ANDROID platforma perzistencija, SQL Lite multithreading

Mobilni i distribuirani informacioni sistemi

Mr Bratislav Predić

2012. godina

SQL baza na Android-u

- Android (kao i iPhone) koristi embedded nezavistan program sqlite3 za rad sa lokalnim bazama
- SQLite
  - Implementira većinu SQL-92 standarda
  - Delimično podržava trigere i omogućava kompleksne upite
  - NE IMPLEMENTIRA referencijalni integritet kroz strane ključeve
  - Koristi model slabih tipova
  - Tip podatka se dodeljuje ne koloni već svakoj ćeliji
  - Slično kao *Variant* tip u VB tako da možete upisati string u numeričku kolonu i sl.

Kreiranje SQLite baze

```
public static SQLiteDatabase.openDatabase(
        String path,
        SQLiteDatabase.CursorFactory factory,
        int flags )
```

- Parametri
  - Path – putanja do datoteke baze
  - Factory – factory klasa koja instancira kursor pri upitu ili null podrazumevano
  - Flags – kontroliše mod pristupa bazi (OPEN_READWRITE, OPEN_READONLY, CREATE_IF_NECESSARY)
  - Vraća referencu na otvorenu bazu
  - Emituje *SQLiteException*

Korišćenje SQLite baze

- Ograničenja pristupa SQLite bazi
  - Ograničenja deljenja
    Ne možete pristupati internim SQLite bazama drugih aplikacija (za to se koristi ContentProvider)
  - Datoteka baze koja je smeštena na SD kartici zahteva da Manifest uključuje

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- SQLITE kao i većina DBMS-ova nije case sensitive

Kreiranje SQLite baze #2

```
SQLiteDatabase db = this.openOrCreateDatabase(
                                    "myfriendsDB",
                                    MODE_PRIVATE,
                                    null);
```

- Podrazumevani prefiks putanje do datoteke baze je *"/data/data/<CURRENT_namespace>/databases/*
- Ovako kreiranu datoteku mogu da koriste druge aktivnosti ili se može eksportovati na desktop (mogu se koristiti svi standardni SQLite alati)
- MODE parametar:
  MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE

Korišćenje SQLite baze

- Preporučuje se rad u transakcijama
- Transakcije obezbeđuju atomičnost operacija bez obzira na abnormalni prekid aplikacije

```
db.beginTransaction();
try{
  //perform your database operations here
  //commit your changes
  db.setTransactionSuccessful();
}catch(SQLiteException e) {
  //report problem
}finally{
  db.endTransaction();
}
```

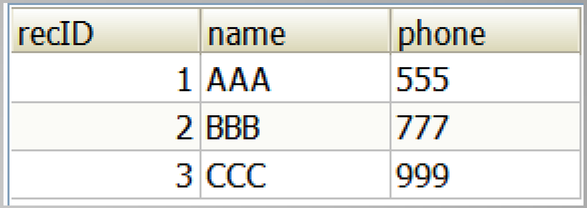
- *setTransactionSuccessful()* commit-uje transakciju, ako ne postoji, automatski rollback se radi

Izvršavanje upita

- Dva tipa upita najčešće
  - *Action query* – kreiranje, brisanje elemenata baze
  - *Retreival query* – upiti za pribavljanje podataka
- Za izvršenje action query-a koristimo metodu
  - *execSQL()*

```
db.execSQL("create table tblAMIGO(" +
       " recIDinteger PRIMARY KEY autoincrement, " +
       " name text, "+
       " phone text ); " );
db.execSQL( "insert into tblAMIGO(name, phone) values
('AAA', '555' );" );
db.execSQL( "insert into tblAMIGO(name, phone) values
('BBB', '777' );" );
db.execSQL( "insert into tblAMIGO(name, phone) values
('CCC', '999' );" );
```

Izvršavanje upita



recID	name	phone
1	AAA	555
2	BBB	777
3	CCC	999

- Polje *recID* je primarni ključ i automatski se inkrementira
- Ako tabela postoji, briše se i kreira nova
- I ovaj kod treba da bude uokviren transakcijom
- Treba obraditi i *SQLiteException*
- SQLite ima nevidljivu kolonu ***ROWID*** (recID i ROWID su praktično iste)

Izvršavanje upita

- Retrieval query su praktično SELECT upiti
- Rezultat ovih upita su uvek tabele
- Rezultujuće tabele se obrađuju korišćenjem ***kursora***
- Kursor omogućava pristup podacima u rezultujućoj tabeli *red-po-red*
- Dva načina izvršavanja SQL SELECT upita
    - *Raw query* – može da izvrši praktično bilo kakav SQL SELECT upit (outer join nije podržan)
    - *Simple query* – jednostavniji način zadavanja parametrizovanih SQL SQLECT upita nad **jednom** tabelom bez detaljnijeg znanja SQL jezika

Izvršavanje upita - RawQuery

- Primer za RawQuery

```
Cursor c1 = db.rawQuery(
        "select count(*) as Total from tblAMIGO",
        null);
```

- Upit prebrojava redove u tabeli *tblAMIGO*
- Rezultat je tabela sa jednimredom i jednom kolonom
- Kursor c1 se koristi za pristup rezultuujućoj tabeli
- Pribavljanje reda rezultata korišćenjem kursora se radi prelaskom na sledeći red
- Polje pribavljenog reda treba iskopirati u neku lokalnu promenljivu

Izvršavanje upita - RawQuery

- RawQuery može biti i parametrizovan

```
String mySQL= "select count(*) as Total "
       + " from tblAmigo"
       + " where recID> ?"
       + " and name = ?";

String[] args= {"1", "BBB"};
Cursor c1 = db.rawQuery(mySQL, args);
```

- Možemo i ručno konkatenirati upit

```
String[] args= {"1", "BBB"};
String mySQL= " select count(*) as Total "
       + " from tblAmigo"
       + " where recID> " + args[0]
       + " and name = '" + args[1] + "'";
Cursor c1 = db.rawQuery(mySQL, null);
```

Izvršavanje upita - SimpleQuery

- SimpleQuery koristi template SELECT naredbu nad jednom tabelom
- Ne zahteva pisanje SQL upita
- Metoda

```
query(String table,
      String[] columns,
      String selection,
      String[] selectionArgs,
      String groupBy,
      String having,
      String orderBy)
```

- Metoda uzima fiksnih 7 argumenata koji se ubacuju u pripremljeni SQL SELECT template

Izvršavanje upita - SimpleQuery

- Primer

```
String[] columns = {"Dno","Avg(Salary) as AVG"};
String[] conditionArgs = {"F","123456789"};
Cursorc = db.query(
              "EmployeeTable", // ime tabele
              columns, // spisak kolona
              "sex = ? And superSsn= ? " , // uslovi
              conditionArgs, // argumenti uslova
              "Dno", // grupisanje po koloni
              "Count(*) > 2", // having klauzula
              "AVG Desc‟ // uređenje rezultata
       );
```

- Primer koristi sve argumente **query** metode
- Neki argumenti mogu biti **null**

Izvršavanje upita - SimpleQuery

- Ako nisu potrebni svi elementi SQL SELECT upita

```
String [] columns = {"recID", "name", "phone"};
Cursor c1 = db.query(
    "tblAMIGO",
    columns,
    "recID> 2 and length(name) >= 3 and name like 'B%' ",
    null, null, null,
    "recID" );
```

- Ne koristimo argumente za selekciju, grupisanje niti having klauzulu
- Umesto nepotrebnih argumenata ide **null**

Kursori

- Kursori se koriste za sekvencijalni pristup tabeli koja je rezultat retrieval query-a
- Metode kursora
  - Pozicija kursora: *isFirst(), isLast(), isBeforeFirst(), isAfterLast()*
  - Navigacija po redovima: *moveToFirst(), moveToLast(), moveToNext(), moveToPrevious(), move(n)*
  - Rad sa poljima reda: *getInt(), getString(), getFloat(), getBlob(), getDate()...*
  - Rad sa šemom:
  - *getColumnName(), getColumnNames(), getColumnIndex(), getColumnCount(), getCount()*

Kursori - primer

```
String[] columns ={"recID", "name", "phone"};
Cursor myCur = db.query(
       "tblAMIGO",
       columns,
       null, null, null, null,
       "recID");
int idCol = myCur.getColumnIndex("recID");
int nameCol = myCur.getColumnIndex("name");
int phoneCol = myCur.getColumnIndex("phone");
while(myCur.moveToNext()) {
  columns[0] = Integer.toString((myCur.getInt(idCol)));
  columns[1] = myCur.getString(nameCol);
  columns[2] = myCur.getString(phoneCol);
  txtMsg.append("\n"+ columns[0] + " "
       + columns[1] + " "
       + columns[2] );
}
```

Kursori

- Kursori i modifikacija sadržaja tabela
- Kursori omogućavaju **READ_ONLY** pristup
- Ranije verzije Android SDK su nudile metode kursora za modifikaciju podataka
- Metode:
  - *cursor.updateInt(...)*
  - *cursor.deleteRow(...)*
    su ***deprecated***
- Preporuka je da se action komande izvršavaju metodom *execSQL(...)*

Modifikacija sadržaja tabele

- Postoje i metode za jednostavniju izmenu podataka koje sadrži tabela

```
public long insert(String table,
                    String nullColumnHack,
                    ContentValues values )
public int update(String table,
                    ContentValuesvalues,
                    String whereClause,
                    String[] whereArgs)
public int delete(String table,
                    String whereClause,
                    String[] whereArgs)
```

- Drugi argument Insert naredbe je nullable kolona kada se ubacuje **null** values

Modifikacija sadržaja tabele

- Insert metoda - primer

```
ContentValues initialValues= new ContentValues();
initialValues.put("name", "ABC");
initialValues.put("phone", "101");
Int rowPosition = (int) db.insert("tblAMIGO",
                                  null,
                                  initialValues);
initialValues.put("name", "DEF");
initialValues.put("phone", "202");
rowPosition = (int) db.insert("tblAMIGO",
                                  null,
                                  initialValues);
initialValues.clear();
rowPosition = (int) db.insert("tblAMIGO",
                                  null,
                                  initialValues);
rowPosition= (int) db.insert("tblAMIGO",
                                  "name",
                                  initialValues);
```

Modifikacija sadržaja tabele

- Update metoda – primer

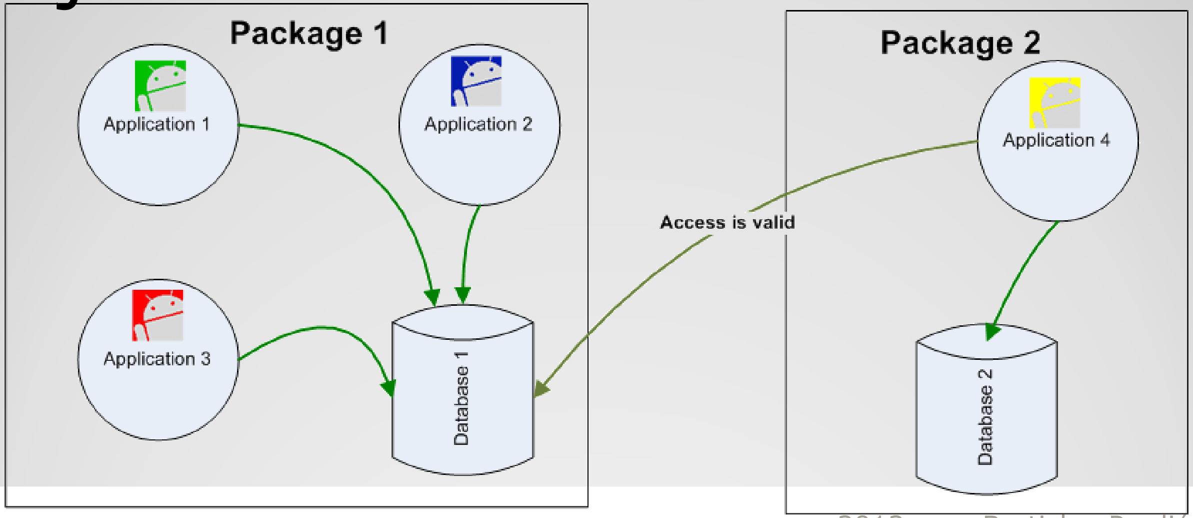
```
1.String [] whereArgs= {"2", "7"};
2.ContentValues updValues = new ContentValues();
3.updValues.put("name", "Maria");
4.Int recAffected=db.update( "tblAMIGO",
                           updValues,
                           "recID > ? and recID < ?",
                           whereArgs);
```

- Delete metoda - primer

```
1. String [] whereArgs= {"2", "7"};
2. recAffected= db.delete("tblAMIGO",
                           "recID> ? and recID< ?",
                           whereArgs);
```

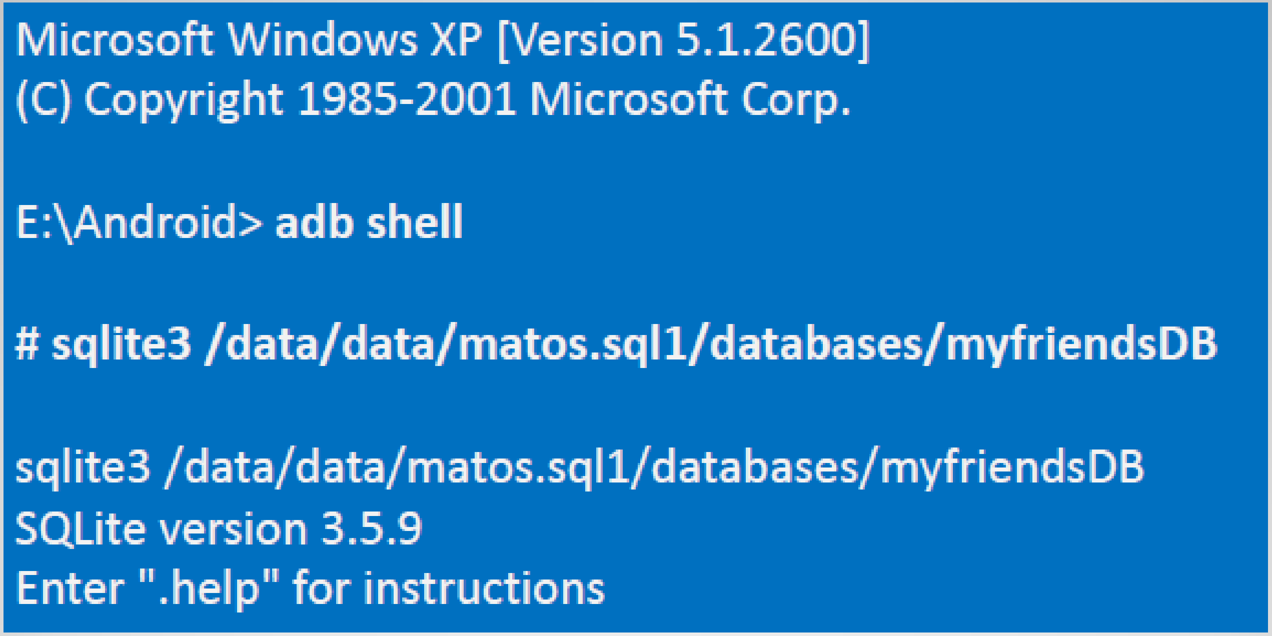
Vidljivost baza

- Bilo koja aplikacija može pristupiti eksetnoj datoteci SQLite baze koja je smeštena na SD kartici
- Samo je potrebno znati putanju do datoteke baze
- Preporučuje se deljenje podataka između aplikacijama ContentProvider mehanizmom



SQLite - Commandline alati

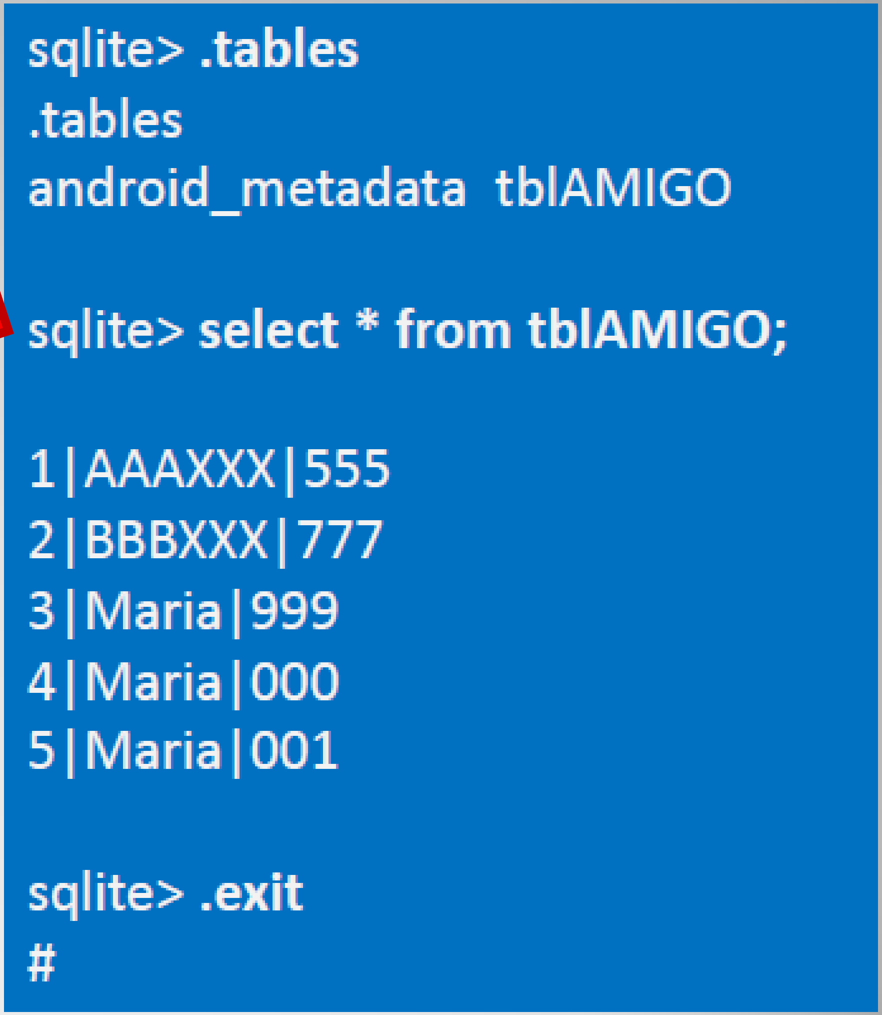
- Android SDK ima command line alat za rad sa SQLite bazama
- Prvo je potrbno povezati se na shell emulatora
  - **adb shell**



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

E:\Android> adb shell

# sqlite3 /data/data/matos.sql1/databases/myfriendsDB

sqlite3 /data/data/matos.sql1/databases/myfriendsDB
SQLite version 3.5.9
Enter ".help" for instructions
```

SQLite - Commandline alati

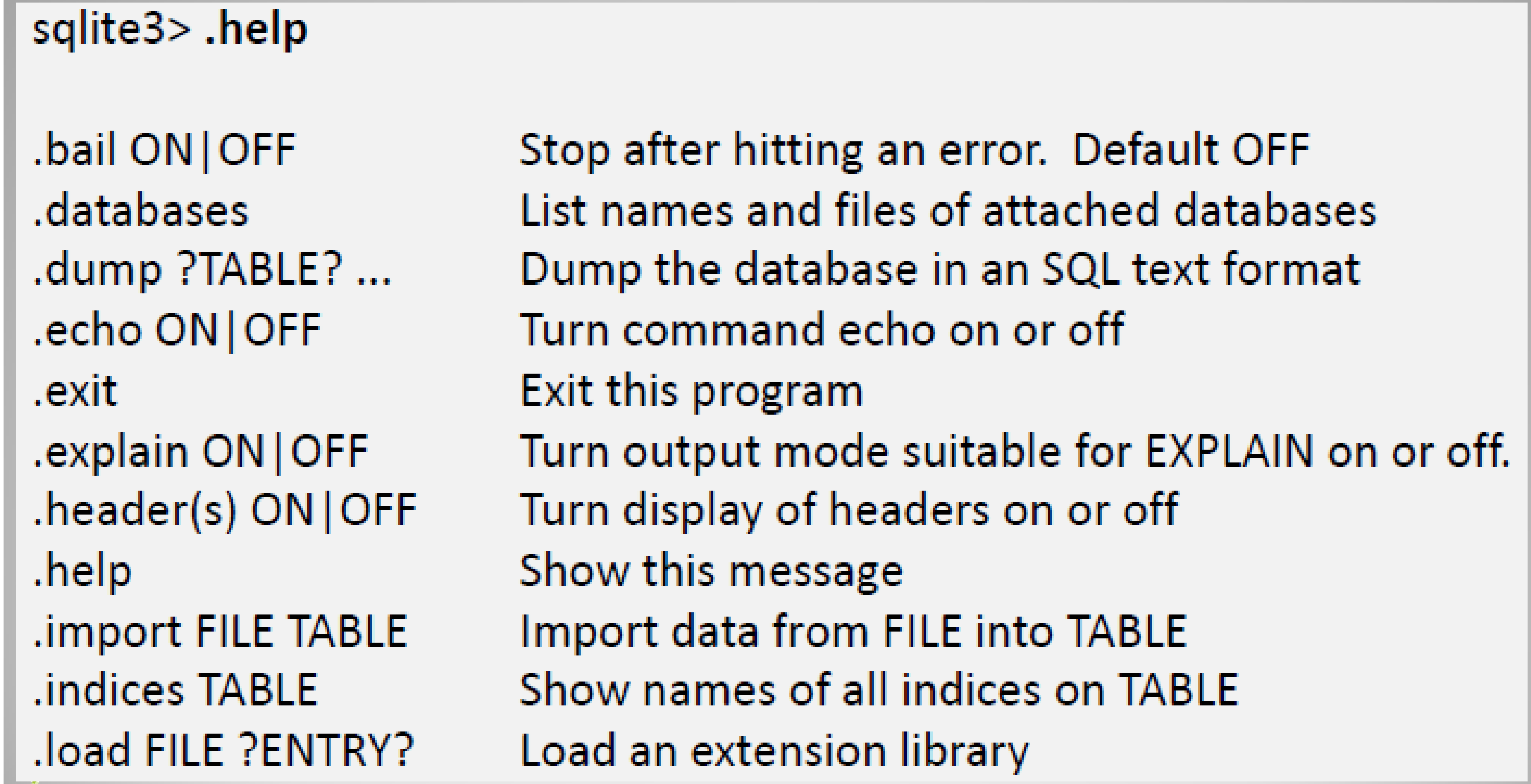
- sqlite3 alat podržava standardne DBMS komande
- Prikaz tabela
- Izvršavanje upita



```
sqlite> .tables
.tables
android_metadata  tblAMIGO

sqlite> select * from tblAMIGO;

1|AAAXXX|555
2|BBBXXX|777
3|Maria|999
4|Maria|000
5|Maria|001

sqlite> .exit
#
```

SQLite - Commandline alati

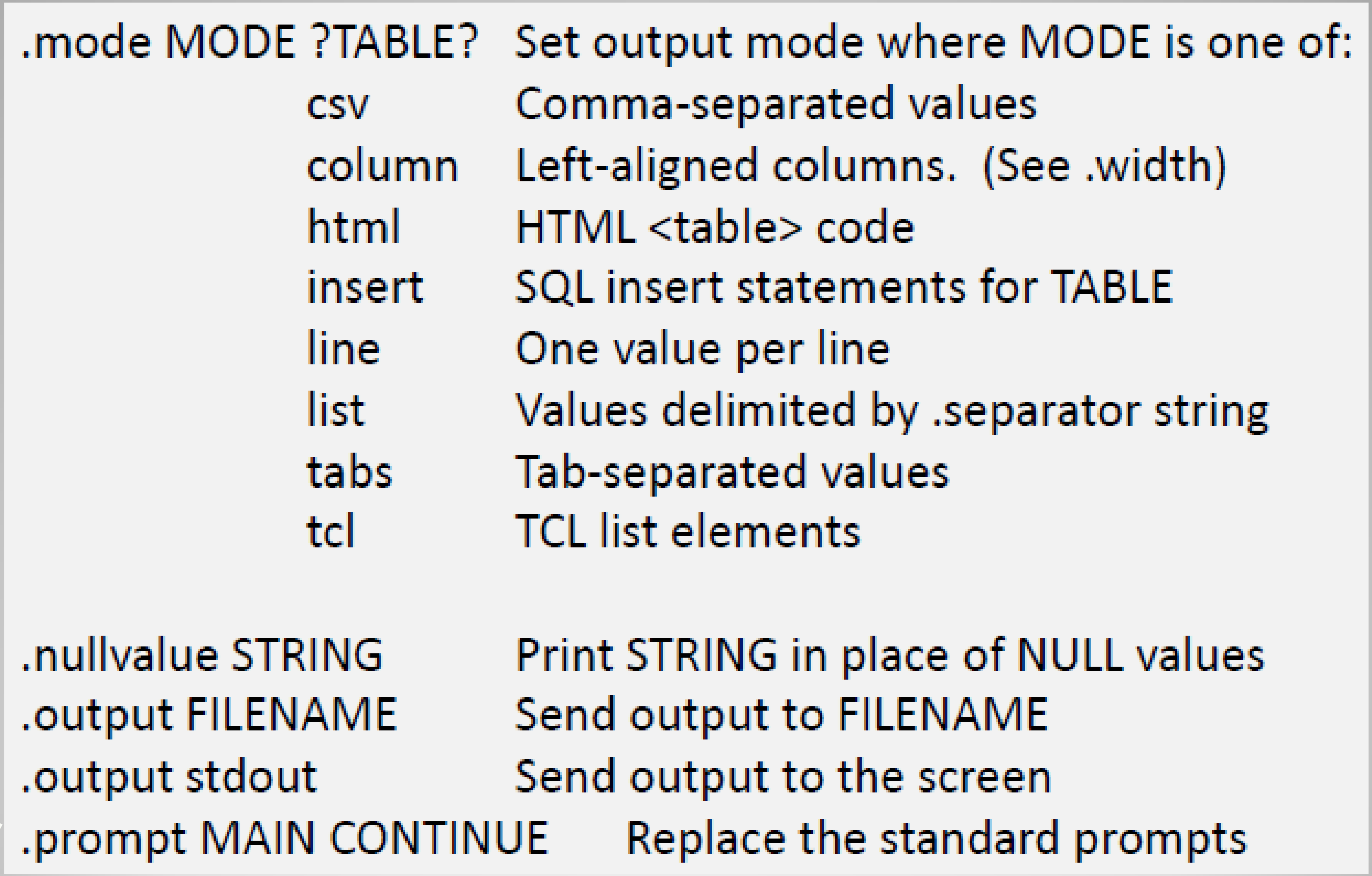
- sqlite3 komande



```
sqlite3> .help

.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
.dump ?TABLE? ...      Dump the database in an SQL text format
.echo ON|OFF           Turn command echo on or off
.exit                  Exit this program
.explain ON|OFF        Turn output mode suitable for EXPLAIN on or off.
.header(s) ON|OFF      Turn display of headers on or off
.help                  Show this message
.import FILE TABLE     Import data from FILE into TABLE
.indices TABLE         Show names of all indices on TABLE
.load FILE ?ENTRY?     Load an extension library
```

SQLite - Commandline alati

- sqlite3 komande



```
.mode MODE ?TABLE?  Set output mode where MODE is one of:
          csv       Comma-separated values
          column    Left-aligned columns.  (See .width)
          html      HTML <table> code
          insert    SQL insert statements for TABLE
          line      One value per line
          list      Values delimited by .separator string
          tabs      Tab-separated values
          tcl       TCL list elements

.nullvalue STRING   Print STRING in place of NULL values
.output FILENAME    Send output to FILENAME
.output stdout      Send output to the screen
.prompt MAIN CONTINUE   Replace the standard prompts
```

SQLite - Commandline alati

- sqlite3 komande

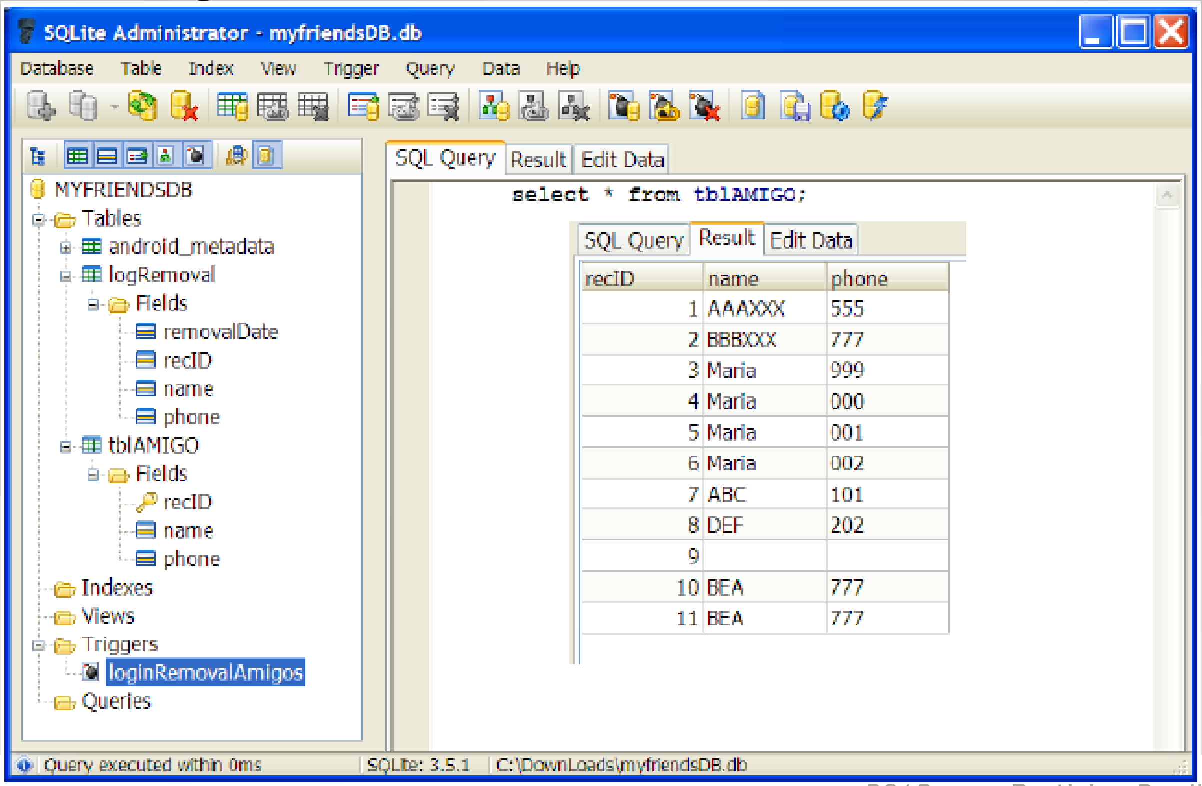
.quit	Exit this program
.read FILENAME	Execute SQL in FILENAME
.schema ?TABLE?	Show the CREATE statements
.separator STRING	Change separator used by output mode and .import
.show	Show the current values for various settings
.tables ?PATTERN?	List names of tables matching a LIKE pattern
.timeout MS	Try opening locked tables for MS milliseconds
.width NUM NUM ...	Set column widths for "column" mode

- SQLite GUI alati
  - Datoteka SQLite baze može da se prebaci na/sa emulatora standardnim komandama

```
adb pull <full_path_to_database>
adb push <full_path_to_database>
```

SQLite – GUI alati alati

- Kada se SQLite datoteka baze prebaci na desktop jednostavnija je manipulacija nekim GUI alatom
- Recimo SQLite Administrator



```
select * from tblAMIGO;
```

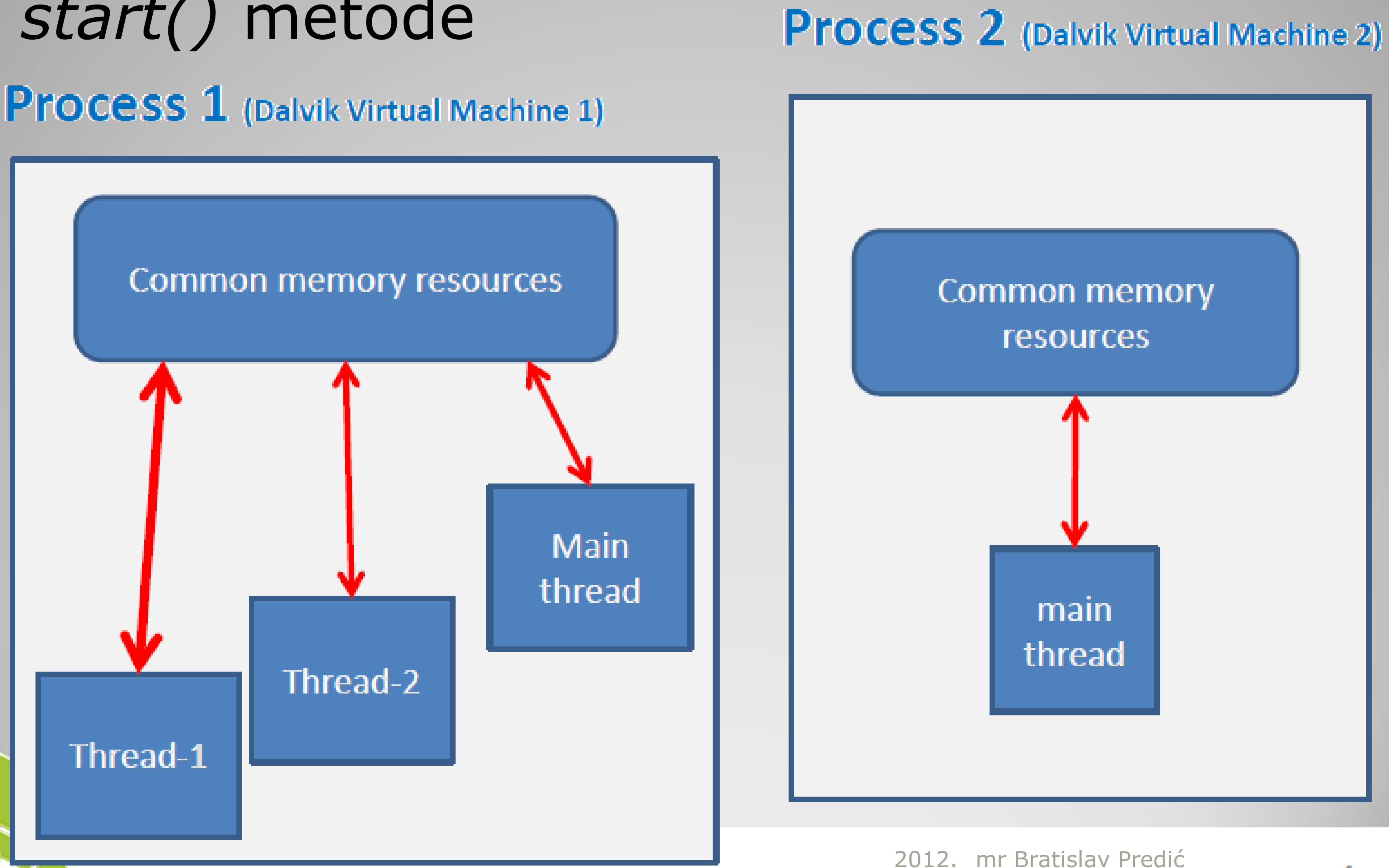
recID	name	phone
1	AAAXXX	555
2	BBBXXX	777
3	Maria	999
4	Maria	000
5	Maria	001
6	Maria	002
7	ABC	101
8	DEF	202
9		
10	BEA	777
11	BEA	777

Android multithreading

- Na Android platformi svaka aplikacija se izvršava u svojoj virtuelnoj mašini (jednom procesu) i barem jednom thread-u
- Proces može da startuje i dodatne thread-ove po potrebi
- Na Android platformi thread-ovi mogu deliti objekte, a sinhronizuju se korišćenjem monitora koji su vezani za te objekte
- Dva načina da thread izvrši neki kod
  - Nova klasa nasledi *Thread* klasu i override-uje metodu *run()*
  - Instancirati *Thread* klasu i konstruktoru proslediti *Runnable* objekat

Android multithreading

- U svakom slučaju se thread startuje pozivom *start()* metode



Android multithreading

- Glavni thread se obično naziva UI thread i ne sme dugo biti zauzet nekom operacijom kako bi UI imao odziv
- Dva načina da se ovo obezbedi
  - Duge operacije mogu da se obavljaju u pozadinskom servisu
  - Duge operacije mogu da se prebace u poseban thread
- Komunikacija između Android thread-ova se obavlja
  - Korišćenjem *Handler* objekata
  - Slanjem *Runnable* objekata glavnoom view-u

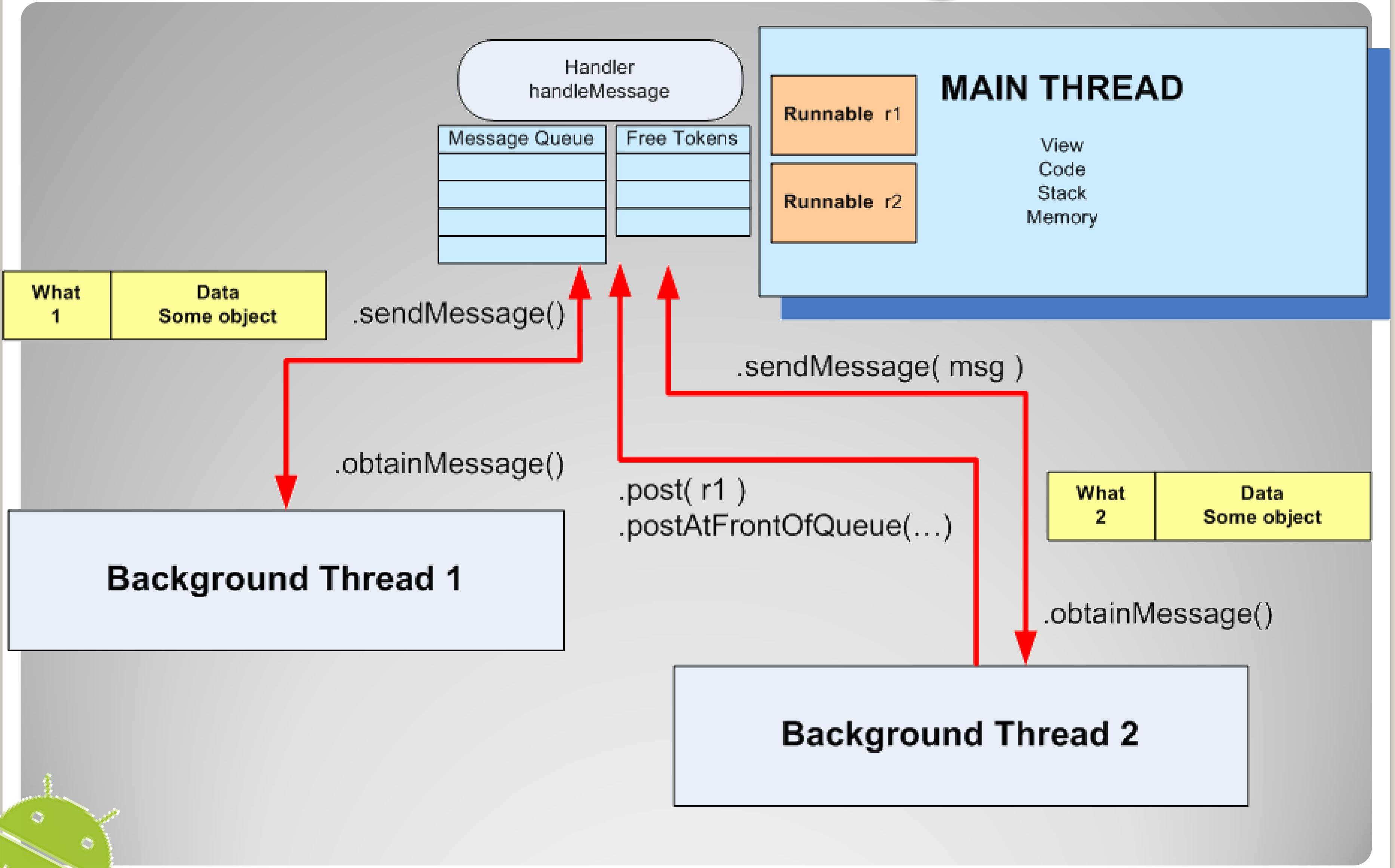
Android multithreading - Handler

- Kada se kreira proces za aplikaciju u glavnom thread-u se izvršava *message queue* koji upravlja aktivnostima, intent receiver-ima itd.
- Svaki novi thread može da komunicira sa glavnim korišćenjem *Handler-a*
- Kada se kreira *Handler* vezuje se za *message queue* thread-a koji ga je kreirao i isporučuje i preuzima poruke i runnables tom redu
- Handler se koristi
  - Da se zakaže izvršenje poruke ili runnable u budućnosti
  - Da se akcija izvrši u drugom thread-u

Android multithreading - Handler

- Handler i message queue
- Sekundarni thread koji želi da komunicira sa glavnim prvo zahteva message token sa
  - *obtainMessage()*
- Zatim može da popuni message token nekim podacima i da ga doda u message queue sa
  - *sendMessage()*
- Handler u metodi *handleMessage()* u glavnom thread-u obradjuje poruke
- na ovaj način glavnom thread-u može da prenese neke podatke ili da zatraži izvršenje runnable objekta metodom *post()*

Android multithreading - Handler



Android multithreading - Handler

- Handler primer

Main Thread

```
...
Handler myHandler = new Handler() {

  @Override
  public void handleMessage(Message msg) {

     // do something with the message...
     // update GUI if needed!
     ...
  }//handleMessage

};//myHandler

...
```

Background Thread

```
...
Thread backgJob = new Thread (new Runnable (){

   @Override
   public void run() {
    //...do some busy work here ...
    //get a token to be added to
     //the main's message queue
    Message msg = myHandler.obtainMessage();
      ...
     //deliver message to the
    //main's message-queue
    myHandler.sendMessage(msg);
   }//run

});//Thread

//this call executes the parallel thread
backgroundJob.start();
...
```

Android multithreading - Handler

- Post primer

Main Thread

```
...
Handler     myHandler = new Handler();
@Override
public void onCreate(
          Bundle savedInstanceState) {
  ...
 Thread myThread1 =
         new Thread(backgroundTask,
                    "backAlias1");
 myThread1.start();

}//onCreate

...
//this is the foreground runnable
private Runnable foregroundTask
   = new Runnable() {
  @Override
  public void run() {
   // work on the UI if needed
}
...
```

Background Thread

```
// this is the "Runnable" object
// that executes the background thread

private Runnable backgroundTask
                  = new Runnable () {
  @Override
  public void run() {
     ... Do some background work here
    myHandler.post(foregroundTask);

  }//run

};//backgroundTask
```

Android multithreading - Handler

- Post primer

Main Thread

```
 ...
Handler     myHandler = new Handler();
@Override
public void onCreate(
          Bundle savedInstanceState) {
  ...
 Thread myThread1 =
         new Thread(backgroundTask,
                    "backAlias1");
 myThread1.start();

}//onCreate

 ...
//this is the foreground runnable
private Runnable foregroundTask
   = new Runnable() {
  @Override
  public void run() {
   // work on the UI if needed
}
 ...
```

Background Thread

```
// this is the "Runnable" object
// that executes the background thread

private Runnable backgroundTask
                 = new Runnable () {
  @Override
  public void run() {
    ... Do some background work here
    myHandler.post(foregroundTask);

  }//run

};//backgroundTask
```

Slanje poruka

- Poruka se uzima od Handler-a sa

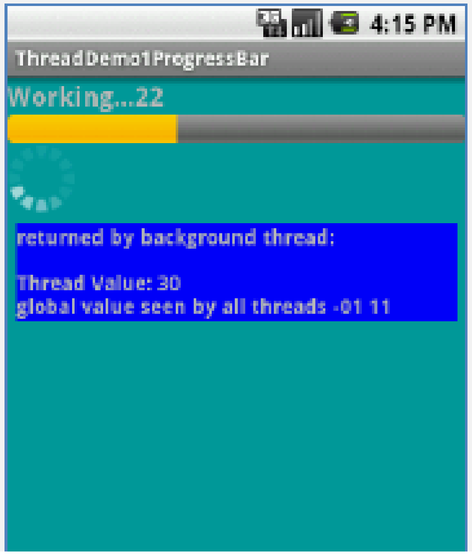
```
String localData = "Greeting from thread 1";
Message mgs = myHandler.obtainMessage (1, localData);
```

- Poruke se vraćaju u red sa *sendMessage()*
  - **sendMessage()** – dodaje poruku na kraj reda
  - **sendMessageAtFrontOfQueue()** – ubacuje poruku na početak reda
  - **sendMessageAtTime()** – poslaće poruku u red u konkretno vreme na osnovu broja millisekundi u odnosu na *SystemClock.UptimeMillis()*
  - **sendMessageDelayed()** – poslaće poruku posle zadatog broja milisekundi

Primer – progress bar

- Linearni i cirkularni progress bar widget-i prikazuju napredak sekundarnog thread-a
- Neki random podaci se šalju iz threada

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff009999"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
      android:id="@+id/TextView01"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Working ....."
      android:textSize="18sp"
      android:textStyle="bold"  />
<ProgressBar
      android:id="@+id/progress"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      style="?android:attr/progressBarStyleHorizontal" />

<ProgressBar
      android:id="@+id/progress2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
<TextView
      android:id="@+id/TextView02"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="returned from thread..."
      android:textSize="14sp"
      android:background="#ff0000ff"
      android:textStyle="bold"
      android:layout_margin="7px"/>

</LinearLayout>
```



Working...22

returned by background thread:

Thread Value: 30
global value seen by all threads -01 11

Primer – progress bar

```
public class ThreadDemo1ProgressBar extends Activity {
  ProgressBar bar1;
  ProgressBar bar2;
  TextView msgWorking;
  TextView msgReturned;
  boolean isRunning = false;
  final int MAX_SEC = 60;
  String strTest = "global value seen by all threads ";
  int intTest = 0;

  Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
      String returnedValue = (String)msg.obj;
      msgReturned.setText("returned by background thread: \n\n"
                          + returnedValue);
      bar1.incrementProgressBy(2);
      if (bar1.getProgress() == MAX_SEC){
        msgReturned.setText("Done \n back thread has been stopped");
       isRunning = false;
      }
```

Primer – progress bar

```
      if (bar1.getProgress() == bar1.getMax()){
        msgWorking.setText("Done");
        bar1.setVisibility(View.INVISIBLE);
        bar2.setVisibility(View.INVISIBLE);
        bar1.getLayoutParams().height = 0;
        bar2.getLayoutParams().height = 0;
      } else {
        msgWorking.setText("Working..." +
        bar1.getProgress() );
      }
    }
  }; //handler

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    bar1 = (ProgressBar) findViewById(R.id.progress);
    bar2 = (ProgressBar) findViewById(R.id.progress2);
    bar1.setMax(MAX_SEC);
    bar1.setProgress(0);
    msgWorking = (TextView)findViewById(R.id.TextView01);
    msgReturned = (TextView)findViewById(R.id.TextView02);
    strTest += "-01"; // slightly change the global string
    intTest = 1;
  }//onCreate
```

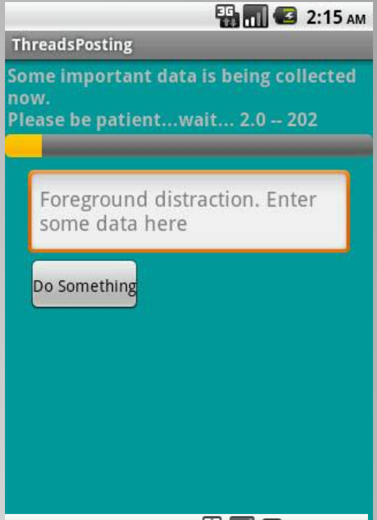
Primer – progress bar

```
  public void onStop() {
    super.onStop();
    isRunning = false;
  }
  public void onStart() {
    super.onStart();
    Thread background = new Thread(new Runnable() {
      public void run() {
        try {
          for (int i = 0; i < MAX_SEC && isRunning; i++) {
            //try a Toast method here (will not work!)
            Thread.sleep(1000); //one second at a time
            Random rnd = new Random();
            String data = "Thread Value: " + (int) rnd.nextInt(101);
            data += "\n" + strTest + " " + intTest;
            intTest++;
            Message msg = handler.obtainMessage(1, (String)data);
            if (isRunning) {
              handler.sendMessage(msg);
            }
          }
        } catch (Throwable t) {}
      }//run
    });//background
    isRunning = true;
    background.start();
  }//onStart
} //class
```

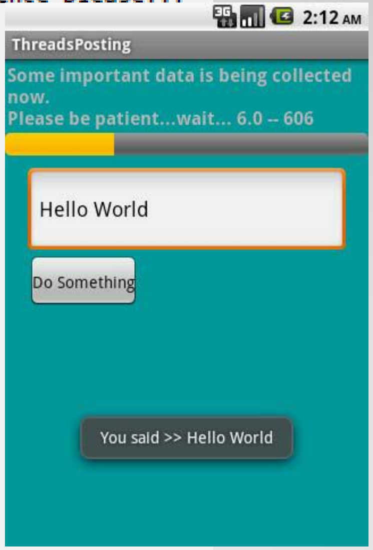
Primer – post() metoda

- Način da se izvrše foreground runnable

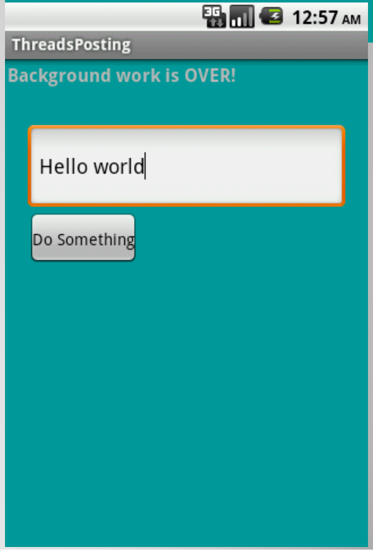
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       android:id="@+id/linearLayout1"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:background="#ff009999"
       android:orientation="vertical"
       xmlns:android=http://schemas.android.com/apk/res/android >
<TextView
       android:id="@+id/lblTopCaption"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:padding="2px"
       android:text="Some important data is being collected now. Patience please..."
       android:textSize="16sp"
       android:textStyle="bold"  />
<ProgressBar
       android:id="@+id/myBar"
       style="?android:attr/progressBarStyleHorizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content" />
<EditText
       android:id="@+id/txtBox1"
       android:layout_width="fill_parent"
       android:layout_height="78px"
       android:layout_marginLeft="20px"
       android:layout_marginRight="20px"
       android:textSize="18sp" android:layout_marginTop="10px"  />
<Button
       android:id="@+id/btnDoSomething"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:padding="4px"
       android:layout_marginLeft="20px"
       android:text="Do Something" />
</LinearLayout>
```



Some important data is being collected now.
Please be patient...wait... 2.0 -- 202



Some important data is being collected now.
Please be patient...wait... 6.0 -- 606



Background work is OVER!

Primer – post() metoda

```
public class ThreadsPosting extends Activity {
  ProgressBar myBar;
  TextView lblTopCaption;
  EditText txtBox1;
  Button btnDoSomething;
  int globalVar = 0; // to be used by threads to exchange data
  int accum = 0;
  long startingMills = System.currentTimeMillis();
  boolean isRunning = false;
  String PATIENCE = "Some important data is being collected now. " +
               "\nPlease be patient...wait... ";
  Handler myHandler = new Handler();

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    lblTopCaption = (TextView)findViewById(R.id.lblTopCaption);
    myBar = (ProgressBar) findViewById(R.id.myBar);
    myBar.setMax(100); // range goes from 0..100
    txtBox1 = (EditText) findViewById(R.id.txtBox1);
    txtBox1.setHint("Foreground distraction. Enter some data here");
```

Primer – post() metoda

```
    btnDoSomething = (Button)findViewById(R.id.btnDoSomething);
    btnDoSomething.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
        Editable txt = txtBox1.getText();
        Toast.makeText(getBaseContext(),
               "You said >> " + txt, 1).show();
      }//onClick
    });//setOnClickListener
  }//onCreate

  @Override
  protected void onStart() {
    super.onStart();
    Thread myThreadBack = new Thread(backgroundTask, "backAlias1" );
    myThreadBack.start();
    myBar.incrementProgressBy(0);
  }
```

Primer – post() metoda

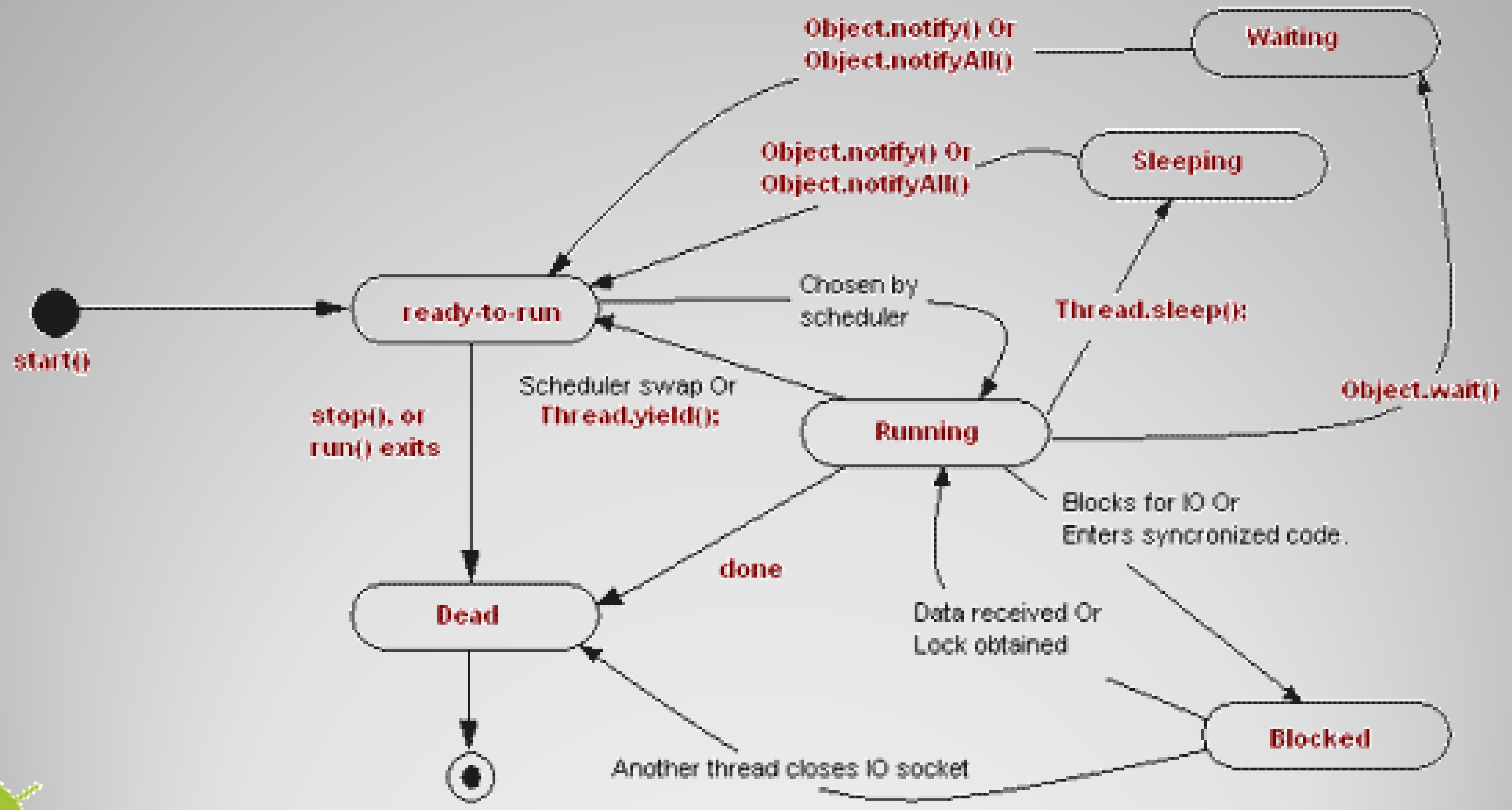
```
private Runnable foregroundTask = new Runnable() {
  @Override
  public void run() {
    try {
      int progressStep = 5;
      double totalTime = (System.currentTimeMillis() -
                     startingMills)/1000;
      synchronized(this) {
        globalVar += 100;
      };
      lblTopCaption.setText(PATIENCE + totalTime + " -- " +
                     globalVar);
      myBar.incrementProgressBy(progressStep);
      accum += progressStep;
      if (accum >= myBar.getMax()){
        lblTopCaption.setText("Background work is OVER!");
        myBar.setVisibility(View.INVISIBLE);
      }
    } catch (Exception e) {
      Log.e("<<foregroundTask>>", e.getMessage());
    }
  }
}; //foregroundTask
```

Primer – post() metoda

```
//this is the "Runnable" object that executes the background thread
  private Runnable backgroundTask = new Runnable () {
    @Override
    public void run() {
      try {
        for (int n=0; n<20; n++) {
          Thread.sleep(1000);
          synchronized(this) {
            globalVar += 1;
          };
          myHandler.post(foregroundTask);
        }
      } catch (InterruptedException e) {
        Log.e("<<foregroundTask>>", e.getMessage());
      }
    }//run
  };//backgroundTask
}//ThreadsPosting
```

Stanja thread-a

- Android thread-ovi imaju ista stanja kao i standardni Java thread-ovi

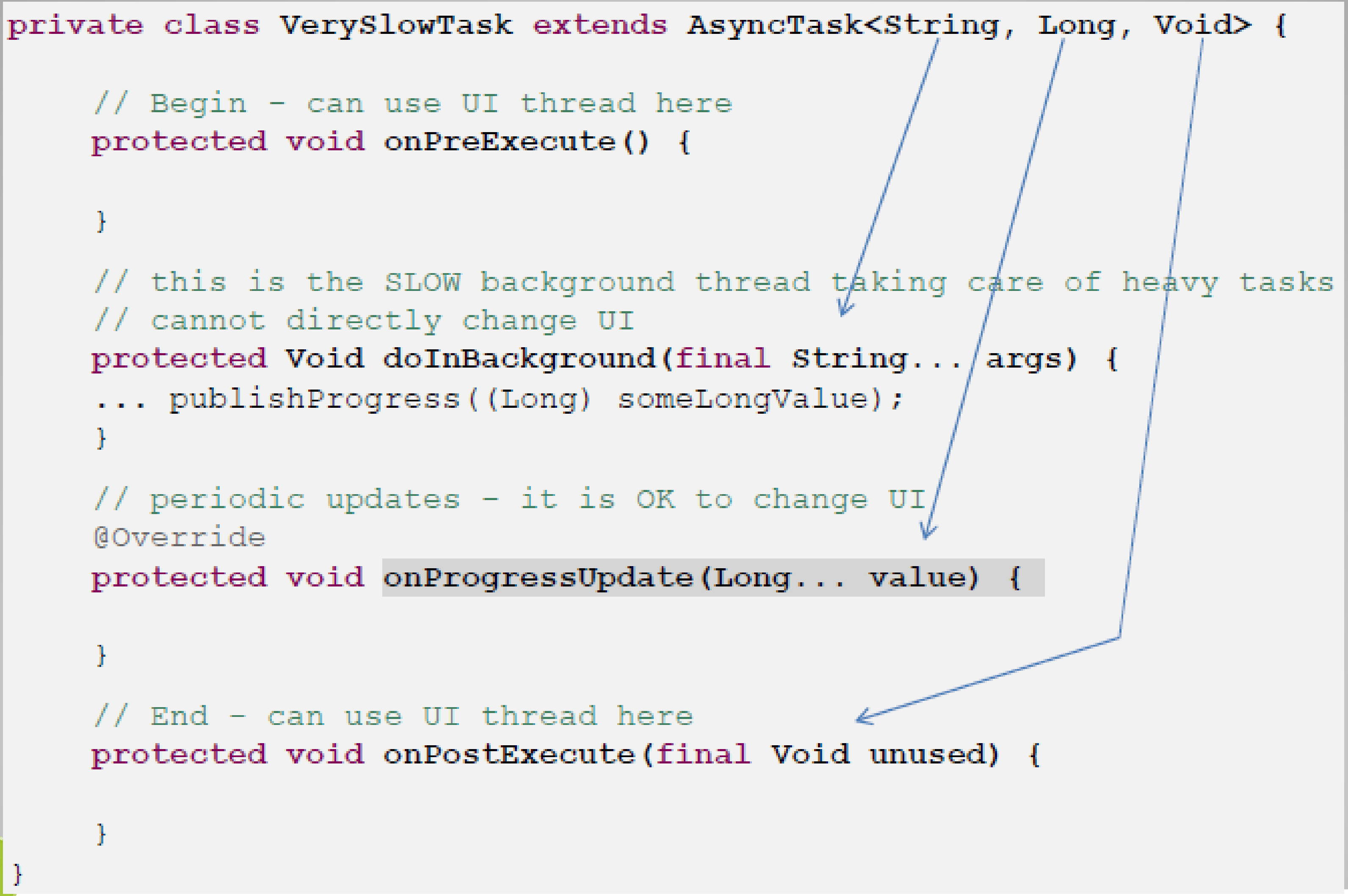


AsyncTask

- AsyncTask je čistija alternativa bez Handlera i poruka
- AsyncTask predstavlja neku operaciju koja će se obaviti u pozadinskom thread-u i čiji će se rezultat predstaviti u UI thread-u
- AsyncTask ima 4 koraka
  - onPreExecute
  - doInBackground
  - onProgressUpdate
  - onPostExecute
- AsyncTask se uvek nasleđuje

AsyncTask

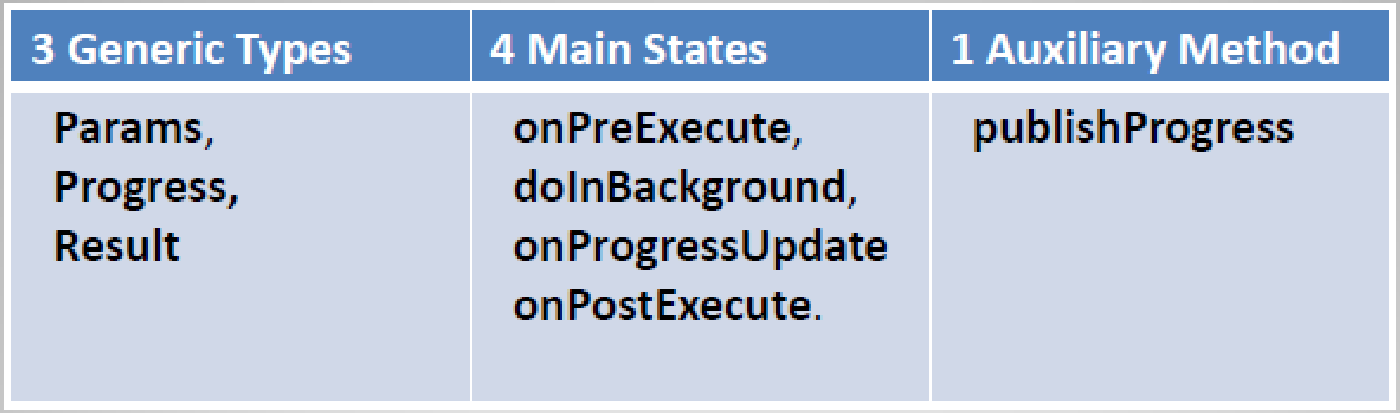
- Parametrizuje se sa *Params, Progress i Result*



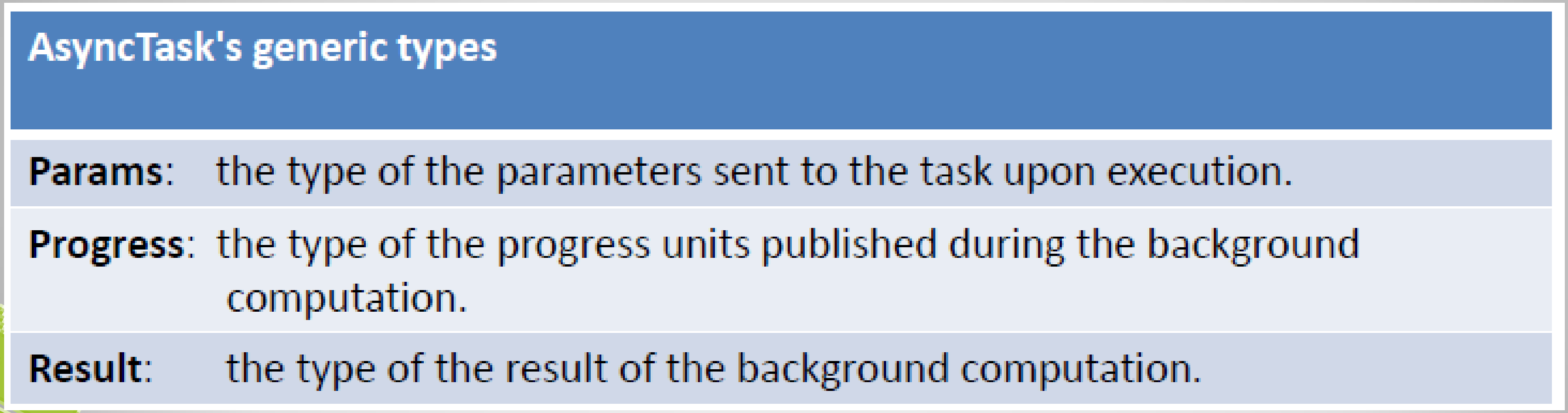
```
private class VerySlowTask extends AsyncTask<String, Long, Void> {

    // Begin - can use UI thread here
    protected void onPreExecute() {

    }

    // this is the SLOW background thread taking care of heavy tasks
    // cannot directly change UI
    protected Void doInBackground(final String... args) {
    ... publishProgress((Long) someLongValue);
    }

    // periodic updates - it is OK to change UI
    @Override
    protected void onProgressUpdate(Long... value) {

    }

    // End - can use UI thread here
    protected void onPostExecute(final Void unused) {

    }
}
```

AsyncTask

- “Čistija” komunikacija sa UI thread-om i widget-ima
- Ne zahteva korišćenje Handler-a i thread-ova



3 Generic Types	4 Main States	1 Auxiliary Method
Params, Progress, Result	onPreExecute, doInBackground, onProgressUpdate onPostExecute.	publishProgress



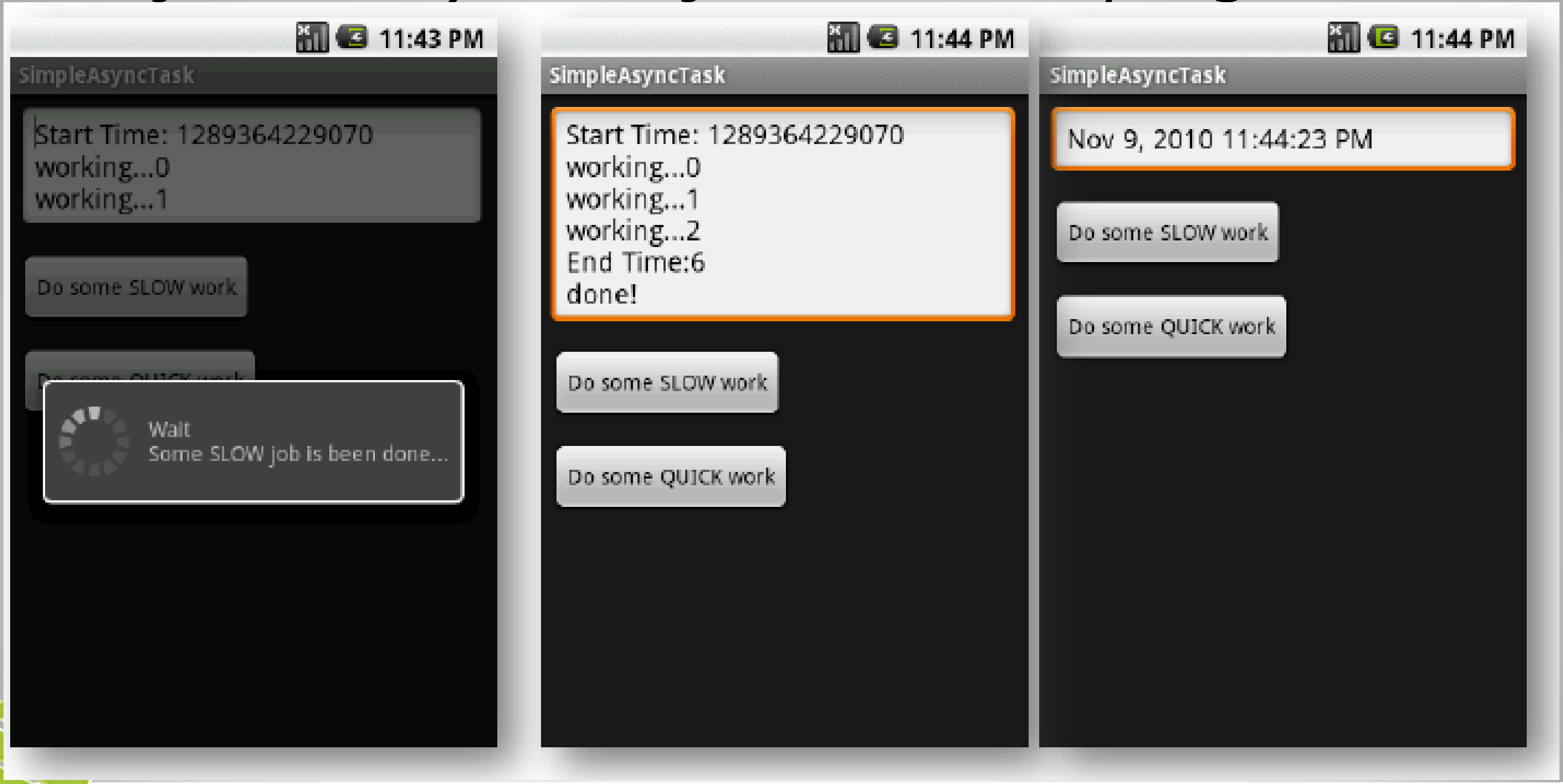
AsyncTask's generic types	
Params:	the type of the parameters sent to the task upon execution.
Progress:	the type of the progress units published during the background computation.
Result:	the type of the result of the background computation.

AsyncTask

- Tip koji se ne koristi može biti označen sa **Void**
- Upotreba callback metoda
  - **onPreExecute()** – izvršava se na UI thread-u i koristi za inicijalno postavljanje UI
  - **doInBackground(Params...) –** pozadinska operacija, u ovoj metodi može da se pozove *publishProgress(Progress..)*
  - **onProgressUpdate(Progress...) –** update-uje UI u skladu sa progresom
  - **onPostExecute(Result) –** prikazuje rezultat na UI widget-ima

AsyncTask - primer

- Glavni task poziva AyncTask koji izračunava nešto i periodično update-uje UI (upisuje linije teksta) i menja cirkularni progress bar



```
11:43 PM
SimpleAsyncTask
Start Time: 1289364229070
working...0
working...1
Wait
Some SLOW job is been done...
```

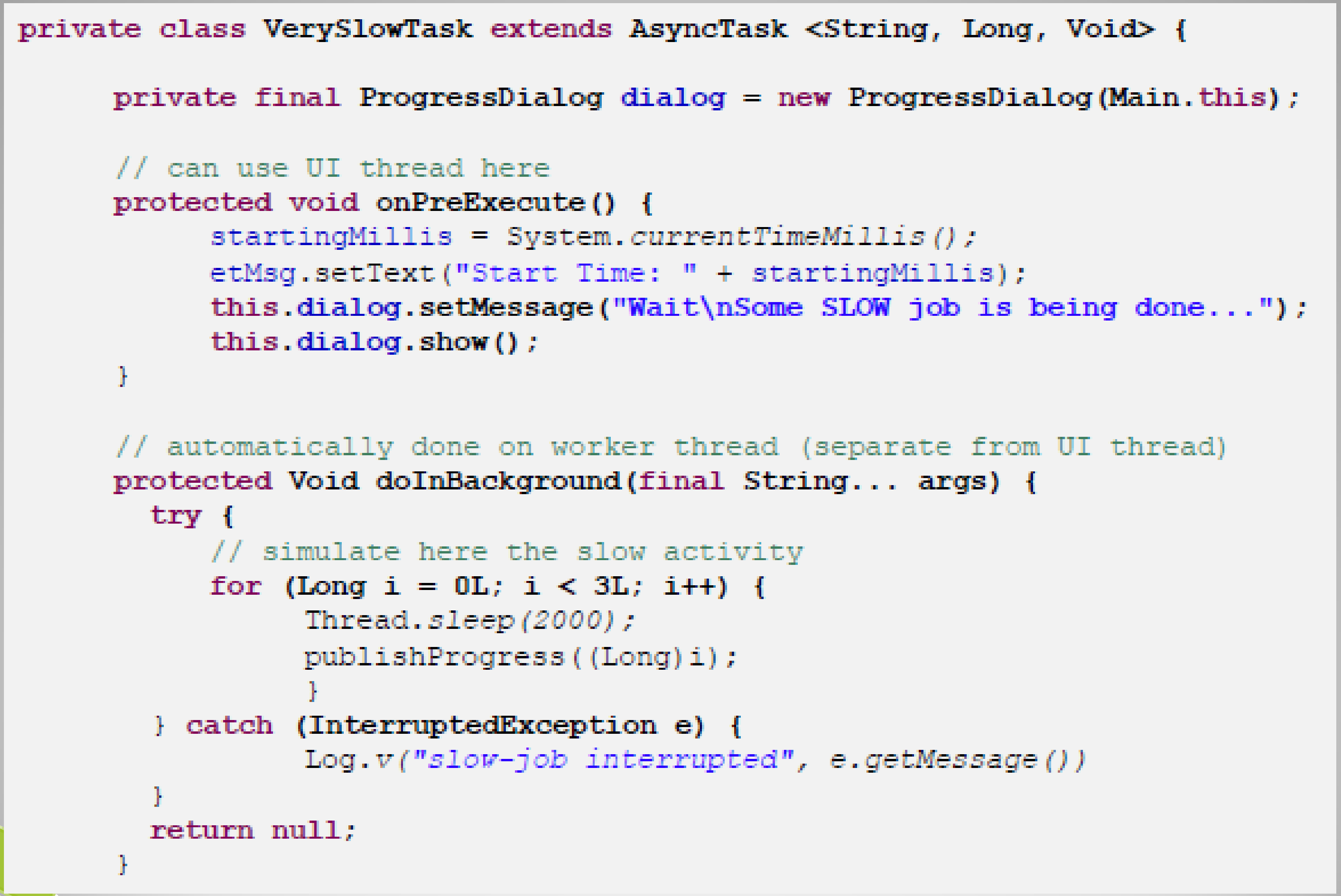
```
11:44 PM
SimpleAsyncTask
Start Time: 1289364229070
working...0
working...1
working...2
End Time:6
done!
```

```
11:44 PM
SimpleAsyncTask
Nov 9, 2010 11:44:23 PM
```

AsyncTask - primer

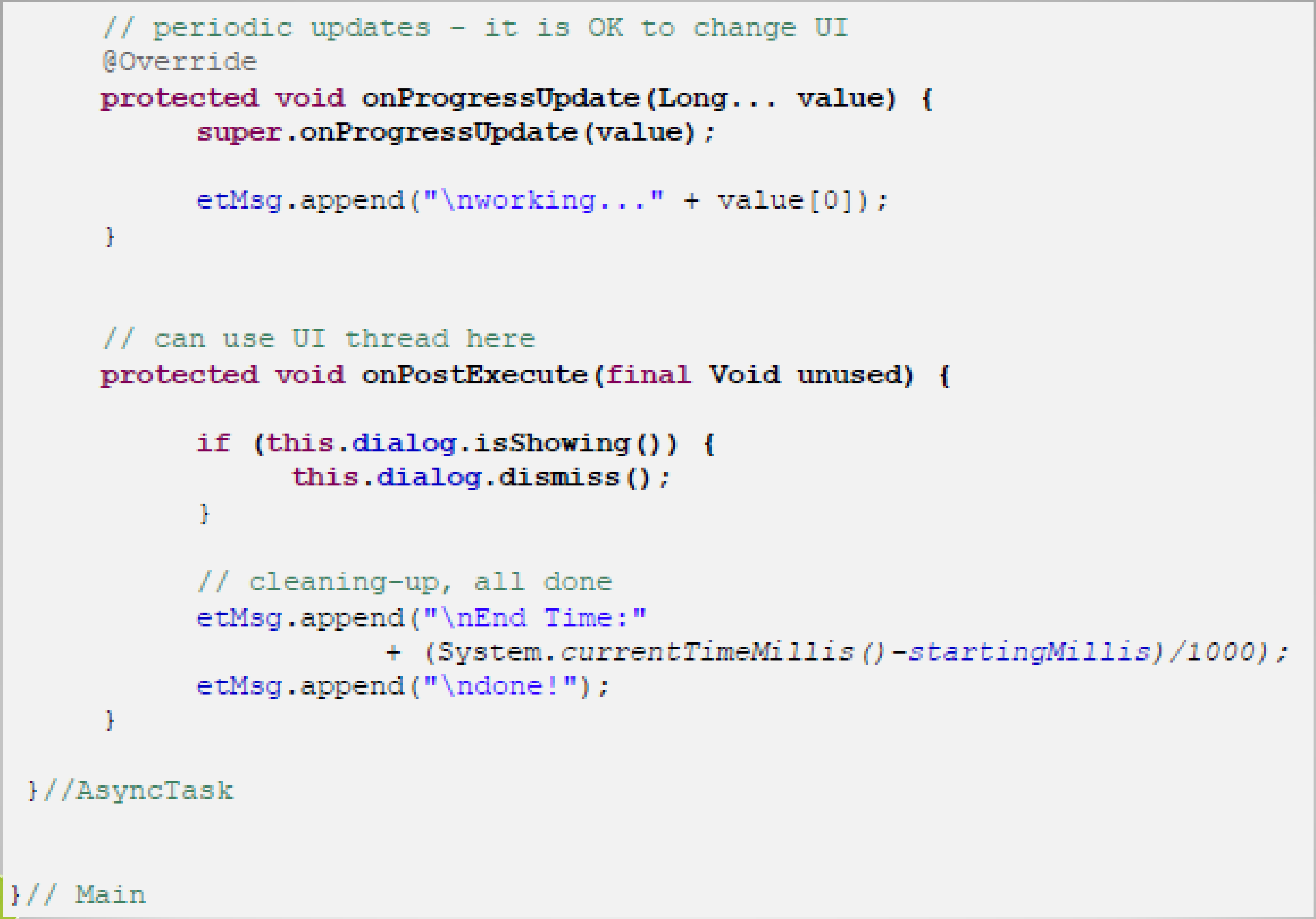
```
public class Main extends Activity {
Button btnSlowWork;
Button btnQuickWork;
EditText etMsg;
Long  startingMillis;

@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      etMsg = (EditText) findViewById(R.id.EditText01);

      btnSlowWork = (Button) findViewById(R.id.Button01);
      // slow work...for example: delete all data from a database or get data from Internet
      this.btnSlowWork.setOnClickListener(new OnClickListener() {
            public void onClick(final View v) {
                  new VerySlowTask().execute();
            }
      });

      btnQuickWork = (Button) findViewById(R.id.Button02);
      // delete all data from database (when delete button is clicked)
      this.btnQuickWork.setOnClickListener(new OnClickListener() {
            public void onClick(final View v) {
                  etMsg.setText((new Date()).toLocaleString());
            }
      });

}// onCreate
```

AsyncTask - primer



```
private class VerySlowTask extends AsyncTask <String, Long, Void> {

     private final ProgressDialog dialog = new ProgressDialog(Main.this);

     // can use UI thread here
     protected void onPreExecute() {
          startingMillis = System.currentTimeMillis();
          etMsg.setText("Start Time: " + startingMillis);
          this.dialog.setMessage("Wait\nSome SLOW job is being done...");
          this.dialog.show();
     }

     // automatically done on worker thread (separate from UI thread)
     protected Void doInBackground(final String... args) {
       try {
          // simulate here the slow activity
          for (Long i = 0L; i < 3L; i++) {
               Thread.sleep(2000);
               publishProgress((Long)i);
               }
       } catch (InterruptedException e) {
               Log.v("slow-job interrupted", e.getMessage())
       }
       return null;
     }
```

AsyncTask - primer



```
     // periodic updates - it is OK to change UI
     @Override
     protected void onProgressUpdate(Long... value) {
          super.onProgressUpdate(value);

          etMsg.append("\nworking..." + value[0]);
     }


     // can use UI thread here
     protected void onPostExecute(final Void unused) {

          if (this.dialog.isShowing()) {
               this.dialog.dismiss();
          }

          // cleaning-up, all done
          etMsg.append("\nEnd Time:"
                    + (System.currentTimeMillis()-startingMillis)/1000);
          etMsg.append("\ndone!");
     }

 }//AsyncTask


}// Main
```