

STUDENT GUIDE: VOLLEY WITH KOTLIN

JOHN P. BAUGH, PH.D.
V. 2023.3.27

A SIMPLE EXAMPLE USING VOLLEY WITH KOTLIN

1. Create a new project
 - a. I named mine **VolleyTestKotlin**
 - b. Use the Empty Activity template
 - c. Make sure the language is Kotlin
2. In the build.gradle (Module:app) you should add the following line to the dependencies near the bottom, in order to download/use Volley

```
dependencies {  
  
    implementation 'androidx.core:core-ktx:1.7.0'  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.8.0'  
    implementation  
    'androidx.constraintlayout:constraintlayout:2.1.4'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-  
core:3.5.1'  
    implementation 'com.android.volley:volley:1.2.1'  
}
```

3. Make sure to Sync Now in the build.gradle file
4. Enable view binding
 - a. In the **Module build.gradle** file, add the

```
buildFeatures {  
    viewBinding true  
}
```
 - b. Select **Sync Now** in the upper right corner
5. Open res/ activity_main.xml layout
6. Drag a button into the ConstraintLayout
 - a. Name it something like **btnGetCatData**
 - b. Change the text to “Get Cat Data”
 - c. Make sure it is constrained properly
7. In **MainActivity.kt**, code the following:

```
package com.profjpbaugh.myapplication  
  
import androidx.appcompat.app.AppCompatActivity
```

```

import android.os.Bundle
import
com.profpbaugh.myapplication.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)

        setContentView(binding.root)

        //set button handler
        binding.btnGetCatData.setOnClickListener {
            printCatData() //call this other function
        }

    }

    // method to interact with API
    fun printCatData() {

    } //end printCatData
}

```

8. Sign up for a Free Cat API key under **Pricing** if you haven't already

Signup for an API key

It's completely free, email is only be used to send you an API Key & your stats, and you can use the API as much as you like. No Spam Ever.

E-mail
✉ jpbaugh@umich.edu

App Description
? Cat API example for class

What type of project will you use the API for?

☐ A personal project

☒ A project for school/college/university

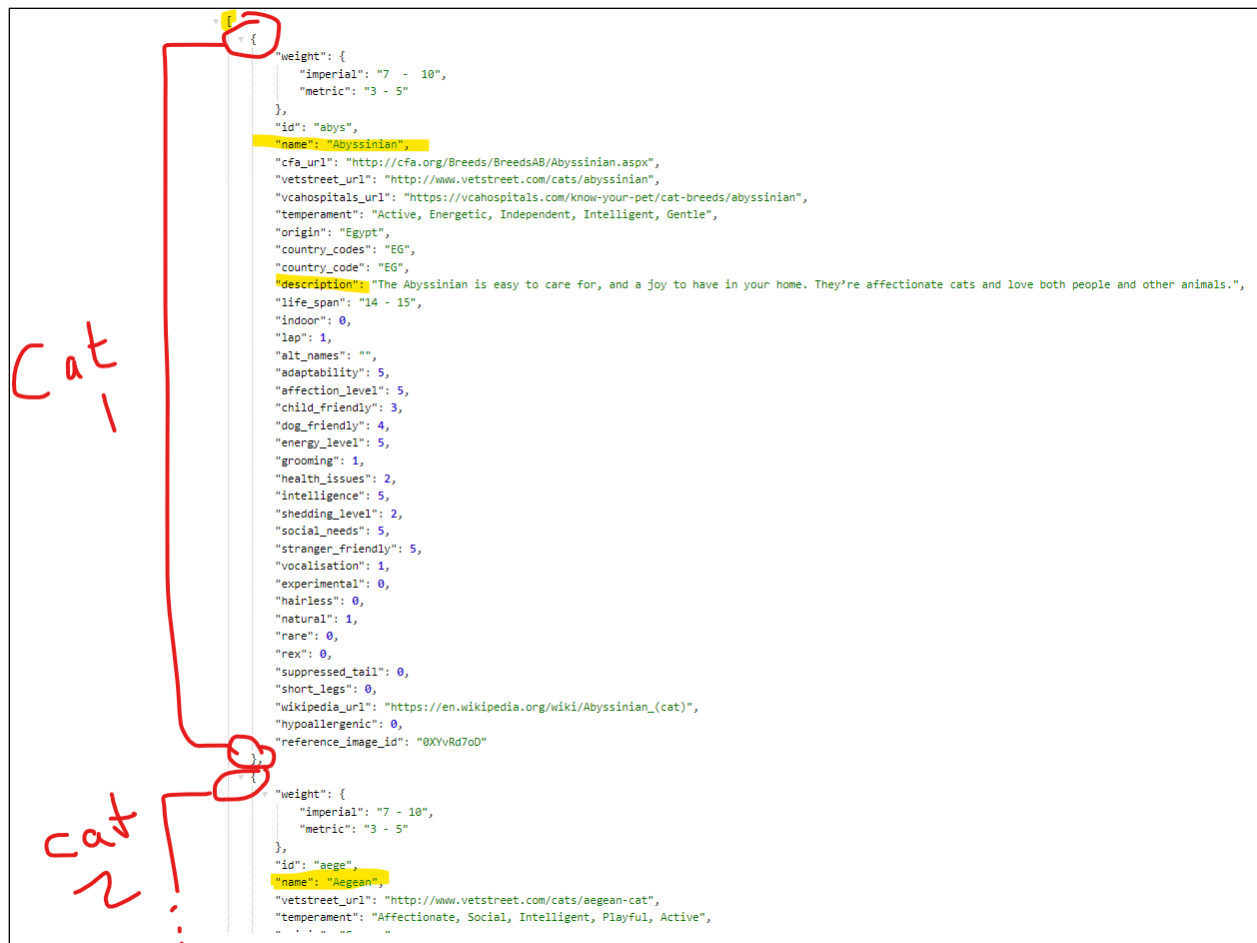
☐ A business project

☐ Opt-in to my newsletter about other APIs i'm building?

SIGNUP

[Privacy Policy](#) | [Terms & Conditions](#)

9. You can view different URL options under the **Documentation** on thecapapi.com if you go under **OpenAPI Spec Doc**
10. E.g., look at the breeds documentation
 - a. <https://developers.thecatapi.com/view-account/yIX4bIBYT9FaoVd6OhvR?report=aZyiLrsCh#tag/Breeds>
11. In a browser, go the the following API URL to view what the JSON for all the breeds looks like:
<https://api.thecatapi.com/v1/breeds>
12. Look at the JSON



In the JSON format, the [] (square brackets) means we're dealing with an array of data. Anything within { } (curly braces) is an object. Sometimes there are sub-objects (e.g., note that **weight** has its own set of curly braces because it contains properties since it's an object itself.)

We'll focus on the **name** and the **description** properties.

13. Under **Authentication** you'll notice they say what a query string (query parameter) call to the API looks like:

- a. https://api.thecatapi.com/v1/images/search?api_key=YOUR_API_KEY

14. We will use, for the breed information:

15. https://api.thecatapi.com/v1/breeds?api_key=your_api_key

- a. Note the question mark acts as a delimiter between the URL and the query string, such as the `api_key`

16. In order to access the API, we must have our app request permission to use the Internet

- a. Therefore, in the **AndroidManifest.xml** (under **manifests** in your project), you must add the following:

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET">
    </uses-permission>
    <application
```

17. Now, inside MainActivity, we will fill out the printCatData
 - a. This is based on the documentation for Volley, which can be found at <https://google.github.io/volley/simple.html>
18. [refers to step 19, below] For various code below, such as **Request** and **StringReuest**, which are part of Volley, note that you will have to do an Alt+Enter, or enter the package imports manually near the top so that the compiler knows where they live
 - a. These come from the Volley library you imported using Gradle near the beginning of this tutorial
 - b. Also note **Log** in the following is part of Android, and logs to the console (it's similar to println()) – it needs imported as well
19. Add the code to MainActivity.kt, as indicated

```
// method to interact with API
fun printCatData() {
    // make sure to replace the fake API key below with
    // your own
    var catUrl = "https://api.thecatapi.com/v1/breeds" +
        "?api_key=USE_YOUR_OWN_KEY_YOU_NAUGHTY_THIEF"

    val queue = Volley.newRequestQueue(this)

    // Request a string response from the provided URL.
    val stringRequest = StringRequest(
        Request.Method.GET, catUrl,
        Response.Listener<String> { response ->
            Log.i("MainActivity", response.toString())
        },
        Response.ErrorListener {
            Log.i("MainActivity", "That didn't work!")
        })
}
```

```
// Add the request to the RequestQueue.
queue.add(stringRequest)
} //end printCatData
```

20. Run the app

21. Click the button

22. Notice the output in both the Run window and the Logcat window below in your app

```
I/MainActivity: [{"weight":{"imperial":"7 - 10","metric":"3 - 5"},"id":"abys","name":"Abyssinian","cfa_url":"http://cfa.org/Breeds/BreedsAB/Abyssinian.aspx",
vetstreet_url":"http://www.vetstreet.com/cats/abyssinian","vcahospitals_url":"https://vcahospitals.com/know-your-pet/cat-breeds/abyssinian","temperament":"Active, Energetic,
Independent, Intelligent, Gentle","origin":"Egypt","country_codes":"EG","country_code":"EG","description":"The Abyssinian is easy to care for, and a joy to have in your home.
They're affectionate cats and love both people and other animals.","life_span":"14 - 15","indoor":0,"lap":1,"alt_names":"","adaptability":5,"affection_level":5,
child_friendly":3,"dog_friendly":4,"energy_level":5,"grooming":1,"health_issues":2,"intelligence":5,"shedding_level":2,"social_needs":5,"stranger_friendly":5,"vocalisation":1,
experimental":0,"hairless":0,"natural":1,"rare":0,"rex":0,"suppressed_tail":0,"short_legs":0,"wikipedia_url":"https://en.wikipedia.org/wiki/Abyssinian_(cat)",
hypoallergenic":0,"reference_image_id":"8XYvRd7o0","image":{"id":"8XYvRd7o0","width":1284,"height":1445,"url":"https://cdn2.thecatapi.com/images/8XYvRd7o0.jpg"}},
{"weight":{"imperial":"7 - 10","metric":"3 - 5"},"id":"","name":"","vetstreet_url":"","cfa_url":"","temperament":"","affectionate":0,"social":0}
]
```

- That's a great first step, but not quite what we want – we want to parse the JSON and display it in a more meaningful format
- You'll want to display it in various views (widgets) in your app, but we'll just use the console in this tutorial

23. Let's do that in the printCatData method

- Again, you'll need to import the packages for JSONArray and JSONObject

24. Code in MainActivity.kt, updated:

```
// method to interact with API
fun printCatData() {
    var catUrl = "https://api.thecatapi.com/v1/breeds" +
        "?api_key=USE_YOUR_OWN_YA_SILLY_GOOSE"

    val queue = Volley.newRequestQueue(this)

    // Request a string response from the provided URL.
    val stringRequest = StringRequest(
        Request.Method.GET, catUrl,
        Response.Listener<String> { response ->
            var catsArray : JSONArray = JSONArray(response)

            //indices from 0 through catsArray.length()-1
            for(i in 0 until catsArray.length()) {
                //${} is to interpolate the string /
                // uses a string template
                var theCat : JSONObject = catsArray.getJSONObject(i)

                //now get the properties we want: name and description
                Log.i("MainActivity",
                    "Cat name: ${theCat.getString("name")}")
                Log.i("MainActivity",
                    "Cat description: ${theCat.getString("description")}")
            } //end for
        }
    ),
```

```

        Response.ErrorListener {
            Log.i("MainActivity", "That didn't work!")
        })

// Add the request to the RequestQueue.
queue.add(stringRequest)
} //end printCatData

```

25. Run it
26. Observe the output under Run, or LogCat
27. Yaaay

ENTIRE MAINACTIVITY.KT FILE SOURCE CODE

For your reference, here's the code for all of MainActivity.kt

```

package com.profpjbbaugh.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import com.android.volley.Request
import com.android.volley.Response
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley
import com.profpjbbaugh.myapplication.databinding.ActivityMainBinding
import org.json.JSONArray
import org.json.JSONObject

class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)

        setContentView(binding.root)

        //set button handler
        binding.btnGetCatData.setOnClickListener {
            printCatData() //call this other function
        }
    }

    // method to interact with API
    fun printCatData() {
        var catUrl = "https://api.thecatapi.com/v1/breeds" +
            "?api_key=USE_YOURS_YOU_INSUFFERABLE_CLOWN"
    }
}

```



```
val queue = Volley.newRequestQueue(this)

// Request a string response from the provided URL.
val stringRequest = StringRequest(
    Request.Method.GET, catUrl,
    Response.Listener<String> { response ->
        var catsArray : JSONArray = JSONArray(response)

        //indices from 0 through catsArray.length()-1
        for(i in 0 until catsArray.length()) {
            //${} is to interpolate the string /
            // uses a string template
            var theCat : JSONObject = catsArray.getJSONObject(i)

            //now get the properties we want: name and description
            Log.i("MainActivity", "Cat name: ${theCat.getString("name")}")
            Log.i("MainActivity", "Cat description:
${theCat.getString("description")}")
        }//end for
    },
    Response.ErrorListener {
        Log.i("MainActivity", "That didn't work!")
    })

// Add the request to the RequestQueue.
queue.add(stringRequest)
} //end printCatData
}
```