You can use this library to consume rest service. You will need to add external library gson (compile files(**'libs/gson-2.2.2.jar'**). I have shown here how to send namevalue pair. You can send JSON object too.

////How to use this//////////

**import** com.atravesar.eiinar.eiinardriver.util.JSONHttpClient;

class LoadNearbyAddresses extends AsyncTask<String, String, String> {

NearbyPlacesReq\_Result places;

@Override

protected String doInBackground(String... params) {

List<NameValuePair> args = new ArrayList<NameValuePair>();

args.add(new BasicNameValuePair("prefix",”searchvalue”));

JSONHttpClient jsonHttpClient = new JSONHttpClient();

try{

places = jsonHttpClient.Get(ServiceUrl, args, NearbyPlacesReq\_Result.class);

//the returned data will be parsed to the given class type

}

catch (Exception ex){return ex.toString();}

return null;

}

@Override

protected void onPreExecute() {

super.onPreExecute();

showProgressDialog();

}

@Override

protected void onPostExecute(String s) {

dismissProgressDialog();

if(s!=null || places ==null){

//error

return;

}

if(places.getcode()==1)

{

//success

//use the data

}

else if(job.geterror\_code()==0)

{

//error

}

}

}

/////////////////////////////Library class////////////**import** com.google.gson.GsonBuilder;  
  
**import** org.apache.http.Header;  
**import** org.apache.http.HttpEntity;  
**import** org.apache.http.HttpResponse;  
**import** org.apache.http.HttpStatus;  
**import** org.apache.http.NameValuePair;  
**import** org.apache.http.client.ClientProtocolException;  
**import** org.apache.http.client.methods.HttpDelete;  
**import** org.apache.http.client.methods.HttpGet;  
**import** org.apache.http.client.methods.HttpPost;  
**import** org.apache.http.client.utils.URLEncodedUtils;  
**import** org.apache.http.entity.StringEntity;  
**import** org.apache.http.impl.client.DefaultHttpClient;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.io.UnsupportedEncodingException;  
**import** java.util.List;  
**import** java.util.zip.GZIPInputStream;  
  
*/\*\*  
 \* Created by Anup on 09/11/2015.  
 \*/***public class** JSONHttpClient {  
  
 **public** <T> T PostObject(**final** String url, **final** T object, **final** Class<T> objectClass) {  
 DefaultHttpClient defaultHttpClient = **new** DefaultHttpClient();  
 HttpPost httpPost = **new** HttpPost(url);  
 **try** {  
  
 StringEntity stringEntity = **new** StringEntity(**new** GsonBuilder().create().toJson(object));  
 httpPost.setEntity(stringEntity);  
 httpPost.setHeader(**"Accept"**, **"application/json"**);  
 httpPost.setHeader(**"Content-type"**, **"application/json"**);  
 httpPost.setHeader(**"Accept-Encoding"**, **"gzip"**);  
  
 HttpResponse httpResponse = defaultHttpClient.execute(httpPost);  
 HttpEntity httpEntity = httpResponse.getEntity();  
 **if** (httpEntity != **null**) {  
 InputStream inputStream = httpEntity.getContent();  
 Header contentEncoding = httpResponse.getFirstHeader(**"Content-Encoding"**);  
 **if** (contentEncoding != **null** && contentEncoding.getValue().equalsIgnoreCase(**"gzip"**)) {  
 inputStream = **new** GZIPInputStream(inputStream);  
 }  
  
 String resultString = convertStreamToString(inputStream);  
  
 inputStream.close();  
 **return new** GsonBuilder().setDateFormat(**"yyyy-MM-dd'T'HH:mm:ss"**).create().fromJson(resultString, objectClass);  
  
 }  
  
 } **catch** (UnsupportedEncodingException e) {  
 *//e.printStackTrace();* } **catch** (ClientProtocolException e) {  
 *//e.printStackTrace();* } **catch** (IOException e) {  
 *//e.printStackTrace();* }  
 **catch** (Exception e) {  
 *//e.printStackTrace();* }  
 **return null**;  
 }  
  
 **public** <T> T PostParams(String url, **final** List<NameValuePair> params, **final** Class<T> objectClass) {  
 String paramString = URLEncodedUtils.*format*(params, **"utf-8"**);  
 url += **"?"** + paramString;  
 **return** PostObject(url, **null**, objectClass);  
 }  
  
 **private** String convertStreamToString(InputStream inputStream) {  
 BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));  
 StringBuilder stringBuilder = **new** StringBuilder();  
 String line = **null**;  
 **try** {  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 stringBuilder.append(line + **"\n"**);  
 }  
 } **catch** (IOException e) {  
 *// e.printStackTrace();* } **finally** {  
 **try** {  
 inputStream.close();  
 } **catch** (IOException e) {  
 *// e.printStackTrace();* }  
 }  
  
 **return** stringBuilder.toString();  
 }  
  
 **public** <T> T Get(String url, List<NameValuePair> params, **final** Class<T> objectClass) {  
 DefaultHttpClient defaultHttpClient = **new** DefaultHttpClient();  
 String paramString = URLEncodedUtils.*format*(params, **"utf-8"**);  
 url += **"?"** + paramString;  
 HttpGet httpGet = **new** HttpGet(url);  
 **try** {  
  
 httpGet.setHeader(**"Accept"**, **"application/json"**);  
 httpGet.setHeader(**"Accept-Encoding"**, **"gzip"**);  
  
 HttpResponse httpResponse = defaultHttpClient.execute(httpGet);  
 HttpEntity httpEntity = httpResponse.getEntity();  
 **if** (httpEntity != **null**) {  
 InputStream inputStream = httpEntity.getContent();  
 Header contentEncoding = httpResponse.getFirstHeader(**"Content-Encoding"**);  
 **if** (contentEncoding != **null** && contentEncoding.getValue().equalsIgnoreCase(**"gzip"**)) {  
 inputStream = **new** GZIPInputStream(inputStream);  
 }  
  
 String resultString = convertStreamToString(inputStream);  
 inputStream.close();  
  
 **return new** GsonBuilder().setDateFormat(**"yyyy-MM-dd'T'HH:mm:ss"**).create().fromJson(resultString, objectClass);  
 }  
  
 } **catch** (UnsupportedEncodingException e) {  
 *//e.printStackTrace();* } **catch** (ClientProtocolException e) {  
 *//e.printStackTrace();* } **catch** (IOException e) {  
 *//e.printStackTrace();* }**catch** (Exception e) {  
 *//e.printStackTrace();* }  
 **return null**;  
 }  
  
 **public boolean** Delete(String url, **final** List<NameValuePair> params) {  
 DefaultHttpClient defaultHttpClient = **new** DefaultHttpClient();  
 String paramString = URLEncodedUtils.*format*(params, **"utf-8"**);  
 url += **"?"** + paramString;  
 HttpDelete httpDelete = **new** HttpDelete(url);  
  
 HttpResponse httpResponse = **null**;  
 **try** {  
 httpResponse = defaultHttpClient.execute(httpDelete);  
 **return** httpResponse.getStatusLine().getStatusCode() == HttpStatus.***SC\_NO\_CONTENT***;  
 } **catch** (IOException e) {  
 *//e.printStackTrace();* }  
  
 **return false**;  
 }  
}

////worker classes////////////

public class Addresses

{

private String Outcode;

public final String getOutcode()

{

return Outcode;

}

public final void setOutcode(String value)

{

Outcode = value;

}

private String Postcode;

public final String getPostcode()

{

return Postcode;

}

public final void setPostcode(String value)

{

Postcode = value;

}

private String Fulladdress;

public final String getFulladdress()

{

return Fulladdress;

}

public final void setFulladdress(String value)

{

Fulladdress = value;

}

private String Category;

public final String getCategory()

{

return Category;

}

public final void setCategory(String value)

{

Category = value;

}

private String Icon\_Path;

public final String getIcon\_Path()

{

return Icon\_Path;

}

public final void setIcon\_Path(String value)

{

Icon\_Path = value;

}

private java.math.BigDecimal Latitude = new java.math.BigDecimal(0);

public final java.math.BigDecimal getLatitude()

{

return Latitude;

}

public final void setLatitude(java.math.BigDecimal value)

{

Latitude = value;

}

private java.math.BigDecimal Longitude = new java.math.BigDecimal(0);

public final java.math.BigDecimal getLongitude()

{

return Longitude;

}

public final void setLongitude(java.math.BigDecimal value)

{

Longitude = value;

}

}

public class NearbyPlacesReq\_Result

{

public NearbyPlacesReq\_Result()

{

setaddresses(new ArrayList<Addresses>());

}

private ArrayList<Addresses> addresses;

public final ArrayList<Addresses> getaddresses()

{

return addresses;

}

public final void setaddresses(ArrayList<Addresses> value)

{

addresses = value;

}

private String message;

public final String getmessage()

{

return message;

}

public final void setmessage(String value)

{

message = value;

}

private int code;

public final int getcode()

{

return code;

}

public final void setcode(int value)

{

code = value;

}

}