FPT Education

FPT POLYTECHNIC

LẬP TRÌNH ANDROID VỚI RESTAPI

QUERYMAP – LOAD MORE

www.poly.edu.vn

GIỚI THIỆU MÔN HỌC

- Giới thiệu QueryMap và cách sử dụng
- Giới thiệu Load More và cách sử dụng

Mục tiêu

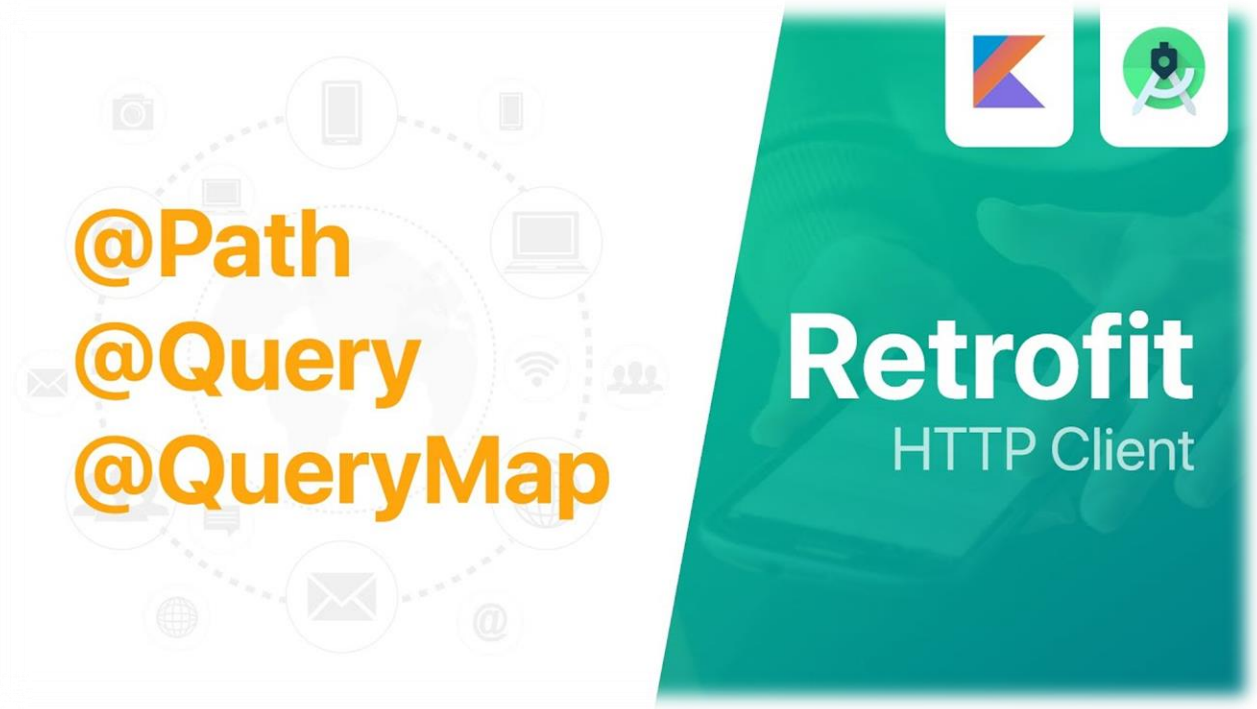
- Giới thiệu querymap và cách sử dụng
- Giới thiệu load more và cách sử dụng

QUERY MAP

...

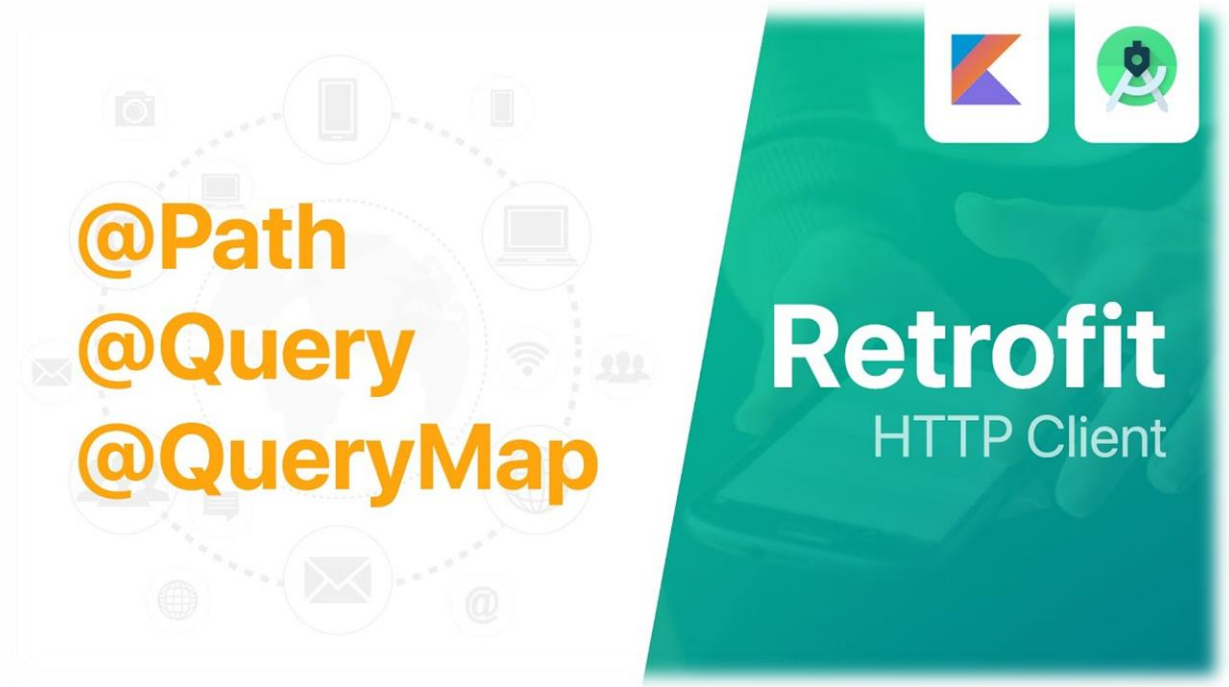
GIỚI THIỆU QUERYMAP

- **@QueryMap** trong Retrofit thường được sử dụng để truyền các tham số truy vấn (**query parameters**) vào các cuộc gọi **API**. Query **parameters** là các thông số bạn có thể gửi cùng với một cuộc gọi HTTP để yêu cầu dữ liệu cụ thể từ máy chủ



GIỚI THIỆU QUERYMAP

- Trong **Retrofit**, bạn có thể sử dụng **@QueryMap** để truyền một Map chứa các tham số truy vấn. Điều này giúp bạn đưa các tham số truy vấn vào một Map thay vì phải cung cấp từng tham số một cách riêng lẻ. Điều này có thể tiết kiệm thời gian và làm mã của bạn trở nên gọn gàng hơn



CÁCH SỬ DỤNG QUERYMAP

- Tạo một Map chứa các tham số truy vấn:
  - **Key** là tên của tham số truy vấn trong URL.
  - **Value** là giá trị của tham số đó.
- Ví dụ:

```
Map<String, String> queryParams = new HashMap<>();
queryParams.put("page", "10");
queryParams.put("sort", "asc");
```

CÁCH SỬ DỤNG QUERYMAP

- Sử dụng annotation **@QueryMap** trong phương thức của interface service:
  - Tham số của annotation là **Map<String, String>** chứa các tham số truy vấn
- Ví dụ:

```
@GET("/users")
Call<List<User>> getUsers(@QueryMap Map<String, String> queryParams);
```

CÁCH SỬ DỤNG QUERYMAP

- Ưu điểm của **@QueryMap**:
  - Giúp code gọn gàng, dễ đọc hơn so với việc định nghĩa từng tham số riêng lẻ.
  - Thích hợp cho các trường hợp bạn có nhiều tham số truy vấn động.
  - Giảm thiểu khả năng sai sót khi định nghĩa URL.
- Lưu ý:
  - Các **key** trong Map không được **null**.
  - Retrofit sẽ tự động **encode** các giá trị trong Map.

CÁCH SỬ DỤNG QUERYMAP

- Ngoài ra:
  - **@QueryMap** cũng có thể được sử dụng với các Converter tùy chỉnh để xử lý các kiểu dữ liệu phức tạp hơn cho các tham số truy vấn.
  - Đối với các API yêu cầu sắp xếp theo thứ tự cụ thể, bạn có thể sử dụng thư viện như **MapSort** để đảm bảo trình tự của các tham số trong URL.

SỬ DỤNG QUERYMAP

...

SỬ DỤNG QUERYMAP

- Trong file **api.js** tạo api **get-page-fruit**

```
router.get(path: '/get-page-fruit',...handlers: async (req,res) => {
    //Auten
    const authHeader = req.headers['authorization']
    const token = authHeader && authHeader.split(separator: ' ')[1]
    if (token == null) return res.sendStatus(code: 401)
    let payload;
    JWT.verify(token, secretOrPublicKey: SECRETKEY , callback: (err,_payload) => {
        if(err instanceof JWT.TokenExpiredError) return res.sendStatus(code: 401)
        if (err) return res.sendStatus(code: 403)
        payload = _payload;
      })
    let perPage = 6; // số lượng sản phẩm xuất hiện trên 1 page
    let page = req.query.page || 1; //Page truyền lên
    let skip = (perPage * page) - perPage; // Phân trang
    let count = await Fruits.find().count();// Lấy tổng số phần tử
    //filtering
    //Lọc theo tên
    const name = { "$regex": req.query.name ?? "", "$options": "i" }
    //Lọc giá lớn hơn hoặc bằng giá truyền vào
    const price = {$gte: req.query.price ?? 0}
    //Lọc sắp xếp theo giá
    const sort = {price: req.query.sort ?? 1}
```

SỬ DỤNG QUERYMAP

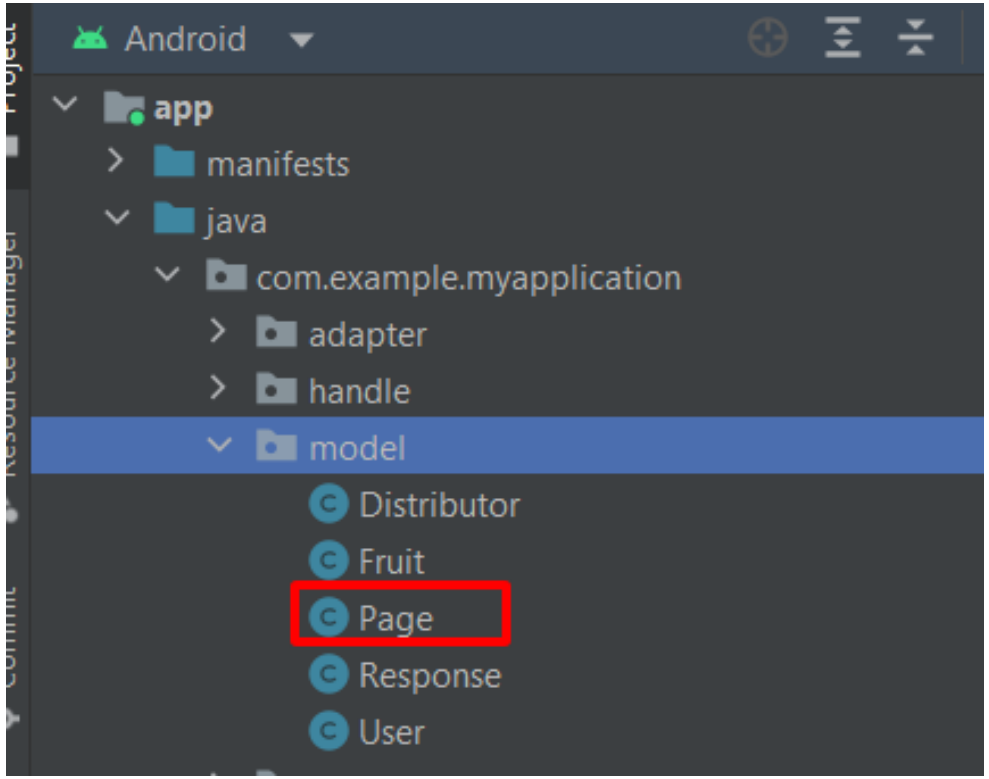
- Trong file **api.js** tạo api **get-page-fruit** *(tiếp theo)*

```
    //Lọc sắp xếp theo giá
    const sort = {price: req.query.sort ?? 1}

    try {
        const data = await Fruits.find({name: name, price: price})
                                 .populate('id_distributor')
                                 .sort(sort)
                                 .skip(skip)
                                 .limit(perPage)
        res.json(body: {
            "status" : 200,
            "messenger" : "Danh sách fruit",
            "data" : {
                "data" : data,
                "currentPage" : Number(value: page),
                "totalPage" : Math.ceil(x: count/perPage)
            }
        })
    } catch (error) {
        console.log(message: error);
    }
})
```

SỬ DỤNG QUERYMAP

- Trong project Android Studio tạo model **Page**



SỬ DỤNG QUERYMAP

- Viết code model Page

Page.java

```
1   package com.example.myapplication.model;
2
    8 usages
3   public class Page <T> {
        3 usages
4       private T data;
        3 usages
5       private int currentPage,totalPage;
6
        no usages
7       public Page() {
8       }
9
        no usages
10      public Page(T data, int currentPage, int totalPage) {
11          this.data = data;
12          this.currentPage = currentPage;
13          this.totalPage = totalPage;
14      }
```

SỬ DỤNG QUERYMAP

- Viết code model Page *(tiếp theo)*

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

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

    no usages
    public int getCurrentPage() { return currentPage; }

    no usages
    public void setCurrentPage(int currentPage) { this.currentPage = currentPage; }

    1 usage
    public int getTotalPage() { return totalPage; }

    no usages
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
}
```

SỬ DỤNG QUERYMAP

- Trong **ApiServices** thêm phương thức **@GET**, sử dụng **@QueryMap**

```
2 usages
@GET("get-page-fruit")
Call<Response<Page<ArrayList<Fruit>>>>
getPageFruit(@QueryMap Map<String,String> stringMap);
```

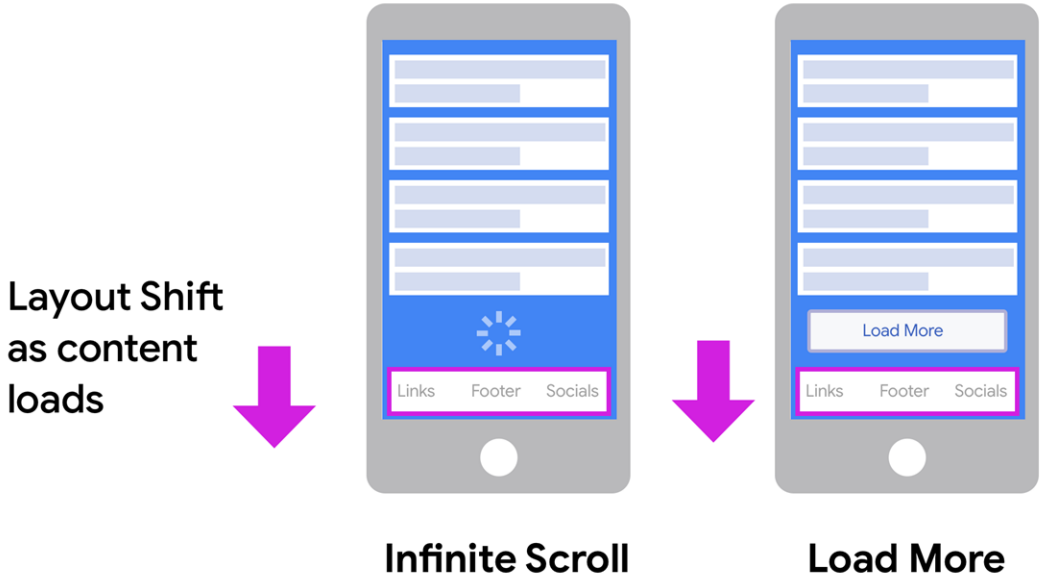
Fpt University
FPT POLYTECHNIC

LOAD MORE

...

GIỚI THIỆU LOAD MORE

- **Load more** được sử dụng để tải thêm dữ liệu khi người dùng cuộn đến cuối danh sách hoặc giao diện. Thường thấy trong danh sách hoặc lưới các mục, tính năng "load more" cho phép người dùng xem thêm nội dung mới mà không cần phải chuyển sang một màn hình khác hoặc tải lại toàn bộ danh sách.



GIỚI THIỆU LOAD MORE

- Một số điều cần lưu ý khi thực hiện loadmore:
  - Hiển thị progress indicator trong khi đang tải dữ liệu.
  - Ngăn chặn cuộn tiếp trong khi đang tải để tránh lỗi.
  - Xử lý trường hợp hết dữ liệu để hiển thị message thông báo.
  - Kiểm soát tốc độ tải để tránh overload server.

SỬ DỤNG LOADMORE

...

SỬ DỤNG LOAD MORE

- Xây dựng Layout hiển thị danh sách fruit

```
<androidx.core.widget.NestedScrollView
    android:id="@+id/nestScrollView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout">

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

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycle_fruits"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:nestedScrollingEnabled="false" />

        <ProgressBar
            android:id="@+id/loadmore"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:indeterminate="true"
            android:indeterminateTint="@color/purple_500"
            android:indeterminateTintMode="src_atop"
            android:visibility="gone" />
    </LinearLayout>
</androidx.core.widget.NestedScrollView>
```

Gọi API

```
@Override
protected void onResume() {
    super.onResume();
    httpRequest.callAPI().getPageFruit( token: "Bearer " + token,page).enqueue(getListFruitRespone);
}

Callback<Response<Page<ArrayList<Fruit>>>> getListFruitRespone = new Callback<Response<Page<ArrayList<Fruit>>>>() {
    @Override
    public void onResponse(Call<Response<Page<ArrayList<Fruit>>>> call, retrofit2.Response<Response<Page<ArrayList<Fruit>>>> response) {
        if(response.isSuccessful())
        {
            //check status code
            if(response.body().getStatus() == 200)
            {
                //Set total Page
                totalPage = response.body().getData().getTotalPage();
                //Lấy data
                ArrayList<Fruit> _ds = response.body().getData().getData();
                //Set dữ liệu lên recycle
                getData(_ds);
                //Toast ra thông tin từ Messenger
            }
        }
    }

    @Override
    public void onFailure(Call<Response<Page<ArrayList<Fruit>>>> call, Throwable t) {
        Log.d( tag: ">>> getListFruit",  msg: "onFailure: " + t.getMessage());
    }
};
```

SỬ DỤNG LOAD MORE

- Nội dung getData()

```
1 usage
private void getData(ArrayList<Fruit> _ds)
{
    //Kiểm tra nếu process load more chạy thì chỉ cần add thêm fruits vào list
    if(loadmore.getVisibility() == View.VISIBLE)
    {
        //Do chạy ở local nên tốt độ mạng tốt
        //nên sẽ thêm 1 đoạn code deplay ( deplay 1s)
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                //
                adapter.notifyItemInserted( position: ds.size() - 1);
                loadmore.setVisibility(View.GONE);
                ds.addAll(_ds);
                //Thông báo adapter dữ liệu thay đổi
                adapter.notifyDataSetChanged();
            }
        }, delayMillis: 1000);
        return;
    }
    ds.addAll(_ds);
    adapter = new Recycle_Item_Fruits(ds, context: this);
    recycle_fruits.setLayoutManager(new GridLayoutManager( context: this, spanCount: 2));
    recycle_fruits.setAdapter(adapter);
}
```

SỬ DỤNG LOAD MORE

- Bắt sự kiện khi load đến item cuối cùng

```
nestScrollView.setOnScrollChangeListener(new NestedScrollView.OnScrollChangeListener() {
    1 usage
    @Override
    public void onScrollChange(@NonNull NestedScrollView v, int scrollX, int scrollY,
                               int oldScrollX, int oldScrollY) {
        if (scrollY == v.getChildAt( index: 0).getMeasuredHeight() - v.getMeasuredHeight()) {
            if(totalPage == page) return;
            if (loadmore.getVisibility() == View.GONE) {
                loadmore.setVisibility(View.VISIBLE);
                page++; // Tăng page
                //Call API
                //load more theo filter
                FilterFruit();
            }
        }
    }
});
```

- Nội dung FilterFruit()

```
2 usages
private void FilterFruit()
{
    String _name = edttimkiem.getText().toString().equals("") ? "" : edttimkiem.getText().toString();
    String _price = edtgia.getText().toString().equals("") ? "0" :  edtgia.getText().toString();
    String _sort = sort.equals("") ? "-1" : sort;
    Map<String,String> map = getMapFilter(page,_name,_price,_sort);
    httpRequest.callAPI().getPageFruit(map).enqueue(getListFruitRespone);
}

//Hàm setup MapQuery
```

- Xem kết quả


2:38
My Application
FPT POLYTECHNIC
Username:
fpl
Password:
LOGIN
Đăng ký tài khoản

Fpt University
FPT POLYTECHNIC


FPT Education
FPT POLYTECHNIC
Thank you
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN