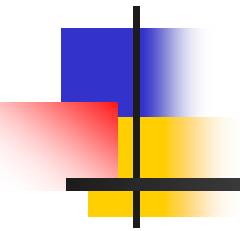
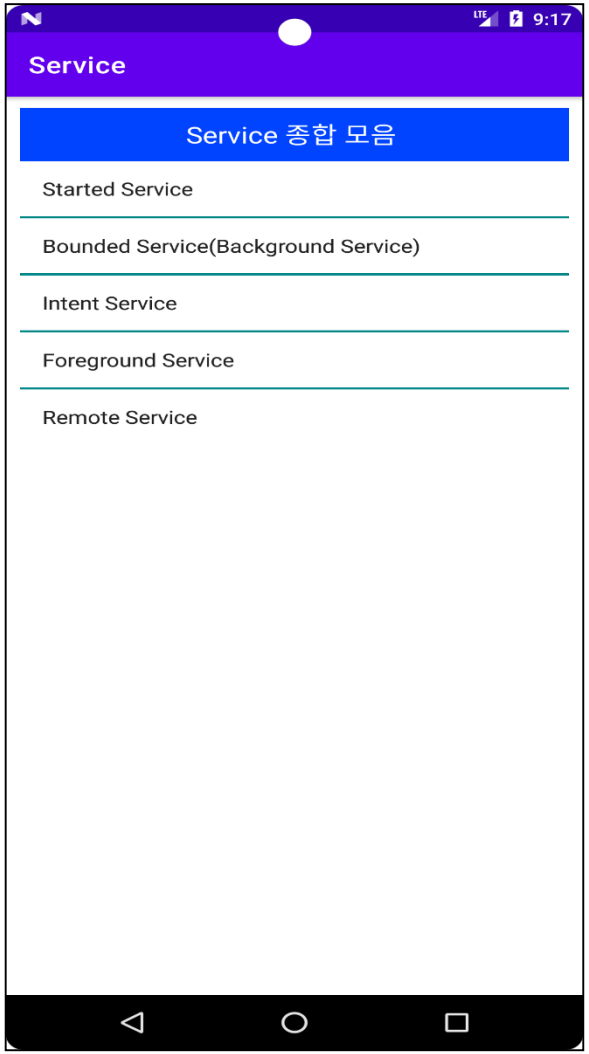


Android Service

배 희호 교수
경복대학교
소프트웨어융합과

Application Service 예제

- 다음과 같은 Service를 만들어보자



Application Service 예제

- MyApplication.JAVA

```
public class MyApplication extends Application {
    private boolean serviced = false;
    private int progress = 0;
    private int maxProgress = 0;
    private Button button;

    public boolean isServiced() {
        return serviced;
    }

    public void setServiced(boolean serviced) {
        this.serviced = serviced;
    }

    public int getProgress() {
        return progress;
    }
```

Application Service 예제

- MyApplication.JAVA

```
public void setProgress(int progress) {
    this.progress = progress;
}

public int getMaxProgress() {
    return maxProgress;
}

public void setMaxProgress(int maxProgress) {
    this.maxProgress = maxProgress;
}

public Button getButton() {
    return button;
}

public void setButton(Button button) {
    this.button = button;
}
```

Application Service 예제

- MyApplication.JAVA

```
public String getTime(int time) {
    int second = time / 1000;
    int minute = second / 60;
    second %= 60;
    return String.format(Locale.KOREA, "%02d:%02d", minute, second);
}

public NotificationCompat.Builder makeNoti() {
    Intent notiIntent = new Intent(this, ForegroundService.class);
    notiIntent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, notiIntent,
                                        PendingIntent.FLAG_IMMUTABLE);
```

Application Service 예제

- MyApplication.JAVA

```
        NotificationCompat.Builder builder =
                                new NotificationCompat.Builder(this, "default")
                .setContentTitle("Music Player")
                .setContentText("음악이 재생중입니다.")
                .setSmallIcon(R.drawable.ic_launcher)
                .setContentIntent(pendingIntent);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel("default",
                            "기본 채널", NotificationManager.IMPORTANCE_LOW);
            NotificationManager manager = (NotificationManager)
                                            getSystemService(NOTIFICATION_SERVICE);
            manager.createNotificationChannel(channel);
        }
        return builder;
    }
}
```

Application Service 예제

- Application 등록

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

    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />

    <application
        android:name=".MyApplication"
        android:allowBackup="true"
```

Application Service 예제

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#0044FF"
        android:gravity="center"
        android:padding="10dp"
        android:text="Service 종합 모음 "
        android:textColor="@color/white"
        android:textSize="20sp" />
```

Application Service 예제

- 사용자 인터페이스

```
    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="@color/teal_700"
        android:dividerHeight="2dp" />
</LinearLayout>
```

Application Service 예제

- MainActivity.JAVA

```
public class MainActivity extends AppCompatActivity
                          implements AdapterView.OnItemClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ArrayList<String> serviceList = new ArrayList<>();
        serviceList.add("Started Service");
        serviceList.add("Bounded Service(Background Service)");
        serviceList.add("Intent Service");
        serviceList.add("Foreground Service");
        serviceList.add("Remote Service");

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, serviceList);
```

Application Service 예제

- MainActivity.JAVA

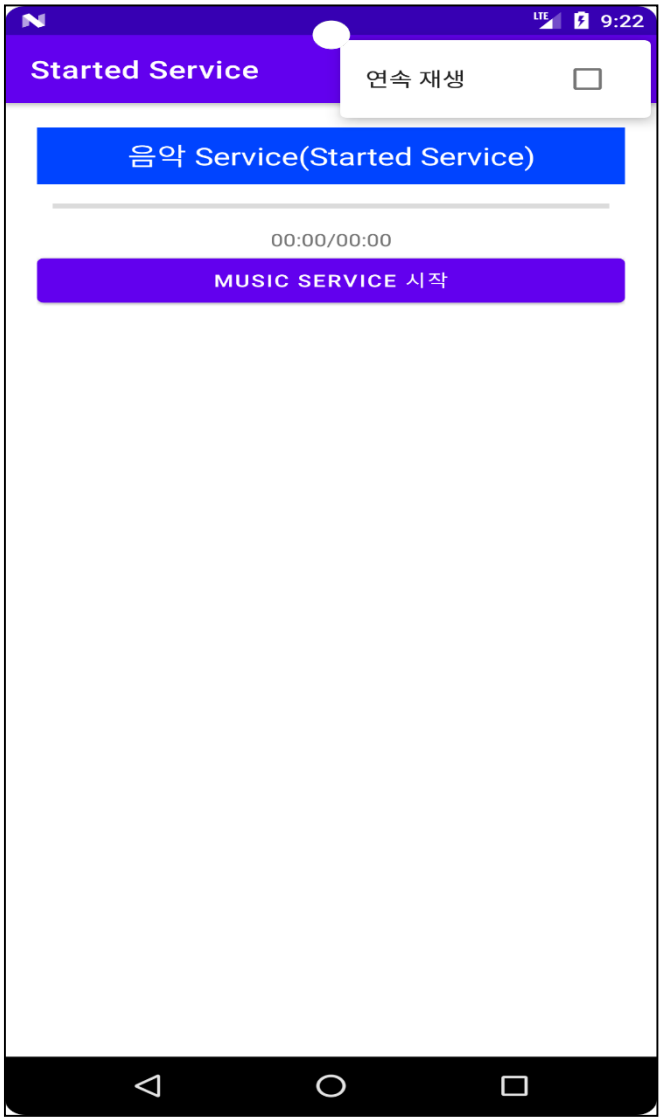
```
    ListView listView = findViewById(android.R.id.list);
    listView.setAdapter(adapter);
    listView.setOnItemClickListener(this);
}

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    Intent intent = null;
    switch (position) {
        case 0:
            intent = new Intent(this, StartedService.class);
            break;
        case 1:
            intent = new Intent(this, BoundedService.class);
            break;
```

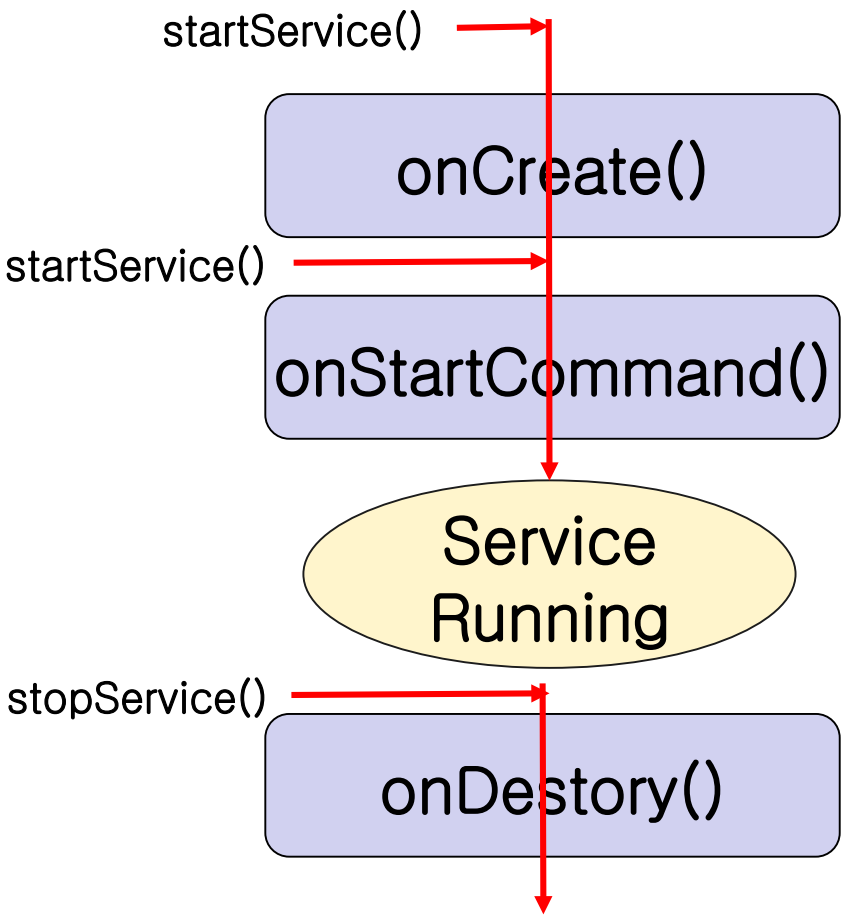
Application Service 예제

- MainActivity.JAVA

```
            case 2:
                intent = new Intent(this, MyIntentService.class);
                break;
            case 3:
                intent = new Intent(this, ForegroundService.class);
                break;
            case 4 :
                intent = new Intent(this, RemoteService.class);
        }
        startActivity(intent);
    }
}
```

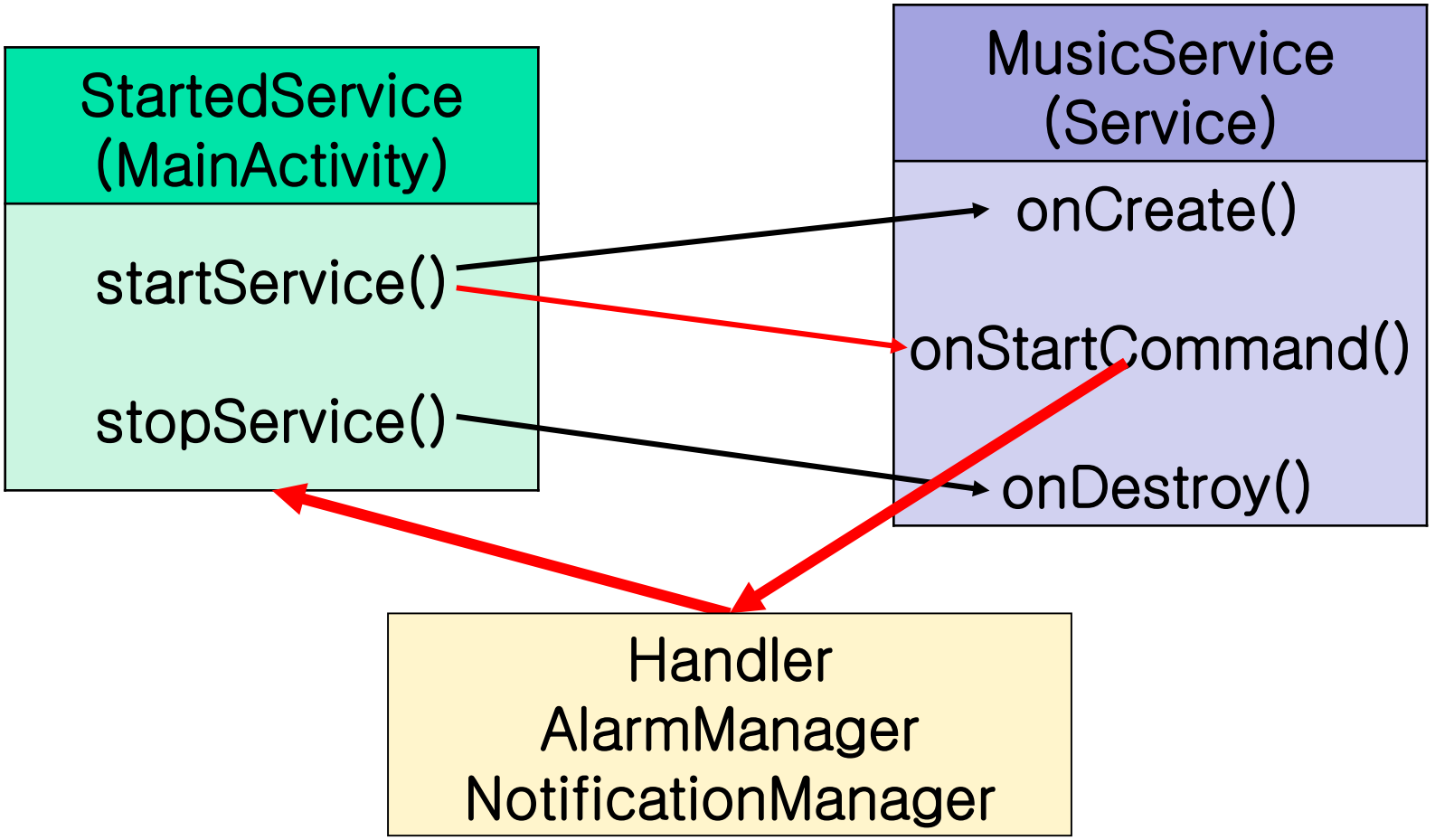
Started Service





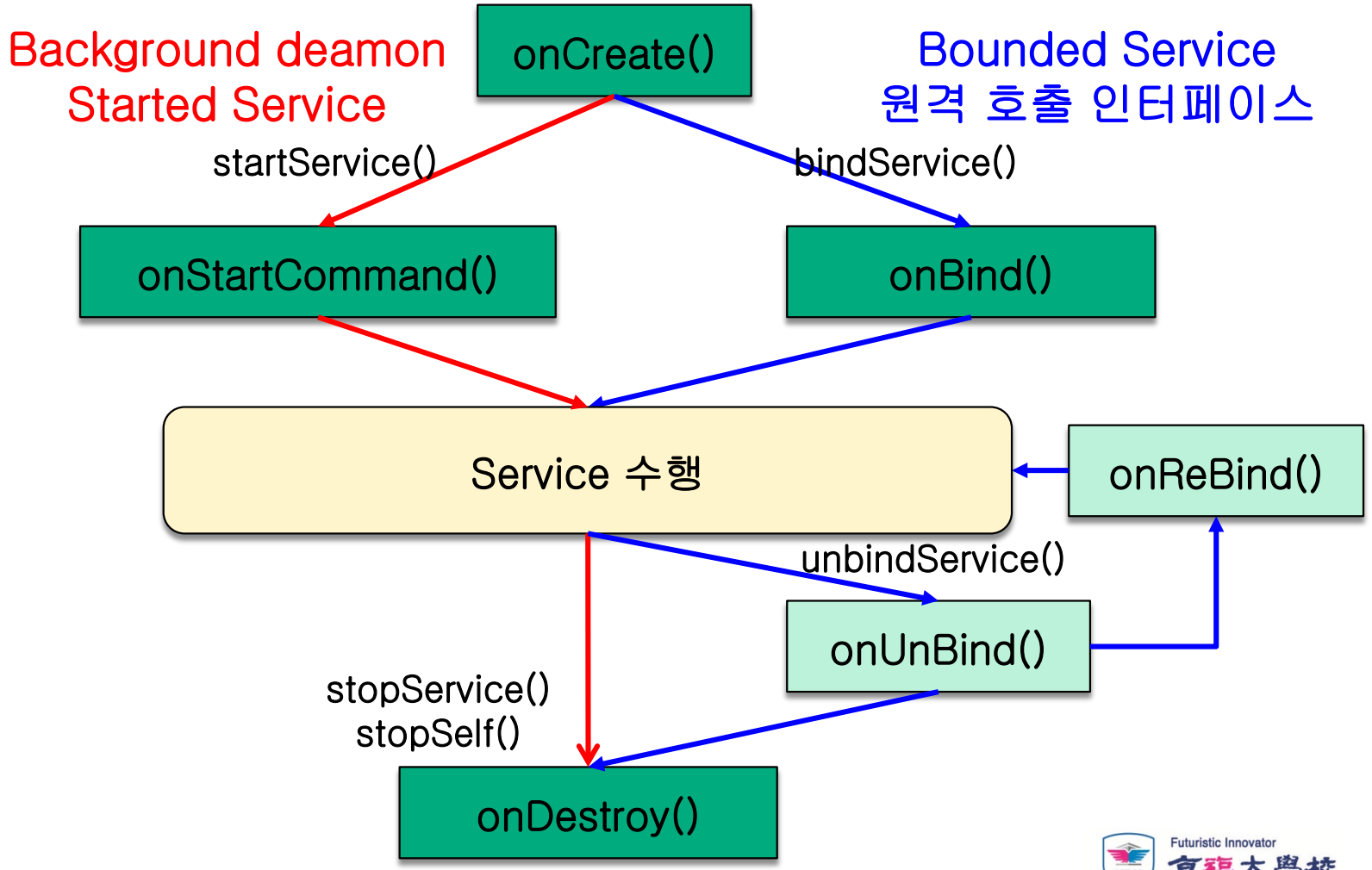
Started Service

- Started Service 생성 및 제어

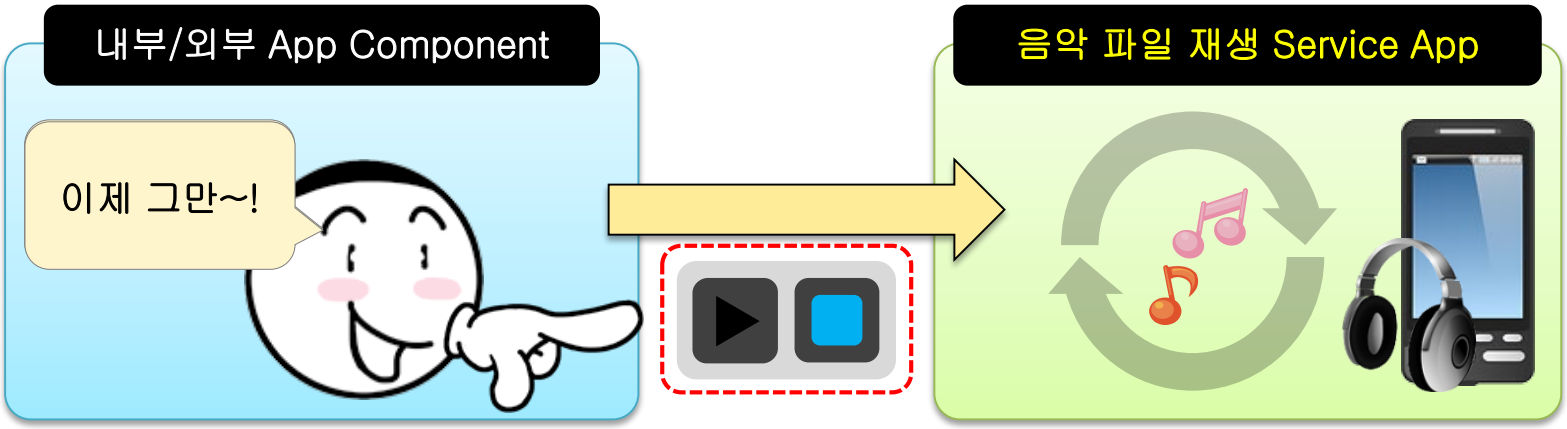


Started Service

- Started Service와 Bounded Service



Started Service



- 내/외부 App과 음악 File 재생 Service의 관계는 서로 독립적
  - Service를 동작 시킨 Component가 종료되더라도 Service는 작업이 끝날 때까지 동작을 유지함

Started Service

- 특정 Service를 Background로 동작시키는 것에 목적을 둔 Service 형태
- Activity와 같은 Component는 startService() 메소드를 호출해서 Service를 시작 시킬 수 있으며, 사용하고자 하는 Service 정보와 Service에서 사용될 Data를 Intent에 담아 전달할 수 있음
- 발생시키는 곳과 Service는 독립적이므로 Activity가 종료되어도 동작함
  - Service를 발생시킨 곳에서는 중단시키는 것 이외에는 어떤 제어도 불가능
  - Service는 자신의 작업을 다 마무리 한 뒤, stopSelf() 메소드를 호출하여 종료될 수도 있고 또는 다른 Component에서 stopService()를 호출해서 종료 시킬 수도 있음

Started Service

- Service가 실행되면 onCreate() -> onStartCommand() Lifecycle 메소드가 호출되고, 종료될 땐 onDestroy() 메소드가 호출 됨
  - Service가 동작 중일 때는 onStartCommand()의 Code가 계속해서 동작하는 것임

Started Service

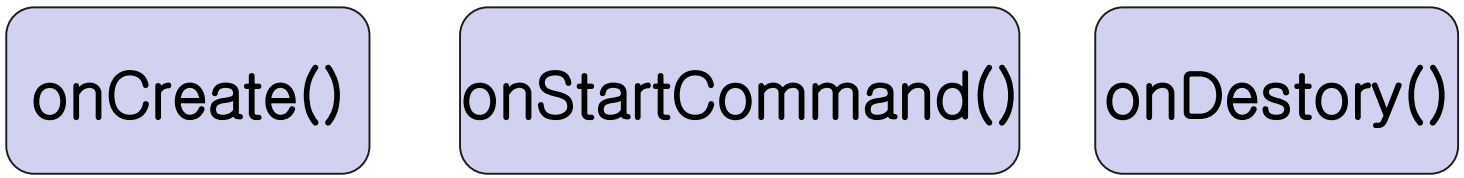
- Started Service의 주요 Lifecycle 메소드
  - void onCreate()
    - Service가 처음 생성될 때 호출
    - 한 번만 호출되며, 초기화 작업(예: Resource 준비, Thread 생성 등)을 수행
  - int onStartCommand(Intent intent, int flags, int startId)
    - startService()가 호출될 때마다 실행
    - 여기서 Intent를 다루고 작업을 수행
    - Service가 실행 중일 때도 onStartCommand()는 반복적으로 호출될 수 있음
    - Service의 작업 내용과 동작 방식을 정의하며, 반환값에 따라 Service 재 시작 동작을 제어할 수 있음
    - 일반적으로 START_STICKY를 반환

Started Service

- Started Service의 주요 Lifecycle 메소드
  - void onDestroy()
    - Service가 종료될 때 호출
      - 외부에서 stopService()를 호출하거나
      - 내부에서 stopSelf()를 호출 시 호출됨
    - Resource 정리(예: Thread 종료, Database 연결 해제 등)를 수행
  - IBinder onBind(Intent intent)
    - startService()로 실행되는 Started Service는 null을 반환

Started Service

- 이미 시작된 Service는 onStartCommand( ) 메소드의 실행을 마친 후 바로 onDestroy( )로 가는 것이 아님
- 다음 실행을 위해 대기 상태로 들어가기 때문에 Process는 계속 살아있게 됨
- 이때 외부에서 Service를 실행시키면 onStartCommand() 메소드가 호출됨
- System에 Resource가 부족하게 되면, OS가 Process를 죽이고 이때 onDestory() 메소드에 의해서 Service가 종료됨
- Service를 다시 시작하게 되면 onCreate( )가 작동하게 됨



Started Service

Started Service

- startService()를 호출함과 동시에 Intent를 전달함으로써 Activity나 다른 Component에서 Service를 시작 시킬 수 있음
- Android System에서 Service와 관련된 onStartCommand()를 호출하고 그 메소드에 Intent를 전달
  - 사용자가 직접 onStartCommand()를 호출할 수는 없음
- startService() 메소드는 호출 되자마자 반환되며, Android System에서는 Service의 onStartCommand() 메소드를 호출
- Service가 생성된 상태가 아니었다면, System은 onCreate() 메소드를 호출한 후에 onStartCommand() 메소드를 호출
- Service가 Binding 기능을 제공하지 않는다면, Service로부터 특정 결과값을 전달받는 것은 불가능

Started Service

- Binding 기능 없이 Service로부터 결과값을 전달받길 원한다면 BroadCast와 관련된 PendingIntent를 생성한 뒤, Service를 시작시키는 startService()의 인자 값으로 전달하면 됨
  - Service가 자신의 작업을 다 끝낸 다음에, 결과 Data를 전달하기 위해서 BroadCast 기능을 이용하면 됨
- Service가 시작되도록 요청은 여러 개가 존재할 수 있고, 그 결과 요청에 대응되는 onStartCommand() 메소드가 호출
- Started Service는 System에서 사용할 Memory가 부족하지 않는 한, System은 현재 작동하고 있는 Service를 절대 멈추게 하거나 종료 시키지 않음
- Service를 종료하고자 할 때는 stopSelf()나 다른 Component에서 stopService() 메소드 만을 한 번 호출함으로써 직접 Service를 종료 해야만 함

Started Service

- stopSelf()나 stopService()가 호출되었을 때, System은 Service를 종료 시킴
- 만약 Service가 onStartCommand() 메소드에서 여러 개의 요청을 동시다발적으로 처리하고 있을 때, Service에 맨 처음 들어온 요청이 끝났다고 전체 Service를 정지시키면 안됨
  - 첫 번째 요청을 받은 이후에 두 번째 요청을 받았을 수도 있기 때문임
  - 처음 들어온 요청이 끝났다고 stopSelf()나 stopService()를 호출한다면 두 번째 요청이 처리되기도 전에 Service가 종료될 것임
  - 이런 문제를 피하기 위해서, stopSelf(int)라는 메소드를 호출함으로써 자신의 작업을 끝낸 Service 요청만 종료시킬 수 있음

Started Service

- stopSelf(int)라는 메소드를 호출할 때, 해당 Service 요청과 관련된 ID값을 인자로 전달하면 됨
  - ID값은 onStartCommand()에서 전달 받을 수 있음
- 두 번째 요청이 들어온 이후 작업을 끝낸 첫 번째 요청을 stopSelf(int) 메소드로 종료해도, 두 번째 요청은 첫 번째 요청과 ID값이 다르기 때문에 종료되지 않음

Started Service

- Handler의 주요 메소드

메소드	설명
sendMessage(Message)	Message를 Queue에 추가하여 처리함
post(Runnable)	특정 Runnable 작업을 Queue에 추가하여 처리함
postDelayed(Runnable, delayMillis)	지정된 시간 이후에 Runnable 작업을 실행
removeCallbacks(Runnable)	특정 Runnable 작업을 Queue에서 제거
removeMessages(what)	특정 Message(what)를 Queue에서 제거

Started Service (공통)

- menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/item"
        android:checkable="true"
        android:title="연속 재생" />
</menu>
```

Started Service (공통)

- activity_service.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context=".StartService">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#0044FF"
        android:gravity="center"
        android:padding="10dp"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />
```

Started Service (공통)

- activity_service.xml

```
<ProgressBar
    android:id="@+id/progress"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp" />
<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="00:00/00:00" />
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Music Service 시작" />
</LinearLayout>
```

Started Service

- StartedService.JAVA

```
public class StartedService2 extends AppCompatActivity {
    private MyApplication app;
    private ProgressBar progressBar;
    private TextView textView;
    private boolean loop = false;

    Handler handler = new Handler(Looper.getMainLooper()) {
        @Override
        public void handleMessage(@NonNull Message msg) {
            if (app.isServiced()) {
                if (msg.what == 0) {
                    Bundle bundle = msg.getData();
                    int current = bundle.getInt("current");
                    int size = bundle.getInt("size");
```

Started Service

- StartedService.JAVA

```
                String temp = app.getTime(current) + "/" + app.getTime(size);
                progressBar.setMax(size);
                progressBar.setProgress(current);
                textView.setText(temp);
            }
        }
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_service);
    setTitle("Started Service");
    app = (MyApplication) getApplication();
    TextView title = findViewById(R.id.textView1);
    title.setText("음악 Service(Started Service)");
    progressBar = findViewById(R.id.progress);
    textView = findViewById(R.id.textView2);
```

Started Service

- StartedService.JAVA

```
    Intent intent = new Intent(this, MusicStartedService2.class);
    Messenger messenger = new Messenger(handler);
    app.setButton(findViewById(R.id.button));
    app.getButton().setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            if (!app.isServiced()) {
                intent.putExtra("Messenger", messenger);
                intent.putExtra("title", "헤어졌다 만났다");
                intent.putExtra("loop", loop);
                startService(intent);
            } else {
                stopService(intent);
            }
        }
    });
}
```

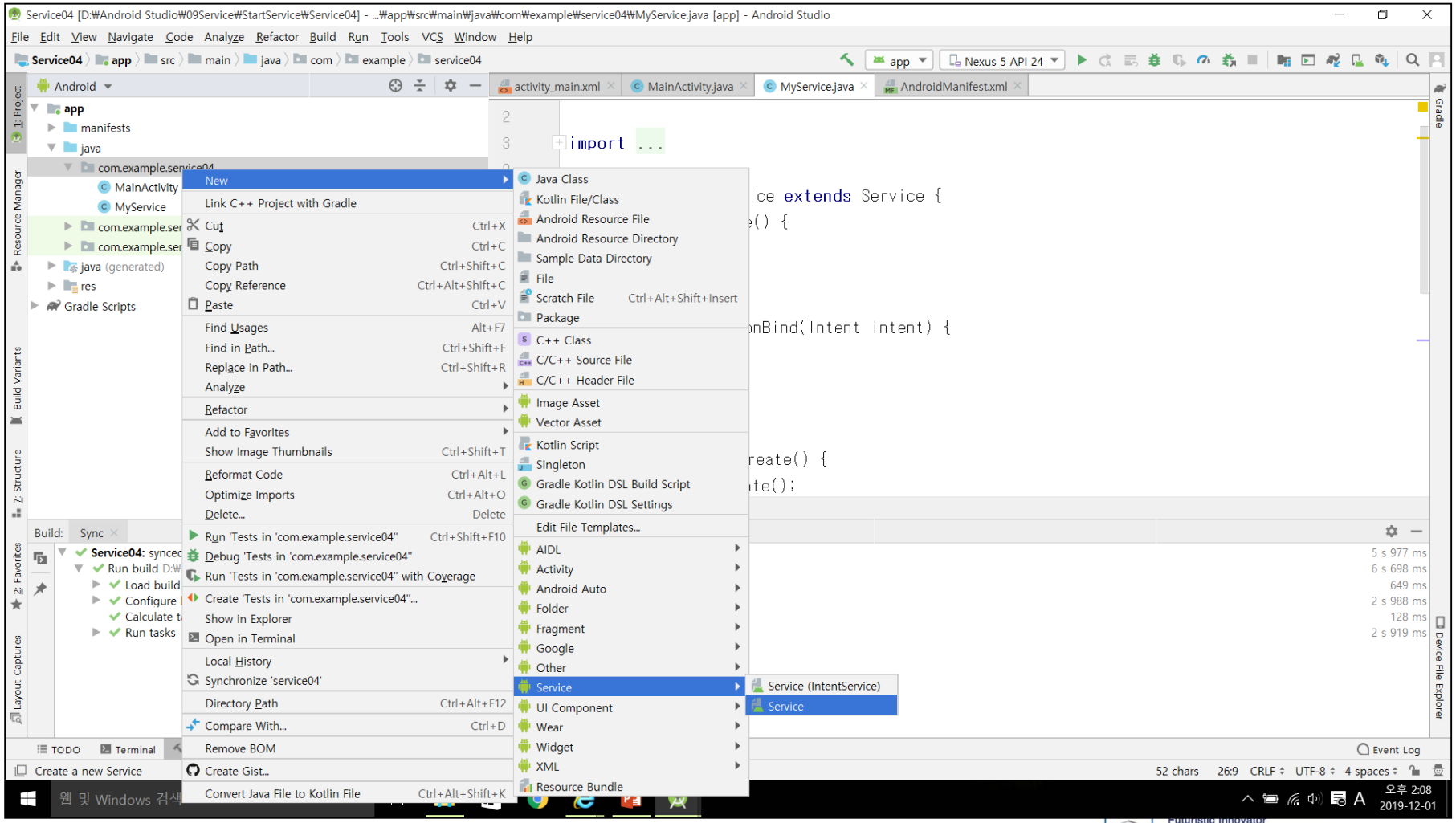
Started Service

- StartedService.JAVA

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.isChecked())
            loop = false;
        else
            loop = true;
        item.setChecked(loop);
        return true;
    }
}
```

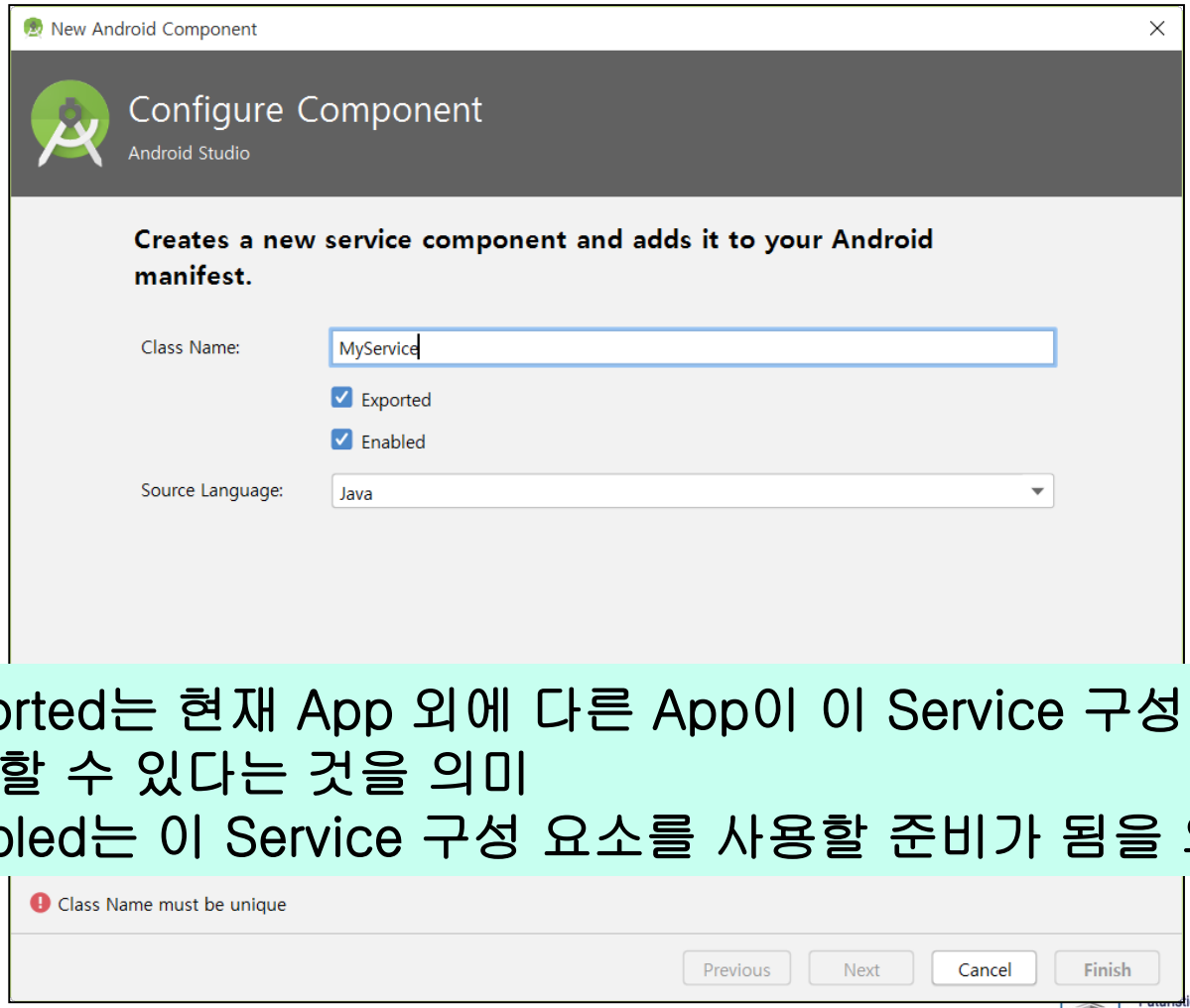
Started Service

- Service 클래스의 상속을 받는 StartedService 클래스 정의



Started Service

- Service 클래스의 상속을 받는 StartedService 클래스 정의



- ✓ Exported는 현재 App 외에 다른 App이 이 Service 구성 요소를 호출할 수 있다는 것을 의미
- ✓ Enabled는 이 Service 구성 요소를 사용할 준비가 됨을 의미

Started Service

- MusicStartedService.JAVA

```
public class MusicStartedService extends Service {
    private MediaPlayer mediaPlayer;
    //  private NotificationManager manager;
    private MyApplication app;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        app = (MyApplication) getApplication();
        // manager = (NotificationManager)
                                getSystemService(Context.NOTIFICATION_SERVICE);
    }
```

Started Service

```
public int onStartCommand(Intent intent, int flags, int startId) {
    app.setServiced(true);
    app.getButton().setText("Music Service 중지");
    Messenger messenger = intent.getParcelableExtra("Messenger");
    String title = intent.getStringExtra("title");
    Toast.makeText(this, "Music Service가 시작되었습니다.₩n 노래제목 : "
                                        +title, Toast.LENGTH_LONG).show();
    manager.notify(1, makeNotification());
    player = MediaPlayer.create(this, R.raw.davichi);
    player.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mediaPlayer) {
            stopSelf();
        }
    });
    player.start();
    MyThread2 thread = new MyThread2(player, player.getDuration(), messenger);
    thread.start();
    return START_STICKY;
}
```

Started Service

- MusicStartedService.JAVA

```
private Notification makeNotification() {
    Notification.Builder builder = new Notification.Builder(this);
    Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
                                                 R.mipmap.ic_launcher);
    builder.setContentTitle("서비스 안내")
           .setContentInfo("Content Info")
           .setContentText("음악 서비스가 시작되었습니다")
           .setSmallIcon(R.drawable.goldcoin)
           .setLargeIcon(bitmap);
    return builder.build();
}
```

Started Service

- MusicStartedService.JAVA

```
    @Override
    public void onDestroy() {
        manager.cancelAll();
        player.stop();
        app.setServiced(false);
        app.getButton().setText("Music Service 시작");
        Toast.makeText(this, "Music Service가 중지되었습니다",
                                                    Toast.LENGTH_LONG).show();
    }
}
```

Started Service (공통)

- MyThread.JAVA

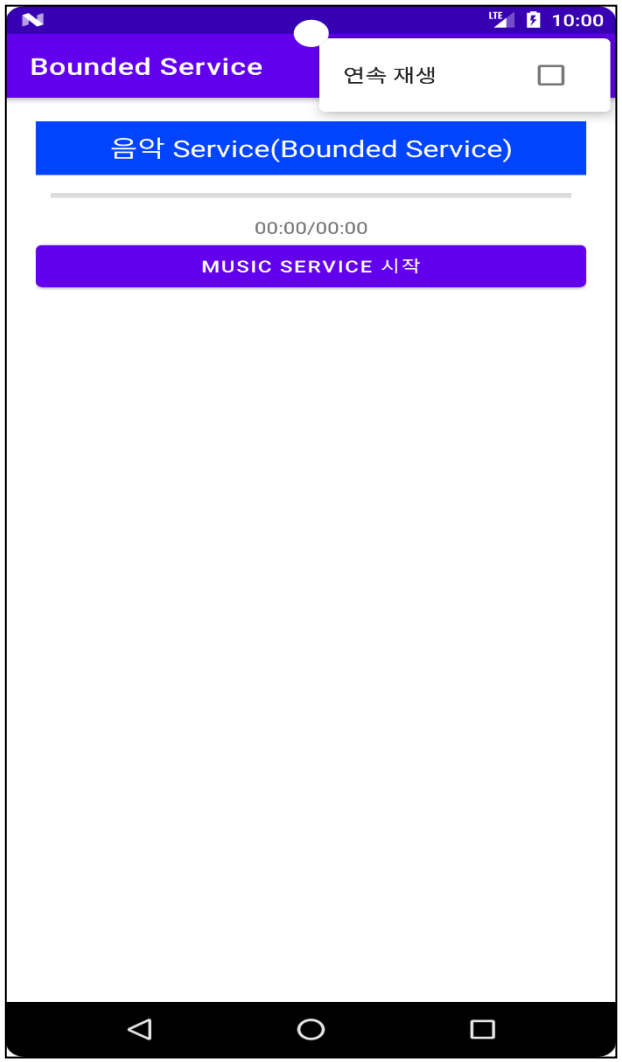
```
public class MyThread2 extends Thread {
    private MediaPlayer player;
    private int size;
    private Messenger messenger;

    public MyThread2(MediaPlayer player, int size, Messenger messenger) {
        this.player = player;
        this.size = size;
        this.messenger = messenger;
    }

    @Override
    public void run() {
        while (player.isPlaying()) {
            int current = player.getCurrentPosition();
            Message message = Message.obtain();
```

Started Service (공통)

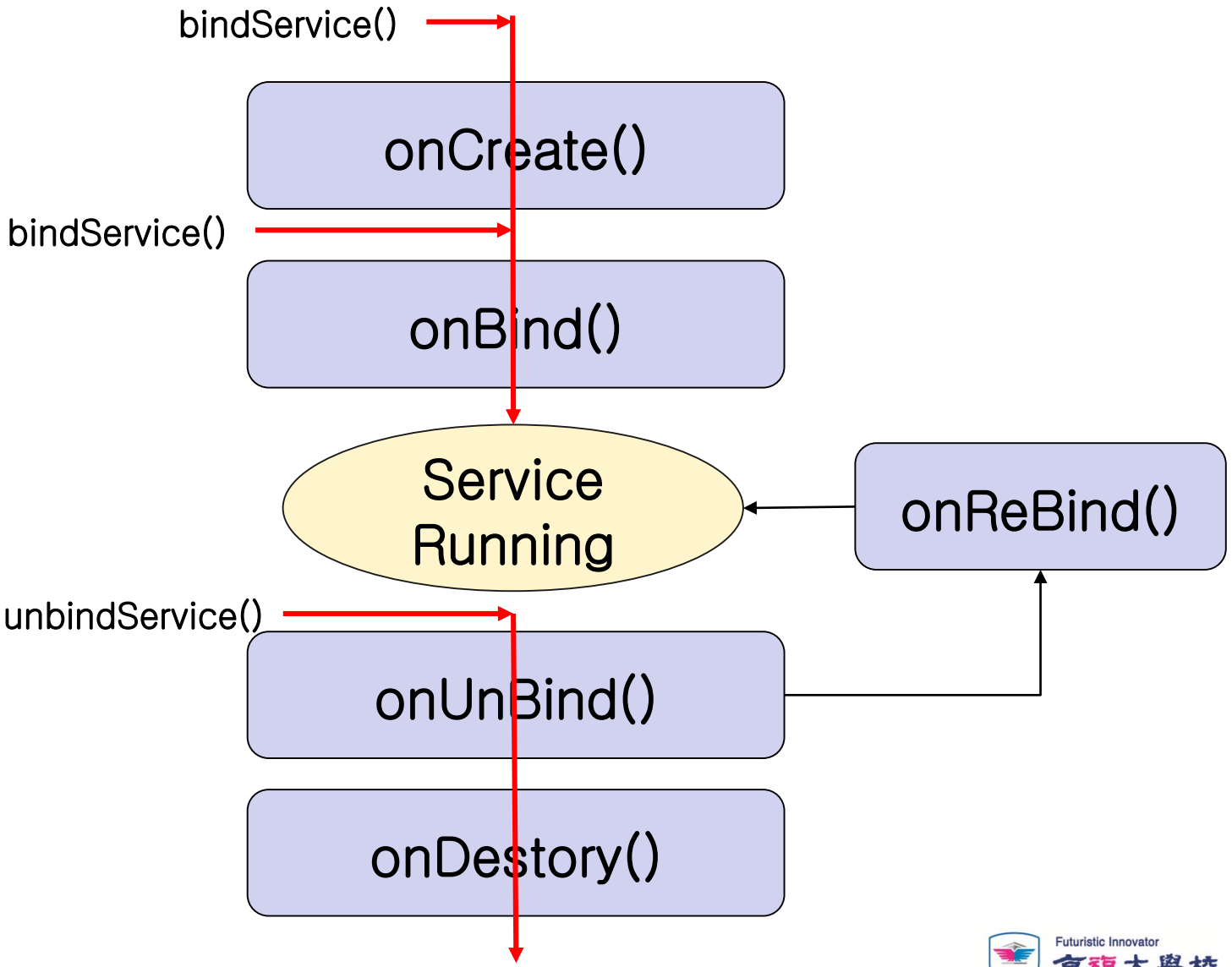
- MyThread.JAVA

```
            Bundle bundle = new Bundle();
            bundle.putInt("current", current);
            bundle.putInt("size", size);
            message.setData(bundle);
            try {
                messenger.send(message);
            } catch (RemoteException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
```

Bounded Service

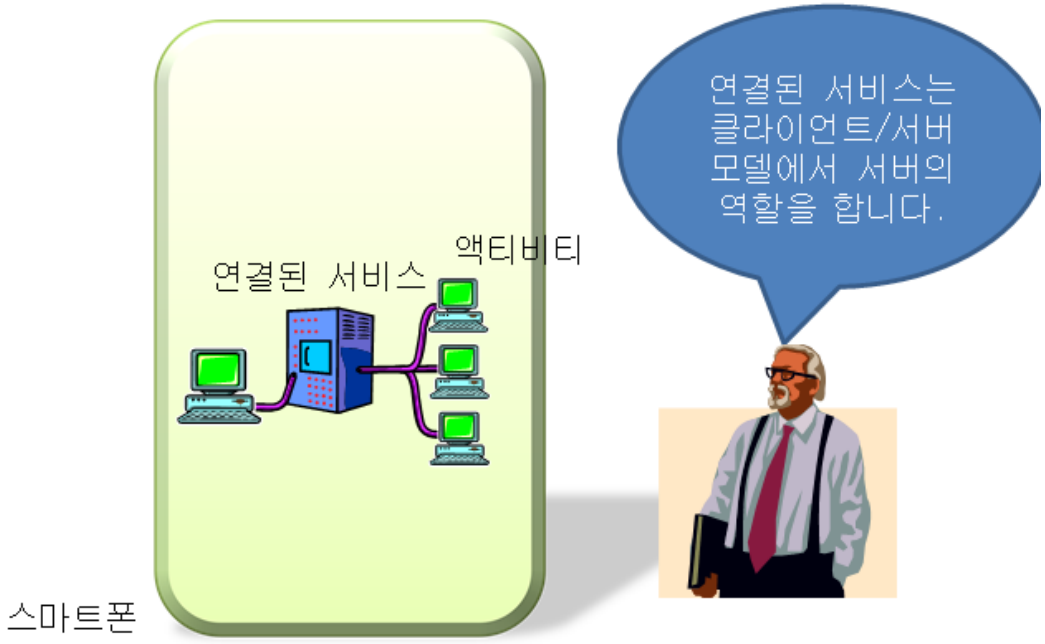


Bounded Service



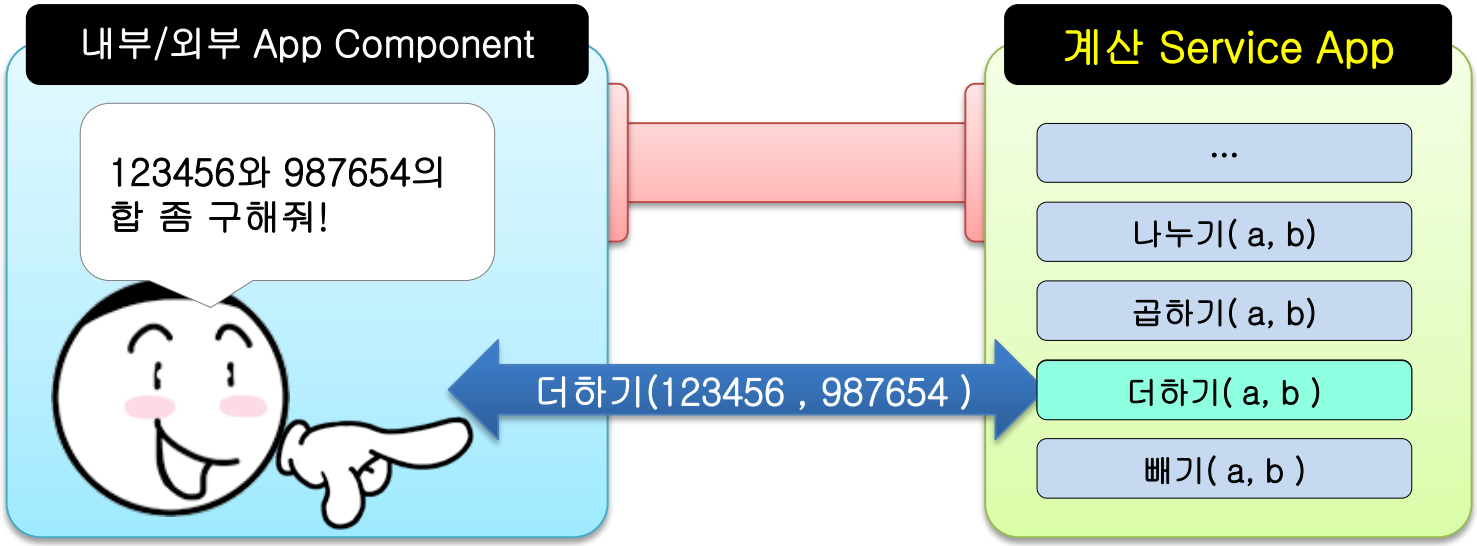
Bounded Service

- 이 Service는 마치 Client-Server와 같이 동작함
  - Service가 Server 역할을 함
  - 작업 결과를 호출자에게 전달할 수 있고 Process간 통신(IPC)를 수행할 수도 있음



Bounded Service

- Bound는 외부 Library를 사용하는 것과 매우 유사함



- Service로 연결될 Component는 Service에 존재하는 메소드들을 마치 Library를 가져다 사용하듯이 쓸 수 있음
- 내/외부 App과 계산 Service의 관계는 서로 의존적
  - Service를 요청한 Component가 종료되면 Service 연결 끊어짐. 반대로 Service는 내/외부 Component가 연결을 끊기 전까지 요청한 작업이 완료되었더라도 연결 유지

Bounded Service

- Binding된 구성 요소가 모두 Binding을 해제하면 Service는 소멸함
- Binding을 하려면 Service와의 연결을 Monitoring하는 ServiceConnection의 구현을 해야 함
  - Android System이 Client와 Service사이의 연결을 생성하는 경우, System은 onServiceConnected()를 ServiceConnection에서 호출하여 Client가 Service와 통신하는 데 사용할 수 있도록 IBinder를 전달함
- bindService() 메소드를 통해 시작되는 Service를 Service Bind 혹은 Bounded Service라고 함

- 호출 메소드 : bindService(), unbindService()
- 콜백 메소드 : onBind()
- 구현 해야하는 클래스 : ServiceConnection

Bounded Service

- void bindService(Intent service, ServiceConnection connection, int flags)
  - Activity를 특정 Service에 연결하기위한 Activity 메소드
  - bindService의 Instance
  - 매개변수
    - 호출하려는 Service를 가리키는 Intent
    - ServiceConnection Instance
    - Option 설정 Flag
      - 대부분의 경우 BIND_AUTO_CREATE 값을 넘겨줌
      - BIND_AUTO_CREATE의 경우 해당 Service가 실행 중이지 않을 경우 Service를 자동으로 시작 시킴

Bounded Service

- bindService
  - Service 받는 쪽과 상관없이 돌아가는 구조
  - 호출 순서 : onCreate() -> onBind() -> ... -> onRebind -> ... -> onUnbind() -> onDestory()
  - 호출한 객체가 존재하면 System은 Service가 필요하다고 판단하여 실행, 재실행 등으로 유지
  - 호출 객체 소멸 시 System에서 필요 없다고 판단하면 언제든지 중지

Bounded Service

- void unbindService(ServiceConnection connection)
  - Android에서 Bound Service와 클라이언트(예: Activity, Fragment) 간의 연결을 해제하는 데 사용
  - Service의 onUnbind() 메소드가 호출
  - 매개변수
    - ServiceConnection Instance

Bounded Service

- onBind() 메소드
  - onCreate()/onDestroy() 메소드 외에 구현해야 할 메소드
  - Bounded Service Biding 객체를 생성하려면 콜백 메소드인 onBind()를 구현해야 함
  - onBind()는 IBinder를 반환하는데, 바로 이 객체가 Service와 Client 사이의 Interface 역할을 함
  - Client가 bindService()를 호출하면, Client가 Service에 연결되면서 IBinder가 반환되고, Client가 IBinder를 받으면 이 Interface를 통해 주고 받는 것이 가능해지는 것
  - Service가 제공하는 다른 메소드 호출 가능
  - 여러 Client가 하나의 Service에 동시 접속이 가능
  - Client가 Service와의 접속을 마치려면 unbindService()를 호출
  - Service에 연결된 Client가 하나도 남아있지 않으면 Service를 소멸

Bounded Service

- BoundedService.JAVA

```
public class BoundedService extends AppCompatActivity {
    private MyApplication app;
    private ProgressBar progressBar;
    private TextView textView;
    private boolean loop;

    ServiceConnection conn = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName,
                                       IBinder iBinder) {
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
        }
    };
```

Bounded Service

- BoundedService.JAVA

```
Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(@NonNull Message msg) {
        if (app.isServiced()) {
            if (msg.what == 0) {
                Bundle bundle = msg.getData();
                int current = bundle.getInt("current");
                int size = bundle.getInt("size");
                String temp = app.getTime(current) + "/" + app.getTime(size);
                progressBar.setMax(size);
                progressBar.setProgress(current);
                textView.setText(temp);
            }
        }
    }
};
```

Bounded Service

- BoundedService.JAVA

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_service);
    setTitle("Bounded Service");

    app = (MyApplication) getApplication();
    TextView title = findViewById(R.id.textView1);
    title.setText("음악 Service(Bounded Service)");

    progressBar = findViewById(R.id.progress);
    textView = findViewById(R.id.textView2);

    Intent intent = new Intent(this, MusicBoundedService2.class);
    Messenger messenger = new Messenger(handler);
```

Bounded Service

- BoundedService.JAVA

```
        app.setButton(findViewById(R.id.button));
        app.getButton().setText(app.isServiced() ?
                                        "Music Service 중지" : "Music Service 시작");
        app.getButton().setOnClickListener(v -> {
            intent.putExtra("Messenger", messenger);
            intent.putExtra("title", "헤어졌다 만났다");
            intent.putExtra("loop", loop);
            if (!app.isServiced())
                bindService(intent, conn, BIND_AUTO_CREATE);
            else
                unbindService(conn);
        });
    }
```

Bounded Service

- BoundedService.JAVA

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.isChecked())
            loop = false;
        else
            loop = true;
        item.setChecked(loop);
        return true;
    }
}
```

Bounded Service

- MusicBoundedService.JAVA

```
public class MusicBoundedService extends Service {
    private MediaPlayer mediaPlayer;
    private MyApplication app;

    @Override
    public void onCreate() {
        app = (MyApplication) getApplication();
    }
```

Bounded Service

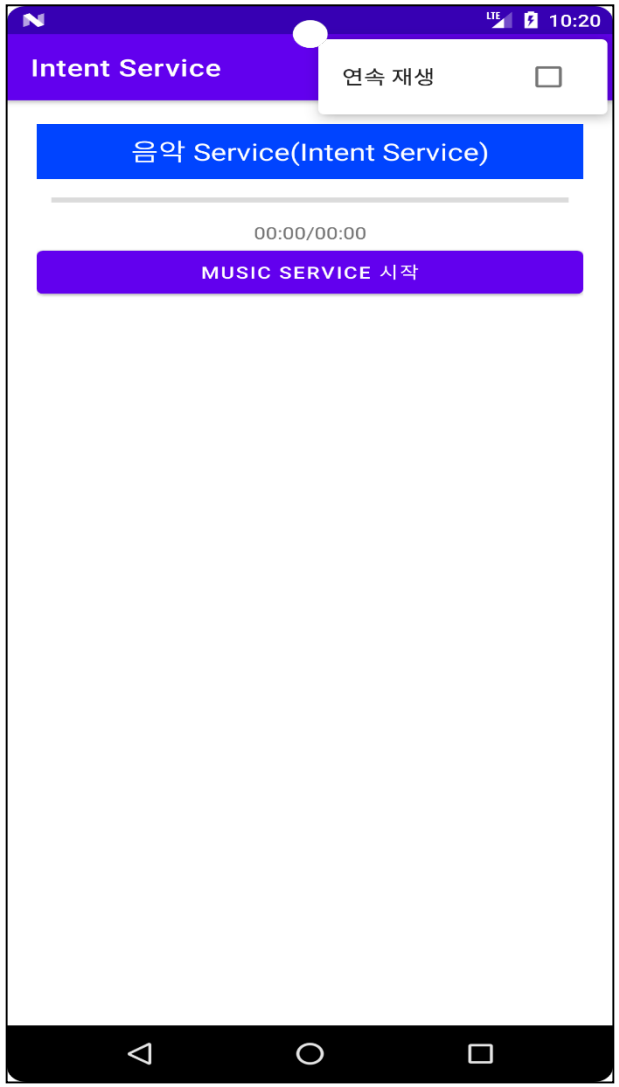
```
@Override
public IBinder onBind(Intent intent) {
    app.setServiced(true);
    app.getButton().setText("Music Service 중지");
    Messenger messenger = intent.getParcelableExtra("Messenger");
    String title = intent.getStringExtra("title");
    boolean loop = intent.getBooleanExtra("loop", false);
    mediaPlayer = MediaPlayer.create(this, R.raw.davichi);
    mediaPlayer.setLooping(loop);
    mediaPlayer.setOnCompletionListener(mediaPlayer -> {
        Toast.makeText(getBaseContext(), "음악 재생 완료",
                                                Toast.LENGTH_SHORT).show();
        app.getButton().performClick();
    });
    mediaPlayer.start();
    Toast.makeText(this, "Music Service가 시작되었습니다.₩n 노래제목 : "
                                            + title, Toast.LENGTH_LONG).show();
    MyThread2 thread = new MyThread2(mediaPlayer,
                                    mediaPlayer.getDuration(), messenger);
    thread.start();
    return null;
}
```

Bounded Service

- MusicBoundedService.JAVA

```
    @Override
    public boolean onUnbind(Intent intent) {
        app.setServiced(false);
        app.getButton().setText("Music Service 시작");
        return super.onUnbind(intent);
    }

    @Override
    public void onDestroy() {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
        }
        Toast.makeText(getBaseContext(), "Music Service가 중지되었습니다",
                                                    Toast.LENGTH_LONG).show();
    }
}
```

Intent Service

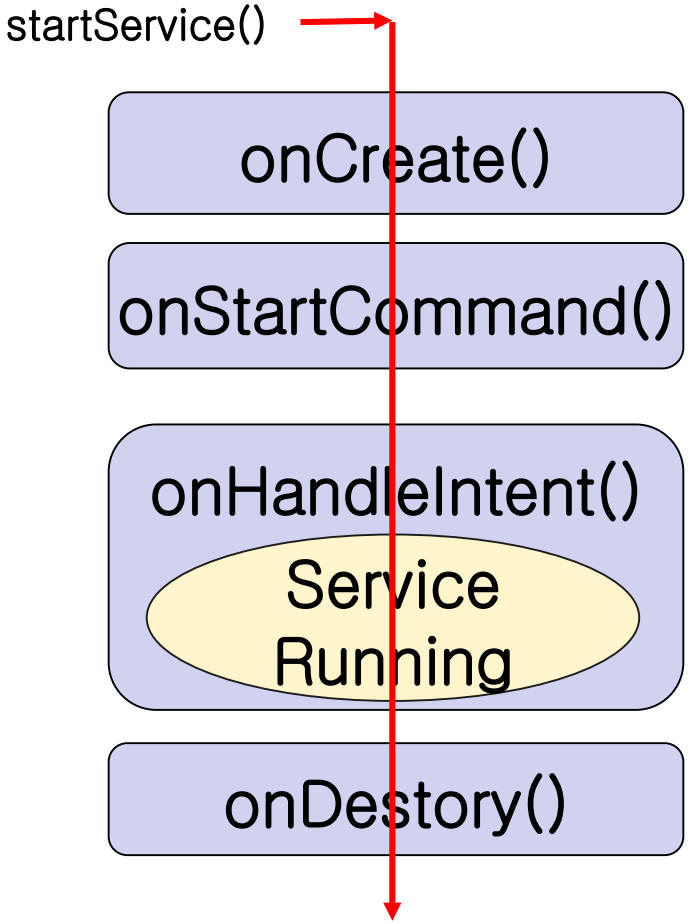


Intent Service

- Intent Service는 Service를 상속받은 클래스로서, 시작 요청이 들어올 때마다 작업을 수행하는 Thread를 별도로 생성하는 Service
- Intent Service도 Started Service이며 Service가 다중 요청을 동시에 처리할 필요가 없는 경우에 가장 좋은 선택이 됨
- Intent Service는 Intent를 전달하여 Service의 어떤 작업을 수행하는데 사용될 수 있음
- File Download나 Upload 등 처리 시간이 긴 작업을 수행하는데 사용할 수 있음
- Intent만 전달하면 되기 때문에 사용하기 간편함
- Intent Service는 Service 진행 중에 Service를 중단시킬 수 있는 방법이 없음

Intent Service

- Intent Service는 자신에게 주어진 업무만 끝나면 자동으로 종료되는 특징



Intent Service

- Intent Service 구현
  - Intent Service를 활용하려면 다음과 같이 Service 클래스가 아닌 IntentService 클래스를 파생
  - onHandleIntent는 Service 별, 일반적으로 시간이 많이 소요되는 처리를 수행

```
public class MyIntentService extends IntentService{

   @Override
   public void onHandleIntent(Intent intent){
      // 여기서 시간이 많이 걸리는 작업을 진행
         예를 들어 웹에서 파일을 다운로드하는
         작업을 할 수 있다
   }
}
```

Intent Service

- 순차 처리
  - Work Queue(작업 큐) Process Pattern
  - onHandleIntent() 메소드가 단일 작업자 Thread에서 처리하기 위해 실행될 때 Service를 시작하기 위한 추가 요청이 이루어지면 요청이 즉시 처리되지 않고 Queue에 배치
  - onHandleIntent 실행이 완료되면 Work Queue가 선택되고 대기가 없을 때 Destroy가 호출
  - Queue의 모든 작업이 완료되면 Service 중단이 자동으로 발생, 이 작업은 stopService() 또는 stopSelf()를 명시적으로 호출하지 않고 수행

Intent Service

- Intent Service의 장점
  - Service의 콜백 메소드들에 대해서 default 구현을 제공하기 때문에 별도로 콜백 메소드를 구현할 필요가 없음
  - 단지 onHandleIntent() 메소드만 구현하면 되고 이 메소드 안에서 Client에 의해 제공되는 작업을 구현
  - 요청한 작업이 완료되면 자동적으로 Service를 중단하기 때문에 stopSelf()또는 stopService()를 호출할 필요 없음

Intent Service

- Started Service와 Intent Service 차이점
  - 언제 사용
    - Started Service
      - UI없이 실행될 수 있지만 매우 길지 않아야 함
      - 만약 오래 걸리는 작업을 Service에서 실행하고자 한다면 Service안에서 Thread를 사용해야 함
    - Intent Service
      - 오래 걸리지만 Main Thread와 관련이 없는 작업을 할 때 주로 이용
      - 만약 Main Thread와 관련된 작업을 해야 한다면 Main Thread Handler나 Boradcast intent를 이용함

Intent Service

- Start Service와 Intent Service 차이점
  - 어떻게 실행시키나
    - Started Service
      - startService() 메소드 호출에 의해 실행
    - Intent Service
      - Intent 사용에 의해 실행
      - 새로운 Thread가 생성되며 onHandleIntent()가 호출됨
  - 호출되는 위치
    - Started Service와 Intent Service 모두 아무 Thread에서나 생성되고, Activity 뿐만 아니라 다른 곳에서도 실행 가능

Intent Service

- Start Service와 Intent Service 차이점
  - 실행중인 위치
    - Started Service
      - Background에서 동작하지만 Main Thread에 포함
    - Intent Service
      - 새로운 Thread에서 동작
  - Service 중지
    - Started Service
      - 순전히 사용자의 몫
      - stopSelf()나 stopService()에 의해 동작이 멈춤
      - Service Binding의 경우 필요 없음
    - Intent Service
      - onHandleIntent()내의 모든 동작이 수행되면 멈춤
      - 멈추기 위한 다른 메소드 호출은 불필요

Intent Service

- Start Service와 Intent Service 차이점
  - 단점
    - Started Service
      - Main Thread에 포함되므로 무거운 작업일 때 Main Thread에 영향을 주어 느려지거나 할 수 있음
    - Intent Service
      - 병렬적으로 수행될 수 없으므로 연속적인 Intent 호출에 관해서 순차적으로 처리됨

Intent Service

- Intent Service는 Background 작업 결과값을 반환하지 않는 단점이 있는데 이를 해결할 방법
  - PendingIntent를 이용해 onActivityResult 메소드 이용
  - 사용자에게 통지 하는 System 통지 이용
  - Messenger를 사용해 Message를 Handler에게 전송
  - 결과값 BroadCast

Intent Service

- MyIntentService.JAVA

```
public class MyIntentService2 extends AppCompatActivity {
    private MyApplication app;
    private ProgressBar progressBar;
    private TextView textView;
    private boolean loop;

    Handler handler = new Handler(Looper.getMainLooper()) {
        @Override
        public void handleMessage(@NonNull Message msg) {
            if (app.isServiced()) {
                if (msg.what == 0) {
                    Bundle bundle = msg.getData();
                    int current = bundle.getInt("current");
                    int size = bundle.getInt("size");
```

Intent Service

- MyIntentService.JAVA

```
                String msg1 = app.getTime(current) + "/" + app.getTime(size);
                progressBar.setMax(size);
                progressBar.setProgress(current);
                textView.setText(msg1);
            }
        }
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_service);
    setTitle("Intent Service");

    app = (MyApplication) getApplication();
    TextView title = findViewById(R.id.textView1);
    title.setText("음악 Service(Intent Service)");
```

Intent Service

- MyIntentService.JAVA

```
    progressBar = findViewById(R.id.progress);
    textView = findViewById(R.id.textView2);

    Intent intent = new Intent(this, MusicIntentService2.class);
    Messenger messenger = new Messenger(handler);
    app.setButton(findViewById(R.id.button));
    app.getButton().setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (!app.isServiced()) {
                intent.putExtra("Messenger", messenger);
                intent.putExtra("title", "헤어졌다 만났다");
                intent.putExtra("loop", loop);
                startService(intent);
            }
        }
    });
}
```

Intent Service

- MyIntentService.JAVA

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.isChecked())
            loop = false;
        else
            loop = true;
        item.setChecked(loop);
        return true;
    }
}
```

Intent Service

- MusicIntentService.JAVA

```
public class MusicIntentService extends IntentService {
    private MediaPlayer mediaPlayer;
    private MyApplication app;
    private Handler handler;

    public MusicIntentService() {
        super("MusicIntentService");
    }
```

Intent Service

- MusicIntentService.JAVA

```
public void onCreate() {
    super.onCreate();
    app = (MyApplication) getApplication();
    handler = new Handler();
    app.setServiced(true);
    app.getButton().setText("Music Service 중지");
    app.getButton().setClickable(false);
    mediaPlayer = MediaPlayer.create(this, R.raw.davichi);
    mediaPlayer.setOnCompletionListener(mp -> {
        Toast.makeText(getBaseContext(), "음악 재생 완료",
                                                Toast.LENGTH_SHORT).show();
    });
}
```

Intent Service

- MusicIntentService.JAVA

```
@Override
protected void onHandleIntent(Intent intent) {
    if (intent == null)
        return;
    Messenger messenger = intent.getParcelableExtra("Messenger");
    String title = intent.getStringExtra("title");
    boolean loop = intent.getBooleanExtra("loop", false);
    mediaPlayer.setLooping(loop);
    mediaPlayer.start();
    int size = mediaPlayer.getDuration();
    handler.post(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(getBaseContext(), "Music Service가 시작되었습니다.
                    ₩n 노래제목 : " + title, Toast.LENGTH_LONG).show();
        }
    });
```

Intent Service

```
NotificationCompat.Builder builder = app.makeNoti();
startForeground(1, builder.build());
while (mediaPlayer != null && mediaPlayer.isPlaying()) {
    int current = mediaPlayer.getCurrentPosition();
    Message message = handler.obtainMessage(0);
    Bundle bundle = new Bundle();
    bundle.putInt("current", current);
    bundle.putInt("size", size);
    message.setData(bundle);
    try {
        messenger.send(message);
    } catch (RemoteException e) {
        handler.post(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(getBaseContext(), e.getMessage(),
                                                    Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Intent Service

- MusicIntentService.JAVA

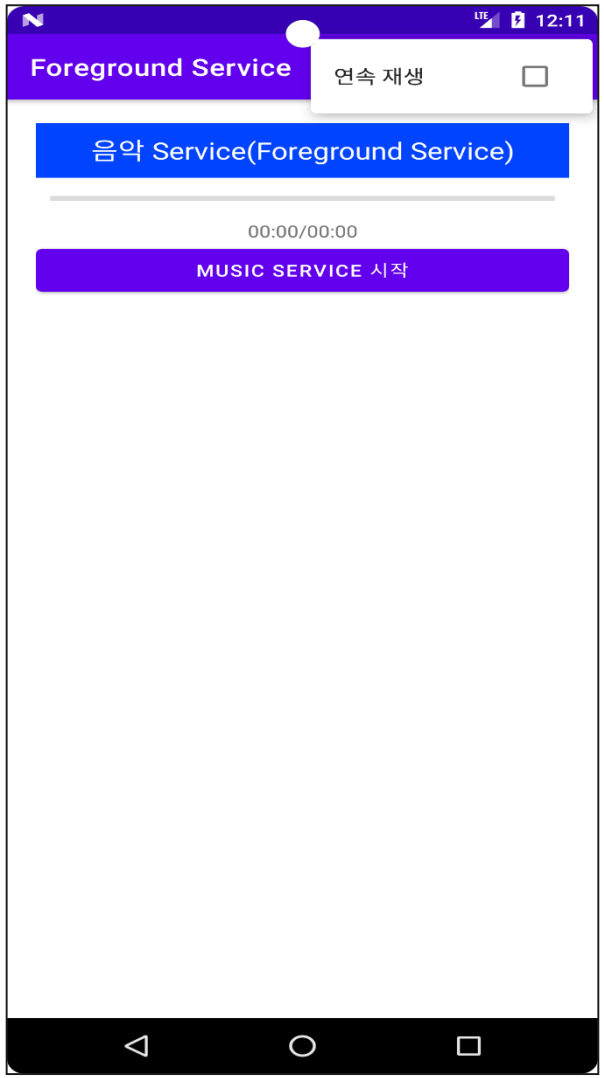
```
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                break;
            }
        }
    }
```

Intent Service

- MusicIntentService.JAVA

```
    @Override
    public void onDestroy() {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
            mediaPlayer.reset();
        }
        app.setServiced(false);
        app.getButton().setText("Music Service 시작");
        app.getButton().setClickable(true);
        Toast.makeText(this, "Music Service가 중지되었습니다",
                                                    Toast.LENGTH_LONG).show();
    }
}
```

Foreground Service



Foreground Service

- Foreground Service는 사용자가 명확히 인식할 수 있도록 Notification을 통해 실행 상태를 지속적으로 표시하며, 중요한 작업을 Background에서 실행하기 위해 설계된 Android Service
- 이 Service는 작업 중단의 가능성을 최소화하며, 주로 사용자가 지속적으로 작업 진행 상황을 알 필요가 있는 경우에 사용
- Foreground Service로 사용할 Service를 정의하고, 반드시 startForeground() 메소드를 호출하여 Notification을 설정해야 함

Foreground Service

- Foreground Service와 Started Service의 주요 차이

특징	Foreground Service	Started Service
Notification 표시	항상 사용자에게 Notification으로 실행 상태를 표시해야 함	Notification 없이 실행 가능 (필요시 선택적으로 표시)
사용자 인지 여부	사용자에게 Service 실행 상태가 명확히 드러남	사용자가 실행 여부를 알지 못할 수도 있음
우선순위	높은 우선순위 (System이 잘 종료하지 않음)	낮은 우선순위 (Memory 부족 시 종료될 가능성 높음)
적용 사례	음악 재생, 위치 추적, 파일 다운로드 등 사용자와 관련된 작업	Data 동기화, 단기간 Background 작업 등
Background 제한	Android 8.0 이상에서도 제한 없이 실행 가능	Android 8.0 이상에서는 Background 제한이 있음
자동 종료 여부	작업이 끝나도 Notification이 남아 수동으로 종료 필요	작업 완료 후 수동 또는 자동으로 종료 가능

Foreground Service

- Foreground Service와 Notification
  - Foreground Service는 항상 사용자에게 알림을 표시해야 함
  - Android 8.0(Oreo) 이상에서는 Foreground Service 시작 시 알림 채널(Notification Channel)을 반드시 생성해야 함

Foreground Service

- 장점
  - 중요 작업 보장
    - Service가 높은 우선순위를 가지며, System에 의해 쉽게 종료되지 않음
  - 사용자 알림
    - 작업 진행 상황을 실시간으로 사용자에게 전달 함
- 단점
  - 알림 필수
    - 항상 알림을 표시해야 하므로 사용자가 불필요한 알림을 볼 수 있음
  - Battery 소모
    - 장기간 실행되는 Service는 Battery 소모를 증가시킬 수 있음
  - Background 제약
    - Android 8.0 이상에서 Background에서 Foreground Service를 실행하려면 추가 권한이 필요

Foreground Service

- ForegroundService.JAVA

```
public class ForegroundService2 extends AppCompatActivity {
    private MyApplication app;
    private ProgressBar progressBar;
    private TextView textView;
    private boolean loop;
```

Foreground Service

- ForegroundService.JAVA

```
Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(@NonNull Message msg) {
        if (app.isServiced()) {
            if (msg.what == 0) {
                Bundle bundle = msg.getData();
                int current = bundle.getInt("current");
                int size = bundle.getInt("size");
                String msg1 = app.getTime(current) + "/" + app.getTime(size);
                progressBar.setMax(size);
                progressBar.setProgress(current);
                textView.setText(msg1);
            }
        }
    }
};
```

Foreground Service

- ForegroundService.JAVA

```
        app.setButton(findViewById(R.id.button));
        app.getButton().setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                intent.putExtra("Messenger", messenger);
                intent.putExtra("title", "헤어졌다 만났다");
                intent.putExtra("loop", loop);
                if (!app.isServiced()) {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
                        startForegroundService(intent);
                    else
                        startService(intent);
                } else
                    stopService(intent);
            }
        });
    }
```

Foreground Service

- ForegroundService.JAVA

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.isChecked())
            loop = false;
        else
            loop = true;
        item.setChecked(loop);
        return true;
    }
}
```

Foreground Service

- MusicForegroundService.JAVA

```
public class MusicForegroundService extends Service {
    private MediaPlayer mediaPlayer;
    private MyApplication app;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
```

Foreground Service

- MusicForegroundService.JAVA

```
@Override
public void onCreate() {
    super.onCreate();
    app = (MyApplication) getApplication();

    mediaPlayer = MediaPlayer.create(this, R.raw.davichi);
    mediaPlayer.setOnCompletionListener(
                                        new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mediaPlayer) {
            Toast.makeText(getBaseContext(), "음악 재생 완료",
                                                Toast.LENGTH_SHORT).show();
            stopSelf();
        }
    });
}
```

Foreground Service

- MusicForegroundService.JAVA

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    app.setServiced(true);
    app.getButton().setText("Music Service 중지");
    Messenger messenger = intent.getParcelableExtra("Messenger");
    String title = intent.getStringExtra("title");
    boolean loop = intent.getBooleanExtra("loop", false);
    mediaPlayer.setLooping(loop);
    mediaPlayer.start();
    Toast.makeText(this, "Music Service가 시작되었습니다.₩n 노래제목 : " + title,
            Toast.LENGTH_LONG).show();

    MyThread2 thread = new MyThread2(mediaPlayer,
                                     mediaPlayer.getDuration(), messenger);
    thread.start();
    NotificationCompat.Builder builder = app.makeNoti();
    startForeground(1, builder.build());
    return START_STICKY;
}
```

Foreground Service

- MusicForegroundService.JAVA

```
    @Override
    public void onDestroy() {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop(); // 음악 종료
            mediaPlayer.reset();
        }
        app.setServiced(false);
        app.getButton().setText("Music Service 시작");
        Toast.makeText(this, "Music Service가 중지되었습니다",
                                                    Toast.LENGTH_LONG).show();
    }
}
```

Remote Service

- RemoteService.JAVA

```
public class RemoteService2 extends AppCompatActivity {
    private MyApplication app;
    private ProgressBar progressBar;
    private TextView textView;
    private Messenger serviceMessenger = null;
    private boolean connection = false;
    private boolean loop;
```

Remote Service

- RemoteService.JAVA

```
ServiceConnection connect = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        serviceMessenger = new Messenger(service);
        connection = true;
        Toast.makeText(getBaseContext(), "서비스 연결 완료",
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        serviceMessenger = null;
        connection = false;
        Toast.makeText(getBaseContext(), "서비스 연결 해제",
                Toast.LENGTH_SHORT).show();
    }
};
```

Remote Service

- RemoteService.JAVA

```
Handler handler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(@NonNull Message msg) {
        if (app.isServiced()) {
            if (msg.what == 0) {
                Bundle bundle = msg.getData();
                int current = bundle.getInt("current");
                int size = bundle.getInt("size");
                String msg1 = app.getTime(current) + "/" + app.getTime(size);
                progressBar.setMax(size);
                progressBar.setProgress(current);
                textView.setText(msg1);
            }
        }
    }
};
```

Remote Service

- RemoteService.JAVA

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_service);
    setTitle("Remote Service");

    app = (MyApplication) getApplication();
    TextView title = findViewById(R.id.textView1);
    title.setText("음악 Service(Remote Service)");

    progressBar = findViewById(R.id.progress);
    progressBar.setMax(app.getMaxProgress());
    progressBar.setProgress(app.getProgress());
    textView = findViewById(R.id.textView2);

    Intent intent = new Intent(this, MusicRemoteService2.class);
    Messenger replyMessenger = new Messenger(handler);
```

Remote Service

- RemoteService.JAVA

```
app.setButton(findViewById(R.id.button));
app.getButton().setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!connection) {
            bindService(intent, connect, Context.BIND_AUTO_CREATE);
        } else if (serviceMessenger != null) {
            Message message;
            if (!app.isServiced()) {
                message = Message.obtain(null, 1);
                Bundle bundle = new Bundle();
                bundle.putString("title", "헤어졌다 만났다");
                bundle.putBoolean("loop", loop);
                message.setData(bundle);
                message.replyTo = replyMessenger;
            } else
                message = Message.obtain(null, 2);
```

Remote Service

- RemoteService.JAVA

```
                try {
                    serviceMessenger.send(message);
                } catch (RemoteException e) {
                    Toast.makeText(getBaseContext(), e.getMessage(),
                            Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = getMenuInflater();
    menuInflater.inflate(R.menu.menu, menu);
    return true;
}
```

Remote Service

- RemoteService.JAVA

```
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.isChecked())
            loop = false;
        else
            loop = true;
        item.setChecked(loop);
        return true;
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (app.isServiced()) {
            unbindService(connect);
        }
    }
}
```

Remote Service

- MusicRemoteService.JAVA

```
public class MusicRemoteService extends Service {
    private MediaPlayer mediaPlayer;
    private MyApplication app;
    final int CONNECT = 1;
    final int DISCONNECT = 2;

    Messenger messenger = new Messenger(new CallbackHandler());

    @Override
    public IBinder onBind(Intent intent) {
        return messenger.getBinder();
    }

    @Override
    public void onCreate() {
        super.onCreate();
        app = (MyApplication) getApplication();
    }
```

Remote Service

- MusicRemoteService.JAVA

```
private class CallbackHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case CONNECT:
                app.setServiced(true);
                app.getButton().setText("Music Service 중지");
                Messenger replayMessenger = msg.replyTo;
                String title = msg.getData().getString("title");
                boolean loop = msg.getData().getBoolean("loop");
                mediaPlayer = MediaPlayer.create(getBaseContext(), R.raw.davichi);
                mediaPlayer.setLooping(loop);
```

Remote Service

- MusicRemoteService.JAVA

```
mediaPlayer.setOnCompletionListener(
                    new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        Toast.makeText(getBaseContext(), "음악 재생 완료",
                        Toast.LENGTH_SHORT).show();
        app.setProgress(0);
        app.getButton().performClick();
    }
});
mediaPlayer.start();
Toast.makeText(getBaseContext(), "Music Service가 시작되었습니다.
                    ₩n 노래제목 : " + title,
        Toast.LENGTH_LONG).show();

MyThread2 thread = new MyThread2(mediaPlayer,
          mediaPlayer.getDuration(), replayMessenger);
thread.start();
```

Remote Service

- MusicRemoteService.JAVA

```
                NotificationCompat.Builder builder = app.makeNoti();
                startForeground(1, builder.build());
                break;
            case DISCONNECT:
                app.setServiced(false);
                if (mediaPlayer.isPlaying()) {
                    mediaPlayer.stop();
                    mediaPlayer.reset();
                }
                Toast.makeText(getBaseContext(),
                 "Music Service가 중지되었습니다", Toast.LENGTH_LONG).show();
                app.getButton().setText("Music Service 시작");
        }
    }
}
```

Remote Service

- MusicRemoteService.JAVA

```
    @Override
    public void onDestroy() {
        if (mediaPlayer != null) {
            mediaPlayer.release();
            mediaPlayer = null;
        }
        Toast.makeText(getBaseContext(), "Music Service가 중지되었습니다",
                                                    Toast.LENGTH_LONG).show();
    }
}
```

Remote Service

특징	Remote Service	Bounded Service
기본 동작	다른 Process(App)에서 접근할 수 있도록 설계된 Service	같은 Process내에서만 Client와 Binding 가능
주 사용 목적	프로세스 간 통신(Inter-Process Communication)을 위해 사용	Client와의 Data 상호작용을 위해 사용
Process 위치	Service가 독립된 Process에서 실행(다른 App에서 접근 가능)	Service가 호출한 App(Client)과 같은 Process에서 실행
통신 방법	Messenger, AIDL(Android Interface Definition Language)을 주로 사용	Binder 또는 Messenger를 사용
설정 방법	Manifest에 android:process 속성을 지정하여 별도의 Process로 실행	별도 Process 지정 없이 기본적으로 호출한 App의 Process에서 실행
보안	IPC를 사용하므로 권한 관리(예: permission)가 중요	같은 Process에서만 작동하므로 보안 이슈가 적음

Remote Service

특징	Remote Service	Bounded Service
서비스 시작 방식	✓ startService() 또는 bindService()로 시작 가능 ✓ 원격에서 bindService()를 통해 Binding이 주로 사용됨	bindService()로만 시작 가능
서비스 종료 방식	✓ stopService(), stopSelf()로 명시적으로 종료 가능 ✓ 모든 Client가 Binding 해제해도 명시적으로 종료되지 않으면 계속 실행	모든 Client가 unbindService()를 호출하면 자동 종료
성능	IPC 사용으로 인해 호출과 Data 전달 시 성능이 상대적으로 낮음	같은 Process 내에서 동작하므로 상대적으로 성능이 높음
수명 주기	Client가 Binding을 해제해도 Service는 계속 실행될 수 있음	모든 Client가 Binding을 해제하면 자동으로 종료
예시	원격 데이터베이스 접근, 메시징 서비스 등	내부 데이터 요청, UI 업데이트 등

Remote Service

- Remote Service는 일반적인 Service Lifecycle과 유사하지만, 주로 Client와의 Binding 과정에서 차별화 됨
- Remote Service는 Bounded Service로 구현되며, Service가 Client와 연결되어 IPC를 수행.
- Remote Service는 일반적으로 AIDL(Android Interface Definition Language) 또는 Binder를 사용하여 IPC(Inter-Process Communication)를 구현

Messenger

- Android의 Messenger는 IPC(Inter-Process Communication)를 위해 사용되는 도구 중 하나
- Handler와 Message 객체를 통해 Client와 Service간 Data를 교환하는 방법을 제공
- Messenger는 상대적으로 간단한 IPC 구현을 가능하게 하며, AIDL보다 쉬운 대안으로 사용할 수 있음
- 하나의 Client-Service 연결에 적합하며, 여러 Client를 처리할 때는 추가 작업이 필요
- 특징
  - Handler 기반으로 동작하므로, Single Thread에서 요청을 순차적으로 처리
  - 복잡한 Data 처리를 요구하지 않으면서 IPC가 필요한 경우 유용
  - Message를 통해 작업 요청을 전달하며, 비동기 작업 처리에도 활용 가능

Messenger

- Messenger의 주요 구성 요소
  - Messenger
    - Handler를 기반으로 IPC를 관리
    - Client와 Service간 통신을 위해 생성
  - Handler
    - Message를 처리하기 위해 Service와 Client 모두에서 사용
    - Message를 수신하고 필요한 작업을 수행
  - Message
    - IPC Data 전달을 위한 객체
    - what, arg1, arg2 및 data(Bundle)를 통해 Data를 보낼 수 있음
  - replyTo
    - Client가 Service로부터 응답을 받을 때 사용하는 Messenger 객체

Bounded Service

- IBinder 생성 방법
  - Binder 클래스 확장
    - 동일한 Process 내에서 Server-Client 형태로 상호작용 하는 경우 사용하는 방식 (대부분)
  - Messenger 사용
    - 여러 Process에서 적용되도록 할 때 사용하는 방식
    - IPC(프로세스간 통신)를 구현하는 가장 간단한 방법
  - AIDL 사용
    - 여러 Process에서 적용되어야 할 때 객체를 OS가 이해할 수 있는 원시 유형으로 해체한 다음 여러 Process에 걸쳐 마샬링하여 IPC를 수행하지만 대부분의 경우에는 Bind된 Service를 사용하기 위해서 AIDL을 사용하지 않음

Bounded Service

- Marshalling(마샬링)
  - 객체의 Memory 구조를 저장이나 전송을 위해서 적당한 자료 형태로 변형하는 것을 의미
  - Marshalling은 보통 서로 다른 Computer 혹은 서로 다른 Program 간에 Data가 이동되어야 할 경우 사용
  - Marshalling을 수행함으로써 복잡한 통신, 사용자 정의/복잡한 구조의 객체들을 사용하는 대신, 단순한 primitive 들을 사용할 수 있음
  - Marshalling의 반대말은 Unmarshalling 이라고 함

Bounded Service

- Activity는 Service에 어떠한 요청을 할 수 있고, Service로부터 어떠한 결과를 받을 수 있음
- Process간 통신에도 사용 가능
- Service Binding은 연결된 Activity가 사라지면 Service도 소멸 됨 (즉 Background에서 무한히 실행되지 않음)
- 하나의 Service에 다수의 Activity 연결 가능
- Application 안의 기능을 외부에 제공하는 경우에 많이 사용

Bounded Service

- Binder
  - Linux Kernel에 있는 Binder를 통해 관리
  - Application과 Service들이 서로 다른 Process에서 실행 중인 경우, Process간의 통신 및 Data 공유의 관련 문제가 발생 -> 이때 IPC를 통해 Process간 통신
    - 필연적으로 IPC 통신의 경우, System Overhead 및 보안 관련 문제를 야기할 수 있는 가능성이 항상 존재
- Android Binder는 IPC 통신을 제공하는 Protocol
  - 공유 Memory를 통한 성능 향상, Process마다 요청 처리를 위한 Thread Pool
  - Object Mapping과 Reference Counting 기능 제공
  - Binder를 통해 IPC가 이루어 질 때는 기본적으로는 동기 방식
  - 요청한 Process는 요청 결과를 얻을 때까지 대기 상태

Bounded Service

- Binder
  - ServiceManager -> BinderDriver에 대한 Special Node로 동작
  - ServiceProvider -> 자신의 RPC에 대한 Interface를 ServiceManager proxy object에 대한 Interface를 통해 등록
    - 이 과정을 통해 BinderDriver에서는 ServiceProvider로의 path를 생성
  - Service User -> 사용하려는 Service에 대한 Interface를 생성 위해 ServiceManager에게 RPC call을 통해 요청
  - ServiceManager -> Bind되어있는 Object들의 목록에서 해당하는 ServiceProvider에 대한 Node를 요청한 ServiceUser에게 반환
  - ServiceUser -> return된 Node 정보를 통해 해당 Service Provider로의 RPC를 수행

Bounded Service

- Service에서 Binder를 사용하는 이유
  - Activity가 Service에 Binding되면, 그 Activity는 해당 Service Instance 자체에 대한 Reference를 유지
  - Service가 Binding 된 후, Service가 가지고 있는 모든 public 메소드와 프로퍼티는 onServiceConnected Handler를 통해서 얻어진 serviceBinder 객체를 통해서 이용 가능
  - 일반 Service는 Client가 Service의 동작을 알지 못하게 은밀히 동작 그러나 Binding된 Service는 Activity가 상태를 파악하고 제어
  - Broadcast Intent 또는 Service 시작 시 사용되는 Intent Extra Bundle을 이용해서 간단히 다른 Process로 실행 중인 Service와 간단히 통신 가능
  - 좀더 단단히 결합된 연결을 원할 경우에는 AIDL 이용

Bounded Service

- startService
  - 받는 쪽과 묶어서 돌아가는 구조
  - 호출 순서 : onCreate() -> onStart() -> ... -> onDestory()
  - Service가 실행중인 상태라면 startService()가 호출되어도 상태 유지, Service 종료 상태에서 stopService()하면 아무 일도 발생하지 않음
  - bindService로 실행된 Service를 startService로 호출하게 되면 이전(bindService) 관리는 무시되고 새로운 상태로 변경
  - Service가 Resource를 너무 많이 사용하게 되면 System에 의해 종료될 수 있음