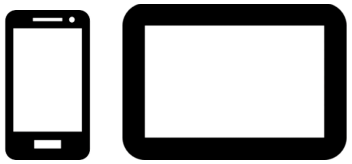
[TD4] Traitement en tâche de fond



GÉNÉRALITÉ

Lors du passage d'une application au premier plan, l'exécution des différents traitements s'effectue dans l'**UI Thread**.

Afin d'obtenir une application fluide il ne faut pas que des traitements lourds soient exécutés dans ce Thread.

GÉNÉRALITÉ

Nous allons voir deux notions :

AsyncTasks

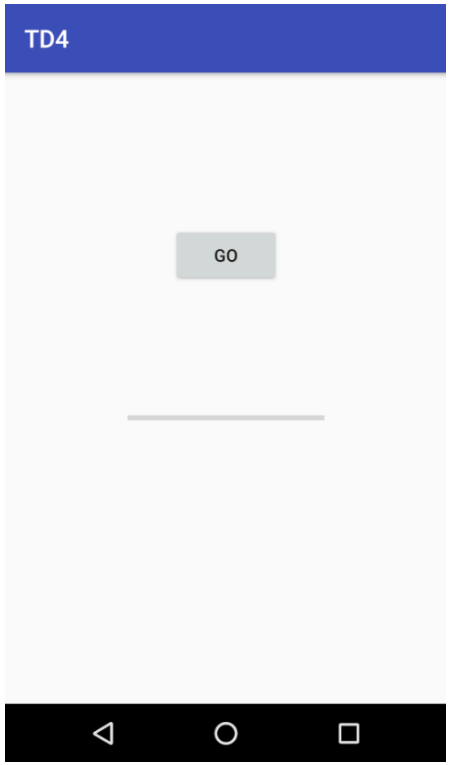
Permettent de réaliser des tâches longues dans un thread différent de l'UI Thread.

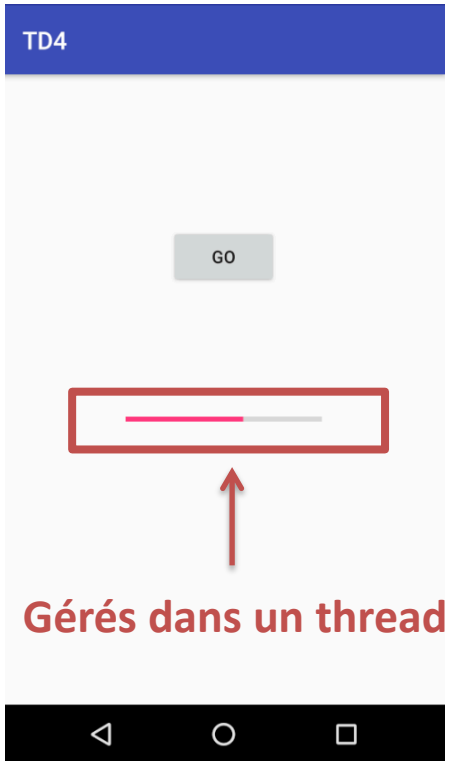
Services

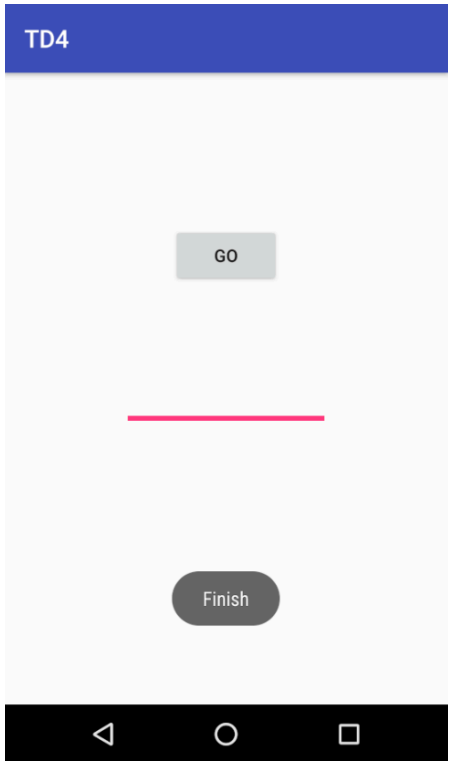
Permettent de réaliser une tâche de fond. Ces tâches peuvent être réalisées dans un thread différent de l'UI Thread.

ASYNCTASK

Permet d'exécuter un traitement long sans bloquer l'UI Thread.







ASYNCTASK

```
protected ProgressBar p;
protected Button b;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    p = findViewById(R.id.progress);
    b = findViewById(R.id.go);
}

@Override
protected void onStart() {
    super.onStart();
    b.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Calculation calculation = new Calculation();
            calculation.execute();
        }
    });
}
```

ASYNCTASK

```
private class Calculation extends AsyncTask<Void, Integer, Void> {

    @Override
    protected void onPreExecute() {
        p.setProgress(0);
    }
    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        p.setProgress(values[0]);
    }
    @Override
    protected Void doInBackground(Void... params) {
        String result = "";
        for(int i=0; i<=10; i++) {
            try {
                Thread.sleep(1000L);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            result += i;
            publishProgress(i*10);
        }
        return null;
    }
    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
        Toast.makeText(MainActivity.this,R.string.finish,Toast.LENGTH_LONG).show();
    }
}
```

Classe paramétrée par trois types de données :

1. Passé en argument à la classe
   (méthode **doInBackground**)

1. Utilisé pour publier l'avancement de la tâche
   (méthode **onProgressUpdate**)

2. Utilisé pour publier le résultat sur l'interface
   (méthode **onPostExecute**)

ASYNCTASK

```
private class Calculation extends AsyncTask< Void, Integer, Void > {

   @Override
   protected void onPreExecute() {
      p.setProgress(0);
   }
   @Override
   protected void onProgressUpdate( Integer... values ) {
      super.onProgressUpdate(values);
      p.setProgress(values[0]);
   }
   @Override
   protected Void doInBackground( Void... params ) {
      String result = "";
      for(int i=0; i<=10; i++) {
         try {
            Thread.sleep(1000L);
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
         result += i;
         publishProgress(i*10);
      }
      return null;
   }
   @Override
   protected void onPostExecute( Void result ) {
      super.onPostExecute(result);
      Toast.makeText(MainActivity.this,R.string.finish,Toast.LENGTH_LONG).show();
   }
}
```

Permet de mettre à jour l'interface de l'application avant le début de l'exécution de la tâche.

Cette méthode est exécutée dans l'UI Thread.

ASYNCTASK

```
private class Calculation extends AsyncTask< Void, Integer, Void > {

    @Override
    protected void onPreExecute() {
        p.setProgress(0);
    }
    @Override
    protected void onProgressUpdate( Integer... values ) {
        super.onProgressUpdate(values);
        p.setProgress(values[0]);
    }
    @Override
    protected Void doInBackground( Void... params ) {
        String result = "";
        for(int i=0; i<=10; i++) {
            try {
                Thread.sleep(1000L);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            result += i;
            publishProgress(i*10);
        }
        return null;
    }
    @Override
    protected void onPostExecute( Void result ) {
        super.onPostExecute(result);
        Toast.makeText(MainActivity.this,R.string.finish,Toast.LENGTH_LONG).show();
    }
}
```

Permet de mettre à jour la progression de la tâche en cours d'exécution.

Cette méthode est appelée grâce à la méthode **publishProgress** dans l'UI Thread.

La méthode publishProgress est invoquée depuis doInBackground.

ASYNCTASK

```
private class Calculation extends AsyncTask< Void, Integer, Void > {

   @Override
   protected void onPreExecute() {
      p.setProgress(0);
   }
   @Override
   protected void onProgressUpdate( Integer... values ) {
      super.onProgressUpdate(values);
      p.setProgress(values[0]);
   }
   @Override
   protected Void doInBackground( Void... params ) {
      String result = "";
      for(int i=0; i<=10; i++) {
         try {
            Thread.sleep(1000L);
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
         result += i;
         publishProgress(i*10);
      }
      return null;
   }
   @Override
   protected void onPostExecute( Void result ) {
      super.onPostExecute(result);
      Toast.makeText(MainActivity.this,R.string.finish,Toast.LENGTH_LONG).show();
   }
}
```

Cette méthode est exécutée dans un thread séparé.

ASYNCTASK

```
private class Calculation extends AsyncTask< Void, Integer, Void > {

    @Override
    protected void onPreExecute() {
        p.setProgress(0);
    }
    @Override
    protected void onProgressUpdate( Integer... values ) {
        super.onProgressUpdate(values);
        p.setProgress(values[0]);
    }
    @Override
    protected Void doInBackground( Void... params ) {
        String result = "";
        for(int i=0; i<=10; i++) {
            try {
                Thread.sleep(1000L);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            result += i;
            publishProgress(i*10);
        }
        return null;
    }
    @Override
    protected void onPostExecute( Void result ) {
        super.onPostExecute(result);
        Toast.makeText(MainActivity.this,R.string.finish,Toast.LENGTH_LONG).show();
    }
}
```

Permet de mettre à jour l'interface avec le résultat obtenu.

LES SERVICES

Les services sont des composants qui ne possèdent pas d'interface utilisateur. Ils permettent de réaliser des traitements assez longs sans interaction avec les utilisateurs.

Un service s'exécutant en tâche de fond est prioritaire sur une activité se trouvant aussi en tâche de fond. **Les services sont donc moins exposés lors de la libération de ressources par l'OS**.

LES SERVICES

Combiner avec les activités

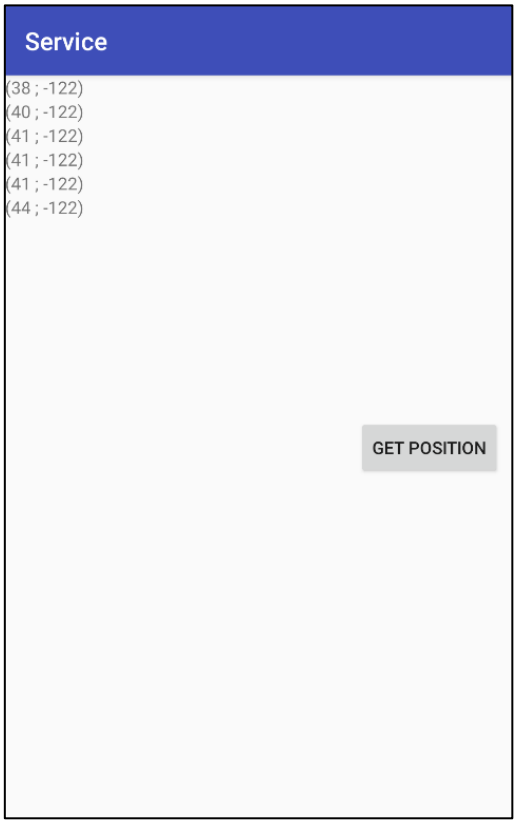
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cfc.td4">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name="com.cfc.td4.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name="com.cfc.td4.AI"
            android:enabled="true"
            android:exported="true" />
    </application>

</manifest>
```

N'oubliez pas de déclarer votre service dans le manifeste de l'application.

Le simple est de créer le service en passant par l'interface :

File → New → Service → Service

LES SERVICES



```java
public class MainActivity extends AppCompatActivity {

    private Button get_position;
    private Localization localization;
    Intent intentService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        get_position = findViewById(R.id.get_position);
        intentService = new Intent(this, Localization.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
        bindService(intentService,myServiceConnection, Context.BIND_AUTO_CREATE);
        get_position.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle bundle = new Bundle();
                bundle.putString("localization",localization.getLocalization());
                Data data = new Data();
                data.setArguments(bundle);
                getSupportFragmentManager().beginTransaction()
                        .add(R.id.fragment_container,data)
                        .commit();
            }
        });
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
        unbindService(myServiceConnection);
    }

    private final ServiceConnection myServiceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            localization = ((Localization.MyBinder) iBinder).getService();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            localization = null;
        }
    };

}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/fragment_container"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintWidth_default="percent"
        app:layout_constraintWidth_percent="0.5">
    </LinearLayout>

    <Button
        android:id="@+id/get_position"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintWidth_default="percent"
        app:layout_constraintWidth_percent="0.5"
        android:text="Get position"/>

</androidx.constraintlayout.widget.ConstraintLayout >
```

```java
public class MainActivity extends AppCompatActivity {

    private Button get_position;
    private Localization localization;
    Intent intentService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        get_position = findViewById(R.id.get_position);
        intentService = new Intent(this, Localization.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
        bindService(intentService,myServiceConnection, Context.BIND_AUTO_CREATE);
        get_position.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle bundle = new Bundle();
                bundle.putString("localization",localization.getLocalization());
                Data data = new Data();
                data.setArguments(bundle);
                getSupportFragmentManager().beginTransaction()
                        .add(R.id.fragment_container,data)
                        .commit();
            }
        });
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
        unbindService(myServiceConnection);
    }

    private final ServiceConnection myServiceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            localization = ((Localization.MyBinder) iBinder).getService();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            localization = null;
        }
    };

}
```

Déclaration du service et d'un intent qui permettra de lancer le service.

Lancement du service.

NB : un même service pourra recevoir plusieurs intents avec de nouvelles instructions à traiter. Il traitera ces instructions comme s'il s'agissait des premières reçues.

Arrêt du service.

*NB : un service pourra également s'arrêter lui-même en invoquant les méthodes **stopSelf** ou **stopSelfResult***

Afin de communiquer avec le service il faut établir une connexion avec celui-ci.

```
public class MainActivity extends AppCompatActivity {

    private Button get_position;
    private Localization localization;
    Intent intentService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        get_position = findViewById(R.id.get_position);
        intentService = new Intent(this, Localization.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
        bindService(intentService,myServiceConnection, Context.BIND_AUTO_CREATE);
        get_position.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle bundle = new Bundle();
                bundle.putString("localization",localization.getLocalization());
                Data data = new Data();
                data.setArguments(bundle);
                getSupportFragmentManager().beginTransaction()
                        .add(R.id.fragment_container,data)
                        .commit();
            }
        });
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
        unbindService(myServiceConnection);
    }

    private final ServiceConnection myServiceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            localization = ((Localization.MyBinder) iBinder).getService();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            localization = null;
        }
    };
}
```

La méthode **bindService** permet de lier l'application et le service, les paramètres sont :

- un intent désignant le service
- une variable pour surveiller l'exécution
- la façon dont doit être exécuté le service (cf. prochain slide)

Déconnexion avec le service.

Instanciation d'une variable de type **ServiceConnection** (un ServiceConnection est une interface pour surveiller l'exécution).

Toutes les méthodes publiques du service seront alors accessibles depuis l'activité.

LES SERVICES

Les drapeaux passés en paramètre de la méthode bindService :

- **0** : aucun paramètre
- **Context.BIND_AUTO_CREATE** : service est automatiquement créé lorsque la liaison est définie
- **Context.BIND_DEBUG_UNBIND** : informations de débogages sont fournies au développeur
- **BIND_NOT_FOREGROUND**, **BIND_ABOVE_CLIENT**, **BIND_WAIVE_PRIORITY** : permettent de spécifier l'importance du service

```java
public class MainActivity extends AppCompatActivity {

    private Button get_position;
    private Localization localization;
    Intent intentService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        get_position = findViewById(R.id.get_position);
        intentService = new Intent(this, Localization.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
        bindService(intentService,myServiceConnection, Context.BIND_AUTO_CREATE);
        get_position.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle bundle = new Bundle();
                bundle.putString("localization",localization.getLocalization());
                Data data = new Data();
                data.setArguments(bundle);
                getSupportFragmentManager().beginTransaction()
                        .add(R.id.fragment_container,data)
                        .commit();
            }
        });
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
        unbindService(myServiceConnection);
    }

    private final ServiceConnection myServiceConnection = new ServiceConnectio
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder
            localization = ((Localization.MyBinder) iBinder).getService();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            localization = null;
        }
    };

}
```

Data.java

```java
public class Data extends Fragment {

    private TextView data;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        String localization = getArguments().getString("localization");
        View view = inflater.inflate(R.layout.fragment_data, container, false);
        data = view.findViewById(R.id.data);
        data.setText(localization);
        return view;
    }

}
```

fragment_data.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".Data">

  <TextView
    android:id="@+id/data"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

</FrameLayout>
```

```java
public class Localization extends Service {

    private String localization;
    private int latitude = 37;
    private Handler handler;
    private Runnable runnable;

    public class MyBinder extends Binder {
        Localization getService() {
            return Localization.this;
        }
    }

    private final MyBinder myBinder = new MyBinder();

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

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        handler = new Handler();
        runnable = new Runnable() {
            public void run() {
                latitude+=1;
                localization = "("+latitude +" ; -122)";
                handler.postDelayed(this, 4000);
            }
        };
        handler.post(runnable);
        return START_STICKY;
    }

    public String getLocalization() {
        return localization;
    }
}
```

```
public class Localization extends Service {

    private String localization;
    private int latitude = 37;
    private Handler handler;
    private Runnable runnable;

    public class MyBinder extends Binder {
        Localization getService() {
            return Localization.this;
        }
    }

    private final MyBinder myBinder = new MyBinder();

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

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        handler = new Handler();
        runnable = new Runnable() {
            public void run() {
                latitude+=1;
                localization = "("+latitude +" ; -122)";
                handler.postDelayed(this, 4000);
            }
        };
        handler.post(runnable);
        return START_STICKY;
    }

    public String getLocalization() {
        return localization;
    }
}
```

Un **IBinder** permet de faire le pont entre le service et l'activité.

Cela permet de pouvoir mettre à jour une activité.

Il faut créer une classe héritant de **Binder**, puis il faut créer une instance de cette classe.

La méthode **onBind** renverra l'instance.

NB : la méthode onBind est appelée automatiquement à chaque demande d'établissement d'une connexion. Cette méthode reçoit en paramètre l'objet intent fourni à la méthode bindService et retourne un objet implémentant l'interface IBinder permettant la communication avec le service.

```java
public class Localization extends Service {

    private String localization;
    private int latitude = 37;
    private Handler handler;
    private Runnable runnable;

    public class MyBinder extends Binder {
        Localization getService() {
            return Localization.this;
        }
    }

    private final MyBinder myBinder = new MyBinder();

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

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        handler = new Handler();
        runnable = new Runnable() {
            public void run() {
                latitude+=1;
                localization = "("+latitude +" ; -122)";
                handler.postDelayed(this, 4000);
            }
        };
        handler.post(runnable);
        return START_STICKY;
    }

    public String getLocalization() {
        return localization;
    }
}
```

La méthode **onStartCommand** est appelée automatiquement à chaque fois qu'un client lance le service en utilisant la méthode startService.

La méthode retourne un entier servant à spécifier le comportement du service :

START_STICKY si le service est tué par le système il sera automatiquement relancé si des ressources sont disponibles.

START_NOT_STICKY ici il ne sera pas automatiquement relancé

START_REDELIVER_INTENT il sera relancé automatiquement en ayant pour argument l'intent précédent du service.

```java
public class Localization extends Service {

    private String localization;
    private int latitude = 37;
    private Handler handler;
    private Runnable runnable;

    public class MyBinder extends Binder {
        Localization getService() {
            return Localization.this;
        }
    }

    private final MyBinder myBinder = new MyBinder();

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

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        handler = new Handler();
        runnable = new Runnable() {
            public void run() {
                latitude+=1;
                localization = "("+latitude +" ; -122)";
                handler.postDelayed(this, 4000);
            }
        };
        handler.post(runnable);
        return START_STICKY;
    }

    public String getLocalization() {
        return localization;
    }
}
```

La classe Handler permet d'envoyer des messages au thread depuis celui dont il a été créé. Cela permet aussi de planifier le traitement de certaines tâches.

LES SERVICES

Résumé des méthodes d'un service

onCreate : appelée automatiquement lors de la création du service (que ce soit via la méthode startService ou la méthode bindService)

onStartCommand : appelée automatiquement à chaque fois qu'un client lance le service en utilisant la méthode startService

onBind : appelée automatiquement à chaque demande d'établissement d'une connexion

onUnbind : appelée automatiquement à chaque fermeture d'une connexion

onDestroy : appelée automatiquement par le système avant la suppression du service

LES SERVICES

Si nous avions voulu lancer un service depuis l'activité et que cela soit lui qui envoie des informations à l'activité il aurait fallu utiliser un **BroadcastReceiver**.

Un BroadcastReceiver permet de réceptionner des événements pour, lorsqu'il reçoit un message, déclencher une action.

```java
public class Main3Activity extends AppCompatActivity {

    private Intent intentService;
    static public String BROADCAST = "com.cfc.cours.event";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        intentService = new Intent(this,MyService.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
    }

    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(receiver,new IntentFilter(BROADCAST));
    }

    @Override
    public void onPause() {
        super.onPause();
        unregisterReceiver(receiver);
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
    }

    private BroadcastReceiver receiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Toast.makeText(getApplicationContext(),
                        bundle.getString("message"),
                        Toast.LENGTH_LONG).show();
            }
        }
    };
}
```

```java
public class Main3Activity extends AppCompatActivity {

    private Intent intentService;
    static public String BROADCAST = "com.cfc.cours.event";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        intentService = new Intent(this,MyService.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
    }

    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(receiver,new IntentFilter(BROADCAST));
    }

    @Override
    public void onPause() {
        super.onPause();
        unregisterReceiver(receiver);
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
    }

    private BroadcastReceiver receiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Toast.makeText(getApplicationContext(),
                        bundle.getString("message"),
                        Toast.LENGTH_LONG).show();
            }
        }
    };
}
```

Permettre à l'activité d'écouter les changements envoyés par le service.

Arrêter l'écoute du service.

```java
public class Main3Activity extends AppCompatActivity {

    private Intent intentService;
    static public String BROADCAST = "com.cfc.cours.event";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        intentService = new Intent(this,MyService.class);
    }

    @Override
    protected void onStart() {
        super.onStart();
        startService(intentService);
    }

    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(receiver,new IntentFilter(BROADCAST));
    }

    @Override
    public void onPause() {
        super.onPause();
        unregisterReceiver(receiver);
    }

    @Override
    protected void onStop() {
        super.onStop();
        stopService(intentService);
    }

    private BroadcastReceiver receiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Toast.makeText(getApplicationContext(),
                        bundle.getString("message"),
                        Toast.LENGTH_LONG).show();
            }
        }
    };
}
```

La méthode **onReceive** est appelée automatiquement lors de la réception d'un événement. Elle reçoit en paramètres le contexte applicatif et l'événement de type Intent reçu.

```java
public class MyService extends Service {

    public class MyBinder extends Binder {
        MyService getService() {
            return MyService.this;
        }
    }

    private final MyBinder myBinder = new MyBinder();

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

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Handler handler = new Handler();

        handler.postDelayed(new Runnable() {
            public void run() {
                Intent intent = new Intent(Main3Activity.BROADCAST);
                intent.putExtra("message", "Hello!");
                sendBroadcast(intent);
            }
        },5000);

        return START_STICKY;
    }
}
```

La méthode **sendBroadcast** permet d'envoyer un événement en diffussant un intent aux récepteurs.

LES SERVICES

Dans le cas où il n'y aurait pas d'interaction avec l'utilisateur (téléchargement d'un fichier, …), ni de sollicitation fréquente au service, vous pourrez utiliser la classe **IntentService** qui implémente nativement la création d'un thread pour l'exécution de la tâche et gère l'arrêt du service.

LES SERVICES

```
package com.cfc.service;

import android.app.IntentService;
import android.content.Intent;

public class MyIntentService extends IntentService {
    public MyIntentService() {
        super("MyIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {

        // traitement

    }
}
```

```
public class Main2Activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this,MyIntentService.class);
        startService(intent);
    }

}
```

PM_EI5_TD4.pdf