Grow with Google Challenge Scholarship
Lesson 3 Notes

GitHub Repo Search Code

The code for this app can be found in the **Lesson02-GitHub-Repo-Search** folder of the **Toy App Repository**.
If you need to a refresher on how the code is organized, please refer the **concept where we introduced the code flow**.

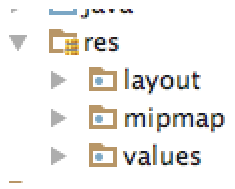
Explanation of GitHub Repo Search

The Github Repo Search app searches for a **GitHub** repository by name. The URL you'll use to get search information will be something like the URL below, which searches for repositories containing the word **android** and sorts by the number of **stars** the repo has:
https://api.github.com/search/repositories?q=android&sort=stars
Which returns information in JSON. We'll be going over how to make sense of this returned JSON, parse it and display it in your app during this lesson. We'll also cover how to connect to the internet and download data. Let's get started!

Resources

What is the res Directory?

The res directory is where you should put things such as images, strings, and layouts. It's included in every Android project, and you can see it in Android Studio here:



Inside of the res directory, are sub folder for the following types of resources. You may have a subset of these directories, depending on the types of resources you're using in your app. Here are some examples

Different Resource Directories

This information can also be found **here**.

Some Common Resource Types

Name	What's Stored Here
values	XML files that contain simple values, such as string or integers
drawable	A bunch of visual files, including Bitmap file types and shapes. More information is **here**
layouts	XML layouts for your app

Other Resource Types

Name	What's stored here
animator	XML files for property animations
anim	XML files for tween animations

Name	What's stored here
color	XML files that define state list colors
mipmap	Drawable files for launcher icons
menu	XML files that define application menus
raw	Resource file for arbitrary files saved in their raw form. For example, you could put audio files here. (You might also be interested in the assets folder, depending on how you use that audio)
xml	Arbitrary XML; if you have XML configuration files, this is a good place to put them

Why Resources

You should always keep things like images and layouts separate in the **res** folder. Keeping resource files and values independent helps you easily maintain them if you need to update, say, all your button images to match a new style. The Android Framework also easily allows for alternative resources that support specific device configurations such as different languages or screen sizes. Providing a customized experience for users from different locations or on different devices becomes increasingly important as more of the world comes online and more devices come on the market. We will see how to provide alternate resources for different configurations and locals later in this course.

Using Resources in XML and Java

You've already seen resources in action. For example, in the `MainActivity`, you have already seen usage of resources. When we say `setContentView(R.layout.activity_main)`, we are referencing a resource (the `activity_main.xml`) file to use as the layout of `MainActivity`. That magical looking R.layout part of the expression above is actually a static class that is generated for us to reference resources in Java code. This is all described in the Android Layouts Primer.

Working with strings.xml

In Java, you can get a String saved in **res** -> **values** -> **strings.xml** by calling the `getString` method. If you're in an Activity, you can just call `getString`, and pass in the String resource ID. The String resource ID can be found in the **strings.xml** XML. For example, let's look at Sunshine's **strings.xml** file:

```
    <string name="today">Today</string>
    <!-- For labelling tomorrow's forecast [CHAR LIMIT=15] -->
    <string name="tomorrow">Tomorrow</string>

    <!-- Date format [CHAR LIMIT=NONE] -->
    <string name="format_full_friendly_date">
        <xliff:g id="month">%1$s</xliff:g>, <xliff:g id="day">%2$s</xliff:g>
    </string>
```
The id of the String with the value "Today" is `today` and the id of the String with the value `<xliff:g id="month">%1$s</xliff:g>, <xliff:g id="day">%2$s</xliff:g>` is `format_full_friendly_date`

If you wanted to reference the **Today** string, you would reference it in Java by doing something like this:

```
String myString = getString(R.string.today);
```
In XML, you can access a String by using the @string accessor method. For the same String defined above, you could access it like this:

```
<TextView text="@string/today" />
```
For more information on String Resources check out the **documentation**.

Create JSON

Given that the JSON result for the Name & Title data in the video above is:

```
{
    "name": {
   "firstName": "John",
   "lastName": "Doe"
 },
    "title":  "Missing Person"
}
```

What would the following look like as JSON?

```
temp
  min = 11.34
  max = 19.01
weather
  id = 801
  condition = Clouds
  description = few clouds
pressure = 1023.51
humidity = 87
```

Reflect

Write your best guess about what the data would look like as JSON

The correct answer is:

```
{
   "temp": {
       "min":11.34,
       "max":19.01
   },
   "weather": {
       "id":801,
       "condition":"Clouds",
       "description":"few clouds"
   },
   "pressure":1023.51,
   "humidity":87
}
```

Write the function to return the condition.

Given the following JSON, write a function to retrieve the weather "condition".

```
{
    "temp": {
        "min":"11.34",
        "max":"19.01"
    },
    "weather": {
        "id":"801",
        "condition":"Clouds",
        "description":"few clouds"
    },
    "pressure":"1023.51",
    "humidity":"87"
}
```

Here is the answer:

```
String getCondition(String JSONString) {
    JSONObject forecast = new JSONObject(JSONString);
    JSONObject weather = forecast.getJSONObject("weather");
    return weather.getString("condition");
}
```