# OPTIMIZING ANDROID UI

## PRO TIPS FOR CREATING SMOOTH AND RESPONSIVE APPS

# @CYRILMOTTIER

# GET TO KNOW JAVA

# DON'T USE BOXED TYPES UNNECESSARILY

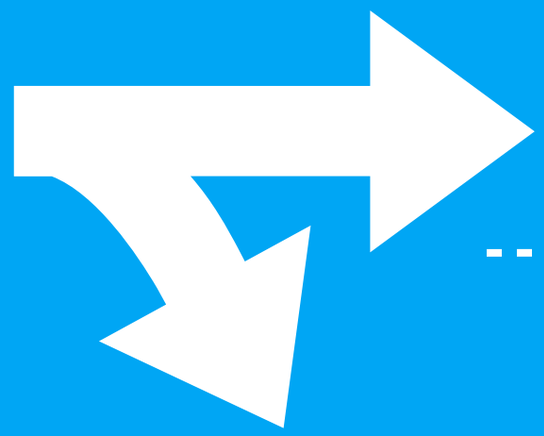
```
HashMap<Integer, String> hashMap = new HashMap<Integer, String>();
hashMap.put(665, "Android");
hashMap.put(666, "rocks");

hashMap.get(666);
```

```
HashMap<Integer, String> hashMap = new HashMap<Integer, String>();
hashMap.put(665, "Android");
hashMap.p

hashMap.get(666);
```

Integer is not int

BECOMES

666



666

DREAMS

new Integer(666)

REALITY

# QUESTION

Would the result have been the same if we had been trying to add "Android" with 42 as key?

# QUESTION

**Would the result have been the same if we had been trying to add "Android" with 42 as key?**

# ANSWER

**Yes**

**42 is the answer to the Java universe**

# QUESTION

**Would the result have been the same if we had been trying to add "Android" with 42 as key?**

# ANSWER

**I AM JOKING**

**42 is the an… …es … universe**

# QUESTION

Would the result have been the same if we had been trying to add "Android" with 42 as key?

# ANSWER

No

Integer has an internal cache for [-128, 127]

# SparseArray:

SparseArrays map integers to Objects. Unlike a normal array of Objects, there can be gaps in the indices. It is intended to be more efficient than using a HashMap to map Integers to Objects.

```
SparseArray<String> sparseArray = new SparseArray<String>();
sparseArray.put(665, "Android");
sparseArray.put(666, "rocks");

sparseArray.get(666);
```

```
SparseArray<String> sparseArray = new SparseArray<String>();
sparseArray.put(665, "Android");
sparseArray.put(666, "rocks");

sparseArray.get(666);
```

ints as keys,
Strings as values

```
SparseArray<String> sparseArray = new SparseArray<String>();
sparseArray.put(665, "Android");
sparseArray.put

sparseArray.get(666);
```

no autoboxing, at all

ints as keys,
Strings as values

# REUSE AS MUCH AS POSSIBLE

# NEVER DO THIS !!!

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    final View itemView = mInflater.inflate(
            android.R.layout.two_line_list_item, // resource
            parent,                              // parent
            false);                              // attach

    ((TextView) itemView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) itemView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return itemView;
}
```

# NEVER DO THIS !!!

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    final View itemView = mInflater.inflate(
            android.R.layout.two_line_list_item, // resource
            parent,                              // parent
            false);                              // attach

    ((TextView) itemView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) itemView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return itemView;
}
```

inflate a new View at every getView call

# convertView:

The old view to reuse, if possible. Note: You should check that this view is non-null and of an appropriate type before using. If it is not possible to convert this view to display the correct data, this method can create a new view.

# (RE)USE THE CONVERTVIEW

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(
                android.R.layout.two_line_list_item, // resource
                parent,                              // parent
                false);                              // attach
    }

    ((TextView) convertView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) convertView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return convertView;
}
```

# (RE)USE THE CONVERTVIEW

```
@Override
public View getVi                                    View, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(
                android.R.layout.two_line_list_item, // resource
                parent,                              // parent
                false);                              // attach
    }

    ((TextView) convertView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) convertView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return convertView;
}
```

check for a convertView

# (RE)USE THE CONVERTVIEW

```
@Override
public View getVi                                    View, ViewGroup parent) {

    if (convertView == null) {
        convertView = mInflater.inflate(
                android.R.layout.two_line_list_item,
                parent,                                          // parent
                false);                                          // attach
    }

    ((TextView) convertView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) convertView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return convertView;
}
```

check for a convertView

create a view only when necessary

# PREFER STATIC FACTORY METHODS TO CONSTRUCTORS

```
private static final int MSG_ANIMATION_FRAME = 0xcafe;

public void sendMessage(Handler handler, Object userInfo) {
    final Message message = new Message();
    message.what = MSG_ANIMATION_FRAME;
    message.obj = userInfo;

    handler.sendMessage(message);
}
```

```
private static final int MSG_ANIMATION_FRAME = 0xcafe;

public void sendMessage(Handler handler, Object userInfo) {
    final Message message = new Message();
    message.what = MSG_ANIMATION_FRAME;
    message.obj = userInfo;

    handler.sendMessage(message);
}
```

creation of a new Message instance

```
private static final int MSG_ANIMATION_FRAME = 0xcafe;

public void sendMessage(Handler handler, Object userInfo) {
    final Message message = Message.obtain();
    message.what = MSG_ANIMATION_FRAME;
    message.obj = userInfo;

    handler.sendMessage(message);
}
```

```
private static final int MSG_ANIMATION_FRAME = 0xcafe;

public void sendMessage(Handler handler, Object userInfo) {
    final Message message = Message.obtain();
    message.what = MSG_ANIMATION_FRAME;
    message.obj = userInfo;

    handler.sendMessage(message);
}
```

try to reuse Message instances

Message.obtain(): Gets an object from a pool or create a new one

# Want this in your app?

## Go to
**github.com/android/platform_frameworks_base**

## Find
**android.util**

## Copy
**PoolableManager, Pool, Poolable, Pools, FinitePool, etc.**

## Paste
**in your project**

## Enjoy
**the easy-to-use object pooling mechanism**

# PREFER STATIC VARIABLES TO TEMPORARY VARIABLES

```java
@Override
public boolean onInterceptTouchEvent(MotionEvent event) {

    final int x = (int) event.getX();
    final int y = (int) event.getY();
    final Rect frame = new Rect();

    // Are we touching mHost ?
    mHost.getHitRect(frame);
    if (!frame.contains(x, y)) {
        return false;
    }

    return true;
}
```

```
@Override
public boolean onInterceptTouchEvent(MotionEvent event) {

    final int x = (int) event.getX();
    final int y = (int) event.getY();
    final Rect frame = new Rect();

    // Are we touching
    mHost.getHitRect(f
    if (!frame.contains(x, y)) {
        return false;
    }

    return true;
}
```

creation of a new Rect instance

```
private final Rect mFrameRect = new Rect();

@Override
public boolean onInterceptTouchEvent(MotionEvent event) {

    final int x = (int) event.getX();
    final int y = (int) event.getY();
    final Rect frame = mFrameRect;

    // Are we touching mHost ?
    mHost.getHitRect(frame);
    if (!frame.contains(x, y)) {
        return false;
    }

    return true;
}
```

```
private final Rect mFrameRect = new Rect();

@Override
public boolean onInterceptTouchEvent(MotionEvent event) {

    final int x = (int) event.getX();
    final int y = (int) event.getY();
    final Rect frame = mFrameRect;

    // Are we touching mHost ?
    mHost.getHitRect(frame);
    if (!frame.contains(x, y)) {
        return false;
    }

    return true;
}
```

create a single Rect

```
private final Rect mFrameRect = new Rect();

@Override
public boolean onInterceptTouchEvent(MotionEvent event) {

    final int x = (int) event.getX();
    final int y = (int) event.getY();
    final Rect frame = mFrameRect;

    // Are we touching mHost ?
    mHost.getHitRect(frame);
    if (!frame.contains(x, y)) {
        return false;
    }

    return true;
}
```

create a single Rect

always use the same instance of Rect

REMEMBER STRINGS ARE ELLIGIBLE TO GARBAGE COLLECTION

```
public class CharArrayBufferAdapter extends CursorAdapter {

    private interface DataQuery {
        int TITLE = 0;
        int SUBTITLE = 1;
    }

    public CharArrayBufferAdapter(Context context, Cursor c) {
        super(context, c);
    }

    @Override public void bindView(View view, Context context, Cursor cursor) {
        final TwoLineListItem item = (TwoLineListItem) view;
        item.getText1().setText(cursor.getString(DataQuery.TITLE));
        item.getText2().setText(cursor.getString(DataQuery.SUBTITLE));
    }

    @Override public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(
                android.R.layout.two_line_list_item, parent, false);
    }
}
```

```
public class CharArrayBufferAdapter extends CursorAdapter {

    private interface DataQuery {
        int TITLE = 0;
        int SUBTITLE = 1;
    }

    public CharArrayBufferAdapter(Context context, Cursor c) {
        super(context, c);
    }

    @Override public void bindView(View view, Context context, Cursor cursor) {
        final TwoLineListItem item = (TwoLineListItem) view;
        item.getText1().setText(cursor.getString(DataQuery.TITLE));
        item.getText2().setText(cursor.getString(DataQuery.SUBTITLE));
    }

    @Override public View newView(Contex                    r, ViewGroup parent) {
        return LayoutInflater.from(contex                    (
                android.R.layout.two_line_list_item, parent, false);
    }
}
```

getString means new String

```
private static class ViewHolder {
    CharArrayBuffer titleBuffer = new CharArrayBuffer(128);
    CharArrayBuffer subtitleBuffer = new CharArrayBuffer(128);
}

@Override public void bindView(View view, Context context, Cursor cursor) {
    final TwoLineListItem item = (TwoLineListItem) view;
    final ViewHolder holder = (ViewHolder) view.getTag();

    final CharArrayBuffer titleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.TITLE, titleBuffer);
    item.getText1().setText(titleBuffer.data, 0, titleBuffer.sizeCopied);

    final CharArrayBuffer subtitleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.SUBTITLE, subtitleBuffer);
    item.getText2().setText(subtitleBuffer.data, 0, subtitleBuffer.sizeCopied);
}

@Override public View newView(Context context, Cursor cursor, ViewGroup parent) {
    final View v = LayoutInflater.from(context).inflate(
            android.R.layout.two_line_list_item, parent, false);
    v.setTag(new ViewHolder());
    return v;
}
```

```
private static class ViewHolder {
    CharArrayBuffer titleBuffer = new CharArrayBuffer(128);
    CharArrayBuffer subtitleBuffer = new CharArrayBuffer(128);
}

@Override public void bindView(View view, Context context, Cursor cursor) {
    final TwoLineListItem item = (TwoLineListItem) view;
    final ViewHolder holder = (ViewHolder) view.getTag();

    final CharArrayBuffer titleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.TITLE, titleBuffer);
    item.getText1().setText(titleBuffer.data, 0, titleBuffer.sizeCopied);

    final CharArrayBuffer subtitleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.SUBTITLE, subtitleBuffer);
    item.getText2().setText(subtitleBuffer.data, 0, subtitleBuffer.sizeCopied);
}

@Override pub                                    ext context, Cursor cursor, ViewGroup parent) {
    final Vie                                    from(context).inflate(
            a                                    ine_list_item, parent, false);
    v.setTag(new ViewHolder());
    return v;
}
```

attach buffers to the itemview

```
private static class ViewHolder {
    CharArrayBuffer titleBuffer = new CharArrayBuffer(128);
    CharArrayBuffer subtitleBuffer = new CharArrayBuffer(128);
}

@Override public void bindView(View view, Context context, Cursor cursor) {
    final TwoLineListItem item = (TwoLineListItem) view;
    final ViewHolder holder = (ViewHolder) view.getTag();

    final CharArrayBuffer titleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.TITLE, titleBuffer);
    item.getText1().setText(titleBuffer.data, 0, titleBuffer.sizeCopied);

    final CharArrayBuffer subtitleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.SUBTITLE, subtitleBuffer);
    item.getText2().setText(subtitleBuffer.data, 0, subtitleBuffer.sizeCopied);
}

@Override pub                                    ext context, Cursor cursor, ViewGroup parent) {
    final Vie                                    from(context).inflate(
            a                                    ine_list_item, parent, false);
    v.setTag(new ViewHolder());
    return v;
}
```

get the buffers

attach buffers to the itemview

```
private static class ViewHolder {
    CharArrayBuffer titleBuffer = new CharArrayBuffer(128);
    CharArrayBuffer subtitleBuffer = new CharArrayBuffer(128);
}

@Override public void bindView(View view, Context context, Cursor cursor) {
    final                    = (TwoLineListItem) view;
    final                    (ViewHolder) view.getTag();

    final                    titleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.TITLE, titleBuffer);
    item.getText1().setText(titleBuffer.data, 0, titleBuffer.sizeCopied);

    final CharArrayBuffer subtitleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.SUBTITLE, subtitleBuffer);
    item.getText2().setText(subtitleBuffer.data, 0, subtitleBuffer.sizeCopied);
}

@Override pub                ext context, Cursor cursor, ViewGroup parent) {
    final Vie                from(context).inflate(
            a                ine_list_item, parent, false);
    v.setTag(new ViewHolder());
    return v;
}
```

copy column content to buffer

get the buffers

attach buffers to the itemview

```
private static class ViewHolder {
    CharArrayBuffer titleBuffer = new CharArrayBuffer(128);
    CharArrayBuffer subtitleBuffer = new CharArrayBuffer(128);
}

@Override public void bindView(View view, Context context, Cursor cursor) {
    final                    = (TwoLineListItem) view;
    final                    (ViewHolder) view.getTag();

    final                    titleBuffer = holder.titleBuffer;
    cursor.copyStringToBuffer(DataQuery.TITLE, titleBuffer);
    item.getText1().setText(titleBuffer.data, 0, titleBuffer.sizeCopied);

    final CharArrayBuffer subtitleBu                    er;
    cursor.copyStringToBuffer(DataQue                   uffer);
    item.getText2().setText(subtitleB                   eBuffer.sizeCopied);
}

@Override pub                       ext context, Cursor cursor, ViewGroup parent) {
    final Vie                       from(context).inflate(
            a                       ine_list_item, parent, false);
    v.setTag(new ViewHolder());
    return v;
}
```

copy column content to buffer

get the buffers

set the buffer to the TextView

attach buffers to the itemview

# FLATTEN VIEW HIERARCHIES

# PERFECTLY MASTER THE ANDROID UI TOOLKIT

1 GridLayout

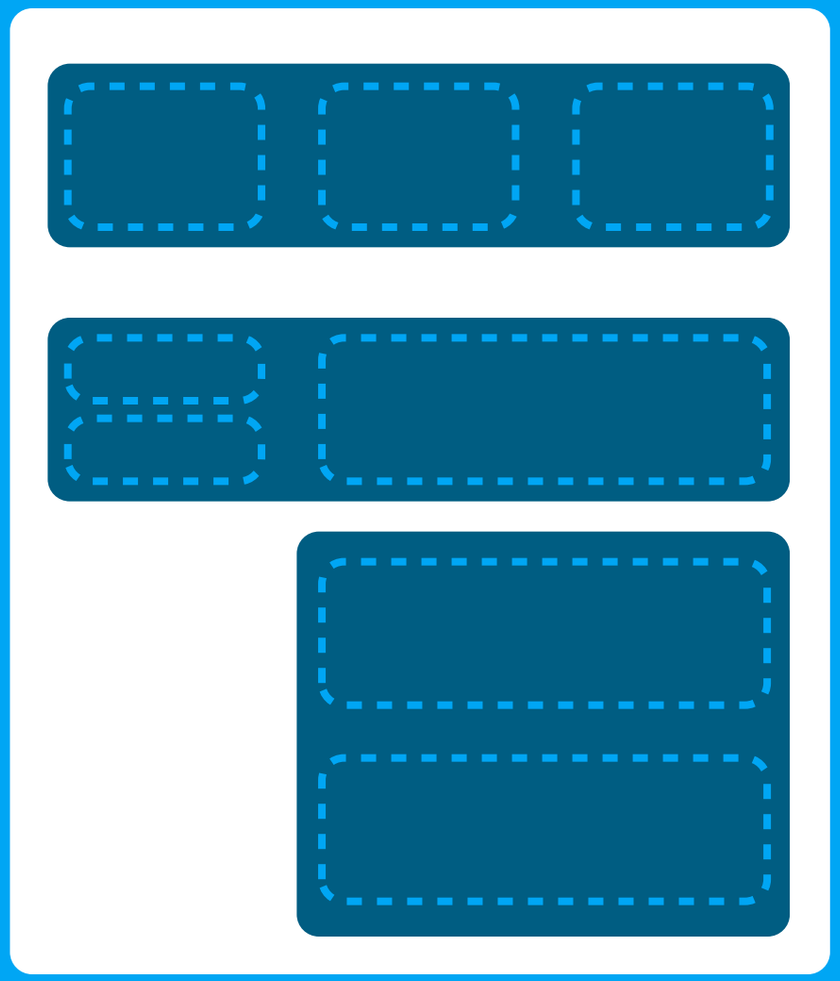
2 RelativeLayout

3 <merge />

# 1 GridLayout

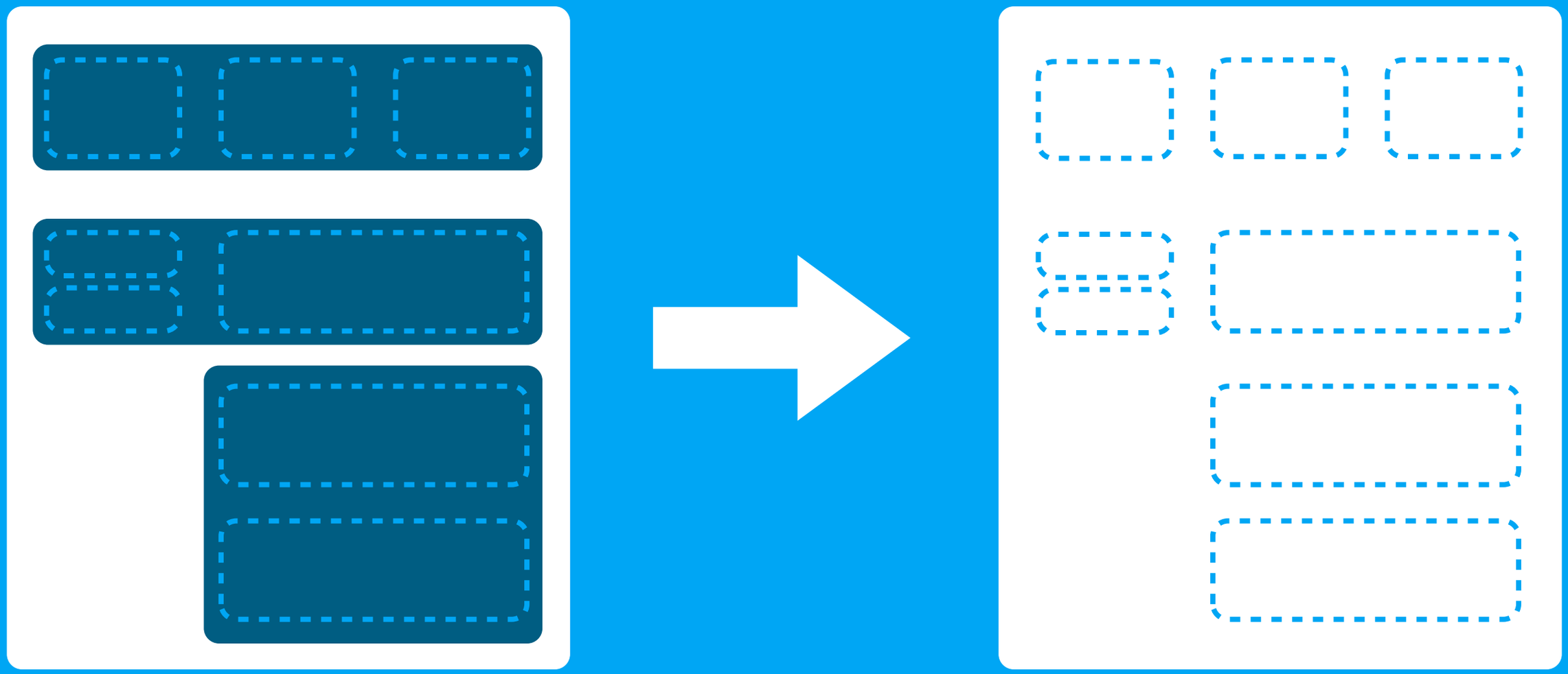
A Layout that places its children in a rectangular grid

# 1 GridLayout



12 VIEWS

# 1 GridLayout



12 VIEWS

9 VIEWS

A Layout where the positions of the children can be described in relation to each other or to the parent.

# RelativeLayout 2

A Layout where the positions of the children can be described in relation to each other or to the parent.

**RelativeLayout 2**

**Flatten hierarchy**

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/icon" />

<TextView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/title" />
```

# 3 <merge />

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/ic_launcher" />
```

**INVALID XML DOCUMENT**

```
<TextView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/title" />
```

## 3 <merge />

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

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/title" />

</FrameLayout>
```

# 3 <merge />

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  an                                  l_parent"
  android:layout_height="fill_parent" >

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/title" />

</FrameLayout>
```

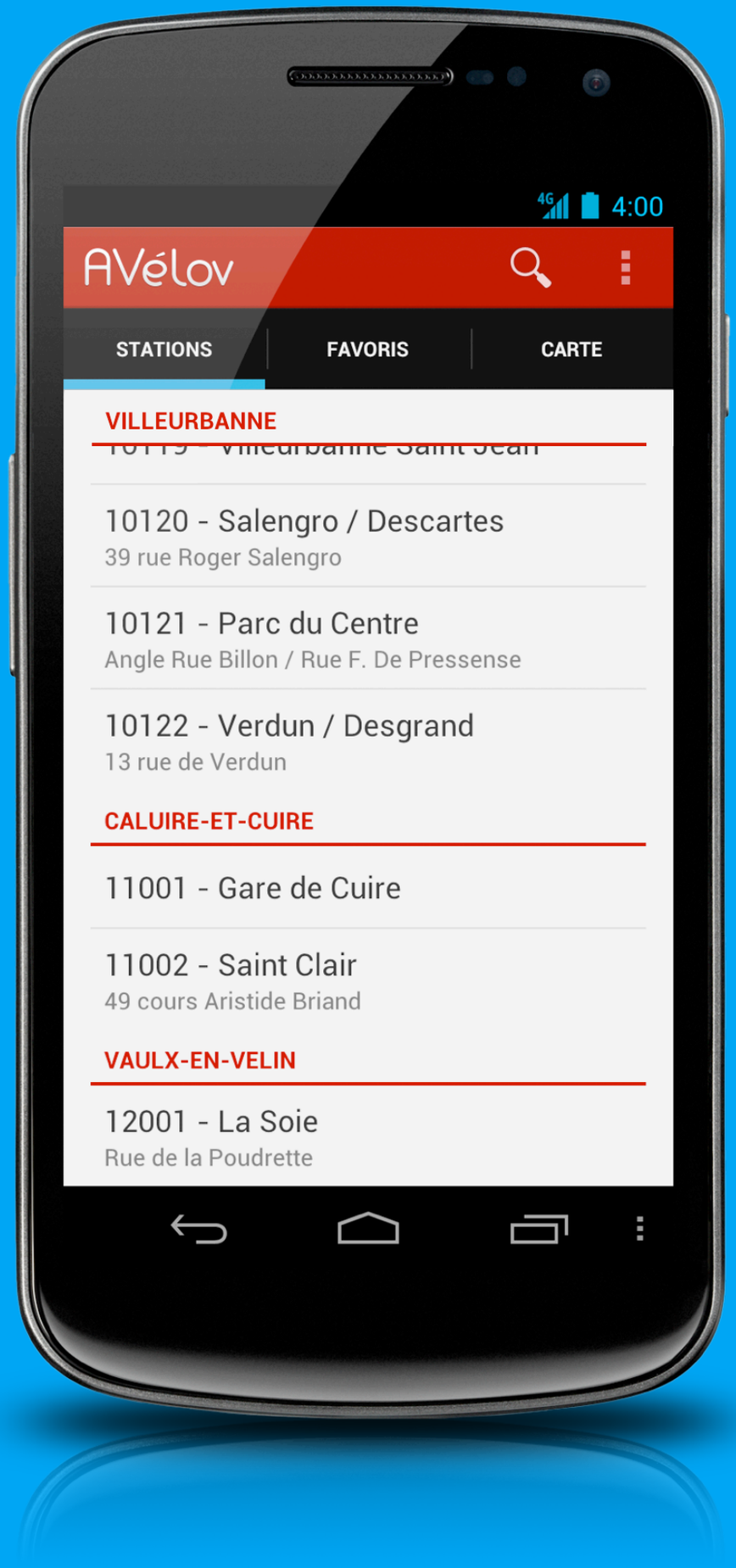
useless ViewGroup

# 3 <merge />

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

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/icon" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/title" />

</merge>
```

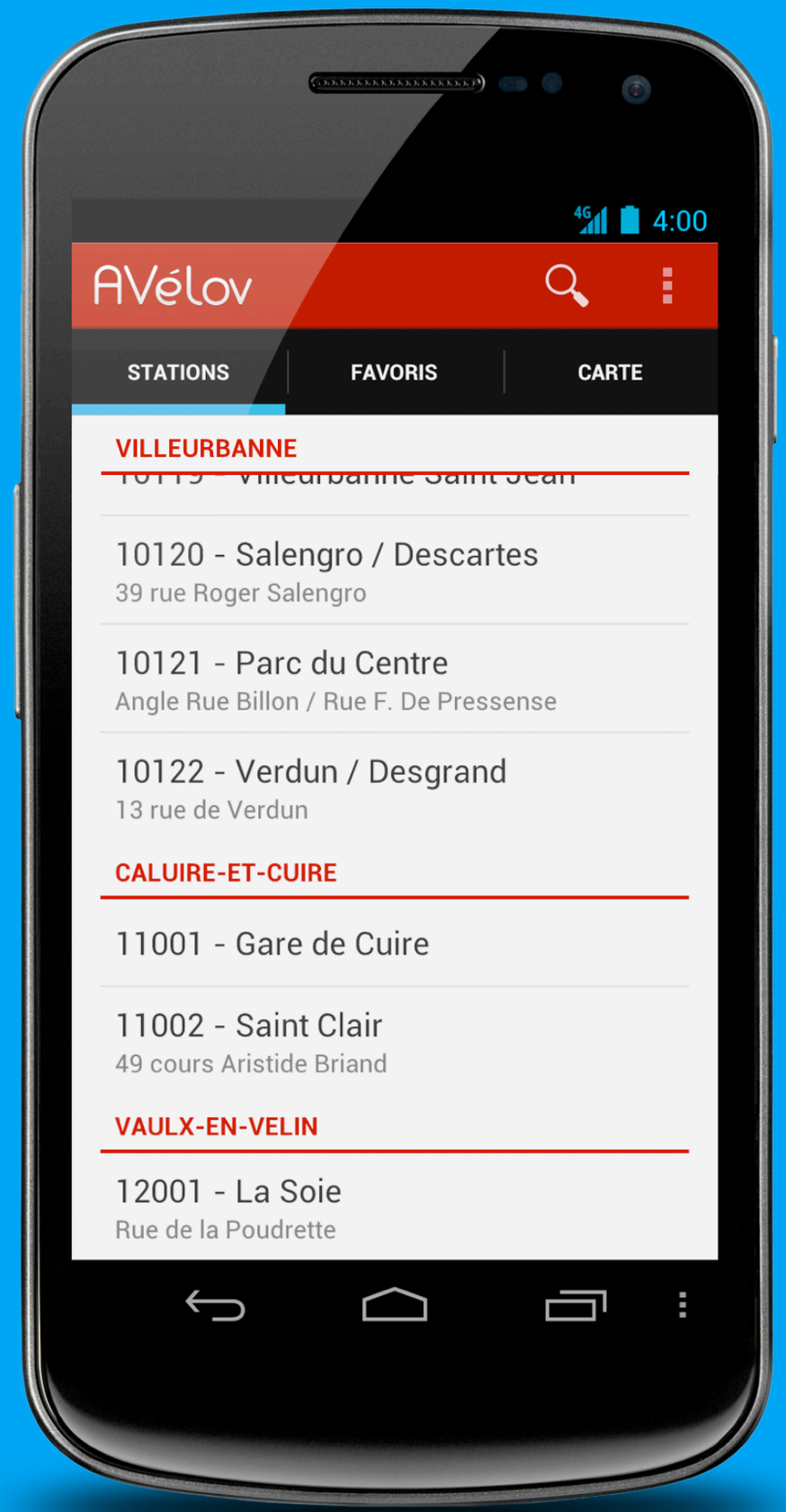
# 3 <merge />

COOK YOUR OWN VIEWS
WHENEVER NECESSARY

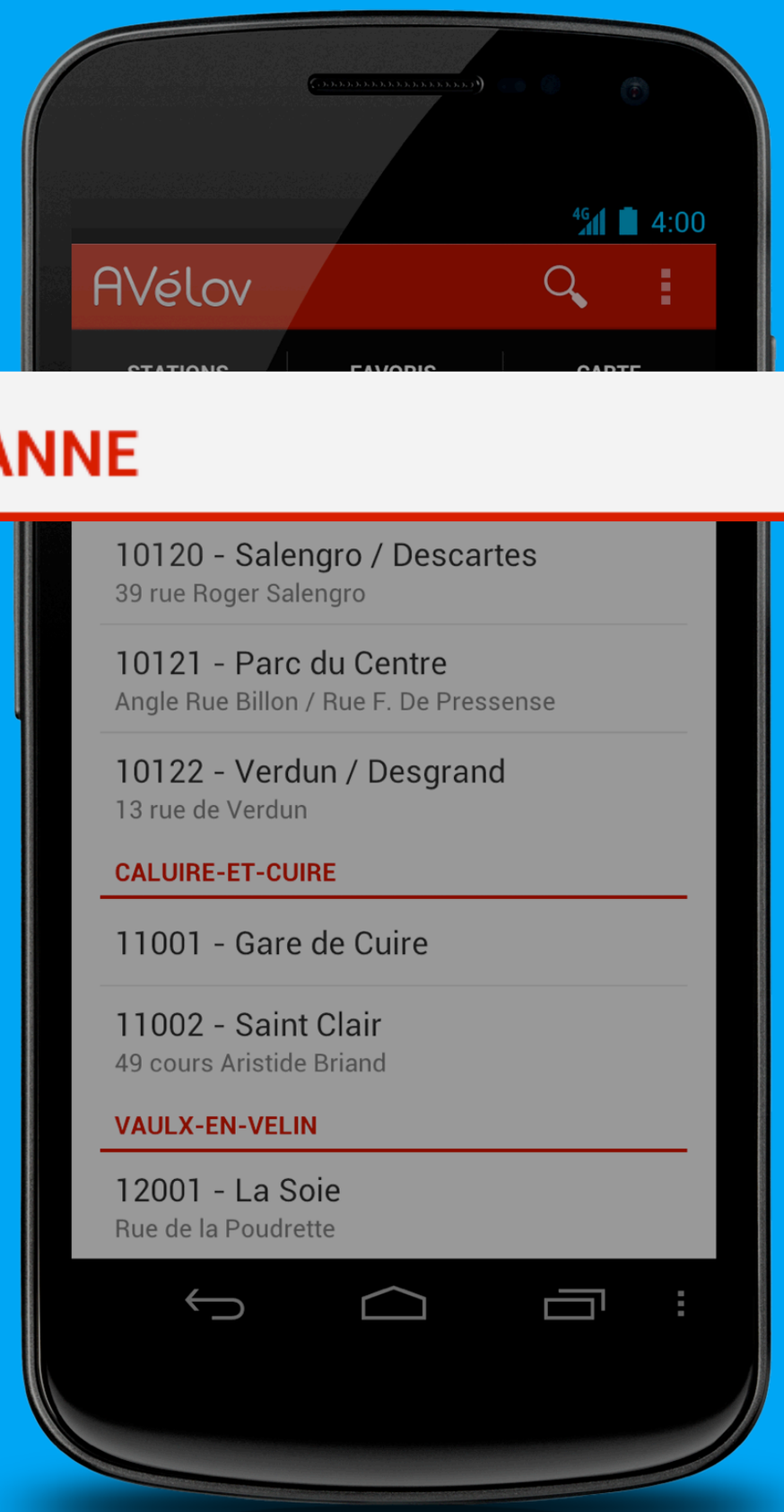



# AVélov

goo.gl/8LDDz

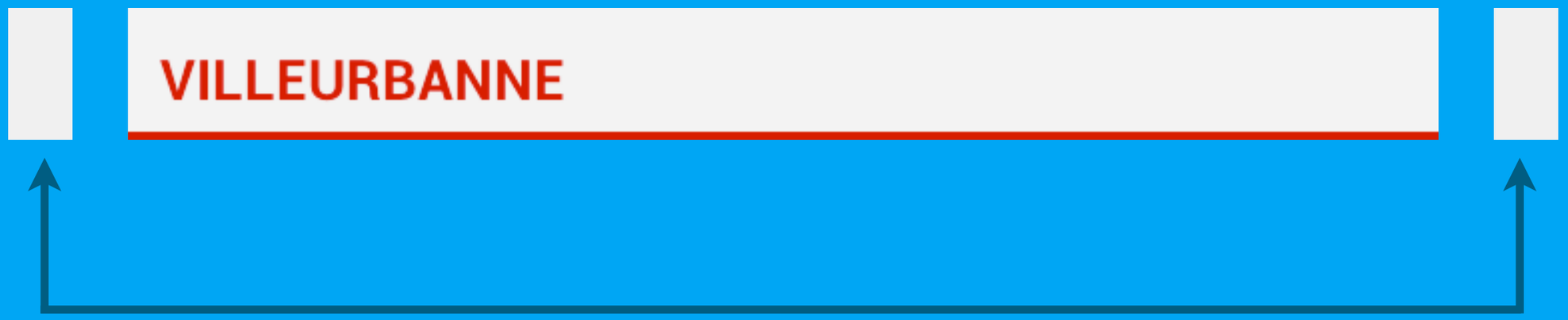

4:00
AVélov
STATIONS
FAVORIS
CARTE
VILLEURBANNE
10120 - Salengro / Descartes
39 rue Roger Salengro
10121 - Parc du Centre
Angle Rue Billon / Rue F. De Pressense
10122 - Verdun / Desgrand
13 rue de Verdun
CALUIRE-ET-CUIRE
11001 - Gare de Cuire
11002 - Saint Clair
49 cours Aristide Briand
VAULX-EN-VELIN
12001 - La Soie
Rue de la Poudrette

# VILLEURBANNE



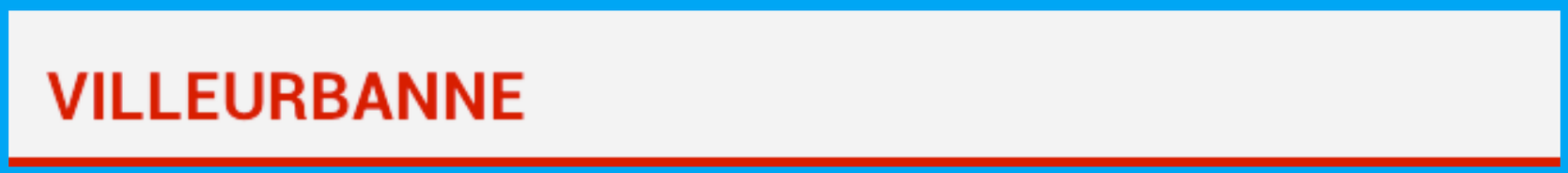
VILLEURBANNE

VILLEURBANNE


VILLEURBANNE
ListView's padding

# QUESTION

What are the available
techniques to create such a
a great ListView header?



```
public class UnderlinedTextView extends TextView {

    private final Paint mPaint = new Paint();
    private int mUnderlineHeight;

    public UnderlinedTextView(Context context) {
        super(context);
    }

    public UnderlinedTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public UnderlinedTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
}
```

```
public class UnderlinedTextView extends TextView {

    private final Paint mPaint = n
    private int mUnderlineHeight;

    public UnderlinedTextView(Context context) {
        super(context);
    }

    public UnderlinedTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public UnderlinedTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
}
```

extends TextView

```
@Override public void setPadding(int left, int top, int right, int bottom) {
    super.setPadding(left, top, right, mUnderlineHeight + bottom);
}

public void setUnderlineHeight(int underlineHeight) {
    if (underlineHeight < 0) underlineHeight = 0;
    if (underlineHeight != mUnderlineHeight) {
        mUnderlineHeight = underlineHeight;
        super.setPadding(getPaddingLeft(), getPaddingTop(), getPaddingRight(),
                getPaddingBottom() + mUnderlineHeight);
    }
}

public void setUnderlineColor(int underlineColor) {
    if (mPaint.getColor() != underlineColor) {
        mPaint.setColor(underlineColor);
        invalidate();
    }
}

@Override protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawRect(0, getHeight() - mUnderlineHeight, getWidth(), getHeight(), mPaint);
}
```

```
@Override public void setPadding(int left, int top, int right, int bottom) {
    super.setPadding(left, top, right, mUnderlineHeight + bottom);
}

public void setUnderlineHeight(int underlineHeight) {
    if (underlineHeight < 0) underlineHeight = 0;
    if (underlineHeight != mUnderlineHeight) {
        mUnderlineHeight = underlineHeight;
        super.setPadding(getPaddingLeft(), getPaddingTop(), getPaddingRight(),
                getPaddingBottom() + mUnderlineHeight);
    }
}

public void setUnderlineColor(int underlineColor) {
    if (mPaint.getColor() != underlineColor) {
        mPaint.setColor(underlineColor);
        invalidate();
    }
}

@Override protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawRect(0, getHeight() - mUnderlineHeight, getWidth(), getHeight(), mPaint);
}
```

use padding to avoid measurements

use padding to avoid measurements

```
@Override public void setPadding(int left, int top, int right, int bottom) {
    super.setPadding(left, top, right, mUnderlineHeight + bottom);
}

public void setUnderlineHeight(int underlineHeight) {
    if (underlineHeight < 0) underlineHeight = 0;
    if (underlineHeight != mUnderlineHeight) {
        mUnderlineHeight = underlineHeight;
        super.setPadding(getPaddingLeft(), getPaddingTop(), getPaddingRight(),
                getPaddingBottom() + mUnderlineHeight);
    }
}

public void setUnderlineColor(int underlineColor) {
    if (mPaint.getColor() != underlineColor) {
        mPaint.setColor(underlineColor);
        invalidate();
    }
}

@Override protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawRect(0, getHeight() - mUnderlineHeight, getWidth(), getHeight(), mPaint);
}
```

use padding to avoid measurements

use padding to avoid measurements

extend TextView default behavior

# ASSERTION

## DEVELOPERS TEND TO BE «NEVER SATISFIED»
## PEOPLE THAT ARE ALWAYS
# ASKING FOR MORE

Yep, this is the exact same
definition of «being French»

# VIEW & VIEWGROUP

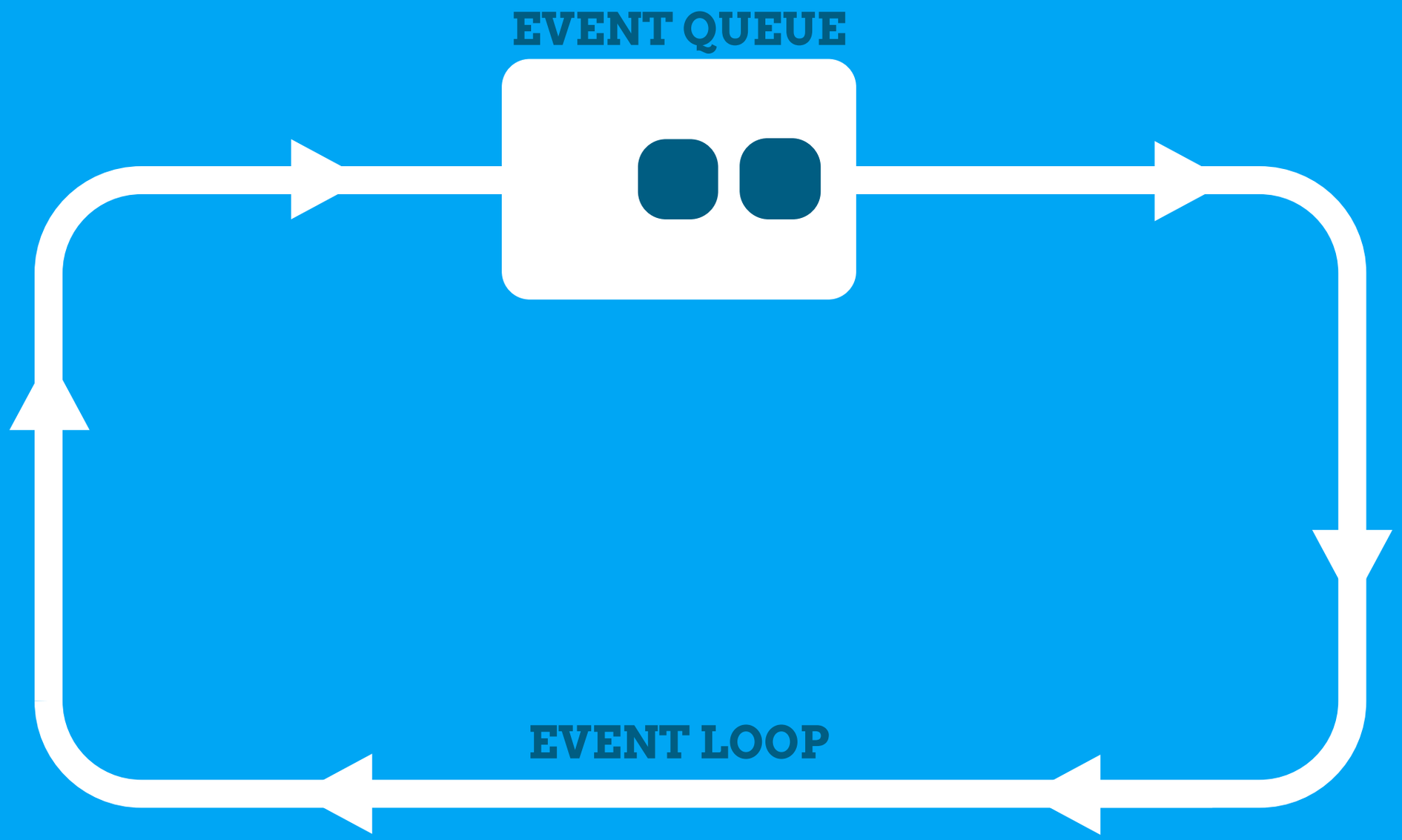
have been designed to be easily extended via inheritance

# GET OFF THE MAIN THREAD

# DON'T BLOCK THE MAIN THREAD

# TRUE FACT

## THE MAIN THREAD
## IS WHERE ALL EVENTS
## ARE DISPATCHED


EVENT QUEUE
EVENT LOOP

# BE MINDFUL WITH CRITICAL METHODS

onMeasure
onLayout
onDraw
onTouchEvent

3

# MAIN OPERATIONS

## to perform in background

disc
network
hardware

# MOVE WORK OFF THE MAIN THREAD

# THE JAVA WAY...

Plain old Java
java.util.concurrent

# THE JAVA WAY...

FOR BEARDED MEN ONLY

ed Java

java.util.concurrent

# YOU ARE MORE THIS KIND OF GUY?

# THE ANDROID WAY...

Handler
AsyncTask
Loader
IntentService

# DO LESS AND APPROPRIATELY

# CACHE VALUES NOT TO OVER COMPUTE THEM

# DO YOU REMEMBER THIS?

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = mInflater.inflate(
                android.R.layout.two_line_list_item, // resource
                parent,                              // parent
                false);                              // attach
    }

    ((TextView) convertView.findViewById(android.R.id.text1))
        .setText(TITLES.get(position));
    ((TextView) convertView.findViewById(android.R.id.text2))
        .setText(SUBTITLES.get(position));

    return convertView;
}
```

# QUESTION

## How to avoid having findViewById executed at every call to getView?

# QUESTION

## How to avoid having findViewById executed at every call to getView?

# ANSWER

## Caching it!

# VIEW HOLDER

```
static class ViewHolder {
        TextView text1;
        TextView text2;
}
```

# AVOID REPETITIVE EXECUTIONS

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    ViewHolder holder;

    if (convertView == null) {
        convertView = mInflater.inflate(
            android.R.layout.two_line_list_item, parent, false);

        holder = new ViewHolder();
        holder.text1 = (TextView) convertView.findViewById(android.R.id.text1);
        holder.text2 = (TextView) convertView.findViewById(android.R.id.text2);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.text1.setText(STRINGS.get(position));
    holder.text2.setText(STRINGS.get(position));

    return convertView;
}
```

# AVOID REPETITIVE EXECUTIONS

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    ViewHolder holder;

    if (convertView == null) {
        convertView = mInflater.inflate(
            android.R.layout.two_
        holder = new ViewHolder();
        holder.text1 = (TextView) convertView.findViewById(android.R.id.text1);
        holder.text2 = (TextView) convertView.findViewById(android.R.id.text2);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.text1.setText(STRINGS.get(position));
    holder.text2.setText(STRINGS.get(position));

    return convertView;
}
```

cache refs to the children

# AVOID REPETITIVE EXECUTIONS

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    ViewHolder holder;

    if (convertView == null) {
        convertView = mInflater.inflate(
            android.R.layout.two_
        holder = new ViewHolder();
        holder.text1 = (TextView) convertView.findViewById(android.R.id.text1);
        holder.text2 = (TextView) convertView.findViewById(android.R.id.text2);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.text1.setText(STRINGS.get(position));
    holder.text2.setText(STRINGS.get(position));

    return convertView;
}
```

cache refs to the children

retrieve cached refs

# AVOID REPETITIVE EXECUTIONS

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    ViewHolder holder;

    if (convertView == null) {
        convertView = mInflater.inflate(
            android.R.layout.two_
        holder = new ViewHolder();
        holder.text1 = (TextView) convertView.findViewById(android.R.id.text1);
        holder.text2 = (TextView) convertView.findViewById(android.R.id.text2);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.text1.setText(STRINGS.get(position));
    holder.text2.setText(STRINGS.get(position));

    return convert
}
```

cache refs to the children

retrieve cached refs

use cached refs

GIVE PRORITITY TO YOUR THREADS

```
private static final int MSG_LOAD = 0xbeef;
private Handler mHandler = new Handler(Looper.getMainLooper());

public void loadUrl(final String url) {
    new Thread(new Runnable() {
        @Override public void run() {
            Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);

            InputStream i = null;
            Bitmap b = null;
            try {
                i = new URL(url).openStream();
                b = BitmapFactory.decodeStream(i);
            } catch (Exception e) {
            } finally {
                if (i != null) try { i.close(); } catch (IOException e) {}
            }

            Message.obtain(mHandler, MSG_LOAD, b).sendToTarget();
        }
    }).start();
}
```

```
private static final int MSG_LOAD = 0xbeef;
private Handler mHandler = new Handler(Looper.getMainLooper());

public void loadUrl(final String url) {
    new Thread(new Runnable() {
        @Override public void run() {
            Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);

            InputStream i = null;
            Bitmap b = null;
            try {
                i = new URL(url).openStream();
                b = BitmapFactory.decodeStream(i);
            } catch (Exception e) {
            } finally {
                if (i != null) try { i.close(); } catch (IOException e) {}
            }

            Message.obtain(mHandler, MSG_LOAD, b).sendToTarget();
        }
    }).start();
}
```

set a low priority

```
private static final int MSG_LOAD = 0xbeef;
private Handler mHandler = new Handler(Looper.getMainLooper());

public void loadUrl(final String url) {
    new Thread(new Runnable() {
        @Override public void run() {
            Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);

            InputStream i = null;
            Bitmap b = null;
            try {
                i = new URL(url).openStream();
                b = BitmapFactory.decodeStream(i);
            } catch (Exception e) {
            } finally {
                if (i != null) try { i.close(); } catch (IOException e) {}
            }

            Message.obtain(mHandler, MSG_LOAD, b).sendToTarget();
        }
    }).start();
}
```

set a low priority

post a Message to callback the UI Thread

# FAVOR UI TO BACKGROUND COMPUTATIONS

# ListView can have

# 3 SCROLL STATES

The current ListView's scroll state can be obtained using an OnScrollListener

1 IDLE

2 TOUCH SCROLL

3 FLING

# 1 IDLE

The view is not scrolling

# 2 TOUCH SCROLL

The user is scrolling using touch, and their finger is still on the screen

The user had previously been scrolling using touch and had performed a fling. The animation is now coasting to a stop

Avoid blocking the animation pause worker Threads

3 FLING

```
getListView().setOnScrollListener(mScrollListener);

private OnScrollListener mScrollListener = new OnScrollListener() {

    @Override
    public void onScrollStateChanged(AbsListView view, int scrollState) {
        ImageLoader imageLoader = ImageLoader.get(getContext());
        imageLoader.setPaused(scrollState == OnScrollListener.SCROLL_STATE_FLING);
    }

    @Override
    public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount,
        int totalItemCount) {
        // Nothing to do
    }
};
```

```
getListView().setOnScrollListener(mScrollListener);

private OnScrollListener mScrollListener = new OnScrollListener() {

    @Override
    public void onScrollStateChanged(AbsListView view, int scrollState) {
        ImageLoader imageLoader = ImageLoader.get(getContext());
        imageLoader.setPaused(scrollState == OnScrollListener.SCROLL_STATE_FLING);
    }

    @Override
    public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount,
        int totalItemCount) {
        // Nothing to do
    }
};
```

set the listener

```
getListView().setOnScrollListener(mScrollListener);

private OnScrollListener mScrollListener = new OnScrollListener() {

    @Override
    public void onScrollStateChanged(AbsListView view, int scrollState) {
        ImageLoader imageLoader = ImageLoader.get(getContext());
        imageLoader.setPaused(scrollState == OnScrollListener.SCROLL_STATE_FLING);
    }

    @Override
    public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount,
        int totalItemCount) {
        // Nothing to do
    }
};
```

set the listener

(un)pause ImageLoader

# CONCLUSION

# DO NOT BLOCK THE MAIN THREAD

# FLATTEN YOUR VIEW HIERARCHY

# LAZY LOAD AND REUSE WHENEVER POSSIBLE

# PRIORITIZE TASKS: UI ALWAYS COMES FIRST

# CYRIL MOTTIER

@cyrilmottier

android.cyrilmottier.com