Rajdip Sanyal (14/09/2024)
sanyalrajdip864@gmail.com

Android Framework With Java

Activity Stack in Android

The activity stack, also known as the back stack, is a stack data structure where Android maintains a list of activities that the user has interacted with. When a new activity is started, it gets pushed onto the stack, and when the user navigates back, activities are popped off the stack.

How it Works:

Starting an Activity: When an activity is launched (using startActivity()), it's pushed onto the top of the stack. The new activity is displayed on top of the previous one.
Navigating Back: When the user presses the back button, the top activity is removed (popped) from the stack, and the previous activity becomes visible again.
Task Affinity: Activities can belong to different tasks, and each task has its own stack. A task represents a collection of activities that the user has interacted with in a particular order.

Audio Playing in Android

We can play audio or video from media files stored in our application's resources (raw resources), from standalone files in the filesystem, or from a data stream arriving over a network connection.

When developing an audio player for Android, several core concepts and theories come into play. Here's a high-level overview of the key components and theories involved:

Audio Formats: Android supports various audio formats like MP3, AAC, WAV, OGG, etc. Each format has its own encoding and decoding requirements.
Sampling Rate & Bit Depth: These parameters define the quality of the audio. Higher sampling rates and bit depths typically result in better audio quality but require more storage.

```
private void playAudio() {  1 usage
    String audioUrl = "https://file-examples.com/storage/feabe3b8ad66e50249523a7/2017/11/file_

    MediaPlayer mediaPlayer = new MediaPlayer();
    mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);

    try {

        mediaPlayer.setDataSource(audioUrl);
        mediaPlayer.prepare();
        mediaPlayer.start();

    } catch (IOException e) {
        e.printStackTrace();
    }

    Toast.makeText( context: this,  text: "Audio Started Playing....", Toast.LENGTH_SHORT).show();

}
```

Video Playing in Android

We can play video from media files stored in your application's resources (raw resources), from standalone files in the filesystem, or from a data stream arriving over a network connection.

```
public void onButtonClick(View v) {  no usages

        VideoView videoview = (VideoView) findViewById(R.id.videoview);
        Uri uri = Uri.parse( uriString: "android.resource://" + getPackageName() + "/" + R.raw.t
        videoview.setVideoURI(uri);
        videoview.start();
    }
```

Bluetooth in Android

Using Bluetooth, Android devices can create personal area networks to send and receive data with nearby Bluetooth devices. In Android 4.3 and later, the Android Bluetooth stack provides the ability to implement Bluetooth Low Energy (BLE). To fully leverage the BLE APIs, follow the Android Bluetooth HCI Requirements.

```
private void checkBluetoothPermission() {  1 usage
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        // Android 12 and above
        if (ContextCompat.checkSelfPermission( context: this, android.Manifest.permission.BLUE
                != PackageManager.PERMISSION_GRANTED) {
            // Request BLUETOOTH_CONNECT permission
            ActivityCompat.requestPermissions( activity: this,
                    new String[]{android.Manifest.permission.BLUETOOTH_CONNECT},
                    REQUEST_BLUETOOTH_PERMISSION);
        }
    }
}

@Override  14 usages
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonN
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_BLUETOOTH_PERMISSION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED)
            // Permission granted, proceed with Bluetooth operation
            Toast.makeText( context: this,  text: "Bluetooth permission granted", Toast.LENGTH_SH
        } else {
            // Permission denied
            Toast.makeText( context: this,  text: "Bluetooth permission denied", Toast.LENGTH_SHO
        }
```

Wifi in Android

Using Wifi,android allows applications to access the state of the wireless connections at a very low level. Application can access almost all the information of a wifi connection.

The information that an application can access includes the connected network's link speed,IP address, negotiation state, and other network information. Applications can also scan, add, save, terminate and initiate Wi-Fi connections.

```
private void turnWifiOn() {  no usages
    WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
    if (wifi.isWifiEnabled()) {
        Toast.makeText( context: this,  text: "Wifi is already on", Toast.LENGTH_SHORT).show();
    } else {
        wifi.setWifiEnabled(true);
        Toast.makeText( context: this,  text: "Turned on Wifi", Toast.LENGTH_SHORT).show();

    }

}
```

Call App And Sim Info in Android

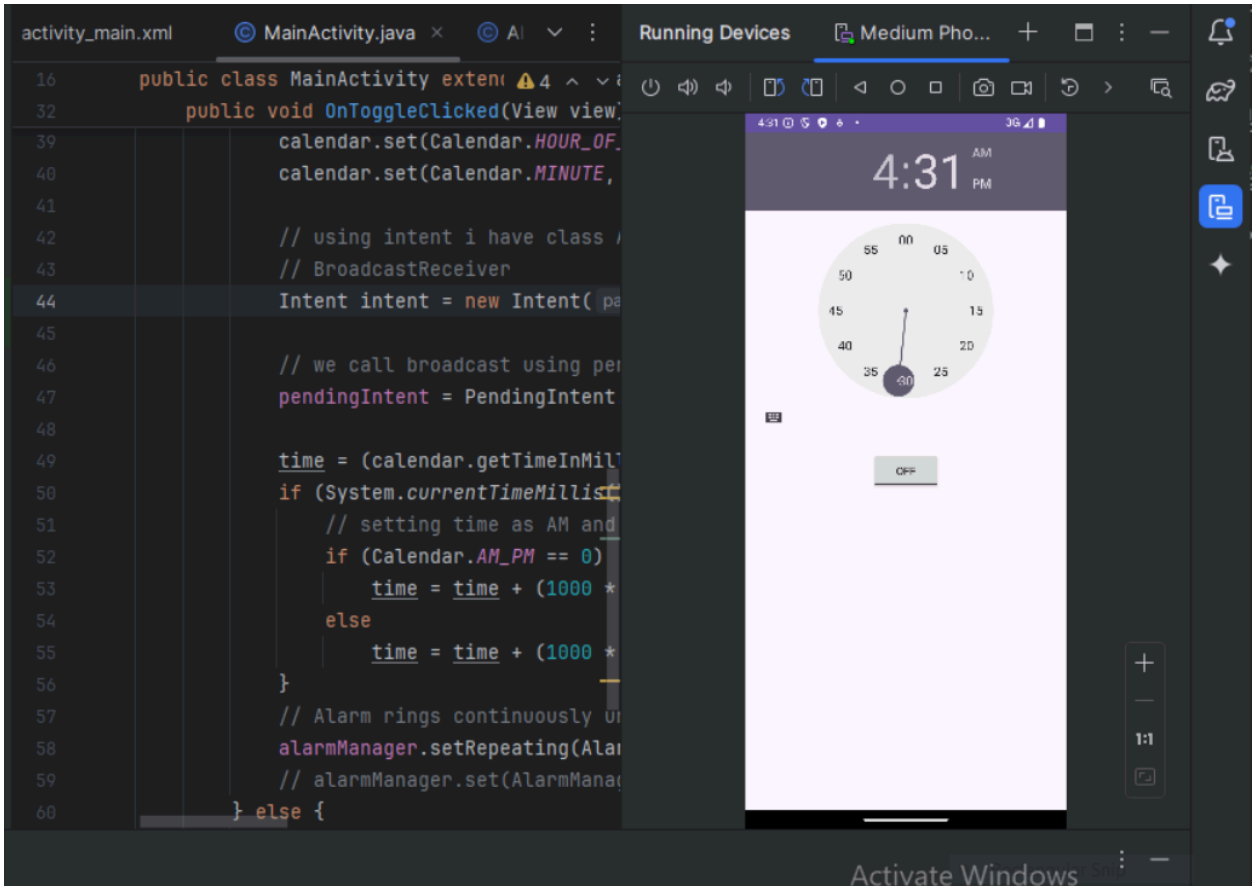
calling app allows users to receive or place audio or video calls on their device. Calling apps use their own user interface for the calls instead of using the default Phone app interface.
A SIM card, also known as a subscriber identity module, is a smart card that stores identification information that pinpoints a smartphone to a specific mobile network.

```
    Intent callintent = new Intent(Intent.ACTION_CALL);
    callintent.setData(Uri.parse( uriString: "tel:"+123456789));
    startActivity(callintent);

if (ActivityCompat.checkSelfPermission( context: this, READ_SMS) != PackageManager.PERMISSION_
{
    ActivityCompat.requestPermissions( activity: this, new String[]{READ_SMS, READ_PHONE_NUMBE
}
else
```

Alarm Manager in Android

In android development, the AlarmManager is a system service that allows us to schedule your application to run at a specific time, even if our app is not currently running. It’s useful for tasks like setting up reminders, performing periodic updates, or triggering background work.


activity_main.xml
MainActivity.java
public class MainActivity exten
public void OnToggleClicked(View view
calendar.set(Calendar.HOUR_OF_
calendar.set(Calendar.MINUTE,
// using intent i have class
// BroadcastReceiver
Intent intent = new Intent(
// we call broadcast using per
pendingIntent = PendingIntent
time = (calendar.getTimeInMil
if (System.currentTimeMillis
// setting time as AM and
if (Calendar.AM_PM == 0)
time = time + (1000 *
else
time = time + (1000 *
}
// Alarm rings continuously u
alarmManager.setRepeating(Ala
// alarmManager.set(AlarmMana
} else {
Running Devices
Medium Pho...
4:31
AM
PM
OFF
1:1
Activate Windows