

# Android HTTP Request

# CONTENTS

- HTTP REQUEST
- Simple example to make HTTP request
- Async method
- Simple example on the working of async method

# HTTP REQUEST

- HttpURLConnection class is an abstract class directly extending from URLConnection class
- HttpsURLConnection is way more secure
- It is used to interact with web servers

# API

- API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.
- Eg. <http://proj309-vc-01.misc.iastate.edu/user> - Say this API gives a list of users online (This is just an example. Please do not click the link)
- Here the API used is [https://api.androidhive.info/volley/string\\_response.html](https://api.androidhive.info/volley/string_response.html)

# Async Task

- AsyncTask enables proper and easy use of the UI thread.
- This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.
- AsyncTask is designed to be a helper class around [Thread](#) and [Handler](#) and does not constitute a generic threading framework.
- AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` package such as [Executor](#), [ThreadPoolExecutor](#) and [FutureTask](#).
- Reference : <https://developer.android.com/reference/android/os/AsyncTask>


# Async Task

- When an asynchronous task is executed, the task goes through 4 steps:
- [onPreExecute\(\)](#), invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
- [doInBackground\(Params...\)](#), invoked on the background thread immediately after [onPreExecute\(\)](#) finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use [publishProgress\(Progress...\)](#) to publish one or more units of progress. These values are published on the UI thread, in the [onProgressUpdate\(Progress...\)](#) step.
- [onProgressUpdate\(Progress...\)](#), invoked on the UI thread after a call to [publishProgress\(Progress...\)](#). The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
- [onPostExecute\(Result\)](#), invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.
- REFERENCE : <https://developer.android.com/reference/android/os/AsyncTask>

# Simple Example

# Android Studio Project

- Open your android studio project.
- Add permission to access the internet.
  - Eg. `<uses-permission android:name="android.permission.INTERNET"/>`
- Where to add it?
  - Ans: AndroidManifest.xml



The screenshot shows the AndroidManifest.xml file in an IDE. The file contains the following XML code:

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

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

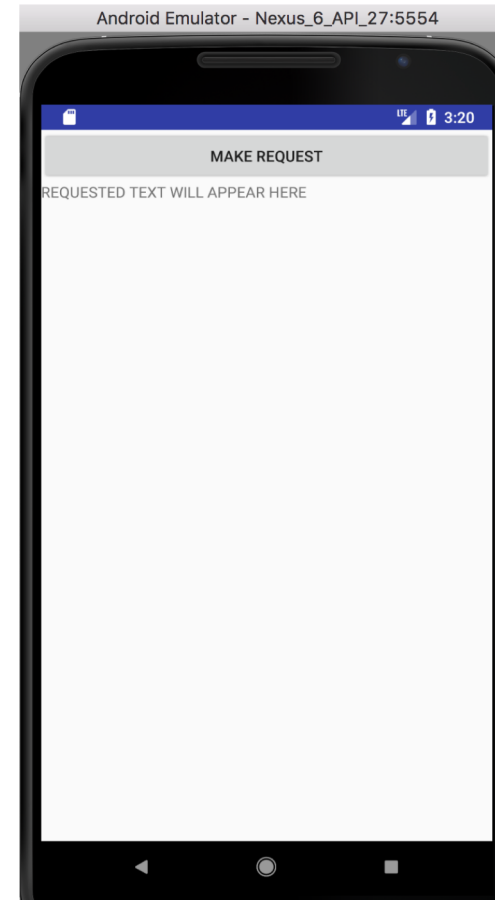
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">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

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



# UI

- Create a button and TextView.
- Add button listeners to call the async function to make http request.



# Making HTTP request

**Make your button listener to call the necessary function.**

```
b1.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        //used for GET/Post request  
        register("vamsi","123");  
    }  
});
```

# Making HTTP request

You can extend an async task by extending this  
`android.os.AsyncTask<Params, Progress, Result>`

Eg.

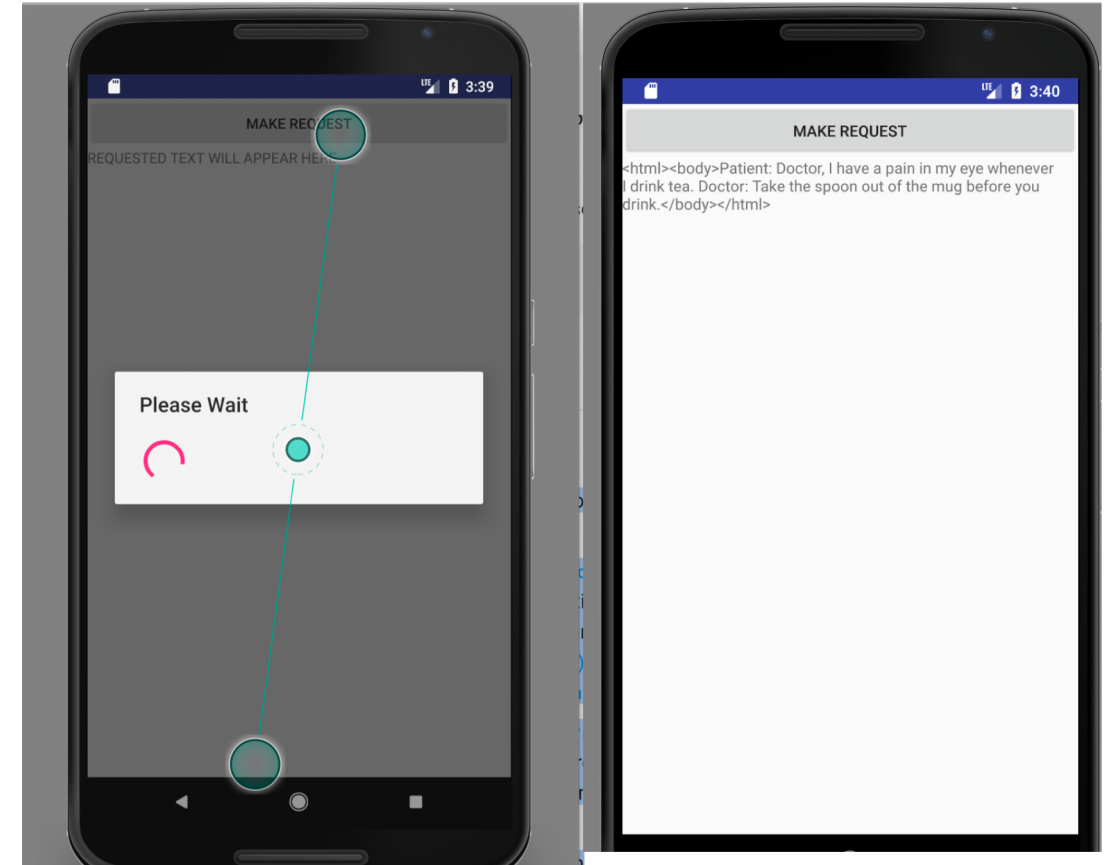
```
private void register(String username, String password) {  
    String urlSuffix =  
    "?username="+username+"&password="+password;  
    class RegisterUser extends AsyncTask<String, Void, String> {
```

# Pre/Post execute and DoInBackground

```
private void register(String username, String password) {  
    //Useful for GET request  
    String urlSuffix = "?username="+username+"&password="+password;  
    //extending the async class  
    class RegisterUser extends AsyncTask<String, Void, String> {  
        ProgressDialog loading;//Loader  
        //implementing methods for asyncTask refer to slide 6  
        @Override  
        protected void onPreExecute() {  
            super.onPreExecute();  
            loading = ProgressDialog.show(MainActivity.this, "Please Wait", null, true, true); //show loader  
        }  
        @Override  
        protected void onPostExecute(String s) {  
            super.onPostExecute(s);  
            loading.dismiss();  
            tx1.setText(s); //assign the text  
        }  
        @Override  
        protected String doInBackground(String... params) {  
            String s = params[0]; //useful when making GET or post request  
            BufferedReader bufferedReader = null; //BR for temporary storage of Input stream  
            try {  
                URL url = new URL(REGISTER_URL); //Register_URL should be your API  
                HttpURLConnection con = (HttpURLConnection) url.openConnection();  
                bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));  
                String result="";  
                for (String line = bufferedReader.readLine(); line != null; line = bufferedReader.readLine()) {  
                    result+=line;  
                }  
                bufferedReader.close();  
                return result;  
            } catch (Exception e) {  
                Toast.makeText(getApplicationContext(), "Check your Internet connection", Toast.LENGTH_SHORT).show();  
                return null;  
            }  
        }  
    }  
    RegisterUser ru = new RegisterUser(); //Create an object  
    ru.execute(urlSuffix); //just call execute it will take care of the sequence of executing like pre/doinbackground/post  
}
```

# Expected Result

- Clicking the button will invoke Async task.
- The async task runs in the background like a thread.
- This async task is independent.
- The result from the async task can be used to update the UI later.
- Go to [https://api.androidhive.info/volley/string\\_response.html](https://api.androidhive.info/volley/string_response.html) and find the exact html code.



# Where to use this in your project?

- This can be used as an alternative for volley.
- Volley library internally uses async request.
- Recommended one is volley.
- Async task can be used anywhere like creating socket connections, calculating something, updating the database.
- Main notion of async task is to not disturb the Main UI thread and run independently.

# Important Links

- [https://git.linux.iastate.edu/vamsi/Async\\_HttpRequest\\_Android](https://git.linux.iastate.edu/vamsi/Async_HttpRequest_Android) - The shown example is available here.
- <https://developer.android.com/reference/android/os/AsyncTask>
- <https://www.mulesoft.com/resources/api/what-is-an-api>
- <https://www.geeksforgeeks.org/java-net-httpurlconnection-class-java/>
- [https://api.androidhive.info/volley/string\\_response.html](https://api.androidhive.info/volley/string_response.html)

# ToDo

- Please Try this by yourself.
- Ask your teammates working on spring to create basic APIs.
- Access that API using this example.
- The API can return a string, html code or a JSON.
- Please use JSON(Recommended)
- Note: This is not compulsory, but please try and ask your doubts.