Лекция #23. Стили и темы

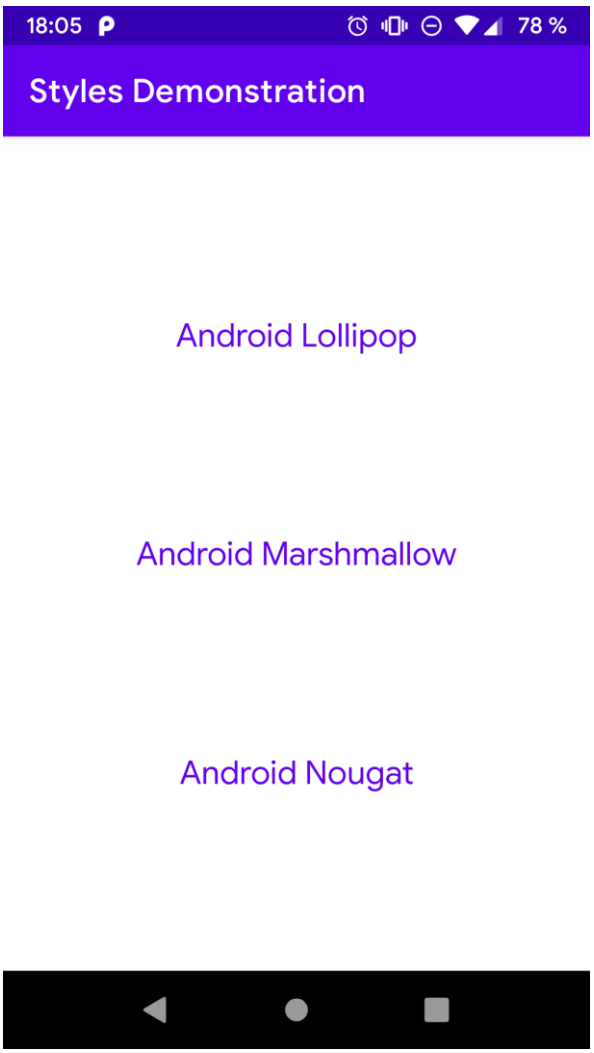
Стили

Мы можем настроить элемент с помощью различных атрибутов, которые задают высоту, ширину, цвет фона, текста и так далее. Но если у нас несколько элементов используют одни и те же настройки, то мы можем объединить эти настройки в стили.

Например, пусть у нас есть несколько элементов **TextView**:

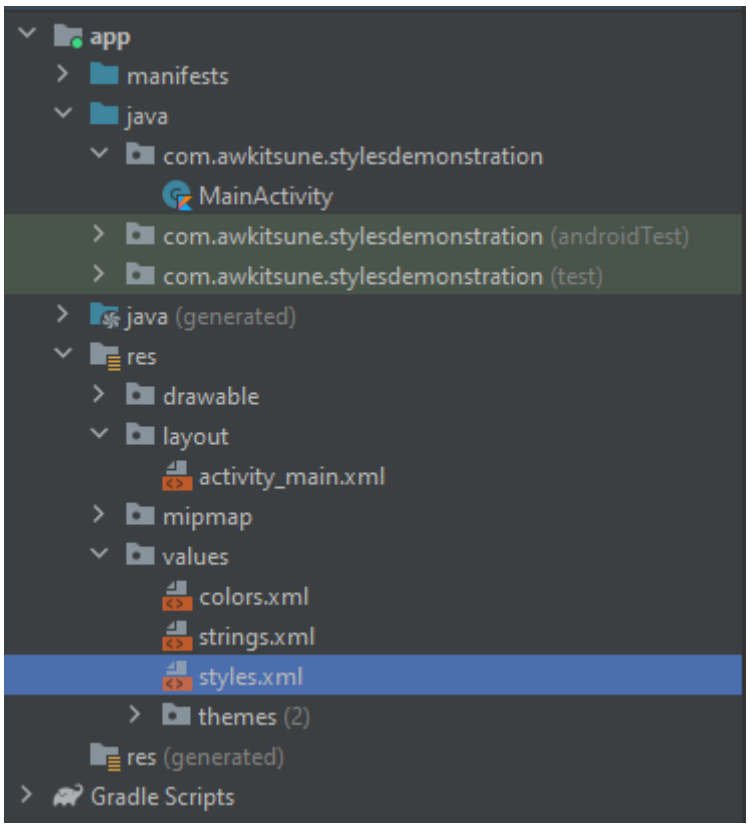
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Lollipop"
        android:textColor="@color/purple_500"
        android:textSize="20sp"

        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Marshmallow"
        android:textColor="@color/purple_500"
        android:textSize="20sp"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Nougat"
        android:textColor="@color/purple_500"
        android:textSize="20sp"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Все эти **TextView** имеют одинаковый набор свойств, и, к примеру, если нам захочется изменить цвет текста, то придется менять его у всех трех **TextView**. Данный подход не оптимален, а более оптимальный подход представляет использование стилей, которые определяются в проекте в папке **res/values**.

Итак, добавим в проект в папку **res/values** новый элемент **Value Resourse File**, который назовем **styles.xml**:



Определим в файле **styles.xml** следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="TextViewStyle">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">@color/purple_500</item>
        <item name="android:textSize">20sp</item>
        <item name="android:gravity">center</item>
    </style>
</resources>
```

Здесь определен новый стиль **TextViewStyle**, который с помощью элементов item задает значения для атрибутов TextView.

Стиль задается с помощью элемента **<style>**. Атрибут name указывает на название стиля, через которое потом можно ссылаться на него. Необязательный атрибут `parent` устанавливает для данного стиля родительский стиль, от которого дочерний стиль будет наследовать все свои характеристики. С помощью элементов **item** устанавливаются конкретные свойства виджета, который принимает в качестве значения атрибута name имя устанавливаемого свойства.

Теперь применим стиль, изменим файл **activity_main.xml:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"

        style="@style/TextViewStyle"

        android:text="Android Lollipop"

        app:layout_constraintBottom_toTopOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView2"

        style="@style/TextViewStyle"

        android:text="Android Marshmallow"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView3"

        style="@style/TextViewStyle"

        android:text="Android Nougat"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Используя определение **`style="@style/TextViewStyle"`** текстовое поле связывается с определением стиля. Итоговый результат буде тот же, что и раньше, только кода становится меньше. А если мы захотим поменять какие-

то характеристики, то достаточно изменить нужный элемент **item** в определении стиля.

Темы

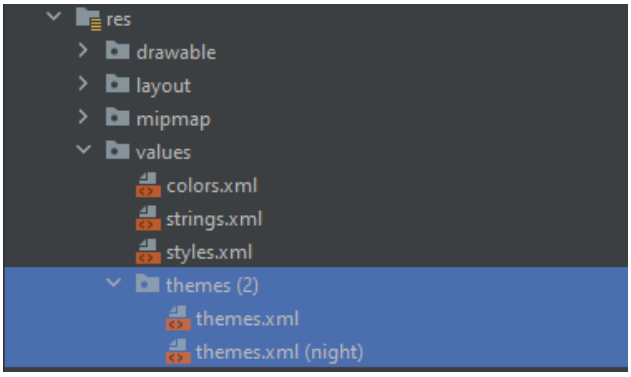
Кроме применение отдельных стилей к отдельным элементам, мы можем задавать стили для всего приложения или activity в виде тем. Тема предтавляет коллекцию атрибутов, которые применяются в целом ко всему приложению, классу activity или иерархии виджетов.

Мы можем сами создать тему. Однако Android уже предоставляет несколько предустановленных тем для стилизации приложения, например, **`Theme.AppCompat.Light.DarkActionBar`** и ряд других.

По умолчанию приложение уже применяет темы. Так, откроем файл **AndroidManifest.xml**. В нем мы можем увидеть следующее определение элемента application, представляющего приложение:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.StylesDemonstration">
```

Задание темы происходит с помощью атрибута **android:theme**. В данном случае используется ресурс, который называется в данном случае **Theme.ViewApp**. По умолчанию файлы тем определены в папке **res/values**. В частности, здесь можно найти условный каталог **themes**, в котором по умолчанию есть два элемента: **themes.xml**:



Один файл представляет светлую тему, а другой - темную. Например, откроем файл **themes.xml** со светлой темой:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.StylesDemonstration"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Здесь мы можем увидеть, что тема определяется как и стиль с помощью элемента **style**. Атрибут **parent** указывает на родительскую тему, от которой текущая тема берет все стилевые характеристики. То есть тема "**Theme.ViewApp**" использует другую тему - "**Theme.MaterialComponents.DayNight.DarkActionBar**". И кроме того, определяет ряд своих собственных стилей.

Также можно заметить, что здесь определяются не только характеристики для атрибутов, но и семантические имена, например, colorPrimary, которому сопоставлен ресурс "**@color/purple_500**".

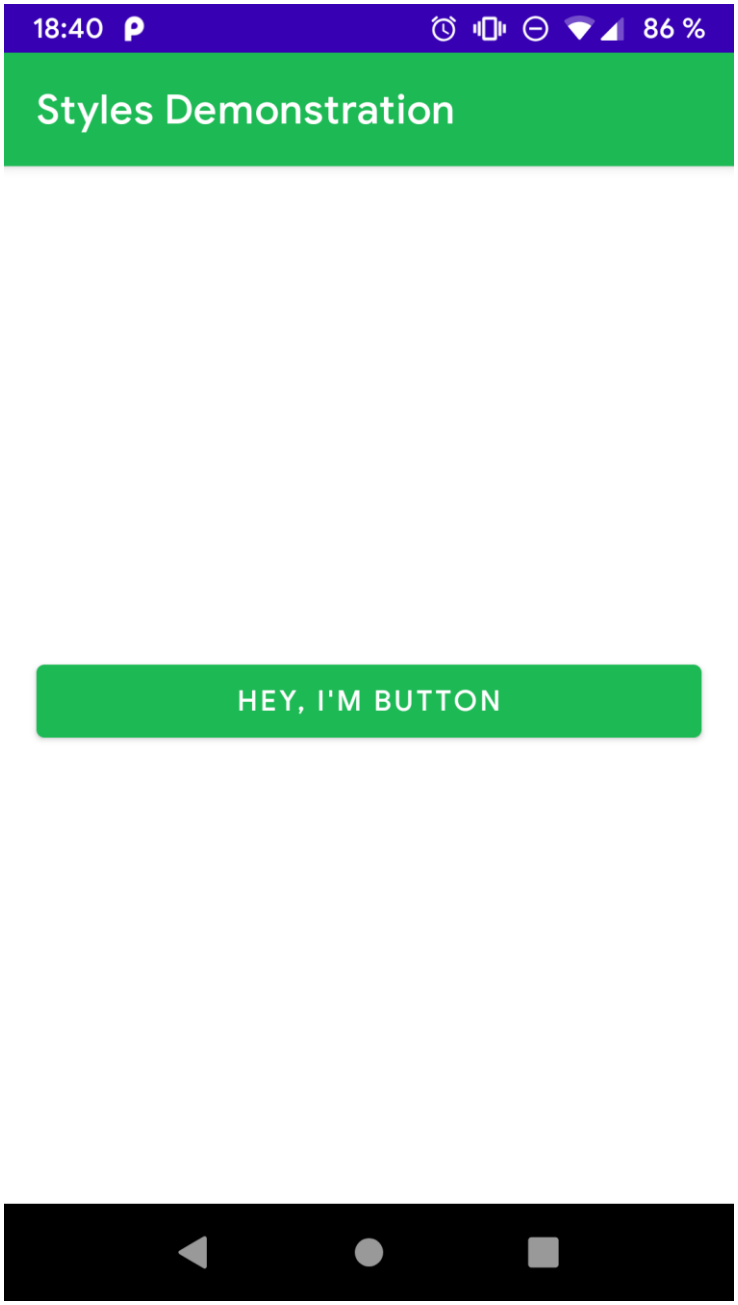
При необходимости мы можем изменить эти характеристики или дополнить тему новыми стилевыми характеристиками. Например, изменим цвет свойства **colorPrimary**, которое применяется в том числе в качестве фонового цвета заголовка и кнопки:

```
<item name="colorPrimary">#1db954</item>
```

И соответственно изменится цвет по умолчанию для фона заголовка и кнопки:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

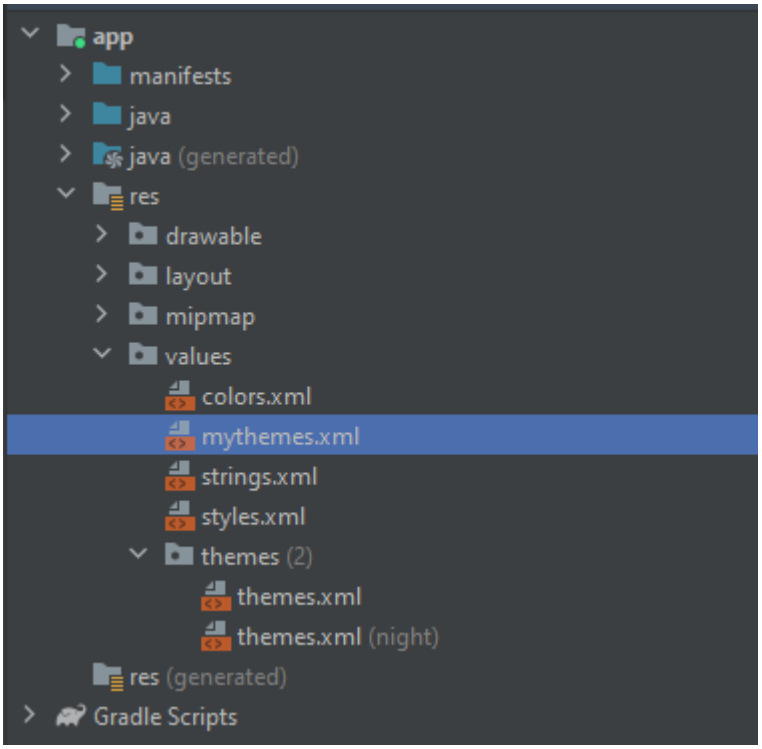
```
    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="Hey, i'm button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Создание собственной темы

Вместо использования встроенных тем мы можем создать свою. Для этого добавим в папку **res/values** новый файл **mythemes.xml** и определим в нем следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme_Light"
        parent="Theme.AppCompat.Light">
        <item name="android:textColor">#447766</item>
        <item name="android:textSize">20sp</item>
    </style>
</resources>
```



Итак, мы создали стиль "**MyTheme_Light**", который унаследован от стиля **Theme.AppCompat.Light**. В этом стиле мы переопределили два свойства: высоту шрифта (**textSize**) - **20sp**, а также цвет текста (**textColor**) - **#447766**.

Теперь определим этот стиль в качестве темы приложения в файле **AndroidManifest.xml**:

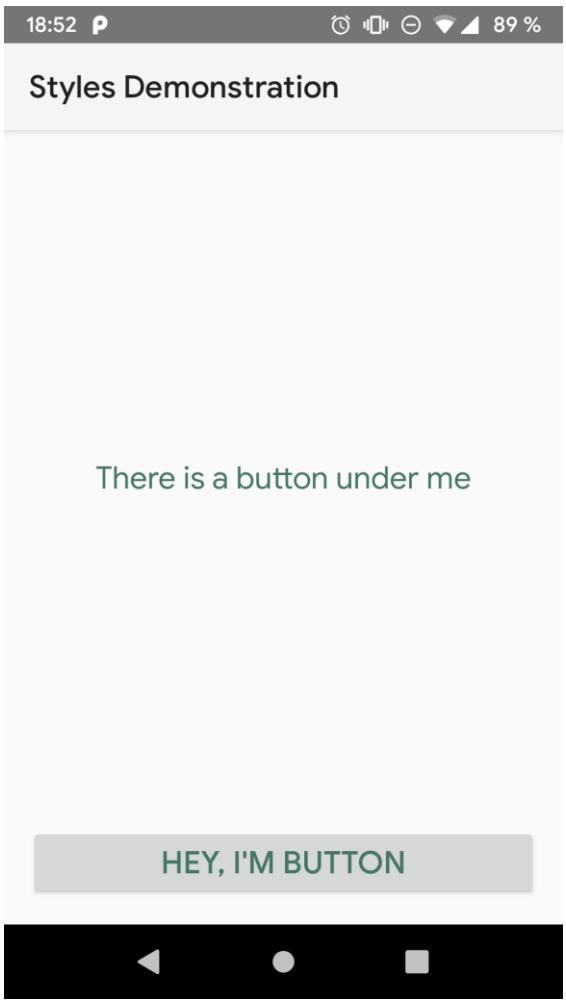
```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
```

```
    android:supportsRtl="true"
    android:theme="@style/MyTheme_Light">
```

Пусть у нас будет следующая разметка в **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="Hey, i'm button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:gravity="center"
        android:text="There is a button under me"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Как видно, для элементов **TextView** не устанавливается атрибут **textSize** и **textColor**, однако поскольку они определены в теме, которая применяется глобально к нашему приложению, то элементы **TextView** будут подхватывать эти стилевые характеристики:



Применение темы к activity

Выше темы применялись глобально ко всему приложению. Но также можно применить их к отдельному классу **Activity**. Для этого надо подкорректировать файл манифеста **AndroidManifest**. Например:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.awkitsune.stylesdemonstration">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.StylesDemonstration">
        <activity
            android:name=".MainActivity"
            android:exported="true"

            android:theme="@style/MyTheme_Light">
            <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Атрибут android:theme элемента <**activity**> указывает на применяемую к **MainActivity** тему. То есть глобально к приложению применяется тема "**Theme.ViewApp**", а к **MainActivity** - "**MyTheme**".

Применение темы к иерархии виджетов

Также можно применить тему к иерархии виджетов, установив атрибут **android:theme** у элемента, к которому (включая его вложенные элементы) мы хотим применить тему. Например, примение темы к **ConstraintLayout** и ее элементам:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/MyTheme_Light"

    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="Hey, i'm button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
```

```
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:gravity="center"
        android:text="There is a button under me"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```