Binder系列8—如何使用AIDL

Nov 23, 2015

> 自定义binder架构的 client/ server组件

一、AIDL

1.1 Server端

RemoteService.java

本例是为了演示进程间的通信机制，故需要将Service与Activity处于不同的进程，需要在AndroidManifest.xml中，把service配置成 `android:process=":remote"` ,进程也可以命名成其他的。

```
public class RemoteService extends Service {
    private static final String TAG = "BinderSimple";

    MyData mMyData;

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, "[RemoteService] onCreate");
        initMyData();
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.i(TAG,"[RemoteService] onBind");
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {
        Log.i(TAG, "[RemoteService] onUnbind");
        return super.onUnbind(intent);
    }

    @Override
    public void onDestroy() {
        Log.i(TAG, "[RemoteService] onDestroy");
        super.onDestroy();
    }

    /**
     * 实现IRemoteService.aidl中定义的方法
     */
    private final IRemoteService.Stub mBinder = new IRemoteService.Stu
b() {

        @Override
        public int getPid() throws RemoteException {
            Log.i(TAG,"[RemoteService] getPid()="+android.os.Process.m
yPid());
            return android.os.Process.myPid();
        }

        @Override
        public MyData getMyData() throws RemoteException {
            Log.i(TAG,"[RemoteService] getMyData()  "+ mMyData.toStrin
g());
            return mMyData;
```

```
        }

        /**此处可用于权限拦截**/
        @Override
        public boolean onTransact(int code, Parcel data, Parcel reply,
int flags) throws RemoteException {
            return super.onTransact(code, data, reply, flags);
        }
    };

    /**
     * 初始化MyData数据
     **/
    private void initMyData() {
        mMyData = new MyData();
        mMyData.setData1(10);
        mMyData.setData2(20);
    }
}
```

1.2 Client端

ClientActivity.java

```
public class ClientActivity extends AppCompatActivity {
    private static final String TAG = "BinderSimple";

    private IRemoteService mRemoteService;

    private boolean mIsBound;
    private TextView mCallBackTv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i(TAG, "[ClientActivity] onCreate");

        setContentView(R.layout.activity_main);

        mCallBackTv = (TextView) findViewById(R.id.tv_callback);
        mCallBackTv.setText(R.string.remote_service_unattached);
    }

    /**
     * 用语监控远程服务连接的状态
     */
    private ServiceConnection mConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder ser
vice) {

            mRemoteService = IRemoteService.Stub.asInterface(service);
            String pidInfo = null;
            try {
                MyData myData = mRemoteService.getMyData();
                pidInfo = "pid="+ mRemoteService.getPid() +
                        ", data1 = "+ myData.getData1() +
                        ", data2="+ myData.getData2();
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            Log.i(TAG, "[ClientActivity] onServiceConnected  "+pidInf
o);
            mCallBackTv.setText(pidInfo);
            Toast.makeText(ClientActivity.this, R.string.remote_servic
e_connected, Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            Log.i(TAG, "[ClientActivity] onServiceDisconnected");
            mCallBackTv.setText(R.string.remote_service_disconnected);
```

```
            mRemoteService = null;
            Toast.makeText(ClientActivity.this, R.string.remote_servic
e_disconnected, Toast.LENGTH_SHORT).show();
        }
    };


    public void clickHandler(View view){
        switch (view.getId()){
            case R.id.btn_bind:
                bindRemoteService();
                break;

            case R.id.btn_unbind:
                unbindRemoteService();
                break;

            case R.id.btn_kill:
                killRemoteService();
                break;
        }
    }

    /**
     * 绑定远程服务
     */
    private void bindRemoteService(){
        Log.i(TAG, "[ClientActivity] bindRemoteService");
        Intent intent = new Intent(ClientActivity.this, RemoteServic
e.class);
        intent.setAction(IRemoteService.class.getName());
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);

        mIsBound = true;
        mCallBackTv.setText(R.string.binding);
    }

    /**
     * 解除绑定远程服务
     */
    private void unbindRemoteService(){
        if(!mIsBound){
            return;
        }
        Log.i(TAG, "[ClientActivity] unbindRemoteService ==>");
        unbindService(mConnection);
        mIsBound = false;
        mCallBackTv.setText(R.string.unbinding);
    }
```

```
    /**
     * 杀死远程服务
     */
    private void killRemoteService(){
        Log.i(TAG, "[ClientActivity] killRemoteService");
        try {
            android.os.Process.killProcess(mRemoteService.getPid());
            mCallBackTv.setText(R.string.kill_success);
        } catch (RemoteException e) {
            e.printStackTrace();
            Toast.makeText(ClientActivity.this, R.string.kill_failure,
Toast.LENGTH_SHORT).show();
        }
    }
}
```

1.3 AIDL文件

(1)IRemoteService.aidl
定义远程通信的接口方法

```
interface IRemoteService {
    int getPid();
    MyData getMyData();
}
```

(2)MyData.aidl
定义远程通信的自定义数据

```
parcelable MyData;
```

1.4 Parcel数据

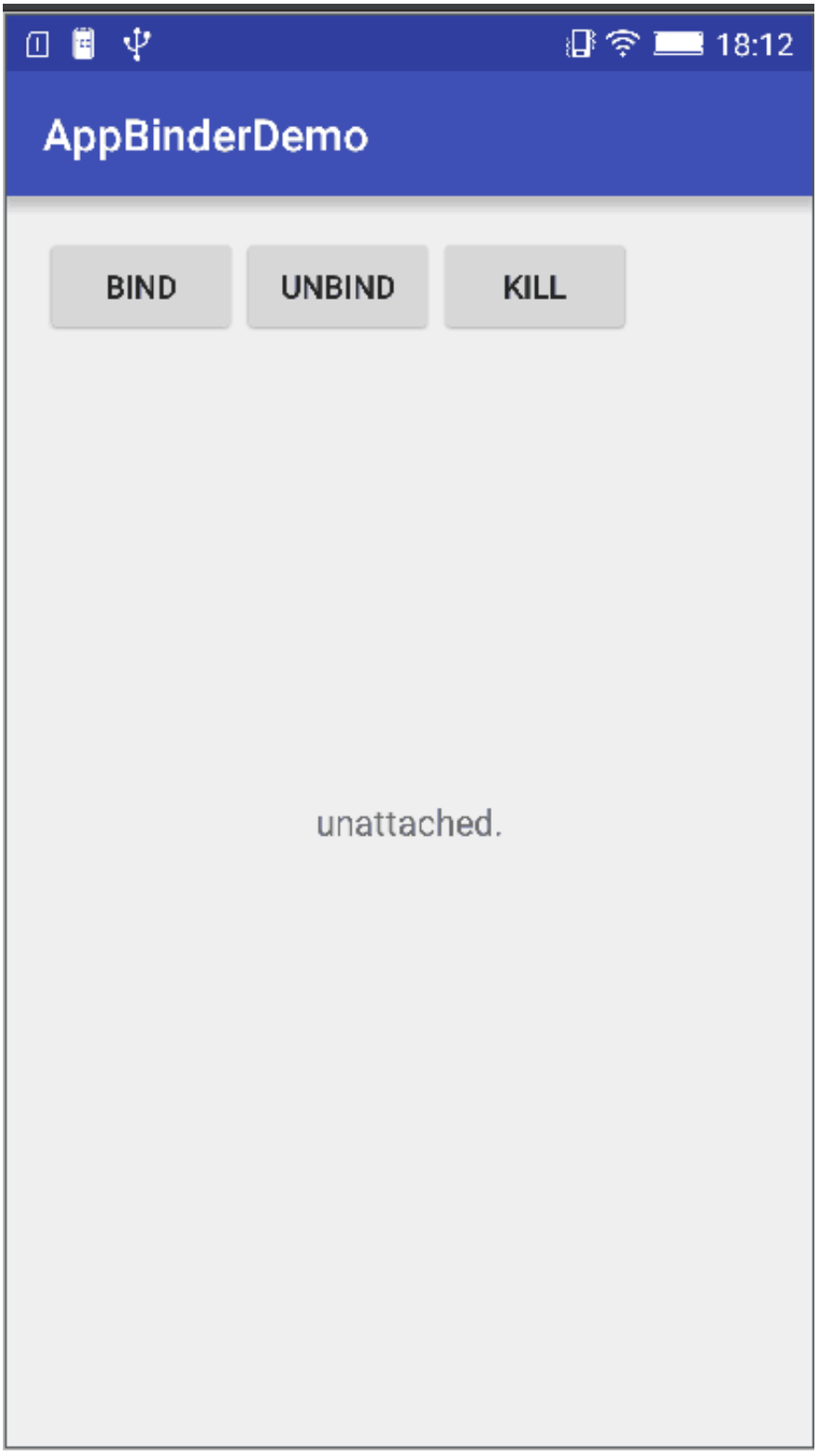
MyData.java

```
public class MyData implements Parcelable {
    private int data1;
    private int data2;

    public MyData(){

    }

    protected MyData(Parcel in) {
        readFromParcel(in);
    }

    public static final Creator<MyData> CREATOR = new Creator<MyData>
() {
        @Override
        public MyData createFromParcel(Parcel in) {
            return new MyData(in);
        }

        @Override
        public MyData[] newArray(int size) {
            return new MyData[size];
        }
    };

    @Override
    public int describeContents() {
        return 0;
    }

    /** 将数据写入到Parcel **/
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(data1);
        dest.writeInt(data2);
    }

    /** 从Parcel中读取数据 **/
    public void readFromParcel(Parcel in){
        data1 = in.readInt();
        data2 = in.readInt();
    }


    public int getData2() {
        return data2;
    }
```

```
    public void setData2(int data2) {
        this.data2 = data2;
    }

    public int getData1() {
        return data1;
    }

    public void setData1(int data1) {
        this.data1 = data1;
    }

    @Override
    public String toString() {
        return "data1 = "+ data1 + ", data2="+ data2;
    }
}
```

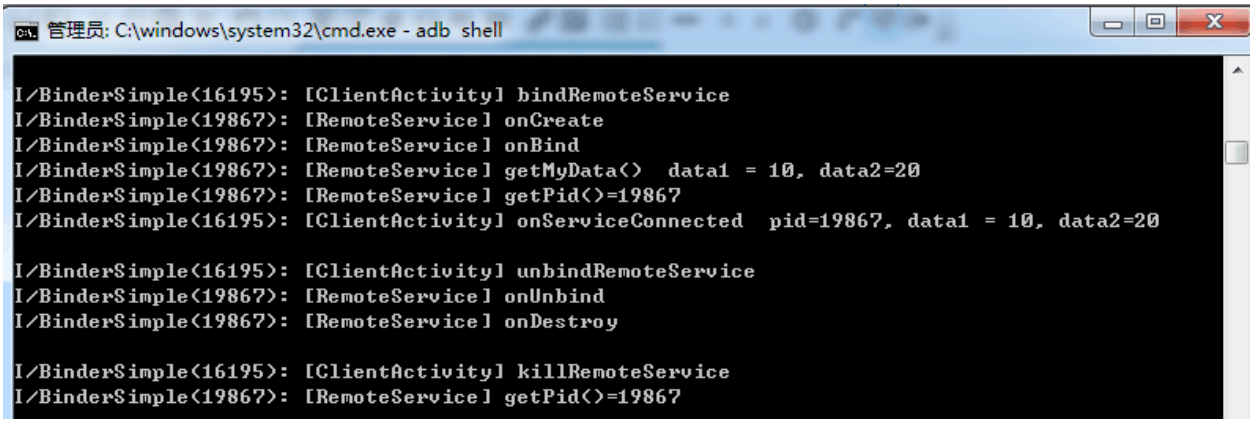
1.5 运行

该工程会生成一个apk，安装到手机，打开apk，界面如下：



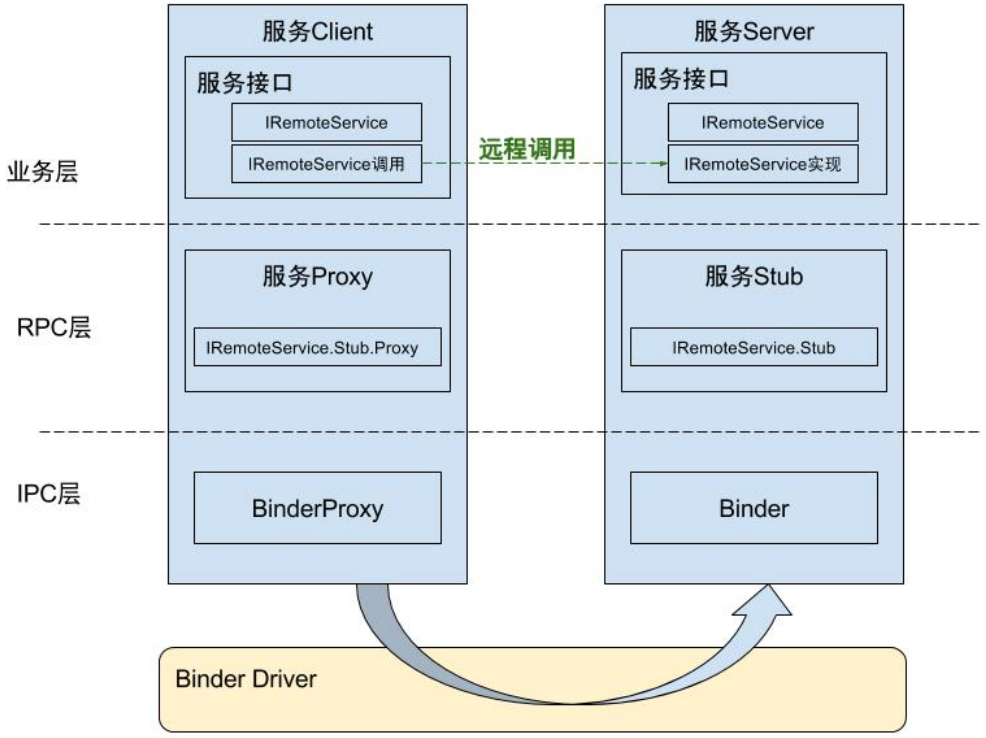
界面上有三个按钮，分别是功能分别是bindService(绑定Service), unbindService(解除绑定Service), killProcess(杀死Service进程)。

从左往右，依次点击界面，可得：



二、原理分析

调用图：



采用AIDL技术，是原理还是利用framework binder的架构。本文的实例AIDL会自动生成一个与之相对应的IRemoteService.java文件，如下：

```
package com.yuanhh.appbinderdemo;
public interface IRemoteService extends android.os.IInterface {
    /**
     * Local-side IPC implementation stub class.
     */
    public static abstract class Stub extends android.os.Binder implem
ents com.yuanhh.appbinderdemo.IRemoteService {
        private static final java.lang.String DESCRIPTOR = "com.yuanh
h.appbinderdemo.IRemoteService";

        /**
         * Stub构造函数
         */
        public Stub() {
            this.attachInterface(this, DESCRIPTOR);
        }

        /**
         * 将IBinder 转换为IRemoteService interface
         */
        public static com.yuanhh.appbinderdemo.IRemoteService asInterf
ace(android.os.IBinder obj) {
            if ((obj == null)) {
                return null;
            }
            android.os.IInterface iin = obj.queryLocalInterface(DESCRI
PTOR);
            if (((iin != null) && (iin instanceof com.yuanhh.appbinder
demo.IRemoteService))) {
                return ((com.yuanhh.appbinderdemo.IRemoteService) ii
n);
            }
            return new com.yuanhh.appbinderdemo.IRemoteService.Stub.Pr
oxy(obj);
        }

        @Override
        public android.os.IBinder asBinder() {
            return this;
        }

        @Override
        public boolean onTransact(int code, android.os.Parcel data, an
droid.os.Parcel reply, int flags) throws android.os.RemoteException {
            switch (code) {
                case INTERFACE_TRANSACTION: {
                    reply.writeString(DESCRIPTOR);
                    return true;
```

```
                }
                case TRANSACTION_getPid: {
                    data.enforceInterface(DESCRIPTOR);
                    int _result = this.getPid();
                    reply.writeNoException();
                    reply.writeInt(_result);
                    return true;
                }
                case TRANSACTION_getMyData: {
                    data.enforceInterface(DESCRIPTOR);
                    com.yuanhh.appbinderdemo.MyData _result = this.get
MyData();
                    reply.writeNoException();
                    if ((_result != null)) {
                        reply.writeInt(1);
                        _result.writeToParcel(reply, android.os.Parcel
able.PARCELABLE_WRITE_RETURN_VALUE);
                    } else {
                        reply.writeInt(0);
                    }
                    return true;
                }
            }
            return super.onTransact(code, data, reply, flags);
        }

        private static class Proxy implements com.yuanhh.appbinderdem
o.IRemoteService {
            private android.os.IBinder mRemote;

            /**
             * Proxy构造函数
             */
            Proxy(android.os.IBinder remote) {
                mRemote = remote;
            }

            @Override
            public android.os.IBinder asBinder() {
                return mRemote;
            }

            public java.lang.String getInterfaceDescriptor() {
                return DESCRIPTOR;
            }

            @Override
            public int getPid() throws android.os.RemoteException {
                android.os.Parcel _data = android.os.Parcel.obtain();
```

```
                android.os.Parcel _reply = android.os.Parcel.obtain();
                int _result;
                try {
                    _data.writeInterfaceToken(DESCRIPTOR);
                    mRemote.transact(Stub.TRANSACTION_getPid, _data,
_reply, 0);
                    _reply.readException();
                    _result = _reply.readInt();
                } finally {
                    _reply.recycle();
                    _data.recycle();
                }
                return _result;
            }

            @Override
            public com.yuanhh.appbinderdemo.MyData getMyData() throws
android.os.RemoteException {
                android.os.Parcel _data = android.os.Parcel.obtain();
                android.os.Parcel _reply = android.os.Parcel.obtain();
                com.yuanhh.appbinderdemo.MyData _result;
                try {
                    _data.writeInterfaceToken(DESCRIPTOR);
                    mRemote.transact(Stub.TRANSACTION_getMyData, _dat
a, _reply, 0);
                    _reply.readException();
                    if ((0 != _reply.readInt())) {
                        _result = com.yuanhh.appbinderdemo.MyData.CREA
TOR.createFromParcel(_reply);
                    } else {
                        _result = null;
                    }
                } finally {
                    _reply.recycle();
                    _data.recycle();
                }
                return _result;
            }
        }

        static final int TRANSACTION_getPid = (android.os.IBinder.FIRS
T_CALL_TRANSACTION + 0);
        static final int TRANSACTION_getMyData = (android.os.IBinder.F
IRST_CALL_TRANSACTION + 1);
    }

    public int getPid() throws android.os.RemoteException;

    public com.yuanhh.appbinderdemo.MyData getMyData() throws androi
```

```
d.os.RemoteException;
}
```

参考

- http://developer.android.com/intl/zh-cn/guide/components/aidl.html (http://developer.android.com/intl/zh-cn/guide/components/aidl.html)

gityuan@gmail.com (mailto:gityuan@gmail.com) · Github (https://github.com/yuanhuihui) · 天道酬勤 · © 2015 Yuanhh · Jekyll (https://github.com/jekyll/jekyll) theme by HyG (https://github.com/Gaohaoyang)