الفصل الثالث عشر:

الخدمات Services

الكلمات المفتاحية:

الخدمة Service، الخدمات والنياسب Services and threading.

ملخص:

نستعرض في هذا الفصل استخدام الخدمات لتنفيذ المهام في خلفية التطبيق، كما نعرض لأهمية استخدام النياسب مع الخدمات.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- دورة حياة الخدمة.
- قالب صف الخدمة.
- إطلاق الخدمة
- استخدام النياسب مع الخدمات

المخطط:

الخدمات Services

- 2 وحدة (Learning Objects)

1. الخدمات Services

الأهداف التعليمية:

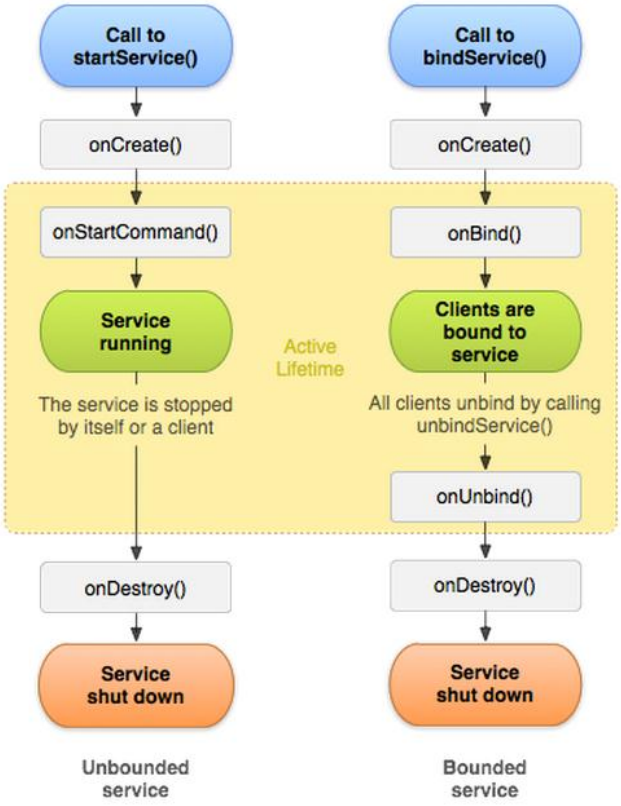
- استخدام الخدمات

الخدمات

- الخدمة هي مهمة تُنفذّ في خلفية التطبيق مثل تشغيل الموسيقا أو تنزيل ملف. تكون الخدمات مفيدة للمهام التي تأخذ وقتاً طويلاً.
- يوفر أندرويد نوعين من الخدمات:
  - الخدمات القياسية standard services: تُستخدم مع المهام الطويلة والتي تبقى تعمل بعد إغلاق التطبيق.
  - خدمات الجسر intent services: تُستخدم مع المهام القصيرة وحيث يقوم التطبيق بإطلاقها مستخدماً الجسور.
- عندما تنتهي الخدمة من عملها، فإنها تقوم بإرسال broadcast هذه المعلومة إلى أي مستقبل receivers يقوم بالاستماع إليها.

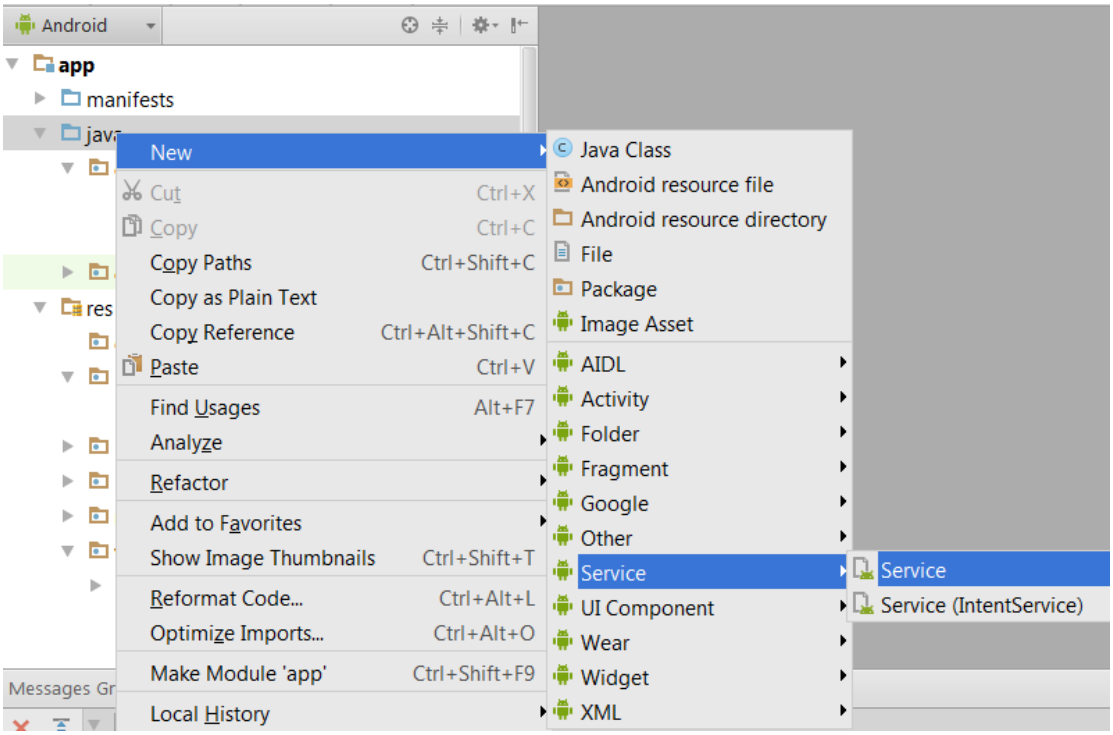
دورة حياة الخدمة

- يتم إنشاء خدمة في تطبيق باستخدام جسر intent.
- يُمكن لخدمة أن تعمل وفق أحد النمطين:
  - نمط البدء start: تبقى الخدمة تعمل حتى إيقافها يدوياً.
  - نمط الربط bind: تبقى الخدمة تعمل حتى لا يبقى تطبيق مرتبط bound معها.
- يكون للخدمة الطرق التالية onCreate, onDestroy.



إضافة خدمة في أندرويد ستديو

من القائمة New، اختر Service/Service:


Android
app
manifests
java
res
New
Cut Ctrl+X
Copy Ctrl+C
Copy Paths Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code... Ctrl+Alt+L
Optimize Imports... Ctrl+Alt+O
Make Module 'app' Ctrl+Shift+F9
Local History
Java Class
Android resource file
Android resource directory
File
Package
Image Asset
AIDL
Activity
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Service
Service (IntentService)
Messages Gr

قالب صف الخدمة

عادةً، يكون صف الخدمة يحوي التعليمات التالية:

```
public class ServiceClassName extends Service {

@Override

public int onStartCommand(Intent intent, int flags, int id) {

// unpack any parameters that were passed to us

        String value1 = intent.getStringExtra("key1");

        String value2 = intent.getStringExtra("key2");

// do the work that the service needs to do ...

        return START_STICKY; // stay running

    }

    @Override

    public IBinder onBind(Intent intent) {

        return null; // disable binding

    }

}
```

التعديلات على الملف AndroidManifest.xml

تأكد من قيام أندرويد استديو بإضافة الأسطر التالي إلى ملف الإعدادات AndroidManifest.xml والتي تسمح للتطبيق باستخدام الخدمة.

```
<service

    android:name=".ServiceClassName"

    android:enabled="true"

    android:exported="true" >

</service>
```

إطلاق الخدمة

يكون إطلاق الخدمة في صف النشاط باستخدام التعليمات:

```
Intent intent = new Intent(this, ServiceClassName.class);

intent.putExtra("key1", "value1");

intent.putExtra("key2", "value2");

startService(intent); // not startActivity!
```

أو في حال أن الخدمة مطلقة من كتلة fragment:

```
Intent intent = new Intent(getActivity(),
                ServiceClassName.class);
```

إجراءات الجسر Intent Actions

يُمكن أن يكون لخدمة عدة إجراءات actions أو أوامر لتقوم بها. مثلاً في حال مشغل الموسيقا، يُمكن أن تكون الإجراءات تشغيل play، إيقاف stop، إيقاف مؤقت pause، ...

يتم في النشاط استخدام التعليمات:

```
Intent intent = new Intent(this, ServiceClassName.class);

intent.setAction("some constant string");

intent.putExtra("key1", "value1");

startService(intent);
```

كما يتم في الخدمة استخدام التعليمات:

```
String action = intent.getAction();

if (action == "some constant string") { ... } else { ...}
```

الإعلام بنتيجة Broadcasting a result

عندما تنتهي خدمة من مهمتها، يُمكن لها إعلام التطبيق عبر إرسال إعلام sending a broadcast للتطبيق الذي يستمع إليها.

يُمكن إسناد إجراء action في الجسر للتفريق بين مختلف النتائج.

```
public class ServiceClassName extends Service {

@Override

public int onStartCommand(Intent tent, int flags, int id) {

// do the work that the service needs to do ...

        ...

// broadcast that the work is done

        Intent done = new Intent();

        done.setAction("action");

        done.putExtra("key1", value1); ...

        sendBroadcast(done);

        return START_STICKY; // stay running

    }
```

استقبال إعلام Receiving a broadcast

يُمكن للنشاط أن يستمع لإعلام باستخدام الصف BroadcastReceiver كما تُبين التعليمات التالية. لاحظ أن أي معامل إضافي في الرسالة يأتي من جسر الخدمة.

```
public class ActivityClassName extends Activity {

    ...

private class ReceiverClassName extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

// handle the received broadcast message

            ...

        }

    }
```

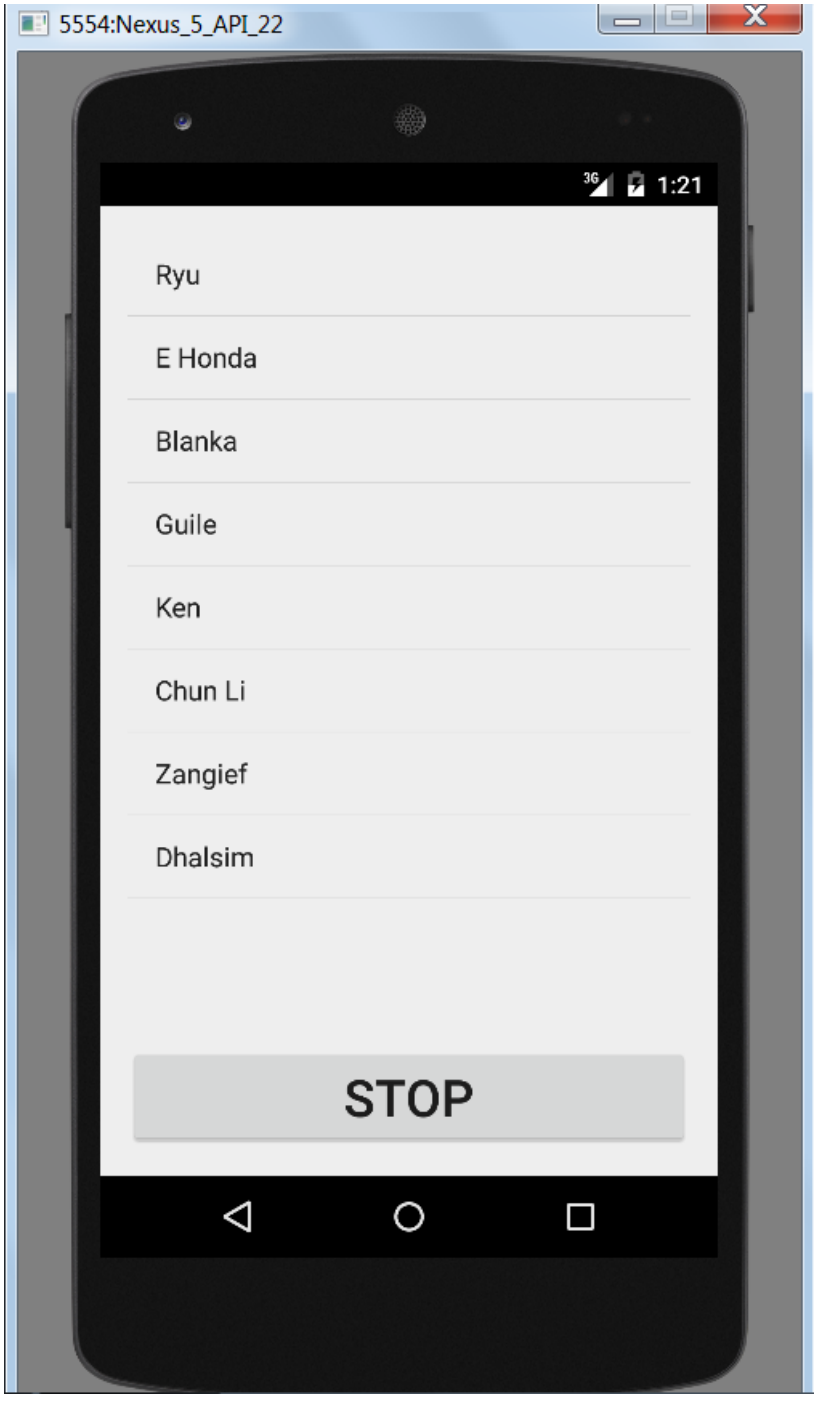
الاستماع إلى إعلام Listening for a broadcast

يُمكن إعداد النشاط بحيث يتم تنبهه عند وقوع إعلام. لاحظ أنه يجب تمرير مرشح جسر intent filter لتحديد الإجراءات المعنية.

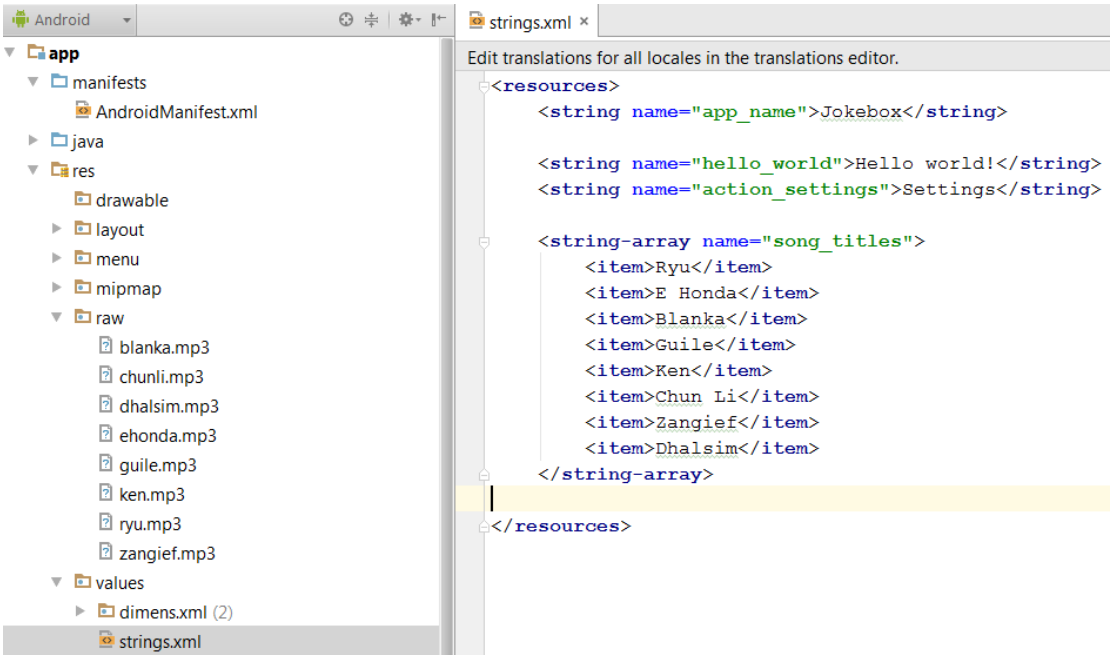
```
public class ActivityClassName extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

        ...

        IntentFilter filter = new IntentFilter();

        filter.addAction("action");

        registerReceiver(new ReceiverClassName(), filter);

}
```

مثال عملي: صندوق الموسيقا Juke Box

تُظهر واجهة التطبيق مجموعة من أسماء المقطوعات الموسيقية. عند النقر على اسم مقطوعة، ستبدأ بالعمل. سوف يستمر العزف حتى لو تم الخروج من التطبيق. يؤدي النقر على زر الإيقاف STOP إلى إيقاف تشغيل المقطوعة.



لتنفيذ التطبيق يجب وضع مجموعة من المقطوعات في المجلد raw والتصريح عن أسماءها في الملف strings.xml: (لاحظ أن اسم المقطوعة هو نفس اسم الملف).



يكون ملف النشاط activity_main.xml:

```
<LinearLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:orientation="vertical"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:paddingLeft="@dimen/activity_horizontal_margin"

    android:paddingRight="@dimen/activity_horizontal_margin"

    android:paddingTop="@dimen/activity_vertical_margin"

    android:paddingBottom="@dimen/activity_vertical_margin"

    tools:context=".JukeboxActivity">


    <ListView

        android:id="@+id/song_list"
```

```
        android:entries="@array/song_titles"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="0dp">
    </ListView>

    <LinearLayout
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
            android:text="Stop"
            android:textSize="30sp"
            android:onClick="onClickStop"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content" />

    </LinearLayout>
</LinearLayout>
```

ويكون ملف الكود MainActivity.java:

```
package com.example.basel.jokebox;

/*
 * This activity is the main GUI for the jukebox app.
 * Click on a song from the list, and it will play in the background
 * using a service.
 * Even if the app is exited, the song will continue playing to completion
 * because it is running in a service.
 */


import android.app.*;
import android.content.*;
import android.os.*;
import android.view.*;
import android.widget.*;


public class MainActivity extends Activity
        implements AdapterView.OnItemClickListener {


    /*
     * This method runs when the activity is started up.
     * Sets up initial state of the GUI.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);



        // listen for clicks on list items

        ListView list = (ListView)
          findViewById(R.id.song_list);

        list.setOnItemClickListener(this);

    }



    /*

     * This method is called when items in the ListView are clicked.

     * It initiates a request to the JukeboxService to play the given song.

     */

    @Override

    public void onItemClick(AdapterView<?> parent,
          View view, int index, long id) {

        String[] songTitles =
     getResources().getStringArray(R.array.song_titles);

        String title =
     songTitles[index].toLowerCase().replace(" ", "");



        Intent intent = new
       Intent(this, JukeboxService.class);

        intent.putExtra("title", title);

        intent.setAction(JukeboxService.ACTION_PLAY);

        startService(intent);

    }



/*
```

```
* This method is called when the Play button is clicked.
* It initiates a request to the JukeboxService to stop the current playback.
*/
    public void onClickStop(View view) {
        Intent intent = new Intent
           (this, JukeboxService.class);
        intent.setAction(JukeboxService.ACTION_STOP);
        startService(intent);
    }
}
```

ويكون ملف الخدمة JukeboxService.java:

```
package com.example.basel.jokebox;

/*
 * This service plays songs in the background.
 * Even if the downloader app is exited, the music keeps playing.
 */

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;

public class JukeboxService extends Service {
// we declare constants for "actions" that are packed into each intent
    public static final String ACTION_PLAY = "play";
```

```
    public static final String ACTION_STOP = "stop";


    // the actual media player to play the music
    private MediaPlayer player;


   /*
    * This method is called each time a request comes in from the app
    * via an intent.  It processes the request by playing or stopping
    * the song as appropriate.
    */
    @Override
    public int onStartCommand(Intent intent,
            int flags, int startId) {
        String action = intent.getAction();
        if (action.equals(ACTION_PLAY)) {
            // convert the song title into a resource ID
            // e.g. "ehonda" -> R.raw.ehonda
            String title = intent.getStringExtra("title");
            int id = getResources().getIdentifier
             (title, "raw", getPackageName());


            // stop any previous playing song
            if (player != null) {
                player.stop();
                player.release();
            }
```

```
            // start the new song
            player = MediaPlayer.create(this, id);
            player.setLooping(true);
            player.start();
        } else if (action.equals(ACTION_STOP)) {
            // stop any currently playing song
            if (player != null) {
                player.stop();
                player.release();
            }
        }
        return START_STICKY;
   // START_STICKY means service will keep running
    }

/*
* This method specifies how our service will deal with binding.
* We don't choose to support binding, which is indicated by returning null.
*/
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

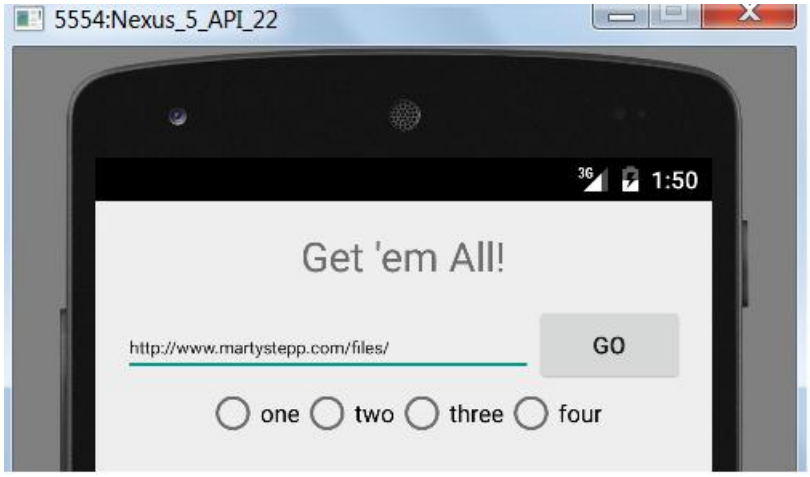
2. الخدمات والنياسب Services and Threads

الأهداف التعليمية:

- استخدام النياسب مع الخدمات

الخدمات والنياسب

- تعيش الخدمة بشكل افتراضي في نفس المعالج process والنيسب thread للتطبيق الذي أنشئها.
- بالطبع، لا يُلائم ذلك المهام الطويلة للخدمات لأنها سوف تقوم بتجميد التطبيق الذي عليه أن ينتظر حتى تنتهي من عملها.
- فمثلاً في تطبيقنا التالي الذي يقوم بتنزيل ملفات من الوب، لن يكون بالإمكان التفاعل مع أزرار الراديو أثناء تحميل الملفات.



- لجعل التطبيق والخدمة مستقلين عن بعضهما يجب إسناد الخدمة إلى نيسب خاص بها.

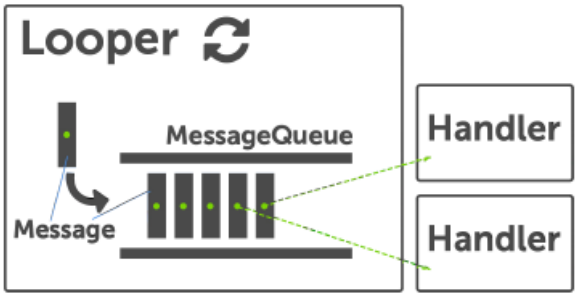
الخدمة مع نيسب

- يكون مخطط تعليمات الخدمة مع نيسب:

```
public class ServiceClassName extends Service {

    /* this method handles a single incoming request */

@Override

public int onStartCommand(Intent intent, int flags, int id) {

   // unpack any parameters that were passed to us

        String value1 = intent.getStringExtra("key1");

        Thread thread = new Thread(new Runnable() {

            public void run() {

     // do the work that the service needs to do

            }

        });

        thread.start();

        return START_STICKY; // stay running

    }
```

الصفوف المساعدة مع النياسب Android thread helper classes

- يستقبل النيسب المهام المطلوبة منه وتوضع عادةً في رتل Queue.



- يُستخدم الصف HandlerThread للتعامل مع رتل المهام المطلوبة.
- يقوم غرض من الصف Looper بحلقة لانتظار المهام ومعالجتها.
- يُمكن إضافة مهام جديدة باستخدام الغرض السابق.
- تُبين التعليمات التالية الشكل العام لاستخدام الصفين السابقين:

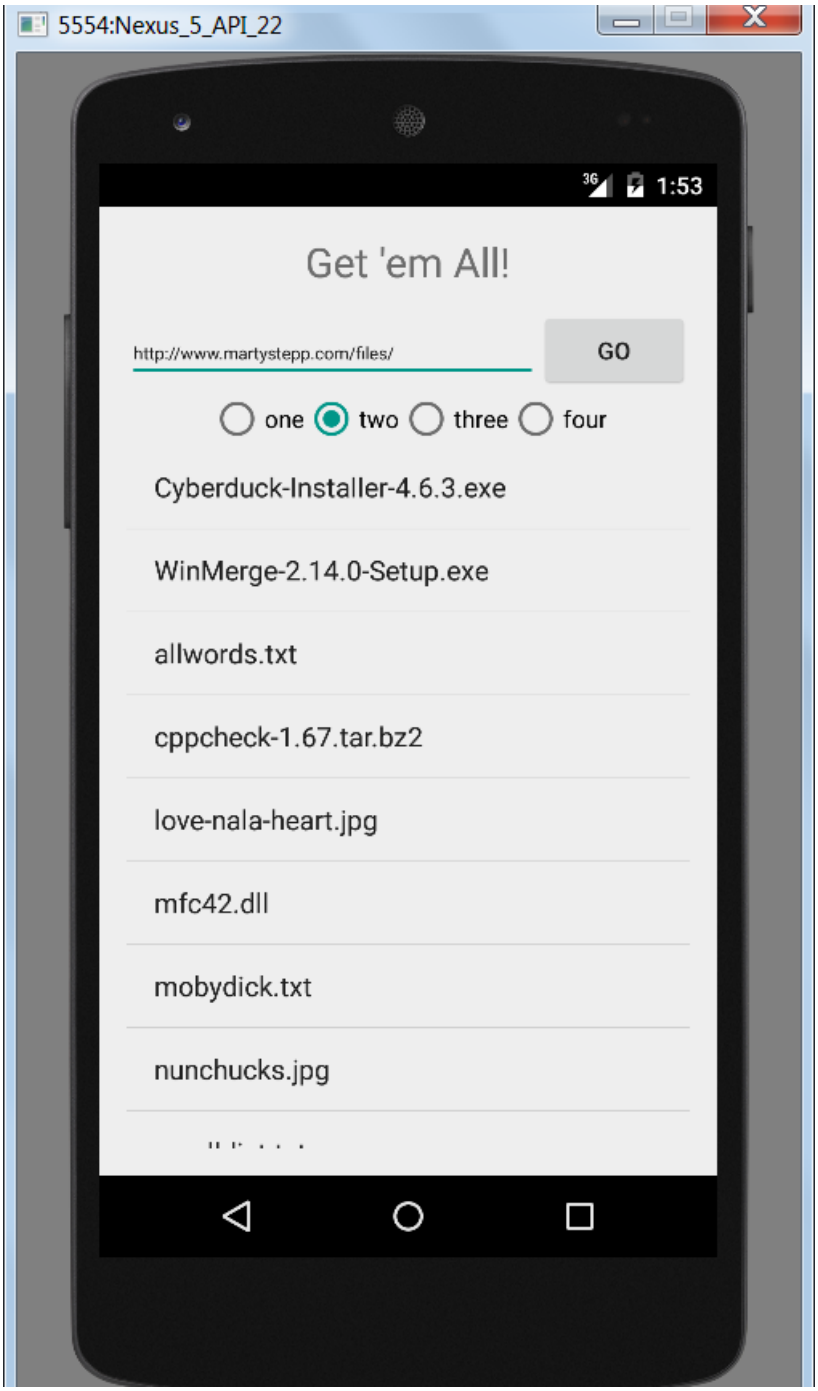
```
HandlerThread hThread = new HandlerThread("name");

hThread.start();


Looper looper = hThread.getLooper();

...
```

- يُستخدم غرض من الصف Handler لمعالجة مهمة واحدة من رتل المهام:
  - عند بناء الغرض، قم بتمرير الغرض المنشئ من الصف Looper.
  - قم بإسناد مهمة باستخدام الطريقة post.

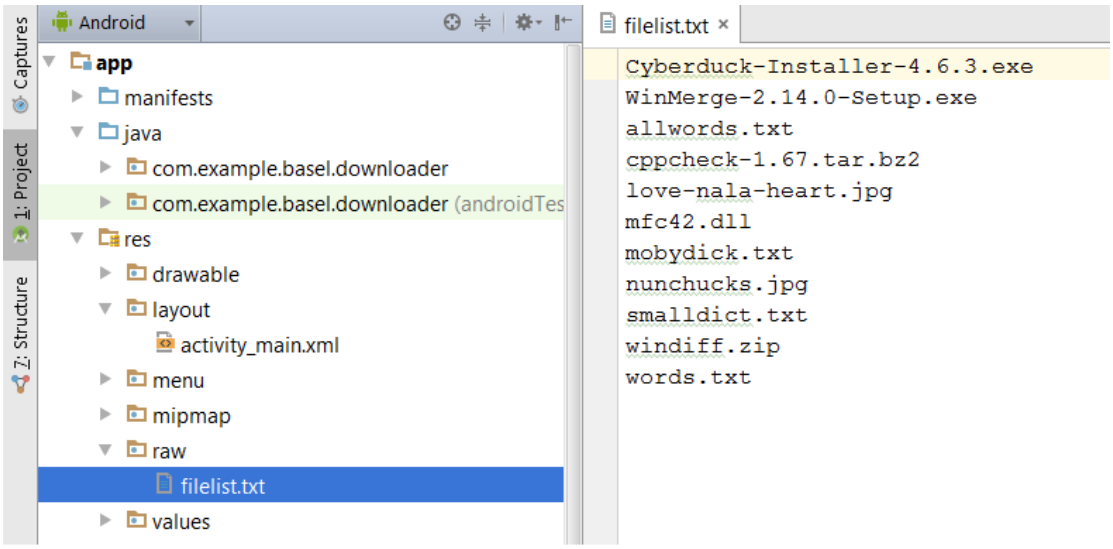
```
Handler handler = new Handler(looper);

        handler.post(new Runnable() {

            public void run() {

// the code to process the job

                ...

            }

        });
```

مثال عملي: تطبيق تنزيل الملفات Downloader

تُظهر واجهة التطبيق زر أمر لتنزيل مجموعة من الملفات من موقع معين. سوف يكون بالإمكان التعامل مع أزرار الراديو أثناء التنزيل لأن التنزيل تقوم به خدمة في نيسب خاص بها.



لتنفيذ التطبيق يجب وضع الملف filelist.txt في المجلد raw والذي يحوي أسماء الملفات المطلوب تنزيلها.



يكون ملف النشاط activity_main.xml:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:gravity="top|center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".DownloaderActivity">
```

```
    <TextView
        android:text="Get 'em All!"
        android:textSize="24sp"
        android:layout_marginBottom="12dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <EditText
            android:id="@+id/the_url"
            android:text="http://www.martystepp.com/files/"
            android:textSize="10sp"
            android:layout_weight="1"
            android:lines="1"
            android:maxLines="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content" />

        <Button
```

```
            android:text="Go"
            android:onClick="onClickGo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>

    <RadioGroup
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <RadioButton
            android:text="one"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <RadioButton
            android:text="two"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <RadioButton
            android:text="three"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <RadioButton
            android:text="four"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </RadioGroup>

    <ListView
        android:id="@+id/list_of_links"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="0dp">

    </ListView>
</LinearLayout>
```

ويكون ملف النشاط MainActivity.java:

```
package com.example.basel.downloader;

/*
 * This activity is the main GUI for the downloader app.
 * You can type a URL in the top edit text pane, then press Go,
 * and all links in that page are shown as items in a list view.
 * You can click the list items to download them in the background
 * using a service.
 * Even if the downloader app is exited, the download will get finished
 * because it is running in a service.
 */


import android.app.*;
import android.content.*;
import android.os.*;
import android.util.*;
import android.view.*;
import android.widget.*;
import java.util.*;


public class MainActivity extends Activity
        implements AdapterView.OnItemClickListener {


// web domain where files will be downloaded from
```

```
private static final String DOMAIN =
"http://www.svuonline.org/files/";



private ArrayList<String> listOfLinks;// these are for the ListView

private ArrayAdapter<String> adapter;

    /*

     * This method runs when the activity is started up.

     * Sets up initial state of the GUI.

     */

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);



        // set up the ListView

        listOfLinks = new ArrayList<>();

        adapter = new ArrayAdapter<String>
     (this, android.R.layout.simple_list_item_1, listOfLinks);

        ListView listOfLinks = (ListView)
                              findViewById(R.id.list_of_links);

        listOfLinks.setAdapter(adapter);

        listOfLinks.setOnItemClickListener(this);



// set up a broadcast receiver to be informed when downloads are finished

        IntentFilter filter = new IntentFilter();
```

```
filter.addAction(DownloaderService.ACTION_DOWNLOAD_COMPLETE);
        registerReceiver(new MyReceiver(), filter);
    }

    /*
     * This method is called when the "Go" button is clicked.
     * It fetches all links contained in the given web page.
     */
public void onClickGo(View view) {
EditText edit = (EditText) findViewById(R.id.the_url);
String webPageURL = edit.getText().toString();
Scanner scan = new
Scanner(getResources().openRawResource(R.raw.filelist));
        while (scan.hasNextLine()) {
            listOfLinks.add(scan.nextLine());
        }
        adapter.notifyDataSetChanged();
    }

/*
* This method is called when items in the ListView are clicked.
* It initiates a request to the DownloadService to download the file.
*/
 @Override
```

```
public void onItemClick(AdapterView<?> parent, View view, int index,
long id) {

        EditText edit = (EditText) findViewById(R.id.the_url);

        String domain = edit.getText().toString();

        String url = domain + listOfLinks.get(index);


// send request to DownloadService using an intent

        Intent intent = new Intent
                        (this, DownloaderService.class);

        intent.putExtra("url", url);

        intent.setAction(DownloaderService.ACTION_DOWNLOAD);

        startService(intent);


    }



/*

* This broadcast receiver listens for "download complete" broadcasts

* sent by the DownloadService and reacts to them by showing a toast.

*/

private class MyReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

   String action = intent.getAction();

   if (action.equals(DownloaderService.ACTION_DOWNLOAD_COMPLETE)) {

     String url = intent.getStringExtra("url");

     Toast.makeText(MainActivity.this, "done downloading " + url,
Toast.LENGTH_SHORT).show();

}}}}
```

ويكون ملف الخدمة DownloaderService.java:

```
package com.example.basel.downloader;

/*
 * This service downloads files in the background.
 * Even if the downloader app is exited, the service remains running.
 */

import android.app.*;

import android.content.*;

import android.os.*;



public class DownloaderService extends Service {

// we declare constants for "actions" that are packed into each intent

public static final String ACTION_DOWNLOAD = "download";

public static final String ACTION_DOWNLOAD_COMPLETE =
"download_complete";



// constant ID sent when we broadcast a download-complete message

public static final int ID_NOTIFICATION_DL_COMPLETE = 1234;



// handler thread loops waiting for jobs (downloads) to run

// in a separate thread

private HandlerThread hThread;

/*

* This method runs when the service is started up.
```

```
* Sets up initial state of the handler thread.

*/

@Override

public void onCreate() {

        super.onCreate();

        hThread = new HandlerThread("Skippy");

        hThread.start();

    }



/*

* This method is called each time a request comes in from the app

* via an intent.  It processes the request by enqueuing a new

* download job in the background handler thread.

*/

@Override

public int onStartCommand(final Intent intent, int flags, int
startId) {

        String action = intent.getAction();

        if (action.equals(ACTION_DOWNLOAD)) {

            // create a "runnable" task for this download

            Runnable runner = new Runnable() {

                public void run() {

                    // download the file

                    String url = intent.getStringExtra("url");


com.example.basel.downloader.Downloader.downloadFake(url);
```

```
// show a notification in the top notifications bar

 Notification.Builder builder = new
Notification.Builder(DownloaderService.this)

                    .setContentTitle("Download complete")

                    .setContentText(url)

                    .setAutoCancel(true)

                .setSmallIcon(R.drawable.icon_download);

            Notification notification = builder.build();

            NotificationManager manager = (NotificationManager)


getSystemService(Context.NOTIFICATION_SERVICE);

      manager.notify
      (ID_NOTIFICATION_DL_COMPLETE, notification);



// broadcast a message back to the app to inform it that we are done

                    Intent done = new Intent();

                    done.setAction(ACTION_DOWNLOAD_COMPLETE);

                    done.putExtra("url", url);

                    sendBroadcast(done);

                }

             };



// wrap the runnable into a Handler job to be given to our background thread

              Handler handler = new
           Handler(hThread.getLooper());

              handler.post(runner);
```

```
        }


   return START_STICKY;// START_STICKY means service will keep running
    }


/*
* This method specifies how our service will deal with binding.
* We don't choose to support binding, which is indicated by returning null.
*/
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

تم استخدام الصف المساعد Downloader.java

```
/*
 *
 * This is just a helper class that we used in the downloader app.
 * It does the actual work of downloading a file from a given URL.
 * It also has "fake" download methods to simulate downloading the file
 * without actually using the network, just for testing.
 */


package com.example.basel.downloader;


import android.os.Environment;
```

```
import android.util.Log;

import org.w3c.dom.*;

import java.io.*;

import java.net.*;

import java.util.*;

import javax.xml.parsers.*;


public class Downloader {

/**

* Downloads the file found at the given URL,

* into the Android device's Downloads folder in its external storage,

* and returns the file name it was saved to.

*/

public static String download(String urlString) {

File folder = Environment.getExternalStoragePublicDirectory
                            (Environment.DIRECTORY_DOWNLOADS);

Log.v("Downloader", "downloading " + urlString + " to " + folder);

        try {

            Thread.sleep(2000);

        } catch (InterruptedException e) {

            // empty

        }


        if (!folder.exists()) {

            folder.mkdirs();

        }
```

```
        // download the file into a memory buffer
        byte[] bytes = downloadToByteArray(urlString);

        // store the memory buffer contents into a file
        File urlFile = new File(urlString);
        String filename = urlFile.getName();
        File outFile = new File(folder, filename);
        try {
            FileOutputStream out = new
                            FileOutputStream(outFile);
            out.write(bytes);
            out.close();
            return filename;
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

/**
* Pretends to download the file found at the given URL,
* and returns the file name it would have been saved to.
*/
public static String downloadFake(String urlString) {
File folder = Environment.getExternalStoragePublicDirectory
```

```
                            (Environment.DIRECTORY_DOWNLOADS);

        File urlFile = new File(urlString);

        String filename = urlFile.getName();

        File outFile = new File(folder, filename);

        Log.v("Downloader", "downloading " + urlString + " to " +
outFile);



        try {

            Thread.sleep(3000);

        } catch (InterruptedException e) {

            // empty

        }



        return filename;

    }



/**

* Retrieves all of the links like <a href="http://example.com/foo.html">...</a>

* from the page and returns their href URLs as an array.

* Doesn't work on some pages due to invalid HTML content.

*/

public static String[] getAllLinks(String webPageURL) {

        try {

            byte[] bytes = downloadToByteArray(webPageURL);

            ArrayList<String> list = new ArrayList<>();
```

```
            DocumentBuilderFactory factory =
                      DocumentBuilderFactory.newInstance();

            factory.setNamespaceAware(true);

            factory.setIgnoringElementContentWhitespace(true);

            factory.setValidating(false);

            DocumentBuilder builder =
                      factory.newDocumentBuilder();

            Document document = builder.parse(
                           new ByteArrayInputStream(bytes));

            NodeList linkNodes =
                    document.getElementsByTagName("a");

            for (int i = 0 ; i < linkNodes.getLength(); i++) {

                Node node = linkNodes.item(i);

                Node hrefNode =
                    node.getAttributes().getNamedItem("href");

                if (hrefNode != null) {

                    String href = hrefNode.getNodeValue();

                    try {

                        URL url = new URL(href);

                        list.add(href);

                    } catch (MalformedURLException mfurle) {

                        // invalid URL; don't add

                    }

                }

            }


            return list.toArray(new String[0]);
```

```
        } catch (Exception e) {

            throw new RuntimeException(e);

        }

    }



/**

* Reads the entire contents of the given file from the device's
Downloads folder,

* returning the file's contents as a text string.

*/

public static String readEntireFile(String filename) {

        try {

            File dir = Environment.getExternalStoragePublicDirectory
                     (Environment.DIRECTORY_DOWNLOADS);

            File file = new File(dir, filename);

            StringBuilder sb = new StringBuilder();

            BufferedReader reader = new
                       BufferedReader(new FileReader(file));

            while (reader.ready()) {

                sb.append((char) reader.read());

            }

            return sb.toString();

        } catch (IOException e) {

            throw new RuntimeException(e);

        }

    }
```

```
/**
* Downloads the file found at the given URL into a memory buffer of bytes,
* then returns the bytes as an array.
* This is used internally as a temporary helper method.
*/
private static byte[] downloadToByteArray(String urlString) {
        try {
            // download the file into a memory buffer
            ByteArrayOutputStream bytes = new
                                    ByteArrayOutputStream();
            URL url = new URL(urlString);
            InputStream stream = url.openStream();
            BufferedReader reader = new BufferedReader
                                (new InputStreamReader(stream));
            int ch;
            while ((ch = reader.read()) != -1) {
                bytes.write(ch);
            }
            stream.close();
            return bytes.toByteArray();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```