LES TRAITEMENTS ASYNCHRONE

I) Objectif: Pour voir le blocage de l'UI THREAD

1) Creation de activity_main.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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:onClick="traitement"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="56dp"
        android:layout_marginLeft="56dp"
        android:layout_marginTop="40dp"
        android:text="envoyer"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="36dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2) Creation de MainActivity.java

```
package com.example.appservice;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
//l'execution des traitements lies a la gestion des interfaces et des interaction
//avec l'utilisateur se fait dans l'UI THREAD
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //ici on va simuler un traitement long qui va s'executer dans le THREADUI
    public void traitement(View view) {
```

```
        //Ici on simule le blocage de l'UITHREAD a cause d'un traitement long
        //Ici on gere les erreurs avec l'instruction Thread.sleep si une erreur
        //se declanche alors on va manipuler un objet de type InterruptedException
        //ici on affiche la trace de l'erreur
        try {
            Thread.sleep(15000L);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

II) Pour utiliser la classe AsynTask

1) Creer activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/bouton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="envoyer"
        android:onClick="traitement"
        />
    <ProgressBar
        style="@android:style/Widget.ProgressBar.Horizontal"
        android:id="@+id/progress"
        android:visibility="gone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

2) Creez MainActivity.java

```
package com.example.appasynchrone;
import androidx.appcompat.app.AppCompatActivity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    Button bouton;
    ProgressBar progressBar;
    MyTask myTask;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        progressBar=findViewById(R.id.progress);
        bouton=findViewById(R.id.bouton);
    }
    public void traitement(View view) {
        //On instancie un objet pour effectuer un traitement asynchrone
        myTask=new MyTask();
        //On doit passer un tableau de String (car c'est le type définie dans
doInBackGround
        //a la tache asynchrone
        String[] url={"url1","url2"};
        //on va lancer l'execution de la tache asynchrone
        myTask.execute(url);
    }
    //Nous allons créer une classe qui va permettre d'effectuer un traitement
asynchrone:
    //0) Dans le THread UI on fera disparaitre lebouton et apparaitre la
progressbar
    //1) Dans un Thread différent du Thread UI nous allons realiser un traitement
long
    //2) A intervalle regulier le Thread communiquera avec le Thread UI pour
mettre a jour
    //la  progress bar
    //3) A la fin du traitement dans le THread celui ci communiquera les resultats
au THread UI.
    //String:specifie le type de parametre de la methode doInBackground
    //Integer:specifie le type de parametre de la methode onProgressUpdate
    //String:specifie le type de parametre de la methode onPostExecute
    public class MyTask extends AsyncTask<String,Integer,String>{
        //Dans cette methode doinbackground on ecrit les traitements longs
        //qui doivent s'executer hors du Thread UI, il est impossible
        //d'ecrire quelque chose dans un View
        //String... strings:ici on passe en parametre un tableau de String
        //on souhaite passer a la methode un tableau de chaine
        //on peut passer d'autre type (Integer...integers,
Personne...personnes,..)
        //On souhaite recuperer les valeurs du tableau de string
        //On va définir une variable chaine (resultat) que l'on va initialiser
avec
        //la concatenation des valeurs recuperes   (url1url2) qui sera
        //la valeur que l'on va traiter et transmettre a postexecute.
        //pour simuler un traitement long on va realiser une boucle (for) qui va
        //s'executer 10 fois , a chaque execution nous mettrons en sommeil le
thread
        //1 seconde et nous concatenerons les valeurs 0123456789 a la variable
        //resultat.
        //on souhaite que la barre de progression avance de 10% a chaque fois que
        //l'on fait un tour de boucle, pour cela on doit envoyer un message à la
methode
        //onProgressUpdate a chaque fois que l'on met a jour le resultat
        @Override
        protected String doInBackground(String... strings) {
```

```
            String url1=strings[0];
            String url2=strings[1];
            String resultat=url1+url2;
            for (int i=1;i<=10;i++){
                try {
                    Thread.sleep(1000L);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                resultat=resultat+i;
                //on envoie un message a onProgressUpdate
                publishProgress(i*10);
            }
            return resultat;
        }
        //Les trois methodes ci-dessous sont des methodes qui s'executent dans le
thread
        //UI
        //Cette methode onPreExecute realise un pretraitement dans le ThreadUI
avant que
        //la methode doInBackground se declanche.
        //
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            //On veut faire disparaitre le bouton et apparaitre la progressbar
            bouton.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
        }
        //cette methode recoit le resultat de la methode doInbackground quand
celle ci
        //a termine son traitement long pour l'afficher
        //Le type String est ici imposé par le type de resultat que va renvoyé la
methode
        //doInBackground
        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            //On veut afficher le resultat
            Toast.makeText(MainActivity.this,s,Toast.LENGTH_LONG).show();
            //on veut faire apparaitre le bouton et faire disparaitre la
progressbar
            bouton.setVisibility(View.VISIBLE);
            progressBar.setVisibility(View.GONE);
        }
        //cette methode recoit des messages de la methode doinbackground afin de
mettre a jour
        //l'interface graphique pendant l'execution de la tache longue.
        //Cette methode a pour parametre un tableau d'entier qui correspond au
type des
        //valeurs envoyes par doInBackground
        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            //on récupére la valeur envoye par publishprogress
```

```
            int valeur=values[0];
            //on met a jour la progressbar
            progressBar.setProgress(valeur);
        }
    }
}
```

Exercice:

On a cree le formulaire qui permet de saisir des Personnes et de les enregistrer dans la base de donnee.

A faire

Lorsque on clique sur le bouton envoyer , on souhaite que les donnees apres avoir ete enregistre dans la base sqlite, la ListView (MainActivity) s'affiche avec laliste des Personnes qui existent dans la base de donnee.

A faire de 2 Maniere:

1 methode: Avec les Thread
2 methode: Avec la classe Asynctask