {
    val now: String = formatter.format(Date())
    val updateViews = RemoteViews(context.getPackageName(), R.layout.activity_main)
    updateViews.setTextViewText(R.id.text, now)
    appWidgetManager.updateAppWidget(appWidgetIds, updateViews)
    super.onUpdate(context, appWidgetManager, appWidgetIds)
}

private SimpleDateFormat formatter = new SimpleDateFormat("EEEEEEEEEE\nd MMM yyyy" );
```

форматируем текущую дату, используя **SimpleDateFormat**, созданный ранее.

Далее мы создаем экземпляр **RemoteViews** для нашего нового макета просмотра, который отобразит виджет. Этот пример, когда обновляется виджет, использует тот же макет (**R.layout.activity\_main**), что и при его старте

`updateViews.setTextViewText(R.id.text, now)` Вывод текущей даты и времени.

```
appWidgetManager.updateAppWidget(appWidgetIds, updateViews);
super.onUpdate(context, appWidgetManager, appWidgetIds);
```

обновленный вьювер для замены текущего содержимого виджета.

# Конфигурационный экран. Обновление.

Некоторые виджеты при размещении отображают конфигурационный экран, который позволяет настроить их. Например, у вас есть электронный счет на каком-либо сайте. И для этого сайта есть приложение-виджет. Чтобы виджет смог показать баланс именно вашего счета, он должен знать логин-пароль. Как вы понимаете, при разработке невозможно (если, конечно, вы не пишете виджет только для себя) зашить в код виджета нужный пароль и логин пользователя, поэтому эти данные надо у пользователя спросить.

Для этих целей и существует конфигурационный экран (конфигурационное Activity). Он предложит пользователю поля для ввода и сохранит куда-либо (БД, Preferences, ...) введенные данные, а при обновлении виджета эти данные будут считаны и использованы для отображения актуальной информации.

Либо, например, мы хотим настроить внешний вид виджета при размещении.

Создадим приложение с виджетом аналогично предыдущему, только добавим к нему возможность конфигурирования. Будем настраивать цвет фона виджета и отображаемый текст.

## Файл res/strings.xml

```
<resources>
  <string name="app_name">Wid3</string>
  <string name="action_settings">Settings</string>
  <string name="widget_text">Hello, Config to Widget!</string>
  <string name="widget_name">WidgetUpdate</string>
  <string name="red">Red</string>
  <string name="green">Green</string>
  <string name="blue">Blue</string>
  <string name="ok">OK</string>
</resources>
```

## файл метаданных xml/widget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
  android:configure="com.example.user.wid3.ConfigActivity"
  android:minWidth="216dip"
  android:minHeight="72dip"
  android:updatePeriodMillis="1800000"
  android:initialLayout="@layout/widget"
/>
```

## файл layout/widget.xml

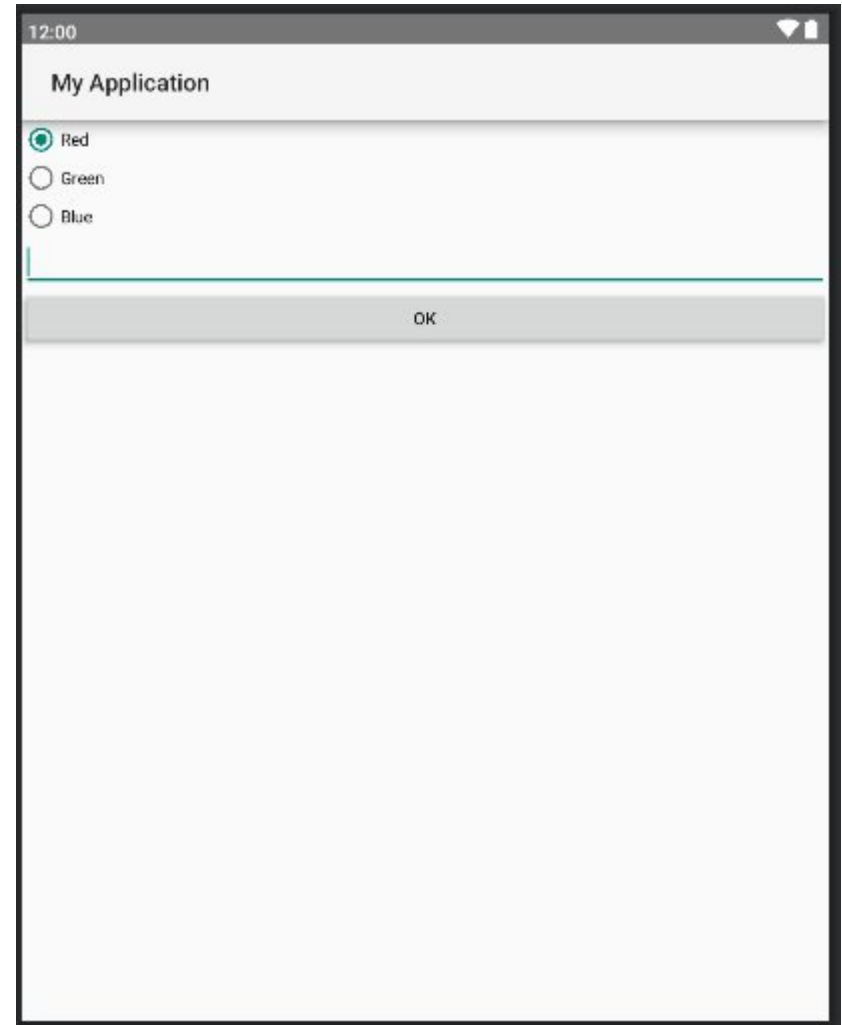
```
<RelativeLayout
  xmlns:android="http://schemas.android.com/
apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
    android:id="@+id/tv"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#6600ff00"
    android:gravity="center"
    android:text="@string/widget_text"
    android:textColor="#000"
    android:textSize="18sp">
  </TextView>
</RelativeLayout>
```

Добавим возможность конфигурирования. Для этого нам надо создать Activity. Это Activity будет запускаться системой при добавлении нового экземпляра виджета и на вход получать ID этого экземпляра.

Конфигурационное Activity - совершенно обычное Activity, состоящее из layout-файла и класса. Начнем с создания layout-файла **config.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <RadioGroup
        android:id="@+id/rgColor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:id="@+id/radioRed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/red">
        </RadioButton>
        <RadioButton
            android:id="@+id/radioGreen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/green">
        </RadioButton>
        <RadioButton
            android:id="@+id/radioBlue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/blue">
        </RadioButton>
    </RadioGroup>
```

```
<EditText
    android:id="@+id/etText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10">
    <requestFocus>
    </requestFocus>
</EditText>
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="@string/ok" >
</Button>
</LinearLayout>
```





Создаем класс для конфигурационного Activity

**ConfigActivity.java:**

**//Java**

```
package com.example.user.wid3;
import android.app.Activity;
import android.appwidget.AppWidgetManager;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.graphics.Color;
import android.widget.EditText;
import android.widget.RadioGroup;

public class ConfigActivity extends Activity {
    int widgetID = AppWidgetManager.INVALID_APPWIDGET_ID;
    Intent resultValue;
    public final static String WIDGET_PREF = "widget_pref";
    public final static String WIDGET_TEXT = "widget_text_";
    public final static String WIDGET_COLOR = "widget_color_";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            widgetID = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID);
        }
        if (widgetID == AppWidgetManager.INVALID_APPWIDGET_ID) {finish();}
        resultValue = new Intent();
        resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
        setResult(RESULT_CANCELED, resultValue);
        setContentView(R.layout.config);
    }
}
```

```

public void onClick(View v) {
    int selRBColor = ((RadioGroup) findViewById(R.id.rgColor))
        .getCheckedRadioButtonId();
    int color = Color.RED;
    switch (selRBColor) {
        case R.id.radioRed:
            color = Color.parseColor("#66ff0000");
            break;
        case R.id.radioGreen:
            color = Color.parseColor("#6600ff00");
            break;
        case R.id.radioBlue:
            color = Color.parseColor("#660000ff");
            break;
    }
    EditText etText = (EditText) findViewById(R.id.etText);
    SharedPreferences sp = getSharedPreferences(WIDGET_PREF, MODE_PRIVATE);
    Editor editor = sp.edit();
    etText.setText("Hello Android");
    editor.putString(WIDGET_TEXT + widgetID, etText.getText().toString());
    editor.putInt(WIDGET_COLOR + widgetID, color);
    editor.commit();

    AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(this);
    MyWidget.updateWidget(this, appWidgetManager, sp, widgetID);
    setResult(RESULT_OK, resultValue);
    finish();
}
}

```

//Kotlin

```
import android.app.Activity
import android.appwidget.AppWidgetManager
import android.content.Context
import android.content.Intent
import android.graphics.Color
import android.os.Bundle
import android.view.View
import android.widget.EditText
import android.widget.RadioGroup

class ConfigActivity : Activity() {
    var widgetID = AppWidgetManager.INVALID_APPWIDGET_ID
    var resultValue: Intent? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val intent = intent
        val extras = intent.extras
        if (extras != null) {
            widgetID = extras.getInt(
                AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID
            )
        }
        if (widgetID == AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish()
        }
        resultValue = Intent()
        resultValue!!.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID)
        setResult(RESULT_CANCELED, resultValue)
        setContentView(R.layout.config)
    }
}
```

```

companion object {
    const val WIDGET_PREF = "widget_pref"
    const val WIDGET_TEXT = "widget_text_"
    const val WIDGET_COLOR = "widget_color_"
}

fun onClick(v: View?) {
    val selRBColor = (findViewById(R.id.rgColor) as
RadioGroup).checkedRadioButtonId
    var color: Int = Color.RED
    when (selRBColor) {
        R.id.radioRed -> color = Color.parseColor("#66ff0000")
        R.id.radioGreen -> color = Color.parseColor("#6600ff00")
        R.id.radioBlue -> color = Color.parseColor("#660000ff")
    }
    val etText = findViewById(R.id.etText) as EditText
    val sp = getSharedPreferences(WIDGET_PREF, Context.MODE_PRIVATE)
    val editor = sp.edit()
    etText.setText("Hello Android")
    editor.putString(WIDGET_TEXT + widgetID, etText.text.toString())
    editor.putInt(WIDGET_COLOR + widgetID, color)
    editor.commit()
    val appWidgetManager = AppWidgetManager.getInstance(this)
    MyWidget.updateWidget(this, appWidgetManager, sp, widgetID)
    setResult(RESULT_OK, resultValue)
    finish()
}
}

```

- 1) При вызове конфигурационное Activity получает Intent, в котором содержится ID создаваемого экземпляра виджета.
- 2) При закрытии оно должно формировать результат методом setResult И в этом ответе передавать Intent с ID экземпляра. Рекомендуется при создании Activity сразу формировать отрицательный результат. В этом случае, если пользователь передумает создавать виджет, то система будет знать, что виджет создавать не надо.

В **onCreate** мы из Intent (параметр EXTRA\_APPWIDGET\_ID) извлекаем ID экземпляра виджета

```
widgetID = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,  
    AppWidgetManager.INVALID_APPWIDGET_ID);
```

который будет конфигурироваться этим экраном. Если получен некорректный ID, то Выходим.

```
if (widgetID == AppWidgetManager.INVALID_APPWIDGET_ID) {  
    finish();}
```

Если все ок, то формируем Intent с ID для метода setResult и говорим, что результат отрицательный.

```
resultValue = new Intent();  
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);  
setResult(RESULT_CANCELED, resultValue);
```

В **onClick** мы читаем выбранный цвет и введенный в поле текст и пишем эти значения в Preferences. В имени записываемого параметра мы используем ID, чтобы можно было отличать параметры разных экземпляров друг от друга.

```
int color = Color.RED;
switch (selRBColor) {
case R.id.radioRed:
    color = Color.parseColor("#66ff0000");
    break;
case R.id.radioGreen:
    color = Color.parseColor("#6600ff00");
    break;
case R.id.radioBlue:
    color = Color.parseColor("#660000ff");
    break;
}
EditText etText = (EditText) findViewById(R.id.etText);
SharedPreferences sp = getSharedPreferences(WIDGET_PREF, MODE_PRIVATE);
Editor editor = sp.edit();
editor.putString(WIDGET_TEXT + widgetID, etText.getText()
    .toString());
editor.putInt(WIDGET_COLOR + widgetID, color);
editor.commit();
```

Далее мы говорим системе, что результат работы положительный, и виджет можно создавать. Закрываем Activity.

необходимо добавить в файл метаданных (xml/widget\_metadata.xml) параметр **android:configure** и указать в нем полный путь к классу Activity

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="com.example.user.wid3.ConfigActivity"
    android:minWidth="216dip"
    android:minHeight="72dip"
    android:updatePeriodMillis="1800000"
    android:initialLayout="@layout/widget"
    />
```

Добавим Activity в манифест и настроим ему фильтр

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.user.wid3"
    android:versionCode="1"
    android:versionName="1.0" >
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name">
        <receiver
            android:name=".Widget"
            android:label="@string/widget_name" >
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
            <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/widget" />
        </receiver>
        <activity android:name=".ConfigActivity"
            android:theme="@android:style/Theme.Dialog" android:exported="false">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE"/>
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />
</manifest>
```

Создадим класс MyWidget.java расширяющий AppWidgetProvider

**//Java**

```
package com.example.user.wid3;
import java.util.Arrays;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.widget.RemoteViews;

public class MyWidget extends AppWidgetProvider {
    @Override
    public void onEnabled(Context context) {
        super.onEnabled(context);
    }
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
        SharedPreferences sp = context.getSharedPreferences(
            ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE);
        for (int id : appWidgetIds) {updateWidget(context, appWidgetManager, sp, id);}
    }
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
        Editor editor = context.getSharedPreferences(
            ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE).edit();
        for (int widgetID : appWidgetIds) {
            editor.remove(ConfigActivity.WIDGET_TEXT + widgetID);
            editor.remove(ConfigActivity.WIDGET_COLOR + widgetID);
        }
        editor.commit();
    }
}
```



@Override

```
public void onDisabled(Context context) {
    super.onDisabled(context);
}

static void updateWidget(Context context, AppWidgetManager appWidgetManager,
    SharedPreferences sp, int widgetID) {
    // Читаем параметры Preferences
    String widgetText = sp.getString(ConfigActivity.WIDGET_TEXT + widgetID, null);
    if (widgetText == null) return;
    int widgetColor = sp.getInt(ConfigActivity.WIDGET_COLOR + widgetID, 0);

    // Настраиваем внешний вид виджета
    RemoteViews widgetView = new RemoteViews(context.getPackageName(),
        R.layout.widget);
    widgetView.setTextViewText(R.id.tv, widgetText);
    widgetView.setInt(R.id.tv, "setBackgroundColor", widgetColor);
    // Обновляем виджет
    appWidgetManager.updateAppWidget(widgetID, widgetView);
}
}
```

## //Kotlin

```
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.content.SharedPreferences
import android.content.SharedPreferences.Editor
import android.widget.RemoteViews

class MyWidget : AppWidgetProvider() {
    override fun onEnabled(context: Context?) {
        super.onEnabled(context)
    }

    override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager?,
        appWidgetIds: IntArray) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        val sp: SharedPreferences =
            context.getSharedPreferences(ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE)
        for (id in appWidgetIds) {
            updateWidget(context, appWidgetManager, sp, id)
        }
    }

    override fun onDeleted(context: Context, appWidgetIds: IntArray) {
        super.onDeleted(context, appWidgetIds)
        val editor: Editor = context.getSharedPreferences(ConfigActivity.WIDGET_PREF,
            Context.MODE_PRIVATE).edit()
        for (widgetID in appWidgetIds) {
            editor.remove(ConfigActivity.WIDGET_TEXT + widgetID)
            editor.remove(ConfigActivity.WIDGET_COLOR + widgetID)
        }
        editor.commit()
    }
}
```

```

override fun onDisabled(context: Context?) {
    super.onDisabled(context)
}

companion object {
    fun updateWidget(
        context: Context,
        appWidgetManager: AppWidgetManager?,
        sp: SharedPreferences,
        widgetID: Int
    ) {
        // Читаем параметры Preferences
        val widgetText = sp.getString(ConfigActivity.WIDGET_TEXT + widgetID,
null) ?: return
        val widgetColor = sp.getInt(ConfigActivity.WIDGET_COLOR + widgetID, 0)
        // Настраиваем внешний вид виджета
        val widgetView = RemoteViews(context.packageName, R.layout.widget)
        widgetView.setTextViewText(R.id.tv, widgetText)
        widgetView.setInt(R.id.tv, "setBackgroundColor", widgetColor)
        // Обновляем виджет
        appWidgetManager?.updateAppWidget(widgetID, widgetView)
    }
}

```

В **onUpdate** мы перебираем все ID экземпляров, которые необходимо обновить и для каждого из них вызываем наш метод обновления, который рассмотрим чуть ниже.

```
SharedPreferences sp = context.getSharedPreferences(  
    ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE);  
for (int id : appWidgetIds) {  
    updateWidget(context, appWidgetManager, sp, id);  
}
```

**onDeleted** у нас вызывается, когда виджет удаляется с экрана. Если виджет удален, то логично будет удалить и все настройки для него из Preferences.

```
Editor editor = context.getSharedPreferences(  
    ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE).edit();  
for (int widgetID : appWidgetIds) {  
    editor.remove(ConfigActivity.WIDGET_TEXT + widgetID);  
    editor.remove(ConfigActivity.WIDGET_COLOR + widgetID);  
}  
editor.commit();
```

Метод **updateWidget** обновляет конкретный экземпляр виджета, получая на вход его ID. Здесь мы читаем настройки (и сразу выходим, если нет настройки WIDGET\_TEXT), которые записало нам конфигурационное Activity для этого экземпляра виджета. Нам надо применить эти параметры к view-компонентам нашего виджета.

```
String widgetText = sp.getString(ConfigActivity.WIDGET_TEXT + widgetID, null);  
if (widgetText == null) return;  
int widgetColor = sp.getInt(ConfigActivity.WIDGET_COLOR + widgetID, 0);
```

Создаем RemoteViews. На вход он принимает имя пакета нашего приложения и ID layout-файла виджета. Теперь RemoteViews знает view-структуру нашего виджета. Используем метод [setTextViewText](#). По названию понятно, что этот метод вставит текст в TextView. Мы вызываем его и передаем ID нашего TextView (из layout-файла виджета) и текст, который хотим в него поместить.

Система потом найдет в виджете view с указанным ID (R.id.tv), и вызовет для него метод `setText` с указанным текстом (`widgetText`).

```
RemoteViews widgetView = new RemoteViews(context.getPackageName(),  
    R.layout.widget);  
widgetView.setTextViewText(R.id.tv, widgetText);
```

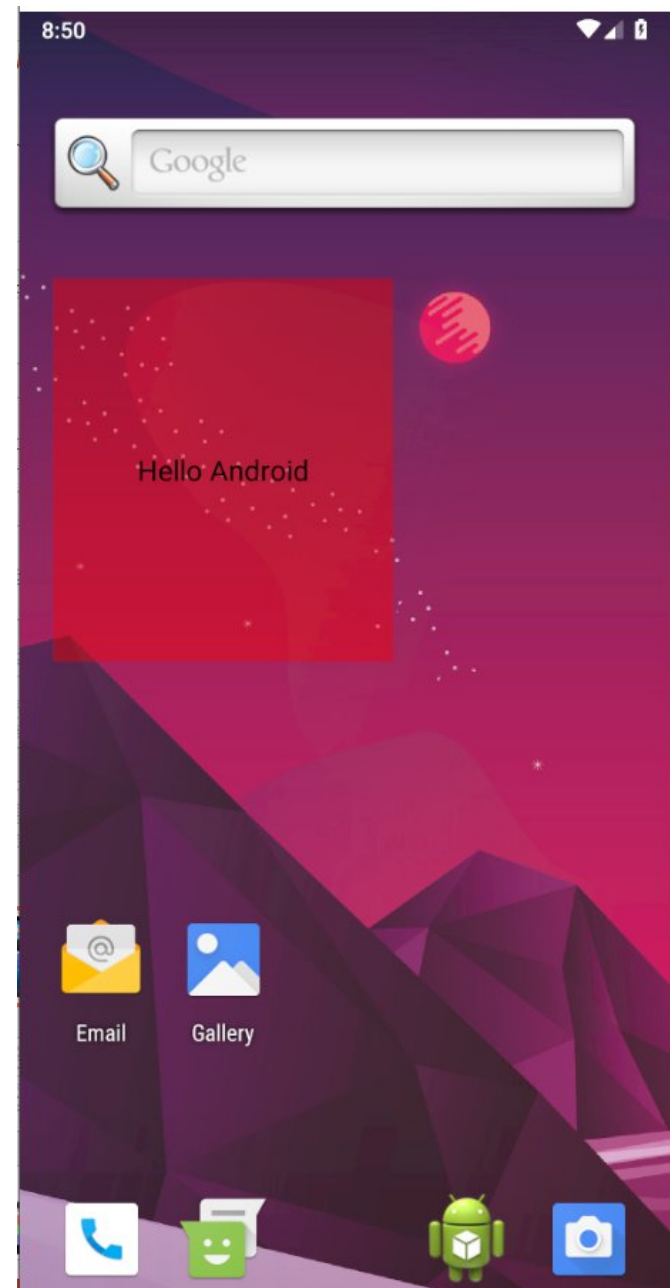
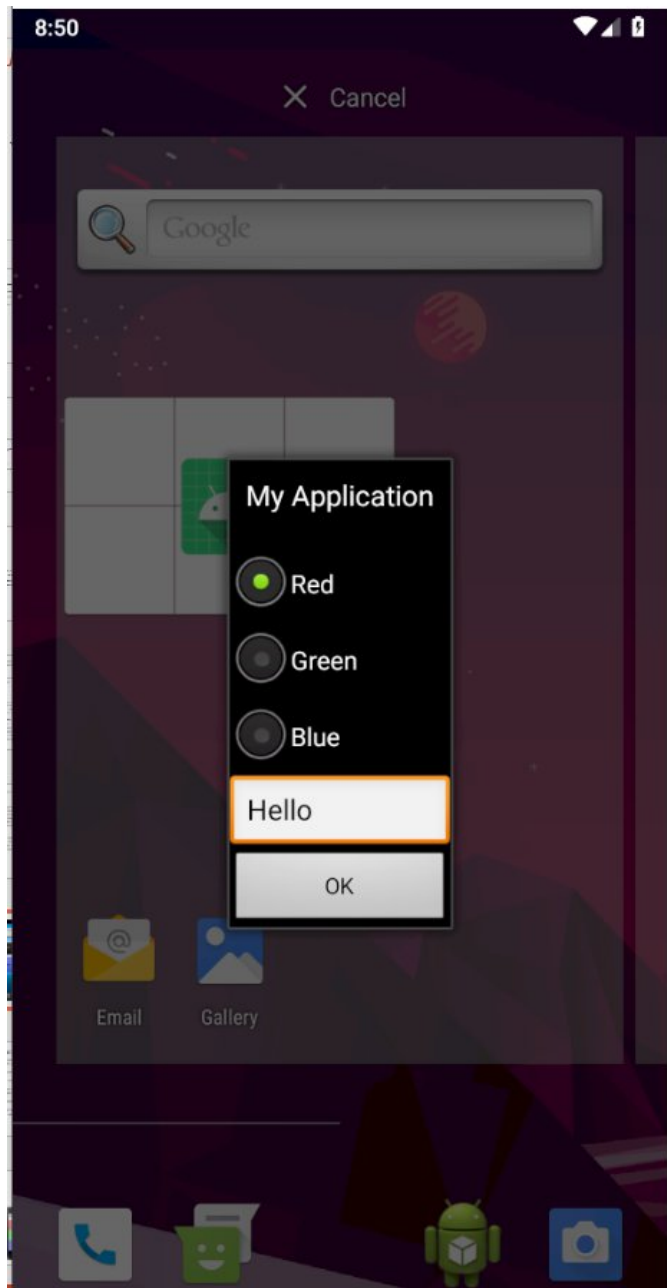
Но таких явных методов немного. Они созданы просто для удобства и являются оболочками общих методов, которые позволяют вызвать любой метод view.

Цвет у нас типа `int`, поэтому мы вызываем метод [setInt](#) и передаем ему ID view, имя метода (который бы вызывали в случае доступа к view – `setBackgroundColor`) и значение цвета. Система найдет в виджете view с указанным ID (R.id.tv) и вызовет для него указанный метод (`setBackgroundColor`) с указанным значением (`widgetColor`).

```
widgetView.setInt(R.id.tv, "setBackgroundColor", widgetColor);
```

`RemoteViews` сформирован. Используем [AppWidgetManager](#), чтобы применить к виджету наши сформированные view-настройки. Для этого используется метод [updateAppWidget](#), который на вход берет ID экземпляра виджета и объект `RemoteViews`. Система найдет указанный экземпляр виджета и настроит его так, как мы указали

```
appWidgetManager.updateAppWidget(widgetID, widgetView);
```



# Обработка нажатия на виджет.

П.к. прямого доступа к view-компонентам виджета мы не имеем, то использовать, как обычно, обработчики нажатий не получится. Но RemoteViews, используемый для работы с view, позволяет настроить реакцию view на нажатие.

Создадим пример отражающий различные техники реагирования на нажатия. Виджет будет состоять из двух текстов и трех зон для нажатий.

Первый текст будет отображать время последнего обновления, а второй – количество нажатий на третью зону нажатия.

Первая зона будет по клику открывать конфигурационное Activity. Это пригодится в том случае, когда вы хотите дать пользователю возможность донстроить виджет после установки. Конфигурировать будем формат отображаемого в первой строке времени.

Вторая зона нажатия будет просто обновлять виджет, тем самым будет меняться время в первом тексте.

Каждое нажатие на третью зону будет увеличивать на единицу счетчик нажатий и обновлять виджет. Тем самым будет меняться второй текст, отображающий текущее значение счетчика.

## Дополним файл res/strings.xml

```
<resources>
  <string name="app_name">Wid3</string>
  <string name="action_settings">Settings</string>
  <string name="widget_text">Hello, Config to Widget!</string>
  <string name="widget_name">WidgetUpdate</string>
  <string name="red">Red</string>
  <string name="green">Green</string>
  <string name="blue">Blue</string>
  <string name="ok">OK</string>
  <string name="config">Config</string>
  <string name="update">Update</string>
  <string name="count">Count</string>
</resources>
```

## Создадим файл layout/widget.xml

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

xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
    android:id="@+id/tvTime"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#fff"
    android:textColor="#000">

  </TextView>
```





```

<TextView
    android:id="@+id/tvCount"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#fff"
    android:textColor="#000">
</TextView>
<TextView
    android:id="@+id/tvPressConfig"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#66ff0000"
    android:gravity="center"
    android:text="@string/config"
    android:textColor="#000">
</TextView>
<TextView
    android:id="@+id/tvPressUpdate"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#6600ff00"
    android:gravity="center"
    android:text="@string/update"
    android:textColor="#000">
</TextView>

```

```

<TextView
    android:id="@+id/tvPressCount"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#660000ff"
    android:gravity="center"
    android:text="@string/count"
    android:textColor="#000">
</TextView>
</LinearLayout>

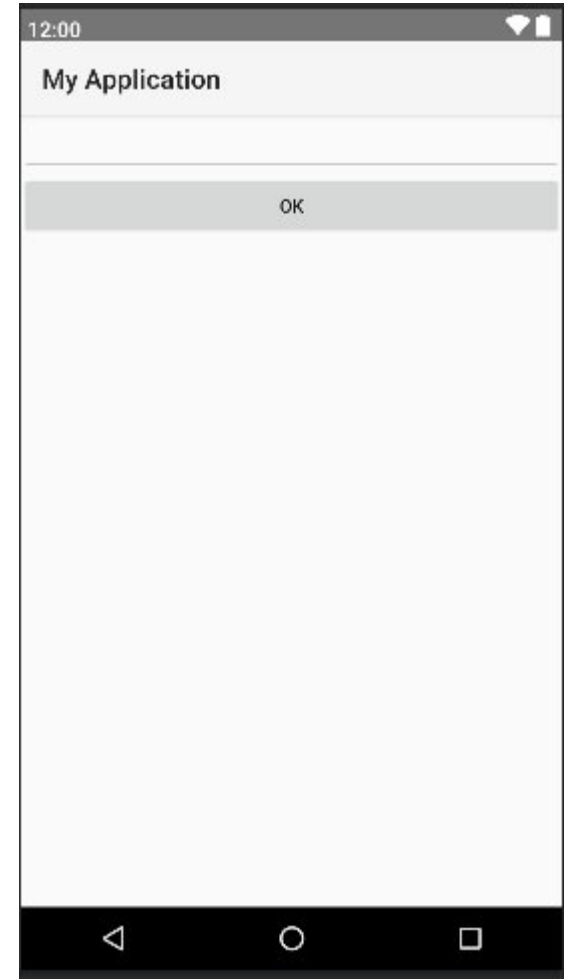
```



Создадим файл для конфигурационного экрана layout/**config.xml**

Поле для ввода формата даты и кнопка подтверждения

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/etFormat"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"/>
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/ok" >
    </Button>
</LinearLayout>
```



# Создадим класс конфигурационного экрана ConfigActivity.java

**//Java**

```
public class ConfigActivity extends Activity {
    public final static String WIDGET_PREF = "widget_pref";
    public final static String WIDGET_TIME_FORMAT = "widget_time_format_";
    public final static String WIDGET_COUNT = "widget_count_";
    int widgetID = AppWidgetManager.INVALID_APPWIDGET_ID;
    Intent resultValue;
    SharedPreferences sp;
    EditText etFormat;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // извлекаем ID конфигурируемого виджета
        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            widgetID = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID);
        }
        // и проверяем его корректность
        if (widgetID == AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish();
        }

        // формируем intent ответа
        resultValue = new Intent();
        resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
        // отрицательный ответ
        setResult(RESULT_CANCELED, resultValue);
        setContentView(R.layout.config);
        sp = getSharedPreferences(WIDGET_PREF, MODE_PRIVATE);
        etFormat = (EditText) findViewById(R.id.etFormat);
        etFormat.setText(sp.getString(WIDGET_TIME_FORMAT + widgetID, "HH:mm:ss"));

        int cnt = sp.getInt(ConfigActivity.WIDGET_COUNT + widgetID, -1);
        if (cnt == -1) sp.edit().putInt(WIDGET_COUNT + widgetID, 0);
    }
}
```

```
public void onClick(View v){
    sp.edit().putString(WIDGET_TIME_FORMAT + widgetID,
etFormat.getText().toString()).commit();
    MyWidget.updateWidget(this, AppWidgetManager.getInstance(this), widgetID);
    setResult(RESULT_OK, resultValue);
    finish();
}
}
```

В **onCreate** мы извлекаем и проверяем ID экземпляра виджета, для которого открылся конфигурационный экран. Далее формируем отрицательный ответ на случай нажатия кнопки Назад. Читаем формат времени и помещаем его в EditText. Читаем значение счетчика и, если этого значения еще нет в Preferences, то пишем туда 0.

В **onClick** мы сохраняем в Preferences формат из EditText, обновляем виджет, формируем положительный ответ и выходим.

**//Kotlin**

```
class ConfigActivity : Activity() {
    var widgetID = AppWidgetManager.INVALID_APPWIDGET_ID
    var resultValue: Intent? = null
    var sp: SharedPreferences? = null
    var etFormat: EditText? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // извлекаем ID конфигурируемого виджета
        val intent = intent
        val extras = intent.extras
        if (extras != null)
        {
            widgetID=extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,AppWidgetManager.INVALID_APPWIDGET_ID)
        }

        // и проверяем его корректность
        if (widgetID == AppWidgetManager.INVALID_APPWIDGET_ID) {finish()}
        // формируем intent ответа
        resultValue = Intent()
        resultValue!!.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID)
        // отрицательный ответ
        setResult(RESULT_CANCELED, resultValue)
        setContentView(R.layout.config)
        sp = getSharedPreferences(WIDGET_PREF, Context.MODE_PRIVATE)
        etFormat = findViewById<View>(R.id.etFormat) as EditText
        etFormat!!.setText(sp?.getString(WIDGET_TIME_FORMAT + widgetID, "HH:mm:ss"))
        val cnt = sp?.getInt(WIDGET_COUNT + widgetID, -1)
        if (cnt == -1) sp?.edit()?.putInt(WIDGET_COUNT + widgetID, 0)
    }
    companion object {
        const val WIDGET_PREF = "widget_pref"
        const val WIDGET_TIME_FORMAT = "widget_time_format_"
        const val WIDGET_COUNT = "widget_count_"
    }
    fun onClick(v: View?) {
        sp!!.edit().putString(WIDGET_TIME_FORMAT + widgetID, etFormat!!.text.toString()).commit()
        MyWidget.updateWidget(this, AppWidgetManager.getInstance(this), widgetID)
        setResult(RESULT_OK, resultValue)
        finish()
    }
}
```

## Создадим класс виджета MyWidget.java:

**//Java**

```
package com.example.user.wid3;
import java.sql.Date;
import java.text.SimpleDateFormat;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.widget.RemoteViews;

public class MyWidget extends AppWidgetProvider {
    final static String ACTION_CHANGE = "com.example.user.wid3.change_count";
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
        // обновляем все экземпляры
        for (int i : appWidgetIds) {
            updateWidget(context, appWidgetManager, i);
        }
    }
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
        // Удаляем Preferences
        Editor editor = context.getSharedPreferences(
            ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE).edit();
        for (int widgetID : appWidgetIds) {
            editor.remove(ConfigActivity.WIDGET_TIME_FORMAT + widgetID);
            editor.remove(ConfigActivity.WIDGET_COUNT + widgetID);
        }
        editor.commit();
    }
}
```

```

static void updateWidget(Context ctx, AppWidgetManager appWidgetManager,
                        int widgetID) {
    SharedPreferences sp = ctx.getSharedPreferences(
        ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE);
    // Читаем формат времени и определяем текущее время
    String timeFormat = sp.getString(ConfigActivity.WIDGET_TIME_FORMAT
        + widgetID, null);
    if (timeFormat == null) return;
    SimpleDateFormat sdf = new SimpleDateFormat(timeFormat);
    String currentTime = sdf.format(new Date(System.currentTimeMillis()));
    // Читаем счетчик
    String count = String.valueOf(sp.getInt(ConfigActivity.WIDGET_COUNT
        + widgetID, 0));
    // Помещаем данные в текстовые поля
    RemoteViews widgetView = new RemoteViews(ctx.getPackageName(),
        R.layout.widget);
    widgetView.setTextViewText(R.id.tvTime, currentTime);
    widgetView.setTextViewText(R.id.tvCount, count);
    // Конфигурационный экран (первая зона)
    Intent configIntent = new Intent(ctx, ConfigActivity.class);
    configIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_CONFIGURE);
    configIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
    PendingIntent pIntent = PendingIntent.getActivity(ctx, widgetID,
        configIntent, 0);
    widgetView.setOnClickPendingIntent(R.id.tvPressConfig, pIntent);
    // Обновление виджета (вторая зона)
    Intent updateIntent = new Intent(ctx, MyWidget.class);
    updateIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
    updateIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
        new int[] { widgetID });
    pIntent = PendingIntent.getBroadcast(ctx, widgetID, updateIntent, 0);
    widgetView.setOnClickPendingIntent(R.id.tvPressUpdate, pIntent);
}

```

```
// Счетчик нажатий (третья зона)
```

```
Intent countIntent = new Intent(ctx, MyWidget.class);  
countIntent.setAction(ACTION_CHANGE);  
countIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);  
pIntent = PendingIntent.getBroadcast(ctx, widgetID, countIntent, 0);  
widgetView.setOnClickListener(R.id.tvPressCount, pIntent);  
// Обновляем виджет  
appWidgetManager.updateAppWidget(widgetID, widgetView); }
```

```
public void onReceive(Context context, Intent intent) {  
    super.onReceive(context, intent);  
    // Проверяем, что это intent от нажатия на третью зону  
    if (intent.getAction().equalsIgnoreCase(ACTION_CHANGE)) {  
        // извлекаем ID экземпляра  
        int mAppWidgetId = AppWidgetManager.INVALID_APPWIDGET_ID;  
        Bundle extras = intent.getExtras();  
        if (extras != null) {  
            mAppWidgetId = extras.getInt(  
                AppWidgetManager.EXTRA_APPWIDGET_ID,  
                AppWidgetManager.INVALID_APPWIDGET_ID);  
        }  
        if (mAppWidgetId != AppWidgetManager.INVALID_APPWIDGET_ID) {  
            // Читаем значение счетчика, увеличиваем на 1 и записываем  
            SharedPreferences sp = context.getSharedPreferences(  
                ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE);  
            int cnt = sp.getInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId, 0);  
            sp.edit().putInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId,  
                ++cnt).commit();  
  
            // Обновляем виджет  
            updateWidget(context, AppWidgetManager.getInstance(context),  
                mAppWidgetId);  
        }  
    }  
}
```



*//Kotlin*

```
import android.app.PendingIntent
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.content.SharedPreferences.Editor
import android.widget.RemoteViews
import sun.invoke.util.VerifyAccess.getPackageName
import java.text.SimpleDateFormat
import java.util.*

class MyWidget : AppWidgetProvider() {
    override fun onUpdate(context: Context?, appWidgetManager: AppWidgetManager?, appWidgetIds:
IntArray) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        // обновляем все экземпляры
        for (i in appWidgetIds) {
            updateWidget(context, appWidgetManager, i)
        }
    }
    override fun onDeleted(context: Context, appWidgetIds: IntArray) {
        super.onDeleted(context, appWidgetIds)
        // Удаляем Preferences
        val editor: Editor = context.getSharedPreferences(ConfigActivity.WIDGET_PREF,
Context.MODE_PRIVATE).edit()
        for (widgetID in appWidgetIds) {
            editor.remove(ConfigActivity.WIDGET_TIME_FORMAT + widgetID)
            editor.remove(ConfigActivity.WIDGET_COUNT + widgetID)
        }
        editor.commit()
    }
}
```

```

companion object { const val ACTION_CHANGE = "com.example.user.wid3.change_count" }
fun updateWidget(ctx: Context?, appWidgetManager: AppWidgetManager?, widgetID: Int) {
    val sp: SharedPreferences = ctx?.getSharedPreferences(ConfigActivity.WIDGET_PREF,
Context.MODE_PRIVATE)!!
    // Читаем формат времени и определяем текущее время
    val timeFormat = sp.getString(ConfigActivity.WIDGET_TIME_FORMAT + widgetID, null) ?: return
    val sdf = SimpleDateFormat(timeFormat)
    val currentTime: String = sdf.format(Date(System.currentTimeMillis()))
    // Читаем счетчик
    val count = sp.getInt(ConfigActivity.WIDGET_COUNT + widgetID, 0).toString()
    // Помещаем данные в текстовые поля
    val widgetView = RemoteViews(ctx.getPackageName(), R.layout.widget)
    widgetView.setTextViewText(R.id.tvTime, currentTime)
    widgetView.setTextViewText(R.id.tvCount, count)
    // Конфигурационный экран (первая зона)
    val configIntent = Intent(ctx, ConfigActivity::class.java)
    configIntent.action = AppWidgetManager.ACTION_APPWIDGET_CONFIGURE
    configIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID)
    var pIntent = PendingIntent.getActivity(ctx, widgetID, configIntent, 0)
    widgetView.setOnClickPendingIntent(R.id.tvPressConfig, pIntent)
    // Обновление виджета (вторая зона)
    val updateIntent = Intent(ctx, MyWidget::class.java)
    updateIntent.action = AppWidgetManager.ACTION_APPWIDGET_UPDATE
    updateIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, intArrayOf(widgetID))
    pIntent = PendingIntent.getBroadcast(ctx, widgetID, updateIntent, 0)
    widgetView.setOnClickPendingIntent(R.id.tvPressUpdate, pIntent)

```

```

// Счетчик нажатий (третья зона)
val countIntent = Intent(ctx, MyWidget::class.java)
countIntent.action = ACTION_CHANGE
countIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID)
pIntent = PendingIntent.getBroadcast(ctx, widgetID, countIntent, 0)
widgetView.setOnClickListener(R.id.tvPressCount, pIntent)
// Обновляем виджет
appWidgetManager!!.updateAppWidget(widgetID, widgetView)
}

override fun onReceive(context: Context, intent: Intent) {
    super.onReceive(context, intent)
    // Проверяем, что это intent от нажатия на третью зону
    if (intent.action.equals(ACTION_CHANGE, ignoreCase = true)) { // извлекаем ID
экземпляра
        var mAppWidgetId = AppWidgetManager.INVALID_APPWIDGET_ID
        val extras = intent.extras
        if (extras != null) { mAppWidgetId =
extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID, AppWidgetManager.INVALID_APPWIDGET_ID) }
        if (mAppWidgetId != AppWidgetManager.INVALID_APPWIDGET_ID) { // Читаем значение
счетчика, увеличиваем на 1 и записываем
            val sp: SharedPreferences =
context.getSharedPreferences(ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE)
            var cnt = sp.getInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId, 0)
            sp.edit().putInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId, ++cnt).commit()
            // Обновляем виджет
            updateWidget(context, AppWidgetManager.getInstance(context), mAppWidgetId)
        }
    }
}
}
}

```

В **onUpdate** мы обновляем все требующие обновления экземпляры

```
for (int i : appWidgetIds) {  
    updateWidget(context, appWidgetManager, i);}
```

в **onDelete** очищаем Preferences после удаления экземпляров.

```
Editor editor = context.getSharedPreferences(  
    ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE).edit();  
for (int widgetID : appWidgetIds) {  
    editor.remove(ConfigActivity.WIDGET_TIME_FORMAT + widgetID);  
    editor.remove(ConfigActivity.WIDGET_COUNT + widgetID);  
}  
editor.commit();
```

Метод **updateWidget** отвечает за обновления конкретного экземпляра виджета. Здесь мы настраиваем внешний вид и реакцию на нажатие.

Сначала мы читаем настройки формата времени (которые были сохранены в конфигурационном экране), берем текущее время и конвертируем в строку согласно формату.

```
String timeFormat = sp.getString(ConfigActivity.WIDGET_TIME_FORMAT + widgetID, null);  
if (timeFormat == null) return;  
SimpleDateFormat sdf = new SimpleDateFormat(timeFormat);  
String currentTime = sdf.format(new Date(System.currentTimeMillis()));
```

Также из настроек читаем значение счетчика.

```
String count = String.valueOf(sp.getInt(ConfigActivity.WIDGET_COUNT + widgetID, 0));
```

Создаем RemoteViews и помещаем время и счетчик в соответствующие TextView.

```
RemoteViews widgetView = new RemoteViews(ctx.getPackageName(), R.layout.widget);  
widgetView.setTextViewText(R.id.tvTime, currentTime);  
widgetView.setTextViewText(R.id.tvCount, count);
```

Далее идет настройка обработки нажатия. Механизм несложен. Сначала мы готовим Intent, который содержит в себе некие данные и знает куда он должен отправиться. Этот Intent мы упаковываем в PendingIntent. Далее конкретному view-компоненту мы методом `setOnClickListenerPendingIntent` сопоставляем PendingIntent. И когда будет совершено нажатие на этот view, система достанет Intent из PendingIntent и отправит его по назначению.

В нашем виджете есть три зоны для нажатия. Для каждой из них мы формируем отдельный Intent и PendingIntent.

Первая зона – по нажатию должно открываться конфигурационное Activity. Создаем Intent, который будет вызывать наше Activity, помещаем данные об ID (чтобы экран знал, какой экземпляр он настраивает), упаковываем в PendingIntent и сопоставляем view-компоненту первой зоны.

```
Intent configIntent = new Intent(ctx, ConfigActivity.class);
configIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_CONFIGURE);
configIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
PendingIntent pIntent = PendingIntent.getActivity(ctx, widgetID, configIntent, 0);
widgetView.setOnClickListener(R.id.tvPressConfig, pIntent);
```

Вторая зона – по нажатию должен обновляться виджет, на котором было совершено нажатие. Создаем Intent, который будет вызывать наш класс виджета, добавляем ему action = ACTION\_APPWIDGET\_UPDATE, помещаем данные об ID (чтобы обновился именно этот экземпляр), упаковываем в PendingIntent и сопоставляем view-компоненту второй зоны.

```
Intent updateIntent = new Intent(ctx, MyWidget.class);
updateIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
updateIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, new int[] { widgetID });
pIntent = PendingIntent.getBroadcast(ctx, widgetID, updateIntent, 0);
widgetView.setOnClickListener(R.id.tvPressUpdate, pIntent);
```

Третья зона – по нажатию должен увеличиваться на единицу счетчик нажатий. Создаем Intent, который будет вызывать наш класс виджета, добавляем ему наш собственный action = ACTION\_CHANGE, помещаем данные об ID (чтобы работать со счетчиком именно этого экземпляра), упаковываем в PendingIntent и сопоставляем view-компоненту третьей зоны.

```
Intent countIntent = new Intent(ctx, MyWidget.class);
countIntent.setAction(ACTION_CHANGE);
countIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, widgetID);
pIntent = PendingIntent.getBroadcast(ctx, widgetID, countIntent, 0);
widgetView.setOnClickListener(R.id.tvPressCount, pIntent);
```

Теперь при нажатии на первую зону будет вызван конфигурационный экран. По нажатию на вторую будет обновлен виджет. А вот нажатие на третью ни к чему не приведет, т.к. наш класс MyWidget знает, как работать с Intent с action вида ACTION\_APPWIDGET\_UPDATE, ACTION\_APPWIDGET\_DELETED и пр. А мы ему послали свой action.

Значит надо научить его понимать наш Intent. Вспоминаем, что MyWidget – это расширение AppWidgetProvider, а AppWidgetProvider – это расширение BroadcastReceiver. А значит, мы можем сами реализовать метод onReceive, в котором будем отслеживать наш action и выполнять нужные нам действия.

В методе **onReceive** мы обязательно выполняем метод onReceive родительского класса, иначе просто перестанут работать обновления и прочие стандартные события виджета.

```
super.onReceive(context, intent);
```

Далее мы проверяем, что intent содержит наш action, читаем и проверяем ID из него

```
if (intent.getAction().equalsIgnoreCase(ACTION_CHANGE)) {  
  
    int mAppWidgetId = AppWidgetManager.INVALID_APPWIDGET_ID;  
    Bundle extras = intent.getExtras();  
    if (extras != null) {  
        mAppWidgetId = extras.getInt(  
            AppWidgetManager.EXTRA_APPWIDGET_ID,  
            AppWidgetManager.INVALID_APPWIDGET_ID);  
    }  
}
```

читаем из настроек значение счетчика, увеличиваем на единицу, пишем обратно в настройки

```
SharedPreferences sp = context.getSharedPreferences(  
    ConfigActivity.WIDGET_PREF, Context.MODE_PRIVATE);  
int cnt = sp.getInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId, 0);  
sp.edit().putInt(ConfigActivity.WIDGET_COUNT + mAppWidgetId, ++cnt).commit();
```

и обновляем экземпляр виджета. Он прочтет новое значение счетчика из настроек и отобразит его.

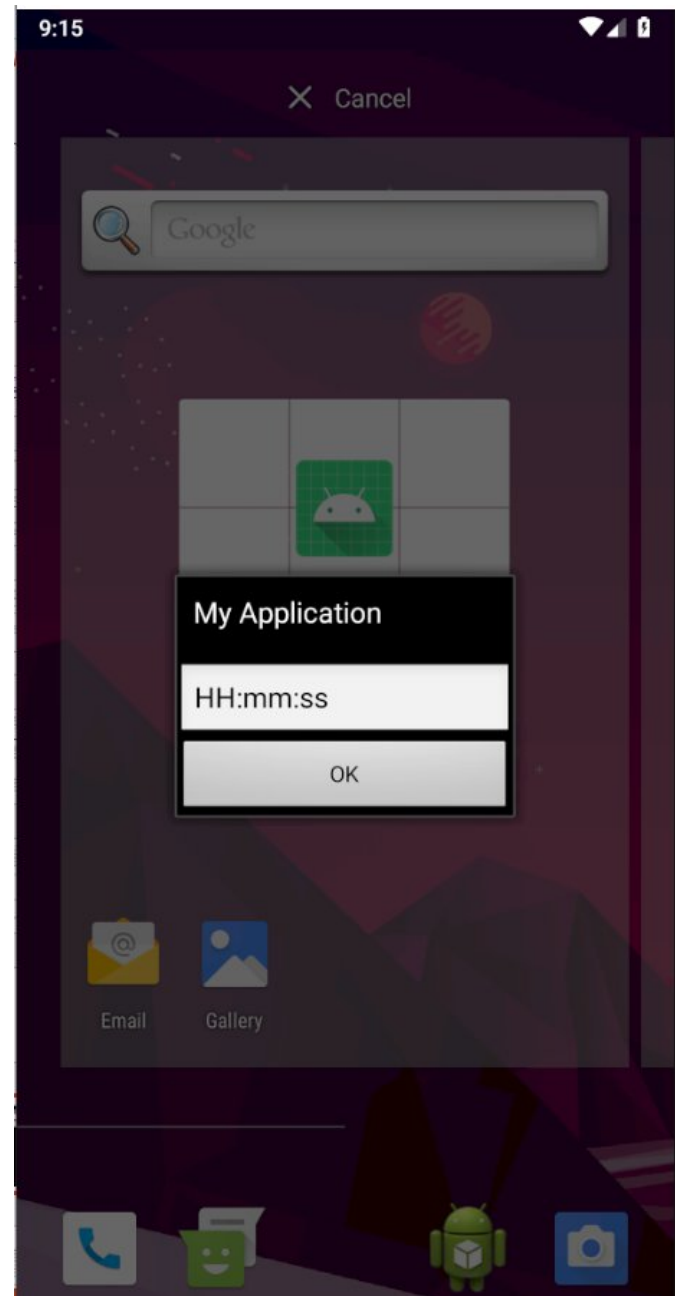
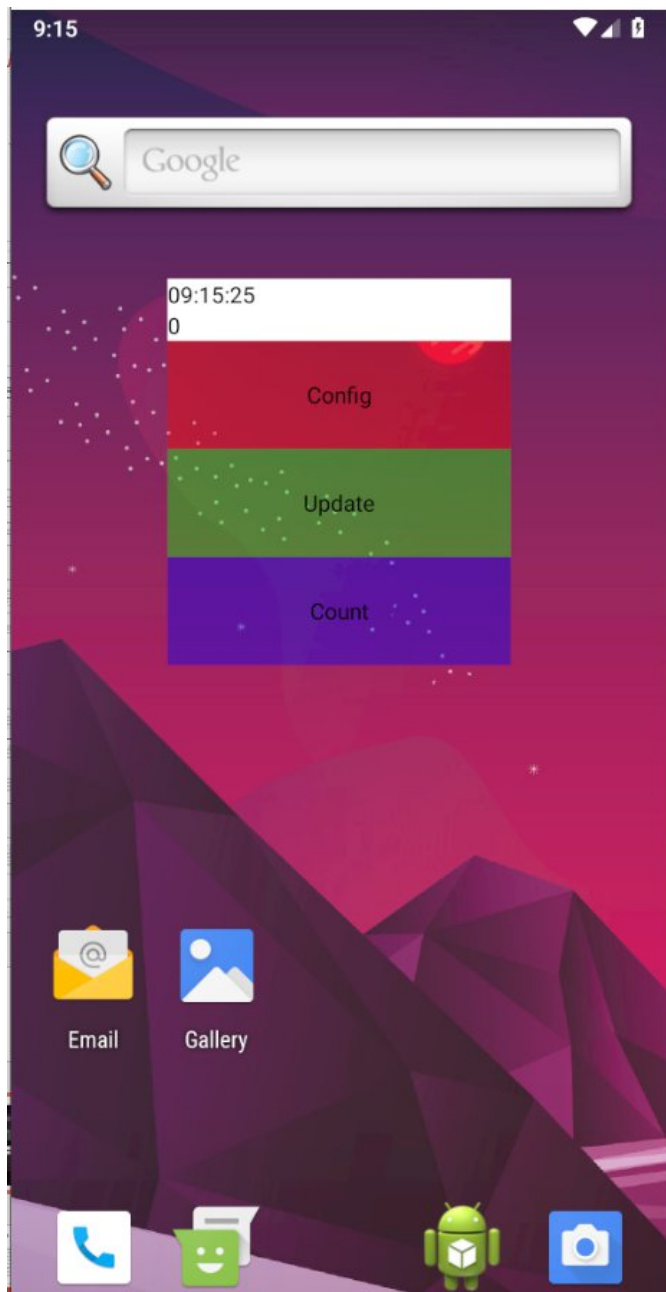
```
updateWidget(context, AppWidgetManager.getInstance(context), mAppWidgetId);
```

Создадим файл метаданных `xml/widget_metadata.xml`:

Виджет будет вертикальным. Число 0 – в `updatePeriodMillis` говорит о том, что виджет не будет обновляться системой. Мы его сами обновлять будем.

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="com.example.user.wid3.ConfigActivity"
    android:minWidth="70dip"
    android:minHeight="140dip"
    android:updatePeriodMillis="0"
    android:initialLayout="@layout/widget"
  />
```





# Виджет со списком

В третьей версии Android у виджетов появилась возможность работать с наборами данных типа списка или грида. Рассмотрим эту технологию на примере списка. В качестве view-компонента используется обычный ListView. Для межпроцессной работы с ним используется, как обычно в виджетах, RemoteViews. Но для заполнения нам придется создать два класса в дополнение к стандартному классу провайдера.

Первый – этот класс будет наполнять наш список значениями. Класс является реализацией интерфейса RemoteViewsService.RemoteViewsFactory, и его методы очень схожи с методами стандартного адаптера. Его обычно везде называют factory (адаптер).

Второй – класс сервиса, наследующий RemoteViewsService. В нем мы реализуем только один метод, который будет создавать и возвращать экземпляр (первого) класса, который будет заполнять список.

При создании и работе со списком в виджете необходимо понимать, как реализованы два момента: заполнение данными и реакция на нажатия.

При подготовке виджета в классе провайдера мы для списка присваиваем Intent, который содержит данные для вызова нашего второго класса-сервиса. Когда система хочет обновить данные в списке (в виджете) она достаёт этот интент, биндится к указанному сервису и берет у него адаптер. И этот адаптер уже используется для наполнения и формирования пунктов списка.

Теперь о реализации нажатий на пункты списка. В обычном виджете использовались PendingIntent. Здесь чуть по-другому. Для каждого пункта в списке НЕ создается свой отдельный PendingIntent. Вместо этого списку дается общий, шаблонный PendingIntent. А для каждого пункта списка мы указываем отдельный Intent с extra-данными. Далее, при создании, каждому пункту списка система присваивает обработчик нажатия, который при срабатывании берет этот общий PendingIntent, добавляет к нему данные из персонального Intent, и отправляет по назначению сформированный таким образом PendingIntent. Т.е. в итоге по клику все равно срабатывает PendingIntent.

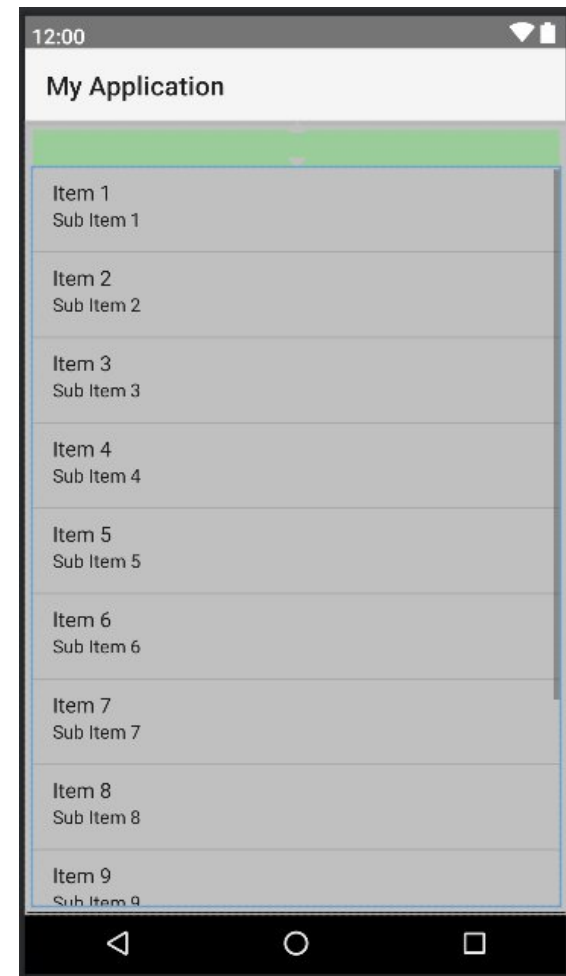
## Создаем layout-виджета - **widget.xml** и **item.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#9999"
    android:padding="5dp">
    <TextView
        android:id="@+id/tvUpdate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="#3300ff00"
        android:gravity="center"

        android:textAppearance="?android:attr/textAppearanceLarge">
    </TextView>
    <ListView
        android:id="@+id/lvList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/tvUpdate">
    </ListView>
</RelativeLayout>
```

Текст будет использован для отображения времени обновления. Он же собственно и будет кнопкой обновления. В списке будем показывать данные.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tvItemText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium">
    </TextView>
</RelativeLayout>
```



## Создадим класс MyFactory.java

**//Java**

```
import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import android.appwidget.AppWidgetManager;
import android.content.Context;
import android.content.Intent;
import android.widget.RemoteViews;
import android.widget.RemoteViewsService.RemoteViewsFactory;

public class MyFactory implements RemoteViewsFactory {
    ArrayList<String> data;
    Context context;
    SimpleDateFormat sdf;
    int widgetID;
    MyFactory(Context ctx, Intent intent) {
        context = ctx;
        sdf = new SimpleDateFormat("HH:mm:ss");
        widgetID = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
    }
    @Override
    public void onCreate() {
        data = new ArrayList<String>();
    }
    @Override
    public int getCount() {
        return data.size();
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
    @Override
    public RemoteViews getLoadingView() {
        return null;
    }
}
```

```

@Override
public RemoteViews getViewAt(int position) {
    RemoteViews rView = new RemoteViews(context.getPackageName(), R.layout.item);
    rView.setTextViewText(R.id.tvItemText, data.get(position));
    Intent clickIntent = new Intent();
    clickIntent.putExtra(MyProvider.ITEM_POSITION, position);
    rView.setOnClickFillInIntent(R.id.tvItemText, clickIntent);

    return rView; }

@Override
public int getViewTypeCount() {
    return 1;
}

@Override
public boolean hasStableIds() {
    return true;
}

@Override
public void onDataSetChanged() {
    data.clear();
    data.add(sdf.format(new Date(System.currentTimeMillis())));
    data.add(String.valueOf(hashCode()));
    data.add(String.valueOf(widgetID));
    for (int i = 3; i < 15; i++) {
        data.add("Item " + i);
    }
}

@Override
public void onDestroy() {
}
}

```

*//Kotlin*

```
import android.appwidget.AppWidgetManager
import android.content.Context
import android.content.Intent
import android.widget.RemoteViews
import android.widget.RemoteViewsService.RemoteViewsFactory
import java.text.SimpleDateFormat
import java.util.*

class MyFactory internal constructor(var context: Context, intent: Intent) :
    RemoteViewsFactory {
    var data: ArrayList<String>? = null
    var sdf: SimpleDateFormat
    var widgetID: Int
    override fun onCreate() {
        data = ArrayList()
    }

    override fun getCount(): Int {
        return data!!.size
    }

    override fun getItemId(position: Int): Long {
        return position.toLong()
    }

    override fun getLoadingView(): RemoteViews {
        return null
    }
}
```

```

init {
    sdf = SimpleDateFormat("HH:mm:ss")
    widgetID = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
AppWidgetManager.INVALID_APPWIDGET_ID)
}

override fun getViewAt(position: Int): RemoteViews? {
    val rView = RemoteViews(context.packageName, R.layout.item)
    rView.setTextViewText(R.id.tvItemText, data!![position])
    val clickIntent = Intent()
    clickIntent.putExtra(MyProvider.ITEM_POSITION, position)
    rView.setOnClickFillInIntent(R.id.tvItemText, clickIntent)
    return rView
}

override fun getViewTypeCount(): Int {
    return 1
}

override fun hasStableIds(): Boolean {
    return true
}

override fun onDataSetChanged() {
    data!!.clear()
    data!!.add(sdf.format(Date(System.currentTimeMillis())))
    data!!.add(hashCode().toString())
    data!!.add(widgetID.toString())
    for (i in 3..14) { data!!.add("Item $i") }
}

override fun onDestroy() {}
}

```



**MyFactory** – конструктор с двумя параметрами – Context и Intent. Этот Intent будет передавать нам сервис при создании адаптера. В нем передается адаптеру ID виджета.

```
widgetID = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,  
    AppWidgetManager.INVALID_APPWIDGET_ID);
```

onCreate– создание адаптера. вызывается, когда он создается для первого своего списка

```
data = new ArrayList<String>();
```

getLoadingView – здесь вам предлагается возвращать View, которое система будет показывать вместо пунктов списка, пока они создаются. Если ничего здесь не создавать, то система использует некое дефолтное View.

getViewAt – создание пунктов списка. Здесь идет стандартное использование RemoteViews

```
RemoteViews rView = new RemoteViews(context.getPackageName(), R.layout.item);  
rView.setTextViewText(R.id.tvItemText, data.get(position));
```

onDataSetChanged – вызывается, когда поступил запрос на обновление данных в списке. Т.е. в этом методе мы подготавливаем данные для списка. Метод заточен под выполнение тяжелого долгого кода. В трех первых пунктах списка мы выводим текущее время, хэш-код адаптера и ID-виджета.

```
data.clear();  
data.add(sdf.format(new Date(System.currentTimeMillis())));  
data.add(String.valueOf(hashCode()));  
data.add(String.valueOf(widgetID));  
for (int i = 3; i < 15; i++) {  
    data.add("Item " + i);}
```

[onDestroy](#) – вызывается при удалении последнего списка, который использовал адаптер (один адаптер может использоваться несколькими списками).

Для каждого пункта списка мы создаем Intent, помещаем в него позицию пункта и вызываем `setOnClickListener`. Этот метод получает на вход ID View и Intent. Для View с полученным на вход ID он создает обработчик нажатия, который будет вызывать `PendingIntent`, который получается следующим образом: берется шаблонный `PendingIntent`, который был привязан к списку методом `setPendingIntentTemplate` (в классе провайдера) и к нему добавляются данные полученного на вход Intent-a. Т.е. получится `PendingIntent`, Intent которого будет содержать `action = ACTION_ON_CLICK` (это мы сделали еще в провайдере) и данные по позиции пункта списка. При нажатии на пункт списка, этот Intent попадет в `onReceive` нашего `MyProvider` и будет обработан

```
public RemoteViews getViewAt(int position) {  
    RemoteViews rView = new RemoteViews(context.getPackageName(), R.layout.item);  
    rView.setTextViewText(R.id.tvItemText, data.get(position));  
    Intent clickIntent = new Intent();  
    clickIntent.putExtra(MyProvider.ITEM_POSITION, position);  
    rView.setOnClickListener(R.id.tvItemText, clickIntent);  
  
    return rView; }  
}
```

## Создаем сервис – **MyService.java**

```
package com.example.user.widlist;
import android.content.Intent;
import android.widget.RemoteViewsService;
public class MyService extends RemoteViewsService {
    @Override
    public RemoteViewsFactory onGetViewFactory(Intent intent) {
        return new MyFactory(getApplicationContext(), intent);
    }
}
```

В нем мы просто реализуем метод `onGetViewFactory`, который создает адаптер, передает ему `Context` и `Intent`, и возвращает этот созданный адаптер системе.

## Класс провайдер – **MyProvider.java**

```
package com.example.user.widlist;
import java.sql.Date;
import java.text.SimpleDateFormat;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.widget.RemoteViews;

public class MyProvider extends AppWidgetProvider {

    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
    final String ACTION_ON_CLICK = "ru.startandroid.develop.p1211listwidget.itemonclick";
    final static String ITEM_POSITION = "item_position";

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                        int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
        for (int i : appWidgetIds) {
            updateWidget(context, appWidgetManager, i);
        }
    }
}
```

```

void updateWidget(Context context, AppWidgetManager appWidgetManager,
                  int appWidgetId) {
    RemoteViews rv = new RemoteViews(context.getPackageName(),
                                     R.layout.widget);

    setUpdateTV(rv, context, appWidgetId);
    setList(rv, context, appWidgetId);
    setListClick(rv, context, appWidgetId);
    appWidgetManager.updateAppWidget(appWidgetId, rv);
    appWidgetManager.notifyAppWidgetViewDataChanged(appWidgetId, R.id.lvList);
}

```

```

void setUpdateTV(RemoteViews rv, Context context, int appWidgetId) {
    rv.setText(R.id.tvUpdate,
              sdf.format(new Date(System.currentTimeMillis())));
    Intent updIntent = new Intent(context, MyProvider.class);
    updIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
    updIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
                      new int[] { appWidgetId });
    PendingIntent updPIntent = PendingIntent.getBroadcast(context,
                                                            appWidgetId, updIntent, 0);
    rv.setOnClickPendingIntent(R.id.tvUpdate, updPIntent);
}

```

```

void setList(RemoteViews rv, Context context, int appWidgetId) {
    Intent adapter = new Intent(context, MyService.class);
    adapter.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);
    Uri data = Uri.parse(adapter.toUri(Intent.URI_INTENT_SCHEME));
    adapter.setData(data);
    rv.setRemoteAdapter(R.id.lvList, adapter);
}

```

```
void setListClick(RemoteViews rv, Context context, int appId) {  
    Intent listClickIntent = new Intent(context, MyProvider.class);  
    listClickIntent.setAction(ACTION_ON_CLICK);  
    PendingIntent listClickPIntent =  
PendingIntent.getBroadcast(context, 0, listClickIntent, 0);  
    rv.setPendingIntentTemplate(R.id.lvList, listClickPIntent);  
}
```

@Override

```
public void onReceive(Context context, Intent intent) {  
    super.onReceive(context, intent);  
    if (intent.getAction().equalsIgnoreCase(ACTION_ON_CLICK)) {  
        int itemPos = intent.getIntExtra(ITEM_POSITION, -1);  
        if (itemPos != -1) {  
            Toast.makeText(context, "Clicked on item " +  
itemPos, Toast.LENGTH_SHORT).show();  
        }  
    }  
}
```

```
//MyService.kt
```

```
import android.content.Intent
import android.widget.RemoteViewsService
import androidx.test.core.app.ApplicationProvider.getApplicationContext
```

```
class MyService : RemoteViewsService() {
    override fun onGetViewFactory(intent: Intent): RemoteViewsFactory {return MyFactory(applicationContext,
    intent) }
}
```

```
//MyProvider.kt
```

```
import android.app.PendingIntent
import android.appwidget.AppWidgetManager
import android.appwidget.AppWidgetProvider
import android.content.Context
import android.content.Intent
import android.widget.RemoteViews
import java.text.SimpleDateFormat
import java.util.*
```

```
class MyProvider : AppWidgetProvider() {
    var sdf: SimpleDateFormat = SimpleDateFormat("HH:mm:ss")
    override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager, appWidgetIds:
IntArray) {
        super.onUpdate(context, appWidgetManager, appWidgetIds)
        for (i in appWidgetIds) { updateWidget(context, appWidgetManager, i) }
    }
```

```
fun updateWidget(context: Context, appWidgetManager: AppWidgetManager, appWidgetId: Int) {
    val rv = RemoteViews(context.packageName, R.layout.widget)
    setUpdateTV(rv, context, appWidgetId)
    setList(rv, context, appWidgetId)
    setListClick(rv, context, appWidgetId)
    appWidgetManager.updateAppWidget(appWidgetId, rv)
}
```

```

fun setUpdateTV(rv: RemoteViews, context: Context?, appWidgetId: Int) {
    rv.setTextViewText(R.id.tvUpdate, sdf.format(Date(System.currentTimeMillis())))
    val updIntent = Intent(context, MyProvider::class.java)
    updIntent.action = AppWidgetManager.ACTION_APPWIDGET_UPDATE
    updIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, intArrayOf(appWidgetId))
    val updPIntent = PendingIntent.getBroadcast(context, appWidgetId, updIntent, 0)
    rv.setOnClickPendingIntent(R.id.tvUpdate, updPIntent)
}

fun setList(rv: RemoteViews, context: Context?, appWidgetId: Int) {
    val adapter = Intent(context, MyService::class.java)
    adapter.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId)
    rv.setRemoteAdapter(R.id.lvList, adapter)
}

fun setListClick(rv: RemoteViews?, context: Context?, appWidgetId: Int) {}

}

```

**onUpdate** вызывается, когда поступает запрос на обновление виджетов. В нем мы перебираем ID, и для каждого вызываем метод `updateWidget`.

```
super.onUpdate(context, appWidgetManager, appWidgetIds);  
for (int i : appWidgetIds) {  
    updateWidget(context, appWidgetManager, i);  
}
```

**updateWidget** – здесь вызываем три метода для формирования виджета

```
setUpdateTV(rv, context, appWidgetId);  
setList(rv, context, appWidgetId);  
setListClick(rv, context, appWidgetId);
```

и затем метод `updateAppWidget`, чтобы применить все изменения к виджету.

```
appWidgetManager.updateAppWidget(appWidgetId, rv);  
appWidgetManager.notifyAppWidgetViewDataChanged(appWidgetId, R.id.lvList);
```

**setUpdateTV** – в этом методе работаем с `TextView` (который над списком). Ставим ему время в качестве текста

```
rv.setText(R.id.tvUpdate, sdf.format(new Date(System.currentTimeMillis())));
```

и добавляем обновление виджета по нажатию.

```
Intent updIntent = new Intent(context, MyProvider.class);  
updIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);  
updIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, new int[] { appWidgetId });  
PendingIntent updPIntent = PendingIntent.getBroadcast(context, appWidgetId, updIntent, 0);  
rv.setOnClickListener(R.id.tvUpdate, updPIntent);
```

**setList** – с помощью метода `setRemoteAdapter` указываем списку, что для получения адаптера ему надо будет обратиться к нашему сервису `MyService`.

Также в `Intent` мы помещаем ID виджета. Этот `Intent` будет передан в метод сервиса `onGetViewFactory`. Этот метод мы реализовывали, в нем мы создаем адаптер и передаем ему тот же `Intent`. А уже в адаптере достаем этот ID и используем (третья строка в списке).



Т.е. этот Intent пройдет через сервис и попадет в адаптер, поэтому если хотите что-то передать адаптеру, используйте этот Intent.

```
Intent adapter = new Intent(context, MyService.class);
adapter.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);
Uri data = Uri.parse(adapter.toUri(Intent.URI_INTENT_SCHEME));
adapter.setData(data);
rv.setRemoteAdapter(R.id.lvList, adapter);
```

**setListClick** – с помощью метода `setPendingIntentTemplate` устанавливаем шаблонный `PendingIntent`, который затем будет использоваться всеми пунктами списка. В нем мы указываем, что необходимо будет вызвать наш класс провайдера (он же `BroadcastReceiver`) с `action = ACTION_ON_CLICK`.

```
Intent listClickIntent = new Intent(context, MyProvider.class);
listClickIntent.setAction(ACTION_ON_CLICK);
PendingIntent listClickPIntent = PendingIntent.getBroadcast(context, 0, listClickIntent, 0);
rv.setPendingIntentTemplate(R.id.lvList, listClickPIntent);
```

**onRecive** - вызываем метод родителя, чтобы не нарушать работу провайдера. Далее проверяем, что `action` тот, что нам нужен - `ACTION_ON_CLICK`, получаем позицию нажатого пункта в списке и выводим сообщение на экран.

```
if (intent.getAction().equalsIgnoreCase(ACTION_ON_CLICK)) {
    int itemPos = intent.getIntExtra(ITEM_POSITION, -1);
    if (itemPos != -1) {
        Toast.makeText(context, "Clicked on item " + itemPos,
            Toast.LENGTH_SHORT).show();
    }
}
```

Создаем файл /xml/widget\_metadata.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/widget"
    android:minHeight="180dp"
    android:minWidth="110dp"
    android:updatePeriodMillis="1800000">
</appwidget-provider>
```

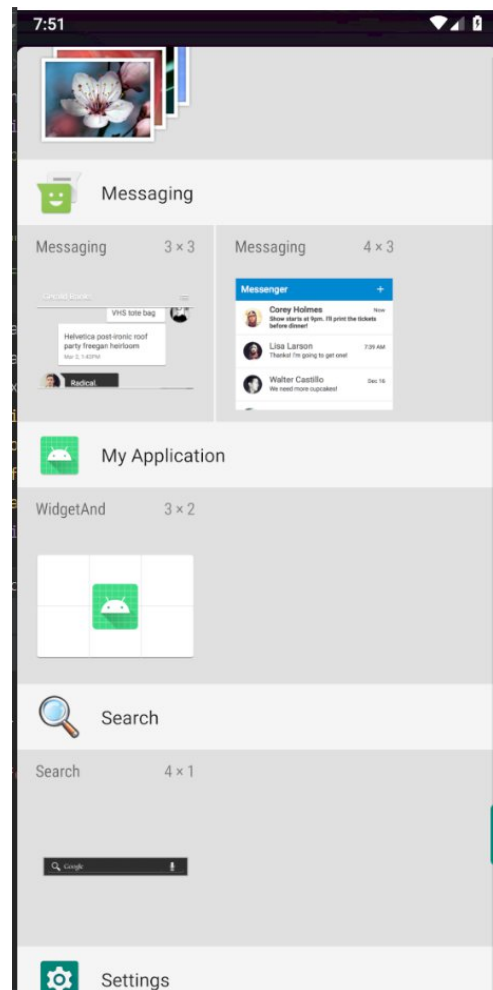
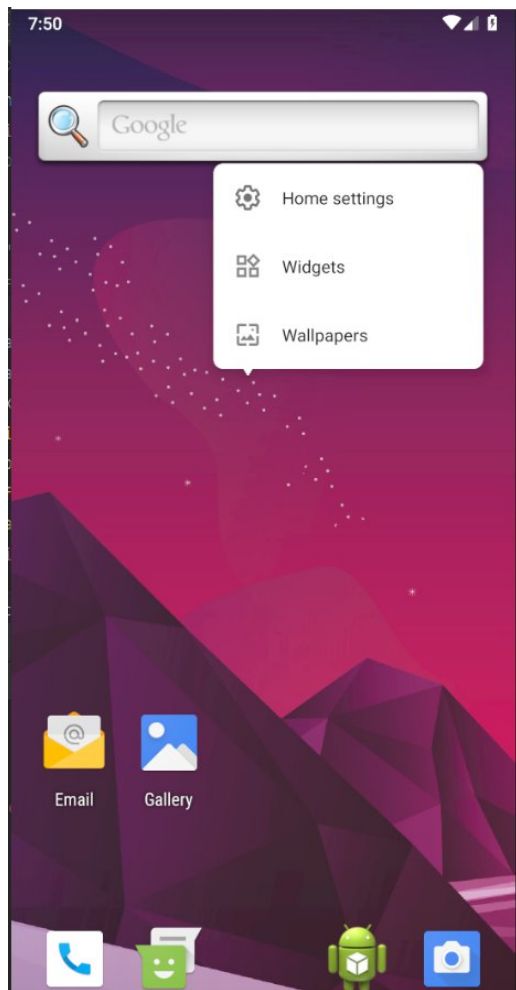
Добавляем в AndroidManifest.xml информацию о сервисе и ресивере

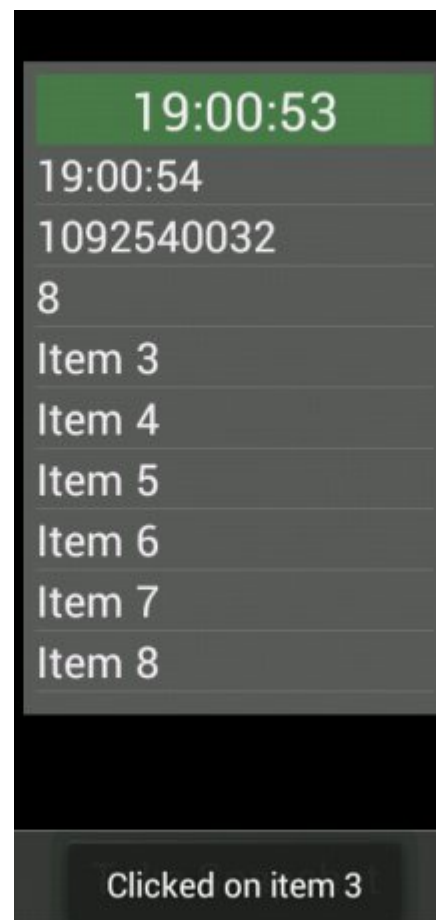
```
<service
    android:name="MyService"
    android:permission="android.permission.BIND_REMOTEVIEWS">
</service>
<receiver
    android:name="MyProvider">
    <intent-filter>
        <action
            android:name="android.appwidget.action.APPWIDGET_UPDATE">
        </action>
    </intent-filter>
    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_metadata">
    </meta-data>
</receiver>
```

Для сервиса, необходимо установить разрешение BIND\_REMOTEVIEWS. Так мы указываем, что этими полномочиями должен быть наделен тот, кто будет этот сервис вызывать. Система имеет такие полномочия, поэтому сможет использовать.

Для запуска приложения необходимо

- Нажмите и удерживайте на главном экране.
- Выберите пункт Виджеты.
- Перетащите виджет на главный экран: нажмите и удерживайте значок виджета, перетащите его в нужное место и отпустите палец.





дим время обновления виджета, время формирования данных в списке, хэш-код адаптера, ID виджета.

и нажатие на зеленую зону для обновления, время обновления виджета меняется. А при нажатие на какой-либо пункт выводится сообщение с его номером