Android Advance Lesson 5 Broadcast Receiver Service & Notification

Outline

I. Introduction to Broadcast Receiver

II. Working with System Broadcast Receiver

III. Working with Custom Broadcast Receiver

IV. Working with the Application Object

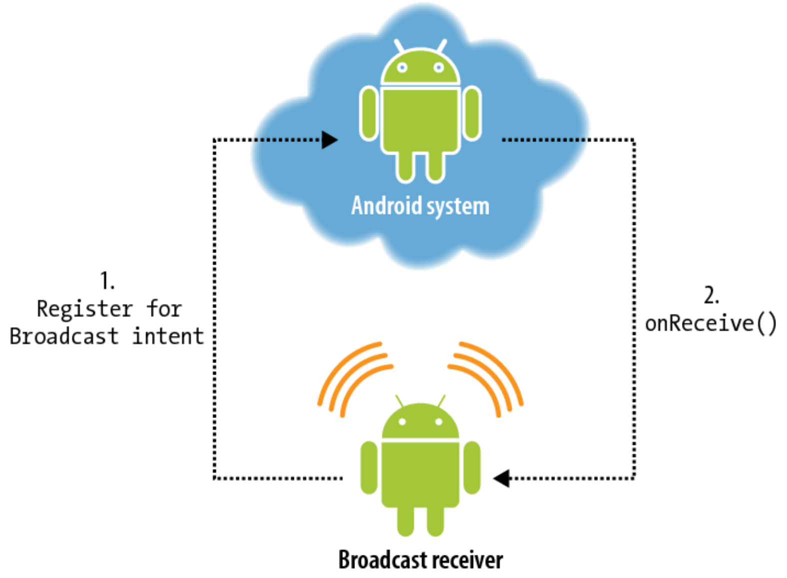
V. Working with Services

VI. Working with Notifications

I. Introduction to Broadcast Receiver

1. What is Broadcast Receiver

- Is a component that responds to system-wide Broadcast announcements
- Listen for a broadcast and executes when it receives that broadcast



1. System broadcast receiver

- The Android operation system broadcast certain actions that occur as a device is being used
- Every broadcast has an action string that uniquely identifies the action

II. Working with System Broadcast Receiver

2. System broadcast receiver

Constant	Action string / Description
ACTION_BOOT_COMPLETED	**android.intent.action.BOOT_COMPLETED** Boot completed. This broadcast is typically used to start services that should alaways run
ACTION_BATTERY_LOW	**android.intent.action.BATTERY_LOW** Battery is low. This broadcast is typically used to pause activites that consume power
ACTION_BATTERY_OKAY	**android.intent.action.BATTERY_OKAY** Battery is OK. This broadcast can be used to resume activities that consume power
ACTION_POWER_CONNECTED	**android.intent.action.ACTION_POWER_CONNECTED** Power source connected to the device
CONNECTION_ACTION	**android.net.conn.CONNECTIVITY_CHANGE** A change in network connectivity has occurred

3. Receiver for the Network Change broadcast

➢ Create BroadcastReceiver

```
public class ConnectivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d("News reader", "Connectivity changed");

        ConnectivityManager connectivityManager = (ConnectivityManager)
                context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo =
                connectivityManager.getActiveNetworkInfo();

        Intent service = new Intent(context, NewsReaderService.class);
        if (networkInfo != null && networkInfo.isConnected()){
            Log.d("News reader", "Connected");
            context.startService(service);
        }
        else {
            Log.d("News reader", "NOT connected");
            context.stopService(service);
        }
    }
}
```

4. Receiver for the Network Change broadcast

➢ Define in AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <receiver
      android:name="ConnectivityReceiver">
              <intent-filter>
                      <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
              </intent-filter>
 </receiver>
```

III. Working with Custom Broadcast Receiver

1. How to create and send a custom broadcast

- Create an Intent object for the broadcast by passing the constant for the action String to the constructor
- Intent intent = **new Intent(*action*)**
- Use method **sendBroadcast**(*intent*)
- Use **putExtra** method to store extra data in intent object

Constructor	Description
Intent(actionString)	Create an Intent with the specified action string

Constructor	Description
sendBroadcast(intent)	Broadcast the specified intent to all register receivers

2. How to create and send a custom broadcast

```
public static final String ACTION_NEW_FEED = “com.framgia.newsreader.ACTION_NEW_FEED”;



public static final String EXTRA_TEXT = “EXTRA_TEST”;

Intent intent = new Intent(NEW_FEED);

intent.putExtra(EXTRA_TEXT, “test”);

sendBroadcast(intent);
```

3. How to code a receiver for a custom broadcast

- **Define a receiver** class for receiver
- Use ***getXxxExtra*** menthod to get data from Intent Object
- Register broadcasts receiver in java class ***onResume*** using ***registerReceiver***
- Unregister broadcast receiver ***onPause*** using ***unregisterReceiver***

Method	Description
registerReceiver(*receiver, filter)*	Register the specified broadcast receiver with the specified intent filter
unregisterReceiver(*receiver)*	Unregister the specified broadcast receiver

4. How to code a receiver for a custom broadcast

- ➢ Register broadcasts receiver in java class ***onResume*** using ***registerReceiver***

```
@Override
onResume(){
        registerReceiver(new  NewFeedReceiver(), new IntentFilter(ACTION_NEW_FEED));
}
```

- ➢ Unregister broadcast receiver ***onPause*** using ***unregisterReceiver***

```
@Override
onPause(){
        unregisterReceiver(new  NewFeedReceiver());
}
```

IV. Working with the Application Object

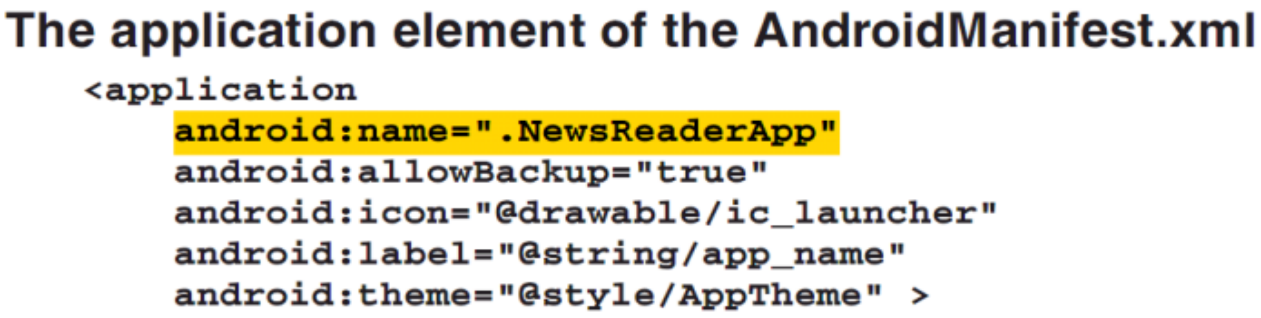
1. How to define the Application object

- To store data and methods that apply to the entire application. The Application object is created when the app starts and remains available until the app ends.
- To run code only once when application starts.

```
package com.murach.newsreader;

import android.app.Application;
import android.util.Log;

public class NewsReaderApp extends Application {

    private long feedMillis = -1;

    public void setFeedMillis(long feedMillis) {
        this.feedMillis = feedMillis;
    }

    public long getFeedMillis() {
        return feedMillis;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d("News reader", "App started");
    }
}
```

2. How to register the Application object

- If you don’t register the custom Application class, Android creates the Application object from the Application object from Application class, not from your custom class.
- To register your custom Application class, open the AndroidManifest.xml file and edit.



The application element of the AndroidManifest.xml

```
<application
    android:name=".NewsReaderApp"
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

3. How use use the Application object

- To get reference to Application object, you can use the getApplication method of the Context class.

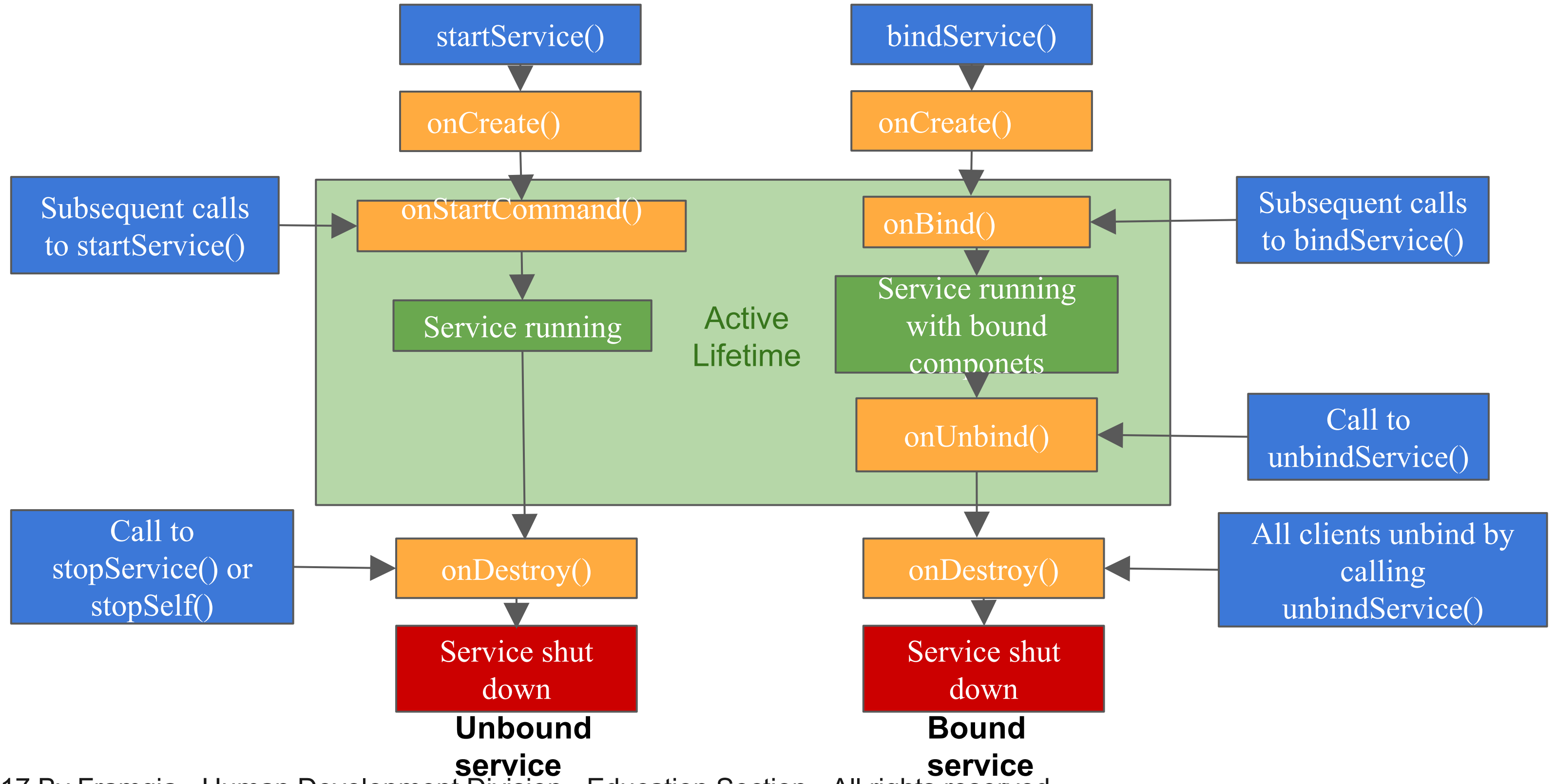
Method	Description
getApplication()	Return the Application objects for your app. You can cast this object to the custom Application class for your app.

V. Working with Services

1. What is service?

- A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application

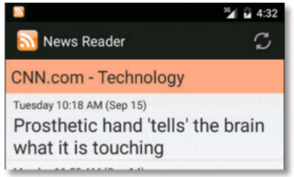
2. Lifecycle of a service



VI. Working with Notifications

1. What is notification?

The News Reader app after it has started a notification

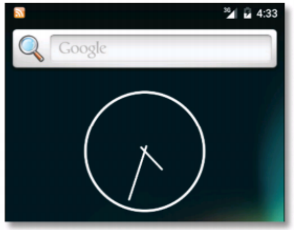


CNN.com - Technology

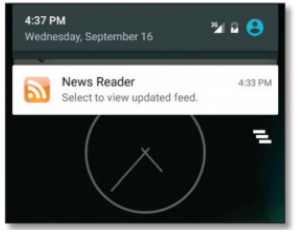
Tuesday 10:18 AM (Sep 15)

Prosthetic hand 'tells' the brain what it is touching

A notification displayed in the notification area



The notification drawer



News Reader 4:33 PM

Select to view updated feed.

2. How to create pending intents

➢ Create a intent

```
Intent notificationIntent = new Intent(this, ItemActivity.class)
                .addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

➢ Create a pendingIntent

```
int flag = PendingIntent.FLAG_UPDATE_CURRENT;
PendingIntent pendingIntent =
                PendingIntent.getActivity(this, 0, notificationIntent, flag);
```

3. How to create a notification

➢ Create the variables for the notification

```
int icon = R.drawable.ic_launcher;
CharSequence tickerText = "Update news feed is available";
CharSequence contentTitle =getText(R.string.app_name);
CharSequence contentText = "Select to view updated feed";
```

➢ Create the notification object

```
Notification notification = new Notification.Builder(this)
                    .setSmallIcon(icon)
                    .setTicker(tickerText)
                    .setContentTitle(contentTitle)
                    .setContentText(contentText)
                    .setContentIntent(pendingIntent)
                    .setAutoCancel(true)
                    .build();
```

4. How to display or remove a notification

Method	**Description**
notify(id, notification**)**	Displays the specified notification and sets its ID to the specified ID. This ID should be unique within your application.
cancel(id**)**	Cancel the notification with the specified ID.

Q&A

Exercise

- Apply knowledge above do Listen Music from resource (source file put in raw folder) have timer off the music (show notification)
- Interface ref: