创建第一个Kotlin应用程序

创建一个新的工程

打开Android Studio，选择Projects>New Project，然后选择Basic Activity.

点击Next，为应用程序命名（例如：My First App），选择Kotlin语言，然后点击Finish。Android Studio将使用系统中最新的API Level创建应用程序，并使用Gradle作为构建系统，在底部的视窗中可以查看整个过程。

探索Android Studio的界面布局

整个Android Studio工作区包括多个部分，

创建模拟器

次步骤创建可以运行APP的模拟器，点击Tool>Device Manager或者工具栏上的按钮

点击Create device，弹出创建模拟器的页面，

选择想要创建模拟器设备（如Pixel 5），点击Next，在系统镜像页面的Recommended标签栏，选择最新镜像，

然后首先下载镜像（Download），下载完成之后点击Next，完成模拟器命名和更多参数选择，最终点击Finish完成。注意：真实型号机型的模拟器镜像往往十分巨大，如果硬盘空间不足，考虑下载通用模拟器镜像。

在模拟器上运行应用程序

选择Run>Run ‘app’，在工具栏上可以看到运行程序的一些选择项。

运行效果：

查看布局编辑器

在Basic Activity中，包含了基本的导航组件，Android app关联两个fragments，第一个屏幕显示了“Hello first fragment”由FirstFragment创建，界面元素的排列由布局文件指定，查看res>layout>fragment\_first.xml，

查看布局的代码（Code），修改Textview的Text属性，

android:text="@string/hello\_first\_fragment"

1

右键该代码，选择Go To > Declaration or Usages，跳转到values/strings.xml，看到高亮文本

<string name="hello\_first\_fragment">Hello first fragment</string>

1

修改字符串属性值为“Hello Kotlin!”。更进一步，修改字体显示属性，在Design视图中选择textview\_first文本组件，在Common Attributes属性下的textAppearance域，设置相关的文字显示属性，

查看布局的XML代码，可以看到新属性被应用。

android:fontFamily="sans-serif-condensed"

android:text="@string/hello\_first\_fragment"

android:textColor="@android:color/darker\_gray"

android:textSize="30sp"

android:textStyle="bold"

1

2

3

4

5

重新运行应用程序，查看显示效果。

向页面添加更多的布局

本步骤将向第一个Fragment添加更多的视图组件

查看视图的布局约束

在fragment\_first.xml，查看TextView组件的约束属性：

在约束布局中，每个组件至少需要一个水平方向和一个垂直方向的约束，更多约束布局的内容，请查看ConstraintLayout。

添加按钮和约束

从Palette面板中拖动Button到

调整Button的约束，设置Button的Top>BottonOf textView，

app:layout\_constraintTop\_toBottomOf="@+id/textview\_first" />

1

随后添加Button的左侧约束至屏幕的左侧，Button的底部约束至屏幕的底部。查看Attributes面板，修改将id从button修改为toast\_button（注意修改id将重构代码）

调整Next按钮

Next按钮是工程创建时默认的按钮，查看Next按钮的布局设计视图，它与TextView之间的连接不是锯齿状的而是波浪状的，表明两者之间存在链（chain），是一种两个组件之间的双向联系而不是单向联系。删除两者之间的链，可以在设计视图右键相应约束，选择Delete（注意两个组件要双向删除）；

或者在属性面板的Constraint Widget中移动光标到相应约束点击删除。

同时，删除Next按钮的左侧约束。

添加新的约束

添加Next的右边和底部约束至父类屏幕（如果不存在的话），Next的Top约束至TextView的底部。最后，TextView的底部约束至屏幕的底部。效果看起来如下图所示：

更改组件的文本

fragment\_first.xml布局文件代码中，找到toast\_button按钮的text属性部分

<Button

android:id="@+id/toast\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button"

1

2

3

4

5

这里text的赋值是一种硬编码，点击文本，左侧出现灯泡状的提示，选择 Extract string resource。

弹出对话框，令资源名为toast\_button\_text，资源值为Toast，并点击OK。

于是，在资源文件string.xml定义了字符串，以上操作可以手动在string.xml文件中定义并引用。

<resources>

...

<string name="toast\_button\_text">Toast</string>

</resources>

1

2

3

4

更新Next按钮

在属性面板中更改Next按钮的id，从button\_first改为random\_button。

在string.xml文件，右键next字符串资源，选择 Refactor > Rename，修改资源名称为random\_button\_text，点击Refactor 。随后，修改Next值为Random。

添加第三个按钮

向fragment\_first.xml文件中添加第三个按钮，位于Toast和Random按钮之间，TextView的下方。新Button的左右约束分别约束至Toast和Random，Top约束至TextView的底部，Buttom约束至屏幕的底部，看起来的效果：

检查xml代码，确保不出现类似app:layout\_constraintVertical\_bias这样的属性，即不手动设置偏移量。

完善UI组件的属性设置

更改新增按钮id为count\_button，显示字符串为Count，对应字符串资源值为count\_button\_text。于是三个按钮的text和id属性如下表：

同时，更改TextView的文本为0。修改后的fragment\_first.xml的代码

<?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=".FirstFragment">

<TextView

android:id="@+id/textview\_first"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:fontFamily="sans-serif-condensed"

android:text="@string/hello\_first\_fragment"

android:textColor="@android:color/darker\_gray"

android:textSize="30sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<Button

android:id="@+id/random\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/random\_button\_text"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/textview\_first" />

<Button

android:id="@+id/toast\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/toast\_button\_text"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/textview\_first" />

<Button

android:id="@+id/count\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/count\_button\_text"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toStartOf="@+id/random\_button"

app:layout\_constraintStart\_toEndOf="@+id/toast\_button"

app:layout\_constraintTop\_toBottomOf="@+id/textview\_first" />

</androidx.constraintlayout.widget.ConstraintLayout>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

尝试运行应用程序查看效果。

更新按钮和文本框的外观

添加新的颜色资源

values>colors.xml定义了一些应用程序可以使用的颜色，添加新颜色screenBackground 值为 #2196F3，这是蓝色阴影色；添加新颜色buttonBackground 值为 #BBDEFB

<color name="screenBackground">#2196F3</color>

<color name="buttonBackground">#BBDEFB</color>

1

2

设置组件的外观

fragment\_first.xml的属性面板中设置屏幕背景色为

android:background="@color/screenBackground"

1

设置每个按钮的背景色为buttonBackground

android:background="@color/buttonBackground"

1

注意：在实验的API level中（31），这种设置并不生效，需修改res/values/themes.xml的style值，添加\*\*.Bridge\*\*。

<style name="Theme.MyFirstApp" parent="Theme.MaterialComponents.DayNight.DarkActionBar.Bridge">

1

移除TextView的背景颜色，设置TextView的文本颜色为color/white，并增大字体大小至72sp

设置组件的位置

Toast与屏幕的左边距设置为24dp，Random与屏幕的右边距设置为24dp，利用属性面板的Constraint Widget完成设置。

设置TextView的垂直偏移为0.3，

app:layout\_constraintVertical\_bias="0.3"

1

拖动左侧的移动条。

运行应用程序

最终效果如下图：

添加代码完成应用程序交互

设置代码自动补全

Android Studio中，依次点击File>New Projects Settings>Settings for New Projects…，查找Auto Import选项，在Java和Kotlin部分，勾选Add Unambiguous Imports on the fly。

TOAST按钮添加一个toast消息

打开FirstFragment.kt文件，有三个方法：onCreateView，onViewCreated和onDestroyView，在onViewCreated方法中使用绑定机制设置按钮的响应事件（创建应用程序时自带的按钮）。

binding.randomButton.setOnClickListener {

findNavController().navigate(R.id.action\_FirstFragment\_to\_SecondFragment)

}

1

2

3

接下来，为TOAST按钮添加事件，使用\*\*findViewById()\*\*查找按钮id，代码如下：

// find the toast\_button by its ID and set a click listener

view.findViewById<Button>(R.id.toast\_button).setOnClickListener {

// create a Toast with some text, to appear for a short time

val myToast = Toast.makeText(context, "Hello Toast!", Toast.LENGTH\_LONG)

// show the Toast

myToast.show()

}

1

2

3

4

5

6

7

代码使用了Lambda表达式的机制。

使Count按钮更新屏幕的数字

此步骤向Count按钮添加事件响应，更新Textview的文本显示。

在FirstFragment.kt文件，为count\_buttion按钮添加事件：

view.findViewById<Button>(R.id.count\_button).setOnClickListener {

countMe(view)

}

1

2

3

countMe()为自定义方法，以View为参数，每次点击增加数字1，具体代码为：

private fun countMe(view: View) {

// Get the text view

val showCountTextView = view.findViewById<TextView>(R.id.textview\_first)

// Get the value of the text view.

val countString = showCountTextView.text.toString()

// Convert value to a number and increment it

var count = countString.toInt()

count++

// Display the new value in the text view.

showCountTextView.text = count.toString()

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

完成第二界面的代码

此步骤将完成按照First Fragment显示数字作为上限，随机在Second Fragment上显示一个数字，即Random按钮的事件响应。效果如下：

向界面添加TextView显示随机数

打开fragment\_second.xml的设计视图中，当前界面有两个组件，一个Button和一个TextView（textview\_second）。

去掉TextView和Button之间的约束

拖动新的TextView至屏幕的中间位置，用来显示随机数

设置新的TextView的id为\*\*@+id/textview\_random\*\*

设置新的TextView的左右约束至屏幕的左右侧，Top约束至textview\_second的Bottom，Bottom约束至Button的Top

设置TextView的字体颜色textColor属性为\*\*@android:color/white\*\*，textSize为72sp，textStyle为bold

设置TextView的显示文字为“R”

设置垂直偏移量layout\_constraintVertical\_bias为0.45

新增TextView最终的属性代码

<TextView

android:id="@+id/textview\_random"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="R"

android:textColor="@android:color/white"

android:textSize="72sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toTopOf="@+id/button\_second"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/textview\_second"

app:layout\_constraintVertical\_bias="0.45" />

1

2

3

4

5

6

7

8

9

10

11

12

13

更新显示界面文本的TextView(textview\_second)

在fragment\_second.xml文件中，选择textview\_second文本框，查看text属性，可见

android:text="@string/hello\_second\_fragment

1

对应的strings.xml文本为Hello second fragment. Arg: %1$s

更改该文本框id为textview\_header

设置layout\_width为match\_parent，layout\_height为wrap\_content。

设置top，left和right的margin为24dp，左边距和右边距也就是start和end边距。

若还存在与Button的约束，则删除。

向colors.xml添加颜色colorPrimaryDark，并将TextView颜色设置为@color/colorPrimaryDark，字体大小为24sp。

<color name="colorPrimaryDark">#3700B3</color>

1

strings.xml文件中，修改hello\_second\_fragment的值为"Here is a random number between 0 and %d."

使用Refactor>Rename将hello\_second\_fragment 重构为random\_heading

因此，显示界面信息的Textview的代码为：

<TextView

android:id="@+id/textview\_header"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginStart="24dp"

android:layout\_marginLeft="24dp"

android:layout\_marginTop="24dp"

android:layout\_marginEnd="24dp"

android:layout\_marginRight="24dp"

android:text="@string/random\_heading"

android:textColor="@color/colorPrimaryDark"

android:textSize="24sp"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

更改界面的背景色和按钮布局

向colors.xml文件添加第二个Fragment背景色的值，修改fragment\_second.xml背景色的属性为screenBackground2

<color name="screenBackground2">#26C6DA</color>

1

将按钮移动至界面的底部，完成所有布局之后，如下图所示：

检查导航图

本项目选择Android的Basic Activity类型进行创建，默认情况下自带两个Fragments，并使用Android的导航机制Navigation。导航将使用按钮在两个Fragment之间进行跳转，就第一个Fragment修改后的Random按钮和第二个Fragment的Previous按钮。

打开nav\_graph.xml文件（res>navigation>nav\_graph.xml），形如：

可以任意拖动界面中的元素，观察导航图的变化。

启用SafeArgs

SafeArgs 是一个 gradle 插件，它可以帮助您在导航图中输入需要传递的数据信息，作用类似于Activity之间传递数据的Bundle。

设置SafeArgs的gradle的脚本步骤可以参考：Navigation API参考文档

首先打开 Gradle Scripts > build.gradle (Project: My First App)

找到buildscript脚本中的dependencies章节，添加如下代码

def nav\_version = "2.3.0-alpha02"

classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav\_version"

1

2

接着打开 Gradle Scripts > build.gradle (Module: app)

apply plugin开头的代码下添加一行

apply plugin: 'androidx.navigation.safeargs.kotlin'

1

Android Studio开始同步依赖库

重新生成工程Build > Make Project

注意：不同版本的Android Studio（连同Gradle）所依赖的导航库不相同，如果遇到问题，查看使用 Safe Args 传递安全的数据以寻找解决方案。

在Android Studio Bumblebee | 2021.1.1 Patch 3中，gradle的脚本发生巨大变化。在新版本中，Safeargs的依赖将这样添加：

Gradle的Project部分，在plugins节添加

id 'androidx.navigation.safeargs.kotlin' version '2.5.0-alpha01' apply false

module部分在plugins节添加

id 'androidx.navigation.safeargs'

不同版本的Gradle在navigation插件版本不相同。

创建导航动作的参数

打开导航视图，点击FirstFragment，查看其属性。

在Actions栏中可以看到导航至SecondFragment

同理，查看SecondFragment的属性栏

点击Arguments \*\*+\*\*符号

弹出的对话框中，添加参数myArg，类型为整型Integer

FirstFragment添加代码，向SecondFragment发数据

初始应用中，点击FirstFragment的Next/Random按钮将跳转到第二个页面，但没有传递数据。在本步骤中将获取当前TextView中显示的数字并传输至SecondFragment。

打开FirstFragment.kt源代码文件

找到onViewCreated()方法，该方法在onCreateView方法之后被调用，可以实现组件的初始化。找到Random按钮的响应代码，注释掉原先的事件处理代码

实例化TextView，获取TextView中文本并转换为整数值

val showCountTextView = view.findViewById<TextView>(R.id.textview\_first)

val currentCount = showCountTextView.text.toString().toInt()

1

2

将currentCount作为参数传递给actionFirstFragmentToSecondFragment()

val action = FirstFragmentDirections.actionFirstFragmentToSecondFragment(currentCount)

1

添加导航事件代码

findNavController().navigate(action)

1

运行代码，点击FirstFragment的Count按钮，然后点击Random按钮，可以看到SecondFragment在头部的TextView已经显示正确的数字，但是屏幕中间还未出现随机数显示。

添加SecondFragment的代码

本节将更新SecondFragment.kt的代码，接受传递过来的整型参数并进行处理

导入navArgs包

import androidx.navigation.fragment.navArgs

1

onViewCreated()代码之前添加一行

val args: SecondFragmentArgs by navArgs()

1

onViewCreated()中获取传递过来的参数列表，提取count数值，并在textview\_header中显示

val count = args.myArg

val countText = getString(R.string.random\_heading, count)

view.findViewById<TextView>(R.id.textview\_header).text = countText

1

2

3

根据count值生成随机数

val random = java.util.Random()

var randomNumber = 0

if (count > 0) {

randomNumber = random.nextInt(count + 1)

}

1

2

3

4

5

textview\_random中显示count值

view.findViewById<TextView>(R.id.textview\_random).text = randomNumber.toString()

1

运行应用程序，查看运行结果。

————————————————

版权声明：本文为CSDN博主「llfjfz」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/llfjfz/article/details/124031522