

Android Threads

Thread

- Main thread (UI thread)
 - When an application is launched, the system creates the main thread
- Worker thread
 - perform non-instantaneous operations in separate threads (background threads)

Why we need worker thread?

- Android enforces a worst case reaction time of applications
 - If an activity does not react within **5 seconds** to user input, the Android system displays an **Application not responding (ANR) dialog**

Rules to use thread in Android

- Do not block the UI thread
- Do not access the Android UI toolkit from outside the UI thread

<http://developer.android.com/guide/components/processes-and-threads.html>

Worker threads

- Java threads
 - Not convenient and has several limitations
- AsyncTask
 - The simplest way to use thread
- Handler
 - Can handle multiple runnable tasks and messages

Java Threads

- Android supports the usage of the Thread class to perform asynchronous processing
- If you need to update the user interface from a new Thread, you need to **synchronize** with the UI thread

Take ImageLoader as an Example

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            Bitmap b = loadImageFromNetwork("http://example.com/image.png");  
            mImageView.setImageBitmap(b);  
        }  
    }).start();  
}
```

Take ImageLoader as an Example

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            Bitmap b = loadImageFromNetwork("http://example.com/image.png");  
            mImageView.setImageBitmap(b);  
        }  
    }).start();  
}
```

This seems to work fine:

a new thread to handle the downloading task

but it violates the second rule

-> change UI from outside UI thread

How to Use Java Thread to Update UI?

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            final Bitmap bitmap =  
                loadImageFromNetwork("http://example.com/image.png");  
            mImageView.post(new Runnable() {  
                public void run() {  
                    mImageView.setImageBitmap(bitmap);  
                }  
            });  
        }  
    }).start();  
}
```

Disadvantages to Use Java Thread in Android

- Without synchronization with the UI thread
- Cannot stop the thread by `destroy()` or `stop()`
- No default for handling configuration changes in Android

The advantages of AsyncTask and Handler

- They provide a function to help you post the resulting data to UI thread
 - conform the second rule
- Automatically handle the configuration changing
- Have function to stop the tasks

AsyncTask

- The easiest way to perform a task in background
- Each task can only be executed once
 - If you want to execute again, you need to create a new task

Steps of AsyncTask

- `onPreExecute()`:
 - used to set up the task
- `doInBackground(Params...)`:
 - perform background computation that can take a long time (must be overridden)
- `onProgressUpdate(Progress...)`:
 - This method is used to display progress
- `onPostExecute(Result)`:
 - result of the background computation is passed to this step

Rules of AsyncTask

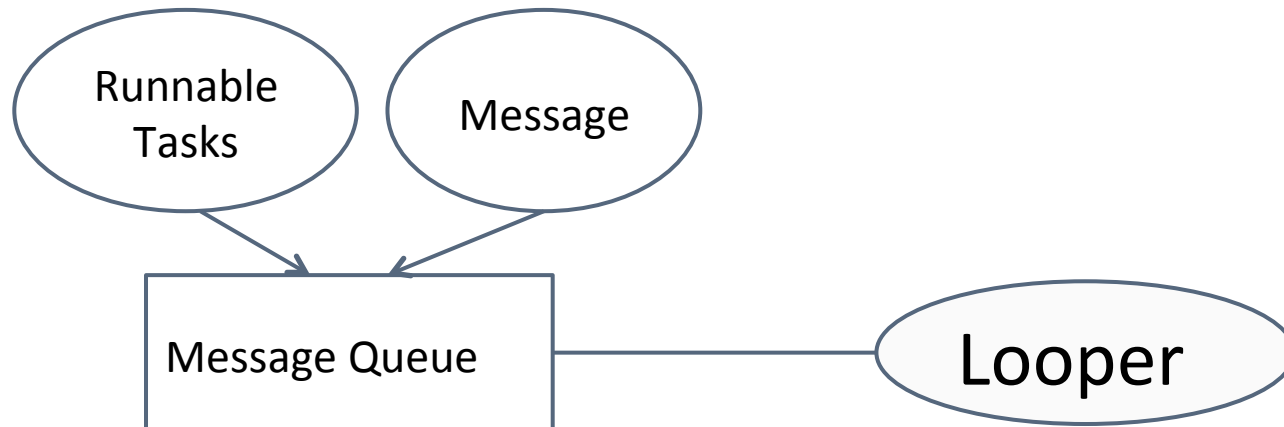
- The AsyncTask class must be loaded on the UI thread
- execute(Params...) must be invoked on the UI thread
- Do not call the functions of 4 steps manually
- The task can be executed only once

Example: SaveFile sample code

<https://dl.dropboxusercontent.com/u/21274694/android/SaveFile.zip>

Handler

- When a Handler is created, it is bound to a specific Looper (and associated thread and message queue)
- A Handler is a utility class that facilitates interacting with a Looper



Create Handler

- A Handler object registers itself with the thread where it is created
- If you create a new instance of the Handler class in the onCreate() method of your activity, the resulting Handler object can be used to post data to the UI thread

How to Use Handler

- To process a **Runnable** you can use the `post()` method
- Override the **`handleMessage()`** method to process messages
 - Your thread can send messages via the **`obtainMessage(Message)`** or **`sendMessage(Message)`** method to the Handler object

Samples

- Handling Runnable Tasks

- BluetoothExample sample

<https://dl.dropboxusercontent.com/u/21274694/android/BluetoothExample.zip>

- Handling Messages

- BluetoothChat Sample

New->Import Sample->search bluetooth->select bluetooth chat

- Make discoverable (A) -> scan devices (B) -> connected to the devices A (B) -> chat with each other

Exercise

- Using the `handler.post(Runnable)` to make your application periodically query the paired devices
 - If no paired device, please show “no paired device”

Steps

- Step1: Create a “Runnable” called “periodicBTTask2”
- Step2: Copy and paste the code implemented in “queryPairedDevices” to your runnable
- Step3: Use `BTHandler.post()` to post your “Runnable (periodicBTTask2)” to message queue
- Step4: in your “Runnable (periodicBTTask2)”, you need to use “`BTHandler.postDelayed()`” to make your Runnable performed periodically