

# 1. अपना पहला ऐप बनाना

## अवलोकन

यह अध्याय Android का परिचय है, जहाँ आप अपना परिवेश सेट करेंगे और Android विकास के मूल सिद्धांतों पर ध्यान केंद्रित करेंगे। इस अध्याय के अंत तक, आपको स्कैच से Android ऐप बनाने और इसे वर्चुअल या भौतिक Android डिवाइस पर इंस्टॉल करने के लिए आवश्यक ज्ञान प्राप्त हो जाएगा। आप **AndroidManifest.xml** फ़ाइल के महत्व का विश्लेषण और समझने में सक्षम होंगे, और अपने ऐप को कॉन्फ़िगर करने और मटीरियल डिज़ाइन से UI तत्वों को लागू करने के लिए Gradle बिल्ड टूल का उपयोग कर सकेंगे।

## परिचय

Android दुनिया में सबसे ज़्यादा इस्तेमाल किया जाने वाला मोबाइल फ़ोन ऑपरेटिंग सिस्टम है, जिसकी वैश्विक बाज़ार में 70% से ज़्यादा हिस्सेदारी है (देखें

<https://gs.statcounter.com/os-market-share/mobile/worldwide>

)। यह Android सीखने और वैश्विक पहुँच वाले ऐप बनाने के ज़रिए योगदान देने और प्रभाव डालने के बेहतरीन अवसर प्रस्तुत करता है। Android के लिए नए डेवलपर के लिए,



**जवाब** | नया-AI द्वारा संचालित

□ प्रतिक्रिया

हम आपके लिए किस प्रश्न का उत्तर दे



मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

सीखने और उत्पादक बनने के लिए कई मुद्दों से निपटना होगा। यह पुस्तक इन मुद्दों को संबोधित करेगी। टूलिंग और डेवलपमेंट एनवायरनमेंट सीखने के बाद, आप Android ऐप बनाने के लिए बुनियादी अभ्यासों का पता लगाएंगे। हम डेवलपर्स द्वारा सामना की जाने वाली वास्तविक दुनिया की विकास चुनौतियों की एक विस्तृत श्रृंखला को कवर करेंगे और उन्हें दूर करने के लिए विभिन्न तकनीकों का पता लगाएंगे।

इस अध्याय में, आप सीखेंगे कि एक बुनियादी Android प्रोजेक्ट कैसे बनाया जाता है और उसमें सुविधाएँ कैसे जोड़ी जाती हैं। आपको Android Studio के व्यापक विकास परिवेश से परिचित कराया जाएगा और आपको उत्पादक रूप से काम करने में सक्षम बनाने के लिए सॉफ्टवेयर के मुख्य क्षेत्रों के बारे में बताया जाएगा। Android Studio एप्लिकेशन विकास के लिए सभी उपकरण प्रदान करता है, लेकिन ज्ञान नहीं। यह पहला अध्याय आपको ऐप बनाने और Android प्रोजेक्ट के सबसे सामान्य क्षेत्रों को कॉन्फ़िगर करने के लिए सॉफ्टवेयर का प्रभावी ढंग से उपयोग करने के बारे में मार्गदर्शन करेगा।

आइये, एंड्रॉइड प्रोजेक्ट बनाना शुरू करें।

## Android Studio के साथ Android प्रोजेक्ट बनाना

Android ऐप्स बनाने के मामले में उत्पादक होने के लिए, **Android Studio** का उपयोग करने के तरीके के बारे में आश्वस्त होना आव-



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

श्यक है। यह Android विकास के लिए आधिकारिक **एकीकृत विकास वातावरण (IDE)** है, जिसे JetBrains के **IntelliJ IDEA IDE** पर बनाया गया है और Google में Android Studio टीम द्वारा विकसित किया गया है। आप इस कोर्स में ऐप्स बनाने और क्रमिक रूप से अधिक उन्नत सुविधाएँ जोड़ने के लिए इसका उपयोग करेंगे।

The development of Android Studio has followed the development of the IntelliJ IDEA IDE. The fundamental features of an IDE are of course present, enabling you to optimize your code with suggestions, shortcuts, and standard refactoring. The programming language you will be using throughout this course to create Android apps is Kotlin. Since Google I/O 2017 (the annual Google developer conference), this has been Google's preferred language for Android app development. What really sets Android Studio apart from other Android development environments is that **Kotlin** was created by JetBrains, the company that created IntelliJ IDEA, the software Android Studio is built on. You can, therefore, benefit from established and evolving first-class support for Kotlin.

Kotlin was created to address some of the shortcomings of Java in terms



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

of verbosity, handling null types, and adding more functional programming techniques, amongst many other issues. As Kotlin has been the preferred language for Android development since 2017, taking over from Java, you will be using it in this book.

Getting to grips and familiarizing yourself with Android Studio will enable you to feel confident working on and building Android apps. So, let's get started creating your first project.

#### Note

*The installation and setup of Android Studio are covered in the Preface. Please ensure you have completed those steps before you continue.*

## Exercise 1.01: Creating an Android Studio Project for Your App

This is the starting point for creating a project structure your app will be built upon. The template-driven approach will enable you to create a basic project in a short timeframe whilst setting up the building blocks you can use to develop your app. To complete



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

this exercise, perform the following steps:

### Note

*The version of Android Studio you will be using is v4.1.1 (or above).*

1. Upon opening Android Studio, you will see a window asking whether you want to create a new project or open an existing one. Select **Create New Project**. The start up window will appear as follows:

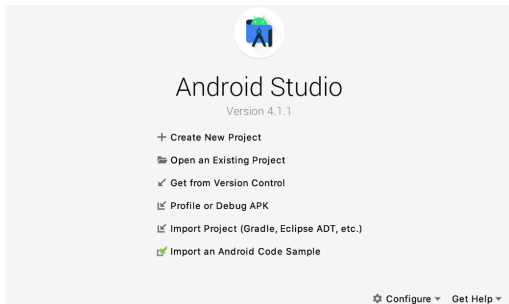


Figure 1.1: Android Studio version 4.1.1

2. Now, you'll enter a simple wizard-driven flow, which greatly simplifies the creation of your first Android project. The next screen you will see has a large number of options for the initial setup you'd like your app to have:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

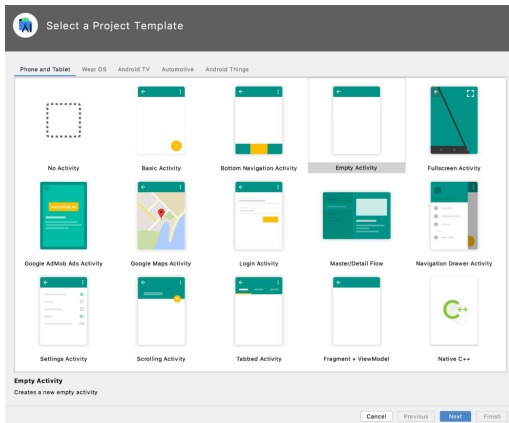
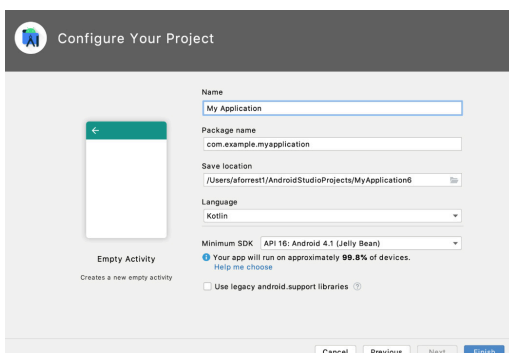


Figure 1.2: Starting a project template for your app

3. Welcome to your first introduction to the **Android development ecosystem**. The word displayed in most of the project types is **Activity**. In Android, an **Activity** is a page or screen. The options you can choose from on the preceding screen all create this initial screen differently. The descriptions describe how the first screen of the app will look. These are templates to build your app with. Select **Empty Activity** from the template and click on next. The project configuration screen is as follows:



**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

Figure 1.3: Project configuration

4. The preceding screen configures your app. Let's go through all the options:

**a. Name:** Similar to the name of your Android project, this name will appear as the default name of your app when it's installed on a phone and visible on Google Play. You can replace the **Name** field with your own or set it now to the app you are going to create.

**b. Package name:** This uses the standard reverse domain name pattern for creating a name. It will be used as an address identifier for source code and assets in your app. It is best to make this name as clear and descriptive and as closely aligned with the purpose of your app as possible. Therefore, it's probably best to change this to use one or more sub-domains (such as **com.sample.shop.myshop**). As shown in *Figure 1.3*, the **Name** of the app (in lowercase with spaces removed) is appended to the domain.

**c. Save location:** This is the local folder on your machine where the app will be initially



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

stored. This can be changed in the future, so you can probably keep the default or edit it to something different (such as **Users/MyUser/android/projects**).

The default location will vary with the operating system you are using.

**d. Language – Kotlin:** This is Google's preferred language for Android app development.

**e. Minimum SDK:** Depending on which version of Android Studio you download, the default might be the same as displayed in *Figure 1.3* or a different version. Keep this the same. Most of Android's new features are made backward compatible, so your app will run fine on the vast majority of older devices. However, if you do want to target newer devices, you should consider raising the minimum API level. There is a link, **Help Me Choose**, to a dialog that explains the feature set that you have access to with a view to development on different versions of Android and the current percentage of devices worldwide running each Android version.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



**f. (Checkbox) use legacy****android.support libraries.**

Leave this unchecked. You will be using AndroidX libraries, which are the replacement for the support libraries that were designed to make features on newer versions of Android backward compatible with older versions, but it provides much more than this. It also contains newer Android components called Jetpack, which, as the name suggests, "boost" your Android development and provide a host of rich features you will want to use in your app, thereby simplifying common operations. Once you have filled in all these details, select **Finish**. Your project will be built and you will then be presented with the following screen or similar: You can immediately see the activity that has been created (**MainActivity**) in one tab and the layout used for the screen in the other tab (**activity\_main.xml**). The application structure folders are in the left panel.

**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

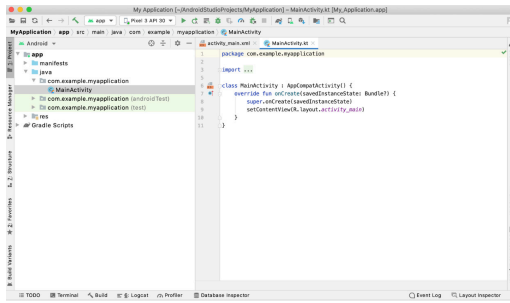


Figure 1.4: Android Studio default project

In this exercise, you have gone through the steps to create your first Android app using Android Studio. This has been a template-driven approach that has shown you the core options you need to configure for your app.

In the next section, you will set up a virtual device and see your app run for the first time.

## Setting Up a Virtual Device and Running Your App

As a part of installing Android Studio, you downloaded and installed the latest Android SDK components. These included a base emulator, which you will configure to create a virtual device to run Android apps on. The benefit is that you can make changes and quickly



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

see them on your desktop whilst developing your app. Although virtual devices do not have all the features of a real device, the feedback cycle is often quicker than going through the steps of connecting a real device.

Also, although you should ensure your app runs as expected on different devices, you can standardize it by targeting a specific device by downloading an emulator skin even if you don't have the real device if this is a requirement of your project.

The screen you will have seen (or something similar) when installing Android Studio is as follows:

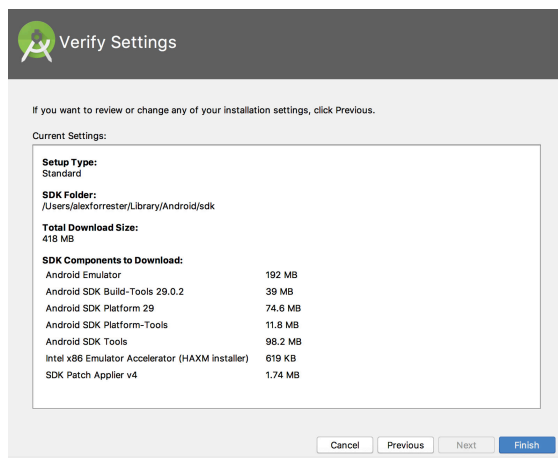


Figure 1.5: SDK components

Let's take a look at the SDK components that are installed and how the virtual device fits in:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

- **Android Emulator:** This is the base emulator, which we will configure to create virtual devices of different Android makes and models.
- **Android SDK Build-Tools:** Android Studio uses the build tools to build your app. This process involves compiling, linking, and packaging your app to prepare it for installation on a device.
- **Android SDK Platform:** This is the version of the Android platform that you will use to develop your app. The platform refers to the API level. The Android version for API level 30 is 11 and the name is Android 11. Before the release of Android 10, the version of Android was also known by a code name, which was different from the version name. The code names used to follow a sweets/dessert theme; therefore, the name **Jelly Bean** was selected above in the Create Project wizard for configuring the minimum API level of your project. From Android 10, the versioning will no longer have a code name that is different from the version name. (The versions of the Build-Tools and Platform



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

will change as new versions are released)

- **Android SDK Platform-Tools:**

These are tools you can use, ordinarily, from the command line, to interact with and debug your app.

- **Android SDK Tools:** In contrast to the platform tools, these are tools that you use predominantly from within Android Studio in order to accomplish certain tasks, such as the virtual device for running apps and the SDK manager to download and install platforms and other components of the SDK.

- **Intel x86 Emulator Accelerator (HAXM installer):** If your OS provides it, this is a feature at the hardware level of your computer you will be prompted to enable, which allows your emulator to run more quickly.

- **SDK Patch Applier v4:** As newer versions of Android Studio become available, this enables patches to be applied to update the version you are running.

With this knowledge, let's start with the next exercise of this chapter.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूं?

## Exercise 1.02: Setting Up a Virtual Device and Running Your App on It

We set up an Android Studio project to create our app in *Exercise 1.01, Creating an Android Studio Project for Your App*, and we are now going to run it on a virtual device. You can also run your app on a real device, but in this exercise, you will use a virtual device. This process is a continuous cycle whilst working on your app. Once you have implemented a feature, you can verify its look and behavior as you require. For this exercise, you will create a single virtual device, but you should ensure you run your app on multiple devices to verify that its look and behavior are consistent. Perform the following steps:

1. In the top toolbar in Android Studio, you will see two drop-down boxes next to each other pre-selected with **app** and **No devices**:



Figure 1.6: Android Studio toolbar

The **app** is the configuration of our app that we are going to run.



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

As we haven't set up a virtual device yet, it says **No devices**.

- In order to create a virtual device, click on the **AVD Manager** (**AVD** stands for **Android Virtual Device**) to open the virtual devices window/screen. The option to do this can also be accessed from the **Tools** menu:

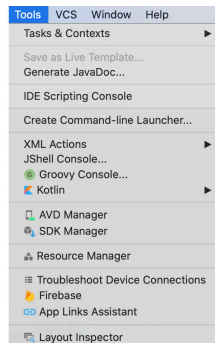


Figure 1.7: AVD Manager in the Tools menu

- Click the button or toolbar option to open the **Your Virtual Devices** window:

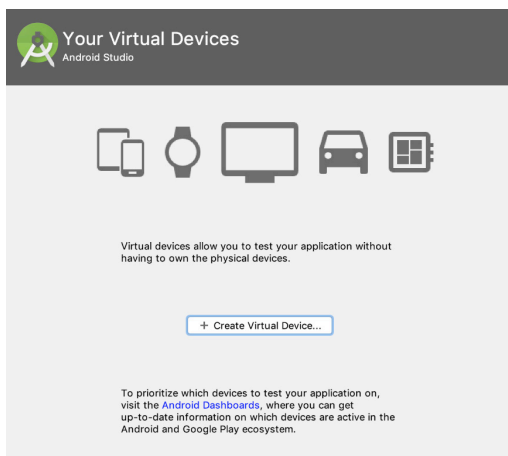


Figure 1.8: The Your Virtual Devices window

- Click the **Create Virtual Device...** button as shown in *Figure 1.8*:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

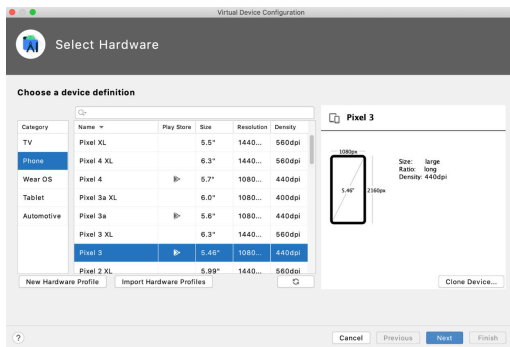


Figure 1.9: Device definition creation

5. We are going to choose the **Pixel 3** device. The real (non-virtual device) Pixel range of devices are developed by Google and have access to the most up-to-date versions of the Android platform. Once selected, click the **Next** button:

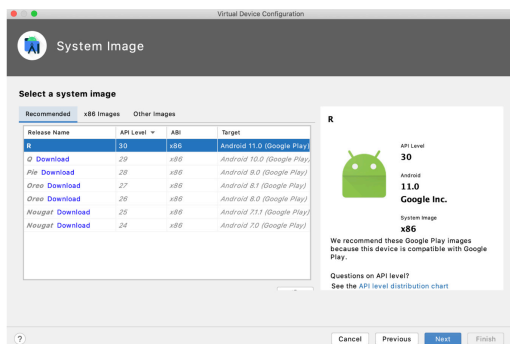


Figure 1.10: System Image

The **R** name displayed here is the initial code/release name for Android 11. Select the latest system image available. The **Target** column might also show **(Google Play)** or **(Google APIs)** in the name. Google APIs mean that the system image comes pre-installed with Google Play Services. This is a rich feature set of Google APIs and Google apps

**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



that your app can use and interact with. On first running the app, you will see apps such as Maps and Chrome instead of a plain emulator image. A Google Play system image means that, in addition to the Google APIs, the Google Play app will also be installed.

- You should develop your app with the latest version of the Android platform to benefit from the latest features. On first creating a virtual device, you will have to download the system image. If a **Download** link is displayed next to **Release Name**, click on it and wait for the download to complete. Select the **Next** button to see the virtual device you have set up:

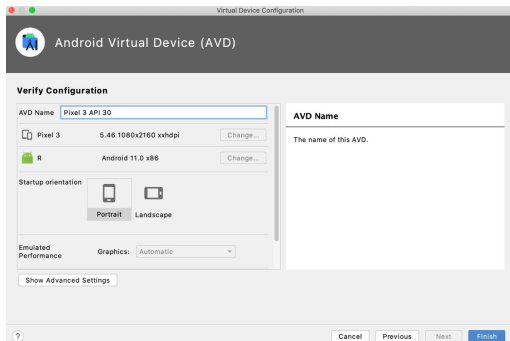


Figure 1.11: Virtual device configuration

You will then see a final configuration screen.

- Click **Finish** and your virtual device will be created. You will then see your device highlighted:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



Figure 1.12: Virtual devices listed

8. Press the right arrow button under the **Actions** column to run up the virtual device:

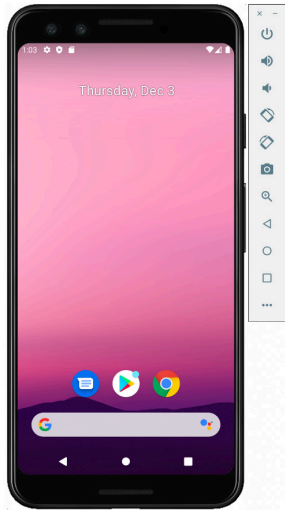


Figure 1.13: Virtual device launched

Now that you've created the virtual device and it's running, you can go back into Android Studio to run your app.

9. The virtual device you have set up and started will be selected. Press the green triangle/play button to launch your app:



Figure 1.14: App launch configuration



**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

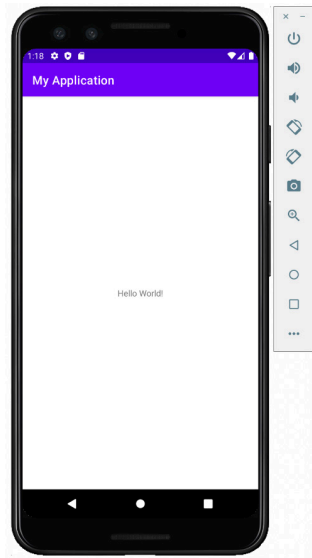


Figure 1.15: App running on a virtual device

In this exercise, you have gone through the steps to create a virtual device and run the app you created on it. The Android Virtual Device Manager, which you have used to do this, enables you to create the device (or range of devices) you would like to target your app for. Running your app on the virtual device allows a quick feedback cycle to verify how a new feature development behaves and that it displays the way you expect it to.

Next, you will explore the **AndroidManifest.xml** file of your project, which contains the information and configuration of your app.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

# The Android Manifest

The app you have just created, although simple, encompasses the core building blocks that you will use in all of the projects you create. The app is driven from the **AndroidManifest.xml** file, a manifest file that details the contents of your app. It has all the components, such as activities, content providers, services, receivers, and the list of permissions that the app requires to implement its features. For example, the camera permission is required to capture a photo in an app. You can find it in the Project view under **MyApplication | app | src | main**. Alternatively, if you are looking at the Android view, it is located at **app | manifests | AndroidManifest.xml**:

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



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

<!--Permissions like
camera go here-->
<application
    android:allowBacku
p="true"
    android:icon="@mip
map/ic_launcher"
    android:label="@st
ring/app_name"
    android:roundIcon=
"@mipmap/ic_launcher_round
"
    android:supportsRt
l="true"
    android:theme="@st
yle/Theme.MyApplication">
    <activity
android:name=".MainActivit
y"
android:screenOrientation=
"portrait">
        <intent-
filter>
            <action
android:name="android.inte
nt.action.MAIN" />
            <category
android:name="android.inte
nt.category.LAUNCHER" />
        </intent-
filter>
    </activity>
</application>
</manifest>

```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

A typical manifest file in general terms is a top-level file that describes the enclosed files or other data and associated metadata that forms a group or unit. The Android Manifest takes this concept and applies it to your Android app as an XML file. The distinguishing feature of the app specified is the package defined at the manifest XML root:

```
package="com.example.myapplication"
```

Every Android app has an application class that allows you to configure the app. By default, in version 4.1.1 of Android Studio, the following XML attributes and values are created in the application element:

- **android:allowBackup="true":**  
This backs up a user's data from apps that target and run on Android 6.0 (API level 23) or later upon reinstall or switching devices.
- **android:icon="@mipmap/ic\_launcher":**  
The resources Android uses are referenced in XML preceded by the @ symbol and mipmap refers to the folder where launcher icons are stored.
- **android:label="@string/app\_name":**  
This is the name you specified



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

when you created the app. It's also currently displayed in the toolbar of the app and will be shown as the name of the app on the user's device in the launcher. It is referenced by the @ symbol followed by a string reference to the name you specified when you created the app.

- **android:roundIcon="@mipmap/ic\_launcher\_r**  
Depending on the device the user has, the launcher icons may be square or round. **roundIcon** is used when the user's device displays round icons in the launcher.
- **android:supportsRtl="true":**  
This specifies whether the app and its layout files support right-to-left language layouts.
- **android:theme="@style/Theme.MyApplicatio**  
This specifies the theme of your app in terms of text styles, colors, and other styles within your app.

After the **<application>** element opens, you define the components your app consists of. As we have just created our app, it only contains the first screen shown in the following code:

```
<activity
    android:name=".MainActivit
y">
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

The next child XML node specified is as follows:

```
<intent-filter>
```

Android uses intents as a mechanism for interacting with apps and system components. Intents get sent and the intent filter registers your app's capability to react to these intents.

**<android.intent.action.MAIN>** is the main entry point into your app, which, as it appears in the enclosing XML of **.MainActivity**, specifies that this screen will be started when the app is launched.

**android.intent.category.LAUNCHER** states that your app will appear in the launcher of your user's device.

As you have created your app from a template, it has a basic manifest that will launch the app and display an initial screen at startup through an **Activity** component. Depending on which other features you want to add to your app, you may need to add permissions in the Android Manifest file.

Permissions are grouped into three different categories: normal, signature, and dangerous.



**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



- **Normal** permissions include accessing the network state, Wi-Fi, the internet, and Bluetooth. These are usually permitted without asking the user's consent at runtime.
- **Signature** permissions are those shared by the same group of apps that have to be signed with the same certificate. This means these apps can share data freely, but other apps can't get access.
- **Dangerous** permissions are centered around the user and their privacy, for example, sending SMS, access to accounts and location, and reading and writing to the filesystem and contacts.

These permissions have to be listed in the manifest, and, in the case of dangerous permissions from Android Marshmallow API 23 (Android 6 Marshmallow) onward, you must also ask the user to grant the permissions at runtime.

In the next exercise, we will configure the Android Manifest file.

## Exercise 1.03: Configuring the Android Manifest Internet Permission



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

The key permission that most apps require is access to the internet. This is not added by default. In this exercise, we will fix that and, in the process, load a **WebView**, which enables the app to show web pages. This use case is a very common one in Android app development as most commercial apps will display a privacy policy, terms and conditions, etc. As these documents are likely to be common to all platforms, the usual way to display them is to load a web page. Perform the following steps:

1. Create a new Android Studio project as you did in *Exercise 1.01, Creating an Android Studio Project for Your App*.
2. Switch tabs to the **MainActivity** class. From the main project window, it's located in **MyApplication | app | src | main | java | com | example | myapplication**. This follows the package structure you defined when creating the app. Alternatively, if you are looking at the Android view within the project window, it is located at **app | java | com | example | myapplication**.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

You can change what the **Project** window displays by opening up the **Tool** window by selecting **View | Tool Windows | Project** - this will select **Project** view. The drop-down options on the top of the **Project** window allow you to change the way you view your project, with the most commonly used displays being **Project** and **Android**.

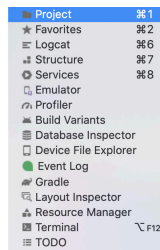


Figure 1.16 Tool Windows drop-down

On opening it, you'll see that it has the following content or similar:

```
package
com.example.myapplication
import
androidx.appcompat.app.
AppCompatActivity
import
android.os.Bundle
class MainActivity :
AppCompatActivity() {
    override fun
    onCreate(savedInstanceState: Bundle?) {
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        super.onCreate(
            savedInstanceState)

        setContentView(
            R.layout.Activity_main)
    }
}

```

You'll examine the contents of this file in more detail in the next section of this chapter, but for now, you just need to be aware that the

**setContentView(R.layout.Activity\_main)** statement sets the layout of the UI you saw when you first ran the app in the virtual device.

3. Use the following code to change this to the following:

```

package
com.example.myapplication

import
androidx.appcompat.app.
AppCompatActivity
import
android.os.Bundle
import
android.webkit.WebView
class MainActivity :
AppCompatActivity() {
    override fun
onCreate(savedInstanceState: Bundle?) {
        super.onCreate(
            savedInstanceState)
    }
}

```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        val webView =
            WebView(this)
            webView.setting
            s.javaScriptEnabled =
            true
            setContentView(
            webView)
            webView.loadUrl
            ("https://www.google.co
            m")
        }
    }

```

So, you are replacing the layout file with a **WebView**. The **val** keyword is a read-only property reference, which can't be changed once it has been set. JavaScript needs to be enabled in the WebView to execute JavaScript.

#### Note

*We are not setting the type, but Kotlin has type inference, so it will infer the type if possible. So, specifying the type explicitly with **val webView: WebView = WebView(this)** is not necessary. Depending on which programming languages you have used in the past, the order of defining the parameter name and type may or may not be familiar. Kotlin follows Pascal notation, that is, name followed by type.*



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

4. Now, run the app up and the text will appear as shown in the screenshot here:

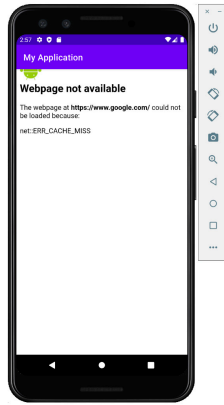


Figure 1.17 No internet permission error message

5. This error occurs because there is no **INTERNET** permission added in your **AndroidManifest.xml** file. (If you get the error **net::ERR\_CLEARTEXT\_NOT\_PERMITTED**, this is because the URL you are loading into the **WebView** is not HTTPS and non-HTTPS traffic is disabled from API level 28, Android 9.0 Pie and above.) Let's fix that by adding the internet permission to the manifest. Open up the Android Manifest and add the following to above the **<application>** tag:

```
<uses-permission
    android:name="android.p
    ermission.INTERNET" />
```

Your manifest file should now look like the following:

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



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">
    <uses-permission
android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity
android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

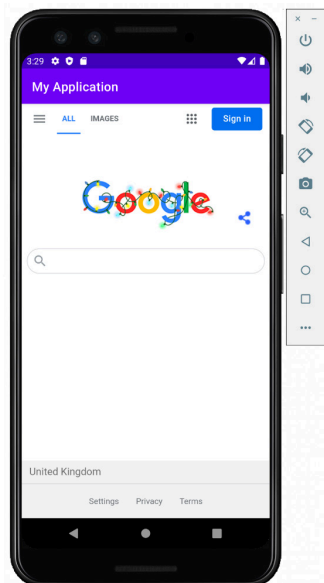
</intent-
filter>
</activity>
</application>
</manifest>

```

Uninstall the app from the virtual device before running up the app again. You need to do this as app permissions can sometimes get cached. Do this by long pressing on the app icon and selecting the **App Info** option that appears and then pressing the Bin icon with **Uninstall** text below it.

Alternatively, long press the app icon and then drag it to the Bin icon with **Uninstall** text beside it in the top-right corner of the screen.

6. Install the app again and see the web page appear in the **WebView**:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



## Figure 1.18 App displaying the WebView

In this example, you learned how to add a permission to the manifest. The Android Manifest can be thought of as a table of contents of your app. It lists all the components and permissions your app uses. As you have seen from starting the app from the launcher, it also provides the entry points into your app.

In the next section, you will explore the Android build system, which uses the Gradle build tool to get your app up and running.

## Using Gradle to Build, Configure, and Manage App Dependencies

In the course of creating this project, you have principally used the Android platform SDK. The necessary Android libraries were downloaded when you installed Android Studio. These are not the only libraries, however, that are used to create your app. In order to configure and build your Android



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

project or app, a build tool called Gradle is used. Gradle is a multi-purpose build tool that Android Studio uses to build your app. By default, in Android Studio, it uses Groovy, a dynamically typed JVM language, to configure the build process and allows easy dependency management so you can add libraries to your project and specify the versions. Android Studio can also be configured to use Kotlin to configure builds, but, as the default language is Groovy, you will be using this. The files that this build and configuration information is stored in are named **build.gradle**. When you first create your app, there are two **build.gradle** files, one at the root/top level of the project and one specific to your app in the app **module** folder.

## Project-Level **build.gradle** file

Let's now have a look at the project-level **build.gradle** file. This is where you add configuration options common to all sub-projects/modules, as shown in the following code:



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
buildscript {  
    ext.kotlin_version =  
    "1.4.21"  
    repositories {  
        google()  
        jcenter()  
    }  
    dependencies {  
        classpath  
        "com.android.tools.build:gradle:4.4.1"  
        classpath  
        "org.jetbrains.kotlin:kotlin-in-gradle-plugin:  
$kotlin_version"  
        // NOTE: Do not  
        place your application  
        dependencies here;  
        //they belong in  
        the individual module  
        build.gradle files  
    }  
}  
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}  
task clean(type: Delete) {  
    delete  
    rootProject.buildDir  
}
```

The **buildscript** block has build and configuration information to



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

actually create your project, whilst the **allprojects** block specifies the configuration for all of the app's modules. Groovy works on a plugin system, so you can write your own plugin that does a task or series of tasks and plug it into your build pipeline. The two plugins specified here are the Android tools plugin, which hooks into the **gradle** build toolkit and provides Android-specific settings and configuration to build your Android app, and the Kotlin **gradle** plugin, which takes care of compiling Kotlin code within the project. The dependencies themselves follow the Maven **Project Object Model (POM)** convention of **groupId**, **artifactId**, and **versionId** separated by ":" colons. So as an example, the Android tools plugin dependency above is shown as:

```
'com.android.tools.build:gradle:4.4.1'
```

The **groupId** is **com.android.tools.build**, the **artifactId** is **gradle**, and the **versionId** is **4.4.1**. In this way, the build system locates and downloads these dependencies by using the repositories referenced in the **repositories** block.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

The specific versions of libraries can be specified directly (as is done with the Android **tools** plugin) in the dependency or added as variables. The **ext.** prefix on the variable means it is a Groovy extension property and can be used in the app **build.gradle** file as well.

### Note

*The dependency versions specified in the previous code section and in the following sections of this and other chapters are subject to change, and are updated over time so are likely to be higher when you create these projects.*

## App-Level build.gradle

The **build.gradle** app is specific to your project configuration:

```
plugins {  
    id  
    'com.android.application'  
    id 'kotlin-android'  
}  
android {  
    compileSdkVersion 30  
    buildToolsVersion  
    "30.0.3"  
    defaultConfig {
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        applicationId
"com.example.myapplication"
"
        minSdkVersion 16
        targetSdkVersion
30
        versionCode 1
        versionName "1.0"
        टेस्टइंस्ट्रूमेंटेशनरनर
"androidx.test.runner.Android
oidJUnitRunner"
        बिल्डटाइप्स {
            मुक्त करना {
                minifyEnab
led गलत
                proguardFi
les
getDefaultProguardFile(
'proguard-android-
optimize.txt'), 'proguard-
rules.pro'
            }
        }
        संकलन विकल्प {
            स्रोतसंगतता
JavaVersion.VERSION_1_8
            लक्ष्यसंगतता
JavaVersion.VERSION_1_8
        }
        कोटलिनविकल्प {
            जेवीएमटार्गेट =
'1.8'
        }
        निर्भरताएँ {

```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

कार्यान्वयन

```
"org.jetbrains.kotlin:kotl
in-stdlib:
$kotlin_version"
```

कार्यान्वयन

```
'androidx.core:core-
ktx:1.3.2'
```

कार्यान्वयन

```
'androidx.appcompat:appcom
pat:1.2.0'
```

कार्यान्वयन

**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

**O'REILLY**

कार्यान्वयन

```
'androidx.constraintlayout
:constraintlayout:2.0.4'
```

परीक्षण कार्यान्वयन

```
'junit:junit:4.+'
```

```
androidTestImpleme
ntation
```

```
'androidx.test.ext:junit:1
.1.2'
```

```
androidTestImpleme
ntation
```

```
'androidx.test.espresso
:espresso-core:3.3.0'
}
```

}

पिछले विवरण में वर्णित एंड्रॉइड और कोटलिन के लिए प्लगइन्स, **प्लगइन्स** पंक्तियों में आईडी द्वारा आपके प्रोजेक्ट पर लागू किए जाते हैं।

**com.android.application** प्लगइन द्वारा प्रदान किया गया android ब्लॉक वह

स्थान है जहाँ आप अपनी Android-विशिष्ट कॉन्फ़िगरेशन सेटिंग्स कॉन्फ़िगर करते हैं:

- **compileSdkVersion** : इसका उपयोग उस API स्तर को परिभाषित करने के लिए किया जाता है जिस पर ऐप संकलित किया गया है और ऐप इस API और उससे कम स्तर की सुविधाओं का उपयोग कर सकता है।
- **buildToolsVersion** : आपके ऐप को बनाने के लिए बिल्ड टूल्स का संस्करण। (डिफ़ॉल्ट रूप से **buildToolsVersion** लाइन आपके प्रोजेक्ट में जोड़ दी जाएगी, लेकिन बिल्ड टूल्स के हमेशा नवीनतम संस्करण का उपयोग करने के लिए आप इसे हटा सकते हैं)।
- **defaultConfig** : यह आपके ऐप का मूल कॉन्फ़िगरेशन है।
- **applicationId** : यह आपके ऐप के पैकेज पर सेट किया जाता है और यह ऐप पहचानकर्ता है जिसका उपयोग Google Play पर आपके ऐप को विशिष्ट रूप से पहचानने के लिए किया जाता है। यदि आवश्यक हो तो इसे पैकेज नाम से अलग किया जा सकता है।
- **minSdkVersion** : वह न्यूनतम API स्तर जिसे आपका ऐप सपोर्ट करता है। यह आपके ऐप को Google Play में इससे कम स्तर वाले डिवाइस पर दिखाए जाने से फ़िल्टर कर देगा।
- **targetSdkVersion** : वह API स्तर जिसे आप लक्षित कर रहे हैं। यह वह API स्तर है जिसके साथ आपका बनाया गया



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



ऐप काम करने के लिए अभिप्रेत है और जिसके साथ इसका परीक्षण किया गया है।

- **versionCode** : आपके ऐप का वर्शन कोड निर्दिष्ट करता है। हर बार जब ऐप में कोई अपडेट करने की आवश्यकता होती है, तो वर्शन कोड को 1 या उससे अधिक से बढ़ाना पड़ता है।
- **versionName** : एक उपयोगकर्ता-अनुकूल संस्करण नाम जो आमतौर पर XYZ के अर्थपूर्ण संस्करण का अनुसरण करता है, जहां X प्रमुख संस्करण है, Y लघु संस्करण है, और Z पैच संस्करण है, उदाहरण के लिए, 1.0.3।
- **testInstrumentationRunner** : आपके UI परीक्षणों के लिए उपयोग किया जाने वाला टेस्ट रनर।
- **buildTypes** : **buildTypes** के अंतर्गत, एक रिलीज़ जोड़ी जाती है जो आपके ऐप को **रिलीज़** बिल्ड बनाने के लिए कॉन्फ़िगर करती है। **minifyEnabled** मान, यदि **true** पर सेट किया जाता है, तो किसी भी अप्रयुक्त कोड को हटाकर आपके ऐप का आकार छोटा कर देगा, साथ ही आपके ऐप को अस्पष्ट भी कर देगा। यह अस्पष्टीकरण चरण स्रोत कोड संदर्भों के नाम को **abc()** जैसे मानों में बदल देता है। यह आपके कोड को रिवर्स इंजीनियरिंग के लिए कम प्रवण बनाता है और निर्मित ऐप के आकार को और कम करता है।
- **संकलन विकल्प** : जावा स्रोत कोड ( **स्रोत संगतता** ) और बाइट कोड ( **लक्ष्य संगतता** ) का भाषा स्तर
- **kotlinOptions** : **jvm** लाइब्रेरी जिसे **kotlin gradle** प्लगइन को उपयोग



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

करना चाहिए

निर्भरता ब्लॉक आपके ऐप द्वारा Android प्लेटफ़ॉर्म SDK के शीर्ष पर उपयोग की जाने वाली लाइब्रेरी को निर्दिष्ट करता है, जैसा कि यहां दिखाया गया है:

```

निर्भरताएँ {
    //आपका ऐप जिस कोटलिन
    संस्करण के साथ बनाया जा रहा है
    कार्यान्वयन
    "org.jetbrains.kotlin:kotl
    in-stdlib-jdk7:
    $kotlin_version"
    //कोटलिन एक्सटेंशन, जेटपैक
    //एंड्रॉइड कोटलिन भाषा सुविधाओं
    वाला घटक
    कार्यान्वयन
    'androidx.core:core-
    ktx:1.3.2'
    //पिछड़े संगत समर्थन पुस्तकालय
    और जेटपैक घटक प्रदान करता है
    कार्यान्वयन
    'androidx.appcompat:appcom
    pat:1.2.0'
    //अपने ऐप को थीम और स्टाइल
    देने के लिए मटेरियल डिज़ाइन
    घटक
    कार्यान्वयन
    'com.google.android.materi
    al:material:1.2.1'
    //ConstraintLayout
    ViewGroup को अलग से अपडेट
    किया गया
    //मुख्य एंड्रॉइड स्रोतों से
  
```



**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूं?

**कार्यान्वयन**

```
'androidx.constraintlayout
:constraintlayout:2.0.4'
//यूनिट परीक्षणों के लिए मानक
परीक्षण लाइब्रेरी.
//'+ ' ग्रेडल का डायनामिक
संस्करण है जो डाउनलोड करने
की अनुमति देता है
//नवीनतम संस्करण। क्योंकि इससे
परिवर्तन होने पर अप्रत्याशित बिल्ड
हो सकते हैं
//सभी परियोजनाएं निश्चित
संस्करण '4.13.1' का उपयोग
करेंगी
```

**परीक्षण कार्यान्वयन**

```
'junit:junit:4.+ '
//यूआई टेस्ट रनर
    androidTest
कार्यान्वयन
'androidx.test.runner:1.1.
2 '
//एंड्रॉइड UI परीक्षण बनाने के
लिए लाइब्रेरी
    androidTestImpleme
ntation
'androidx.test.espresso:es
presso-core:3.3.0'
}
```

इन लाइब्रेरीज़ को जोड़ने के लिए कार्यान्वयन संकेतन का अर्थ है कि उनकी आंतरिक निर्भरताएं आपके ऐप पर प्रदर्शित नहीं होंगी, जिससे संकलन तेज़ हो जाएगा ।

आप यहाँ देखेंगे कि **androidx** घटक Android प्लेटफ़ॉर्म स्रोत के बजाय निर्भरता

**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

के रूप में जोड़े गए हैं। ऐसा इसलिए है ताकि उन्हें Android संस्करणों से स्वतंत्र रूप से अपडेट किया जा सके। **androidx** पुनः पैक की गई सहायता लाइब्रेरी और Jetpack घटक है। यह जोड़ने या सत्यापित करने के लिए कि आपकी **gradle.properties** फ़ाइल में **androidx** सक्षम है, आपको अपने प्रोजेक्ट के रूट पर **gradle.properties** फ़ाइल का निरीक्षण करना होगा और **android.useAndroidX** और **android.enableJetifier** गुणों को देखना होगा और सुनिश्चित करना होगा कि वे **true** पर सेट हैं।

**अब आप **gradle.properties** फ़ाइल खोल सकते हैं , और आपको निम्नलिखित दिखाई देगा:**

```
# प्रोजेक्ट-व्यापी ग्रेडल सेटिंग्स.
# IDE (जैसे Android Studio)
उपयोगकर्ता:
# IDE के माध्यम से कॉन्फ़िगर
की गई Gradle सेटिंग्स
*ओवरराइड हो जाएंगी*
# इस फ़ाइल में निर्दिष्ट कोई भी
सेटिंग.
# अपने बिल्ड वातावरण को
कॉन्फ़िगर करने के तरीके के बारे
में अधिक जानकारी के लिए यहां
जाएं
#
http://www.gradle.org/docs
/current/userguide/build_e
nvironment.html
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूं?

```
# डेमॉन प्रक्रिया के लिए प्रयुक्त
JVM तर्कों को निर्दिष्ट करता है।
# यह सेटिंग विशेष रूप से मेमोरी
सेटिंग में बदलाव करने के लिए
उपयोगी है।
org.gradle.jvmargs=-
Xmx2048m -
Dfile.encoding=UTF-8
# कॉन्फिगर होने पर, Gradle
इनक्यूबेटिंग समानांतर मोड में
चलेगा।
# इस विकल्प का उपयोग केवल
पृथक परियोजनाओं के साथ किया
जाना चाहिए।
# अधिक जानकारी के लिए यहां
जाएं
#
http://www.gradle.org/docs
/current/userguide/multi_p
roject_builds
# .html#सेक
#:डिकूपल्ड_प्रोजेक्ट्स
# org.gradle.समानांतर=सत्य
# AndroidX पैकेज संरचना यह
स्पष्ट करने के लिए कि कौन से
पैकेज हैं
# एंड्रॉइड ऑपरेटिंग सिस्टम के
साथ बंडल किया गया है, और जो
पैकेज किए गए हैं
# अपने ऐप के APK के साथ
#
https://developer.android.
com/topic/libraries/suppor
t-library/
#एंड्रॉइडx-आरएन
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूं?

```

android.useAndroidX=सत्य
# AndroidX का उपयोग करने
के लिए तृतीय-पक्ष लाइब्रेरीज़ को
स्वचालित रूप से परिवर्तित करें
android.enableJetifier=सत्य
# इस परियोजना के लिए कोटलिन
कोड शैली: "आधिकारिक" या
"अप्रचलित":
kotlin.code.style=आधिकारिक
क

```

जैसे ही आपने Android Studio टेम्पलेट के साथ प्रोजेक्ट बनाया, इसने इन फ़्लैग को **true** पर सेट कर दिया, साथ ही ऐप द्वारा उपयोग की जाने वाली प्रासंगिक **androidx** निर्भरता को ऐप की **build.gradle** फ़ाइल के **dependencies** ब्लॉक में जोड़ दिया। पहले से टिप्पणी की गई व्याख्या के अलावा,

**android.useAndroidX=true** फ़्लैग बताता है कि प्रोजेक्ट पुराने सपोर्ट लाइब्रेरी के बजाय **androidx लाइब्रेरी का उपयोग कर रहा है और**

**android.enableJetifier=true** थर्ड-पार्टी लाइब्रेरी में उपयोग की जाने वाली सपोर्ट लाइब्रेरी के किसी भी पुराने संस्करण को AndroidX फ़ॉर्मेट में बदल देगा।

**kotlin.code.style=official** कोड स्टाइल को डिफ़ॉल्ट Android Studio के बजाय आधिकारिक kotlin पर सेट करेगा।

The final Gradle file to examine is **settings.gradle**. This file shows which modules your app uses. On first creating a project with Android



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

Studio, there will only be one module, **app**, but when you add more features, you can add new modules that are dedicated to containing the source of this feature rather than packaging it in the main **app** module. These are called feature modules, and you can supplement them with other types of modules such as shared modules, which are used by all other modules like a networking module. The **settings.gradle** file will look like this:

```
include ':app'
rootProject.name='My
Application'
```

## Exercise 1.04: Exploring how Material Design is used to theme an app

In this exercise, you will learn about Google's new design language, **Material Design**, and use it to load a **Material Design** themed app. **Material Design** is a design language created by Google that adds enriched UI elements based on real-world effects such as lighting, depth, shadows, and animations. Perform the following steps:



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

1. Create a new Android Studio project as you did in *Exercise 1.01, Creating an Android Studio Project for Your App*.
2. Firstly, look at the **dependencies** block and find the material design dependency

```
implementation  
'com.google.android.mat  
erial:material:1.2.1'
```

3. Next, open the **themes.xml** file located at **app | src | main | res | values | themes.xml**:

```
<resources  
  xmlns:tools="http://sch  
emas.android.com/tools"  
>  
  <!-- Base  
  application theme. -->  
  <style  
    name="Theme.MyApplicati  
on"  
    parent="Theme.MaterialC  
omponents.DayNight.Dark  
ActionBar">  
    <!-- Primary  
    brand color. -->  
    <item  
      name="colorPrimary">@co  
lor/purple_500</item>  
    <item  
      name="colorPrimaryVaria  
nt">@color/purple_700</  
item>
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



```

        <item
name="colorOnPrimary">@
color/white</item>

        <!-- Secondary
brand color. -->

        <item
name="colorSecondary">@
color/teal_200</item>

        <item
name="colorSecondaryVar
iant">@color/teal_700</
item>

        <item
name="colorOnSecondary"
>@color/black</item>

        <!-- Status bar
color. -->

        <item
name="android:statusBar
Color"
tools:targetApi="1">?
attr/colorPrimaryVarian
t</item>

        <!-- Customize
your theme here. --
>    </style>
</resources>

```

Notice that the parent of **Theme.MyApplication** is **Theme.MaterialComponents.DayNight.DarkAc**

The Material Design dependency added in the **dependencies** block is being used here to apply the theme of the app.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

4. If you run the app now, you will see the default Material themed app as shown in *Figure 1.15*

In this exercise, you've learned how **Material Design** can be used to theme an app. As you are currently only displaying a **TextView** on the screen, it is not clear what benefits material design provides, but this will change when you start using Material UI design widgets more. Now that you've learned how the project is built and configured, in the next section, you'll explore the project structure in detail, learn how it has been created, and gain familiarity with the core areas of the development environment.

## Android Application Structure

Now that we have covered how the Gradle build tool works, we'll explore the rest of the project. The simplest way to do this is to examine the folder structure of the app. There is a tool window at the top left of Android Studio called **Project**, which allows you to browse the contents of your app. By default, it is **open/selected** when



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

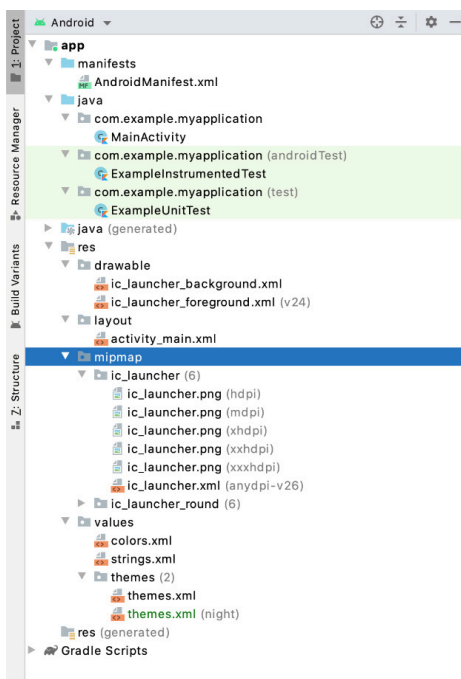
मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

your Android project is first created.

When you select it, you will see a view similar to the screenshot in *Figure 1.19*. (If you can't see any window bars on the left-hand side of the screen, then go to the top toolbar and select **View** | Appearance | **Tool Window Bars** and make sure it is ticked). There are many different options for how to browse your project, but **Android** will be pre-selected. This view neatly groups the **app** folder structure together, so let's take a look at it.

Here is an overview of these files with more detail about the most important ones. On opening it, you will see that it consists of the following folder structure:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

Figure 1.19: Overview of the files and folder structure in the app

The Kotlin file (**MainActivity**), which you've specified as running when the app starts, is as follows:

```
package
com.example.myapplication
import
androidx.appcompat.app.App
CompatActivity
import android.os.Bundle
class MainActivity :
AppCompatActivity() {
    override fun
    onCreate(savedInstanceState: Bundle?) {
        super.onCreate(sav
edInstanceState)
        setContentView(R.l
ayout.activity_main)
    }
}
```

The **import** statements include the libraries and the source of what this activity uses. The class header **class MainActivity :** **AppCompatActivity()** creates a class that extends **AppCompatActivity**. In Kotlin, the **:** colon character is used for both deriving from a class (also known as inheritance) and implementing an interface.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

**MainActivity** derives from **androidx.appcompat.app.AppCompatActivity**, which is the backward-compatible activity designed to make your app work on older devices.

Android activities have many callback functions that you can override at different points of the activity's life. This is known as the **Activity Lifecycle**. For this activity, as you want to display a screen with a layout, you override the **onCreate** function as shown here:

```
override fun  
onCreate(savedInstanceState:  
Bundle?)
```

The **override** keyword in Kotlin specifies that you are providing a specific implementation for a function that is defined in the parent class. The **fun** keyword (as you may have guessed) stands for *function*. The **savedInstanceState: Bundle?** parameter is Android's mechanism for restoring previously saved state. For this simple activity, you haven't stored any state, so this value will be **null**. The question mark, **?**, that follows the type declares that this type can be **null**. The **super.onCreate(savedInstanceState)**



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

line calls through to the overridden method of the base class, and finally,

**setContentView(R.layout.Activity\_main)**

loads the layout we want to display in the activity; otherwise, it would be displayed as a blank screen as no layout has been defined.

Let's have a look at some other files (*Figure 1.19*) present in the folder structure:

- **ExampleInstrumentedTest:** This is an example UI test. You can check and verify the flow and structure of your app by running tests on the UI when the app is running.
- **ExampleUnitTest:** This is an example unit test. An essential part of creating an Android app is writing unit tests in order to verify that the source code works as expected.
- **ic\_launcher\_background.xml/ic\_launcher\_f**  
These two files together make up the launcher icon of your app in vector format, which will be used by the launcher icon file, **ic\_launcher.xml**, in Android API 26 (Oreo) and above.
- **activity\_main.xml:** This is the layout file that was created by Android Studio when we created



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

the project. It is used by **MainActivity** to draw the initial screen content, which appears when the app runs:

```
<?xml version="1.0"
encoding="utf-8"?>
<androidx.constraintlay
out.widget.ConstraintLa
yout
xmlns:android="http://s
chemas.android.com/apk/
res/android"
    xmlns:app="http://s
chemas.android.com/apk/
res-auto"
    xmlns:tools="http:/
/schemas.android.com/to
ols"
    android:layout_widt
h="match_parent"
    android:layout_heig
ht="match_parent"
    tools:context=".Mai
nActivity">
    <TextView
        android:layout_
width="wrap_content"
        android:layout_
height="wrap_content"
        android:text="H
ello World!"
        app:layout_cons
traintBottom_toBottomOf
="parent"
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintEnd_toEndOf="parent"

        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

In order to support internationalization of your app and right-to-left (rtl) layouts, you should remove these attributes if they are present:

```

        app:layout_constraintStart_toLeftOf="parent"

        app:layout_constraintEnd_toRightOf="parent"

```

Replace them with the following:

```

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintEnd_toEndOf="parent"

```

In this way, start and end are determined by the app language, whereas left and right mean start



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



and end only in left-to-right languages.

Most screen displays in Android are created using XML layouts. The document starts with an XML header followed by a top-level **ViewGroup** (which here is **ConstraintLayout**) and then one or more nested **Views** and **ViewGroups**.

The **ConstraintLayout ViewGroup** allows very precise positioning of views on a screen constraining views with parent and sibling views, guidelines, and barriers.

**TextView**, which is currently the only child view of **ConstraintLayout**, displays text on the screen through the **android:text** attribute. The positioning of the view horizontally is done by constraining the view to both the start and end of the parent, which, as both constraints are applied, centers the view horizontally. (start and end in left-to-right languages (**ltr**) are left and right, but right-to-left in **non ltr** languages). The view is positioned vertically in the center by constraining the view to both the top and the bottom of its parent. The result of applying all four



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

constraints centers **TextView** both horizontally and vertically within **ConstraintLayout**.

There are three XML namespaces in the **ConstraintLayout** tag:

- **xmlns:android** refers to the Android-specific namespace and it is used for all attributes and values within the main Android SDK.
- The **xmlns:app** namespace is for anything not in the Android SDK. So, in this case, **ConstraintLayout** is not part of the main Android SDK but is added as a library.
- **xmlns:tools** refers to a namespace used for adding metadata to the XML, which is used to indicate here where the layout is used (**tools:context=".MainActivity"**).

The two most important attributes of an Android XML layout file are **android:layout\_width** and **android:layout\_height**.

These can be set to absolute values, usually of density-independent pixels (known as **dip** or **dp**) that scale pixel sizes to be roughly equivalent on different density



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

devices. More commonly, however, these attributes have the values of **wrap\_content** or **match\_parent** set for them. **wrap\_content** will be as big as required to enclose its contents only. **match\_parent** will be sized according to its parent.

There are other **ViewGroups** you can use to create layouts. **LinearLayout** lays out views vertically or horizontally, **FrameLayout** is usually used to display a single child view, and **RelativeLayout** is a simpler version of **ConstraintLayout**, which lays out views positioned relative to the parent and sibling views.

The **ic\_launcher.png** files are **.png** launcher icons that have an icon for every different density of devices. As the minimum version of Android we are using is API 16: Android 4.1 (Jelly Bean), these **.png** images are included as support for the launcher vector format was not introduced until Android API 26 (Oreo).

The **ic\_launcher.xml** file uses the vector files (**ic\_launcher\_background.xml/ic\_launcher\_for** to scale to different density devices in Android API 26 (Oreo) and above.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूं?

## Note

*In order to target different density devices on the Android platform, besides each one of the `ic_launcher.png` icons, you will see in brackets the density it targets. As devices vary widely in their pixel densities, Google created density buckets so that the correct image would be selected to be displayed depending on how many dots per inch the device has.*

The different density qualifiers and their details are as follows:

- **nodpi**: Density-independent resources
- **ldpi**: Low-density screens of 120 dpi
- **mdpi**: Medium-density screens of 160 dpi (the baseline)
- **hdpi**: High-density screens of 240 dpi
- **xhdpi**: Extra-high-density screens of 320 dpi
- **xxhdpi**: Extra-extra-high-density screens of 480 dpi
- **xxxhdpi**: Extra-extra-extra-high-density screens of 640 dpi
- **tvdpi**: Resources for televisions (approx 213 dpi)



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

The baseline density bucket was created at **160** dots per inch for medium-density devices and is called **mdpi**. This represents a device where an inch of the screen is **160** dots/pixels, and the largest display bucket is **xxxhdpi**, which has **640** dots per inch. Android determines the appropriate image to display based on the individual device. So, the Pixel 3 emulator has a density of approximately **443dpi**, so it uses resources from the extra-extra-high-density bucket (xxhdpi), which is the closest match. Android has a preference for scaling down resources to best match density buckets, so a device with **400dpi**, which is halfway between the **xhdpi** and **xxhdpi** buckets, is likely to display the **480dpi** asset from the **xxhdpi** bucket.

To create alternative bitmap drawables for different densities, you should follow the **3:4:6:8:12:16** scaling ratio between the six primary densities. For example, if you have a bitmap drawable that's **48x48** pixels for medium-density screens, all the different sizes should be:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

- **36x36 (0.75x)** for low density (**ldpi**)
- **48x48 (1.0x baseline)** for medium density (**mdpi**)
- **72x72 (1.5x)** for high density (**hdpi**)
- **96x96 (2.0x)** for extra-high density (**xhdpi**)
- **144x144 (3.0x)** for extra-extra-high density (**xxhdpi**)
- **192x192 (4.0x)** for extra-extra-extra-high density (**xxxhdpi**)

For a comparison of these physical launcher icons per density bucket, refer to the following table:






mdpe	hdpi	xhdpi	xxhdpi	xxxhdpi
				

Figure 1.20: Comparison of principal density bucket launcher image sizes

*Note*

*Launcher icons are made slightly larger than normal images within your app as they will be used by the device's launcher. As some launchers can scale up the image, this is to ensure there is no pixelation and blurring of the image.*



जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

Now you are going to look at some of the resources the app uses. These are referenced in XML files and keep the display and formatting of your app consistent.

In the **colors.xml** file, you define the colors you would like to use in your app in hexadecimal format.

```
<?xml version="1.0"
encoding="utf-8"?>
<resources>
    <color
name="purple_200">#FFBB86F
C</color>
    <color
name="purple_500">#FF6200E
E</color>
    <color
name="purple_700">#FF3700B
3</color>
    <color
name="teal_200">#FF03DAC5<
/color>
    <color
name="teal_700">#FF018786<
/color>
    <color
name="black">#FF000000</co
lor>
    <color
name="white">#FFFFFFF</co
lor>
</resources>
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

The format is based on the RGB color space, so the first two characters are for red, the next two for green, and the last two for blue, where **#00** means none of the color is added to make up the composite color, and **#FF** means all of the color is added.

If you would like some transparency in the color, then precede it with two hexadecimal characters, from **#00** for completely transparent to **#FF** for completely opaque. So, to create blue and 50% transparent blue characters, here's the format:

```
<color
name="colorBlue">#0000FF</
color>

<color
name="colorBlue50PercentTr
ansparent">#770000FF</colo
r>
```

The **strings.xml** file displays all the text displayed in the app:

```
<resources>
    <string
name="app_name">My
Application</string>
</resources>
```

You can use hardcoded strings in your app, but this leads to duplication and also means you



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



cannot customize the text if you want to make the app multilingual. By adding strings as resources, you can also update the string in one place if it is used in different places in the app.

Common styles you would like to use throughout your app are added to the **themes.xml** file.

```
<resources
xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style
name="Theme.MyApplication"
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. -->
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        <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>

```

It is possible to apply style information directly to views by setting **android:textStyle="bold"** as an attribute on **TextView**. However, you would have to repeat this in multiple places for every **TextView** you wanted to display in bold. When you start to have multiple style attributes added to individual views, it adds a lot of duplication and can lead to errors when you want to make a change to all similar views and miss changing a style attribute on one view. If you define a style, you only have to



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

change the style and it will update all the views that have that style applied to them. A top-level theme was applied to the application tag in the **AndroidManifest.xml** file when you created the project and is referred to as a theme that styles all views contained within the app. The colors you have defined in the **colors.xml** file are used here. In effect, if you change one of the colors defined in the **colors.xml** file, it will now propagate to style the app as well.

You've now explored the core areas of the app. You have added **TextView** views to display labels, headings, and blocks of text. In the next exercise, you will be introduced to UI elements that will allow the user to interact with your app.

## Exercise 1.05: Adding Interactive UI Elements to Display a Bespoke Greeting to the User

The goal of this exercise is to add the capability of users to add and edit text and then submit this information to display a bespoke greeting with the entered data. You



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

will need to add editable text views to achieve this. The **EditText** View is typically how this is done and can be added in an XML layout file like this:

```
<EditText
    android:id="@+id/full_
name"
    style="@style/TextAppe
arance.AppCompat.Title"
    android:layout_width="
wrap_content"
    android:layout_height=
"wrap_content"
    android:hint="@string/
first_name" />
```

This uses an android style **TextAppearance.AppCompat.Title** to display a title as shown below:

A screenshot of an Android application showing a text input field with the hint "First name:". The text is in a light gray font on a white background.

Figure 1.21: EditText with hint

Although this is perfectly fine to enable the user to add/edit text, the material **TextInputEditText** and it's wrapper View **TextInputLayout** view gives some polish to the **EditText** display. Let's use the following code:

```
<com.google.android.ma
terial.textfield.TextInput
Layout
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        android:id="@+id/first_name_wrapper"
        style="@style/text_input_greeting"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/first_name_text">
        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/first_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </com.google.android.material.textfield.TextInputLayout>

```

The output is as follows:

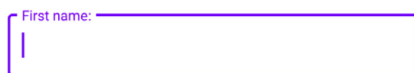


Figure 1.22: Material `TextInputLayout/TextInputEditText` with hint

**`TextInputLayout`** allows us to create a label for the **`TextInputEditText`** view and does a nice animation when the **`TextInputEditText`** view is focused (moving to the top of the



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

field) while still displaying the label.

The label is specified with

**android:hint.**

You are going to change the **Hello World** text in your app so a user can enter their first and last name and further display a greeting on pressing a button. Perform the following steps:

1. Create the labels and text you are going to use in your app by adding these entries to **app | src | main | res | values |**

**strings.xml:**

```
<resources>
    <string
name="app_name">My
Application</string>
    <string
name="first_name_text">
First name:</string>
    <string
name="last_name_text">L
ast name:</string>
    <string
name="enter_button_text
">Enter</string>
    <string
name="welcome_to_the_ap
p">Welcome to the
app</string>
    <string
name="please_enter_a_na
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
me">Please enter a full
name!
</string>
</resources>
```

2. Next, we are going to update our styles to use in the layout by adding the following styles to **app** | **src** | **main** | **res** | **themes.xml** after the Base application theme)

```
<resources
xmlns:tools="http://schemas.android.com/tools"
>

    <!-- Base
application theme. -->
    <style
name="Theme.MyApplication"
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>
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

        <!-- Secondary
brand color. -->
        <item
name="colorSecondary">@
color/teal_200</item>
        <item
name="colorSecondaryVar
iant">@color/teal_700</
item>
        <item
name="colorOnSecondary"
>@color/black</item>
        <!-- Status bar
color. -->
        <item
name="android:statusBar
Color"
tools:targetApi="l">?
attr/colorPrimaryVarian
t</item>
        <!-- Customize
your theme here. -->
    </style>
    <style
name="text_input_greeti
ng"
parent="Widget.Material
Components.TextInputLay
out.OutlinedBox">
        <item
name="android:layout_ma
rgin">8dp</item>
    </style>
    <style
name="button_greeting">

```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



```

        <item
name="android:layout_ma
rgin">8dp</item>
        <item
name="android:gravity">
center</item>
    </style>
</style>
name="greeting_display"
parent="@style/TextAppe
arance.MaterialComponen
ts.Body1">
        <item
name="android:layout_ma
rgin">8dp</item>
        <item
name="android:gravity">
center</item>
        <item
name="android:layout_he
ight">40dp</item>
    </style>
</style>
name="screen_layout_mar
gin">
        <item
name="android:layout_ma
rgin">12dp</item>
    </style>
</resources>

```

### Note

*The parents of some of the styles refer to material styles, so these styles will be applied directly to*



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

*the views as well as the styles that are specified.*

3. Now that we have added the styles we want to apply to views in the layout and the text, we can update the layout in **activity\_main.xml** in **app** | **src** | **main** | **res** | **layout** folder. The code below is truncated for space, but you can view the full source code using the link below.
- activity\_main.xml

```

10    <com.google.andro
id.material.textfield.T
extInputLayout
11        android:id="@
+id/first_name_wrapper"
12        style="@style
/text_input_greeting"
13        android:layou
t_width="match_parent"
14        android:layou
t_height="wrap_content"
15        android:hint=
"@string/first_name_tex
t"
16        app:layout_co
nstraintTop_toTopOf="pa
rent"
17        app:layout_co
nstraintStart_toStartOf
="parent">
18

```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
19         <com.google.a
ndroid.material.textfie
ld.TextInputEditText
20             android:i
d="@+id/first_name"
21             android:l
ayout_width="match_pare
nt"
22             android:l
ayout_height="wrap_cont
ent" />
23
24     </com.google.andr
oid.material.textfield.
TextInputLayout>
25
26     <com.google.andro
id.material.textfield.T
extInputLayout
27         android:id="@
+id/last_name_wrapper"
28         style="@style
/text_input_greeting"
29         android:layu
t_width="match_parent"
30         android:layu
t_height="wrap_content"
31         android:hint=
"@string/last_name_text
"
32         app:layout_co
nstraintTop_toBottomOf=
"@id/first_name_wrapper
"
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
33         app:layout_co
nstraintStart_toStartOf
="parent">
34
35         <com.google.a
ndroid.material.textfie
ld.TextInputEditText
36             android:i
d="@+id/last_name"
37             android:l
ayout_width="match_pare
nt"
38             android:l
ayout_height="wrap_cont
ent" />
39
40     </com.google.andr
oid.material.textfield.
TextInputLayout>
41
42     <com.google.andro
id.material.button.Mate
rialButton
43         android:layou
t_width="match_parent"
44         android:layou
t_height="wrap_content"
45         style="@style
/button_greeting"
46         android:id="@
+id/enter_button"
47         android:text=
"@string/enter_button_t
ext"
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

48         app:layout_co
nstraintTop_toBottomOf=
"@id/last_name_wrapper"
49         app:layout_co
nstraintStart_toStartOf
="parent"/>
50
51     <TextView
52         android:id="@
+id/greeting_display"
53         android:layou
t_width="match_parent"
54         style="@style
/greeting_display"
55         app:layout_co
nstraintTop_toBottomOf=
"@id/enter_button"
56         app:layout_co
nstraintStart_toStartOf
="parent" />

```

*The complete code for this step can be found at*

<http://packt.live/35T5IMN>.

You have added IDs for all the views so they can be constrained against their siblings and also provided a way in the activity to get the values of the `TextInputEditText` views. The `style="@style..."` notation applies the style from the `themes.xml` file.

4. Run the app and see the look and feel. If you select one of the



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

**TextInputEditText** views, you'll see the label animated and move to the top of the view:

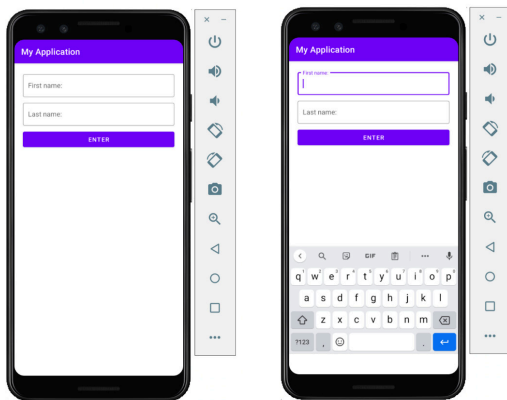


Figure 1.23: **TextInputEditText** fields with label states with no focus and with focus

5. Now, we have to add the interaction with the view in our activity. The layout by itself doesn't do anything other than allow the user to enter text into the **EditText** fields. Clicking the button at this stage will not do anything. You will accomplish this by capturing the entered text by using the IDs of the form fields when the button is pressed and then using the text to populate a **TextView** message.
6. Open **MainActivity** and complete the next steps to process the entered text and use this data to display a greeting and handle any form input errors.
7. In the **onCreate** function, set a click listener on the button so we

**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

can respond to the button click and retrieve the form data by updating **MainActivity** to what is displayed below:

```
package
com.example.myapplication
import
androidx.appcompat.app.
AppCompatActivity
import
android.os.Bundle
import
android.view.Gravity
import
android.widget.Button
import
android.widget.TextView
import
android.widget.Toast
import
com.google.android.material.textfield.TextInputEditText
class MainActivity :
AppCompatActivity() {
    override fun
    onCreate(savedInstanceState: Bundle?) {
        super.onCreate(
savedInstanceState)
        setContentView(
R.layout.activity_main)
        findViewById<Bu
tton>
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```

(R.id.enter_button)?.setOnClickListener {
    //Get the
    greeting display text
    val
    greetingDisplay =
    findViewById<TextView>
    (R.id.greeting_display)
    //Get the
    first name
    TextInputEditText value
    val
    firstName =
    findViewById<TextInputE
    ditText>
    (R.id.first_name)?.text
    .toString().trim()
    //Get the
    last name
    TextInputEditText value
    val
    lastName =
    findViewById<TextInputE
    ditText>
    (R.id.last_name)?.text.
    toString().trim()
    //Check
    names are not empty
    here:
    }
}
}

```

8. Then, check that the trimmed names are not empty and format



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



the name using Kotlin's string templates:

```
if
(firstName.isNotEmpty()
&&
lastName.isNotEmpty())
{
    val nameToDisplay =
firstName.plus("
").plus(lastName)
    //Use Kotlin's
string templates
feature to display the
name
    greetingDisplay?.te
xt =
        "
    ${getString(R.string.we
lcome_to_the_app)}
    ${nameToDisplay}!"
}
```

9. Finally, show a message if the form fields have not been filled in correctly:

```
else {
    Toast.makeText(this
,
getString(R.string.plea
se_enter_a_name),
Toast.LENGTH_LONG).
    apply{
        setGravity(Grav
ity.CENTER, 0, 0)
        show()
    }
```



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

}

The **Toast** specified is a small text dialog that appears above the main layout for a short time to display a message to the user before disappearing.

- Run up the app and enter text into the fields and verify that a greeting message is shown when both text fields are filled in, and a pop-up message appears with why the greeting hasn't been set if both fields are not filled in. You should see the following display for each one of these cases:

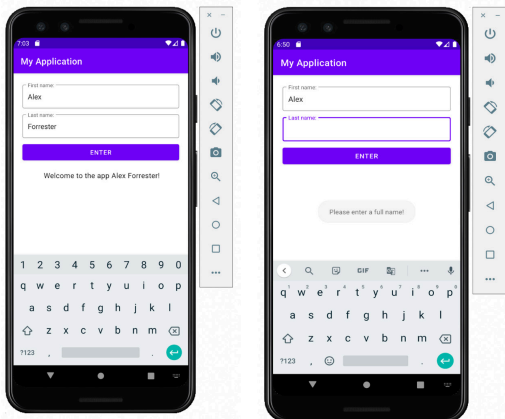


Figure 1.24: App with name filled in correctly and with error

The full exercise code can be viewed here: <http://packt.live/39JyOzB>.

The preceding exercise has introduced you to adding interactivity to your app with **EditText** fields that a user can fill in, adding a click listener to respond



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

to button events and perform some validation.

## Accessing Views in layout files

The established way to access Views in layout files is to use **findViewById** with the name of the View's id. So the **enter\_button** **Button** is retrieved by the syntax **findViewById<Button>(R.id.enter\_button)** after the layout has been set in **setContentView(R.layout.activity\_main)** in the Activity. You will use this technique in this course. Google has also introduced ViewBinding to replace **findViewById** which creates a binding class to access Views and has the advantage of null and type safety. You can read about this here: <https://developer.android.com/topic/libraries/view-binding>.

## Further Input Validation

Validating user input is a key concept in processing user data and you must have seen it in action many times when you've not entered a required field in a form. This is what the previous exercise was validating when it checked that



**जवाब** | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

the user had entered values into both the first name and last name field.

There are other validation options that are available directly within XML view elements. Let's say, for instance, you wanted to validate an IP address entered into a field. You know that an IP address can be four numbers separated by periods/dots where the maximum length of a number is 3. So, the maximum number of characters that can be entered into the field is 15, and only numbers and periods can be entered. There are two XML attributes that can help us with the validation:

- **`android:digits="0123456789."`**: Restricts the characters that can be entered into the field by listing all the permitted individual characters.
- **`android:maxLength="15"`**: Restricts the user from entering more than the maximum number of characters an IP address will consist of.

So, this is how you could display this in a form field:



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

```
<com.google.android.material.textfield.TextInputLayout

    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"

    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <com.google.android.material.textfield.TextInputEditText

        android:id="@+id/ip_address"
        android:digits="0123456789."
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxLength="15" />
</com.google.android.material.textfield.TextInputLayout>
```

This validation restricts the characters that can be input and the maximum length. Additional validation would be required on the sequence of characters and whether they are periods/dots or numbers, as per the IP address format, but it is the first step to assist the user in entering the correct characters.



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

With the knowledge gained from the chapter, let's start with the following activity.

## Activity 1.01: Producing an App to Create RGB Colors

In this activity, we will look into a scenario that uses validation. Suppose you have been tasked with creating an app that shows how the RGB channels of Red, Green, and Blue are added to the RGB color space to create a color. Each of the RGB channels should be added as two hexadecimal characters, where each character can be a value of 0-9 or A-F. The values will then be combined to produce a 6-character hexadecimal string that is displayed as a color within the app.

The aim of this activity is to produce a form with editable fields in which the user can add two hexadecimal values for each color. After filling in all three fields, the user should click a button that takes the three values and concatenates them to create a valid hexadecimal color string. This should then be converted to a color and displayed in the UI of the app.

The following steps will help you to complete the activity:



### जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

1. Create a new project called **Colors**
2. Add a title to the layout constrained to the top of the layout.
3. Add a brief description to the user on how to complete the form.
4. Add three material **TextInputLayout** fields wrapping three **TextInputEditText** fields that appear under **Title**. These should be constrained so that each view is on top of the other (rather than to the side). Name the **TextInputEditText** fields **Red Channel**, **Green Channel**, and **Blue Channel**, respectively, and add a restriction to each field to only be able to enter two characters and add hexadecimal characters.
5. Add a button that takes the inputs from the three-color fields.
6. Add a view that will display the produced color in the layout.
7. Finally, display the RGB color created from the three channels in the layout.

The final output should look like this (the color will vary depending



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?

on the inputs):

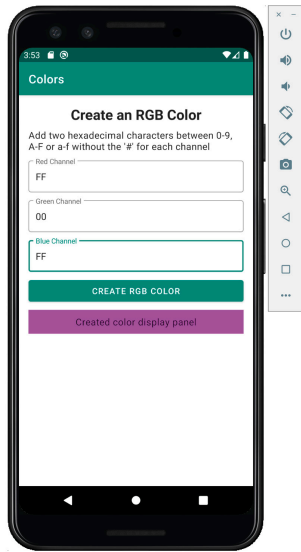


Figure 1.25: Output when the color is displayed

#### Note

The solution to this activity can be found at: <http://packt.live/3sKj1cp>.

The sources for all the exercises and the activity in this chapter are located here:

<http://packt.live/2LLY9kb>

#### Note

When loading all completed projects from the Github repository for this course into Android Studio for the first time, do not open the project using **File | Open from the Top** menu. Always use **File | New | Import Project**. This is needed to build the app correctly. When



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दे

मैं Gradle के साथ Kotlin कैसे बनाऊं?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?



*opening projects after the initial import, you can use File | Open or File | Open Recent.*

## Summary

इस अध्याय में Android विकास की नींव के बारे में बहुत कुछ बताया गया है। आपने Android Studio का उपयोग करके Android प्रोजेक्ट बनाने के तरीके से शुरुआत की और फिर वर्चुअल डिवाइस पर ऐप बनाए और चलाए। फिर अध्याय

**AndroidManifest** फ़ाइल की खोज करके आगे बढ़ा, जो आपके ऐप की सामग्री और अनुमति मॉडल का विवरण देता है, इसके बाद Gradle का परिचय और निर्भरता जोड़ने और अपना ऐप बनाने की प्रक्रिया है। इसके बाद Android एप्लिकेशन और फ़ाइलों और फ़ोल्डर संरचना के विवरण में जाना गया। लेआउट और दृश्य पेश किए गए, और Google के मटीरियल डिज़ाइन के परिचय के साथ UI का निर्माण करने के तरीके को स्पष्ट करने के लिए अभ्यास दोहराए गए। अगला अध्याय गतिविधि जीवनचक्र, गतिविधि कार्यों और लॉन्च मोड, स्क्रीन के बीच डेटा को बनाए रखने और साझा करने और अपने ऐप्स के माध्यम से मजबूत उपयोगकर्ता यात्रा बनाने के तरीके के बारे में सीखकर इस ज्ञान का निर्माण करेगा।



## जवाब | नया-AI द्वारा संचालित

हम आपके लिए किस प्रश्न का उत्तर दें

मैं Gradle के साथ Kotlin कैसे बनाऊँ?

मैं मौजूदा जावा प्रोजेक्ट में कोटलिन का उपयोग कैसे करूँ?