

Android practicum part 3

# Introduction

In Hollywood it’s all about….



\*Puts on sunglasses\*

Style..

Now that we have a rudimentary version of the CSI app from the previous week, we will now continue with style.

CSI is still the subject, but this time we will use the Miami spinoff.

This means: Warm ocher yellow, dry soft orange, and scorching hot matt black.

Focus is on styles, and we will add a list view to the app.

As a cherry on the cake, we’ll send data to activities by using Intents.

Good luck, and have fun.

# New project

Part 1: Copy thatproject

Usually it’s a bad idea to jus blindly copy a project, but to highlight the details between this week’s app and last week’s let’s do so anyway. Create a copy of CSI\_week1 directory using your favorite file explorer. After copying add your project to your repository and open the project using your development environment (probably Android Studio).

# UI style

Part 1: Once again, with style

In your layout xml files the colors, fonts and images are now specified per UI item. They are ‘hardcoded’ in the layout.

What if your company’s marketing department decides that a popular TV series is getting a spin off (in this case CSI -> CSI Miami)?

Now you have to adjust ALL the layout files.

This can be prevented by keeping the layout and the style separated.

A lot of information about styles can be found at the official developer site, so go ahead and read it:

<http://developer.android.com/guide/topics/ui/themes.html>

Create an overview of the UI elements that you have used so far: Buttons, tables, images etc.

Which elements of these UI components can be moved to a separate style?

Separate style:

* Button
* Images
* Text
* Layouts

After creating the overview, go right ahead and move the style attributes to a separate style resource.

Tips:Use the Eclipse style extractor to extract the styles.

Make sure that the UI components use the extracted style.

When making these adjustments, you’ll occasionally notice that not all elements use the same style, but share quite a bit of attributes.

For example all buttons should have the same font and background color, but gravity and alignment are not the same.

This can be optimized by using style inheritance.

You do this by first determining a set of all common attributes.

Define the base styles and set the shared attributes.

Then create the deviating styles, and let them ‘inherit ‘ from the base style.

Let’s take a look at an example.

Parent:

    <style name="CodeFont.Red">  
        <item name="android:textColor">#FF0000</item>  
    </style>

Afgeleide:

    <style name="CodeFont.Red.Big">  
        <item name="android:textSize">30sp</item>  
    </style>

Part 2: More style

Since this week’s theme is CSI Miami, the styles should be adjusted to reflect this. Be creative with colors. Use warm yellow/orange notes, and less sharp transitions. Make sure that all colors are nicely placed in a colors resource file.

For more information about colors, see:

[http://developer.android.com/guide/topics/resources/more-resources.html#Color](http://developer.android.com/guide/topics/resources/more-resources.html" \l "Color)

Part 3: Most style

By using drawables, you can adjust the shape and appearance of UI elements.

Take a look at the following link for details about drawables:

[http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape](http://developer.android.com/guide/topics/resources/drawable-resource.html" \l "Shape)

Add a new XML file to the res/drawable folder of your project.

(Tip: Use file -> new -> Android XML file, and as a resource type chose “Drawable”. This helps you with setting up the basic XML structure).

Call it buttonshape.xml.

In this drawable you must describe a button shape. Keep the following directives in mind:

* The button should be rectangular.
* It should be gradient from left to right, where left is darker than right.
* The top left and bottom right corner should be curved. (use 25 dp for the angle of the corner). Bonus points if you use a dimension resource.

Now adjust the styles.xml file, so all buttons use this newly defined gradient.

Bonus: a common issue when using custom buttons is, that it will no longer give visual feedback when it is selected.You can fix this by using a color state list. See here for details:

<http://developer.android.com/guide/topics/resources/color-list-resource.html>

Repeat the steps above to create a new XML file, and call it lineshape.xml.

The shape should represent a line, with a solid color (any color, as long as it’s read from a resource file) and should have a thickness of 5dp.

The line should be drawn beneath the picture of the criminal.

# **List**

Part 1: More activity

Create a new activity, just like in the previous practicum.

Als add a new layout xml file to correspond with this activity. Give these files a good name, such as ‘criminalslist’. Override the Oncreate of the activity and load the layout xml file.

Part 2: First activity

The criminalslist activity is going to show a list with criminals. It makes sense that it is the first activity that is shown when the app is started. Make adjustments to manifest to achieve this. Take note of the highlighted lines. Make sure that you use the name chosen in step 1.

<application

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"* >

<activity

android:name=*"~~.CSI\_week2Activity~~" -> “.criminalslistactivity”*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*".reportActivity"*></activity>

<activity android:name=*".CSI\_week2Activity"*></activity>

</application>

Part 3: Activity UI

Maak de UI van deze activity.

Construct the UI of the activity. Start with a vertical linear layout and add a listview to it. Don’t forget to create/use styles for the components.

The default listview looks a bit boring. Take a look at the documentation about what options there are regarding ListView styles. Be creative and make something pretty.

Part 4: Data

Create a new XML file in the res/values folder, and call it criminals.xml.

Add an array list to this xml file and add 3 fictional names.

It might look like this

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<resources>

<string-array name=*"names"*>

<item >John Cheese</item>

<item >Bert</item>

<item >Gil G</item>

</string-array>

</resources>

Part 5:show the list

Adjust the OnCreate method of the criminalsListActivity so the list gets loaded and shown.

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*criminalslist*);

//Get a reference to the listview

ListView listview = (ListView) findViewById(R.id.*listView1*);

//Get a reference to the list with names

**final** String[] criminals = getResources().getStringArray(R.array.*names*);

//Create an adapter that feeds the data to the listview

listview.setAdapter(

**new** ArrayAdapter<String>(

**this**,

android.R.layout.*simple\_list\_item\_1*,

criminals

)

);

}

Part 6: Choose

When a criminal is selected from the list, the details should be shown on the next activity.

But how do you know which criminal was selected?

One way to do this, is by using the OnItemClickListener interface that the ListView can respond to. When the event is fired, you can check which item is selected in the list, and choose the corresponding item from the array list. You can add a new OnItemClickListener to the listview, just like you did with the OnClickListener for a button:

listview.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> parent, View view,

**int** position, **long** id) {

//Get the name from the array that is in the same position as the chosen listitem.

String name = criminals[position];

//Todo start intent and pass name using putExtra

}

});

For now the details activity should only show the name of the selected activity. Next week we will work on showing the image, description etc.

You can do this by creating a new Intent, and use it to start the activity from last week.

With the method: putExtra, you can add additional data to the Intent that can be sent to the next activity. In our case, we will send the name of the selected criminal.

Now it’s up to you to finish the todo above, so that the intent is started, and the selected name is passed along.

Part 7: Show the selected criminal

When the details activity is started, the intent is passed along. This Intent contains the selected name. Use the method getStringExtra to extract this name from the intent, and put it in the ‘name’ text field.