How to Get User's Current Location in an Android App: A Step-by-Step Guide

In today's world, where location-based services have become an integral part of our daily lives, integrating location functionalities into your Android apps is not just an option, but often a necessity. Whether you're building a navigation app, a fitness tracker, or simply want to provide location-based recommendations, understanding how to retrieve and display the user's current location is a critical skill for any Android developer.

In this article, we'll walk you through the process of getting the user's current location in an Android app using Kotlin. We'll be using a practical example project—"Task 1"—that showcases how to request location permissions, retrieve the current location, and display it on a Google Map.

Prerequisites

Before diving into the implementation, ensure you have the following set up:

- **Android Studio**: Installed and updated to the latest version.
- **Kotlin**: Basic understanding of Kotlin programming.
- **Google Maps API Key**: You'll need this to integrate Google Maps into your Android app.

Project Setup

To get started, create a new project in Android Studio:

1. **Create a New Project**: Open Android Studio and start a new project. Choose the "Empty Activity" template.
2. **Add Dependencies**: Open the build.gradle file and add the necessary dependencies for Google Maps and Android Lifecycle components.

kotlin

```
dependencies {
    implementation("androidx.core:core-ktx:1.10.1")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("com.google.android.gms:play-services-maps:18.1.0")
    implementation("com.google.android.gms:play-services-location:21.0.1")
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.1")
    implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1")
    implementation("androidx.activity:activity-ktx:1.6.0")
```

}

Updating the Android Manifest

Next, we need to declare the required permissions and configure the Google Maps API key in the AndroidManifest.xml file.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.locationapp">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <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:supportsRtl="true"
        android:theme="@style/Theme.LocationApp">

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="YOUR_GOOGLE_MAPS_API_KEY" />

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
    </activity>

  </application>
</manifest>
```

Replace YOUR_GOOGLE_MAPS_API_KEY with your actual API key from Google Cloud.

Building the Location ViewModel

A good practice in Android development is to separate your logic into different layers. We'll use a LocationViewModel to manage location data in a lifecycle-aware way. The LocationViewModel will handle permission checks, location updates, and expose this data to the UI.

```
package com.example.locationapp

import android.Manifest
import android.app.Application
import android.content.Context
import android.content.pm.PackageManager
import android.location.Location
import android.location.LocationListener
import android.location.LocationManager
import androidx.core.content.ContextCompat
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import kotlinx.coroutines.*

class LocationViewModel(application: Application) : AndroidViewModel(application) {

    private val _locationData = MutableLiveData<Location>()
    val locationData: LiveData<Location> get() = _locationData
```

```
    private val locationManager = application.getSystemService(Context.LOCATION_SERVICE) as
LocationManager
    private lateinit var locationListener: LocationListener

    private val viewModelJob = Job()
    private val viewModelScope = CoroutineScope(Dispatchers.Main + viewModelJob)

    fun startLocationUpdates() {
        locationListener = object : LocationListener {
            override fun onLocationChanged(location: Location) {
                _locationData.value = location
            }

            override fun onProviderEnabled(provider: String) {}
            override fun onProviderDisabled(provider: String) {}
        }

        val hasFineLocationPermission = ContextCompat.checkSelfPermission(
            getApplication(),
            Manifest.permission.ACCESS_FINE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED

        if (hasFineLocationPermission) {
            locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                0L,
                0f,
                locationListener
```

```
            )
        }

        viewModelScope.launch {
            while (true) {
                delay(10 * 60 * 1000L) // Update every 10 minutes
                locationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, locationListener, null)
            }
        }
    }

    fun stopLocationUpdates() {
        locationManager.removeUpdates(locationListener)
    }

    override fun onCleared() {
        super.onCleared()
        stopLocationUpdates()
        viewModelJob.cancel()
    }
}
```

Designing the User Interface

In the activity_main.xml layout file, we'll design a simple UI with two TextView elements for latitude and longitude, a button to show the map, and a FragmentContainerView to host the Google Map.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView
    android:id="@+id/latitudeTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/latitude_label"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"
    android:visibility="gone"/>

<TextView
    android:id="@+id/longitudeTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/longitude_label"
    app:layout_constraintTop_toBottomOf="@id/latitudeTextView"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"
    android:visibility="gone"/>

<Button
    android:id="@+id/showMapButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="Show My Location"
    app:layout_constraintTop_toBottomOf="@id/longitudeTextView"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"/>

<TextView
    android:id="@+id/locationStatusTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Location services are not enabled or permissions are not granted."
    app:layout_constraintTop_toBottomOf="@id/showMapButton"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"
    android:visibility="gone"/>

<androidx.fragment.app.FragmentContainerView
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:layout_constraintTop_toBottomOf="@id/locationStatusTextView"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="16dp"
    android:layout_marginStart="16dp"
```

```
    android:layout_marginEnd="16dp"
    android:visibility="gone"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Implementing the Main Activity

Now, let's bring everything together in the MainActivity. This class will manage permissions, observe location data from the LocationViewModel, and update the UI accordingly.

```
package com.example.locationapp

import android.Manifest
import android.content.pm.PackageManager
import android.location.Location
import android.os.Bundle
import androidx.activity.result.contract.ActivityResultContracts
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.lifecycle.Observer
import com.example.locationapp.databinding.ActivityMainBinding
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions

class MainActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var binding: ActivityMainBinding
    private val locationViewModel: LocationViewModel by viewModels()
```

```
private lateinit var googleMap: GoogleMap

private val requestPermissionLauncher = registerForActivityResult(
    ActivityResultContracts.RequestMultiplePermissions()
) { permissions ->
    if (permissions[Manifest.permission.ACCESS_FINE_LOCATION] == true ||
        permissions[Manifest.permission.ACCESS_COARSE_LOCATION] == true) {
        locationViewModel.startLocationUpdates()
        binding.locationStatusTextView.visibility = android.view.View.GONE
        binding.latitudeTextView.visibility = android.view.View.VISIBLE
        binding.longitudeTextView.visibility = android.view.View.VISIBLE
        binding.showMapButton.visibility = android.view.View.VISIBLE
    } else {
        binding.locationStatusTextView.visibility = android.view.View.VISIBLE
        binding.latitudeTextView.visibility = android.view.View.GONE
        binding.longitudeTextView.visibility = android.view.View.GONE
        binding.showMapButton.visibility = android.view.View.GONE
    }
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    val mapFragment = supportFragmentManager.findFragmentById(R.id.map) as SupportMapFragment
    mapFragment.getMapAsync(this)

    if (allPermissionsGranted()) {
```

```
            locationViewModel.startLocationUpdates()
            binding.locationStatusTextView.visibility = android.view.View.GONE
            binding.latitudeTextView.visibility = android.view.View.VISIBLE
            binding.longitudeTextView.visibility = android.view.View.VISIBLE
            binding.showMapButton.visibility = android.view.View.VISIBLE
        } else {
            requestPermissions()
        }

        binding.showMapButton.setOnClickListener {
            if (allPermissionsGranted() && locationViewModel.locationData.value != null) {
                binding.map.visibility = android.view.View.VISIBLE
            } else {
                binding.locationStatusTextView.text = "Unable to display map. Location services or permissions
are disabled."
                binding.locationStatusTextView.visibility = android.view.View.VISIBLE
            }
        }

        locationViewModel.latitude.observe(this, Observer { latitude ->
            binding.latitudeTextView.text = getString(R.string.latitude_text, latitude.toString())
        })

        locationViewModel.longitude.observe(this, Observer { longitude ->
            binding.longitudeTextView.text = getString(R.string.longitude_text, longitude.toString())
        })

        locationViewModel.locationData.observe(this, Observer { location ->
            location?.let {
```

```
            if (::googleMap.isInitialized) {
                updateMap(it)
            }
        }
    })
}

private fun updateMap(location: Location) {
    val latLng = LatLng(location.latitude, location.longitude)
    googleMap.addMarker(MarkerOptions().position(latLng).title("Current Location"))
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15f))
}

override fun onMapReady(map: GoogleMap) {
    googleMap = map
}

override fun onResume() {
    super.onResume()
    if (allPermissionsGranted()) {
        locationViewModel.startLocationUpdates()
    }
}

override fun onPause() {
    super.onPause()
    locationViewModel.stopLocationUpdates()
}
```

```
    override fun onDestroy() {
        super.onDestroy()
        locationViewModel.stopLocationUpdates()
    }

    private fun allPermissionsGranted(): Boolean {
        return ContextCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_FINE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED ||
                ContextCompat.checkSelfPermission(
                    this, Manifest.permission.ACCESS_COARSE_LOCATION
                ) == PackageManager.PERMISSION_GRANTED
    }

    private fun requestPermissions() {
        requestPermissionLauncher.launch(
            arrayOf(
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_COARSE_LOCATION
            )
        )
    }
}
```

Key Points to Remember

1. **Permissions**: Always check and request the necessary permissions at runtime. Location access requires both ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION permissions.
2. **Lifecycle Management**: Use ViewModel to handle location updates efficiently and ensure that location updates are paused or stopped when the activity is not in focus.
3. **UI Updates**: Make sure to update the UI based on the location data and user interactions.

Conclusion

By following this guide, you've built an Android app that can effectively request location permissions, retrieve the user's current location, and display it on a Google Map. This foundational knowledge opens the door to creating more advanced location-based features in your apps, such as location tracking, geofencing, and more.

Feel free to explore and expand on this project by adding new functionalities or integrating additional location services. Remember, the sky's the limit when it comes to mobile development!

If you have any questions or want to share your experience, drop a comment below. Happy coding!