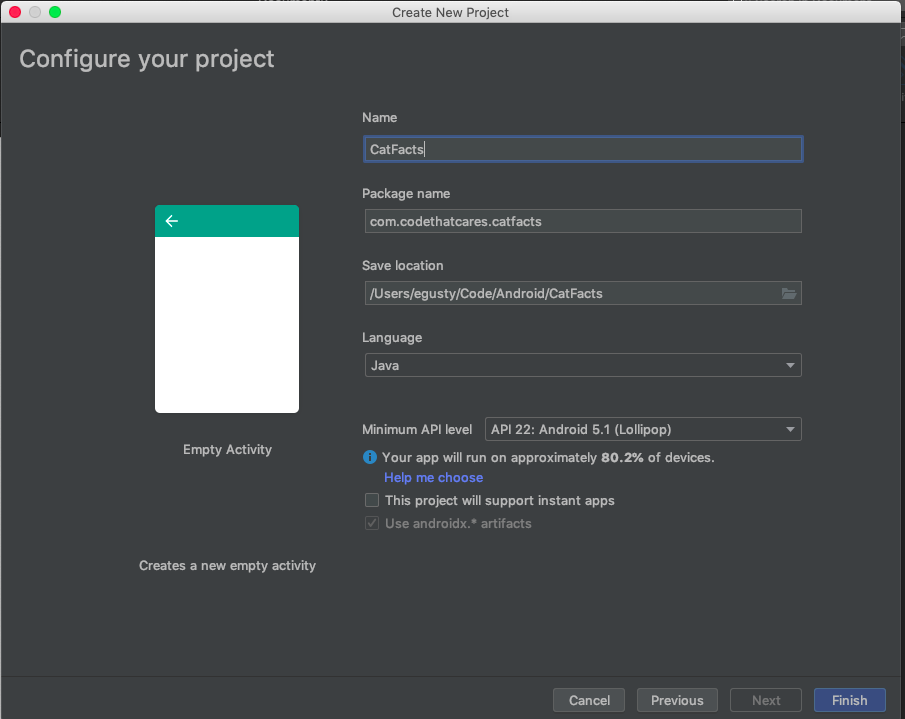
**Introduction:** In this tutorial we are going to make a simple app that shows the user a list of cat facts. While this might sound like this is a simple task, it will quickly get more intricate as we go along. This tutorial will be broken down into the following smaller steps:

1. Creating a new android project
2. Creating a scrollable list
3. Making an HTTP request on android
4. Parsing JSON data into java objects

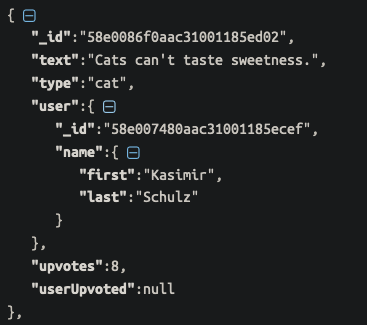
In this tutorial I hope to introduce you to the following concepts

1. HTTP requests using java
2. Android Logs
3. Recycler views and adapters
4. Android Manifest
5. Working with JSON data

**Creating a new android project:**

* Open android studio
* Create a new project
  + If a project is already open, select File > New Project
* Select ‘Empty Activity’ and click next
  + Name the project CatFacts
  + Language -> Java
  + Minimum api -> 22
  + Select ‘use androidx artifacts’ and unselect ‘this project supports instant apps’
* Once your screen looks like this one below, click finish.
* Allow android studio some time to create the project
* Run the project (on either device or emulator) by hitting the green run arrow. You should see the Hello World app on your screen. We are doing this to make sure no gradle errors exist before we start coding our project

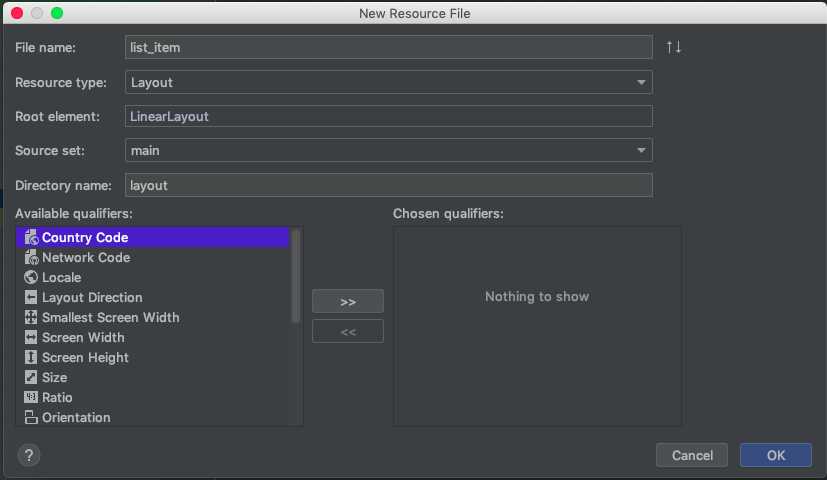
**Making a scrollable list in java/android:**

* In order to know what our list should look like we first need to know what our data looks like
  + Travel to [this](https://cat-fact.herokuapp.com/facts) url in order to get the JSON data
  + Copy the json data and paste it into a JSON formatter so we can clearly see what our object should look like. My favorite JSON formatter is [this one](https://jsonformatter.curiousconcept.com/), but you can use any one you prefer
* Our JSON object looks something like this:
* Looking at our JSON Object, our object is going to need the following fields
  + Fact/Text
  + Submitter name
  + Upvotes
* Let’s create a new Java class to model what a cat fact object should look like
  + File > new > Java Class
  + Name it CatFact
  + Type -> Class
  + Do not implement any interfaces or extends any classes
  + Click okay
* Create the java object that holds a fact (string) first name (string) last name (string) and upvotes (int)
  + Make a constructor to initialize these fields
  + Create getter methods to return these fields
  + The code should [look like this for CatFact](https://github.com/gusty9/CatFacts/blob/master/app/src/main/java/com/codethatcares/catfacts/CatFact.java)
* Now let’s create a list of cat facts, populating it from json data
  + Open up MainActivity.java
  + For now, we are going to hard code the json data, we will replace this with an http request in a future step
  + Paste this into the file above the onCreate method



* Now lets parse this Json object into a list of CatFacts
  + Create a new arraylist of CatFacts
  + Create a method where you pass the json string and the arraylist of cats
    - This method should parse all of the json objects into a new catfact object
    - See [here](https://www.testingexcellence.com/how-to-parse-json-in-java/) for more info on parsing json (use the org.json example)
  + The MainActivity should look something like [this](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/java/com/codethatcares/catfacts/MainActivity.java), your implementation could be different, and that’s okay! Just make sure it works and that it is called in the onCreate method
* Now we are going to create the recyclerview based on our list of cat facts
  + Open activity\_main.xml (res > layout > activity\_main.xml)
  + Make sure the text tab at the bottom is selected
  + Select all the code delete it
  + Paste the following into the file:



* + LinearLayout our root element. This says to display every child element one after another in the orientation described by the android:orientation attribute, so in this case vertical
  + The xmlns attributes tell the xml interpreter where to find libraries used in layout files, you don’t need to worry about these. Just remember to include them on the root node.
  + The android layout height and width attributes describe how big the view should be. In this case it will match the parent, but things like wrap content and specific sizes are also used frequently
  + The android:id tag assigns a variable name to the view so it can be used in java code
  + Now we are going to add a child of linear layout. We are going to use a RecyclerView so we can create our list of CatFacts.
  + Add the recyclerview (the tag is called androidx.recyclerview.widget.RecyclerView)
  + Give this element an id, and have it fill the screen
  + The file should now look like [this](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/res/layout/activity_main.xml)
* What is a recycler view?
  + If the android system had to load every element in our list at one time, our phone would light on fire because that is a difficult task
  + Recyclerviews handle what list item to show while scrolling to avoid slowdowns on the android system
  + To learn more about recyclerviews, check out this [resource](https://guides.codepath.com/android/using-the-recyclerview)
* Using recyclerview
  + First and foremost to use this component we have to import it to the android project
    - Go to Gradle Scripts > build.gradle (module: app)
    - At the bottom of the file there should be many lines that start with ‘implementation’, add the following line with the others:
    - implementation 'androidx.recyclerview:recyclerview:1.0.0'
    - you have now successfully added the recycler view android library, click ‘Sync Now’ on the top of the editor to sync the gradle script with the project
  + Now that we have our recycler view, we need to define what each row looks like
    - File > new > android resource file
    - Call the file ‘list\_item’
    - Resource type should be ‘Layout’
    - Make sure the fields match the ones below, and click ok
* In order to describe our row we are going to use a ConstraintLayout
  + Constraint layouts allow you to place elements relative to one another. Constraint Layout is the most powerful layout and is used the most in android development
  + The layout file should look like [this](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/res/layout/list_item.xml)
* **Creating a recyclerview adapter**
  + Create a new Java class and call it CatAdapter
  + The recycler view adapter is in charge of supplying data to each row of the list
  + Copy and paste [this](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/java/com/codethatcares/catfacts/CatAdapter.java) code into the class. Read through it to try and see what is does.
* **Setting up the recycler view**
  + Open up MainActivity.java
  + Create a method called setUpRecyclerView
  + This method should link the resource view to the java object, and set up the view adapter
  + The method should look like the one in this [file](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/java/com/codethatcares/catfacts/MainActivity.java)
* **Creating an HTTP GET Request**
  + Create a new java class called CatHttpRequestTask
  + When making a request to the web we need to do it In another [Thread](https://developer.android.com/guide/components/processes-and-threads?authuser=1)
    - If it was done on the main thread, the user would not be able to interact with the app while data is downloaded.
  + Make the class extend AsyncTask<String, Void, String> The three parameterize types indicate the input to the thread (a string url), the progress update output (in this case void, we do not want progress updates), and the output (the string of json)
  + An async task has 3 main methods
    - **onPreExecute**
    - **doInBackground**
    - **onPostExecute**
  + On pre execute is done before the task on the ui thread
  + doInbackground is the task in the background thread
  + onPostExecute in on the ui thread after the task is finished
  + Make the class look like [this](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/java/com/codethatcares/catfacts/CatHttpRequestTask.java)
  + You might notice that there is a CallbackInterface that we have not yet implemented, so create a new interface called [CallbackInterface](https://github.com/gusty9/CatFacts/blob/parsingJsonStep/app/src/main/java/com/codethatcares/catfacts/CallbackInterface.java)
    - Implement this interface in your main activity
    - Having a activity communicate through a thread is easiest through an interface
* **The final of state of the code looks like** [**this**](https://github.com/gusty9/CatFacts/tree/parsingJsonStep)**. This tutorial skipped a few steps.**