Android Volley

Recitation COM S 309

Introduction

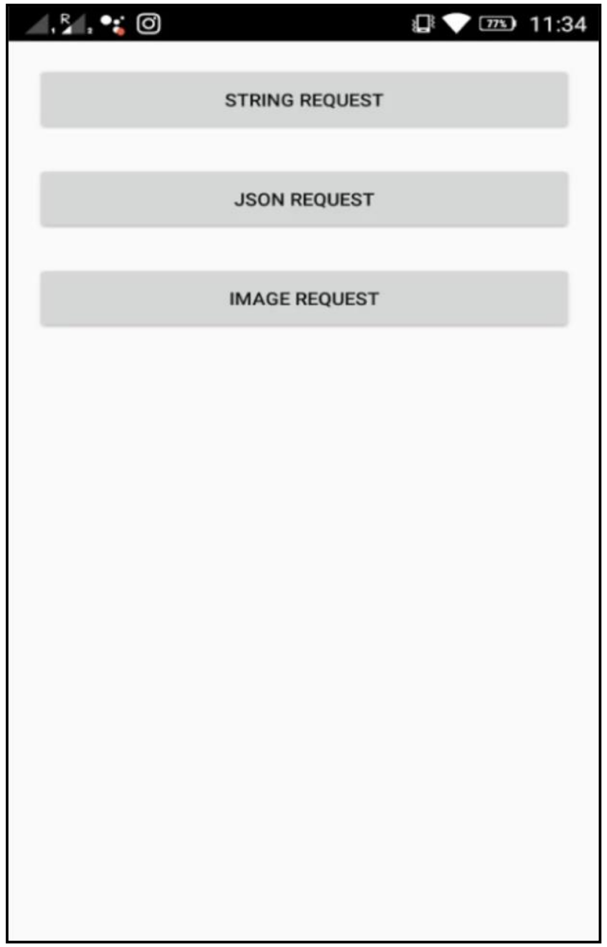
- Android volley is a networking library to make networking calls much easier and is developed by Google
- Download a Volley example project from:
  https://git.linux.iastate.edu/cs309/Fall2018/ExampleProjects
  OR
  https://git.linux.iastate.edu/cs309/tutorials_android
  First, you can download and import this project in Android Studio and then go through the slides. Then create your own app using Volley and run.
  - **READ THE THINGS TO REMEMBER SECTION AT THE END!!!**

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

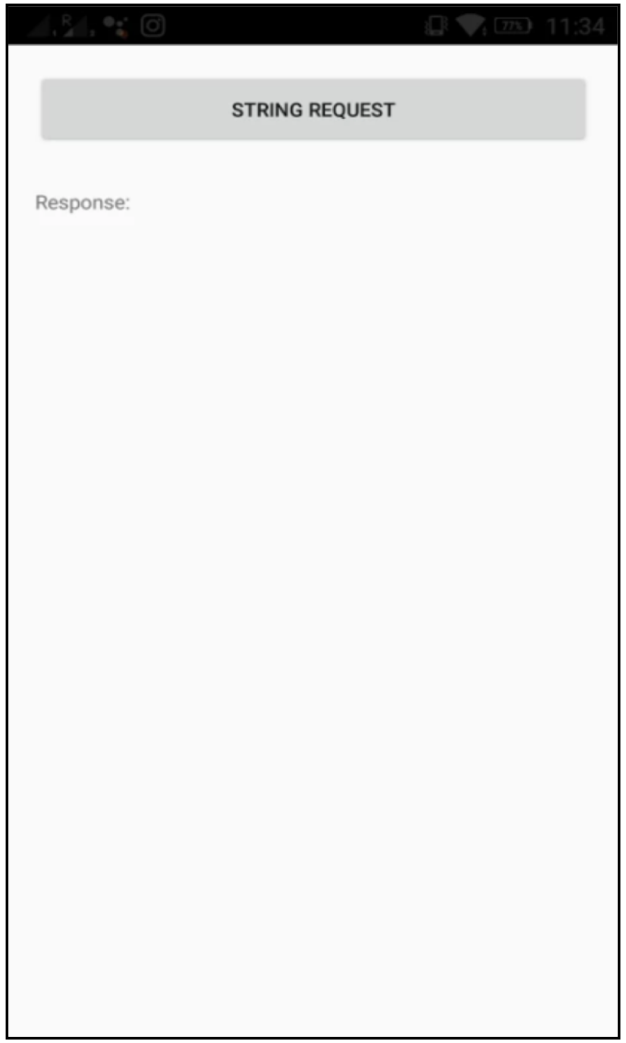
1. Create a new project.
2. Create UI with 3 buttons.

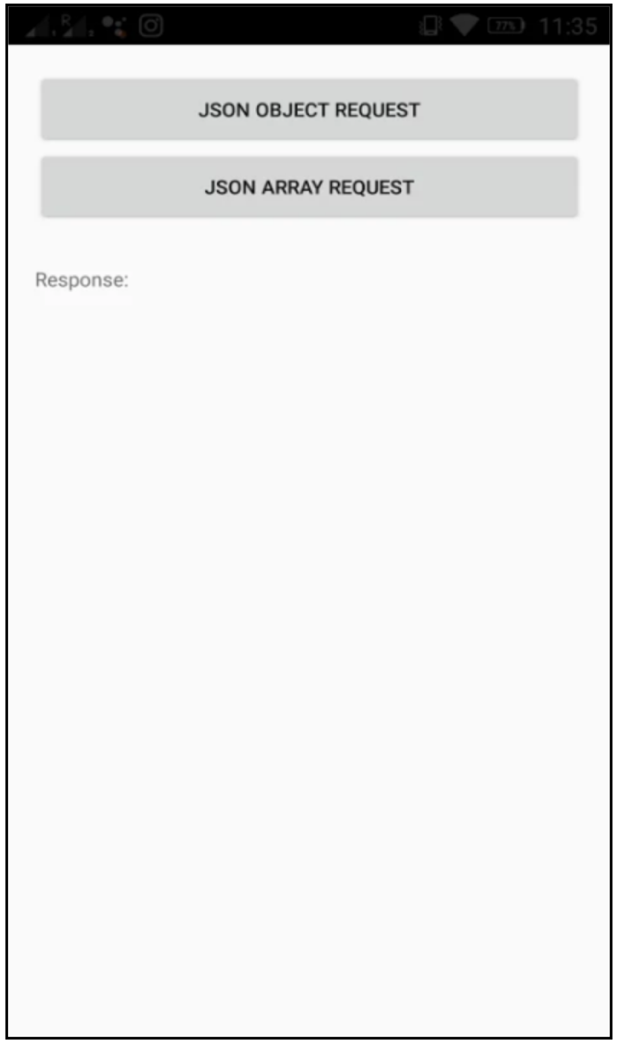


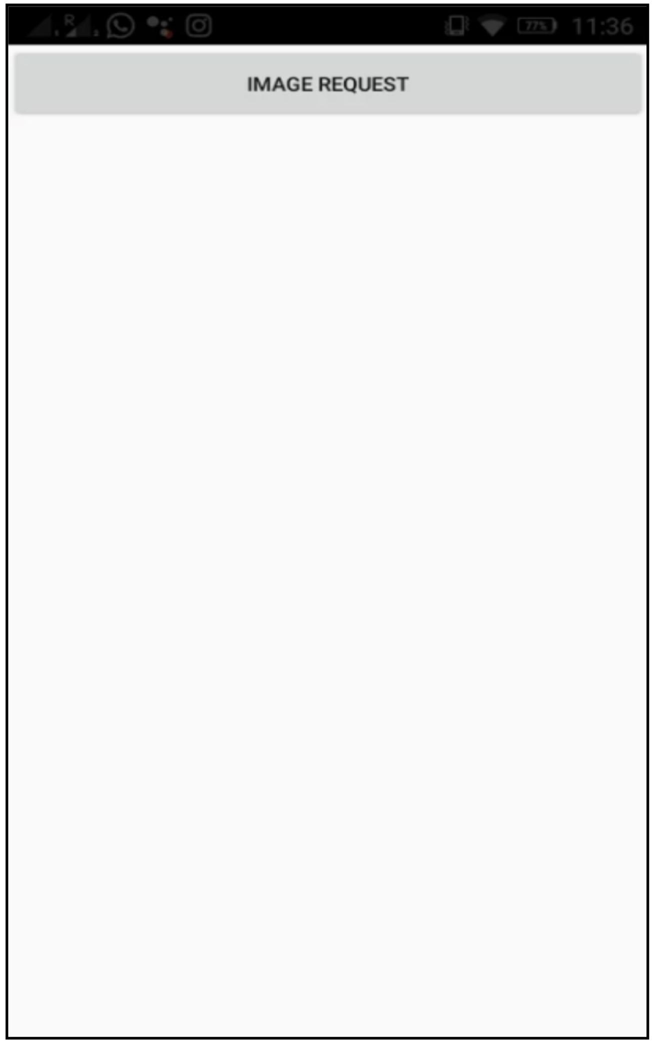
```xml
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:paddingBottom="16dp"
   android:paddingLeft="16dp"
   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

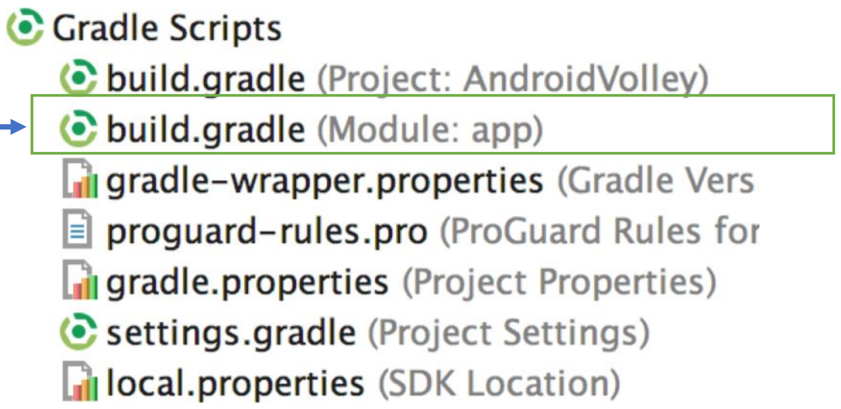
Three steps:

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

STEP 1. Add Dependencies

- Open build.gradle and add the volley dependency.
- Add this line under dependencies:
  - implementation **'com.android.volley:volley:1.1.1'**

```
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'
   })
   implementation 'com.android.support:appcompat-v7:26.+'
   implementation 'com.android.support.constraint:constraint-layout:1.0.2'
   implementation 'com.android.support:design:26.+'
   testImplementation 'junit:junit:4.12'

   implementation 'com.android.volley:volley:1.1.1'
}
```



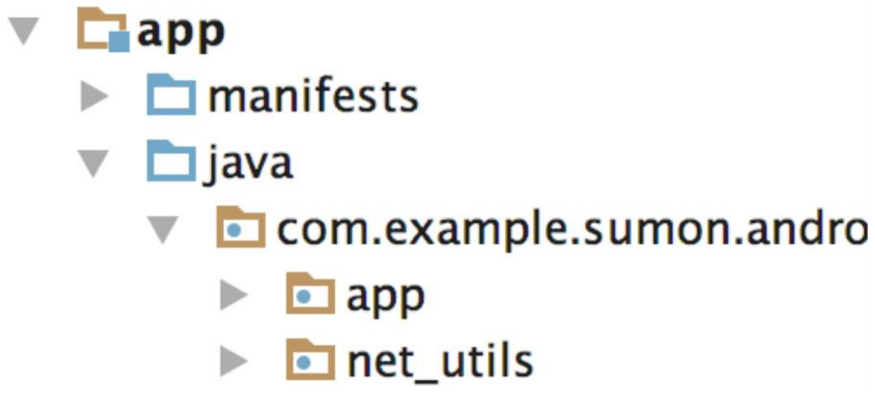
```
Gradle Scripts
   build.gradle (Project: AndroidVolley)
   build.gradle (Module: app)
   gradle-wrapper.properties (Gradle Vers
   proguard-rules.pro (ProGuard Rules for
   gradle.properties (Project Properties)
   settings.gradle (Project Settings)
   local.properties (SDK Location)
```

STEP 2. Add Controller Classes

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

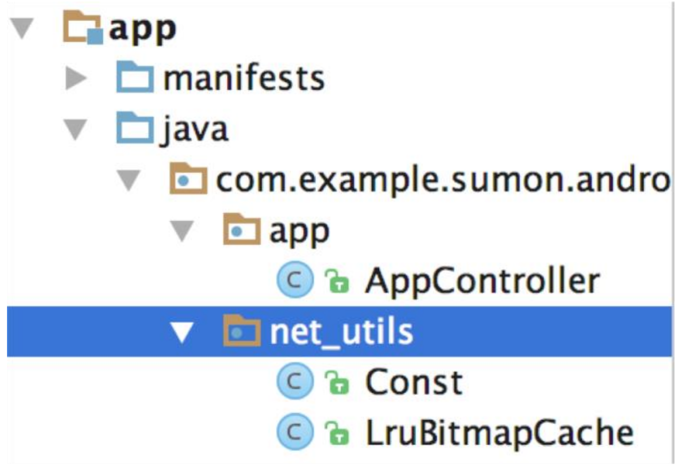
**This is one way to implement volley requests,*

Look at things to remember section!



Then, create these classes under the packages.

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



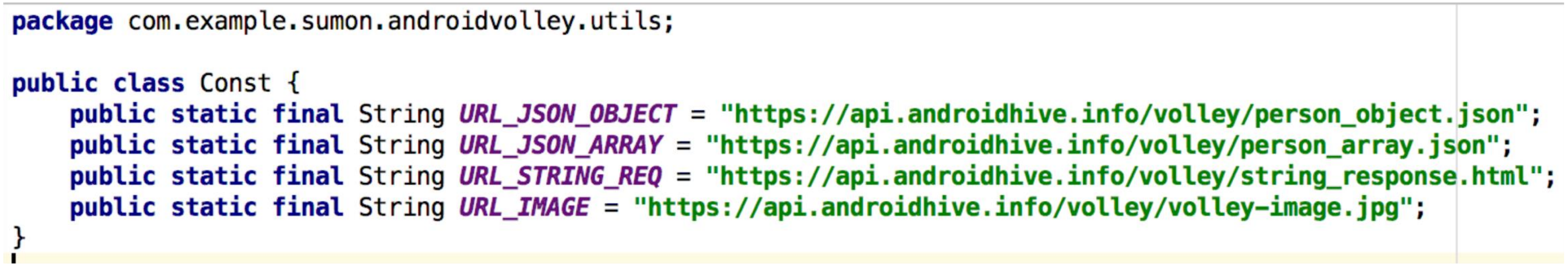
AppController.java

- 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.

```java
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.

When trying to connect to localhost through your phone/emulator, you need to use http://10.0.2.2 not localhost.

LruBitmapCache.java

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

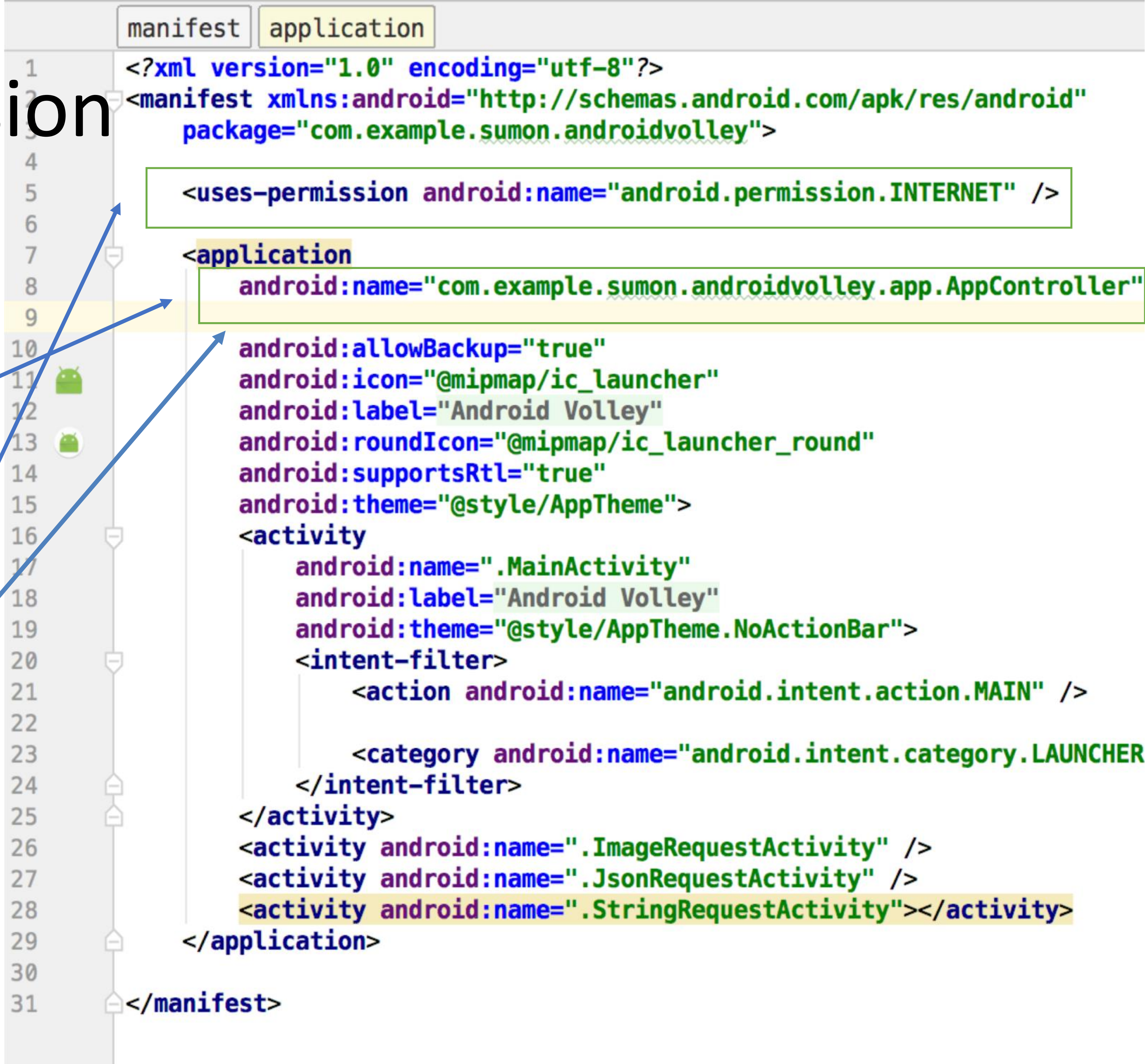
```
import com.android.volley.toolbox.ImageLoader.ImageCache; import
android.graphics.Bitmap;
import android.support.v4.util.LruCache;

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.
- **Also add android:usesCleartextTraffic="true" to the application tag**

- **These two tags are required for volley to work!!!**



manifest application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sumon.androidvolley">

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

    <application
        android:name="com.example.sumon.androidvolley.app.AppController"

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Android Volley"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="Android Volley"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER
            </intent-filter>
        </activity>
        <activity android:name=".ImageRequestActivity" />
        <activity android:name=".JsonRequestActivity" />
        <activity android:name=".StringRequestActivity"></activity>
    </application>

</manifest>
```

Task 3: Send Request to Server

Create Three Classes

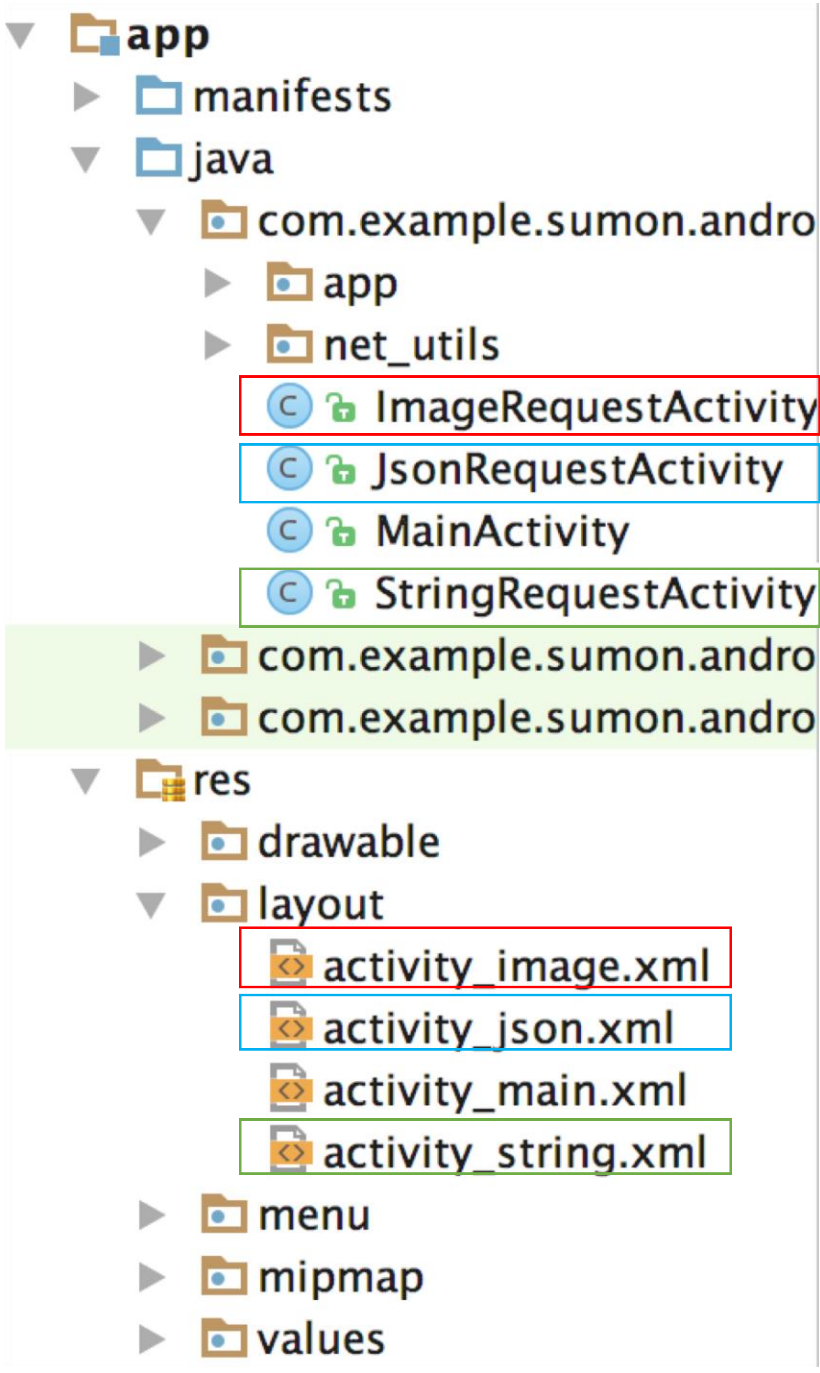
Create three classes for three different requests. These classes are linked to the respective views we already created.

Step 1. String Request

Step 2. JSON Request

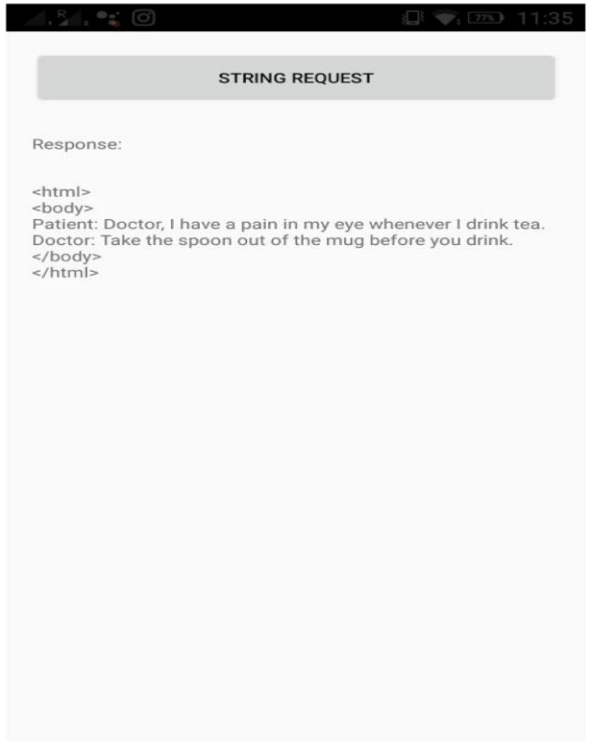
Step 3. Image Request

*The color corresponds 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



```
<html>
<body>
Patient: Doctor, I have a pain in my eye whenever I drink tea.
Doctor: Take the spoon out of the mug before you drink.
</body>
</html>
```

```
// 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
                    publicvoidonResponse(String response) {
                        Log.d(TAG, response.toString());
                        pDialog.hide();

                    }
                },newResponse.ErrorListener() {

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

```
AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
```

Step 2. JSON Request

We will do

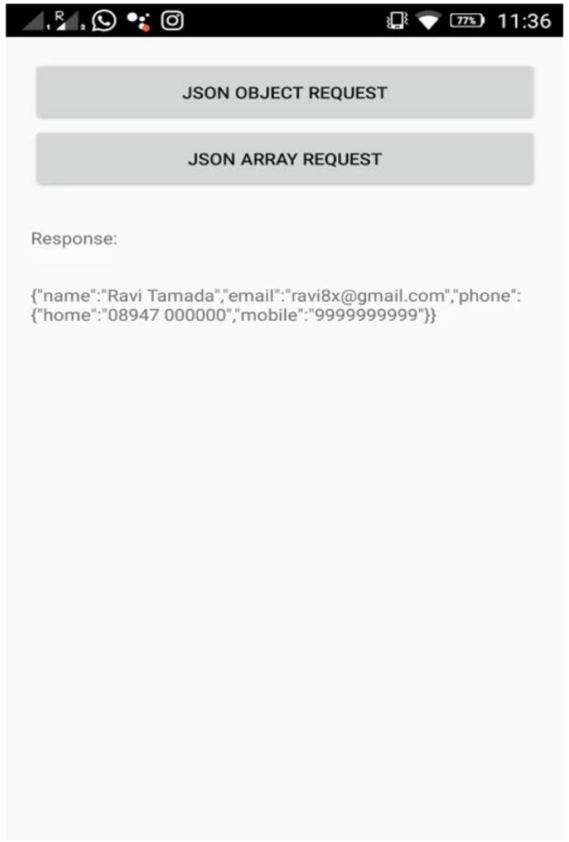
a) JSON Object Request using GET Method
b) JSON Array request
c) JSON Object Request using POST Method

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

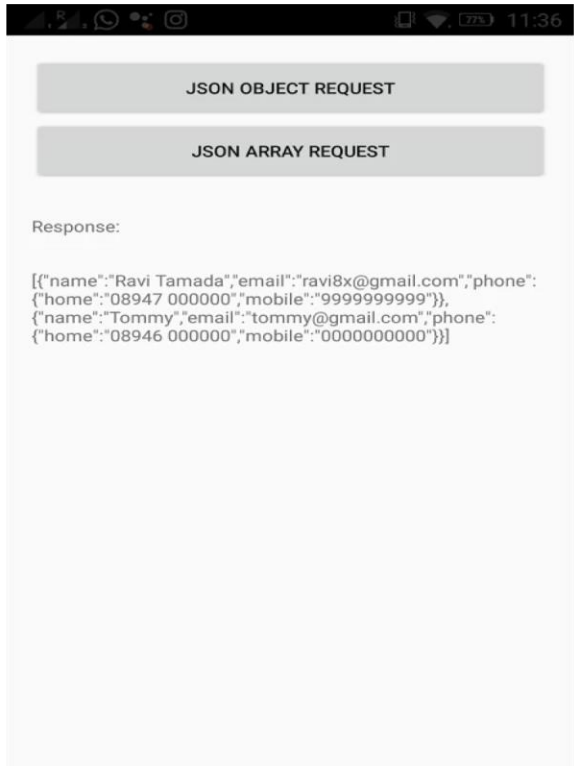


```
{"name":"Ravi Tamada","email":"ravi8x@gmail.com","phone":
{"home":"08947 000000","mobile":"9999999999"}}
```

```
// 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
                    publicvoidonResponse(JSONObject response) {
                        Log.d(TAG, response.toString());
                        pDialog.hide();
                    }
                },newResponse.ErrorListener() {

                    @Override
                    publicvoidonErrorResponse(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



Response:

```
[{"name":"Ravi Tamada","email":"ravi8x@gmail.com","phone":
{"home":"08947 000000","mobile":"9999999999"}},
{"name":"Tommy","email":"tommy@gmail.com","phone":
{"home":"08946 000000","mobile":"0000000000"}}]
```

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

String url ="https://api.androidhive.info/volley/person_array.json";

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

JsonArrayRequest req =newJsonArrayRequest(url,
                newResponse.Listener<JSONArray>() {
                    @Override
                    publicvoidonResponse(JSONArray response) {
                        Log.d(TAG, response.toString());
                        pDialog.hide();
                    }
                },newResponse.ErrorListener() {
                    @Override
                    publicvoidonErrorResponse(VolleyError error) {
                        VolleyLog.d(TAG,"Error: "+ error.getMessage());
                        pDialog.hide();
                    }
                });

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

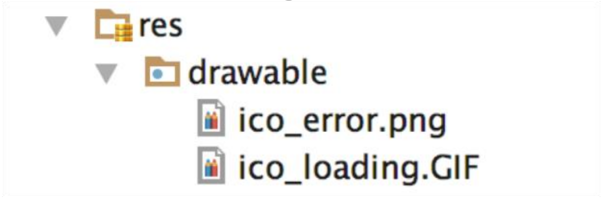
2(c) JSON object request using POST method: JsonRequestActivity.java

- 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
                    publicvoidonResponse(JSONObject response) {
                        Log.d(TAG, response.toString());
                        pDialog.hide();
                    }
                },newResponse.ErrorListener() {

                    @Override
                    publicvoidonErrorResponse(VolleyError error) {
                        VolleyLog.d(TAG,"Error: "+ error.getMessage());


                        pDialog.hide();
                    }
                }) {

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

                returnparams;
            }

        };
 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


```
res
  drawable
    ico_error.png
    ico_loading.GIF
```

```
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
    publicvoidonErrorResponse(VolleyError error) {
        Log.e(TAG,"Image Load Error: "+ error.getMessage());
    }

    @Override
    publicvoidonResponse(ImageContainer response,booleanarg1) {
        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");
```

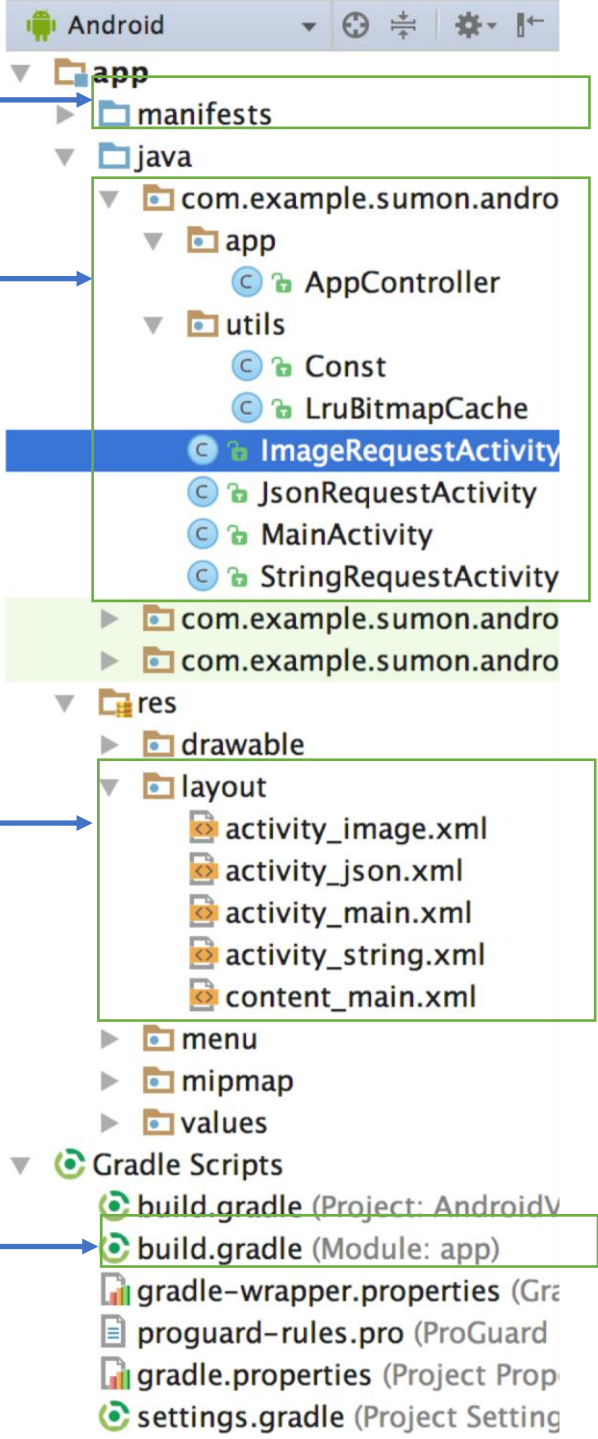
$\xrightarrow{}$ Tag that was used while adding request

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

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

Code

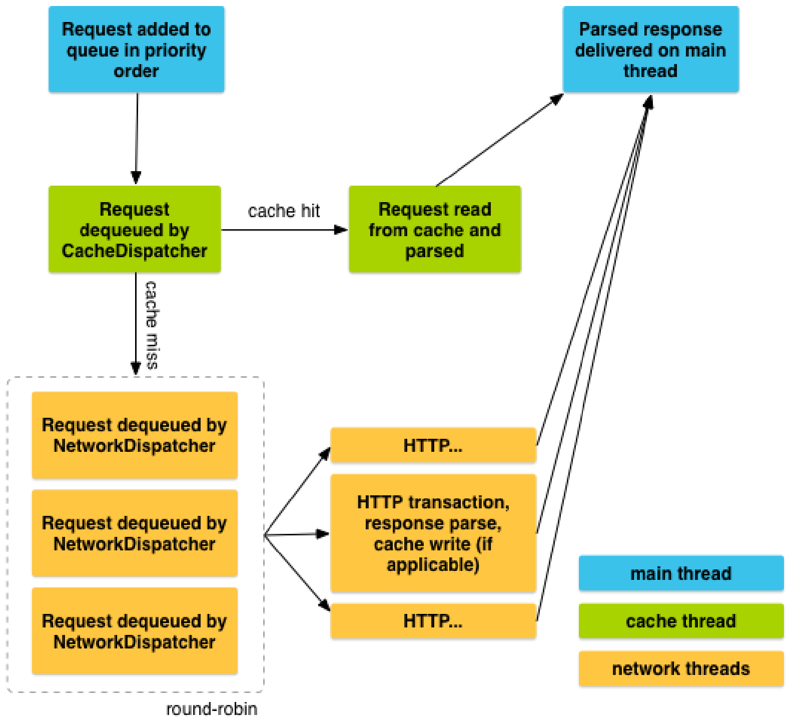
- The files we worked on:
  - Manifest
  - Activities/Java files
  - Views
  - Gradle
- Pull code from following GIT repository.
  - https://git.linux.iastate.edu/cs309/Fall2018/ExampleProjects
- You can learn more about Volley: https://developer.android.com/training/volley/ind ex.html



```
Android
▼ app
  ▶ manifests
  ▼ java
    ▼ com.example.sumon.andro
      ▼ app
          AppController
      ▼ utils
          Const
          LruBitmapCache
        ImageRequestActivity
        JsonRequestActivity
        MainActivity
        StringRequestActivity
    ▶ com.example.sumon.andro
    ▶ com.example.sumon.andro
  ▼ res
    ▶ drawable
    ▼ layout
        activity_image.xml
        activity_json.xml
        activity_main.xml
        activity_string.xml
        content_main.xml
    ▶ menu
    ▶ mipmap
    ▶ values
▼ Gradle Scripts
    build.gradle (Project: AndroidV
    build.gradle (Module: app)
    gradle-wrapper.properties (Gra
    proguard-rules.pro (ProGuard
    gradle.properties (Project Prop
    settings.gradle (Project Setting
```

Things to remember

- Volley runs on the **main thread** but is ***asynchronous***. This means you cannot expect volley to execute your request at a certain point in time/line of code.
  - You **can** modify UI elements within response handlers because they run on the main thread!
  - Request lifecycle:


Request added to queue in priority order
Parsed response delivered on main thread
Request dequeued by CacheDispatcher
cache hit
Request read from cache and parsed
cache miss
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
HTTP...
HTTP transaction, response parse, cache write (if applicable)
HTTP...
main thread
cache thread
network threads
round-robin

- Make sure you are using the correct version of volley!
  - If you are using volley 1.1+, the import statement might need to be changed:
    - import com.android.volley.toolbox.(…);
  - If this isn’t working automatically, resync gradle then restart Android Studio.
- The design provided in the code above is **NOT** the only way to implement Volley Requests. You may not need a separate AppContoller, Constants, etc. **Use what you need for your own project**!
  - For example, you can implement a request and queue in one activity without using AppController.
  - The solution above allows you to reuse code and makes use of an application wide request queue.