

Android Volley

Recitation COM S 309

Introduction

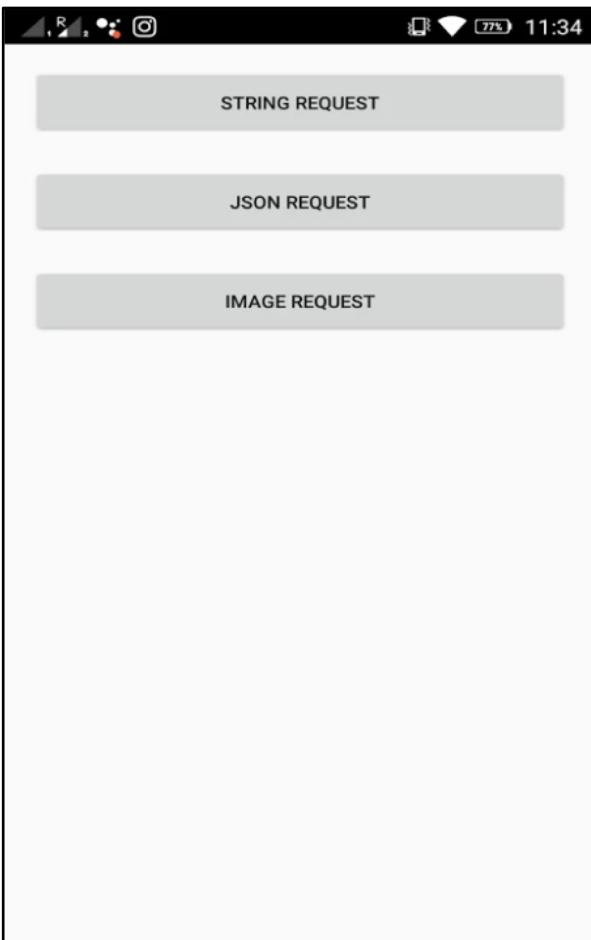
- Android volley is a **networking library** to make networking calls much easier
- Introduced by Google
- Download an Volley example project from
<https://git.linux.iastate.edu/cs309/Fall2018/ExampleProjects>
 - First, you can download and import this project in Android Studio and then go through the slides. Then create your own app using Volley and run.

Get Started

- We will create a sample app that sends the following requests to web API and show responses on Android activity.
 1. String request
 2. JSON request (JSON object and JSON array)
 - JSON is a text format to exchange data
 3. Image request
- The basic tasks to do that are:
 1. Create activities
 2. Import Volley library
 3. Make requests

Task 1: Create Activities

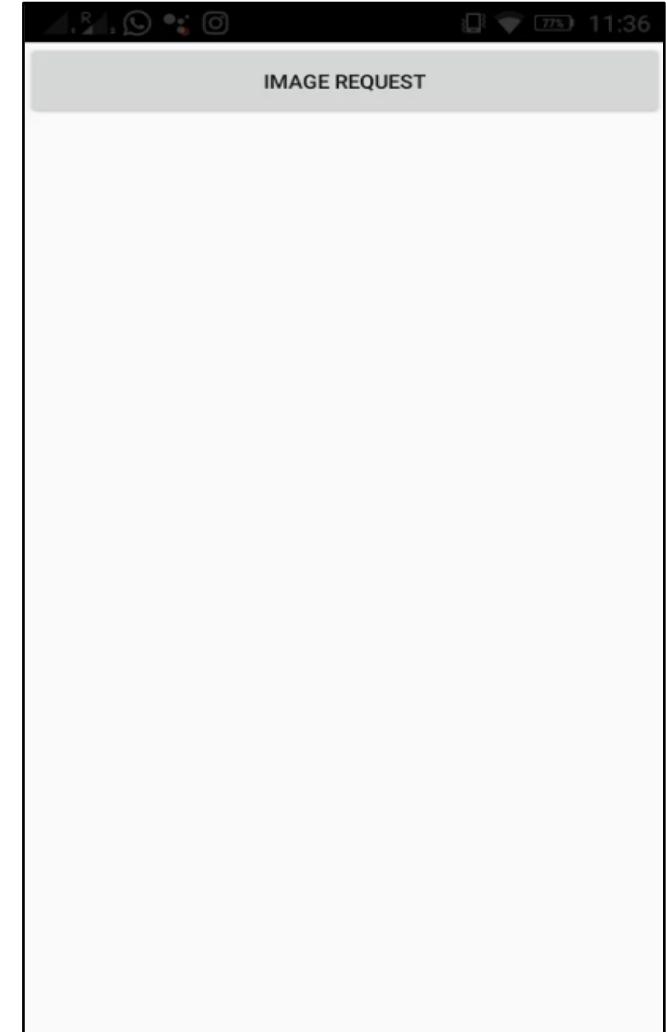
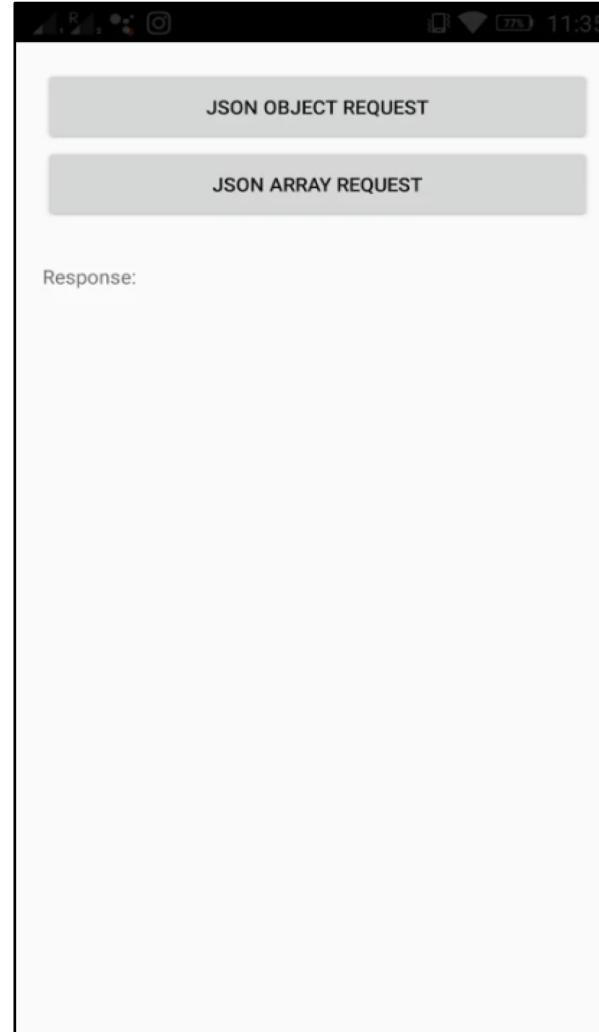
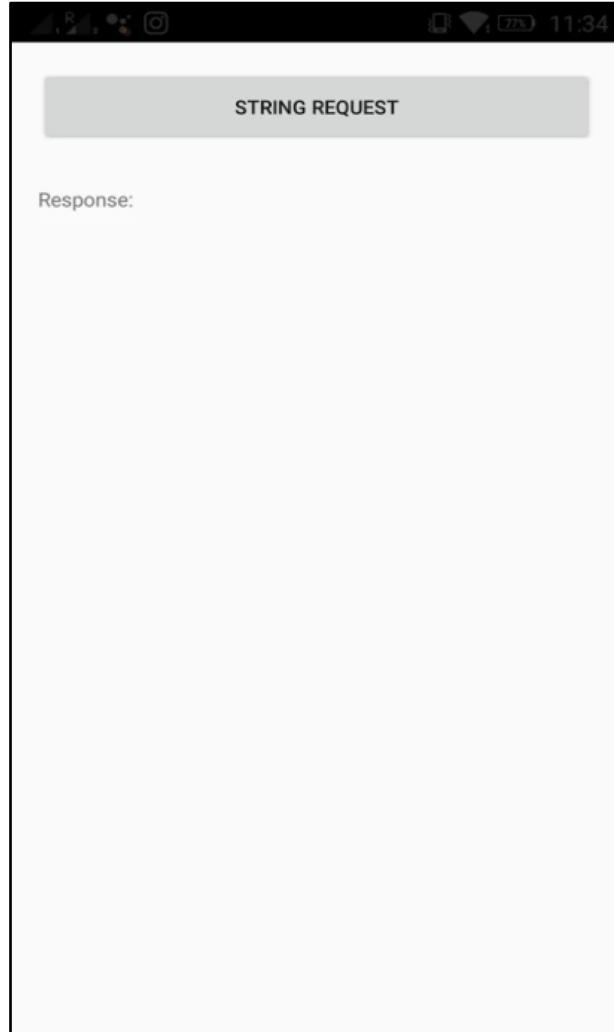
1. Create a new project.
2. Create UI with 3 buttons.



```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    android:paddingBottom="16dp"  
    android:paddingLeft="16dp"  
    android:paddingRight="1dp"  
    android:paddingTop="16dp"  
    tools:context=".MainActivity" >  
  
<Button android:id="@+id	btnStringRequest"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginBottom="20dp"  
        android:text="String Request" />  
  
<Button android:id="@+id	btnJsonRequest"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginBottom="20dp"  
        android:text="JSON Request" />  
  
<Button android:id="@+id	btnImageRequest"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginBottom="20dp"  
        android:text="Image Request" />  
  
</LinearLayout>
```

XML Layout for this activity

3. Create three more activities and link them to their respective buttons.



Task 2: Import Volley Library

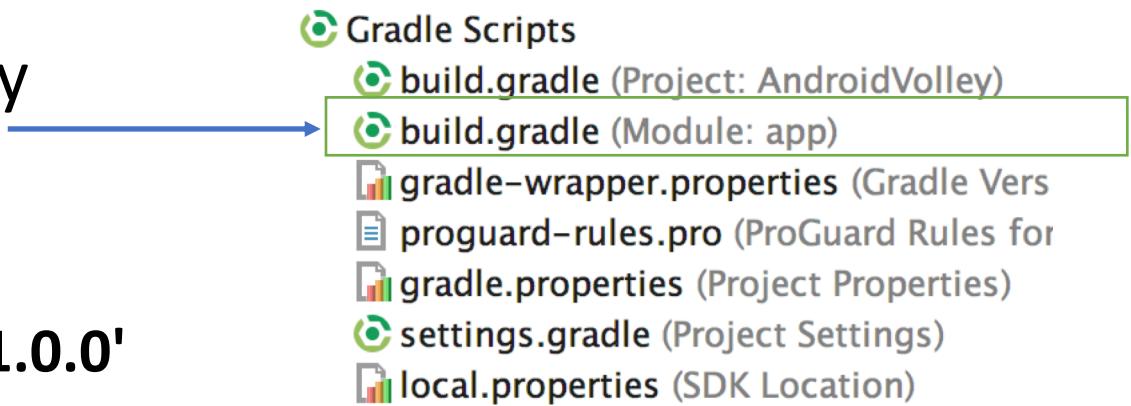
Three steps:

1. Add dependencies
2. Add controller classes
3. Add permission

STEP 1. Add Dependencies

- Open build.gradle and add the volley dependencies.
- Add this line under dependencies:
 - **compile 'com.android.volley:volley:1.0.0'**

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
        exclude group: 'com.android.support', module: 'support-annotations'  
    })  
    compile 'com.android.support:appcompat-v7:26.+'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    compile 'com.android.support:design:26.+'  
    testCompile 'junit:junit:4.12'  
  
    compile 'com.android.volley:volley:1.0.0'  
}
```



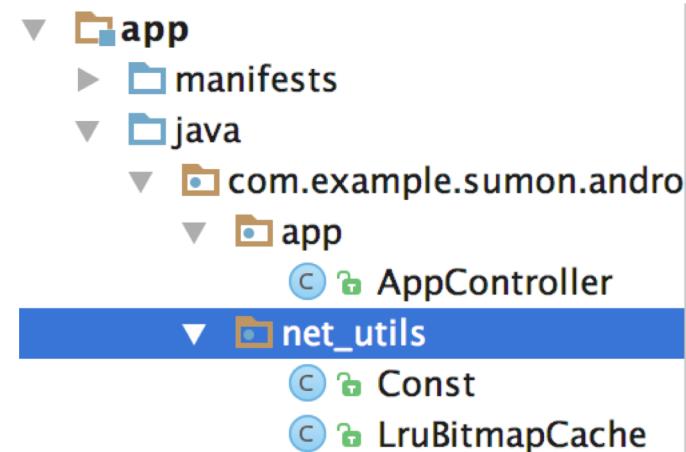
STEP 2. Add Controller Classes

First, create two packages *app* and *net_utils* to keep project organized.



Then, create these classes under the packages.

- i. Appcontroller.java
- ii. Const.java
- iii. LruBitmapCache.java



AppController.java

- This class is used to handle all the http requests that we make from the app.
- You don't have to change anything in the class.

```
import info.vamsikrishna.volleyeg.utils.LruBitmapCache;
import android.app.Application;
import android.text.TextUtils;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.ImageLoader;
import com.android.volley.toolbox.Volley;

public class AppController extends Application {

    public static final String TAG = AppController.class
            .getSimpleName();

    private RequestQueue mRequestQueue;
    private ImageLoader mImageLoader;

    private static AppController mInstance;

    @Override
    public void onCreate() {
        super.onCreate();
        mInstance = this;
    }

    public static synchronized AppController getInstance() {
        return mInstance;
    }

    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            mRequestQueue = Volley.newRequestQueue(getApplicationContext());
        }

        return mRequestQueue;
    }

    public ImageLoader getImageLoader() {
        getRequestQueue();
        if (mImageLoader == null) {
            mImageLoader = new ImageLoader(this.mRequestQueue,
                    new LruBitmapCache());
        }
        return this.mImageLoader;
    }

    public <T> void addToRequestQueue(Request<T> req, String tag) {
        // set the default tag if tag is empty
        req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
        getRequestQueue().add(req);
    }

    public <T> void addToRequestQueue(Request<T> req) {
        req.setTag(TAG);
        getRequestQueue().add(req);
    }

    public void cancelPendingRequests(Object tag) {
        if (mRequestQueue != null) {
            mRequestQueue.cancelAll(tag);
        }
    }
}
```

Const.java

```
package com.example.sumon.androidvolley.utils;

public class Const {
    public static final String URL_JSON_OBJECT = "https://api.androidhive.info/volley/person_object.json";
    public static final String URL_JSON_ARRAY = "https://api.androidhive.info/volley/person_array.json";
    public static final String URL_STRING_REQ = "https://api.androidhive.info/volley/string_response.html";
    public static final String URL_IMAGE = "https://api.androidhive.info/volley/volley-image.jpg";
}
```

In this class, we store the URLs for network calls.

Visit the URLs to look at the data present in that URL.

LruBitmapCache.java

This class is required to handle image cache. The images are in general stored using bitmaps.

```
import com.android.volley.toolbox.ImageLoader.ImageCache;
import android.graphics.Bitmap;
import android.support.v4.util.LruCache;

public class LruBitmapCache extends LruCache<String, Bitmap>
implements
    ImageCache {
    public static int getDefaultLruCacheSize() {
        final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
        final int cacheSize = maxMemory / 8;
        return cacheSize;
    }

    public LruBitmapCache() {
        this(getDefaultLruCacheSize());
    }

    public LruBitmapCache(int sizeInKiloBytes) {
        super(sizeInKiloBytes);
    }

    @Override
    protected int sizeOf(String key, Bitmap value) {
        return value.getRowBytes() * value.getHeight() / 1024;
    }

    @Override
    public Bitmap getBitmap(String url) {
        return get(url);
    }

    @Override
    public void putBitmap(String url, Bitmap bitmap) {
        put(url, bitmap);
    }
}
```

STEP 3. Add Permission in AndroidManifest.xml

- Open AndroidManifest.xml and add this line in <application> tag using android:name property to execute the class automatically whenever app launches.
- Also add INTERNET permission as we are going to make network calls.

```
manifest application
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.sumon.androidvolley">
4
5      <uses-permission android:name="android.permission.INTERNET" />
6
7      <application
8          android:name="com.example.sumon.androidvolley.app.AppController"
9
10         android:allowBackup="true"
11         android:icon="@mipmap/ic_launcher"
12         android:label="Android Volley"
13         android:roundIcon="@mipmap/ic_launcher_round"
14         android:supportsRtl="true"
15         android:theme="@style/AppTheme">
16             <activity
17                 android:name=".MainActivity"
18                 android:label="Android Volley"
19                 android:theme="@style/AppTheme.NoActionBar">
20                 <intent-filter>
21                     <action android:name="android.intent.action.MAIN" />
22
23                     <category android:name="android.intent.category.LAUNCHER"
24                 </intent-filter>
25             </activity>
26             <activity android:name=".ImageRequestActivity" />
27             <activity android:name=".JsonRequestActivity" />
28             <activity android:name=".StringRequestActivity"></activity>
29         </application>
30
31     </manifest>
```

Task 3: Send Request to Server

Create Three Classes

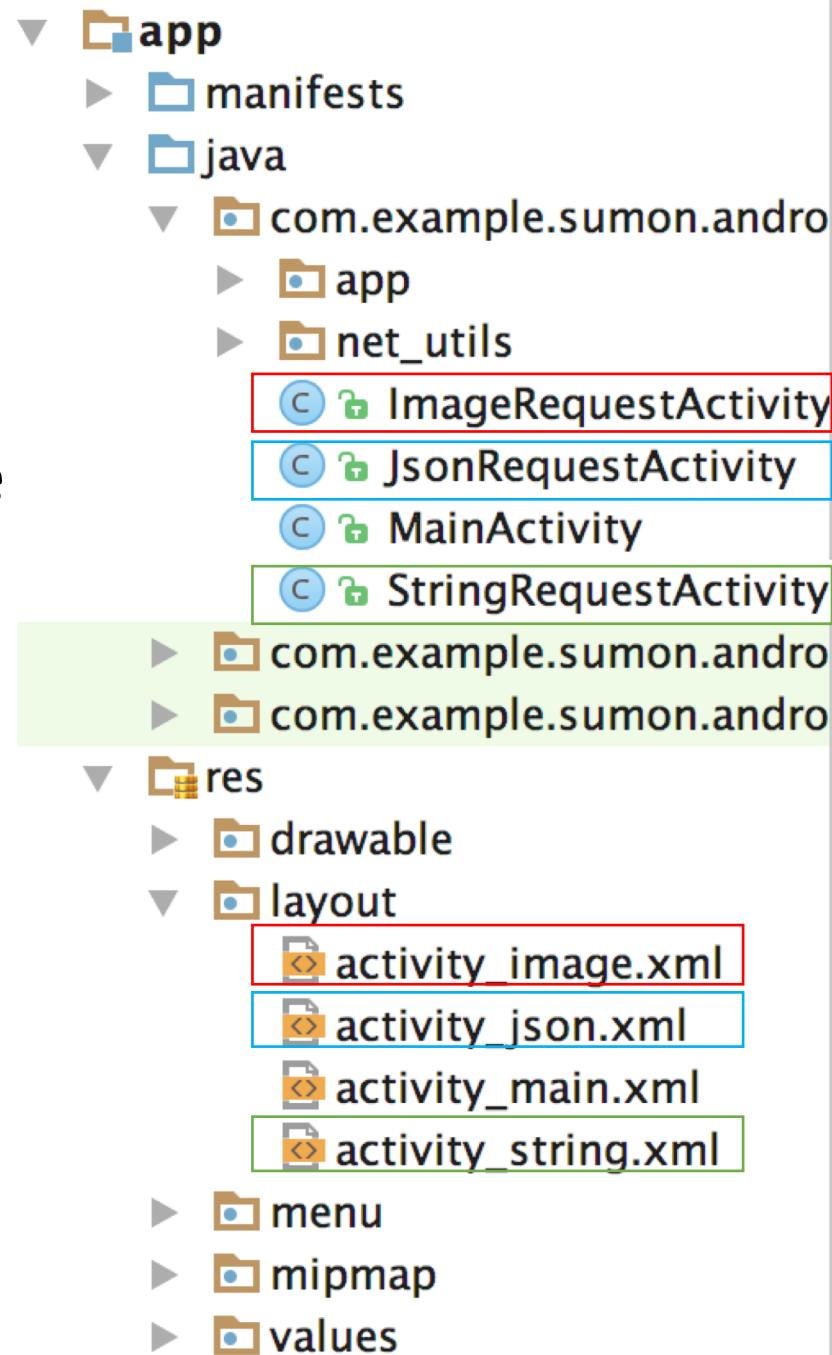
Create three classes for three different requests.
These classes are linked to the respective views we already created.

Step 1. String Request

Step 2. JSON Request

Step 3. Image Request

*The color correspond to the same activity



Step 1. String request: StringRequestActivity.java

- StringRequest class from Volley library will be used to fetch any kind of string data.
- Manipulate your response [response.toString()] here.
- Adding request to the queue



```
// Tag used to cancel the request  
String tag_string_req = "string_req";  
  
String url ="https://api.androidhive.info/volley/string_response.html";  
  
ProgressDialog pDialog =newProgressDialog(this);  
pDialog.setMessage("Loading...");  
pDialog.show();  
  
StringRequest strReq =newStringRequest(Method.GET,  
url,newResponse.Listener<String>() {  
  
    @Override  
    public void onResponse(String response) {  
        Log.d(TAG, response.toString());  
        pDialog.hide();  
    }  
},newResponse.ErrorListener() {  
  
    @Override  
    public void onErrorResponse(VolleyError error) {  
        VolleyLog.d(TAG,"Error: "+ error.getMessage());  
        pDialog.hide();  
    }  
});  
  
AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
```

Step 2. JSON Request

We will do

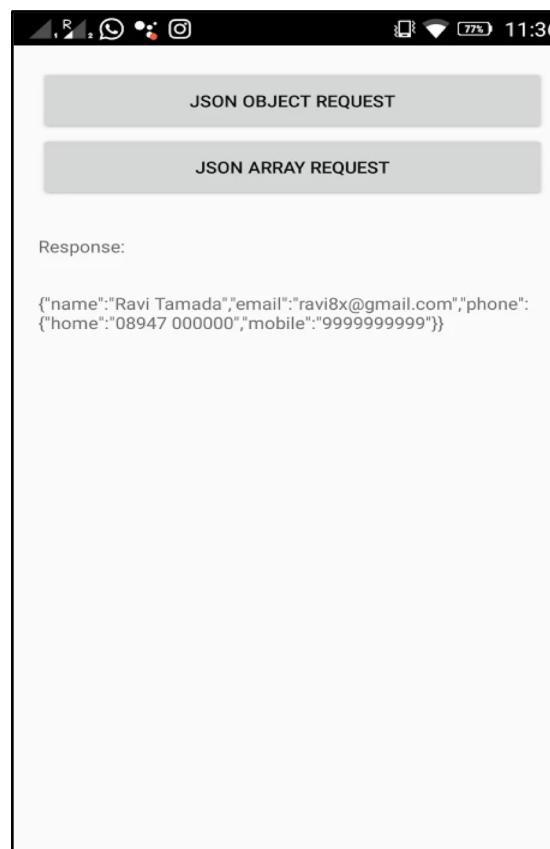
- a) JSON Object Request using GET Method
- b) JSON Array request
- c) JSON Object Request using POST Method

GET Method: Requests data from a specified resource

POST Method: Submits data to be processed to a specified resource

2(a) JSON object request using GET method: JsonRequestActivity.java

- Request creation
- json response will start with object notation '{'
- Manipulate your response
- Adding request to the queue



```
// Tag used to cancel the request
String tag_json_obj = "json_obj_req";

String url ="https://api.androidhive.info/volley/person\_object.json";

ProgressDialog pDialog =newProgressDialog(this);
pDialog.setMessage("Loading...");
pDialog.show();

JsonObjectRequest jsonObjReq =newJsonObjectRequest(Method.GET,
    url,null,
    newResponse.Listener<JSONObject>() {

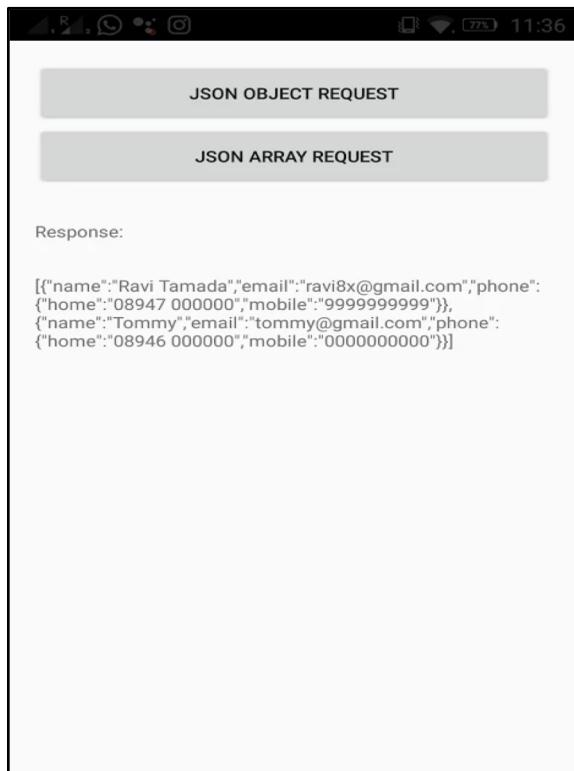
        @Override
        public void onResponse(JSONObject response) {
            Log.d(TAG, response.toString());
            pDialog.hide();
        }
    },newResponse.ErrorListener() {

        @Override
        public void onErrorResponse(VolleyError error) {
            VolleyLog.d(TAG,"Error: "+ error.getMessage());
            // hide the progress dialog
            pDialog.hide();
        }
    });

AppController.getInstance().addToRequestQueue(jsonObjReq, tag_json_obj);
```

2(b) JSON array request: JsonRequestActivity.java

- Request creation
- Adding request to request queue
- You can parse this json data to get values into array



```
// Tag used to cancel the request
String tag_json_array = "json_array_req";

String url = "https://api.androidhive.info/volley/person\_array.json";

ProgressDialog pDialog = new ProgressDialog(this);
pDialog.setMessage("Loading...");
pDialog.show();

JsonArrayRequest req = new JsonArrayRequest(url,
    new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray response) {
            Log.d(TAG, response.toString());
            pDialog.hide();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            VolleyLog.d(TAG, "Error: " + error.getMessage());
            pDialog.hide();
        }
});

AppController.getInstance().addToRequestQueue(req, tag_json_array);
```

2(c) JSON object request using POST method: JsonRequestActivity.java

- This is similar but using POST Method.
- submitting name, email and password as request parameters.
- Your web service (URL) should be able to capture the request and send appropriate response.

```
String tag_json_obj = "json_obj_req";
String url ="https://api.androidhive.info/volley/person\_object.json";

ProgressDialog pDialog =newProgressDialog(this);
pDialog.setMessage("Loading...");
pDialog.show();

JsonObjectRequest jsonObjReq =newJsonObjectRequest(Method.POST,
url,null,
newResponse.Listener<JSONObject>() {

    @Override
    public void onResponse(JSONObject response) {
        Log.d(TAG, response.toString());
        pDialog.hide();
    }
},newResponse.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        VolleyLog.d(TAG,"Error: "+ error.getMessage());
        pDialog.hide();
    }
}) {

    @Override
    protected Map<String, String> getParams() {
        Map<String, String>params =newHashMap<String, String>();
        params.put("name","Androidhive");
        params.put("email","abc@androidhive.info");
        params.put("password","password123");

        return params;
    }

};

AppController.getInstance().addToRequestQueue(jsonObjReq, tag_json_obj);
```

3. Image Request

- We can use two different views in our activity to show image
 - NetworkImageView
 - ImageView
- This will load an image from an URL into **NetworkImageView**.
- load image into **ImageView** instead of **NetworkImageView** with success and error callbacks.
- Handle response.
- We can display default loading image or failure image.
 - **Loader**: The loader image will be displayed until the image gets downloaded.
 - **Error**: If the image fails to download, the error image will be displayed.
- Place these icon images in res > drawable



```
ImageLoader imageLoader = AppController.getInstance().getImageLoader();  
// If you are using NetworkImageView  
imgNetworkView.setImageUrl(Const.URL_IMAGE, imageLoader);  
  
ImageLoader imageLoader = AppController.getInstance().getImageLoader();  
// If you are using normal ImageView  
imageLoader.get(Const.URL_IMAGE, newImageListener() {  
  
    @Override  
    public void onErrorResponse(VolleyError error) {  
        Log.e(TAG, "Image Load Error: " + error.getMessage());  
    }  
  
    @Override  
    public void onResponse(ImageContainer response, boolean arg1) {  
        if(response.getBitmap() != null) {  
            // load image into imageview  
            imageView.setImageBitmap(response.getBitmap());  
        }  
    }  
});  
  
// Loading image with placeholder and error image  
imageLoader.get(Const.URL_IMAGE, ImageLoader.getImageListener(  
    imageView, R.drawable.ico_loading, R.drawable.ico_error));
```



Cancelling Requests

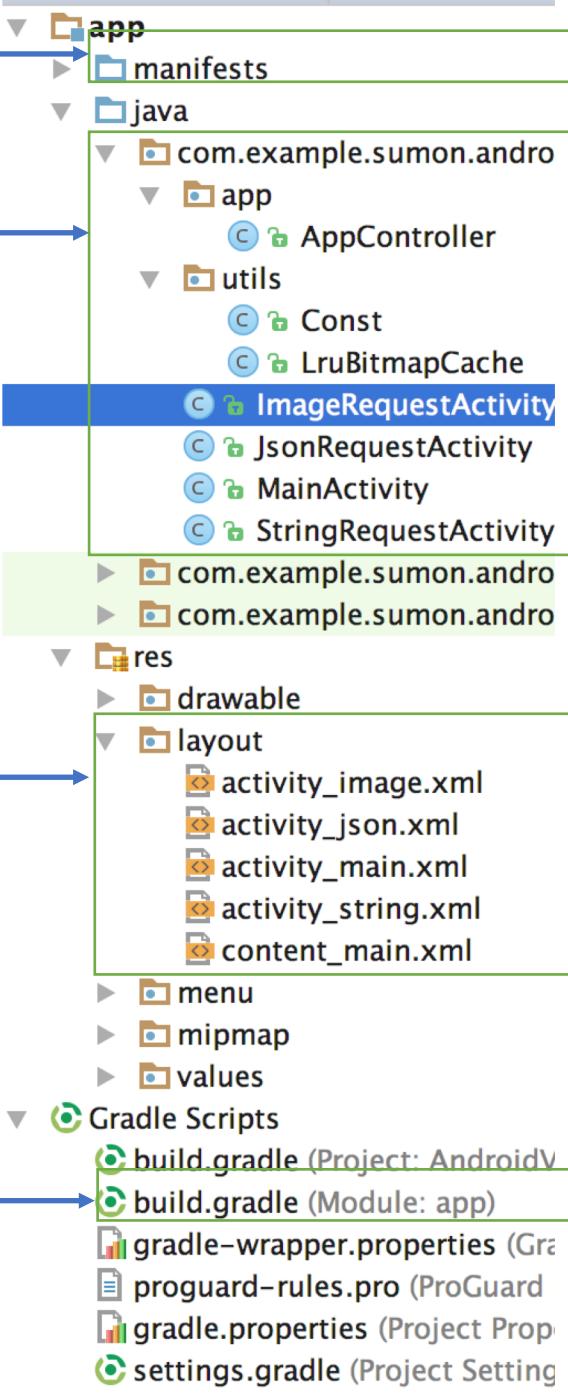
- You might want to cancel network requests. You can do it by this line

```
ApplicationController.getInstance().getRequestQueue().cancelAll("feed_request");
```

Tag that was used
while adding request

- If the tag is same for multiple requests, all the requests with that tag will be cancelled.
- You can also use **cancelAll()** method to cancel all the the requests.

```
ApplicationController.getInstance().getRequestQueue().cancelAll();
```



Code

- The files we worked on:
 - Manifest
 - Activities/Java files
 - Views
 - Gradle
- Pull code from following GIT repository.
 - <https://git.linux.iastate.edu/cs309/Fall2018/ExampleProjects>
- You can learn more on Volley:
<https://developer.android.com/training/volley/index.html>