Make an HTTP request with android

I have searched everywhere but I couldn't find my answer, is there a way to make an simple HTTP request? I want to request an PHP page / script on one of my website but I don't want to show the webpage.

If possible I even want to do it in the background (in an BroadcastReceiver)

android httpwebrequest androidhttpclient

edited May 15 at 10:38
kablu
1 ●16

asked Aug 17 '10 at 19:00

Mats Hofman
2,264 ●3 ●18 ●41

Related: Download a resource, and show a progress dialog – rds Jun 11 at 8:35

add a comment

6 Answers

First of all, request a permission to access network, add following to your manifest:

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

Then the easiest way is to use Apache http client bundled with Android:

```
    HttpClient httpclient = new DefaultHttpClient();
    HttpResponse response = httpclient.execute(new HttpGet(URL));
    StatusLine statusLine = response.getStatusLine();
    if(statusLine.getStatusCode() == HttpStatus.SC_OK){
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        response.getEntity().writeTo(out);
        out.close();
        String responseString = out.toString();
        //..more logic
    } else{
        //Closes the connection.
        response.getEntity().getContent().close();
        throw new IOException(statusLine.getReasonPhrase());
    }
```

If you want it to run on separate thread I'd recommend extending AsyncTask:

```
class RequestTask extends AsyncTask<String, String, String>{

    @Override
    protected String doInBackground(String... uri) {
        HttpClient httpclient = new DefaultHttpClient();
        HttpResponse response;
        String responseString = null;
        try {
            response = httpclient.execute(new HttpGet(uri[0]));
            StatusLine statusLine = response.getStatusLine();
            if(statusLine.getStatusCode() == HttpStatus.SC_OK){
                ByteArrayOutputStream out = new ByteArrayOutputStream();
                response.getEntity().writeTo(out);
                out.close();
                responseString = out.toString();
            } else{
                //Closes the connection.
                response.getEntity().getContent().close();
                throw new IOException(statusLine.getReasonPhrase());
            }
        } catch (ClientProtocolException e) {
            //TODO Handle problems..
        } catch (IOException e) {
            //TODO Handle problems..
        }
        return responseString;
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        //Do anything with response..
    }
}
```

You then can make a request by:

```
new RequestTask().execute("http://stackoverflow.com");
```

edited Nov 8 '10 at 8:50

answered Aug 17 '10 at 19:11

Konstantin Burov

33.3k ●10 ●68 ●69

8 Here is an article from the official android developer blog on AsyncTask: android-developers.blogspot.com/2010/07/… – Austyn Mahoney Aug 17 '10 at 21:44

44 for gingerbread or greater its actually advised to use HttpURLConnection over the apache library, see android-developers.blogspot.com/2011/09/… . Its less taxing on the battery and has better performance – Marty Dec 5 '11 at 21:54

7 responseString = out.toString() needs to be before the out.close() call. Actually, you should probably have the out.close() in a finally block. But overall, very helpful answer (+1), thanks! – dcp May 1 '12 at 15:34

9 As of Honeycomb (SDK 11) the asynchronous approach is the way to go. A NetworkOnMainThreadException gets thrown when you try to run an HTTP request from the main thread. – user14269686 Aug 26 '12 at 21:15

2 This answer is quite excellent. But I would advise not to use AsyncTasks for Networking. They can create memory leaks very easily (and actually the provided example does leak), and don't provide all features one can expect for network requests. Consider using RoboSpice for this kind of background tasks : github.com/octo-online/robospice – Snicolas Nov 9 '12 at 12:59

show **7** more comments

---

unless you have an explicit reason to choose the Apache HttpClient, you should prefer java.net.URLConnection. you can find plenty of examples of how to use it on the web.

we've also improved the Android documentation since your original post: http://developer.android.com/reference/java/net/HttpURLConnection.html

and we've talked about the trade-offs on the official blog: http://android-developers.blogspot.com/2011/09/androids-http-clients.html

edited Jan 22 '12 at 23:01

answered Aug 18 '10 at 5:55

Elliott Hughes

3,148 ●11 ●17

10 Why is using Apache HttpClient not recommended? – Ted Oct 26 '11 at 20:09

3 A co-conspirator of mine went into this in great detail on the official blog: android-developers.blogspot.com/2011/09/… – Elliott Hughes Jan 22 '12 at 23:00

@ElliottHughes : I agree 100%. There's no denying that Apache httpclient offers easy methods and a more abstracted view of protocol, but java's native urlconnection is in no way less useful. With a bit of hands-on, its as easy to use as httpclient, and is way more portable – Nitin Bansal Aug 8 '12 at 20:19

Actually if u take a look at the video Google I/O 2010 - Android REST client applications(youtube.com/watch?v=xHXn3Kg2IQE 57min21sec) you would see that Apache HttpClient it is the most recommended one. I quote Virgil Dobjanschi(a software engineer on google that works on Android Application Group) "I would simply advise that you use the HTTP Apache client, because it has a more robust implementation.The URL connection type of HTTP transaction is not the most efficient implementation.And the way it terminates connections sometimes can have an adverse effect on the network." – Alan May 14 at 17:51

add a comment

Try this, it is working for me.

First, add permissions in manifest file:

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

Then use below code to make the HTTP request:

```
private void getdata() {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.
          ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL("http://ruit.mytechlabs.com/responsive_web_testing.php");
        HttpURLConnection con = (HttpURLConnection) url
          .openConnection();
        readStream(con.getInputStream());
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void readStream(InputStream in) {
  BufferedReader reader = null;
  try {
    reader = new BufferedReader(new InputStreamReader(in));
    String line = "";
    while ((line = reader.readLine()) != null) {
      System.out.println(line);
    }
  } catch (IOException e) {
    e.printStackTrace();
  } finally {
    if (reader != null) {
      try {
        reader.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
}
```

edited Nov 10 '13 at 19:08
Adi Inbar
4,368 ●8 ●15 ●30

answered Mar 7 '13 at 15:26
Muktesh Kumar
251 ●2 ●8

1 You have too many curly brackets... – sirvon Mar 27 '13 at 1:55

How about some more curly brackets - i.e., a finally block containing con.disconnect() within a try/catch ?? – ban-geoengineering May 9 at 15:46

add a comment

With a thread:

```
private class LoadingThread extends Thread {
    Handler handler;

    LoadingThread(Handler h) {
        handler = h;
    }
    @Override
    public void run() {
        Message m = handler.obtainMessage();
        try {
            BufferedReader in =
                new BufferedReader(new InputStreamReader(url.openStream()));
            String page = "";
            String inLine;

            while ((inLine = in.readLine()) != null) {
                page += inLine;
            }

            in.close();
            Bundle b = new Bundle();
            b.putString("result", page);
            m.setData(b);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        handler.sendMessage(m);
    }
}
```

edited Apr 18 '13 at 21:12

Troy Alford
9,036 ●5 ●30 ●52

answered Aug 17 '10 at 19:03

Tom Medley
9,767 ●10 ●60 ●121

add a comment

```
private String getToServer(String service) throws IOException {
    HttpGet httpget = new HttpGet(service);
    ResponseHandler<String> responseHandler = new BasicResponseHandler();
    return new DefaultHttpClient().execute(httpget, responseHandler);

}
```

Regards

answered Jul 30 at 18:07

Gabriel Gómez
31 ●4

add a comment

I made this for a webservice to requerst on URL, using a Gson lib:

Client:

```
public EstabelecimentoList getListaEstabelecimentoPorPromocao(){

        EstabelecimentoList estabelecimentoList  = new EstabelecimentoList();
        try{
            URL url = new URL("http://" +  Conexao.getSERVIDOR()+ "/cardapio.online/res
            HttpURLConnection con = (HttpURLConnection) url.openConnection();

            if (con.getResponseCode() != 200) {
                    throw new RuntimeException("HTTP error code : "+ con.getResponseCod
            }

            BufferedReader br = new BufferedReader(new InputStreamReader((con.getInputS
            estabelecimentoList = new Gson().fromJson(br, EstabelecimentoList.class);
            con.disconnect();

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

answered Jun 5 at 13:51

Deividson Calixto
4 ●2
add a comment

Not the answer you're looking for? Browse other questions tagged android httpwebrequest androidhttpclient **or ask your own question.**