# Mobile Application Development

(Restful API)

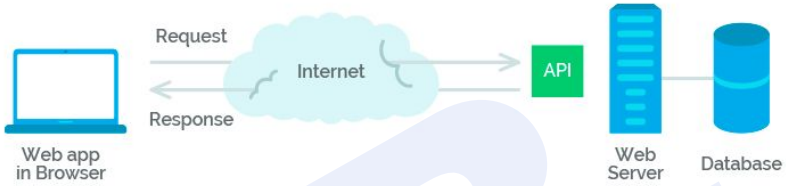
Instructor: Thanh Binh Nguyen

February 1st, 2020

S3Lab
*Smart Software System Laboratory*

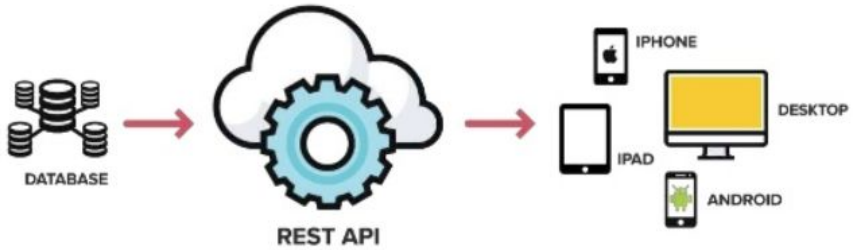

“The future of mobile is the future of online. It is how people access online content now.”
– David Murphy, Founder and Editor of Mobile Marketing Daily

# Restful API



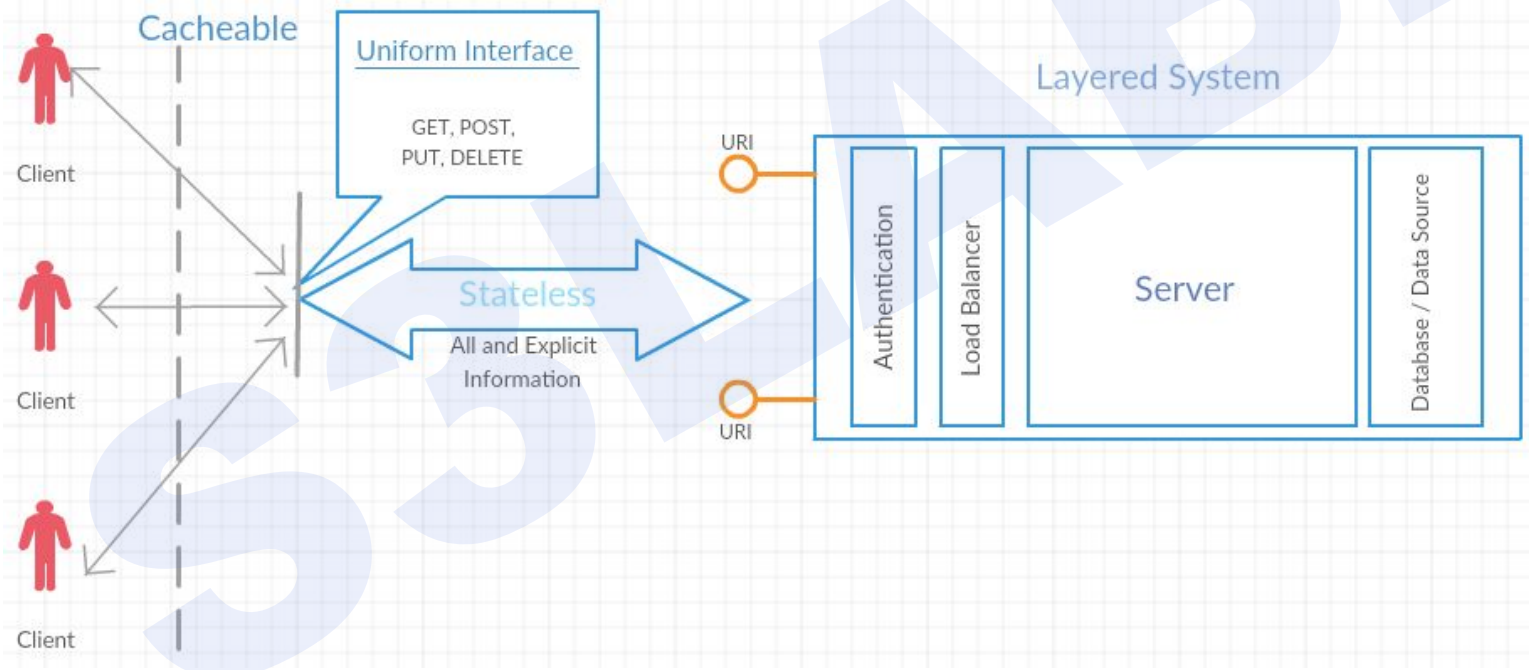

*Overview*

- **Web APIs**: use web protocols such as HTTP, HTTPs, JSON, XML, etc. For example, a web API can be used to obtain data from a resource (such as U.S. postal service zip codes) without having to actually visit the application itself (checking usps.com).
- **REST**: is a short for **R**epresentational **S**tate **T**ransfer.
  - An architectural style for distributed hypermedia systems.
  - A set of rules and conventions for the creation of an API.
  - Was first presented by Roy Fielding in 2000 in his famous dissertation.

# Restful API

*6 Constraints*

# Restful API

*6 Constraints - 1. Client-Server architecture*

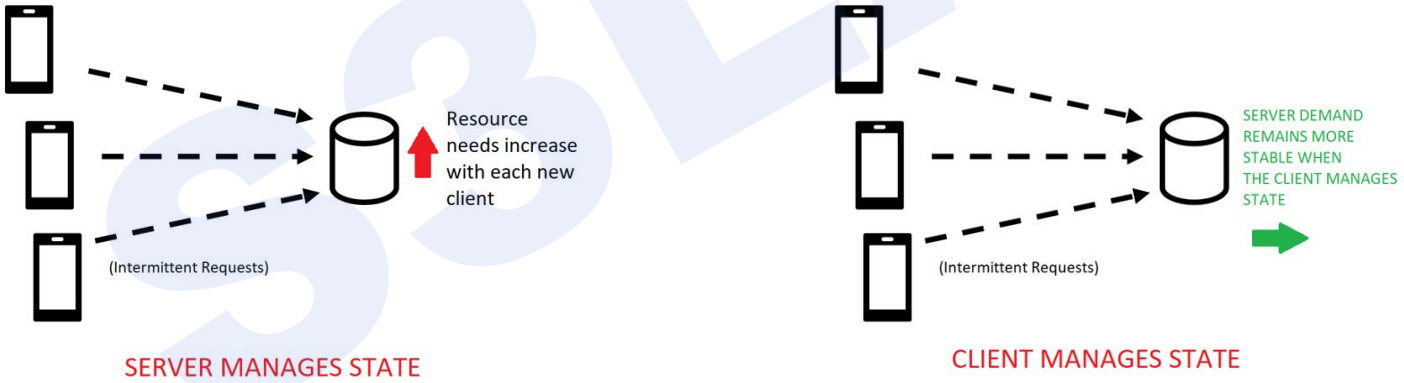
- Separate the systems responsible for storing and processing the data (**the server**) from the systems responsible for collecting, requesting, consuming, and presenting the data to a user (**the client**).
- This separation should be so distinct that the client and server systems can be **improved** and **updated independently** each other.

# Restful API

*6 Constraints - 2. Statelessness*

- As far as the server is concerned, all client requests are treated equally. There's no special, server-side memory of past client activity. The responsibility of managing state (for example, logged in or not) is on the client. This constraint is what makes the **REST**ful approach so scalable.

***the state gets transferred with each request***



# Restful API

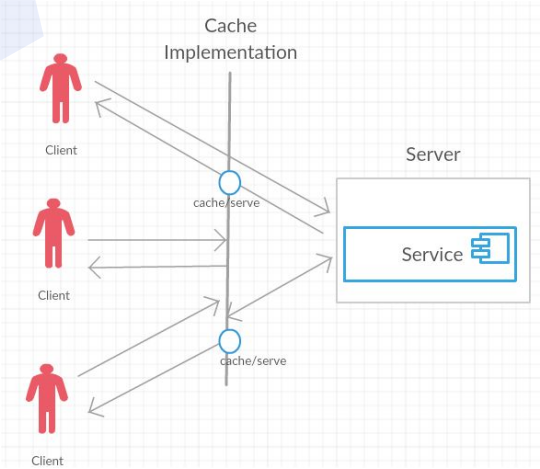
*6 Constraints - 3. Cacheability*

- **Clients** and **servers** should be able to cache resource data that changes infrequently.
- **Example**: there are 52 states and other jurisdictions in the U.S.A.  That's not likely to change soon.  So, it is inefficient to build a system that queries a database of states each and every time you need that data. Clients should be able to cache that infrequently updated date and web servers should be able to control the duration of that cache.

**Browser caches**

**Proxy caches**

**Gateway caches**
**(reverse-proxy)**



# Restful API

*6 Constraints - 3. Cacheability*

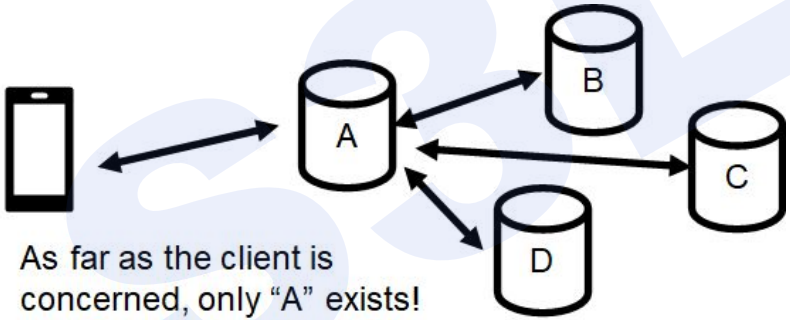
**Several ways that we can control the cache behavior**

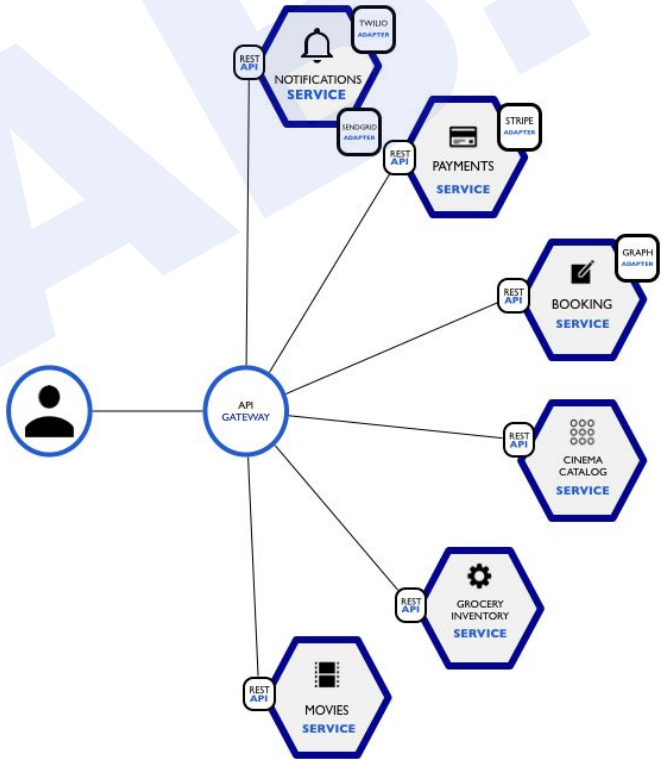
| Headers | Description | Samples |
|---|---|---|
| Expires | Header attribute to represent date/time after which the response is considered stale | Expires: Fri, 12 Jan 2018 18:00:09 GMT |
| Cache-control | A header that defines various directives (for both requests and responses) that are followed by caching mechanisms | `Max age=4500` , cache-extension |
| E-Tag | Unique identifier for server resource states | ETag: `uqv2309u324klm` |
| Last-modified | Response header helps to identify the time the response was generated | Last-modified: Fri, 12 Jan 2018 18:00:09 GMT |

# Restful API

*6 Constraints - 4. Layered System*

- A client cannot tell whether it is connected directly to an end server, or to an intermediary along the way.
- Intermediary servers can also improve system scalability

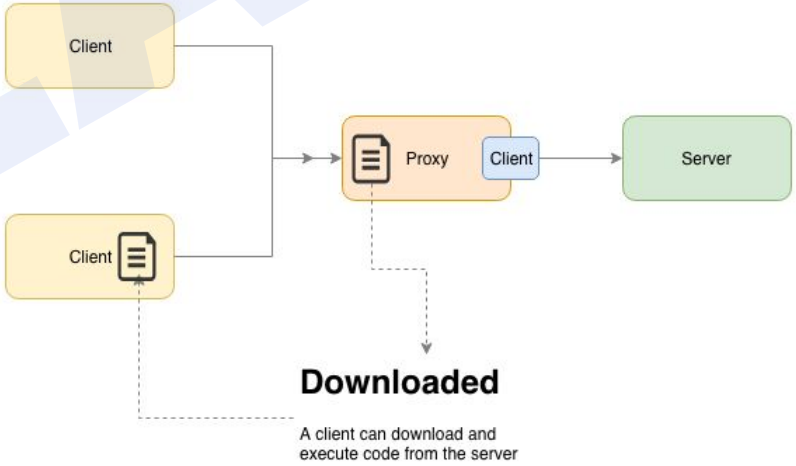


# Restful API

*6 Constraints - 5. Code on demand (Optional)*

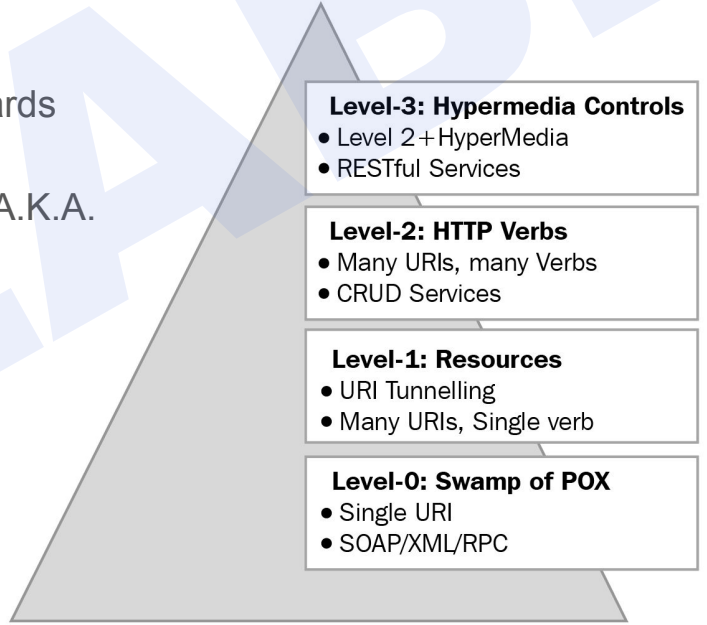
- Servers can temporarily extend or customize the functionality of a client by transferring executable code (scripts / applets).
- Allows server to decide how some things will be done.
- This is constraint is optional.

# Restful API

*6 Constraints - 6. Uniform Interface*

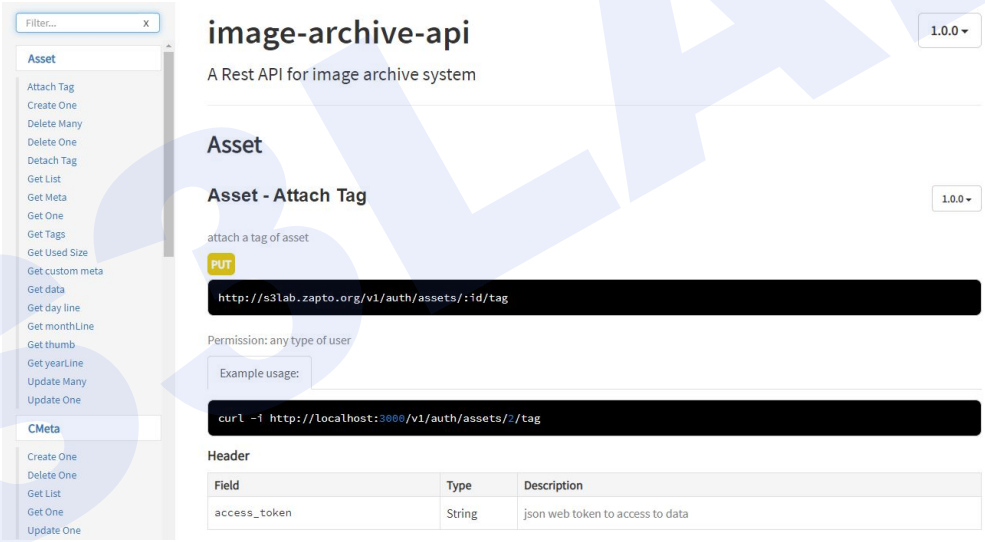
- Identification of resources (URL or IRI)
- Manipulation of resources through HTTP standards
- Self descriptive messages
- Hypermedia as the engine of application state (A.K.A. HATEOAS)



# Restful API

*API Documents*

- All of Restful API should be documented using Swagger or API Doc utilities
- Testing the Restful API using Postman



# Consume Restful API

*Traditional Way*

- Enable Internet Access:
  - <uses-permission android:name="android.permission.INTERNET" />
- Create Background Threads
  - AsyncTask.execute(new Runnable() {
  - @Override
  - public void run() {
  - // All your networking logic
  - // should be here
  - }
  - });

# Consume Restful API

*Traditional Way*

- Using HttpURLConnection

  new HTTPReqTask().execute();

```
private static class HTTPReqTask extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        HttpURLConnection urlConnection = null;

        try {
            URL url = new URL("https://reqres.in/api/users?page=2");
            urlConnection = (HttpURLConnection) url.openConnection();

            int code = urlConnection.getResponseCode();
            if (code !=  200) {
                throw new IOException("Invalid response from server: " + code);
            }

            BufferedReader rd = new BufferedReader(new InputStreamReader(
                    urlConnection.getInputStream()));
            String line;
            while ((line = rd.readLine()) != null) {
                Log.i("data", line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }

        return null;
    }
}
```

# Consume Restful API

*Traditional Way*

- Using HttpURLConnection

  new HTTPReqTask().execute();

  implementation
  ‘com.google.code.gson:gson:2.8.5’

```
private static class HTTPReqTask extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        HttpURLConnection urlConnection = null;

        try {
            JsonObject postData = new JsonObject();
            postData.addProperty("name", "morpheus");
            postData.addProperty("job", "leader");

            URL url = new URL("https://reqres.in/api/users");
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestProperty("Content-Type", "application/json");
            urlConnection.setRequestMethod("POST");
            urlConnection.setDoOutput(true);
            urlConnection.setDoInput(true);
            urlConnection.setChunkedStreamingMode(0);

            OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(
                    out, "UTF-8"));
            writer.write(postData.toString());
            writer.flush();

            int code = urlConnection.getResponseCode();
            if (code !=  201) {
                throw new IOException("Invalid response from server: " + code);
            }

            BufferedReader rd = new BufferedReader(new InputStreamReader(
                    urlConnection.getInputStream()));
            String line;
            while ((line = rd.readLine()) != null) {
                Log.i("data", line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }

        return null;
    }
}
```

# Consume Restful API

*Traditional Way*

- Receive and parse the content with JSON
  - InputStream responseBody = myConnection.getInputStream();
  - InputStreamReader responseBodyReader =
    new InputStreamReader(responseBody, "UTF-8");
  - JsonReader jsonReader = new JsonReader(responseBodyReader);
  - jsonReader.close();
  - myConnection.disconnect();

```
01  jsonReader.beginObject(); // Start processing the JSON object
02  while (jsonReader.hasNext()) { // Loop through all keys
03      String key = jsonReader.nextName(); // Fetch the next key
04      if (key.equals("organization_url")) { // Check if desired key
05          // Fetch the value as a String
06          String value = jsonReader.nextString();
07
08          // Do something with the value
09          // ...
10
11          break; // Break out of the loop
12      } else {
13          jsonReader.skipValue(); // Skip values of other keys
14      }
15  }
```

# Consume Restful API

*Using Library*

- Volley: https://github.com/google/volley
- **Retrofit**: https://github.com/square/retrofit

# Consume Restful API

*With Retrofit 2*

**Adding Dependency in Build.gradle-**

```
dependencies {
    implementation 'com.squareup.retrofit2:retrofit:2.1.0'
    implementation 'com.google.code.gson:gson:2.6.2'
    implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
}
```

**Adding Internet Permission in Manifest-**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.kripashankar.countryappusingretrofit">
    <uses-permission android:name="android.permission.INTERNET" />
```

# Consume Restful API

*With Retrofit 2 - Retrofit Instance*

```
public class RestAPIServiceBuilder {

    private static OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout( timeout: 30, TimeUnit.SECONDS)
            .readTimeout( timeout: 30, TimeUnit.SECONDS)
            .writeTimeout( timeout: 15, TimeUnit.SECONDS)
            .build();

    public static Retrofit CreatRetrofitForAPI(){
        return new Retrofit.Builder()
                .baseUrl(DefSetting.sRestServerUrl)
                .client(okHttpClient)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }

    public static Retrofit CreatRetrofitForDownload(){
        return new Retrofit.Builder()
                .baseUrl(DefSetting.sRestServerUrl)
                .client(okHttpClient)
                .build();
    }
}
```

# Consume Restful API

*With Retrofit 2 - Retrofit Interface*

```
public interface RestAPIService {
    @Streaming
    @GET
    Call<ResponseBody> downloadFileWithDynamicUrlAsync(@Url String fileUrl);

    @POST("login")
    Call<User> login(@Body User user);
    @POST("login_google")
    Call<User> loginGoogle(@Body User user);

    @GET("auth/tags")
    Call<CatalogList> getTags(@Header("x-access-token") String token, @Query("filter") String sFilter);

    @GET("auth/books")
    Call<BookList> getBooks(@Header("x-access-token") String token, @Query("filter") String sFilter, @Query("q") String q,
                                                                    @Query("page") int page, @Query("perPage") int perPage);
    @PUT("auth/books/{id}")
    Call<Book> updateBooks(@Header("x-access-token") String token, @Path("id") String sId, @Body Book book);

    @GET("auth/audio_parts")
    Call<AudioPartList> getBookParts(@Header("x-access-token") String token, @Query("filter") String sFilter, @Query("sort") String sSort,
                                     @Query("page") int page, @Query("perPage") int perPage);
    @PUT("auth/audio_parts/{id}")
    Call<AudioPart> updateAudioParts(@Header("x-access-token") String token, @Path("id") String sId, @Body AudioPart audio);

    @POST("auth/messages/send_system")
    Call<BaseResponse> sendtosystem(@Header("x-access-token") String token, @Body Message message);

    @POST("auth/user_books")
    Call<UserBook> modifyUserBook(@Header("x-access-token") String token, @Body UserBook oUserBook);
    @GET("auth/user_books/creator_book")
    Call<UserBook> getByCreatorAndBook(@Header("x-access-token") String token, @Query("creator") String sCreator, @Query("book") String sBook);
}
```

# Consume Restful API

*With Retrofit 2 - Retrofit Model Class*

```java
public class Book {
    @SerializedName("id")
    public String sId;
    @SerializedName("name")
    public String sName;
    @SerializedName("partCount")
    public int iPartCount;

    @SerializedName("author")
    public String sAuthor;
    @SerializedName("translator")
    public String sTranslator;
    @SerializedName("isRecommend")
    public boolean bIsRecommend;
    @SerializedName("status")
    public String sStatus;

    @SerializedName("viewCount")
    public int iViewCount;
    @SerializedName("likeCount")
    public int iLikeCount;

    public Book(String id) { this.sId = id; }
    public Book(String sId, String sName, int iPartCount, String sAuthor, String sTranslator, boolean bIsRecommend, String sStatus, int iViewCount, int iLikeCount)
    {
        this.sId = sId;
        this.sName = sName;
        this.iPartCount = iPartCount;
        this.sAuthor = sAuthor;
        this.sTranslator = sTranslator;
        this.bIsRecommend = bIsRecommend;
        this.sStatus = sStatus;
        this.iViewCount = iViewCount;
        this.iLikeCount = iLikeCount;
    }
}
```

# Consume Restful API

*With Retrofit 2 - Using*

```
public class PostMessageToCloudService extends AsyncTask<Void, Integer, Boolean > {

    private String sToken;
    private String sContent;
    private String sExtraInfo;

    Message oMessage;

    public OnTaskCompleted oITaskCompleted;

    public interface OnTaskCompleted {
        void onTaskCompleted(boolean bResult);
    }

    public PostMessageToCloudService(String sToken, String sContent, String sExtraInfo, String sAction)
    {
        oMessage = new Message(sAction);
        this.sToken = sToken;
        this.sContent = sContent;
        this.sExtraInfo = sExtraInfo;
    }

    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected void onProgressUpdate(Integer... values) { super.onProgressUpdate(values); }
```

# Consume Restful API

*With Retrofit 2 - Using*

```
@Override
protected void onPostExecute(Boolean result) {
    super.onPostExecute(result);
    oITaskCompleted.onTaskCompleted(result);
}

@Override
protected Boolean doInBackground(Void... params)
{
    try {
        //RestAPIService gitHubService = RestAPIServiceBuilder.retrofit.create(RestAPIService.class);
        RestAPIService gitHubService = RestAPIServiceBuilder.CreatRetrofitForAPI().create(RestAPIService.class);
        oMessage.content += sContent;
        oMessage.extraInfo += sExtraInfo;
        Call<BaseResponse> call = gitHubService.sendtosystem(sToken, oMessage);
        BaseResponse oBaseResponse = call.execute().body();

        if(oBaseResponse != null && oBaseResponse.id != null){
            return true;
        }
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }catch (IOException e){
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
}
```

# Homeworks

- Making an simple application to connect to given API to get data, put the data on UI such as recyclerview, textview, grid, ... and try to support the specific technology like paginage.
  - https://codelabs.developers.google.com/codelabs/android-training-create-recycler-view/#0
- API free:
  - https://gorest.co.in/
  - https://apipheny.io/free-api/
- Retrofit:
  - https://www.section.io/engineering-education/making-api-requests-using-retrofit-android/
  - https://www.journaldev.com/13639/retrofit-android-example-tutorial
  - https://howtodoandroid.com/retrofit-android-example/

# Q & A

## Thank you for listening

*“Coming together is a beginning;*
*Keeping together is progress;*
*Working together is success.”*
- HENRY FORD