NEW MEDIA &
COMMUNICATION
TECHNOLOGY

Mobile App Development

“Redesigning your application to run multithreaded on a multicore machine

is a little like learning to swim by jumping into the deep end.”

User Interface Threads

Also known as Main Thread

An academic demonstration


```
public class MainActivity extends Activity {
    private TextView output;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        output = (TextView) findViewById(R.id.out);
    }

    public void bigTask(View v) {
        try {
            Thread.sleep(100000); // Wow, much work, such speed
        } catch (InterruptedException e) {}
        output.setText("Big Task Done");
    }

    public void smallTask(View v) {
        output.setText("Small Task Done");
    }

}
```

An academic demonstration


```
03-18 09:30:50.378: I/Choreographer(14043): Skipped 600 frames!  The application may be doing too much work on its main thread.
```

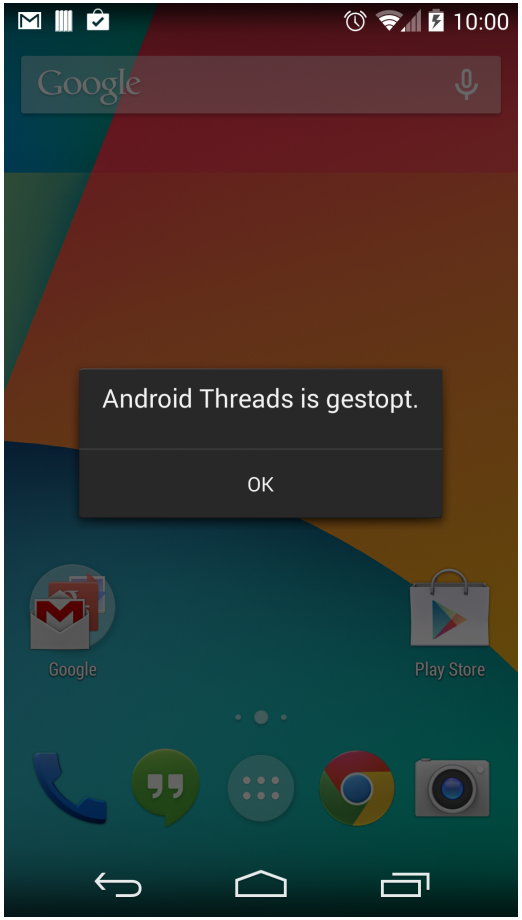
ANR: Application Not Responding

"You should not perform
the work on the UI thread"

Adding Threads, the wrong way

```
public class MainActivity extends Activity {
    private TextView output;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        output = (TextView)  findViewById(R.id.out);
    }

    public void bigTask(View v) {
        Thread t =  new Thread() {
            public void run() {
                try {
                    Thread. sleep(1000);
                } catch (InterruptedException e) {}
                output. setText("Big Tast Done");
            }
        };
        t.start();
    }

    public void smallTask(View v) {
        output.setText("Small Task Done");
    }
}
```

You can’t touch its views




```
android.view.ViewRootImpl$CalledFromWrongThreadException:
    Only the original thread that created a view hierarchy can touch its views.
at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:6094)
at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:824)
at android.view.View.requestLayout(View.java:16431)
at android.view.View.requestLayout(View.java:16431)
at android.view.View.requestLayout(View.java:16431)
at android.view.View.requestLayout(View.java:16431)
at android.widget.RelativeLayout.requestLayout(RelativeLayout.java:352)
at android.view.View.requestLayout(View.java:16431)
at android.widget.TextView.checkForRelayout(TextView.java:6600)
at android.widget.TextView.setText(TextView.java:3813)
at android.widget.TextView.setText(TextView.java:3671)
at android.widget.TextView.setText(TextView.java:3646)
at be.howest.demo.at.MainActivity$1.run(MainActivity.java:28)
```

The UI-Thread

- All apps in the same process share 1 thread:
  The main Thread (= UI-Thread)
- All App Components Are in the same process.
- The Lifecycle methods are executed in the Main-Thread
- Android UI is not Thread Safe

Long running tasks & the Main-thread

- Block User interface
- Accessing UI from other threads can corrupt UI.
  - Activity.runOnUIThread
  - View.post

Executing code on the Main-Thread

```
public void bigTask(View v) {

    Thread t = new Thread() {
        public void run() {

            try {
                Thread.sleep(10000); // Wow, much work, such speed
            } catch (InterruptedException e) {}

            output.post(new Runnable() {
                public void run() { output.setText("Big Tast Done"); };
            });
        }
    };

    t.start();

}
```

Threads & configuration changes

retaining fragments

Retain instance: the task

```
final Thread mThread = new Thread() {
    public void run() {
        int max = 10000;
        while (true) {
            synchronized (this) {
                while (!mReady || mPosition >= max) {
                    if (mQuiting) { return; }
                    try { wait(); } catch (InterruptedException e) {}
                }

                mPosition++;
                max = mProgressBar. getMax();
                mProgressBar. setProgress(mPosition);
            }

            synchronized (this) {
                try { wait(50); } catch (InterruptedException e) { }
            }

        }

    }

};
```

Retain instance: the fragment

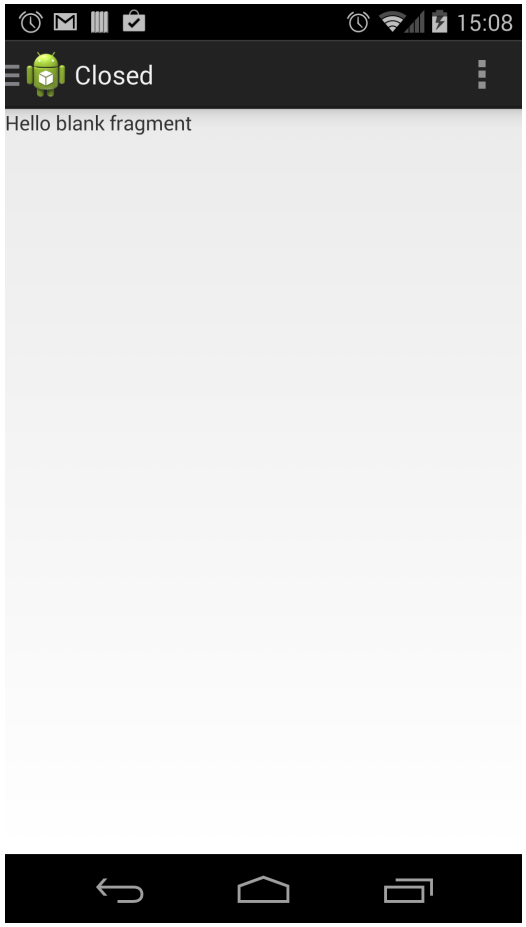
```
public class RetainedFragment extends Fragment {
    ProgressBar mProgressBar;
    int mPosition;
    boolean mReady = false;
    boolean mQuiting = false;
    final Thread mThread = new Thread() {};

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
        mThread.start();
    }

    public void onDestroy() {
        synchronized (mThread) {
            mReady = false;
            mQuiting = true;
            mThread.notify();
        }
        super.onDestroy();
    }

}
```

Retain instance: the fragment

```
public class RetainedFragment extends Fragment {

    public void onActivityCreated (Bundle savedInstanceState) {
        super.onActivityCreated (savedInstanceState);
        mProgressBar = (ProgressBar) getTargetFragment ().getView ().findViewById (R.id.prgrs);
        synchronized (mThread) {
            mReady = true;
            mThread. notify ();
        }
    }

    public void onDetach () {
        synchronized (mThread) {
            mProgressBar = null;
            mReady = false;
            mThread. notify ();
        }
        super.onDetach ();
    }

    public void restart () {
        synchronized (mThread) {
            mPosition = 0;
            mThread. notify ();
        }
    }
}
```

DrawerLayout & DrawerToggle

Drawers



DrawerLayout

```
<android.support.v4.widget.DrawerLayout
    xmlns:android=http://schemas.android.com/apk/res/android
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- The main content view -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- The navigation drawer -->
    <fragment android:name="com.example.drawers.DrawerFragment"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:id="@+id/left_drawer"
        android:background="#111"/>

</android.support.v4.widget.DrawerLayout>
```

Activity host

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mDrawerLayout = (DrawerLayout)  findViewById(R.id.drawer_layout);
    mDrawer =  findViewById(R.id.left_drawer);

    mDrawerToggle =  new ActionBarDrawerToggle(this, mDrawerLayout,
            R.drawable.ic_drawer, R.string.drawer_open,
            R.string.drawer_close);


    mDrawerLayout.setDrawerListener(mDrawerToggle);


    getActionBar().setDisplayHomeAsUpEnabled(true);
    getActionBar().setHomeButtonEnabled(true);

    FragmentManager fragmentManager =  getFragmentManager();
    fragmentManager.beginTransaction()
                    .replace(R.id.content_frame, ContentFragment.newInstance())
                    .commit();


}
```

Activity host

```
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    mDrawerToggle.syncState();
}


public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    mDrawerToggle.onConfigurationChanged(newConfig);
}

public boolean onOptionsItemSelected(MenuItem item) {
    if (mDrawerToggle.onOptionsItemSelected(item)) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
```

Activity host

```
mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
            R.drawable.ic_drawer, R.string.drawer_open,
            R.string.drawer_close) {

        public void onDrawerClosed(View view) {
            super.onDrawerClosed(view);
            getActionBar().setTitle("Closed");
            invalidateOptionsMenu();
        }

        public void onDrawerOpened(View drawerView) {
            super.onDrawerOpened(drawerView);
            getActionBar().setTitle("open");
            invalidateOptionsMenu();
        }
    };
```

Activity host

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        //boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerLayout);
        // Modify Menu when drawer is open of closed
        return super.onPrepareOptionsMenu(menu);
    }
```

Data

Parsing and creating JSON

JSON

```
[

   {
     "id": 912345678901,
     "text": "How do I read JSON on Android?" ,
     "geo": null,
     "user": {
       "name": "android_newb",
       "followers_count": 41
      },

   {

     "id": 912345678902,
     "text": "@android_newb just use android.util.JsonReader!" ,
     "geo": [50.454722, -104.606667],
     "user": {
       "name": "jesse",
       "followers_count": 2
     }

   }

 ]
```

JSON: reading

```
public List readJsonStream(InputStream in) throws IOException {
  JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
  try {
    return readMessagesArray(reader);
   finally {
    reader.close();
  }
}


public List readMessagesArray(JsonReader reader) throws IOException {
  List messages = new ArrayList();
  reader.beginArray();
  while (reader.hasNext()) {
    messages.add(readMessage(reader));
  }
  reader.endArray();
  return messages;
}
```

JSON: reading

```
public Message readMessage(JsonReader reader) throws IOException {
    long id = -1;
    String text = null;
    User user = null;
    List geo = null;

    reader.beginObject();
    while (reader.hasNext()) {
      String name = reader.nextName();
      if (name.equals("id")) {
        id = reader.nextLong();
      } else if (name.equals("text")) {
        text = reader.nextString();
      } else if (name.equals("geo") && reader.peek() != JsonToken.NULL) {
        geo = readDoublesArray(reader);
      } else if (name.equals("user")) {
        user = readUser(reader);
      } else {
        reader.skipValue();
      }
    }
    reader.endObject();
    return new Message(id, text, user, geo);
  }
```

JSON: reading

```
public List readDoublesArray(JsonReader reader) throws IOException {
  List doubles = new ArrayList();
  reader.beginArray();
  while (reader.hasNext()) {
    doubles.add(reader.nextDouble());
  }
  reader.endArray();
  return doubles;
}

public User readUser(JsonReader reader) throws IOException {
  String username = null;
  int followersCount = -1;
  reader.beginObject();
  while (reader.hasNext()) {
    String name = reader.nextName();
    if (name.equals("name")) {
      username = reader.nextString();
    } else if (name.equals("followers_count")) {
      followersCount = reader.nextInt();
    } else {
      reader.skipValue();
    }
  }
  reader.endObject();
  return new User(username, followersCount);
}
```

JSON: writing

```
public void writeJsonStream(OutputStream out, List messages)  throws IOException {
   JsonWriter writer =  new JsonWriter(new OutputStreamWriter(out, "UTF-8"));
   writer.setIndent("  ");
   writeMessagesArray(writer, messages);
   writer.close();
 }

 public void writeMessagesArray(JsonWriter writer, List messages)  throws IOException {
   writer.beginArray();
   for (Message message : messages) {
     writeMessage(writer, message);
   }
   writer.endArray();
 }
```

JSON: writing

```
public void writeMessage(JsonWriter writer, Message message)  throws IOException {
  writer.beginObject();
  writer.name("id").value(message.getId());
  writer.name("text").value(message.getText());
  if (message.getGeo() != null) {
    writer.name("geo");
    writeDoublesArray(writer, message.getGeo());
  } else {
    writer.name("geo").nullValue();
  }
  writer.name("user");
  writeUser(writer, message.getUser());
  writer.endObject();

}

public void writeUser(JsonWriter writer, User user)  throws IOException {
  writer.beginObject();
  writer.name("name").value(user.getName());
  writer.name("followers_count").value(user.getFollowersCount());
  writer.endObject();
}


public void writeDoublesArray(JsonWriter writer, List doubles)  throws IOException {

  writer.beginArray();

  for (Double value : doubles) {
    writer.value(value);
```

JSON: writing

```
public void writeDoublesArray(JsonWriter writer, List doubles)  throws IOException {
  writer.beginArray();
  for (Double value : doubles) {
    writer.value(value);
  }
  writer.endArray();
}
```

GSON for JSON

```
public class Message {
    @SerializedName ("id")
    public int id;

    @SerializedName ("text")
    public String text;

    @SerializedName ("geo")
    public double[] geo = null;

    @SerializedName ("user")
    public User user = null;
}


public class User {
    @SerializedName ("name")
    public String name;

    @SerializedName ("followers_count" )
    public int followersCount;
}
```

GSON for JSON

```
Collection<Message> messages =  new ArrayList<Message>();

Message message1 = new Message();


message1.id = 123454545;
message1.text = "some text";

message1.geo = new double[2];
message1.geo[0] = 50.1;
message1.geo[1] = 50.2;

message1.user = new User();


messages.add(message1);



Gson gson = new Gson();
String json  = gson.toJson(messages);


Log.v("GSON DEMO", json);
```

RecycleView

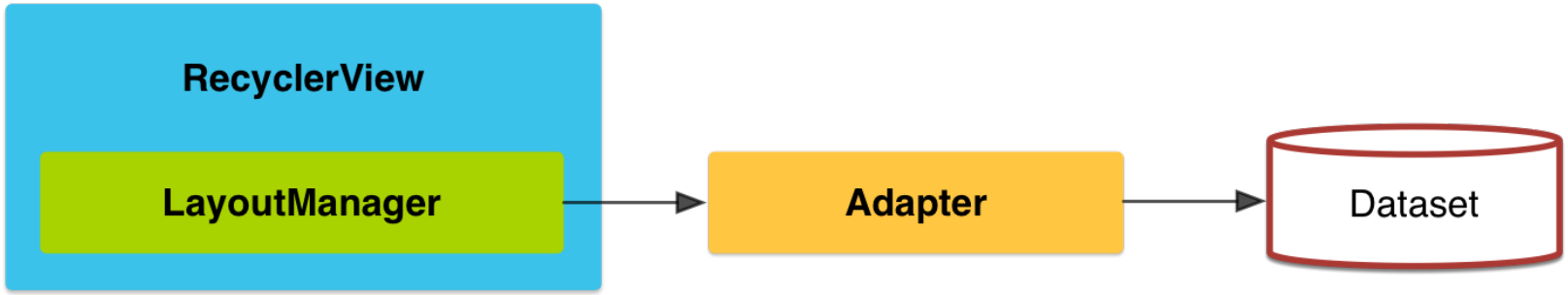
Less is more, is less, is more, ...

Why RecyclerView?

- listview -> layout, animation, recycle, events, binding, …. < Do one thing and do it well.

- RecyclerView -> recycle …. and that is all
  (and also scroll events)

- but
  - uses layoutmanager
  - uses adapter
  - uses animators
  - has first hand knowledge of viewholder


RecyclerView
LayoutManager
Adapter
Dataset

Setting the dependencies

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "be.verborgh.recycle"
        minSdkVersion 21
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile 'com.android.support:recyclerview-v7:+'
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Creating the layout

```
<RelativeLayout 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" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <RecyclerView
        android:id="@+id/rv"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

    </RecyclerView>

</RelativeLayout>
```

Creating an itemview

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/tvVariety" />

</LinearLayout>
```

Creating a ViewHolder

```
public class PotatoViewHolder extends RecyclerView.ViewHolder{
    public TextView tvVariety;

    public PotatoViewHolder(View itemView) {
        super(itemView);
        tvVariety = (TextView) itemView.findViewById(R.id.tvVariety);
    }

    public void bindPotato(Potato potato) {
        tvVariety.setText(potato.getVariety());
    }
}
```

Creating an Adapter

```
public class PotatoesAdapter extends RecyclerView.Adapter<PotatoViewHolder> {

    private final List<Potato> potatoes;

    public PotatoesAdapter(List<Potato> potatoes) {
        this.potatoes = potatoes;
    }

    @Override
    public PotatoViewHolder onCreateViewHolder(ViewGroup parent, int i) {
        View potatoView = LayoutInflater.from(parent.getContext()).inflate(R.layout.potato_item, parent, false);
        return new PotatoViewHolder(potatoView);
    }

    @Override
    public void onBindViewHolder(PotatoViewHolder potatoViewHolder, int i) {
         potatoViewHolder.bindPotato(potatoes.get(i));
    }

    @Override
    public int getItemCount() {
        return potatoes.size();
    }
}
```

Putting it together

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

    rv = (RecyclerView)findViewById(R.id.rv);
    rv.setLayoutManager(new LinearLayoutManager(this));
    rv.setAdapter(new PotatoesAdapter(Potatoes.get()));
}
```

Adding ItemSelection

```
public class PotatoViewHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {

    public TextView tvVariety;

    public PotatoViewHolder(View itemView) {
        super(itemView);
        itemView.setOnClickListener(this);
        tvVariety = (TextView) itemView.findViewById(R.id.tvVariety);
    }

    public void bindPotato(Potato potato) {
        tvVariety.setText(potato.getVariety());
    }

    @Override
    public void onClick(View v) {
        //CLICKY
    }
}
```

https://www.bignerdranch.com/blog/recyclerview-part-1-fundamentals-for-listview-experts/
https://github.com/saulmm/RecyclerView-demo
https://github.com/commonsguy/cw-omnibus/tree/master/RecyclerView