**Chapter 10: Location In Android**

The app shows the device location as latitude and longitude coordinates, then shows the location as a physical address. The completed app does periodic location updates and shows an animation that gives the user a visual cue that the app is tracking the device's location.

**Part 1: Setup an empty project with Google Play Services**

1. Create an empty project called **Location**
2. Update the **strings.xml** file

<**string name="get\_location"**>Get Location</**string**>  
<**string name="textview\_hint"**>Press the button to get the last known location</**string**>  
<**string name="location\_permission\_denied"**>Location permissions denied,  
 please enable the permissions to use this app.  
</**string**>  
<**string name="address\_text"**>"Address: %1$s \n Timestamp: %2$tr"</**string**>  
<**string name="no\_location"**>No location available</**string**>  
<**string name="no\_address\_found"**>No address found</**string**>  
<**string name="loading"**>Loading…</**string**>  
<**string name="service\_not\_available"**>Service not available</**string**>  
<**string name="invalid\_lat\_long\_used"**>Invalid coordinates used</**string**>  
<**string name="start\_tracking\_location"**>Start tracking location</**string**>  
<**string name="stop\_tracking\_location"**>Stop tracking location</**string**>

<**string name="location\_text"**>"Latitude: %1$.4f \n Longitude: %2$.4f \n Timestamp: %3$tr"</**string**>

1. Add the following widgets in **activity\_main.xml** file.

<**Button  
 android:id="@+id/button\_location"  
 style="@style/Widget.AppCompat.Button.Colored"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginLeft="16dp"  
 android:layout\_marginRight="16dp"  
 android:text="@string/get\_location"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"** />  
  
<**TextView  
 android:id="@+id/textview\_location"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_margin="16dp"  
 android:gravity="center"  
 android:text="@string/textview\_hint"  
 android:textAppearance="@style/TextAppearance.AppCompat.Headline"  
 android:textSize="22sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/button\_location"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />

1. Install the Google Repository and update the Android SDK Manager.
2. To add Google Play services to your project, add the following line of code to the dependencies section in your app-level build.gradle (Module: app) file:

implementation 'com.google.android.gms:play-services-location:21.0.1'

1. Configure view binding for the project. Sync the project files.

**Part 2: Get the location**

1. Add the following element to your manifest file, above the <application> element:

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>

1. Add a constant value in the MainActivity class.

companion object {  
public val REQUEST\_LOCATION\_PERMISSION = 1  
}

1. Create an OnClickListener for the **Get Location** button in onCreate() in MainActivity.

1. Create a method stub called getLocation() that takes no arguments and doesn't return anything. Invoke the getLocation() method from the button's onClick() method.
2. In the getLocation() method, check for the ACCESS\_FINE\_LOCATION permission.

* If the permission has not been granted, request it.
* If the permission has been granted, display a message in the logs

fun getLocation() {  
 if (ActivityCompat.checkSelfPermission(  
 this,  
 android.Manifest.permission.*ACCESS\_FINE\_LOCATION* )  
 != PackageManager.*PERMISSION\_GRANTED* ) {  
 ActivityCompat.requestPermissions(  
 this, *arrayOf*<String>(  
 android.Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 android.Manifest.permission.*ACCESS\_COARSE\_LOCATION* ),  
 MainActivity.REQUEST\_LOCATION\_PERMISSION  
 )  
 } else {  
 Log.d("Location", "Location Permission Granted")  
 }  
}

1. Override the onRequestPermissionsResult() method. If the permission was granted, call getLocation(). Otherwise, show a Toast saying that the permission was denied.

override fun onRequestPermissionsResult(  
 requestCode: Int,  
 permissions: Array<String?>, grantResults: IntArray  
) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
 when (requestCode) {  
 MainActivity.REQUEST\_LOCATION\_PERMISSION -> *// If the permission is granted, get the location,  
 // otherwise, show a Toast* if (grantResults.size > 0  
 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED* ) {  
 getLocation()  
 } else {  
 Toast.makeText(  
 this,  
 R.string.*location\_permission\_denied*,  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 }  
}

1. In your MainActivity class, create a member variable of the Location type called mLastLocation.

1. Create a member variable of the FusedLocationProviderClient type called mFusedLocationClient.
2. Initialize mFusedLocationClient in onCreate() with the following code:

*// Initialize the FusedLocationClient.*mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this)

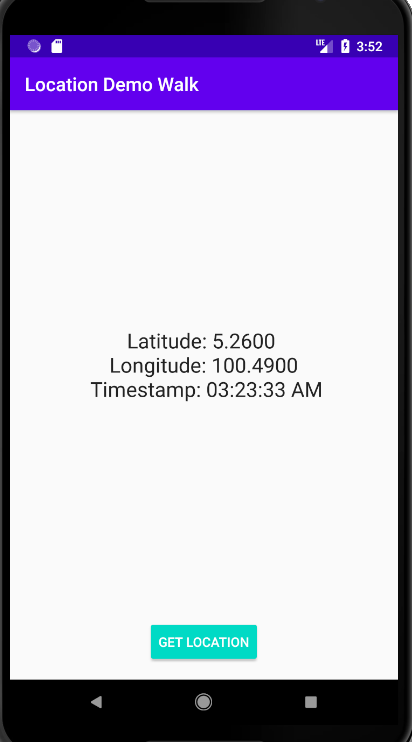
1. Replace the log statement in the getLocation() method with the following code snippet. The code obtains the device's most recent location and assigns it to mLastLocation

mFusedLocationClient!!.*lastLocation*.addOnSuccessListener **{** location **->** if (location != null) {  
  
 mLastLocation = location  
 val locationText =  
 getString(R.string.*location\_text*,  
 mLastLocation?.getLatitude(),  
 mLastLocation?.getLongitude(),  
 mLastLocation?.getTime())

binding.textviewLocation.*text* = locationText

} else {  
 binding.textviewLocation.setText(R.string.*no\_location*)  
 }  
**}**

1. Run the app. You now see the latest location that is stored in the fused location provider.



**Part 3: Get the location as address**

1. Create a new class called FetchAddressTask.

public class FetchAddressTask(private val mContext: Context, var location: Location) {  
 private val TAG = FetchAddressTask::class.*java*.*simpleName* var resultMessage = ""  
  
 fun fetchAddress() {  
 val geocoder = Geocoder(mContext, Locale.getDefault())  
  
 *// Get the passed in location  
 //Location location = locations[0]; //main address stored in the location 0* var addresses: List<Address>? = null  
 resultMessage = ""  
 try {  
 addresses = geocoder.getFromLocation(  
 location.*latitude*, location.*longitude*,  
 1  
 )  
 } catch (ioException: IOException) {  
 *// Catch network or other I/O problems* resultMessage = "Service not available"  
 Log.e(TAG, resultMessage, ioException)  
 } catch (illegalArgumentException: IllegalArgumentException) {  
 *// Catch invalid latitude or longitude values* resultMessage = "Invalid coordinates used"  
 Log.e(  
 TAG, resultMessage + ". " +  
 "Latitude = " + location.*latitude* +  
 ", Longitude = " +  
 location.*longitude*, illegalArgumentException  
 )  
 }  
  
 *// If no addresses found, print an error message.* if (addresses == null || addresses.size == 0) {  
 if (resultMessage.*isEmpty*()) {  
 resultMessage = "No address found"  
 Log.e(TAG, resultMessage)  
 }  
 } else {  
 *// If an address is found, read it into resultMessage* val address = addresses.*first*()  
 val addressParts = ArrayList<String?>()  
  
 *// Fetch the address lines using getAddressLine,  
 // join them, and send them to the thread* for (i in 0..address.*maxAddressLineIndex*) {  
 addressParts.add(address.getAddressLine(i))  
 }  
 resultMessage = TextUtils.join(  
 "\n",  
 addressParts  
 )  
 }  
 }  
}

1. Back in the MainActivity, declare a Corountine object.

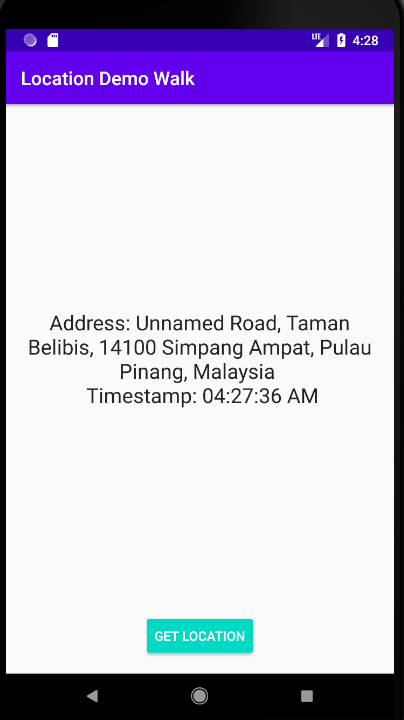
val coroutineScope = *CoroutineScope*(Dispatchers.Main)

1. In the getLocation() method, inside the onSuccess() callback, replace the lines that assigns the passed-in location to mLastLocation and sets the TextView with the following line of code. This code creates a new FetchAddressTask and executes it, passing in the Location object. You can also remove the now unused mLastLocation member variable.

coroutineScope.*launch*(Dispatchers.Main) **{** val addressObject = FetchAddressTask(this@MainActivity, location)  
  
 addressObject.fetchAddress()  
  
 result = addressObject.resultMessage  
 binding.textviewLocation.*text* = locationText + "\n\n" + result

**}**

1. Run the app. After briefly loading, the app displays the location address in the TextView.



1. [Enhance the project] . Add in an additional button call View Location. Create an Implicit intent to view the location using the Google Map in Chrome.