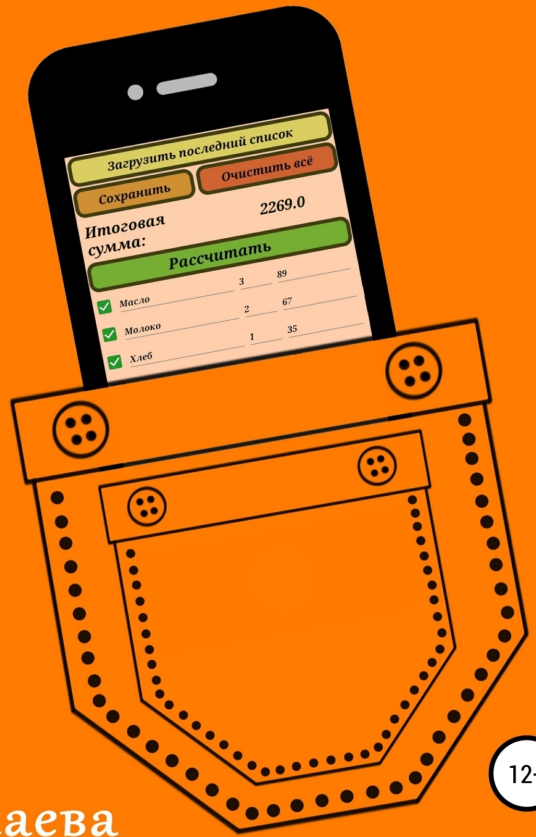
Android Studio

полезные шпаргалки для начинающих



12+

Т. Кучаева

Татьяна Анатольевна Кучаева

Android Studio. Полезные шпаргалки для начинающих

http://www.litres.ru/pages/biblio_book/?art=67900008
SelfPub; 2022

Аннотация

Программирование и разработка мобильных приложений на Android уже давно покорила весь мир. С каждым годом всё больше людей хотят научиться создавать приложения и игры на планшет или смартфон. В этой книге есть всё необходимое для того, чтобы научиться самостоятельно разрабатывать приложения, как самые элементарные, так и более функциональные. Скорее всего Вам уже пришла в голову гениальная идея и Вы хотите воплотить ее в жизнь? Тогда Вы попали по адресу. Вы научитесь правильно создавать приложения, проектировать интерфейс, применять различные способы и методы на практике.

Татьяна Кучаева
Android Studio. Полезные шпаргалки для начинающих

Введение.

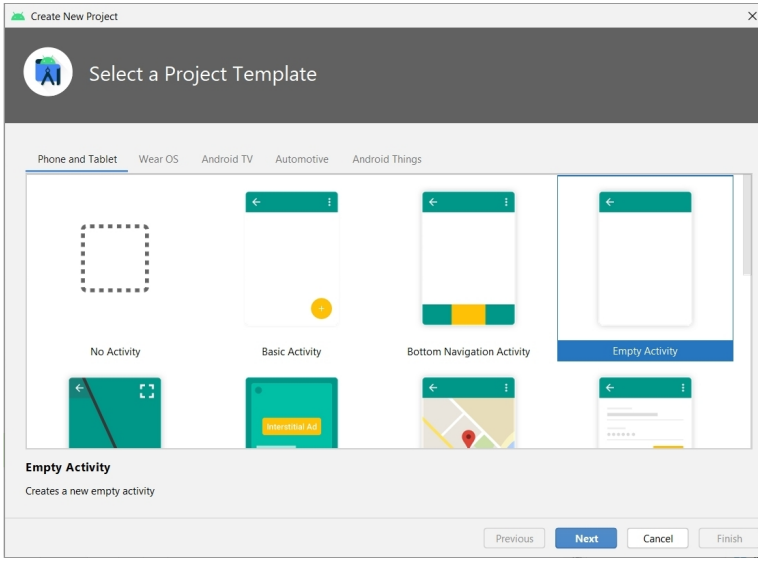
Программирование и разработка мобильных приложений на Android уже давно покорила весь мир. С каждым годом всё больше людей хотят научиться создавать приложения и игры на планшет или смартфон. В этой книге есть всё необходимое для того, чтобы научиться самостоятельно разрабатывать приложения, как самые элементарные, так и более функциональные.

Скорее всего Вам уже пришла в голову гениальная идея и Вы хотите воплотить ее в жизнь? Тогда Вы попали по адресу. Вы научитесь правильно создавать приложения, проектировать интерфейс, применять различные способы и методы на практике.

Если Вы читаете эту книгу, предполагается, что Вы уже изучили первые шаги, имеете базовые знания Java, установили на свой компьютер Android Studio и, возможно, создали свой первый проект. Но, скорее всего, Вам не хватает практики, чтобы продвинуться дальше. Именно в этой книге со-

браны самые необходимые шпаргалки для их практического применения. Все шаги будут обязательно проиллюстрированы и прописаны простым и понятным языком, а также включать в себя фрагменты кода.

Для того, чтобы начать разработку мобильного приложения, необходимо запустить Android Studio и создать новый проект. Для этого на панели управления нажимаем File → New → New Project. Для начала выбираем Empty Activity, что означает «пустая активность», для базового изучения этого будет достаточно. После приобретения достаточного количества знаний можно будет пользоваться шаблонами, которые предоставляет нам среда разработки. Нажимаем Next.

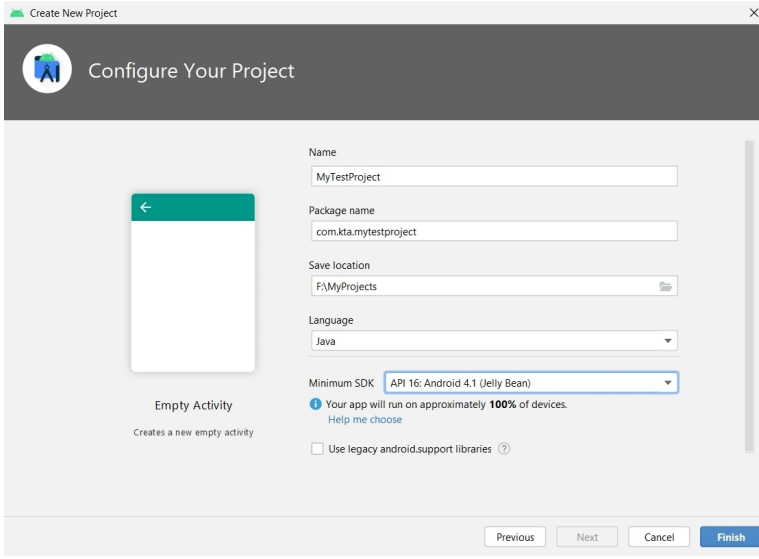


В поле Name необходимо указать имя Вашего проекта, например MyTestProject, или Вы можете задать ему любое другое имя, какое захотите.

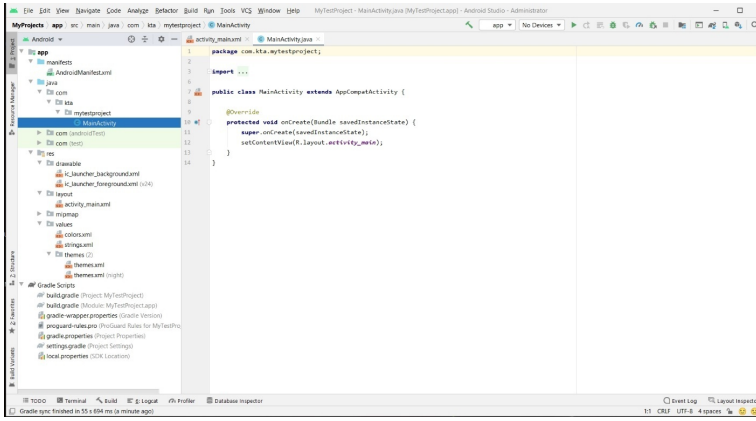
В поле Package name прописываем имя пакета, то есть уникальное название готового приложения. Как правило оно совпадает с именем приложения.

В поле Save location выбираем место на компьютере, куда сохраним проект.

Язык программирования (Language) выбираем Java, а Minimum SDK выбираем такой, чтобы наше приложение поддерживалось на 100% устройств.



Нажимаем Finish и ждём пока Android Studio загрузит наше приложение.



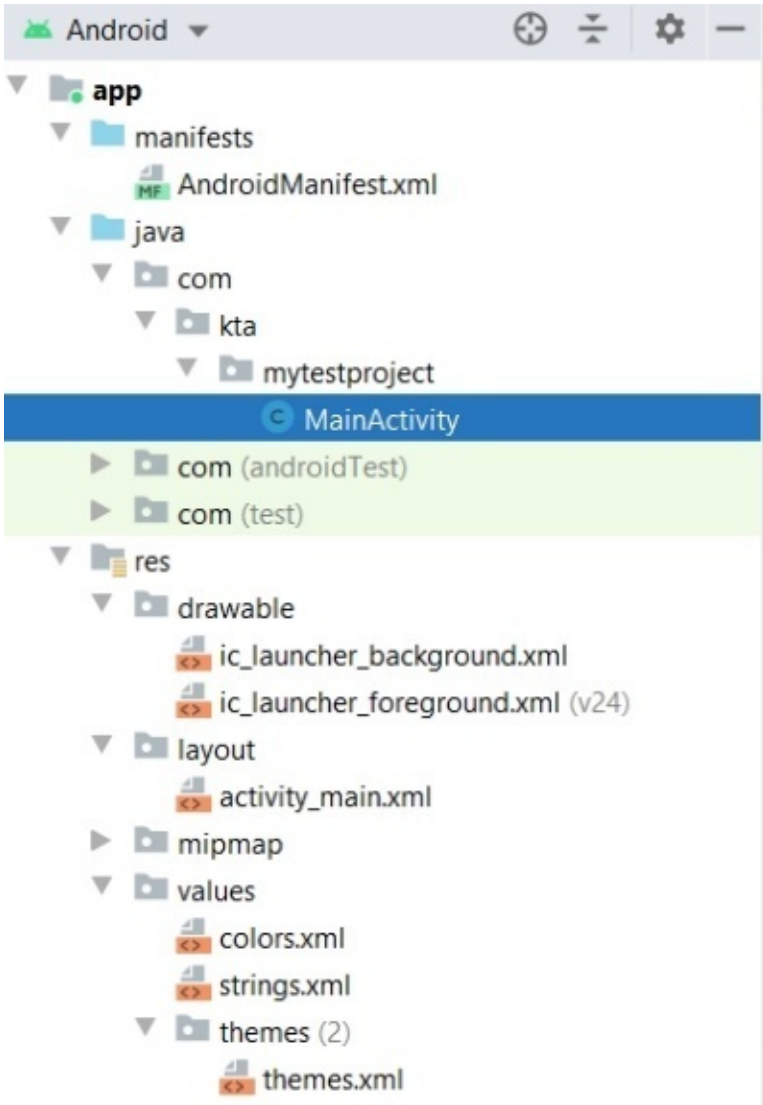
Слева мы видим содержимое нашего проекта, всё, с чем мы будем работать:

AndroidManifest – это некий набор правил, по которым работает готовое приложение. Файл манифеста всегда находится в корневой папке, называется AndroidManifest.xml и содержит в себе всю важную информацию, без которой система просто не сможет запустить приложение.

Java-файл, в котором прописывается весь код для взаимодействия приложения с пользователем. Таких файлов может быть сколько угодно много, но главный всегда будет один – MainActivity, именно с него начинается запуск приложения.

res – это папка, которая содержит в себе все необходимые ресурсы для работы приложения: layout, картинки, звуки, музыку, видео, цвета, текст, темы.

Gradle Scripts – система автоматической сборки приложения.


Android
app
manifests
AndroidManifest.xml
java
com
kta
mytestproject
MainActivity
com (androidTest)
com (test)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
activity_main.xml
mipmap
values
colors.xml
strings.xml
themes (2)
themes.xml

Справа находится поле, где прописывается код для определенных действий и видна вся разметка и структура приложения.

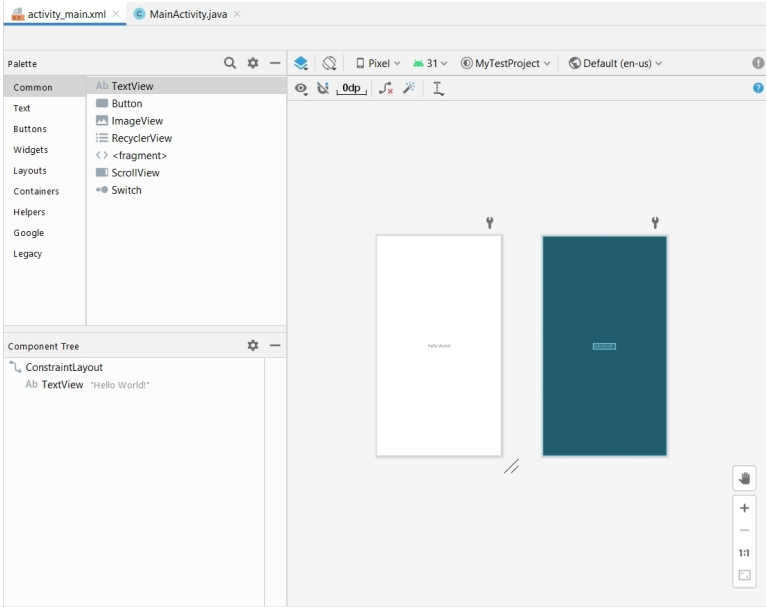
Итак, приложение создано. Но, если мы запустим его на устройстве, не важно – виртуальном или физическом, мы ничего не увидим, кроме белого чистого листа. Это потому, что мы ещё ничего не создали и не добавили в приложение.

Именно этим мы и займёмся в практической части книги.

Шпаргалка № 1. Layout.

Layout – это класс, управляющий как его компоненты бу-

дут располагаться и выглядеть на экране телефона в готовом приложении. Каждому Java-файлу соответствует свой xml-файл. Здесь всё просто – в xml находится разметка, мы видим, как располагаются текстовые поля, кнопки и прочее, в java – код, который позволяет всеми этими компонентами управлять. Например, по умолчанию файлу MainActivity.java соответствует файл activity_main.xml



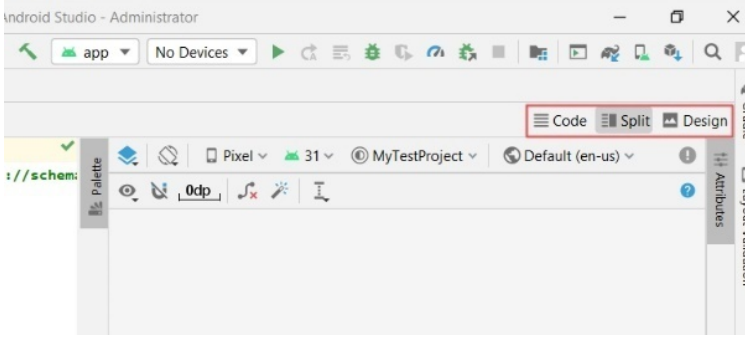
В верхнем правом углу можно заметить три кнопки раз-

личного представления:

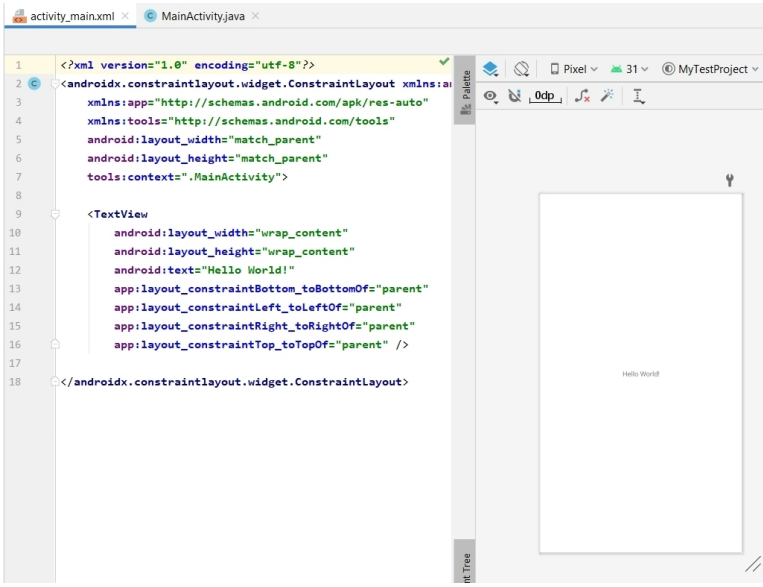
1) Code – означает, что на экране можно видеть только xml разметку, то есть код, который описывает вид и расположение полей, кнопок и прочего.

2) Split – можно видеть как код, так и визуальное представление.

3) Design – видим только визуальное расположение элементов, при этом можем управлять ими в полуавтоматическом режиме. На самом деле, это не очень удобно, поэтому советуем пользоваться вариантом Split.



Именно в таком положении мы можем спокойно менять параметры визуализации, и одновременно видеть все изменения на экране.



Шпаргалка № 2. Макет.

В каждом приложении обязательно присутствуют различные компоненты, такие как текстовые поля, кнопки, чекбоксы, картинки, звуковые дорожки и много другое. Все эти элементы должны каким-то образом располагаться на экране телефона, чтобы пользователь мог с ними взаимодействовать. И именно за это отвечает макет (или разметка). Если представить себе, что макет – это контейнер, а элементы – например кубики, то говоря простыми словами: мы раскла-

дываем кубики в контейнер так, как нам хочется, как того требует ситуация.

Разметки бывают нескольких видов, некоторые из которых мы рассмотрим отдельно.

1) FrameLayout – это, пожалуй, самый простой вид разметки. Как правило, это пустое пространство на экране, которое возможно заполнить элементами только путём прикрепления их к верхнему левому углу экрана. В такой разметке никак нельзя определить желаемое местоположение для объектов. Каждый последующий добавленный объект будет накладываться поверх предыдущих, при этом частично или полностью затеняя их.

2) LinearLayout – название говорит само за себя: все элементы в данной разметке располагаются в одну линию. Макет автоматически выравнивает все находящиеся в нём объекты в одном заданном направлении – либо вертикально, либо горизонтально. Все элементы становятся один за другим в ряд и имеют одинаковый размер, за исключением случаев, когда для отдельных элементов прописывается конкретный размер или конкретный вес. Об этом мы поговорим чуть позже и рассматривать будем уже на практике.

3) TableLayout – табличная разметка, которая позиционирует добавленные элементы в строки и столбцы, она не отображает линии для них, и вполне может иметь строки с разным количеством ячеек. При формировании такой разметки некоторые из ячеек при необходимости можно оставлять

пустыми либо заполнять другими макетами.

4) RelativeLayout – это относительная разметка, в которой все элементы расположены так, что, если первый элемент расположен по центру экрана, другие элементы, будут выравниваться относительно первого элемента.

5) ConstraintLayout – представляет собой контейнер, который позволяет создавать гибкие интерфейсы. Для определения позиции элемента внутри данного макета необходимо указать ограничения либо по горизонтали, либо по вертикали. Позиционирование может производиться как относительно границ самого макета, так и относительно любого другого элемента внутри разметки.

6) ScrollView – скроллинг. Используется при больших объёмах текста, картинок и прочего, которые не помещаются на экран и требуется прокрутка. ScrollView внутри себя может иметь только один дочерний элемент, например LinearLayout, а в этом макете уже размещаются остальные элементы.

Шпаргалка № 3. Параметры.

Абсолютно все макеты и элементы, которые содержит в себе приложение, должны иметь некие параметры: название, размер (высота и ширина), вес, ориентация, задний фон, цвет и многое другое. В этой главе мы рассмотрим основные, самые важные параметры для разметок и их элементов.

android:id="@+id/abc" – ID элемента (имя), пожалуй

самый важный параметр, так как при любом действии Android Studio будет обращаться к элементу именно по ID.

android:layout_width="match_parent" – ширина элемента, бывает двух типов **"match_parent"** – будет растянут на весь контейнер, или **"wrap_content"** – займёт места столько, сколько внутри будет контента (например, текста). Также, при необходимости, можно указывать размер вручную, для этого необходимо указать следующий параметр в пикселях, например **"200dp"**. Цифра может быть любая, хоть 121, хоть 367.

android:layout_height="wrap_content" – высота элемента. Все параметры такие же, как и у ширины.

android:layout_weight="1" – вес. Данный параметр чаще всего используется в разметке LinearLayout, когда элементы должны занимать на экране определенное количество места, но делать это должны автоматически, а не вручную. Параметр может быть любой, начиная от числа 0,1.

android:orientation="vertical" – ориентация макета. Бывает **"vertical"** – вертикальная, и **"horizontal"** – горизонтальная.

android:background="@color/cherry" – задний фон. На

любой макет или элемент можно установить задний фон. Это может быть просто сплошной цвет, который взят из базы цветов в разделе color (**"@color/cherry"**), или прописан кодом (**"#D12B68"**), а также это может быть любая картинка (**"@drawable/print"**, **"@mipmap/print"**).

android:gravity="center" – гравитация. Определяет местоположение текста внутри элемента, в данном примере – по центру.

android:layout_gravity="center_horizontal" – определяет местоположение элемента внутри контейнера. Так же, как и **android:gravity** может быть разных видов: **"top"** – сверху, **"bottom"** – снизу, **"left"** – слева, **"right"** – справа.

android:text="@string/text" – данный параметр используется чаще всего для текстовых элементов или кнопок, которые содержат в себе надписи, подписи, тексты.

android:paddingTop="10dp" – параметр, который используется для отступа содержимого элемента внутри этого самого элемента сверху, значение может быть любое.

android:paddingBottom="10dp" – отступ снизу.

android:paddingLeft="10dp" – отступ слева.

android:paddingRight="10dp" – отступ справа.

android:padding="20dp" – используется для равномерного отступа со всех сторон.

android:layout_marginTop="15dp" – в отличие от предыдущего, данный параметр используется для отступа элемента внутри контейнера, либо для отступа от соседнего элемента, так же может быть сверху (**layout_marginTop**), снизу (**layout_marginBottom**), слева (**layout_marginLeft**) и справа (**layout_marginRight**). Значение в dp может быть любое.

android:layout_margin="25dp" – используется для равномерного отступа со всех сторон.

style="@style/MyTextStyle" – если приложение содержит в себе множество элементов с одинаковыми параметрами, гораздо удобнее создать стиль, указав его для элемента. Так и код смотрится более читабельно, и, если будет необходимость что-то в этих параметрах изменить – изменять придётся только в одном месте, а не в 10, 20, 50 и более элементах.

Конечно, это лишь малая часть из всех существующих

параметров, но это самые наиболее часто используемые и нужные. Более подробно мы ещё будем их рассматривать на практике в последующих главах.

Шпаргалка № 4. Текстовое поле.

Ни одно приложение не обходится без текста, каких-либо надписей, подписей, информации о приложении, в некоторых случаях даже ввода текста самим пользователем. За всё это отвечает текстовое поле. Об этом и поговорим.

Текстовые поля в редакторе бывают двух видов TextView и EditText.

TextView – элемент, содержащий в себе определённое количество любого текста и предназначен он только для его просмотра, без возможности редактирования пользователем.

EditText – это также текстовое поле, но уже с возможностью ввода или редактирования текста.

Добавляем текстовое поле в приложение:

```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

По умолчанию только что созданное текстовое поле всегда имеет только два параметра, ширину и высоту.

В первую очередь нужно задать имя, для этого добавляем строку:

android:id="@+id/tv1"

Назовем его, например, tv1. Но в Вашем случае название может быть каким угодно, главное, чтобы оно было, и Вы могли к нему обратиться в случае необходимости.

Определяемся с текстом, допустим, это будет что-то приветственное, «Добро пожаловать!»

В ресурсах ищем файл strings.xml, там хранятся все строковые ресурсы, используемые в приложении, создаём строку:

<string name="str1">Добро пожаловать!**</string>**

Добавляем текст в наше поле:

android:text="@string/str1"

Чтобы текст на экране мобильного телефона выглядел красиво и приятно читался, необходимо прописать параметры, которые его улучшат, так как Android Studio по умолчанию предлагает стандартный вид.

Задаём размер текста:

android:textSize="35sp" – Цифра может быть любой, начиная с единицы.

А чтобы текст не был БОЛЬШИМИ БУКВАМИ, добавим следующий параметр:

android:textAllCaps="false"

Можно поменять цвет текста. Допустим, мы хотим сделать его голубым. Для этого переходим в ресурсы, открываем вкладку colors.xml и добавляем строку:

<color name="blue">#1E88E5**</color>**

Возвращаемся к нашему текстовому полю и добавляем строку:

android:textColor="@color/blue"

Теперь поместим наше текстовое поле на середину экрана, для этого добавим строку в параметры макета:

android:gravity="center"

И строку в параметры элемента:

android:gravity="center"
android:layout_gravity="center"

Добро пожаловать!

Таким образом мы получили приветственный экран нашего приложения. Пользователь увидит его, как только приложение загрузится. Вот его полный код:

```
<TextView
android:id="@+id/tv1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/str1"
android:textColor="@color/blue"
android:textSize="35sp"
android:textAllCaps="false"
android:gravity="center"
android:layout_gravity="center"/>
```

Но, давайте рассмотрим ещё вариант текстового поля EditText, в которое можно что-то вписывать, например имя.

Добавим такое поле чуть ниже.

```
<EditText
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
```

Ширину установим "match_parent" – то есть на всю ширину макета, а высоту оставим по количеству контента. Теперь в это поле пользователь может ввести своё имя.

Добро пожаловать!

Но, как же он узнает, что туда нужно вводить, если мы ему об этом не подскажем? Да и вообще, поле выглядит как обычная полоса, совсем не похоже, что это текстовое поле. И оно как бы «прилипает» к первому приветственному полю, что совсем не красиво смотрится. Давайте это исправим. Для начала сделаем отступы:

android:layout_margin="40dp"

А затем сделаем для пользователя подсказку. В строковый ресурс добавим:

<string name="str2">Введите своё имя**</string>**

А в текстовое поле :

android:hint="@string/str2"

И выравнивание текста посередине:

android:gravity="center"

Итого, мы получили:

<EditText
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="40dp"
android:hint="@string/str2"

android:gravity="center"/>

Добро пожаловать!

Введите своё имя

Теперь пользователь может ввести своё имя. Подобным образом можно делать поля для ввода текста, цифр, номера телефона, логина, пароля и прочее.

Конец ознакомительного фрагмента.