Android Upload Image to Server using Volley Tutorial

October 21, 2015 by Belal Khan — 8 Comments

Uploading images to our server is a very frequently used thing. In most of the apps, we need user avatar, i.e. user pro le image. So here is Android Upload Image to Server Tutorial.
In this post, we will see how we can upload images from our Android app to our server. Then we will also see how do we fetch the uploaded images back to our android application. So let's begin.

Contents [hide]

Tools Required

- Xampp/Wamp as for the server end I am going to use PHP and MySQL.
- Android Studio, it is obvious that we need it for building our application

Creating Database and RESTful API

Database

Here, we will upload and store the image file to our server's directory, and in the database, we will save the path to the image with some tags for that picture. So the first thing here we need is the database.

- Go to **localhost/phpmyadmin** and create the following table names images in your database.

Database Table

- To create the above-given table, you can use the following SQL Query.

```
CREATE TABLE `images` (
  `id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  `image` varchar(500) NOT NULL,
  `tags` varchar(500) NOT NULL
)
```

- Once you have the database table, you can start building the API.

API

- Now create a folder inside **c:/xampp/htdocs (the root directory if you are using xampp),** you can name this folder anything. I have created a folder named **MyApi**.
- Inside the folder create a file named **api.php** and write the following php code.

```php
//Constants for database connection
define('DB_HOST','localhost');
define('DB_USER','root');
define('DB_PASS','password');
define('DB_NAME','simplifiedcoding');

//We will upload files to this folder
//So one thing don't forget, also create a folder named uploads inside your project folder i.e.
define('UPLOAD_PATH', 'uploads/');

//connecting to database
$conn = new mysqli(DB_HOST,DB_USER,DB_PASS,DB_NAME) or die('Unable to connect');


//An array to display the response
$response = array();

//if the call is an api call
if(isset($_GET['apicall'])){

//switching the api call
switch($_GET['apicall']){

//if it is an upload call we will upload the image
case 'uploadpic':

//first confirming that we have the image and tags in the request parameter
if(isset($_FILES['pic']['name']) && isset($_POST['tags'])){

//uploading file and storing it to database as well
try{
move_uploaded_file($_FILES['pic']['tmp_name'], UPLOAD_PATH . $_FILES['pic']['name']);
$stmt = $conn->prepare("INSERT INTO images (image, tags) VALUES (?,?)");
$stmt->bind_param("ss", $_FILES['pic']['name'],$_POST['tags']);
if($stmt->execute()){
$response['error'] = false;
$response['message'] = 'File uploaded successfully';
}else{
```

Android Upload Image to Server Using PHP MySQL

Android TabLayout Example using ViewPager and Fragments

Firebase Cloud Messaging Tutorial for Android

Android Push Noti cation using GCM Tutorial

Android Volley Tutorial – User Registration and Login

Retrieve Data From MySQL Database in Android using Volley

```
$response['error'] = true;
$response['message'] = 'Could not upload file';
}

}else{
$response['error'] = true;
$response['message'] = "Required params not available";
}

break;

//in this call we will fetch all the images
case 'getpics':

//getting server ip for building image url
$server_ip = gethostbyname(gethostname());

//query to get images from database
$stmt = $conn->prepare("SELECT id, image, tags FROM images");
$stmt->execute();
$stmt->bind_result($id, $image, $tags);

$images = array();

//fetching all the images from database
//and pushing it to array
while($stmt->fetch()){
$temp = array();
$temp['id'] = $id;
$temp['image'] = 'http://' . $server_ip . '/MyApi/'. UPLOAD_PATH . $image;
$temp['tags'] = $tags;

array_push($images, $temp);
}

//pushing the array in response
$response['error'] = false;
$response['images'] = $images;
break;
```

```
$response['message'] = 'Invalid api call';
}

}else{
header("HTTP/1.0 404 Not Found");
echo "<h1>404 Not Found</h1>";
echo "The page that you have requested could not be found.";
exit();
}

//displaying the response in json
header('Content-Type: application/json');
echo json_encode($response);
```

- Now we need to test the API.

Testing the API

- For testing our API I am here using POSTMAN, but you can use any tool you want.

Upload Image to Server API

- So, the upload call is perfect. Now let's test the fetch images call.

- It is perfect as well. Now we can jump on Android Side.

Android Upload Image to Server using Volley

Creating an Android Project

- Now, we will create a new Android Studio project with an Empty Activity.
- As we are going to use Volley rst, we will add it to the project.

Adding Volley

- Come inside dependencies block of app level build.gradle le and add volley here, as shown below.

```
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'

    //adding volley library
    compile 'com.android.volley:volley:1.1.0-rc1'
```

- Now sync your project.

Adding Permissions

- We also need Internet and Read Storage permission. So inside **AndroidManifest.xml** add these permissions.

AndroidManifest.xml

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

    <!-- adding permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

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

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

</manifest>
```

User Interface

- Now come inside **activity_main.xml** and put the following xml code.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="net.simplifiedlearning.androiduploadimage.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical">

        <EditText
            android:id="@+id/editTextTags"
            android:hint="Enter tags"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/buttonUploadImage"
            android:layout_width="wrap_content"
```

```
27
28          <ImageView
29              android:id="@+id/imageView"
30              android:layout_width="match_parent"
31              android:layout_height="wrap_content" />
32
33      </LinearLayout>
34
35  </RelativeLayout>
```

- The above code will generate the following layout.

Upload Image to Server Interface

- The layout is very simple, we have an EditText and a Button only.

De ning EndPoints

- We will create a separate class to store our URL Endpoints. **Remember using localhost will not work and you need to nd the ip of your xampp. You can nd the ip using ipcon g command in windows and ifcon g in linux/mac**.
- Create a class named **EndPoints** and write the following code here.

EndPoints.java Java

```
1  package net.simplifiedlearning.androiduploadimage;
2
3  /**
4   * Created by Belal on 10/24/2017.
5   */
6
7  public class EndPoints {
8      private static final String ROOT_URL = "http://192.168.101.1/MyApi/Api.php?apicall=";
9      public static final String UPLOAD_URL = ROOT_URL + "uploadpic";
10     public static final String GET_PICS_URL = ROOT_URL + "getpics";
11 }
```

request directly. So that is why we need our Custom Volley Request. I found this code on GitHub and made little modi cations. **I am thankful to the author for his contribution**.

- Create a class named **VolleyMultipartRequest** and write the following code.

VolleyMultipartRequest.java Java

```java
package net.simplifiedlearning.androiduploadimage;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkResponse;
import com.android.volley.ParseError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.HttpHeaderParser;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.Map;

/**
 * Created by Belal on 10/24/2017.
 */

public class VolleyMultipartRequest extends Request<NetworkResponse> {

    private final String twoHyphens = "--";
    private final String lineEnd = "\r\n";
    private final String boundary = "apiclient-" + System.currentTimeMillis();

    private Response.Listener<NetworkResponse> mListener;
    private Response.ErrorListener mErrorListener;
    private Map<String, String> mHeaders;


    public VolleyMultipartRequest(int method, String url,
```

```
        this.mListener = listener;
        this.mErrorListener = errorListener;
    }

    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        return (mHeaders != null) ? mHeaders : super.getHeaders();
    }

    @Override
    public String getBodyContentType() {
        return "multipart/form-data;boundary=" + boundary;
    }

    @Override
    public byte[] getBody() throws AuthFailureError {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);

        try {
            // populate text payload
            Map<String, String> params = getParams();
            if (params != null && params.size() > 0) {
                textParse(dos, params, getParamsEncoding());
            }

            // populate data byte payload
            Map<String, DataPart> data = getByteData();
            if (data != null && data.size() > 0) {
                dataParse(dos, data);
            }

            // close multipart form data after text and file data
            dos.writeBytes(twoHyphens + boundary + twoHyphens + lineEnd);

            return bos.toByteArray();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
79      /**
80       * Custom method handle data payload.
81       *
82       * @return Map data part label with data byte
83       * @throws AuthFailureError
84       */
85      protected Map<String, DataPart> getByteData() throws AuthFailureError {
86          return null;
87      }
88
89      @Override
90      protected Response<NetworkResponse> parseNetworkResponse(NetworkResponse response) {
91          try {
92              return Response.success(
93                      response,
94                      HttpHeaderParser.parseCacheHeaders(response));
95          } catch (Exception e) {
96              return Response.error(new ParseError(e));
97          }
98      }
99
100     @Override
101     protected void deliverResponse(NetworkResponse response) {
102         mListener.onResponse(response);
103     }
104
105     @Override
106     public void deliverError(VolleyError error) {
107         mErrorListener.onErrorResponse(error);
108     }
109
110     /**
111      * Parse string map into data output stream by key and value.
112      *
113      * @param dataOutputStream data output stream handle string parsing
114      * @param params           string inputs collection
115      * @param encoding         encode the inputs, default UTF-8
116      * @throws IOException
117      */
```

```
121                 buildTextPart(dataOutputStream, entry.getKey(), entry.getValue());
122             }
123         } catch (UnsupportedEncodingException uee) {
124             throw new RuntimeException("Encoding not supported: " + encoding, uee);
125         }
126     }
127 
128     /**
129      * Parse data into data output stream.
130      *
131      * @param dataOutputStream data output stream handle file attachment
132      * @param data             loop through data
133      * @throws IOException
134      */
135     private void dataParse(DataOutputStream dataOutputStream, Map<String, DataPart> data) throw
136         for (Map.Entry<String, DataPart> entry : data.entrySet()) {
137             buildDataPart(dataOutputStream, entry.getValue(), entry.getKey());
138         }
139     }
140 
141     /**
142      * Write string data into header and data output stream.
143      *
144      * @param dataOutputStream data output stream handle string parsing
145      * @param parameterName    name of input
146      * @param parameterValue   value of input
147      * @throws IOException
148      */
149     private void buildTextPart(DataOutputStream dataOutputStream, String parameterName, String
150         dataOutputStream.writeBytes(twoHyphens + boundary + lineEnd);
151         dataOutputStream.writeBytes("Content-Disposition: form-data; name=\"" + parameterName +
152         dataOutputStream.writeBytes(lineEnd);
153         dataOutputStream.writeBytes(parameterValue + lineEnd);
154     }
155 
156     /**
157      * Write data file into header and data output stream.
158      *
159      * @param dataOutputStream data output stream handle data parsing
```

```
163      */
164     private void buildDataPart(DataOutputStream dataOutputStream, DataPart dataFile, String inpu
165         dataOutputStream.writeBytes(twoHyphens + boundary + lineEnd);
166         dataOutputStream.writeBytes("Content-Disposition: form-data; name=\"" +
167                 inputName + "\"; filename=\"" + dataFile.getFileName() + "\"" + lineEnd);
168         if (dataFile.getType() != null && !dataFile.getType().trim().isEmpty()) {
169             dataOutputStream.writeBytes("Content-Type: " + dataFile.getType() + lineEnd);
170         }
171         dataOutputStream.writeBytes(lineEnd);
172 
173         ByteArrayInputStream fileInputStream = new ByteArrayInputStream(dataFile.getContent());
174         int bytesAvailable = fileInputStream.available();
175 
176         int maxBufferSize = 1024 * 1024;
177         int bufferSize = Math.min(bytesAvailable, maxBufferSize);
178         byte[] buffer = new byte[bufferSize];
179 
180         int bytesRead = fileInputStream.read(buffer, 0, bufferSize);
181 
182         while (bytesRead > 0) {
183             dataOutputStream.write(buffer, 0, bufferSize);
184             bytesAvailable = fileInputStream.available();
185             bufferSize = Math.min(bytesAvailable, maxBufferSize);
186             bytesRead = fileInputStream.read(buffer, 0, bufferSize);
187         }
188 
189         dataOutputStream.writeBytes(lineEnd);
190     }
191 
192     class DataPart {
193         private String fileName;
194         private byte[] content;
195         private String type;
196 
197         public DataPart() {
198         }
199 
200         DataPart(String name, byte[] data) {
201             fileName = name;
```

```
205         String getFileName() {
206             return fileName;
207         }
208 
209         byte[] getContent() {
210             return content;
211         }
212 
213         String getType() {
214             return type;
215         }
216 
217     }
218 }
```

Upload Image to Server

- Now the last thing is uploading the image, and we will do it inside **MainActivity.java**.

MainActivity.java Java

```
1  package net.simplifiedlearning.androiduploadimage;
2  
3  import android.Manifest;
4  import android.content.Intent;
5  import android.content.pm.PackageManager;
6  import android.graphics.Bitmap;
7  import android.net.Uri;
8  import android.os.Build;
9  import android.os.Bundle;
10 import android.provider.MediaStore;
11 import android.provider.Settings;
12 import android.support.v4.content.ContextCompat;
13 import android.support.v7.app.AppCompatActivity;
14 import android.view.View;
15 import android.widget.EditText;
16 import android.widget.ImageView;
17 import android.widget.Toast;
18 
```

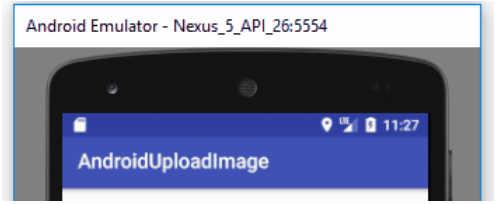
```
22 import com.android.volley.Response;
23 import com.android.volley.VolleyError;
24 import com.android.volley.toolbox.Volley;
25
26 import org.json.JSONException;
27 import org.json.JSONObject;
28
29 import java.io.ByteArrayOutputStream;
30 import java.io.IOException;
31 import java.util.HashMap;
32 import java.util.Map;
33
34
35 public class MainActivity extends AppCompatActivity {
36
37     //ImageView to display image selected
38     ImageView imageView;
39
40     //edittext for getting the tags input
41     EditText editTextTags;
42
43     @Override
44     protected void onCreate(Bundle savedInstanceState) {
45         super.onCreate(savedInstanceState);
46         setContentView(R.layout.activity_main);
47
48         //initializing views
49         imageView = (ImageView) findViewById(R.id.imageView);
50         editTextTags = (EditText) findViewById(R.id.editTextTags);
51
52         //checking the permission
53         //if the permission is not given we will open setting to add permission
54         //else app will not open
55         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && ContextCompat.checkSelfPermission(
56                 Manifest.permission.READ_EXTERNAL_STORAGE)
57                 != PackageManager.PERMISSION_GRANTED) {
58             Intent intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
59                     Uri.parse("package:" + getPackageName()));
60             finish();
```

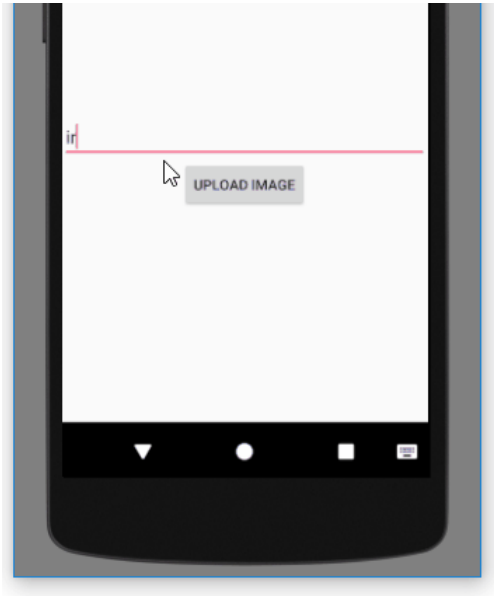
```
64
65
66          //adding click listener to button
67          findViewById(R.id.buttonUploadImage).setOnClickListener(new View.OnClickListener() {
68              @Override
69              public void onClick(View view) {
70
71                  //if the tags edittext is empty
72                  //we will throw input error
73                  if (editTextTags.getText().toString().trim().isEmpty()) {
74                      editTextTags.setError("Enter tags first");
75                      editTextTags.requestFocus();
76                      return;
77                  }
78
79                  //if everything is ok we will open image chooser
80                  Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTE
81                  startActivityForResult(i, 100);
82              }
83          });
84      }
85
86      @Override
87      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
88          super.onActivityResult(requestCode, resultCode, data);
89          if (requestCode == 100 && resultCode == RESULT_OK && data != null) {
90
91              //getting the image Uri
92              Uri imageUri = data.getData();
93              try {
94                  //getting bitmap object from uri
95                  Bitmap bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), ima
96
97                  //displaying selected image to imageview
98                  imageView.setImageBitmap(bitmap);
99
100                 //calling the method uploadBitmap to upload image
101                 uploadBitmap(bitmap);
102             } catch (IOException e) {
```

```
106     }
107
108     /*
109     * The method is taking Bitmap as an argument
110     * then it will return the byte[] array for the given bitmap
111     * and we will send this array to the server
112     * here we are using PNG Compression with 80% quality
113     * you can give quality between 0 to 100
114     * 0 means worse quality
115     * 100 means best quality
116     * */
117     public byte[] getFileDataFromDrawable(Bitmap bitmap) {
118         ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
119         bitmap.compress(Bitmap.CompressFormat.PNG, 80, byteArrayOutputStream);
120         return byteArrayOutputStream.toByteArray();
121     }
122
123     private void uploadBitmap(final Bitmap bitmap) {
124
125         //getting the tag from the edittext
126         final String tags = editTextTags.getText().toString().trim();
127
128         //our custom volley request
129         VolleyMultipartRequest volleyMultipartRequest = new VolleyMultipartRequest(Request.Metho
130                 new Response.Listener<NetworkResponse>() {
131                     @Override
132                     public void onResponse(NetworkResponse response) {
133                         try {
134                             JSONObject obj = new JSONObject(new String(response.data));
135                             Toast.makeText(getApplicationContext(), obj.getString("message"), To
136                         } catch (JSONException e) {
137                             e.printStackTrace();
138                         }
139                     }
140                 },
141                 new Response.ErrorListener() {
142                     @Override
143                     public void onErrorResponse(VolleyError error) {
144                         Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH
```

```
148             /*
149             * If you want to add more parameters with the image
150             * you can do it here
151             * here we have only one parameter with the image
152             * which is tags
153             * */
154             @Override
155             protected Map<String, String> getParams() throws AuthFailureError {
156                 Map<String, String> params = new HashMap<>();
157                 params.put("tags", tags);
158                 return params;
159             }
160 
161             /*
162             * Here we are passing image by renaming it with a unique name
163             * */
164             @Override
165             protected Map<String, DataPart> getByteData() {
166                 Map<String, DataPart> params = new HashMap<>();
167                 long imagename = System.currentTimeMillis();
168                 params.put("pic", new DataPart(imagename + ".png", getFileDataFromDrawable(bitma
169                 return params;
170             }
171         };
172 
173         //adding the request to volley
174         Volley.newRequestQueue(this).add(volleyMultipartRequest);
175     }
176 }
```

- That's it now you can try running your application.





Android Upload Image to Server

- So upload is working ne. Now the next step is fetching the uploaded images back.

Getting the Images Back to our Application

- We already created the API call to get all the uploaded images.

```
      "id": 3,
      "image": "http://192.168.225.1/MyApi/uploads/1508910774346.png",
      "tags": "spidy"
    },
    {
      "id": 4,
      "image": "http://192.168.225.1/MyApi/uploads/icon_gift.png",
      "tags": "gift"
    },
    {
      "id": 5,
      "image": "http://192.168.225.1/MyApi/uploads/1508911054989.png",
      "tags": "iron man"
    }
  ]
}
```


- Now we can use this URL to get the images back. I have already posted a tutorial about this. So for this part you can visit the below tutorial.

Building a RecyclerView with Images and Text

Android Upload Image to Server Source Code

- If you are having any trouble following the post, then you can get my source code as well.