CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

JSON stands for **JavaScript Object Notation**. It is a text-based data **interchange** format to maintain the structure of the data.

Characteristics of JSON:

- JSON stores all the data in an **array** so data transfer makes easier. That's why JSON is the best for **sharing** data of any size even audio, video, etc.
- Its syntax is very **easy** to use. Its syntax is very **small** and **light-weighted** that's the reason that it executes and responds in a **faster** way.
- JSON has a wide range of browser **support** compatibility with the operating systems, it doesn't require much effort to make it all browser compatible.
- On the server-side **parsing** the most important part is that developers want, if the parsing will be fast on the server side then the user can get a fast response, so in this case, JSON server-side parsing is the strong point compared to others.

The official media type for the JSON is application/json and to save those file **.json** extension. JSON was initially created by Douglas Crockford in the early 2000s, and first standardized in 2013 after that in 2007 JSON's latest standard was published.

Syntax:

```
{
  "Name": "KUET",
  "Estd": 1967,
  "age": 56,
  "address": {
    "buildingAddress": "KUET Campus",
    "city": "Khulna",
    "state": "Khulna Division",
    "postalCode": "9203"
  }
}
```

API: The acronym of API is **Application Programming Interface**. And Create simple API is a very simple and easy process to do. API is a set of code that allows data transmission between one software to another. Software that wants to access information, calls the API with data requirements. To access data from any other application or server, you just need to check if they provide API access or not.

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

Rapid API (https://rapidapi.com/) is a site where you can get free public APIs that you can use in your project. For today, we will create our own API using JSON data.

- First **create** your own **JSON data** following the right syntax. (Create a JSON array of 3 students with **student_id, student_name and student_age)**
- You can validate your data in https://jsonlint.com/ to check if there's any syntax error or not.
- Then you need to host your data https://myjson.online/ here or any free JSON host site which will generate the **API URL** that you need to use from your application.
- With the API URL, the data will look like below. Inside the JSON Object "**data**", there is a JSON array of 3 students with three attributes.

Android

JSON in Android is used to **exchange** data in Android from the server. Parsing JSON Data in Android is a simple and easy process to implement. Parsing JSON Data in Android is done with **some classes** provided by Android Library.

Android supports different kinds of classes to use JSON efficiently. which are

- JSON Object
- JSON Array
- And JSON Stringer etc.

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

These methods are used to implement and get the required information from Android Applications. So, let's jump to the Android Studio and start Implementing Data Parsing in Android.

Step-1

Open Android Studio and create a new project. Then Select Activity of Android Application. Generally, select **Empty Views** Activity.

We will be using **Google Volley** for fetching data from the URL. (Google Volley is one of the best library use for making HTTP connections. It makes networking easy, fast and secure. Volley is a library built and maintained by Google)

So, now open the **Gradle** module file and add following the dependency.

```
dependencies {

    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
    implementation (libs.volley)
}
```

If the above line shows error in your Android Studio version, try to add:
`implementation("com.android.volley:volley:1.2.1")`

And make sure you **sync** the files after adding despondency.

Step-2

Then open the **AndroidMenifest.xml file**(app>manifest>AndroidMenifest.xml). And add user permission to **INTERNET**.

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

<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
```

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

Step-3

Now open actvity_main.xml(res>layout>activity_main.xml). and add **TextView** and **Button**.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/text_view_results"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@android:color/black"
        android:textSize="20sp" />

    <Button
        android:id="@+id/button_parse"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:layout_marginTop="100dp"
        android:text="Parse Data"
        tools:ignore="HardcodedText" />
</LinearLayout>
```

Step-4

Open MainActivity.java and then Declare two global variables for **TextView** and **Button** before **onCreate** method. **Assign** TextView and Button from activity_main.xml to MainActivity.java.

In **Volley**, the **RequestQueue** is a core component that handles and manages all network requests. It acts as a **queue** where all HTTP requests are placed and processed efficiently. So, we need to decalare a global variable for **RequestQueue** as well and create Instance of it.

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

```
public class MainActivity extends AppCompatActivity {

    TextView textview;
    Button button;
    RequestQueue requestQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);

        textview = findViewById(R.id.text_view_results);
        button = findViewById(R.id.button_parse);
        requestQueue = Volley.newRequestQueue( context: this);

    }
}
```

Step-5

Set **onClickerListener** to the button. and create new method as **jsonParse()** after onCreate() method.

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        jsonParse();
    }
});
```

Inside the **jsonParse()** function:

- Create a String for the **URL** to store. copy the URL and paste the URL at String. Replace the url to yours.

```
String url = "https://api.myjson.online/v1/records/e58dca13-7b19-4d05-955d-7335c1def9f5";
```

- Create an object for **JSON Object** requests. This method will take 5 parameters.
    - Method of Request
    - URL String
    - Listener: Generally, make it NULL
    - Response Listener
    - Response Error Listener

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

To get information of error if occurred then write error.printStackTrace().

```
JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url,
        jsonRequest: null, new Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {

    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        error.printStackTrace();
    }
});
```

- Now, remember that square brackets [ ] show the array and curly brackets { } show the object when the **API URL** opened. Here in our JSON we have an array named "**data**". So let's Initialize JSONArray by fetching the array from the response. We have to surround it with a **try catch block** to handle exceptions.

- Create **JSON Object** instance and wrap it with **For loop**. and also initialize the content of the JSON file. Here **jsonArray.length()** gets the length of the array. To get the JSON object from the array students, we use **jsonArray.getJSONObject(i)**.

- Now the variable **i** keeps on changing till the array length and we will get all the objects likewise. To get the specific item from the object we use **jsonObject.get()** here we used **student** as the object name. So it will be stu.get('studentNum'). There also getString(), getInt() to match with the data type.

```
@Override
public void onResponse(JSONObject response) {
    try {
        JSONArray jsonArray = response.getJSONArray( name: "data");
        for(int i=0;i<jsonArray.length();i++){
            JSONObject student = jsonArray.getJSONObject(i);
            int id = student.getInt( name: "student_id");
            int age = student.getInt( name: "student_age");
            String name = student.getString( name: "student_name");
            textview.append(name + ", " + String.valueOf(id) + ", " + String.valueOf(age) + "\n\n");
        }
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }
}
```

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

Step-6

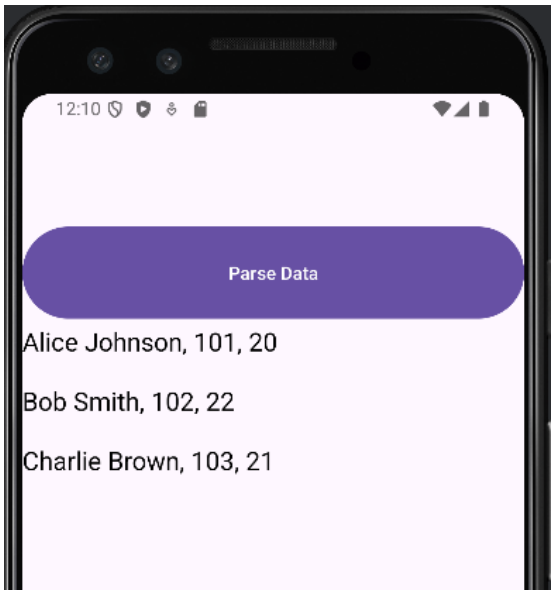
Append Data to **TextView**. You could also use ListView, RecyclerView, etc to give a better UI for the app. For this example, we are using TextView.

```
textview.append(name + ", " + String.valueOf(id) + ", " + String.valueOf(age) + "\n\n");
```

Add the current request to your requestQueue.

```
requestQueue.add(request);
```

And we are done **Passing JSON Data** in the Android project is done, Now, **run** your Application.



CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

Desktop

JSON parsing in Desktop applications is a more or less similar process. We will do the same as we did for Android.

Step-1

Create an empty **maven** project with a **button** and textview (**label** in this case). So as the button is pressed the parsed data will appear in the label section. We will implement the JSON parsing task inside the Button Click function.

Step-2

To parse the JSON data from the response body, you can use a JSON parsing library like **Jackson**, **Gson**, or **org.json**. **Gson** is lightweight and easy to use for parsing JSON. To use Gson, first we need to add the necessary dependency to our project. You add the dependencies in your project's **build configuration file**. The type of file depends on the build tool you are using, typically **Maven** or **Gradle**.

As, we are using Maven, add the dependency to your project's **pom.xml** file inside the <dependencies> tag.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
<junit.version>5.10.2</junit.version>  </properties>

  <dependencies>
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.10</version>
    </dependency>
```

Once added, Maven will automatically **download** the required library when you run mvn clean install or **refresh/sync** your project in your IDE (like IntelliJ IDEA or Eclipse).

Step-3 (Inside Button Click Function)

Now, we need to fetch data from a specified API URL. For that we will use Java's `HttpClient`. The whole process is as follows:

- Define a `String` variable named `url` that contains the API endpoint you want to call. Replace the url to yours.

```
String url = "https://api.myjson.online/v1/records/e58dca13-7b19-4d05-955d-7335c1def9f5";
```

- Create a new instance of the `HttpClient` class. `HttpClient` is used to send HTTP requests and receive responses. It supports both synchronous and asynchronous calls and is part of the Java 11+ standard library.

```
HttpClient client = HttpClient.newHttpClient();
```

- Build an HTTP Request to fetch data from the specified url.

```
HttpRequest request = HttpRequest.newBuilder()
        .uri(URI.create(url))
        .GET()
        .build();
```

`HttpRequest.newBuilder()`: Creates a new HttpRequest builder object that will help construct the HTTP request.
`.uri(URI.create(url))`: Specifies the target URI (in this case, the url defined earlier) for the request. The URI.create(url) method converts the string URL into a URI object.
`.GET()`: Specifies that the request is an HTTP GET request. A GET request is used to retrieve data from a server.
`.build()`: Finalizes the configuration and creates an immutable HttpRequest object.

- Now, send the HTTP request. The response is captured in the variable response as a HttpResponse object.

```
HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
```

`HttpResponse.BodyHandlers.ofString()` tells the HttpClient to handle the response body as a String. The server's **response**, which may include status codes, headers, and the body content (usually JSON for APIs), is stored in the response object.

- Finally, Surround the code section with **try-catch block**. You can use **response.body()** and send it your own **jsonParse() function** to parse and use the data.

After step-3, the function body of the button click will look like this:

CSE 2200: Advanced Programming
Lab Tutorial: JSON Parsing

```java
public class HelloController {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onHelloButtonClick() {
        String url = "https://api.myjson.online/v1/records/e58dca13-7b19-4d05-955d-7335c1def9f5";
        try {
            HttpClient client = HttpClient.newHttpClient();
            HttpRequest request = HttpRequest.newBuilder()
                    .uri(URI.create(url))
                    .GET()
                    .build();
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
            jsonParse(response.body());

        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
    public void jsonParse(String response) {  1 usage

    }

}
```

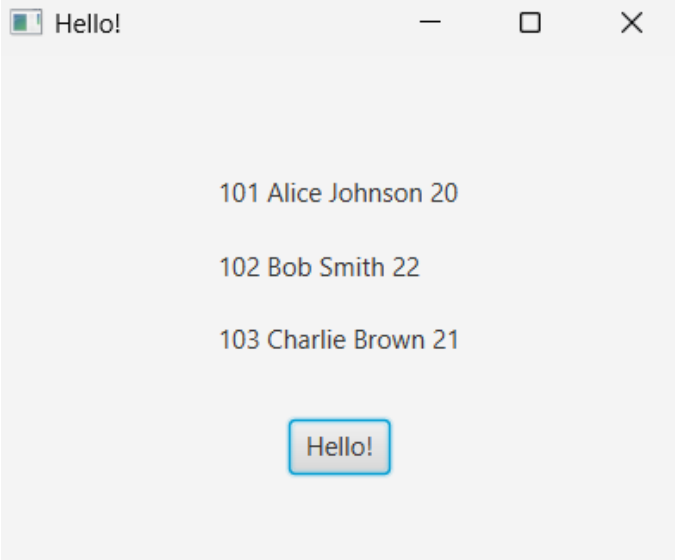
Step-4 (Inside jsonParse Function)

The process to parse the response is same as we did in Android with slight syntax changes.

```java
public void jsonParse(String response) {  1 usage

    JsonObject jsonObject = JsonParser.parseString(response).getAsJsonObject();
    JsonArray jsonArray = jsonObject.get("data").getAsJsonArray();
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < jsonArray.size(); i++) {
        JsonObject student = jsonArray.get(i).getAsJsonObject();
        int id  = student.get("student_id").getAsInt();
        String name  = student.get("student_name").getAsString();
        int age = student.get("student_age").getAsInt();
        stringBuilder.append(id).append(" ").append(name).append(" ").append(age).append("\n\n");
    }
    welcomeText.setText(stringBuilder.toString());
}
```

After writing the function body, the process will be complete. Try to run the application and check the output. It will be like this below:



Finally, you have completed JSON parsing for both Android and Desktop Applications successfully. Thank you 😊