# 活动与服务进行通信

## 1. 实训目的

（1）掌握定义一个服务的方法；

（2）掌握启动和停止服务的方法；

（3）掌握活动和服务进行通信的方法。

## 2. 实训要求

（1）在项目中定义一个服务，提供模拟下载的方法；

（2）在活动中添加两个按钮，一个用于绑定服务，可以与服务进行通信，调用服务中的方法，一个用于取消绑定服务；

（3）完成实验报告。

## 3. 实训指导

（1）定义一个服务

新建一个 ServiceTest 项目，然后在这个项目中新增一个名为 MyService 的类，并让它继承自 Service，完成后的代码如下所示：

```
public class MyService extends Service {
@Override
public IBinder onBind(Intent intent) {
return null;
}}
```

我们希望在 MyService 里提供一个下载功能，然后在活动中可以决定何时开始下载，以及随时查看下载进度。实现这个功能的思路是创建一个专门的 Binder 对象来对下载功能进行管理，修改 MyService 中的代码，如下所示：

```
public class MyService extends Service {
private DownloadBinder mBinder = new DownloadBinder();
class DownloadBinder extends Binder {
    public void startDownload() {
Log.d("MyService", "startDownload executed");
}
  public int getProgress() {
Log.d("MyService", "getProgress executed");
```

```
    return 0;
    }
}
    @Override
    public IBinder onBind(Intent intent) {
    return mBinder;
    }
    ……
    }
```

可以看到,这里我们新建了一个 DownloadBinder 类,并让它继承自 Binder,然后在它的内部提供了开始下载以及查看下载进度的方法。当然这只是两个模拟方法，并没有实现真正的功能，我们在这两个方法中分别打印了一行日志。

接着，在 MyService 中创建了 DownloadBinder 的实例，然后在 onBind()方法里返回了这个实例，这样 MyService 中的工作就全部完成了。

（2）修改布局文件

下面就来学习活动中如何去调用服务里的这些方法。首先需要在布局文件里新增两个按钮，修改 activity_main.xml 中的代码，如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
……
<Button android:id="@+id/bind_service"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Bind Service" />
<Button android:id="@+id/unbind_service"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Unbind Service" />
</LinearLayout>
```

这两个按钮分别是用于绑定服务和取消绑定服务的,当一个活动和服务绑定了之后，就可以调用该服务里的 Binder 提供的方法了。

（3）修改 MainActivity 中的代码

```
public class MainActivity extends Activity implements OnClickListener {
private Button startService;
private Button stopService;
private Button bindService;
private Button unbindService;
private MyService.DownloadBinder downloadBinder;
private ServiceConnection connection = new ServiceConnection() {
@Override
public void onServiceDisconnected(ComponentName name) {
}
@Override
public void onServiceConnected(ComponentName name, IBinder service) {
downloadBinder = (MyService.DownloadBinder) service;
downloadBinder.startDownload();
downloadBinder.getProgress();
}};
  @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
……
bindService = (Button) findViewById(R.id.bind_service);
unbindService = (Button) findViewById(R.id.unbind_service);
bindService.setOnClickListener(this);
unbindService.setOnClickListener(this);
}
@Override
public void onClick(View v) {
switch (v.getId()) {
……
 //绑定服务
    case R.id.bind_service:
       Intent bindIntent=new Intent(this,MyService.class);
       //绑定服务，bindService()方法有三个参数，第一个为 Intent 实例，第二个为 ServiceCon
nection 实例，第三个是标志位，BIND_AUTO_CREATE 表示活动和服务绑定后自动创建服务，即执行 onCre
ate()方法。
       bindService(bindIntent,connection,BIND_AUTO_CREATE);
       break;
```

```
            //解除绑定服务
            case R.id.unbind_service:
              unbindService(connection);//解绑服务
            default:
}
    } }
```

可以看到，这里我们首先创建了一个 ServiceConnection 的匿名类，在里面重写了 onServiceConnected()方法和 onServiceDisconnected()方法，这两个方法分别会在活动与服务成功绑定以及解除绑定的时候调用。在 onServiceConnected()方法中，我们又通过向下转型得到了 DownloadBinder 的实例，有了这个实例，活动和服务之间的关系就变得非常紧密了。现在我们可以在活动中根据具体的场景来调用 DownloadBinder 中的任何 public 方法，即实现了指挥服务干什么，服务就去干什么的功能。这里仍然只是做了个简单的测试，在 onServiceConnected()方法中调用了 DownloadBinder 的 startDownload()和 getProgress()方法。

活动和服务的绑定功能是在 Bind Service 按钮的点击事件里完成的。可以看到，这里我们仍然是构建出了一个 Intent 对象，然后调用 bindService()方法将 MainActivity 和 MyService 进行绑定。解除活动和服务之间的绑定需要调用一下 unbindService()方法，这也是 Unbind Service 按钮的点击事件里实现的功能。