西南交通大學
Southwest Jiaotong University

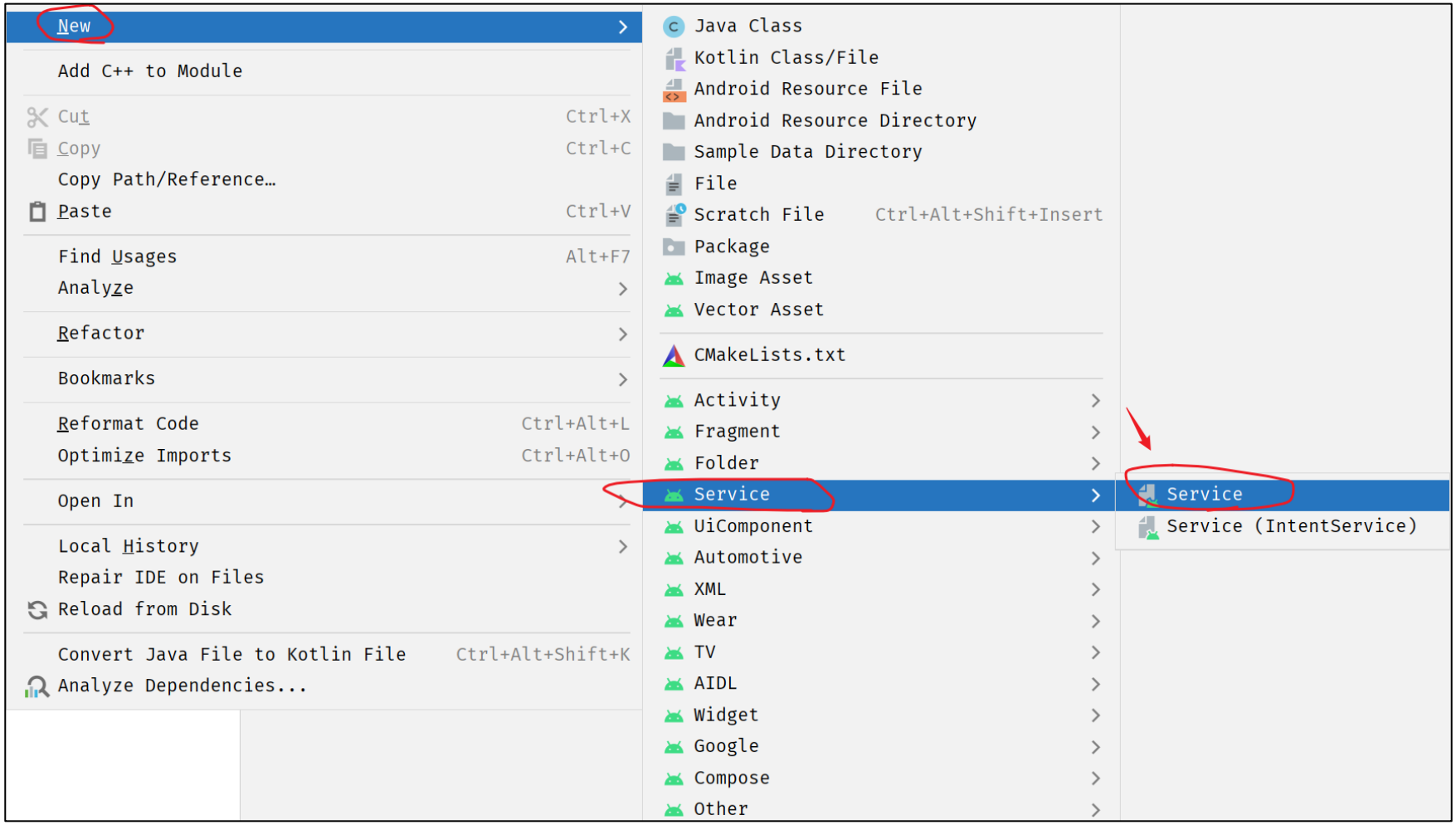
# 移动商务应用开发

## 第9章 服务 Service

# 服务概述

- **服务**（Service）是Android四大组件之一。
- Service和Activity都是继承于Context。
- 服务可**在后台长时间运行**而**不提供用户界面**。
  - 例如服务可在后台处理网络事务、播放音乐、执行文件读写等。

# 服务的创建


New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference…
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Open In
Local History
Repair IDE on Files
Reload from Disk
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Analyze Dependencies...
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
TV
AIDL
Widget
Google
Compose
Other
Service
Service (IntentService)

# 服务的创建

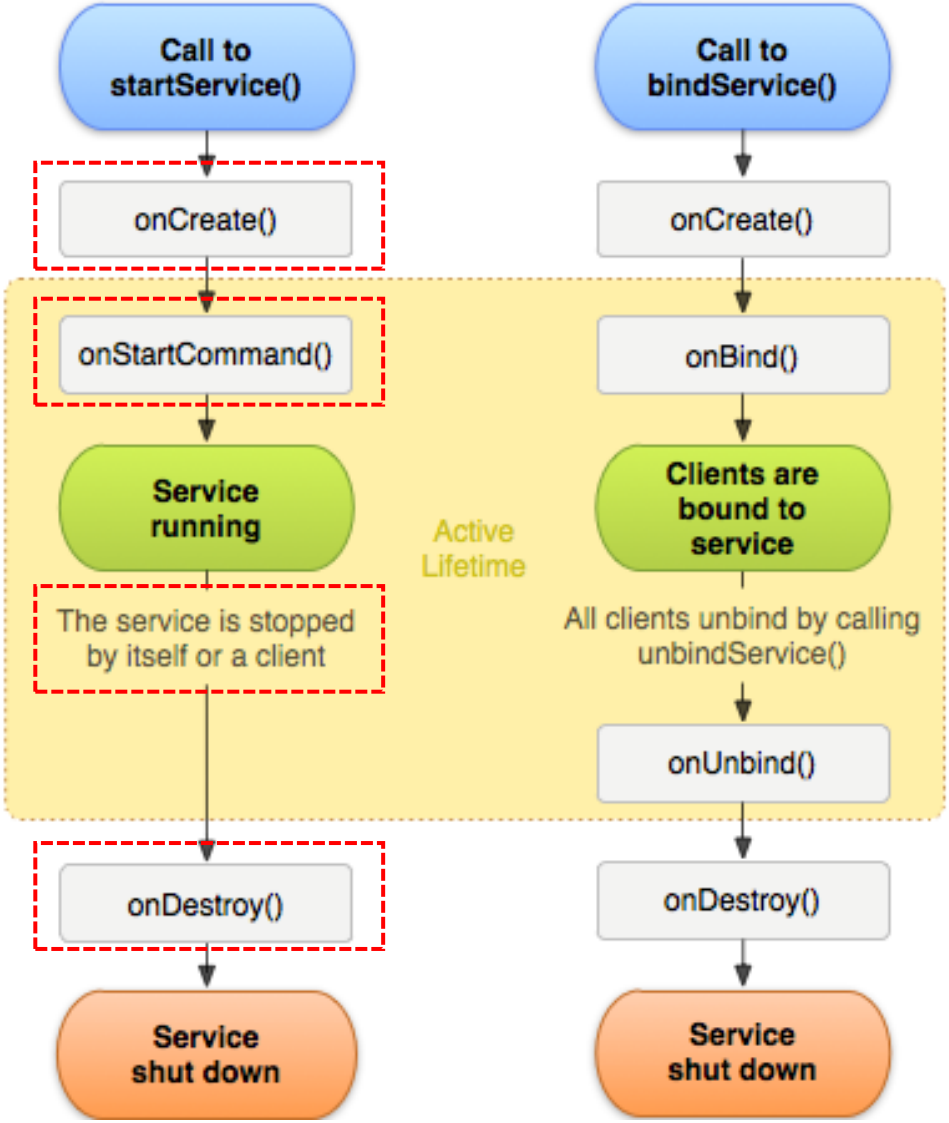
- Android Studio创建Service的过程中完成了：
  - 创建了一个继承Service的java类
  - 将该Service在AndroidManifest.xml注册

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

# 服务的运行

- Android提供了两种运行服务的方式：
  - 通过调用**startService()**启动服务，并通过调用**stopService()或stopSelf()**关闭服务。
  - 通过调用**bindService()**绑定服务，并通过调用**unbindService()**解绑服务。
- **采用不同的运行方式，具有不同的生命周期。**

# 服务的生命周期


Call to startService()
onCreate()
onStartCommand()
Service running
The service is stopped by itself or a client
onDestroy()
Service shut down
Call to bindService()
onCreate()
onBind()
Clients are bound to service
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service shut down
Active Lifetime

# 服务的生命周期

- 生命周期中的回调方法：

| 回调方法 | 描述 |
| --- | --- |
| onCreate() | 创建服务时回调 |
| onDestory() | 销毁服务时回调 |
| onStartCommond() | 调用startService()时回调 |
| onBind() | 在服务未绑定的情况下调用bindService()时回调 |
| onUnbind() | 在服务绑定的情况下调用unbindService()时回调 |

# 服务的生命周期

1. **onCreate()**是生命周期中**第一个**被回调的方法，用于创建服务对象，只会被**执行一次**。
   - 当调用startService()时，如果未创建服务对象，那么系统将依次调用onCreate()和onStartCommand()，如果已创建服务对象，那么系统将只调用onStartCommand()。
2. **onStartService()**回调用于启动服务。服务一旦启动，将会无期限运行，直到调用stopService()或stopSelf()。**每次**调用startService()都将会回调onStartService()。
3. **onDestory()**是生命周期中**最后一个**被回调的方法，用于销毁服务对象，只会被**执行一次**。当调用stopService()或stopSelf() 时，系统将会回调onDestory()。

# 服务的生命周期

1. 当调用bindService()时，如果未创建服务，那么系统将回调**onCreate()**。当调用unbindService()时，如果服务对象未销毁，系统将会回调**onDestory()**。
2. 并不是每次调用bindService()时都会回调**onBind()**。当**服务未绑定时**调用bindService()将**会**回调onBind()，而当**服务绑定时**调用bindService()将**不会**回调onBind()。
3. 并不是每次调用unbindService()时都会回调**onUnbind()**。当**服务绑定时**调用unbindService()将**会**回调onUnbind()，当**服务未绑定时**调用unbindService()将**不会**回调onUnbind()。
4. **onBind()**方法会返回一个**IBinder对象**，该对象用于Activity与Service之间的通信。

# 服务的两种运行方式

- **采用不同的运行方式，Activity与Service的关系不同**

**通过 startService() 启动服务**

如果Activity通过这种方式运行Service，那么**Activity与Service之间没有关联，当Activity的生命周期结束时，Service仍然会继续运行。**

- 退出Activity时**不会回调Service的onDestroy()**

**通过 bindService() 绑定服务**

如果Activity通过这种方式运行Service，那么**Activity与Service绑定在一起了，当Activity被销毁时，Service也会被销毁。**

- 退出Activity时**会回调Service的onUnbind()、onDestroy()**

# 服务的两种运行方式

- **采用不同的运行方式，参数不同**

**通过startService()方法启动服务**

- startService(Intent intent)
  - **Intent intent**指定了启动的Service
- stopService(Intent intent)
  - **Intent intent**指定了关闭的Service

# 服务的两种运行方式

- **采用不同的运行方式，参数不同**

**通过startService()方法启动服务**

```
Intent intent = new Intent(MainActivity.this, MyService.class);
button1.setOnClickListener(v -> {
    startService(intent); // 通过startService启动服务
});
button2.setOnClickListener(v -> {
    stopService(intent); // 通过stopService关闭服务
});
```

# 服务的两种运行方式

- **采用不同的运行方式，参数不同**

**通过bindService()方法绑定服务**

- bindService(Intent intent, ServiceConnection conn, int flags)
  - **Intent intent**指定了启动的Service
  - **ServiceConnection conn**用于监听Activity与Service的连接情况

    连接成功时回调onServiceConnected()，连接断开时回调onServiceDisconnected()
  - **int flags**指定是否自动创建Service
- unbindService(ServiceConnection conn)
  - **ServiceConnection conn**用于监听Activity与Service的连接情况

# ServiceConnection介绍

## 为什么需要serviceConnection?

- 调用**bindService()**时系统会回调**onBind()**，onBind()方法会返回一个**IBinder对象**，该对象用于Activity与Service之间的通信。
- IBinder对象无法直接返回给Activity，因此**Activity需要ServiceConnection来接收IBinder对象**。

# ServiceConnection介绍

## 为什么需要serviceConnection?

- Activity中创建一个ServiceConnection对象，将该对象传给bindService(Intent intent, ServiceConnection conn, int flags)。
- 当Activity与Service连接成功时，系统会回调ServiceConnection的onServiceConnected(ComponentName name, IBinder service)，其中IBinder service即为Service的onBind()所返回的IBinder对象。
- 所以，Activity可以通过ServiceConnection接收IBinder对象，从而实现Activity与Service的通信。

# ServiceConnection介绍

**如何创建ServiceConnection对象?**

- ServiceConnection是一个**接口**，具有两个回调方法：
  - onServiceConnected(ComponentName name, IBinder service)
    - 当Activity与Service的连接成功时系统回调该方法。
    - 系统回调该方法来传送Service的onBind()所返回的IBinder对象
  - onServiceDisconnected(ComponentName name)
    - 当Activity与Service的连接意外断开时系统会回调该方法，当调用unbindService()解绑服务时，该方法不会被调用。

# ServiceConnection介绍

**如何创建ServiceConnection对象?**

1. 定义一个实现ServiceConnection接口的子类

```
class MyServiceConnection implements ServiceConnection{

    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        myBinder = (MyBinder) service;
    }

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

# ServiceConnection介绍

**如何创建ServiceConnection对象?**

2. 创建一个ServiceConnection对象

```
MyServiceConnection serviceConnection = new MyServiceConnection();
```

3.传递给bindService()、unbindService()

```
button3.setOnClickListener(v -> {
    // 通过bindService绑定服务
    bindService(intent, serviceConnection, BIND_AUTO_CREATE);
});
button5.setOnClickListener(v -> {
    // 通过unbindService解绑服务
    unbindService(serviceConnection);
});
```

# IBinder介绍

## 如何使用IBinder?

- IBinder定义了Activity与Service通信的方式。
- IBinder是一个接口，Binder是实现了IBinder的类。

1. 定义一个继承Binder类的子类

```
public class MyBinder extends Binder {
    private MyService myService;

    public MyBinder(MyService myService) {
        this.myService = myService;
    }

    // 其它自定义方法
}
```

# IBinder介绍

## 如何使用IBinder?

2. 在Activity中通过ServiceConnection接收MyBinder对象

```
private MyBinder myBinder;

class MyServiceConnection implements ServiceConnection{
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        myBinder = (MyBinder) service;
    }

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

# IBinder介绍

## 如何使用IBinder?

3.通过MyBinder对象实现Activity与Service的交互

```
button4.setOnClickListener(v -> {
    // 通过Binder与Service通信
    myBinder.playMusic();
});
```

# 服务的两种运行方式

- **不同的运行方式的适用场景不同**

**通过startService()方法启动服务**

- 操作简单，适用于Activity与Service**无需通信**的场景。

**通过bindService()方法绑定服务**

- 适用于Activity与Service**需要交互、通信**的场景。

# 练习：通过服务播放音乐

采用方式一

- 启动服务：startService
- 关闭服务：stopService

采用方式二

- 绑定服务：bindService
- 与Service通信：播放/暂停音乐
- 解绑服务：unbindService

# 练习：查看服务的生命周期

- 方式一：startService()+ stopService()

```
14:04:52.226 myTAG  I  onCreate:
14:04:52.240 myTAG  I  onStartCommand:
14:05:02.096 myTAG  I  onStartCommand:
14:05:22.852 myTAG  I  onStartCommand:
14:05:26.957 myTAG  I  onDestroy:
```

- 用户初次调用startService时，系统会依次回调onCreate和onStartCommand
- 用户在服务已启动的情况下调用startService时，系统会回调onStartCommand
- 用户在服务已启动的情况下调用stopService时，系统会回调onDestroy

# 练习：查看服务的生命周期

- 方式二：bindService() + unbindService()

```
14:14:05.626 myService  I  onCreate:
14:14:05.647 myService  I  onBind:
14:14:17.901 myService  I  onUnbind:
14:14:17.904 myService  I  onDestroy:
```

- 用户初次调用bindService时，系统会依次回调onCreate和onBind
- 用户在服务已绑定的情况下调用unbindService时，系统会依次回调onUnbind、onDestroy
- 用户在服务已绑定的情况下调用bindService时，系统不会再调用onCreate或onBind

# 练习：查看Activity与Service的关联

- 方式一：通过startService()启动服务。
  - 退出Activity时**不会回调**Service的onDestroy()

```
14:37:16.704 myService                I  onCreate:
14:37:16.724 myService                I  onStartCommand:
```

- 方式二：通过bindService()绑定服务。
  - 退出Activity时**会回调**Service的onUnbind()、onDestroy()

```
14:39:31.974 myService                I  onCreate:
14:39:31.985 myService                I  onBind:
14:40:18.144 myService                I  onUnbind:
14:40:18.158 myService                I  onDestroy:
```

# 作业

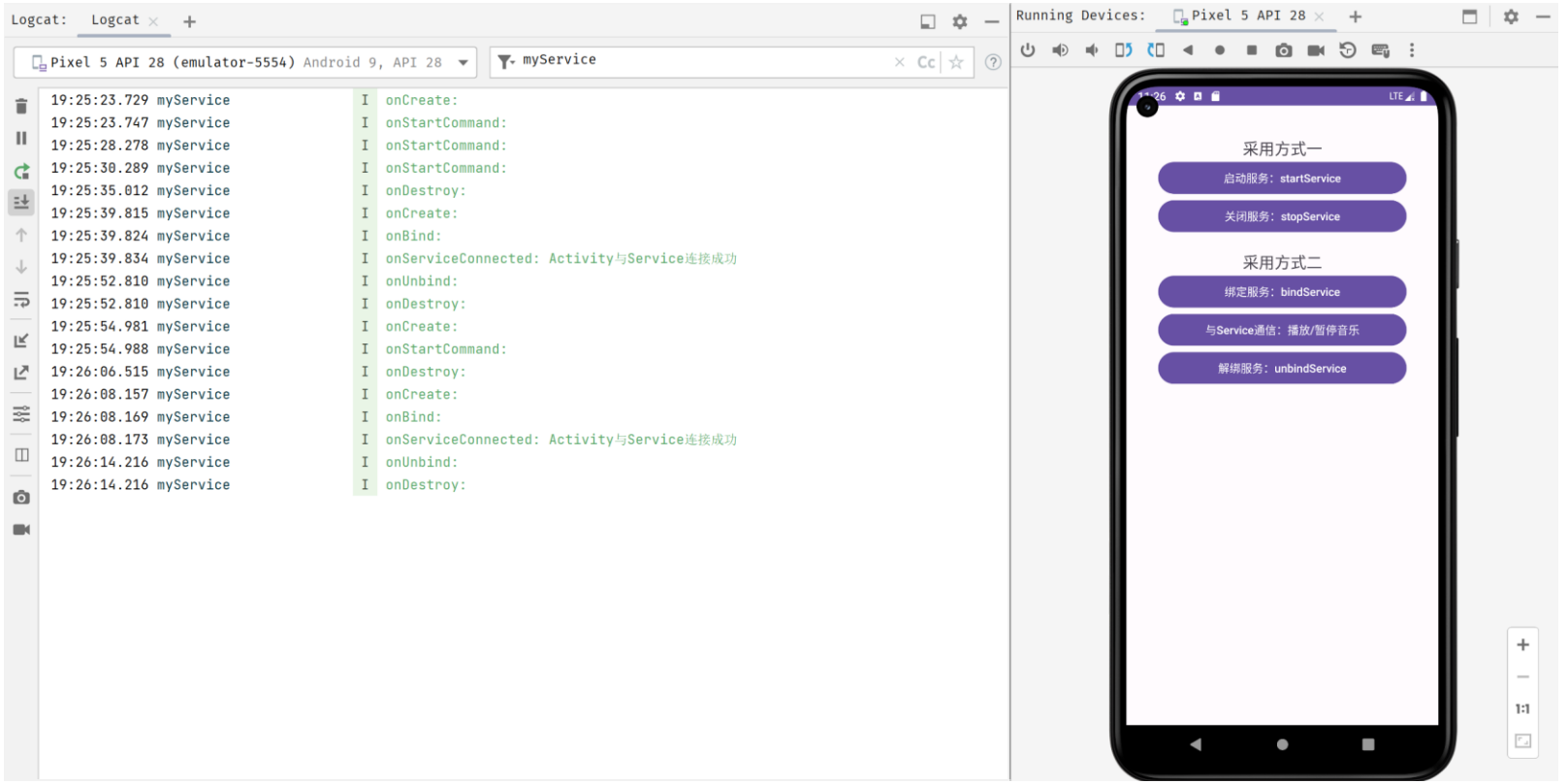
## 本章作业

- 分别使用两种方式启动和关闭服务。
  - 截屏 + 视频记录

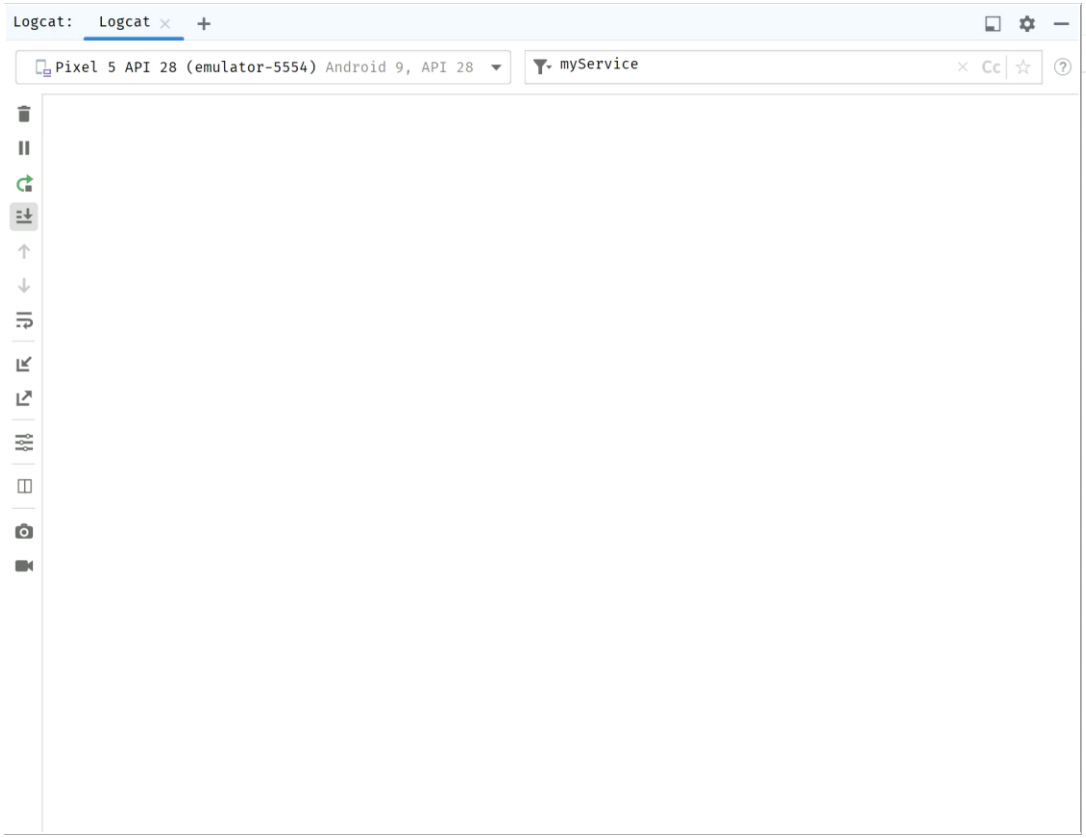
## 作业提交方式

- 视频，文件名：学号+姓名+第9章作业
- 邮件给助教，主题：学号+姓名+第9章作业

# 作业示例


Logcat: Logcat
Pixel 5 API 28 (emulator-5554) Android 9, API 28
myService
19:25:23.729 myService I onCreate:
19:25:23.747 myService I onStartCommand:
19:25:28.278 myService I onStartCommand:
19:25:30.289 myService I onStartCommand:
19:25:35.012 myService I onDestroy:
19:25:39.815 myService I onCreate:
19:25:39.824 myService I onBind:
19:25:39.834 myService I onServiceConnected: Activity与Service连接成功
19:25:52.810 myService I onUnbind:
19:25:52.810 myService I onDestroy:
19:25:54.981 myService I onCreate:
19:25:54.988 myService I onStartCommand:
19:26:06.515 myService I onDestroy:
19:26:08.157 myService I onCreate:
19:26:08.169 myService I onBind:
19:26:08.173 myService I onServiceConnected: Activity与Service连接成功
19:26:14.216 myService I onUnbind:
19:26:14.216 myService I onDestroy:
Running Devices: Pixel 5 API 28
采用方式一
启动服务：startService
关闭服务：stopService
采用方式二
绑定服务：bindService
与Service通信：播放/暂停音乐
解绑服务：unbindService

# 作业示例


Logcat:
Logcat
Pixel 5 API 28 (emulator-5554) Android 9, API 28
myService


Running Devices:
Pixel 5 API 28
采用方式一
启动服务：startService
关闭服务：stopService
采用方式二
绑定服务：bindService
与Service通信：播放/暂停音乐
解绑服务：unbindService