



android

LẬP TRÌNH ANDROID VỚI RESTAPI

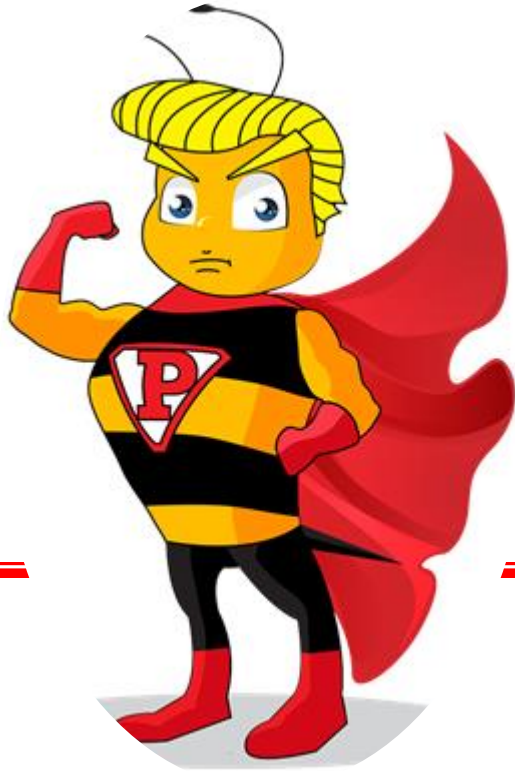
THREAD – THREAD POOL – VOLLEY – RETROFIT

- ☐ Tìm hiểu và sử dụng thread – thread pool
- ☐ Tìm hiểu và sử dụng volley
- ☐ Tìm hiểu và sử dụng retrofit

MỤC TIÊU

- ◎ TÌM HIỂU THREAD – THREAD POOL
- ◎ TÌM HIỂU VOLLEY
- ◎ TÌM HIỂU VÀ SỬ DỤNG RETROFIT

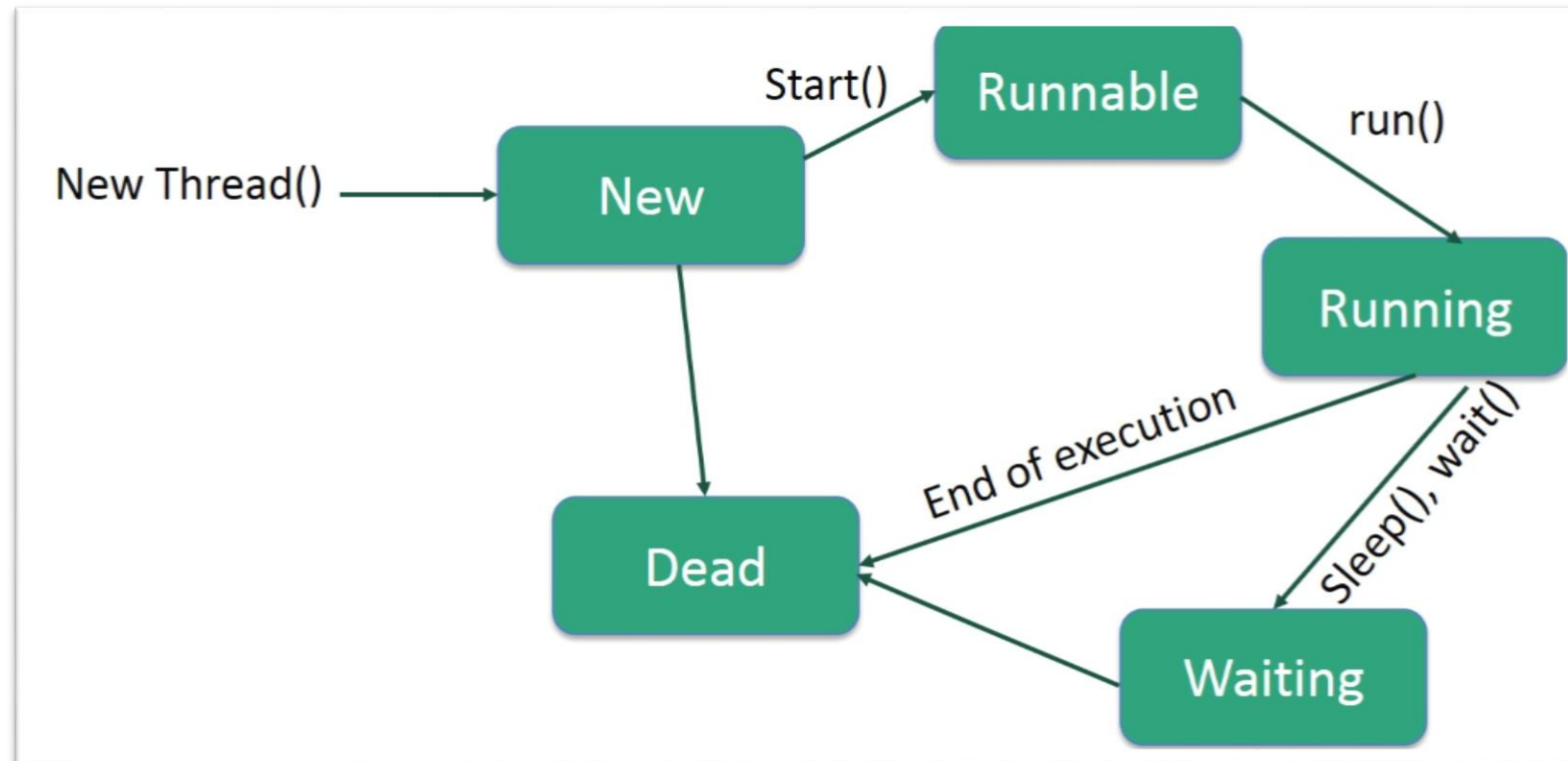




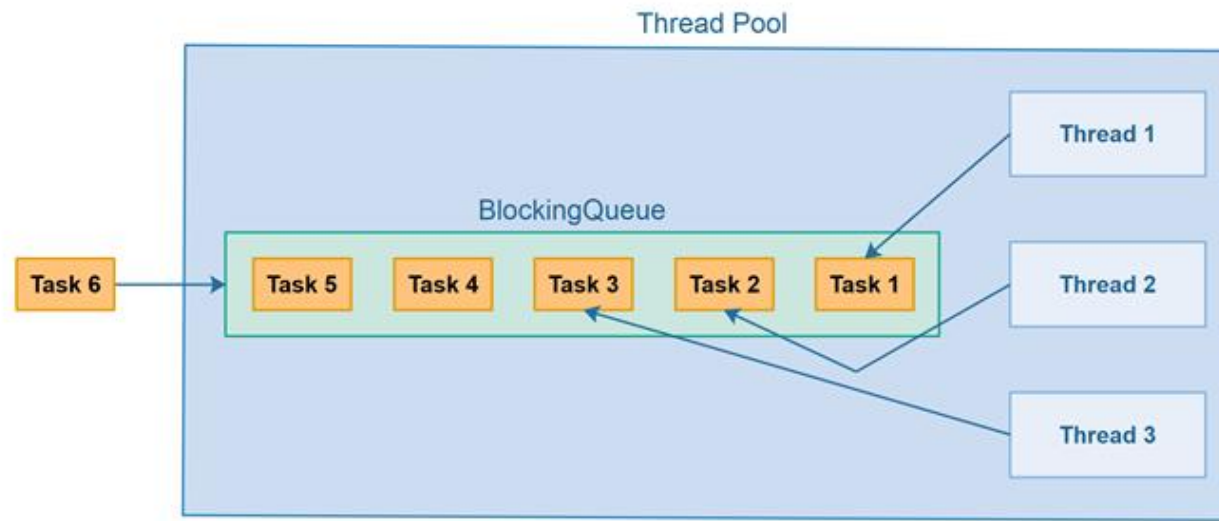
THREAD – THREAD POOL

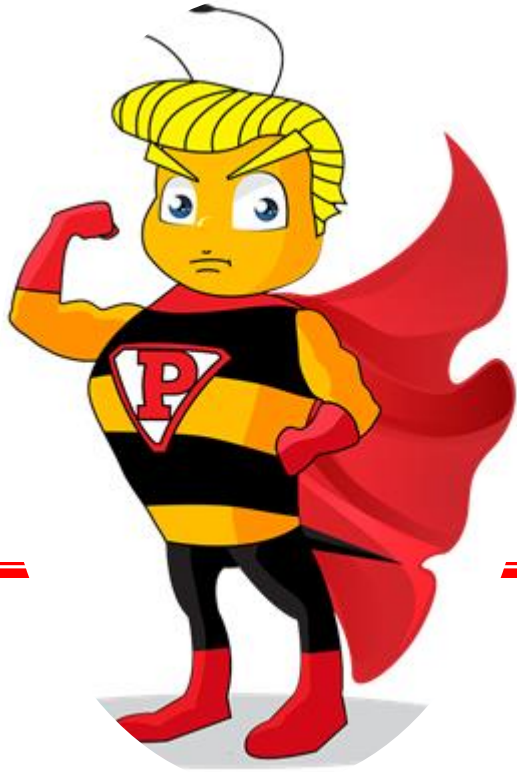
...

❑ **Thread** được định nghĩa là một luồng dùng để thực thi một chương trình. Java Virtual Machine cho phép một chương trình có thể có nhiều Thread thực thi đồng thời.



❑ **ThreadPool** được dùng để giới hạn số lượng được chạy bên trong ứng dụng của chúng ta trong cùng một thời điểm. Nếu chúng ta không có sự giới hạn này, mỗi khi có một mới được tạo ra và được cấp phát bộ nhớ bằng từ khóa new thì sẽ có vấn đề về bộ nhớ và hiệu suất, có thể dẫn đến lỗi crash chương trình.



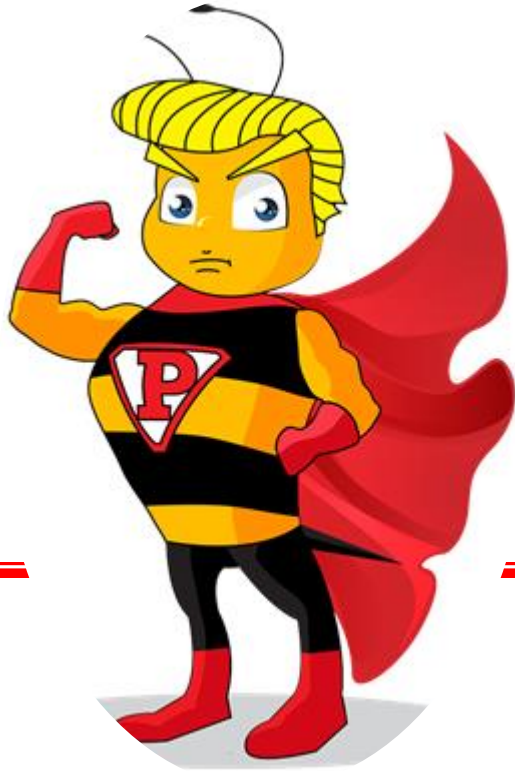


VOLLEY

...

- ❑ **Volley** library Android là thư viện networking cho các dự án Android. Volley được dùng để quản lý các network request, giúp cho developer đơn giản hóa việc thực hiện kết nối và xử lý kết quả trả về từ server. Volley hỗ trợ đầy đủ các HTTP request như GET, POST, PUT, DELETE

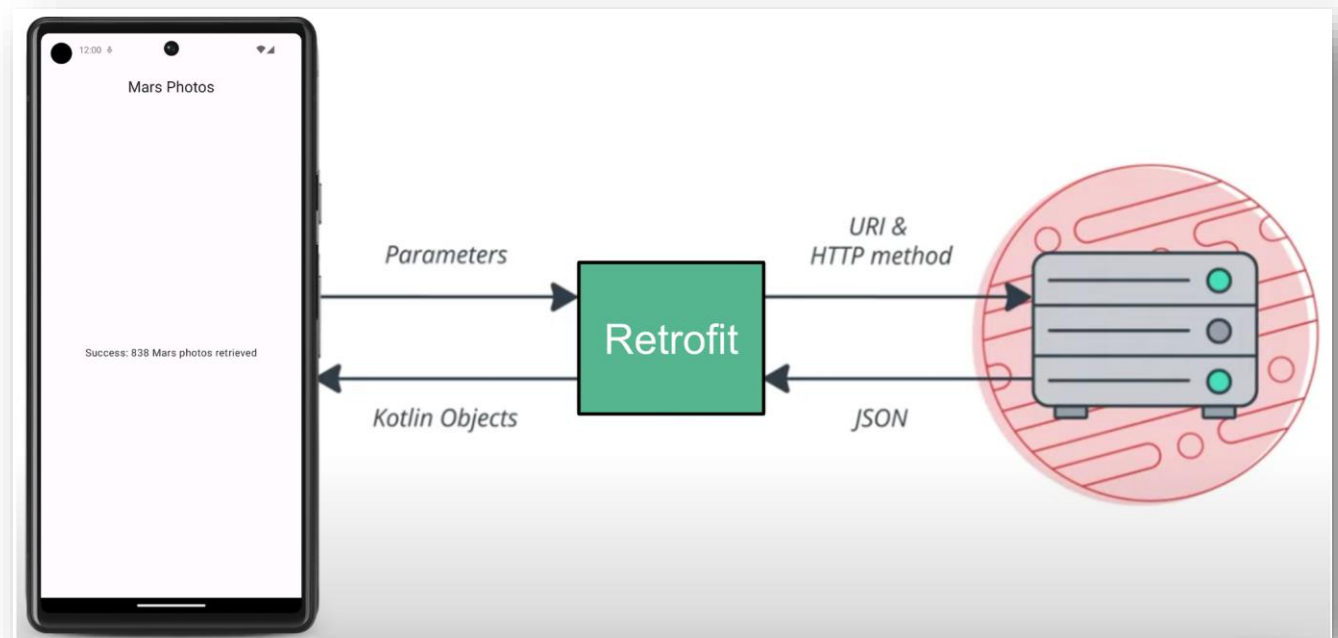




RETROFIT

...

❑ **Retrofit** là một HTTP client type-safe cho Android và Java. Retrofit giúp dễ dàng kết nối đến một dịch vụ REST trên web bằng cách chuyển đổi API thành Java Interface. Retrofit rất mạnh mẽ giúp bạn dễ dàng xử lý dữ liệu JSON hoặc XML sau đó phân tích cú pháp thành Plain Old Java Objects (POJOs)



Nội dung	Volley	Retrofit
Điểm mạnh	<ul style="list-style-type: none">- Cung cấp tính năng cache tự động, xử lý đa luồng linh hoạt	<ul style="list-style-type: none">- Cho phép tùy chỉnh dễ dàng- Hiệu suất tốt- Dễ dàng tìm hiểu
Điểm yếu	<ul style="list-style-type: none">- Khó tùy chỉnh- Hiệu suất trung bình- Ít tài liệu hướng dẫn	<ul style="list-style-type: none">- Không cung cấp cache tự động

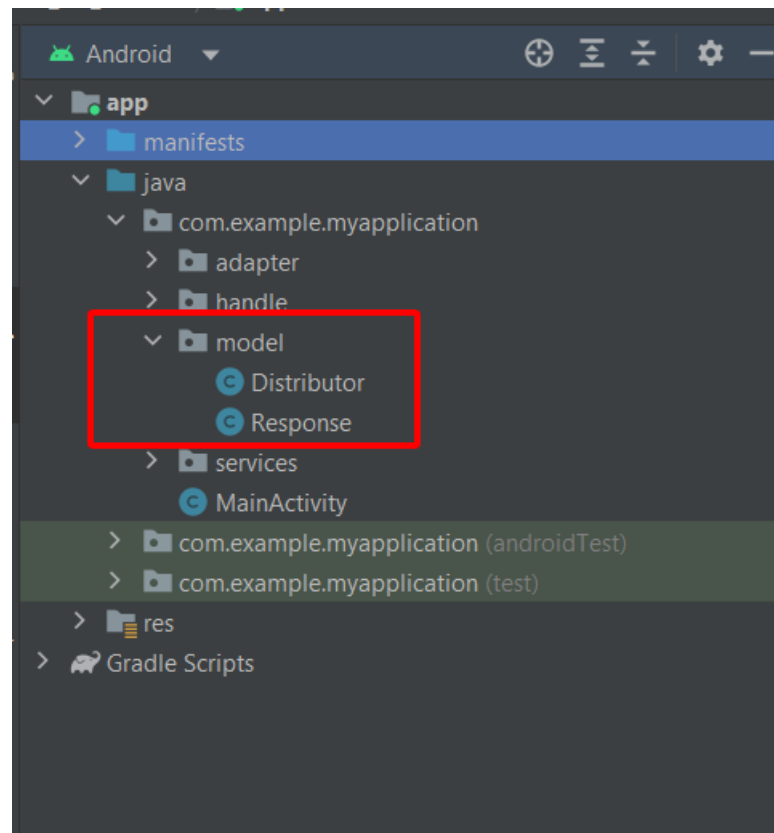
❑ Hiệu suất vượt trội của Retrofit

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

❑ Mở file api.js viết api lấy danh sách **distributor**

```
router.get(path: '/get-list-distributor',...handlers: async (req,res) => {
  try {
    //Lấy danh sách theo thứ tự distributors mới nhất
    const data = await Distributors.find().sort(arg: {createdAt: -1});
    if(data)
    {
      // Trả về danh sách
      res.json(body: {
        "status" : 200,
        "messenger" : "thành công",
        "data" : data
      })
    }
    else
    {
      // Nếu thêm không thành công result null, thông báo không thành công
      res.json(body: {
        "status" : 400 ,
        "messenger" : "Lỗi,không thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(message: error);
  }
});
```

- ❑ Tiếp theo tạo project trong **Android Studio**, tạo folder **model**, trong folder này tạo file **Response** và **Distributor**



Nội dung file **Response**

```
package com.example.myapplication.model;

17 usages
public class Response<T> {
    3 usages
    private int status;
    3 usages
    private String messenger;
    //T là kiểu Generic
    3 usages
    private T data;

    no usages
    public Response() {
    }

    no usages
    public Response(int status, String messenger, T data) {
        this.status = status;
        this.messenger = messenger;
        this.data = data;
    }
}
```

Nội dung file **Response** (*tiếp theo*)

```
2 usages
public int getStatus() { return status; }

no usages
public void setStatus(int status) { this.status = status; }

2 usages
public String getMessenger() { return messenger; }

no usages
public void setMessenger(String messenger) { this.messenger = messenger; }

1 usage
public T getData() { return data; }

no usages
public void setData(T data) { this.data = data; }
}
```


□ Nội dung file **Distributor**

```
package com.example.myapplication.model;

import com.google.gson.annotations.SerializedName;

30 usages
public class Distributor {

    //Có thể dùng Annotations của gson để đổi tên cho các trường nhận vào
    //Ví dụ trường _id nhận từ api, thay vì đặt tên trường trong object là _id
    //Có thể đặt là id và thêm vào Annotations @SerializedName("_id")
    3 usages
    @SerializedName("_id")
    private String id;
    3 usages
    private String name, createdAt, updatedAt;

    1 usage
    public Distributor() {
    }

    no usages
    public Distributor(String id, String name, String createdAt, String updatedAt) {
        this.id = id;
        this.name = name;
        this.createdAt = createdAt;
        this.updatedAt = updatedAt;
    }
}
```

❑ Nội dung file **Distributor** (tiếp theo)

```
2 usages
public String get_id() { return id; }

no usages
public void set_id(String id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

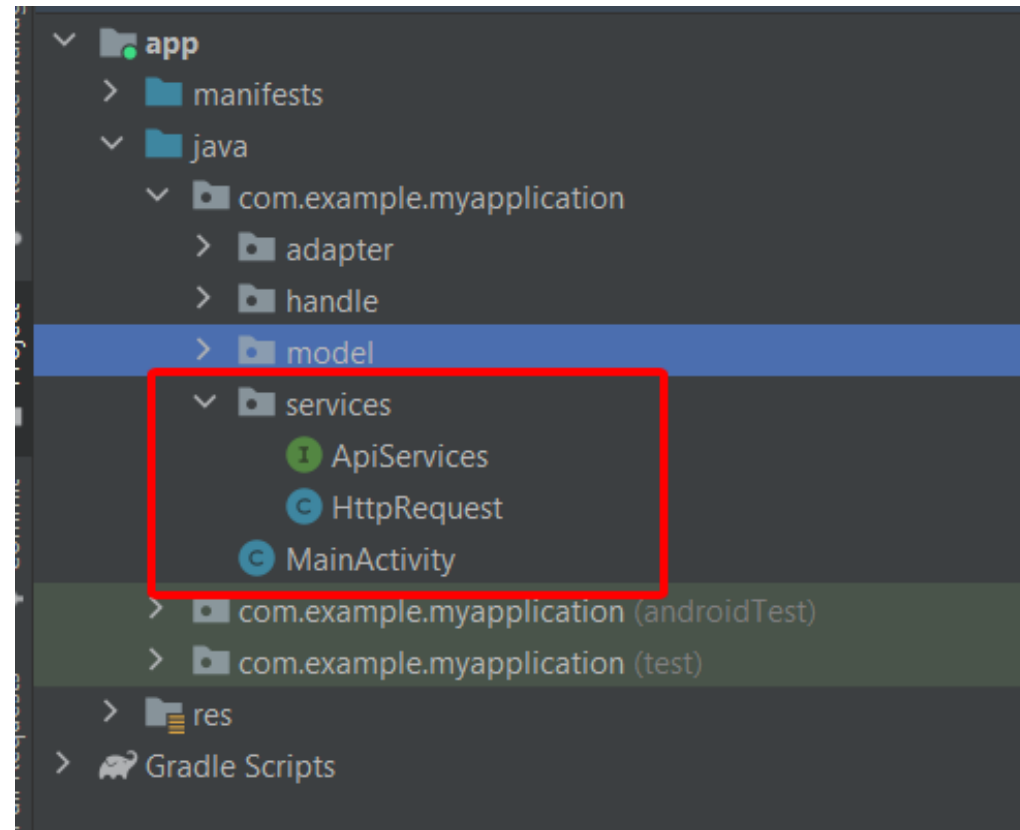
no usages
public String getCreatedAt() { return createdAt; }

no usages
public void setCreatedAt(String createdAt) { this.createdAt = createdAt; }

no usages
public String getUpdatedAt() { return updatedAt; }

no usages
public void setUpdatedAt(String updatedAt) { this.updatedAt = updatedAt; }
}
```

- Tiếp theo tạo folder **services**, trong folder này tạo 2 file **ApiServices** và **HttpRequest**



□ Nội dung file **ApiServices**

```
package com.example.myapplication.services;

import ...

4 usages
public interface ApiService {
    //Sử dụng máy ảo android studio thì localhost thay thành ip 10.0.0.2
    //Đối với máy thật ta sử dụng ip của máy
    //Base_URL là url của api
    2 usages
    public static String BASE_URL = "http://10.0.2.2:3000/api/";

    //Annotations @GET cho method GET và url phương gọi
    // Base_URL + @GET("get-list-distributor") = http://10.0.2.2:3000/api/get-list-distributor
    2 usages
    @GET("get-list-distributor")
    Call<Response<ArrayList<Distributor>>> getListDistributor();
    //Call<Giá trị trả về của api>

    1 usage
    @GET("search-distributor")
    Call<Response<ArrayList<Distributor>>> searchDistributor(@Query("key") String key);
```

❑ Nội dung file **ApiServices** (tiếp theo)

```
1 usage
@POST("add-distributor")
Call<Response<Distributor>> addDistributor(@Body Distributor distributor);

//Param url sẽ bỏ vào {}
1 usage
@DELETE("delete-distributor-by-id/{id}")
Call<Response<Distributor>> deleteDistributorById(@Path("id") String id);

1 usage
@PUT("update-distributor-by-id/{id}")
Call<Response<Distributor>> updateDistributorById(@Path("id") String id, @Body Distributor distributor);
}
```

□ Nội dung file **HttpRequest**

```
package com.example.myapplication.services;

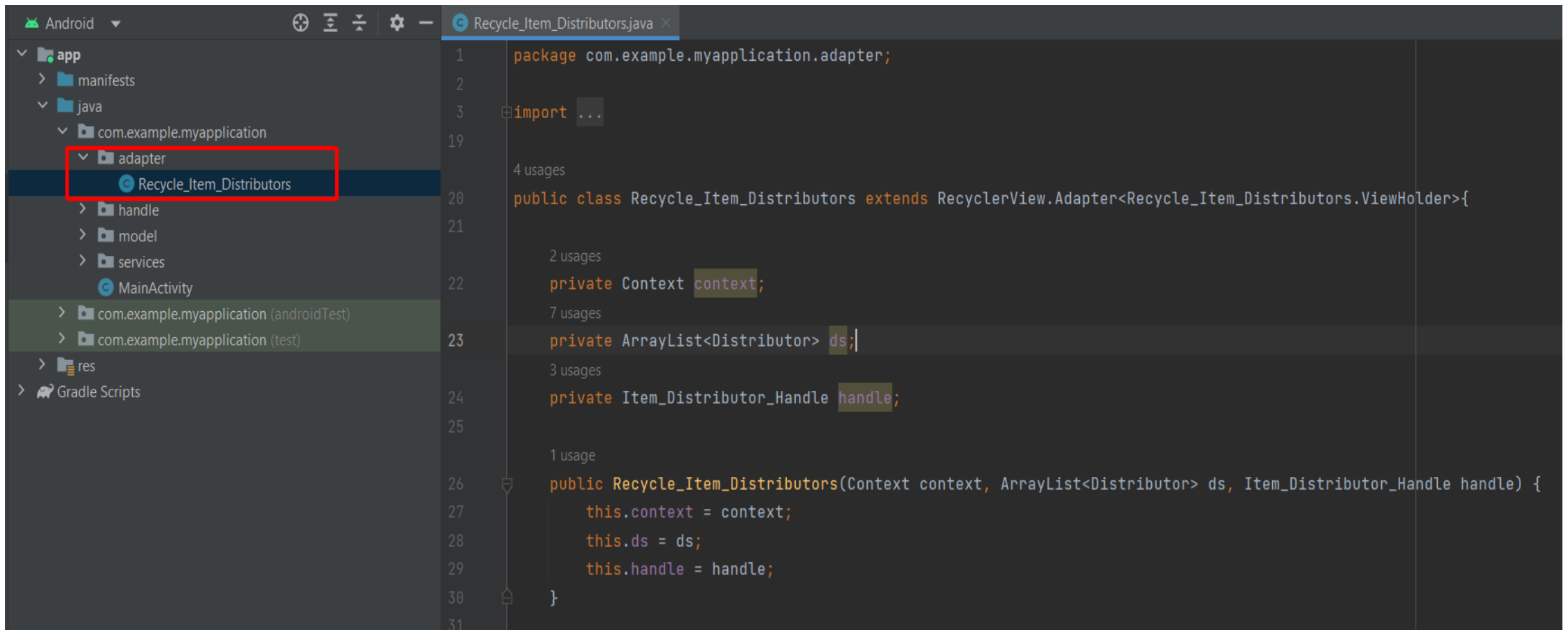
import ...

3 pages
public class HttpRequest {
    //Biến Interface ApiService
    2 usages
    private ApiService requestInterface;

    //Hàm tạo
    1 usage
    public HttpRequest(){
        //Create Retrofit
        requestInterface = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build().create(ApiServices.class);
    }

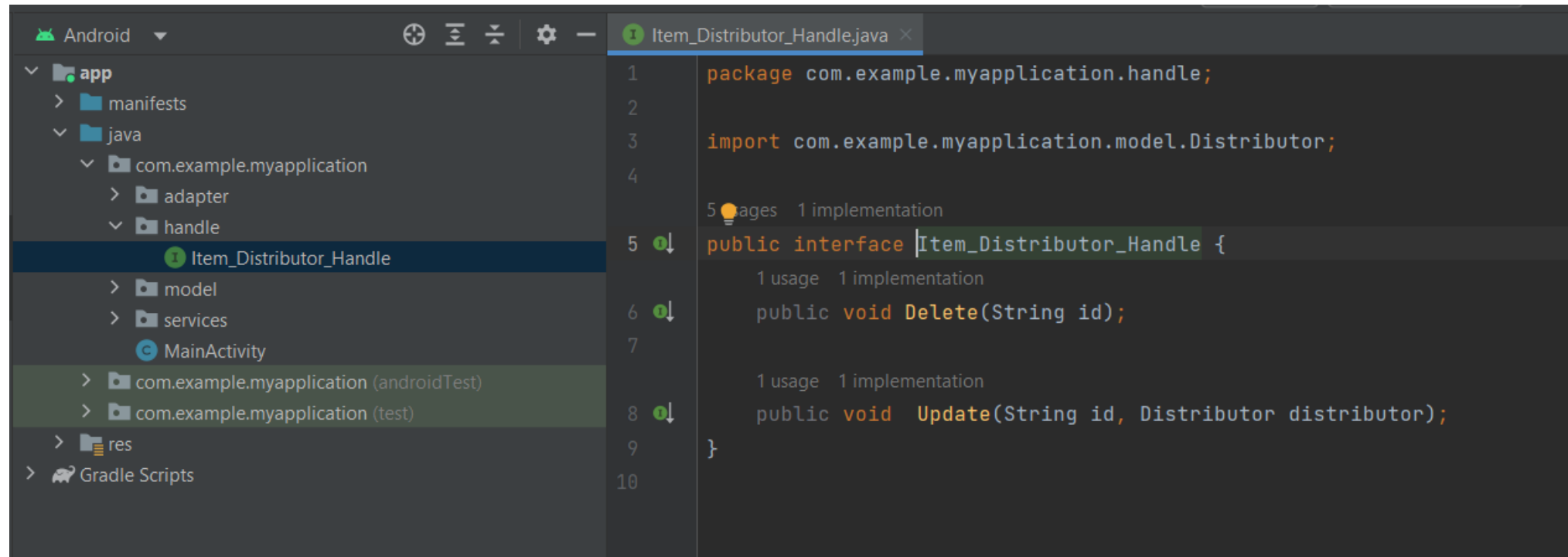
    6 usages
    public ApiService callAPI() {
        //Get Retrofit
        return requestInterface;
    }
}
```

□ Tạo adapter hiển thị dữ liệu



```
1 package com.example.myapplication.adapter;
2
3 import ...
4
19 4 usages
20 public class Recycle_Item_Distributors extends RecyclerView.Adapter<Recycle_Item_Distributors.ViewHolder>{
21
22     2 usages
23     private Context context;
24
25     7 usages
26     private ArrayList<Distributor> ds;
27
28     3 usages
29     private Item_Distributor_Handle handle;
30
31     1 usage
32     public Recycle_Item_Distributors(Context context, ArrayList<Distributor> ds, Item_Distributor_Handle handle) {
33         this.context = context;
34         this.ds = ds;
35         this.handle = handle;
36     }
37 }
```

❑ Tạo xử lý khi click vào item trong adapter



The screenshot shows the Android Studio IDE with the following components:

- Project Explorer (Left):** Displays the project structure. The package `com.example.myapplication` is expanded, showing sub-packages `adapter` and `handle`. The file `Item_Distributor_Handle` is selected under the `handle` package.
- Editor (Right):** Displays the code for `Item_Distributor_Handle.java`. The code defines a public interface with two methods: `Delete` and `Update`.

```
1 package com.example.myapplication.handle;  
2  
3 import com.example.myapplication.model.Distributor;  
4  
5 1 pages 1 implementation  
6 public interface Item_Distributor_Handle {  
7     1 usage 1 implementation  
8     public void Delete(String id);  
9  
10    1 usage 1 implementation  
11    public void Update(String id, Distributor distributor);  
12 }
```


Call API trong file MainActivity

```
public class MainActivity extends AppCompatActivity {
    private HttpRequest httpRequest;
    private RecyclerView recycle_distributors;
    private Recycle_Item_Distributors adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recycle_distributors = findViewById(R.id.recycle_distributors);
        //Khởi tạo services Request
        httpRequest = new HttpRequest();
        //Thực thi call API
        httpRequest.callAPI()
            .getListDistributor() //Phương thức API cần thực thi
            .enqueue(getDistributorAPI); // Xử lý bất đồng bộ
    }

    private void getData(ArrayList<Distributor> ds){
        adapter = new Recycle_Item_Distributors( context: this,ds);
        recycle_distributors.setLayoutManager(new LinearLayoutManager( context: this));
        recycle_distributors.setAdapter(adapter);
    }

    Callback<Response<ArrayList<Distributor>>> getDistributorAPI = new Callback<Response<ArrayList<Distributor>>>() {
        @Override
        public void onResponse(Call<Response<ArrayList<Distributor>>> call, retrofit2.Response<Response<ArrayList<Distributor>>> response) {
            //Khi call thành công sẽ chạy vào hàm này
            if(response.isSuccessful())
            {
                //check status code
                if(response.body().getStatus() == 200)
                {
                    //Lấy data
                    ArrayList<Distributor> ds = response.body().getData();
                    //Set dữ liệu lên recycle
                    getData(ds);
                    //Toast ra thông tin từ Messenger
                    Toast.makeText( context: MainActivity.this, response.body().getMessenger(), Toast.LENGTH_SHORT).show();
                }
            }
        }

        @Override
        public void onFailure(Call<Response<ArrayList<Distributor>>> call, Throwable t) {
            //Khi call thất bại sẽ chạy vào hàm này
            Log.d( tag: ">>> GetListDistributor", msg: "onFailure: " + t.getMessage());
        }
    };
}
```

❑ Mở file api.js viết api tìm kiếm distributor

```
router.get(path: '/search-distributor',...handlers: async (req,res) => {
  try {
    const key = req.query.key; // Nhận từ query
    //Lấy danh sách theo thứ tự distributors mới nhất
    const data = await Distributors.find(filter: {name: { "$regex": key, "$options": "i" }})
                                     .sort(arg: {createdAt: -1});
    if(data)
    {
      // Trả về danh sách
      res.json(body: {
        "status" : 200,
        "messenger" : "thành công",
        "data" : data
      })
    }
    else
    {
      // Nếu thêm không thành công result null, thông báo không thành công
      res.json(body: {
        "status" : 400 ,
        "messenger" : "Lỗi,thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(message: error);
  }
});
```

- ❑ Trong layout activity_main, thêm thuộc tính **android:imeOptions="actionSearch"**

```
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/edttimkiem"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:hint="Tìm kiếm nhà phân phối"
    android:imeOptions="actionSearch"
    android:inputType="text"
    android:maxLines="1"
    android:padding="0dp"
    android:textSize="12sp" />
</com.google.android.material.textfield.TextInputLayout>
```

□ Gọi **API** và trả về **kết quả** khi nhập chữ vào **EditText**

```
//Khởi tạo services Request
httpRequest = new HttpRequest();
//Thực thi call API
httpRequest.callAPI()
    .getListDistributor() //Phương thức API cần thực thi
    .enqueue(getDistributorAPI); // Xử lý bất đồng bộ

//Lắng nghe sự kiện nút tìm kiếm trên bàn phím
edtTimkiem.setOnEditorActionListener(new TextView.OnEditorActionListener() {
    no usages
    @Override
    public boolean onEditorAction(TextView textView, int i, KeyEvent keyEvent) {
        if(i == EditorInfo.IME_ACTION_SEARCH)
        {
            //Lấy từ khóa từ ô tìm kiếm
            String key = edtTimkiem.getText().toString();

            httpRequest.callAPI()
                .searchDistributor(key) //Phương thức API cần thực thi
                .enqueue(getDistributorAPI); // Xử lý bất đồng bộ
            //Vì giá trị trả về vẫn là một list Distributor
            //nên có thể sử dụng lại Callback của getListDistributor()
            return true;
        }
        return false;
    }
});
```

phương thức tạo trước đó





FPT Education

FPT POLYTECHNIC

Thank you