INTRODUCTION TO RETROFIT 2

A **type-safe** HTTP client for Android and Java

Why to use an HTTP client?

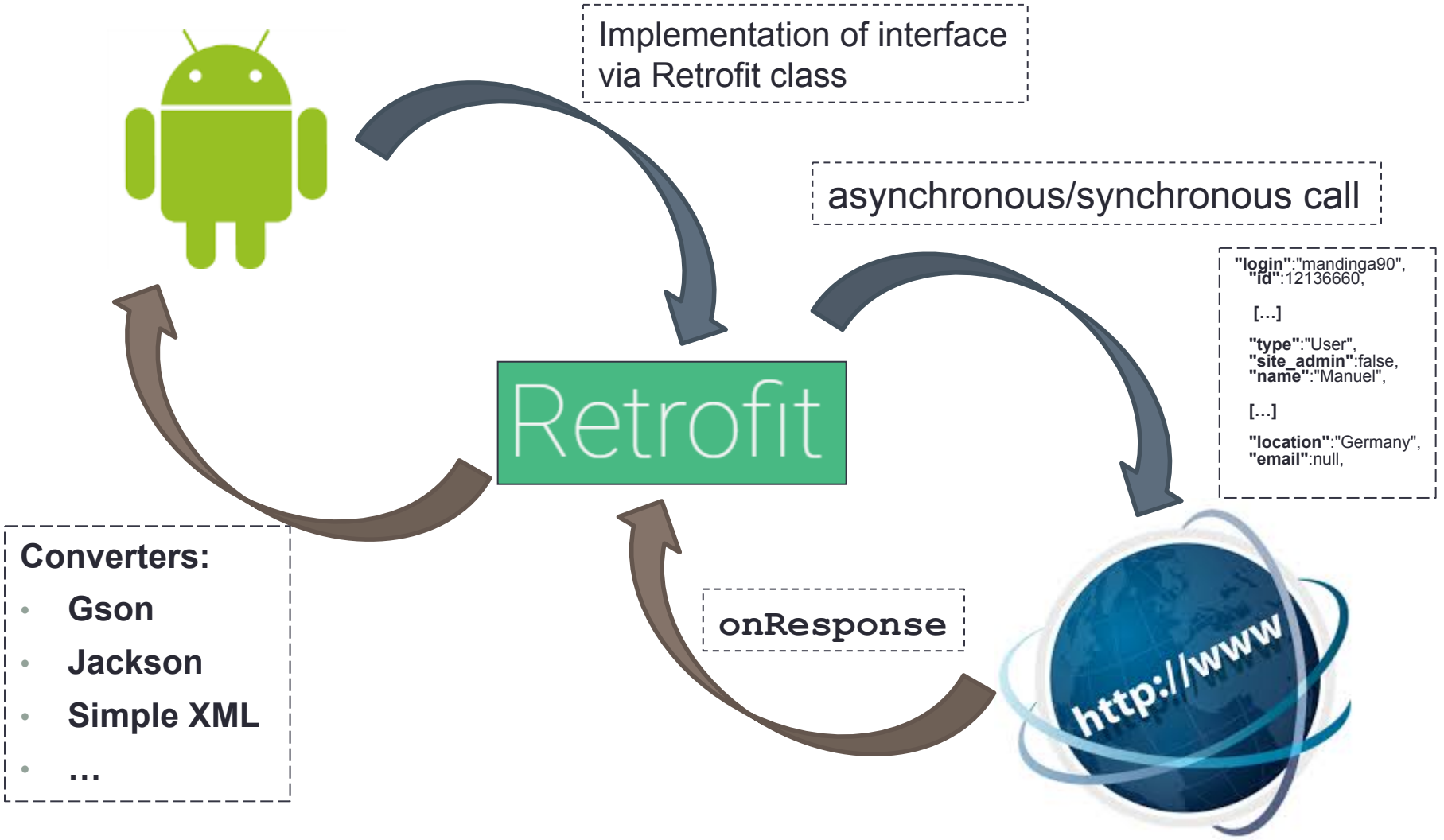
- **`HttpURLConnection`**

```
URL url = new URL("http://www.android.com/");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try {
  InputStream in = new BufferedInputStream(urlConnection.getInputStream());
  readStream(in);
} finally {
  urlConnection.disconnect();
}
```

- Many actions done automatically by client

➔ simplier & less work!

How does the client work?

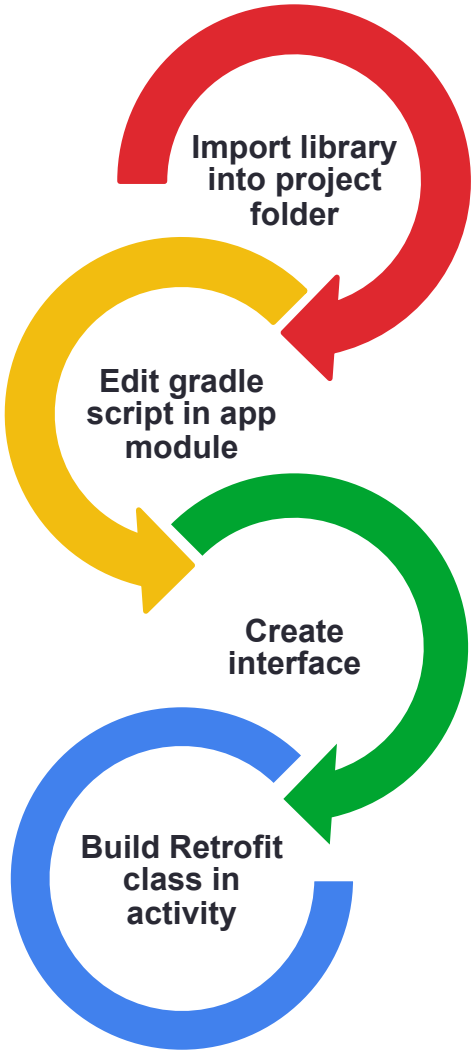


How to use it?

- square.github.io/retrofit/
- **compile** **'com.squareup.retrofit2:retrofit:*(insert latest version)*'** in gradle script
- ```
  public interface GithubAPI {

      @GET("/users/{user}")
      Call<GithubUser> getUser(@Path("user")
         String user);

  }
  ```
- ```
  Retrofit retrofit = new
  Retrofit.Builder() .baseUrl("https://
  api.github.com/") .build();
  ```



Retrofit API Declaration

- *@Annotations* → request handling
  - GET, POST, PUT, DELETE, and HEAD

```
@GET("/users/{user}")
Call<GithubUser> getUser(@Path("user") String user);
```

RESTful API – example: GitHub


```
"login":"mandinga90",
  "id":12136660,

   […]

  "type":"User",
  "site_admin":false,
  "name":"Manuel",

  […]

  "location":"Germany",
  "email":null,
```

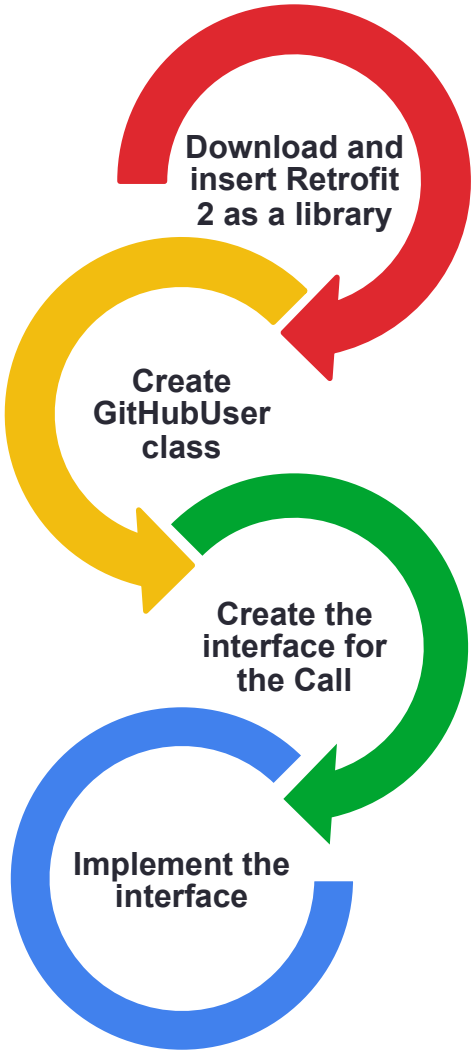

https://api.github.com/users/mandinga90

https://api.github.com/users/{user}

Task (1)

Receive user data from the GitHub API using Retrofit 2

- Download and insert Retrofit 2 as a library https://square.github.io/retrofit/
- Create a class GitHubUser with the members login, name and email (all String)
- Create an interface with a @GET request and offering a call to receive a GitHub user **Remember**: **https://api.github.com/users/{user}** **Hint:** Use @Path to declare a parameter (user) for the Call. See Retrofit website (see above).
- Implement the interface in a activity
  - Create a GsonBuilder before
  - Add a GsonConverterFactory to the Retrofit instance while building it
  - Use `create(Class<T> service)` to implement the interface



Task (2)

- Create a Call and use **`enqueue(Callback<T> callback)`** to start the service aysynchronously
  Hint: Your activity has to implement **`Callback<GitHubUser>`** to enqueue itself as the callback.
- Implement **`onResponse`** and **`onFailure`** to handle the callback (e. g. show it as a toast or within a textview)

Questions ???

Sources

- Android Developers
  developer.android.com
- Retrofit
  https://square.github.io/retrofit/
- Vogella - Using Retrofit 2.0 as REST client – Tutorial
  http://www.vogella.com/tutorials/Retrofit/article.html#exercise-using-retrofit-to-access-github-api-in-android