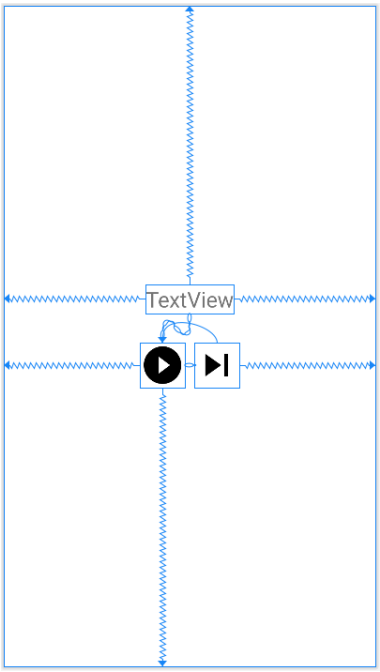
IT2010 – Mobile Application Development
BSc (Hons) in Information Technology
2nd Year
Faculty of Computing
SLIIT
2023 - Tutorial

Android Services

Android Services are a component of the Android operating system that allow apps to run tasks in the background without needing a user interface. They are useful for running long-running tasks, such as playing music or syncing data, and can be started and stopped by other components of an app or by the system itself.

Creating a Music Player App using Android Services

In this Tutorial we will create an app that can play music in the background.



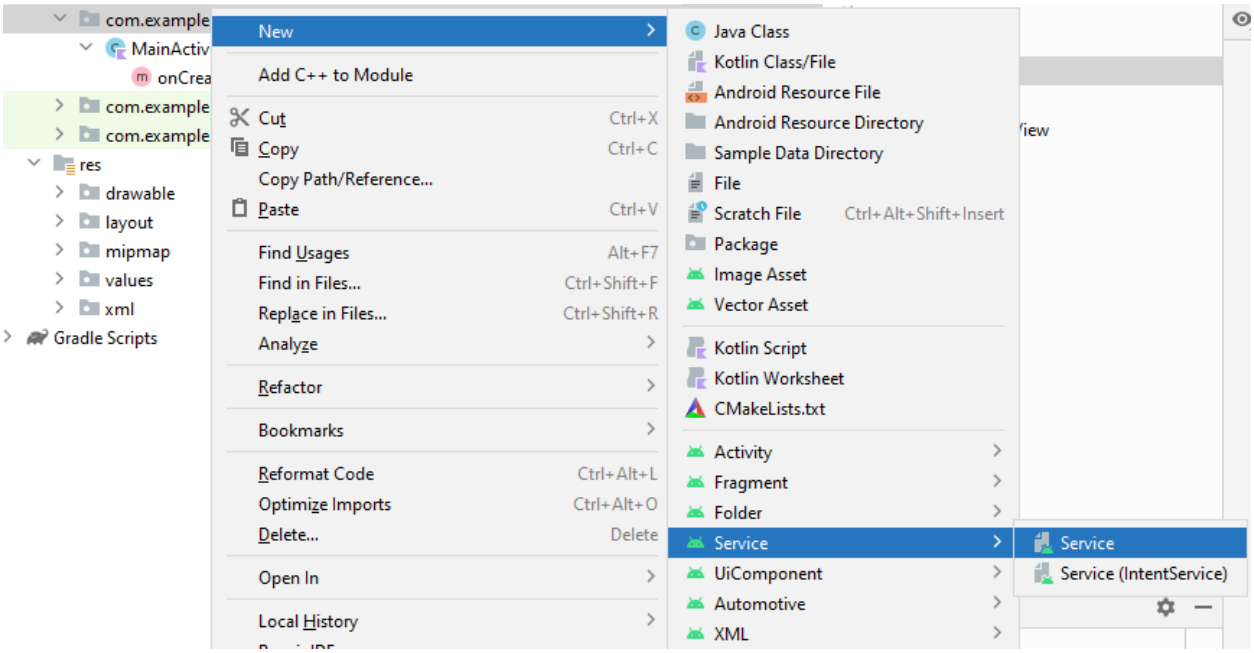
UI components

Import the following vectors to create buttons: round play, round pause, skip

1. TextView – tvMusicTitle
2. ImageView – btnPlayPause
3. ImageView – btnSkip

Service Class

1. Create a service named MusicPlayerService.

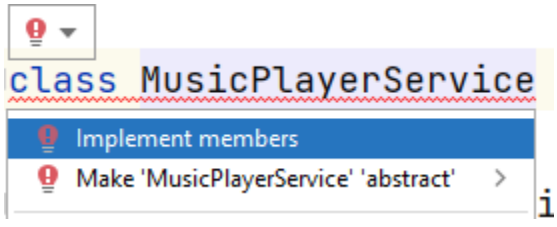


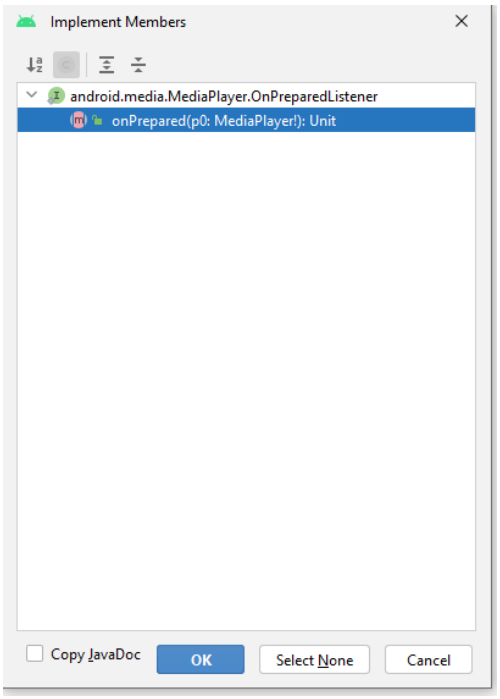
2. Modify the code by implementing MediaPlayer.OnPreparedListener

```
class MusicPlayerService : Service(),
MediaPlayer.OnPreparedListener {

    override fun onBind(intent: Intent): IBinder {
        TODO("Return the communication channel to the
service.")
    }
}
```

3. Implement the members

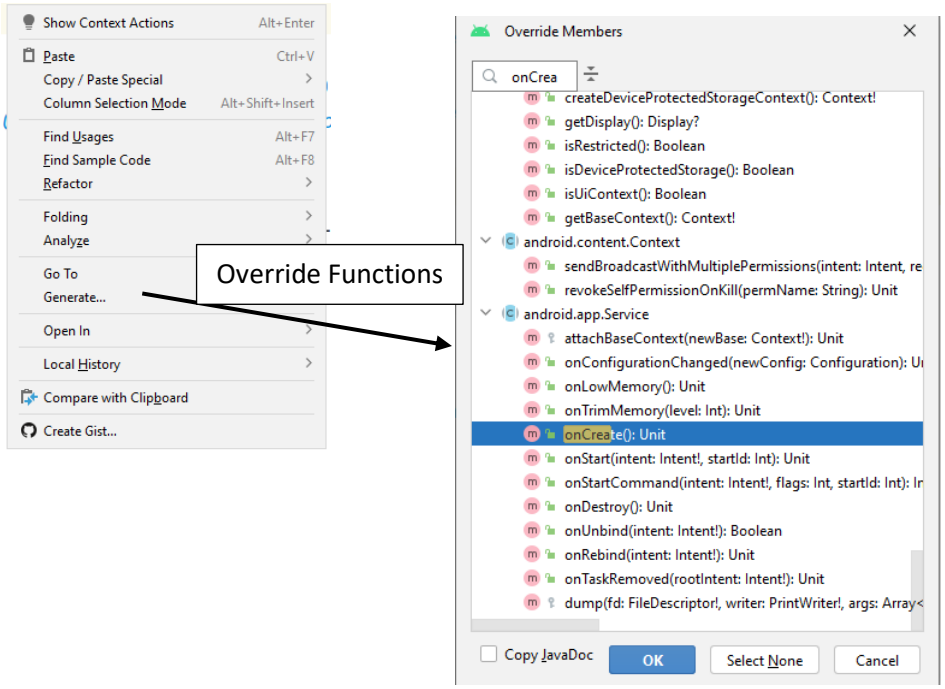




4. define the following variables

```
private lateinit var mediaPlayer: MediaPlayer
private var currentTrack = 0
private lateinit var trackList: List<Int>
private var isPaused = false
```

5. Implement onCreate method for the service



6. Modify it as follows

```
override fun onCreate() {
        super.onCreate()
        mediaPlayer = MediaPlayer()
        mediaPlayer.setOnPreparedListener(this)
        trackList = listOf(R.raw.track1, R.raw.track2 ,R.raw.track3)
    }
```

7. To call the tracks from the raw directory, first you need to create a directory under the res directory.
8. Download 3 music tracks from the following website and rename them as (track1, track2, and track3)

https://www.bensound.com/free-music-for-videos

9. Copy the music tracks into the
10. Implement the following functions

```
private fun playTrack(trackIndex:Int){
        val uri =
Uri.parse("android.resource://$packageName/${trackList[trackIndex]}")
        nowPlaying = "Now Playing: Track: Track ${trackIndex +1}"
        if (isPaused){
            mediaPlayer.start()
            isPaused = false
        }else{
            try {
                mediaPlayer.reset()
                mediaPlayer = MediaPlayer.create(this,uri)
                mediaPlayer.setOnPreparedListener(this)
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
```

```
fun play(){
        playTrack(currentTrack)
}
```

```
fun pauseTrack(){
   mediaPlayer.pause()
   isPaused = true
}
```

```
    fun skipTrack() {
```

```
        currentTrack = (currentTrack + 1) % trackList.size
        playTrack(currentTrack)
    }
```

```
fun stopTrack() {
   mediaPlayer.stop()
   mediaPlayer.release()
   mediaPlayer = MediaPlayer()
}
```

11. Modify the onBind() method as follows

```
override fun onBind(intent: Intent): IBinder? {
    return null
}
```

12. Modify the onPreparedMethd as follows

```
override fun onPrepared(mp: MediaPlayer?) {
    mp?.start()
}
```

13. Add the onDestroy () Method

```
override fun onDestroy() {
    super.onDestroy()
    mediaPlayer.release()
}
```

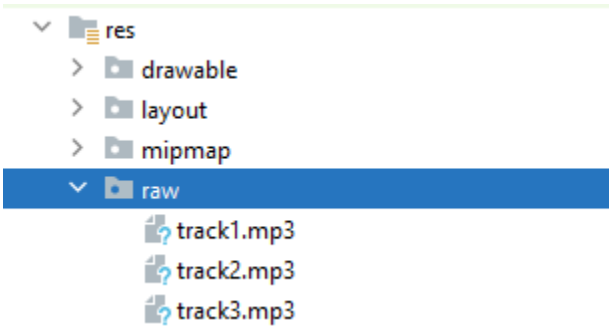
14. Add the onStartCommand below the onCreate method and modify it as follows.

```
override fun onStartCommand(intent: Intent?, flags: Int,
startId: Int): Int {
    intent?.action?.let { action ->
        when (action){
            "ACTION_PLAY" -> playTrack(currentTrack)
            "ACTION_PAUSE" -> pauseTrack()
            "ACTION_SKIP" -> skipTrack()
            "ACTION_STOP" -> {
                stopTrack()
                stopSelf()
            }
        }
    }

    return START_NOT_STICKY;
}
```

Activity

1. Download three tracks from following and rename them: track1, track2, track3.
2. Add a new directory named ‘raw’ under the res folder.
3. Copy the downloaded tracks into the raw directory.



4. Implement the UI components in the MainActivity

```
class MainActivity : AppCompatActivity() {
   private lateinit var playPauseButton:ImageView
   private lateinit var skipButton: ImageView
   private lateinit var tvMusicTitle:TextView
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)

      playPauseButton = findViewById(R.id.btnPlayPause)
      skipButton = findViewById(R.id.btnSkip)
      tvMusicTitle = findViewById(R.id.tvMusicTitle)
   }
}
```

5. Add the following states at the top level

```
private var isPlaying = false
private var isBound = false
private lateinit var musicPlayerService: MusicPlayerService
```

6. Implement the service connection as follows

```
private val serviceConnection = object: ServiceConnection {
      override fun onServiceConnected(name: ComponentName?, service: IBinder?) {
         val binder = service as MusicPlayerService.LocalBinder
         musicPlayerService = binder.getService()
         isBound = true
      }

      override fun onServiceDisconnected(name: ComponentName?) {
         isBound = false
      }
   }
```

7. Now there will be an error in the 6th step. To fix that go back to the service and implement following

```
inner class LocalBinder:Binder() {
    fun getService():MusicPlayerService = this@MusicPlayerService

}
```

8. Modify the onBind method as follows

```
override fun onBind(intent: Intent): IBinder? {
    return LocalBinder()
}
```

9. Go back to the MainActivity and implement the following

```
    playPauseButton.setOnClickListener {
        if (!isPlaying){
            musicPlayerService.play()
            tvMusicTitle.text = musicPlayerService.nowPlaying
            isPlaying = true
        }else{
            musicPlayerService.pauseTrack()
            isPlaying = false
        }
    }

    skipButton.setOnClickListener {
        musicPlayerService.skipTrack()
        tvMusicTitle.text = musicPlayerService.nowPlaying
    }
```

10. Implement the onStart and onStop lifecycle methods for the Activity

```
override fun onStart() {
    super.onStart()
    val intent = Intent(this, MusicPlayerService::class.java)
    bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE)
    Log.d("MainActivity","Service Started")
  }

  override fun onStop() {
    super.onStop()
    if (isBound) {
      unbindService(serviceConnection)
      isBound = false
    }
  }
```

11. Run the app
12. Test all the scenarios
13. Now open another app and observe what happens
14. Comment the unbindService(serviceConnection) in onStop method and observe what happens when you open another app.

Exercise

Modify the code to change the play icon to pause and vise versa when playing and pausing the music.