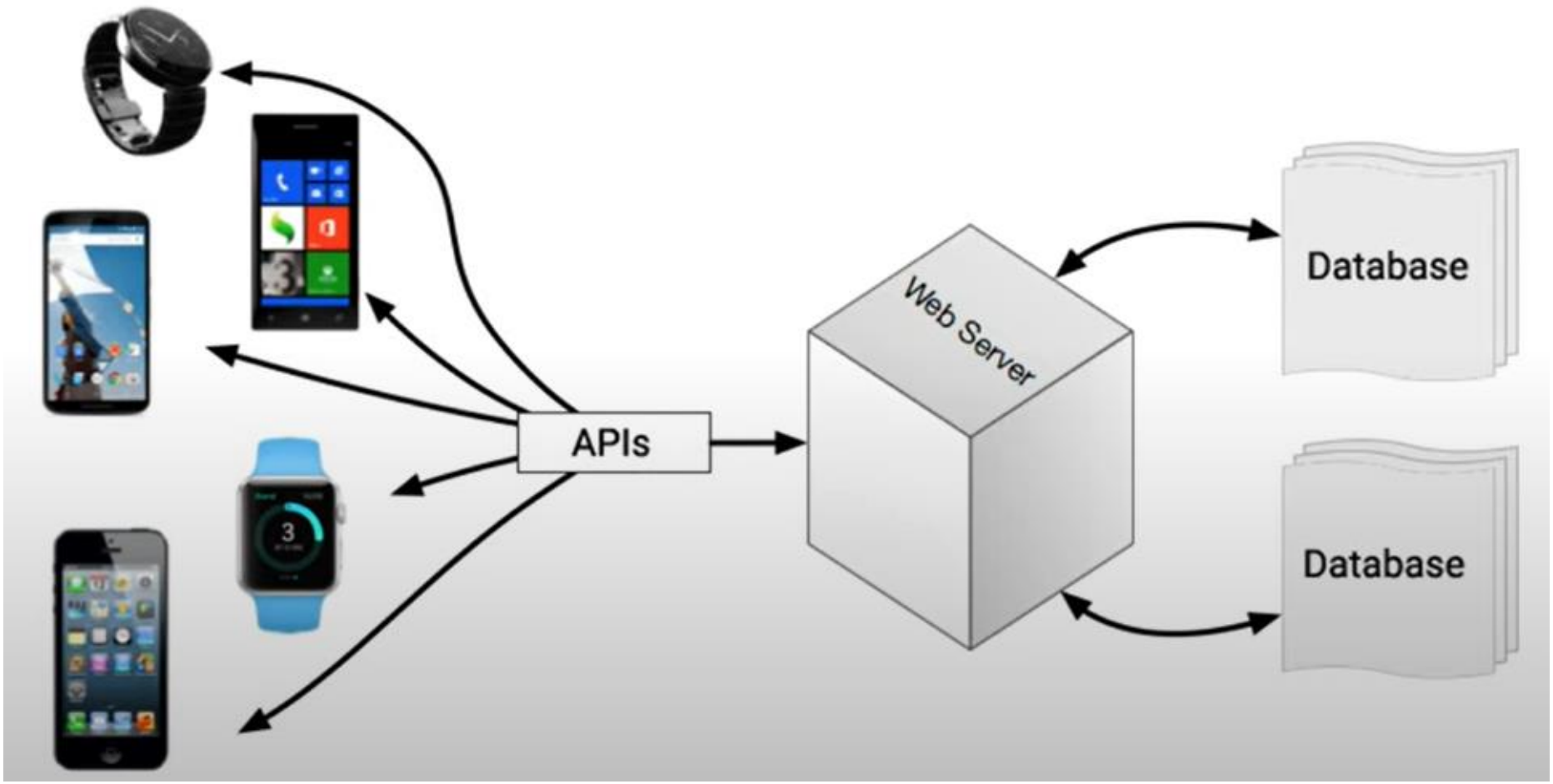
# REST API integration

Module 6

# What is API?

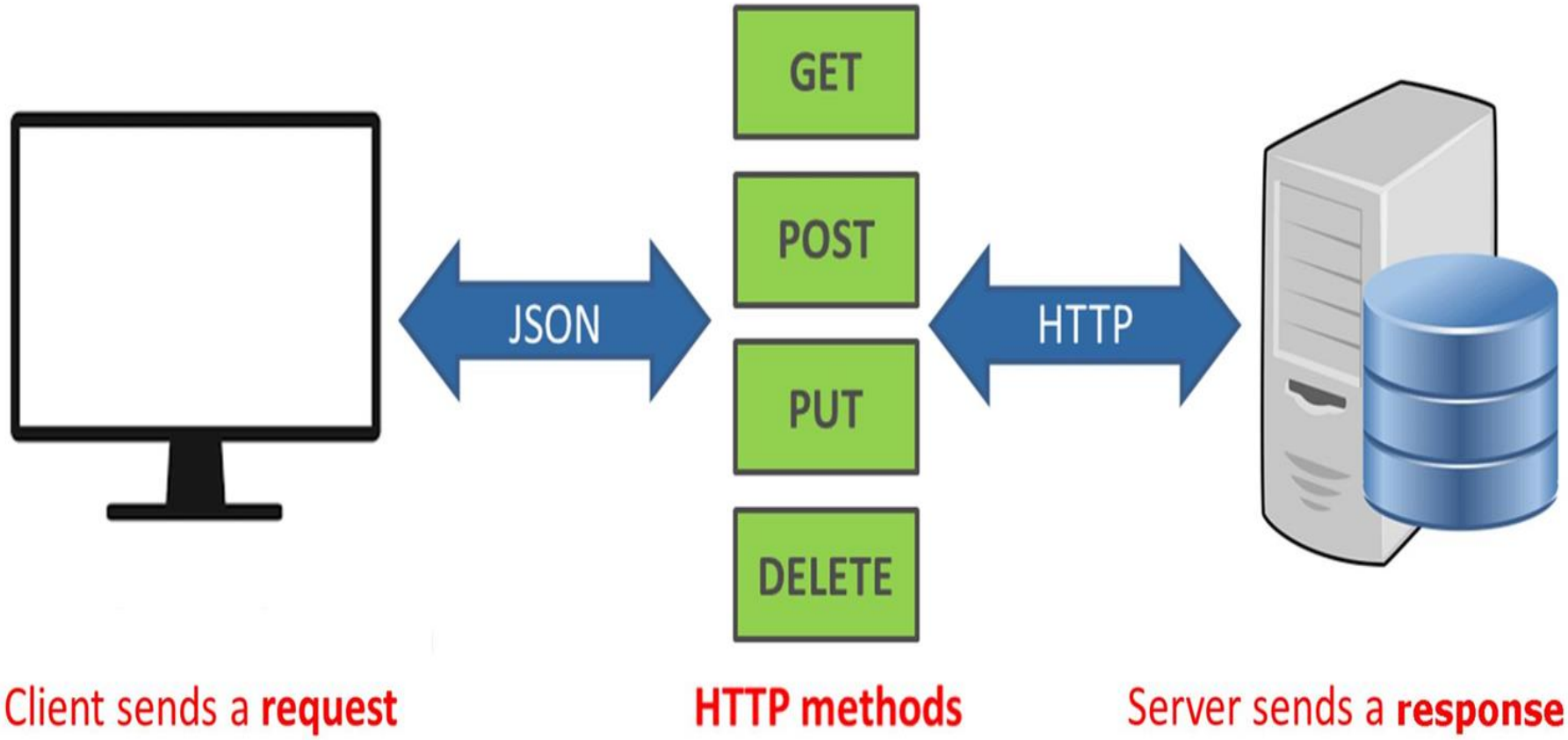


# REST

- REST  is a very popular web API architecture.
- REST stands for **REpresentational State Transfer.**
- It is an architectural style of building APIs that uses HTTP requests to access and use data; not a programming  language.

# What is a REST API?

- An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other.
- A REST API is an API that conforms to the design principles of the REST, or representational state transfer architectural style. For this reason, REST APIs are sometimes referred to RESTful APIs.
- A REST API is designed to provide a lightweight form of communication (less bandwidth) between producer (ex: Twitter) and consumer (ex: Twitter client)
- REST APIs provide a flexible, lightweight way to integrate applications, and have emerged as the most common method for connecting components in microservices architectures.




RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.
This data can be delivered to a client in virtually any format including **JavaScript Object Notation (JSON), HTML, XLT, Python, PHP, or plain text.**

# JSON – Data example

```
{
"_id": "5c8a1d5b0190b214360dc097",
"category": “Crio Bytes",
"clusterName": "OOP Foundations",
"bytes": [“Abstraction”, “Inheritance”, “Encapsulation”, “Polymorphism”]
"isFree": true
}
```

- JSON is data oriented

# How REST APIs work

REST APIs communicate via HTTP requests to perform standard database functions like creating, reading, updating, and deleting records (also known as CRUD) within a resource.

For example, a REST API would use a GET request to retrieve a record, a POST request to create one, a PUT request to update a record, and a DELETE request to delete one.

All HTTP methods can be used in API calls.

A well-designed REST API is similar to a website running in a web browser with built-in HTTP functionality.

The state of a resource at any particular instant, or timestamp, is known as the resource representation. This information can be delivered to a client in virtually any format including JavaScript Object Notation (JSON), HTML, XLT, Python, PHP, or plain text.

JSON is popular because it's readable by both humans and machines—and it is programming language-agnostic.

Request headers and parameters are also important in REST API calls because they include important identifier information such as metadata, authorizations, uniform resource identifiers (URIs), caching,

cookies and more.

Request headers and response headers, along with conventional HTTP status codes, are used within well-designed REST APIs.

# Synchronous vs Asynchronous calls

- **Synchronous** means that you call a web service (or function or whatever) and wait until it returns - all other code execution and user interaction is stopped until the call returns.

- **Asynchronous** means that you do not halt all other operations while waiting for the web service call to return. Other code executes and/or the user can continue to interact with the page (or program UI).

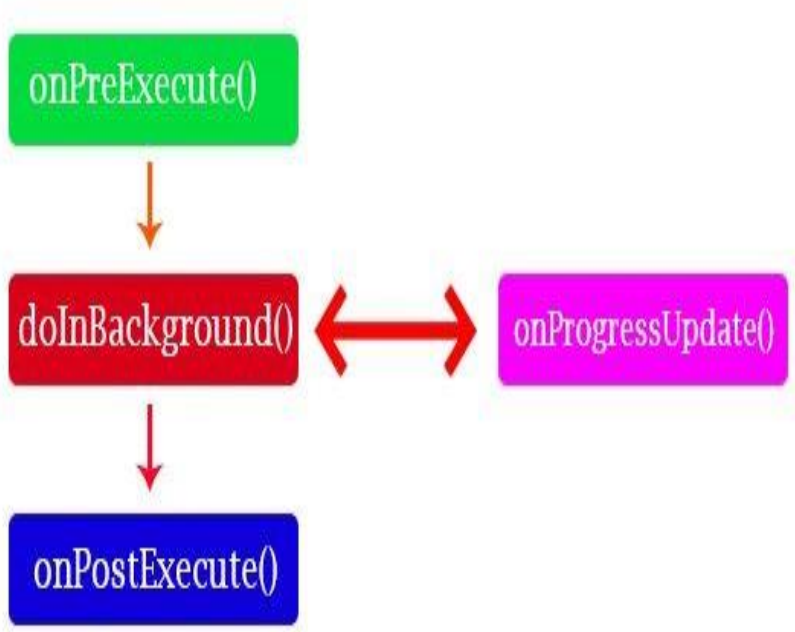
# Android AsyncTask

Android AsyncTask is an abstract class provided by Android which gives us the liberty to perform heavy tasks in the background and keep the UI thread light thus making the application more responsive.

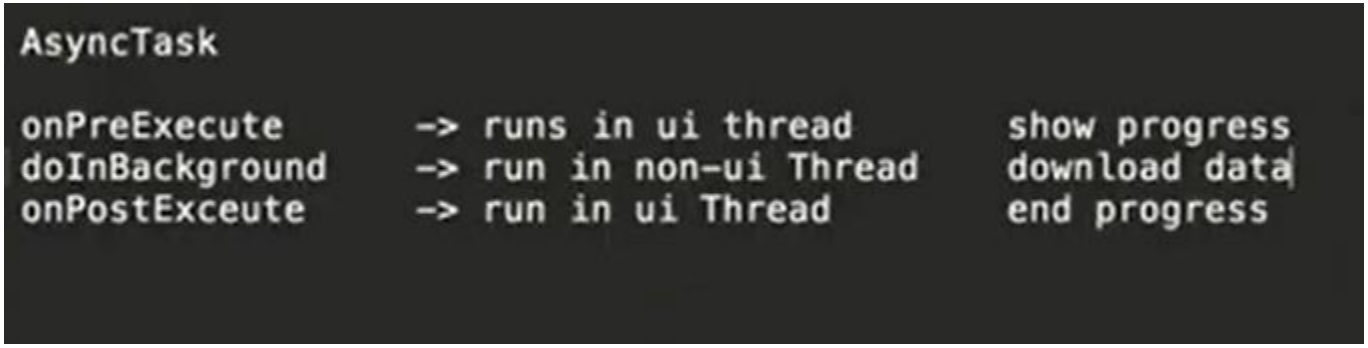
Android application runs on a single thread when launched. Due to this single thread model tasks that take longer time to fetch the response can make the application non-responsive.

To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread. Hence use of AsyncTask in android application keeps the UI thread responsive at all times

- AsyncTask class is used to do background operations that will update the UI(user interface).
- AsyncTask class is firstly executed using **execute()** method.
- In the first step AsyncTask is calls **onPreExecute()** then **onPreExecute()** calls **doInBackground()** for background processes and then **doInBackground()** calls **onPostExecute()** method to update the UI.




UI thread
Non UI Thread
line 1 10 ms
line 2 5 minutes ....
show progress
line 3 20 ms
line 2 callback


AsyncTask
onPreExecute -> runs in ui thread show progress
doInBackground -> run in non-ui Thread download data
onPostExceute -> run in ui Thread end progress

# Basic Methods

- **doInBackground() :** This method contains the code which needs to be executed in background. In this method we can send results multiple times to the UI thread by **publishProgress()** method. To notify that the background processing has been completed we just need to use the return statements
- **onPreExecute()** : This method contains the code which is executed before the background processing starts
- **onPostExecute()** : This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method
- **onProgressUpdate() :** This method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update the UI thread

# Parameters/Generic Types

- **Params :** The type of the parameters sent to the task upon execution
- **Progress :** The type of the progress units published during the background computation
- **Result :** The type of the result of the background computation

# Android AsyncTask Example

•To start an AsyncTask the following snippet must be present in the MainActivity class :

– `MyTask myTask = new MyTask();`
`myTask.execute();`

•In the above snippet we've used a sample classname that extends AsyncTask and execute method is used to start the background thread.

•**Note:**

- The AsyncTask instance must be created and invoked in the UI thread.
- The methods overridden in the AsyncTask class should never be called. They're called automatically.
- AsyncTask can be called only once. Executing it again will throw an exception.

## Syntax of AsyncTask class:

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
protected Long doInBackground(URL... urls) {
// code that will run in the background
return ; }
protected void onProgressUpdate(Integer... progress) {
// receive progress updates from doInBackground }
protected void onPostExecute(Long result) {
// update the UI after background processes completes
}
}
```

```java
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

# JSON Parser

JSON stands for JavaScript Object Notation.
JSON is a lightweight data-interchange format
JSON is plain text written in JavaScript object notation
JSON is used to send data between computers
JSON is language independent

It is an independent data exchange format  and is the best alternative for XML.

Android provides four different classes to manipulate JSON data.

These classes are

- **JSONArray**
- **JSONObject**
- **JSONStringer**
- **JSONTokenizer**

## Methods provided by this class for better parsing JSON files

- **get(String name) :** This method just Returns the value but in the form of Object type
- **getBoolean(String name) :** This method returns the boolean value specified by the key
- **getDouble(String name)** : This method returns the double value specified by the key
- **getInt(String name)** : This method returns the integer value specified by the key
- **getLong(String name)** : This method returns the long value specified by the key
- **length()** : This method returns the number of name/value mappings in this object
- **names() :** This method returns an array containing the string names in this object.

# JSON - Elements

An JSON file consist of many component

**Array([)**

In a JSON file , square bracket ([) represents a JSON array

**Objects({)**

In a JSON file, curly bracket ({) represents a JSON object

**Key**

A JSON object contains a key that is just a string.

Pairs of key/value make up a JSON object

**Value**

Each key has a value that could be string , integer or double e.t.c

# JSON – Data example

```
{
"_id": "5c8a1d5b0190b214360dc097",
"category": “Crio Bytes",
"clusterName": "OOP Foundations",
"bytes": [“Abstraction”, “Inheritance”, “Encapsulation”, “Polymorphism”]
"isFree": true
}
```

- JSON is data oriented

# JSON - Parsing

Android supports all the JSON classes such as **JSONStringer, JSONObject, JSONArray,** and all other forms to parse the JSON data and fetch the required information by the program.

For parsing a JSON object, we will create an object of class JSONObject and specify a string containing JSON data to it.

Syntax

```
String in;
JSONObject reader = new JSONObject(in);
```

The last step is to parse the JSON.

A JSON file consist of different object with different key/value pair e.t.c.

So JSONObject has a separate function for parsing each of the component of JSON file.

Syntax

```
JSONObject sys = reader.getJSONObject("sys");
 country = sys.getString("country");

JSONObject main = reader.getJSONObject("main");
 temperature = main.getString("temp");
```

The method getJSONObject returns the JSON object.

The method getString returns the string value of the specified key.

# Introduction to HttpUrlConnection

- HttpUrlConnection is HTTP library used to to send http request and get http server response in android application.
- Use **java.net.HttpURLConnection** class to send http request and get http server response in android application.
- HttpUrlConnection is not prepared for native asynchronous calls, requiring us to make use of workers, such as AsyncTasks, to perform HTTP network calls outside the main UI thread.

- **HttpURLConnection use steps.**
  - Create a URL instance.
    - **URL url = new URL("http://www.yahoo.com");**
  - Open url connection and cast it to HttpURLConnection.
    - **HttpURLConnection httpConn = (HttpURLConnection)url.openConnection();**
  - Set request method.
    - The request method can be GET, POST, DELETE etc.
    - Request method must be uppercase, otherwise exception will be thrown.

# Introduction to HttpUrlConnection and JSON

1. If request method is **GET**, then use below code to open input stream and read server response data.

```
InputStream inputStream = httpConn.getInputStream();
InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
String line = bufferedReader.readLine();
```

2. For **POST** request method, use below code to open output stream to the server url and write post data to it. After post, open the input stream again to get server response.

```
OutputStream outputStream = httpConn.getOutputStream();
OutputStreamWriter outputStreamWriter = new OutputStreamWriter(outputStream);
BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);
bufferedWriter.write("user=jerry&pasword=666666");
```

3. After use HttpURLConnection, do not forget close related reader or writer and call HttpURLConnection's disconnect() method to close the connection to release network resources.

```
httpConn.disconnect();
```

# ScrollView

- In Android, a ScrollView is a view group that is used to make vertically scrollable views.

# Attributes

| Attributes | Description |
| --- | --- |
| android:alpha | alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque). |
| android:background | A drawable to use as the background. |
| android:clickable | Defines whether this view reacts to click events. |
| android:contentDescription | Defines text that briefly describes content of the view. |
| android:id | Supply an identifier name for this view, to later retrieve it with View.findViewById() or Activity.findViewById(). |
| android:isScrollContainer | Set this if the view will serve as a scrolling container, meaning that it can be resized to shrink its overall window so that there will be space for an input method. |
| android:minHeight | Defines the minimum height of the view. |
| android:minWidth | Defines the minimum width of the view. |

# Attributes

| | |
|---|---|
| android:onClick | Name of the method in this View's context to invoke when the view is clicked. |
| android:padding | Sets the padding, in pixels, of all four edges. |
| android:scrollbars | Defines which scrollbars should be displayed on scrolling or not. |

- **Create an android application to demonstrate JSON data parsing using HTTPUrlConnection (you can use https://api.github.com/users json data).**

***AndroidManifest.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jasondataparsingusinghttpurl">
<uses-permission android:name="android.permission.INTERNET"/>
    <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/Theme.JasonDataParsingUsingHttpurl">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

***activty_main.xml***

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingHorizontal="16dp"
        android:orientation="vertical">

        <Button
            android:id="@+id/btn_fetch_data"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Fetch Data" />

        <TextView
            android:id="@+id/result_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="result" />

    </LinearLayout>

</ScrollView>
```

MainActivity.java

```java
package com.example.jasondataparsingusinghttpurl;

import androidx.appcompat.app.AppCompatActivity;

import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    Button btnFetchData;
    TextView resultView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnFetchData = findViewById(R.id.btn_fetch_data);
        btnFetchData.setOnClickListener(view -> {
            Users users = new Users();
            users.execute();
        });
        resultView = findViewById(R.id.result_view);
    }
```

```
// class which fetch data from API in separate thread
    class Users extends AsyncTask<String, String, String> {

        @Override
        protected String doInBackground(String... strings) {
            try {
                URL url = new URL("https://api.github.com/users");
                HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                connection.connect();
                InputStream stream = connection.getInputStream();
                BufferedReader reader = new BufferedReader(new
InputStreamReader(stream));
                StringBuffer buffer = new StringBuffer();
                String line;
                while((line = reader.readLine()) != null) {
                    buffer.append(line).append("\n");
                }
                return buffer.toString();
            }
            catch (IOException ex) {
                Toast.makeText(getApplicationContext(), ex.getMessage(),
Toast.LENGTH_SHORT).show();
            }
            return "";
        }
```

```
@Override
        protected void onPreExecute() {
            super.onPreExecute();
            btnFetchData.setEnabled(false);
        }

        @Override
        protected void onProgressUpdate(String... values) {
            super.onProgressUpdate(values);
            resultView.setText("Loading..." + values + "% done");
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            resultView.setText(s);
            btnFetchData.setEnabled(true);
        }
    }

}
```