Phát triển ứng dụng Smartphone

Tài liệu lưu hành nội bộ

Đây là tài liệu tham khảo sử dụng trong môn học Lập trình ứng dụng Smartphone – Android được tổng hợp, biên soạn từ nhiều nguồn bởi các thành viên của Nhóm nghiên cứu và ứng dụng công nghệ A106-Đại học Hoa Sen.

Nhóm A106-http://profiles.hoasen.edu.vn/groups/584/

Phát triển ứng dụng Smartphone

Phần 06: Service

Lê Đức Huy

Email: leduchuy89vn@gmail.com

Mục lục

1 Service trên Android

Service là một thành phần của ứng dụng Android dùng để thực thi một tác vụ ngầm bên dưới nền và không có giao diện hiển thị nội dung. Service cũng giống như các thành phần khác của ứng dụng(Activity, BroadcastReceiver…), nó sẽ chạy trên luồng chính của tiến trình mà ứng dụng đang chạy trên đó. Điều này có nghĩa là nếu bạn cần thực hiện một công việc nào đó tốn nhiều thời gian như chơi nhạc, tải dữ liệu từ trên mạng về thì bạn phải đưa công việc đó vào một luồng riêng để thực thi. Việc này sẽ tránh cho các công việc đang thực thi trên luồng chính không bị gián đoạn. Chúng ta cần xác định rỏ các đặc trưng của Service:

- Một Service không phải là một tiến trình tách biệt. Một đối tượng Service không hề chạy trên tiến trình của riêng nó mà nó chạy trên tiến trình của ứng dụng.
- Một Service không phải là một luồng. Điều này có nghĩa là mọi công việc sẽ được luồng chính thực thi. Chính vì vậy một đối tượng Service thường định nghĩa một luồng của riêng nó để thực hiện các công việc nhằm tránh tình trạng gián đoạn các công việc đang thực thi ở luồng chính.

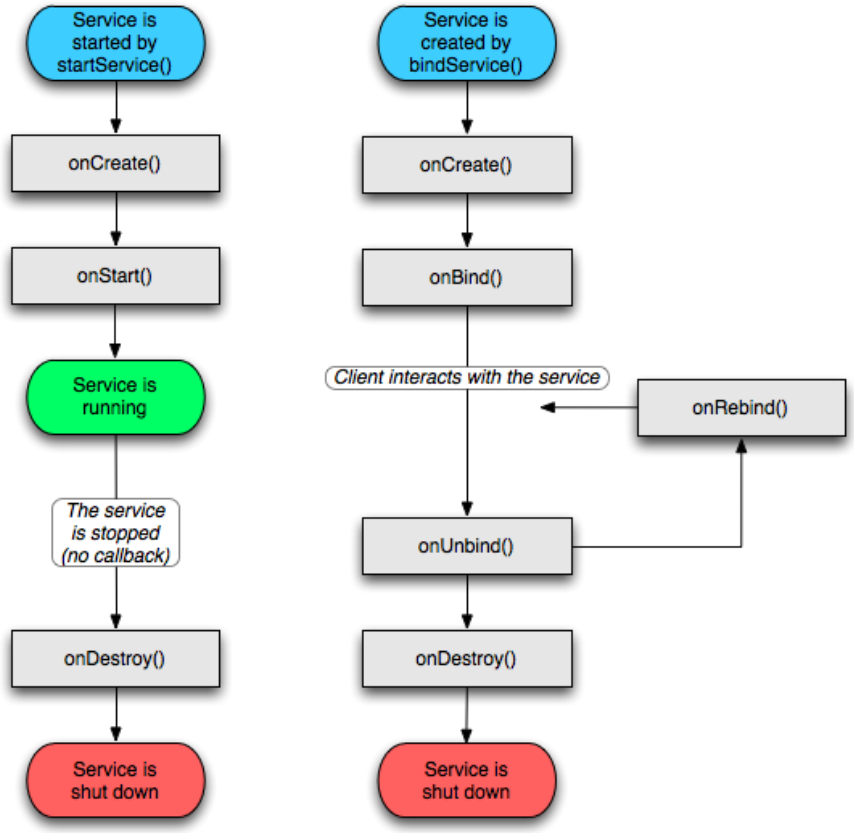
Một Service có thể được sử dụng theo hai cách

- Một Service có thể được dùng để thực thi một công việc dưới nền mà không cần hiển thị giao diện người dùng. Loại service này được bắt đầu và được cho phép hoạt động cho đến khi một người nào đó dừng nó lại hoặc nó tự ngắt. Ở chế độ này, nó được bắt đầu bằng cách gọi **Context.startService()** và dừng bằng lệnh **Context.stopService()**. Một Service có thể tự ngắt bằng lệnh **Service.stopSelf()** hoặc **Service.stopSelfResult()**. Mỗi Service chỉ có một thể hiện duy nhất, do đó chỉ cần một lệnh stopService() để ngừng một Service lại cho dù lệnh startService() được gọi ra bao nhiêu lần.
- Một Service còn có thể được sử dụng để cung cấp một tính năng nào đó cho ứng dụng khác kết nối và sử dụng. Một ứng dụng có thể thiết lập một đường truyền tới đối tượng Service và sử dụng đường kết nối đó để điều khiểnService. Kết nối này được thiết lập bằng cách gọi lệnh **Context.bindService()** và được đóng lại bằng cách gọi lệnh **Context.unbindService()**. Nhiều ứng dụng có thể kết nối tới cùng một đối tượng Service. Nếu Service được một ứng dụng khách kết nối đến vẫn chưa được khởi chạy thì lệnh bindService() có thể tùy ý khởi chạy nó.

Hai chế độ này thì không tách biệt toàn bộ. Bạn có thể kết nối với một Service mà nó đã được bắt đầu với lệnh **startService()**.

Ví dụ: Một Service nghe nhạc ở chế độ nền có thể được bắt đầu bằng cách gọi lệnh startService() cùng với một đối tượng Intent xác định bài hát cần chơi. Sau đó, có thể người sử dụng muốn kiểm soát trình chơi nhạc hoặc biết thêm thông tin về bài hát hiện tại đang chơi, thì sẽ có một Activity tạo lập một đường truyền tới Service bằng cách gọi bindService(). Trong trường hợp này, stopService() sẽ không thực sự ngừng Service cho đến khi liên kết cuối cùng được đóng lại.

Sơ đồ dưới đây thể hiện vòng đời của Service:



.

Dựa vào lược đồ trên ta có hai vòng lặp quan trọng trong vòng đời của một đối tượng Service cục bộ:

- Vòng đời toàn diện (**entire lifetime**):Bắt đầu từ lúc gọi phương thức onCreate() đến lúc gọi phương thức onDestroy(). Cũng giống như Activity, đối tượng Service sẽ khởi tạo các giá trị tại phương thức onCreate() và dọn dẹp bộ nhớ tại phương thức onDestroy().

  VD:Một đối tượng Service dùng để chơi nhạc sẽ tạo một luồng để chơi nhạc ở phương thức onCreate() và ngưng luồng chơi nhạc này ở phương thức onDestroy() cũng như dọn dẹp bộ nhớ trước khi hủy đối tượng.

- Vòng đời thực thi(**active lifetime**): Bắt đầu từ lúc gọi phương thức onStart().

  VD: Phương thức onStart(…) sẽ đón đối tượng Intent được gởi đi khi khởi động Service bằng phương thức startService(…). Với đối tượng Service dùng để chơi nhạc ta sẽ gởi tên bài hát cần phát trong đối tượng Intent.

Nếu một đối tượng Service cho phép một ứng dụng khác kết nối đến nó thì nó phải cài đặt 03 phương thức sau:

- IBinder onBind(Intent *intent*): Khi một đối tượng muốn tạo kết nối đến một đối tượng Servicer thì nó sẽ gọi phương thức Context.bindService(…) và gởi đi một đối tượng

Intent. Phương thức onBind() sẽ được gọi để sử lý yêu cầu kết nối này. Nó sẽ trả về các kênh giao tiếp mà đối tượng cần kết nối có thể sử dụng để tương tác với Service.

- boolean onUnbind(Intent *intent*):Phương thức này tương tự như phương thức onBind(). Tuy nhiên nó sẽ được gọi khi có một đối tượng gọi phương thức Context.unbindService() để ngắt kết nối với Service. Lúc này phương thức onUnbind() sẽ được gọi để sử lý yêu cầu ngắt kết nối đến Service.
- void onRebind(Intent *intent*):Phương thức này được gọi khi có một đối tượng khách mới muốn kết nối đến Service.

Nghe có vẻ thật phức tạp nên ta sẽ tiến hành làm hai ví dụ về cách sử dụng local service và remote service để hiểu rỏ cách thức hoạt động của chúng.

1.1 Local Service

Tạo project với các thông số như sau:

Project name: LocalServiceExample

Build target: Android 2.3.3.

Application name: LocalServiceExample

Package name: niit.android

Create Activity: main

Thêm tệp BackgroundService.java với nội dung như sau:

```
package niit.android;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

publicclass BackgroundService extends Service{

     private NotificationManager notificationMgr;
     private   ServiceThread serviceThread;

     @Override
     publicvoid onCreate() {
          super.onCreate();
```

```
	notificationMgr
=(NotificationManager)getSystemService(NOTIFICATION_SERVICE);

	displayNotificationMessage("Starting Background Service");

	serviceThread = new ServiceThread();
	serviceThread.start();

	}
	@Override
	publicvoid onStart(Intent intent, int startId) {
	super.onStart(intent, startId);

	}

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

	@Override
	publicvoid onDestroy()
	{
	displayNotificationMessage("Stopping Background Service");
	super.onDestroy();

	}

	privatevoid displayNotificationMessage(String message)
	{
	Notification notification = new
Notification(R.drawable.icon,message,System.currentTimeMillis())
;

	PendingIntent contentIntent =
PendingIntent.getActivity(this, 0, new Intent(this, main.class),
0);

	notification.setLatestEventInfo(this, "Background
Service",message,contentIntent);
	notificationMgr.notify(123456, notification);
	}

	class ServiceThread extends Thread
	{
		@Override
		publicvoid run() {
```

```
                    // Thực thi các tác vụ mong muốn.
               }
          }
}
```

Thay đổi tệp main.xml với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<!-- This file is /res/layout/main.xml -->
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/a
ndroid"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:gravity="center_horizontal"
>
     <Button
          android:id="@+id/startBtn"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Start BackgroupService"
/>

<Button
     android:id="@+id/stopBtn"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Stop BackgroupService"
/>
</LinearLayout>
```

Thay đổi tệp main.java với nội dung như sau:

```
package niit.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

publicclass main extends Activity implements OnClickListener{
@Override
publicvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button bindBtn = (Button)findViewById(R.id.startBtn);
```

```
        bindBtn.setOnClickListener(this);

        Button unbindBtn = (Button)findViewById(R.id.stopBtn);
        unbindBtn.setOnClickListener(this);

    }

     @Override
     publicvoid onClick(View v) {
          switch (v.getId()) {
               case R.id.startBtn:
               {
                    Intent intent = new
Intent(this,BackgroundService.class);
                    startService(intent);
               }
                    break;
               case R.id.stopBtn:
               {
                    Intent intent = new
Intent(this,BackgroundService.class);
                    stopService(intent);
               }
                    break;

               default:
                    break;
          }
     }
}
```

Bổ sung khai báo <service> trên tệp AndroidManifest.xml với nội dung như sau:

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/andro
id"
     package="niit.android"
     android:versionCode="1"
     android:versionName="1.0">
     <applicationandroid:icon="@drawable/icon"android:label="@st
ring/app_name">
          <activity
               android:name=".main"
               android:label="@string/app_name">
               <intent-filter>
                    <actionandroid:name="android.intent.action.M
               AIN"/>
```

```
                    <categoryandroid:name="android.intent.catego
                ry.LAUNCHER"/>
                </intent-filter>
            </activity>
            <serviceandroid:name="BackgroundService"/>
        </application>
    </manifest>
```

Lớp BackgroundService là một lớp mở rộng lớp Service của Android. Trên lớp này ta tiến hành ghi đè các phương thức onCreate(), onStart(), onBind() để sử lý các sự kiện liên quan đến vòng đời của Service. Như đã trình bày ở phần định nghĩa, một Service chạy trên luồng chính của tiến trình mà ứng dụng đang chạy. Do đó ta cần xây dựng thêm một luồng để thực thi các công việc mong muốn nếu không muốn các tác vụ này làm cản trở công việc đang thực thi ở luồng chính. Do đó, trên lớp BackgroundService ta định lớpServiceThread mở rộng Thread của Android để thực thi các tác vụ mong muốn. Ở đây phương thức run của ServiceThread để trống, bạn có thể bổ sung các câu lệnh thực thi một tác vụ nào đó như: chơi nhạc, tải dữ liệu… cần thực thi.Ta sẽ tiến hành phân tích lớp BackgroundService để thấy rỏ cách thức hoạt động của nó.

Trên hệ điều hành Android hổ trợ một khái niệm Notification cho phép một ứng dụng gởi một thông báo đến người dùng thông qua một đoạn text hiển thị trên thanh trạng thái của thiết bị.

VD: Hình sau thể hiện một thông báo được đưa ra khi có tin nhắn được gởi đến thiết bị:

Ta sẽ sử dụng hình thức thông báo này để hiện các thông báo về sự thay đổi trạng thái của BackgroupService. Để có thể đưa một thông báo lên thanh trạng thái ta sẽ sử dụng một đối tượng kiểu NotificationManager. Ta có thể lấy đối tượng này bằng lệnh:

```
notificationMgr=(NotificationManager)getSystemService(NOTIFICATION_SERVICE);
```

Gởi một thông báo lên thanh trạng thái của thiết bị bằng các câu lệnh sau:

```
Notification notification = new
Notification(R.drawable.icon,message,System.currentTimeMillis())
;

PendingIntent contentIntent =  PendingIntent.getActivity(this,
0, new Intent(this, main.class), 0);

notification.setLatestEventInfo(this, "Background
Service",message,contentIntent);
notificationMgr.notify(123456, notification);
```

Đầu tiên ta khởi tạo một đối tượng notification kiểu Notification với ba thông số:

- Biểu tượng hiển thị bên cạnh thông báo.
- Chuỗi kí tự sẽ hiển thị trên thanh trạng thái.
- Thời gian hiển thị chuỗi kí tự.

Tiếp theo ta khởi tạo một đối tượng contentIntent kiểu PendingIntent. Khái niệm này đã được nhắc đến ở phần Network của tài liệu này. Đối tượng PendingIntent chỉ một hành động sẽ được thực thi ở một thời điểm trong tương lai. Đối tượng PendingIntent được tạo bằng phương thức PendingIntent.getActivity()dùng tạo một PendingIntent dùng để khởi động một Activity. Activity này sẽ được khởi động bằng đối tượng new Intent(this, main.class) chỉ việc sẽ khởi động Activity tên main. Bạn có thể thay đổi tên Activity phù hợp với bản thân.

Mỗi loại thông báo (Thông báo tin nhắn mới, mail mới…) có thể có nhiều thông báo tạo các thời điểm khác nhau. Tuy nhiên khi hiển thị trên thanh trạng thái để tiết kiệm không gian thì cùng một loại thông báo chỉ hiển thị thông báo mới nhất. Ta sử dụng phương thức Notification.setLatestEventInfo(…) để gán thông báo mới nhất cho đối tượng Notification. Phương thức này nhận vào 03 tham số lần lượt là:

- Đối tượng Context.
- Tiêu đề của thông báo.
- Đối tượng kiểu PendingIntent dùng để thực thi một hành vi khi người dùng nhấn vào thông báo ở trang xem chi tiết thông báo.

Sau đó ta đưa thông báo lên thanh trạng thái bằng cách gọi lệnh:

```
notificationMgr.notify(123456, notification);
```

Phương thức notify() nhận vào hai tham số là id để phân biệt loại (Gởi hai thông báo cùng id đi thì chỉ thông báo sau mới được hiển thị lên thanh trạng thái) thông báo và đối tượng notification chứa thông báo.

Từ đó ta dựng lên phương thức displayNotificationMessage() dùng để hiển thị một thông báo về trạng thái hiện tại của Service lên thanh trạng thái của thiết bị. Ta gọi phương thức này ở onCreate() và onDestroy() để hiện thông báo về trạng thái của Service.

Trên Activity main ta có hai nút nhấn: một để khởi động service và một để dừng service.

1.2 Remote Service

Phần trên ta đã đề cập đến local service như là một dịch vụ làm nhiệm vụ thực thi một tác vụ bên dưới nền và không có giao diện hiển thị nội dung đến người dùng. Do đó, local service được khởi động trên cùng tiến trình với tiến trình của ứng dụng khởi động nó. Vòng đời của một đối tượng local service phụ thuộc vào vòng đời của ứng dụng. Khác với local service, remote service chạy trên tiến trình của riêng nó. Từ đây xuất hiện khó khăn trong việc giao tiếp giữa các tiến trình với nhau. Nếu một tiến trình muốn gởi một đối tượng đến tiến trình khác thì ta phải bắt một cây cầu nối giữa hai tiến trình. Với mục đích này, Android cung cấp khái niệm AIDL (Android Interface Definition Language) hỗ trợ việc bắt cầu nối giữa hai tiến trình cũng như giao tiếp giữa chúng. Ta cần tạo một AIDL dùng để định nghĩa một giao diện lập trình (interface) cung cấp cho ứng dụng khách. Bằng cách nhận lấy giao diện lập trình này, ứng dụng khách sẽ biết được các phương thức mà remote service cung cấp. Khi ứng dụng khách kết nối đến remote service, remote service sẽ trả về một đối tượng đã cài đặt giao diện lập trình kể trên, ứng dụng khách sẽ dùng đối tượng này thực thi các phương thức cần thiết. Ta sẽ lần lượt tiến hành định nghĩa remote service và ứng dụng khách để thấy được cách hoạt động của remote service:

1.2.1 Hiện thực remote service

Để hiện thực một remote service ta thực hiện các bước sau:

1. Viết một tệp tin AIDL dùng để khai báo giao diện lập trình(interface) cho ứng dụng khách. Một tệp tin AIDL sử dụng cú phá java và có phần mở rộng aidl. Sử dụng cùng package với package của ứng dụng.
2. Tạo mới một project Android và chép tệp .aidl vào thư mục scr. Eclise sẽ tự động gọi trình biên dịch AIDL để biên dịch tệp tin .aidl và phát sinh ra một giao diện lập trình (interface).
3. Cài đặt một Service và trả về một đối tượng cài đặt giao diện lập trình kể trên trong phương thức onBind().
4. Bổ xung khai báo service trong tệp AndroidManifest.xml của project.

1.2.1.1 Tạo project Android

Tạo mới một project với các thông số sau:

Project name: RemoteServiceExample

Build target: Android 2.3.3.

Application name: RemoteServiceExample

Package name: niit.android.remoteservice

1.2.1.2 Tạo AIDL

Soạn tệp tin IStockQuoteService.aidl với nội dung như sau:

```
package niit.android.remoteservice;
interface IStockQuoteService
{
   double getQuote(String ticker);
}
```

Ghi chú: Package của tệp tin kể trên chính là package của project mà ta sẽ định nghĩa Service.

Tệp tin này khai báo một giao diện lập trình tên là IstockQuoteService chỉ có một phương thức duy nhất là double getQuote(String ticker). Phương thức này sẽ sẽ nhận vào tên mã chứng khoán và trả về giá trị hiện tại của nó.

Kéo thả tệp tin IstockQuoteService.aidl bỏ vào thư mục src của project hiện tại. Sau khi kéo thả vào project, Eclipse sẽ tự động gọi trình biên dịch AIDL để biên dịch tệp tin trên và tạo ra một giao diện lập trình tên là IStockQuoteService đặt trong thư mục gen/[package] của project. Trong giao diện IstockQuoteService có một lớp tên là Stub extends android.os.Binder và cài đặt giao diện lập trình IstockQuoteService. Do Stub mở rộng lớp android.os.Binder nên mọi đối tượng có kiểu này có thể được điều khiển từ xa bởi một ứng dụng khác, và do lớp Stub này cài đặt lại giao diện lập trình IstockQuoteService nên nó có những phương thức khai báo trong giao diện lập trình này. Lớp Stub là một lớp trừu tượng nên ta phải tạo một lớp cài đặt lại lớp này và dùng nó để cung cấp cho ứng dụng khách.

1.2.1.3 Tạo Service

Tạo một tệp IstockQuoteService.java với nội dung sau:

```
package niit.android.remoteservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
```

```
import android.os.RemoteException;

public class StockQuoteService extends Service
{
     public class StockQuoteServiceImpl extends
IStockQuoteService.Stub
    {
        @Override
        public double getQuote(String ticker) throws
RemoteException
        {
            return 20.0;
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public void onDestroy()
    {
        super.onDestroy();
    }

    @Override
    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
    }

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

Lớp trên định nghĩa một service có tên là IstockQuoteService, trong đó có một lớp nội StockQuoteServiceImpl kế thừa IStockQuoteService.Stub. Trong lớp nội này cài đặt lại các phương thức khai báo trong tệp .aidl kể trên.

Khi một ứng dụng kết nối đến IStockQuoteService thì nó sẽ gọi phương thức onBind() và trả về một đối tượng kiểu StockQuoteServiceIml. Ứng dụng khách sẽ sử dụng đối tượng được trả về ở onBind() để gọi các phương thức cần thiết. Các phương thức này sẽ được IStockQuoteService thực thi.

Lưu ý: Bổ xung khai báo service trong tệp AndroidManifest.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="niit.android.remoteservice"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
          <service android:name="StockQuoteService">
          <intent-filter>
               <action
android:name="niit.android.remoteservice.IStockQuoteService" />
          </intent-filter>
     </service>
     <activity
          android:name=".NotificationActivity"
     ></activity>
    </application>
</manifest>
```

Thực thi chương trình để Android khởi động IStockQuoteService.

1.2.2 Hiện thực ứng dụng khách

1.2.2.1 Tạo project Android

Tạo mới một project với các thông số sau:

Project name: RemoteServiceClientExample

Build target: Android 2.3.3.

Application name: RemoteServiceClientExample

Package name: niit.android.remoteserviceclient

Create Activity: main

1.2.2.2 Chép tệp tin IStockQuoteService.aidl vào project

Tạo package tên remoteservice trong thư mục scr/niit/android/ của project và chép tệp tin IStockQuoteService vào package mới tạo. Eclipse sẽ tự động gọi trình biên dịch AIDL để biên dịch tệp tin IStockQuoteService.aidl thành giao diện lập trình(interface) giống hệt như giao diện lập trình đã được tạo ra ở project trên.

1.2.2.3 Thay đổi activity main

Thay đổi tệp main.xml với nội dung sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button  android:id="@+id/bindBtn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Bind"/>

    <Button android:id="@+id/callBtn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Call Again"
    />

    <Button android:id="@+id/unbindBtn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="UnBind"
    />
</LinearLayout>
```

Thay đổi tệp tin main.java với nội dung sau:

```
package niit.android.remoteserviceclient;

// This file is MainActivity.java
import niit.android.remoteservice.IStockQuoteService;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
```

```
public class main extends Activity {

    protected static final String TAG = "StockQuoteClient";
     private IStockQuoteService stockService = null;

     private Button bindBtn;
     private Button callBtn;
     private Button unbindBtn;

     /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        bindBtn = (Button)findViewById(R.id.bindBtn);
        bindBtn.setOnClickListener(new OnClickListener(){

            @Override
            public void onClick(View view) {
                 bindService(new Intent(IStockQuoteService.class
                         .getName()),
                            serConn, Context.BIND_AUTO_CREATE);
                 bindBtn.setEnabled(false);
                 callBtn.setEnabled(true);
                 unbindBtn.setEnabled(true);
            }});

        callBtn = (Button)findViewById(R.id.callBtn);
        callBtn.setOnClickListener(new OnClickListener(){

            @Override
            public void onClick(View view) {
                callService();
            }});

        callBtn.setEnabled(false);

        unbindBtn = (Button)findViewById(R.id.unbindBtn);
        unbindBtn.setOnClickListener(new OnClickListener(){

            @Override
            public void onClick(View view) {
                unbindService(serConn);
                bindBtn.setEnabled(true);
                callBtn.setEnabled(false);
```

```
                unbindBtn.setEnabled(false);
            }});
        unbindBtn.setEnabled(false);
    }

    private void callService() {
        try {
            double val = stockService.getQuote("SYH");
            Toast.makeText(main.this, "Value from service is
"+val,
Toast.LENGTH_SHORT).show();
        } catch (RemoteException ee) {
            Log.e("MainActivity", ee.getMessage(), ee);
        }
    }

    private ServiceConnection serConn = new ServiceConnection()
    {

        @Override
        public void onServiceConnected(ComponentName name,
IBinder service)
        {
            Log.v(TAG, "onServiceConnected() called");
            stockService =
IStockQuoteService.Stub.asInterface(service);
            callService();
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            Log.v(TAG, "onServiceDisconnected() called");
          stockService = null;
        }
    };
}
```

Trên đó ta có ba nút nhấn dùng để kết nối service (Bind), gọi phương thức trên remote service (Call Again) và ngắt kết nối đến service(UnBind).

Để kết nối đến một service ta sẽ dùng phương thức bindService() và truyền vào 03 tham số:

- Đối tượng Intent xác định service cần kết nối.
- Đối tượng sử lý sự kiện service kết nối thành công và service ngắt kết nối.
- Biến cờ nhận ba giá trị:
    - BIND_AUTO_CREATE: Tự khởi tạo service nếu nó chưa tồn tại.

- BIND_DEBUG_UNBIND: Chạy dưới chế độ gỡ rối. Mọi lời gọi ngắt kết nối đến service sẽ được lưu lại. Khi có một lời gọi yêu cầu ngắt kết nối đến service bị sai thì in ra lời gọi đúng trước đó. Việc này gây tốn khá nhiều tài nguyên nên chỉ sử dụng để gỡ rối.
- BIND_NOT_FOREGROUND: [**Link**]

Ta tiến hành khai báo tắt một đối tượng serConn cài đặt giao diện lập trình ServiceConnection bằng lệnh:

```
private ServiceConnection serConn = new ServiceConnection() {

	@Override
	public void onServiceConnected(ComponentName name, IBinder
service)
	{
		Log.v(TAG, "onServiceConnected() called");
		stockService =
	IStockQuoteService.Stub.asInterface(service);
		  callService();
	}

	@Override
	public void onServiceDisconnected(ComponentName name) {
		Log.v(TAG, "onServiceDisconnected() called");
		stockService = null;
	}
};
```

Đối tượng serConn có hai phương thức:

- onServiceConnected: Phương thức này sẽ được gọi khi kết nối thành công đến service. Phương thức này sẽ nhận vào đối tượng IBinder service. Đây là đối tượng được trả về trên phương thức onBind() của service khai báo ở trên. Ta sẽ dùng đối tượng service này để tạo thành một đối tượng stockService dùng để gọi các phương thức được khai báo trên service.
- onServiceDisconnected: Phương thức này sẽ được gọi khi kết nối đến service đã ngắt.

Ta sẽ truyền serConn vào phương thức bindService() để thông báo rằng đối tượng serConn sẽ là đối tượng nhận các thông báo khi đã kết nối được đến remote service và đã ngắt kết nối với service. Ta cũng sẽ dùng serConn để ngắt kết nối đến service bằng cách gọi phương thức unbindService() và truyền vào serConn.

2 Tham khảo

http://saigeethamn.blogspot.com/2009/09/android-developer-tutorial-part-9.html