

● 서비스: 백그라운드 작업을 위한 컴포넌트로 화면과 상관없이 장시간 동안 처리해야 하는 업무를 구현할 때 사용한다. 하지만 시간이 오래 걸리는 업무라도 앱의 화면이 스마트폰에 떠 있는 동안만 진행되어야 한다면, 서비스가 아닌 액티비티로 작성하는 게 좋다. 즉, 서비스는 다른 앱이 사용자 화면을 점유하고 있더라도 앱의 프로세스가 살아서 계속 무언가 작업을 수행해야 할 때 사용하는 컴포넌트이다.

▶ 작성 방법: Service라는 클래스를 상속 받아서 작성한다.

```
public class PlayService extends Service {
    //...
}
```

이렇게 작성한 서비스도 컴포넌트 클래스이므로 AndroidManifest.xml 파일에 등록해야 한다. 액티비티나 브로드캐스트 리시버와 마찬가지로 암시적 인텐트로 수행되게 하려면 <intent-filter>를 서브 태그로 등록해 주어야 한다.

```
<service
    android:name=".PlayService"
    android:enabled="true"
    android:exported="true"></service>
```

▶ 실행/종료 방법: 서비스를 실행하고 종료하기 위한 두 가지 함수가 있다. 서비스를 실행하면서 인텐트의 Extra 데이터를 전달할 수 있다. 그런데 서비스는 액티비티나 브로드캐스트 리시버와는 다르게 서비스를 종료하는 인텐트 발생도 제공한다. (서비스를 구동시킨 곳에서 종료 인텐트를 날리는 것이 아니다)

1. startService(), stopService()

```
Intent intent = new Intent(this, PlayService.class);
startService(intent);
```

```
Intent intent = new Intent(this, PlayService.class);
stopService(intent);
```

2. bindService(), unbindService(): 서비스를 bindService() 함수로 실행하려면, ServiceConnection 인터페이스를 구현한 객체를 준비해야 한다. ServiceConnection 인터페이스의 추상 함수인 onServiceConnected()는 bindService() 함수로 인해 서비스가 구동하는 시점에 호출되며, onServiceDisconnected() 함수는 unbindService() 함수로 인해 서비스가 종료된 시점에 호출된다.

```
ServiceConnection connection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName, IBinder service) {
    }

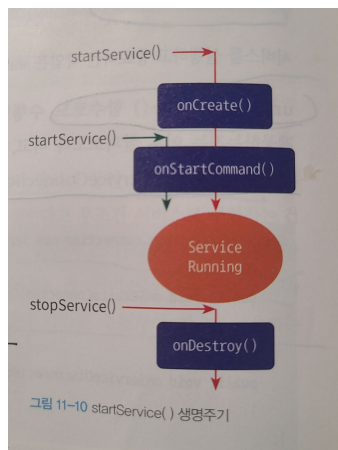
    @Override
    public void onServiceDisconnected(ComponentName name) {
    }
};
```

bindService() 함수로 서비스를 구동하면서 두 번째 매개변수로 ServiceConnection을 구현한 객체를 알려주어야 한다. bindService() 함수로 실행된 서비스는 unbindService() 함수로 종료할 수 있다.

```
Intent intent = new Intent(this, BindService.class);
bindService(bIntent, connection, Context.BIND_AUTO_CREATE);
// ....
unbindService(connection);
```

▶ 서비스 생명주기: 서비스는 구동 방법이 startService()와 bindService() 두 가지이므로 어느 함수로 실행하는지에 따라 생명주기가 다르다.

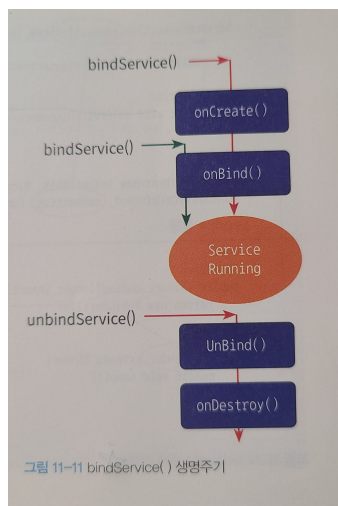
1. startService() 함수로 서비스를 실행하고, stopService() 함수로 종료할 때의 서비스 생명주기



startService() 함수가 서비스를 실행하면 onCreate() → onStartCommand() 함수가 자동으로 호출되면서 Service Running 상태가 되고, stopService() 함수가 서비스를 종료할 때는 onDestroy() 함수가 자동으로 호출되면서 종료된다.

서비스는 액티비티와 다르게 싱글톤으로 동작하는 컴포넌트다. 이미 실행되어 Running 상태인 서비스를 다시 startService() 함수로 실행되면 객체는 다시 생성되지 않는다. onCreate() 함수도 호출되지 않고, onStartCommand() 함수는 다시 호출된다. 즉, onCreate()는 최초 한 번만 호출되는 함수이고, onStartCommand()는 반복 호출할 수 있는 함수이다.

2. bindService() 함수로 서비스를 실행하고, unbindService() 함수로 종료할 때의 서비스 생명주기



bindService() 함수가 서비스를 시작하면 onCreate() → onBind() 함수가 호출되면서 Running 상태가 되고, unbindService() 함수로 서비스를 종료하면서 unbind() → onDestroy() 함수가 호출된다. 서비스는 싱글톤이며 이미 구성된 서비스를 다시 bindService()로 실행하면 객체가 매번 생성되지 않고 onBind() 함수만 반복 호출된다.

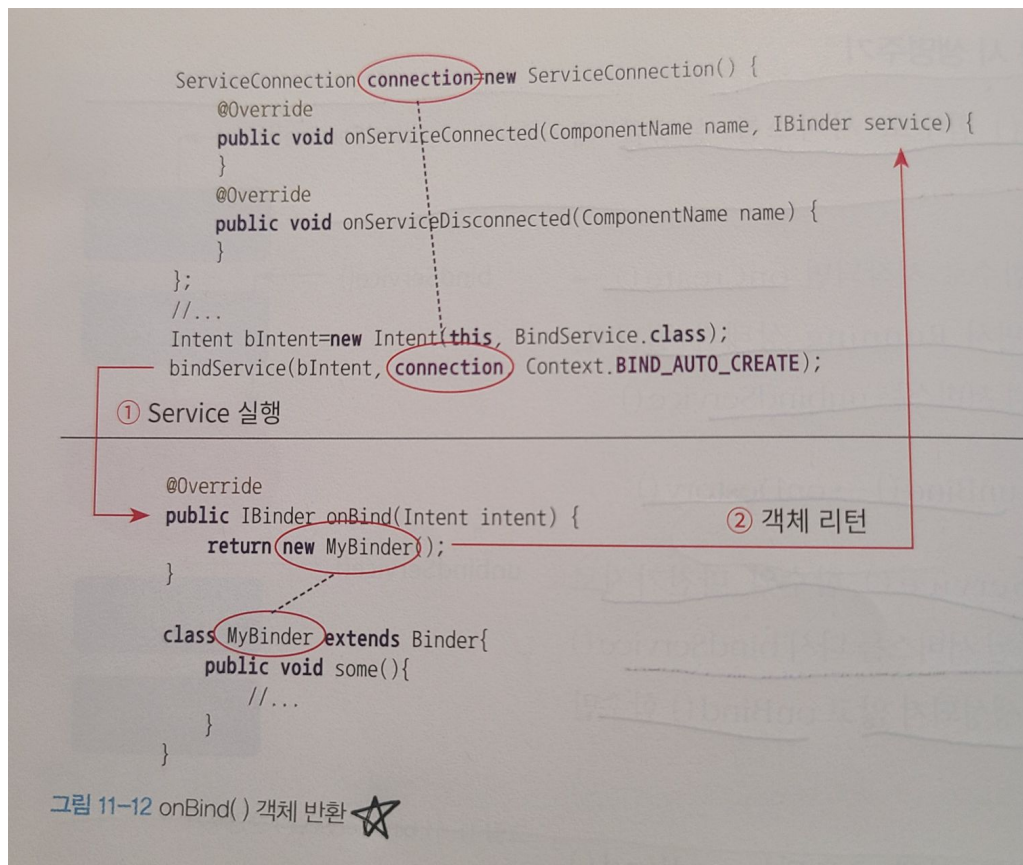
bindService() 함수로 인해 호출되는 onBind() 함수는 다른 생명주기 함수와 다르게 반환값이 있다.

onBind() 함수에서 반환할 객체는 Binder를 상속받아 개발자가 만든 클래스의 객체이다. Binder를 상속받아야 한다는 규칙 외에 클래스 내부의 함수 구현 규칙은 없고, 개발자가 임의로 함수를 정의하면 된다. 이 클래스의 객체를 생성해서 onBind() 함수에서 반환하면 서비스가 bindService() 함수로 실행한 곳에 전달된다. 함수 이름처럼 서비스가 실행되면서 onBind() 함수에서 반환한 객체가 서비스를 실행한 곳에 바

인딩 되는 방법인 것이다. onBind() 함수에서 반환한 객체는 bindService() 함수의 매개변수로 지정한 ServiceConnection 구현 객체의 onServiceConnected() 함수의 매개변수로 전달된다.

```
@Override
public IBinder onBind(Intent intent) {
    return new MyBinder();
}

class MyBinder extends Binder {
    public void some() {
        // ...
    }
}
```



`bindService()` 함수로 실행된 서비스는 자신에게 바인드된 컴포넌트가 하나도 없으면 자동으로 종료된다. 즉, 하나의 액티비티가 `bindService()` 함수로 서비스를 실행한 것이라면, 해당 액티비티가 종료되는 시점이나, `unBindService()` 함수로 종료를 명시하는 시점에 서비스도 함께 종료된다. 물론 여러 개의 액티비티가 `bindService()` 함수로 서비스를 이용하고 있다면, 하나의 액티비티가 종료되거나 `unBindService()` 함수를 호출하더라도 종료되지 않고 계속 움직이게 된다.