



Tương tác với API

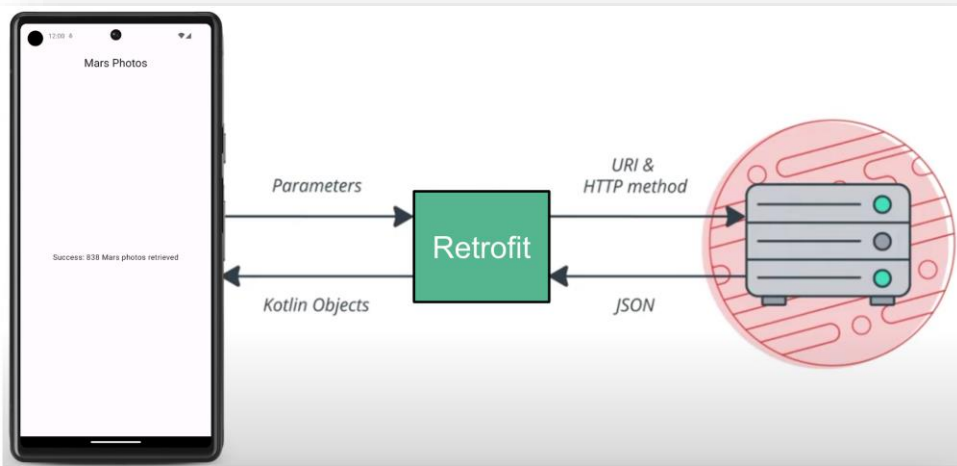
- Giới thiệu về Retrofit
- Ưu điểm của Retrofit so với các thư viện khác
- Các tính năng chính trong Retrofit
- Tương tác với API

Nội dung

- Giới thiệu về Retrofit
- Ưu điểm của Retrofit so với các thư viện khác
- Các tính năng chính trong Retrofit
- Tương tác với API

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)



Retrofit

Giống như hầu hết các phần mềm mã nguồn mở khác, Retrofit được xây dựng dựa trên một số thư viện mạnh mẽ và công cụ khác. Đằng sau nó, Retrofit làm cho việc sử dụng OkHttp (từ cùng một nhà phát triển) để xử lý các yêu cầu trên mạng. Ngoài ra, Retrofit không tích hợp bất kỳ một bộ chuyển đổi JSON nào để phân tích từ JSON thành các đối tượng Java. Thay vào đó nó đi kèm với các thư viện chuyển đổi JSON sau đây để xử lý điều đó

Tại sao lại dùng Retrofit

- Retrofit đơn giản trong việc setup và sử dụng : phát triển thư viện type-safe HTTP của riêng của bạn để giao tiếp với một REST API có thể thật sự rất khó: bạn phải xử lý nhiều khía cạnh, chẳng hạn như kết nối, bộ nhớ đệm, thử lại yêu cầu sai, luồng, phân tích phản hồi, xử lý lỗi và nhiều thứ khác.
- Là một thư viện được tổ chức tốt, tài liệu hướng đầy đủ và đã thử nghiệm sẽ giúp bạn tiết kiệm rất nhiều thời gian quý báu và những đau đầu không cần thiết.

Tại sao lại dùng Retrofit

- Retrofit là một type-safe HTTP client: trình biên dịch sẽ xác nhận hợp lệ các kiểu dữ liệu trong khi biên dịch và ném một lỗi nếu bạn cố gán kiểu sai cho một biến.
- Nhanh hơn rất nhiều so với việc sử dụng **Volley**, **AysncTask**

So sánh Retrofit với AsyncTask, Volley

	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

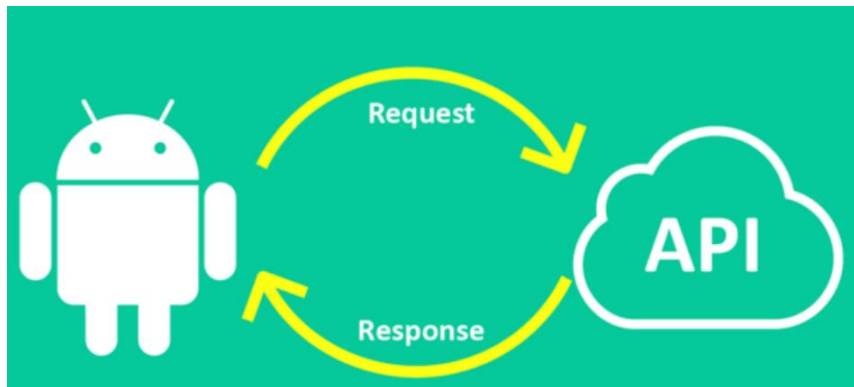
Ưu điểm/Nhược điểm Retrofit

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

Cách dùng Retrofit

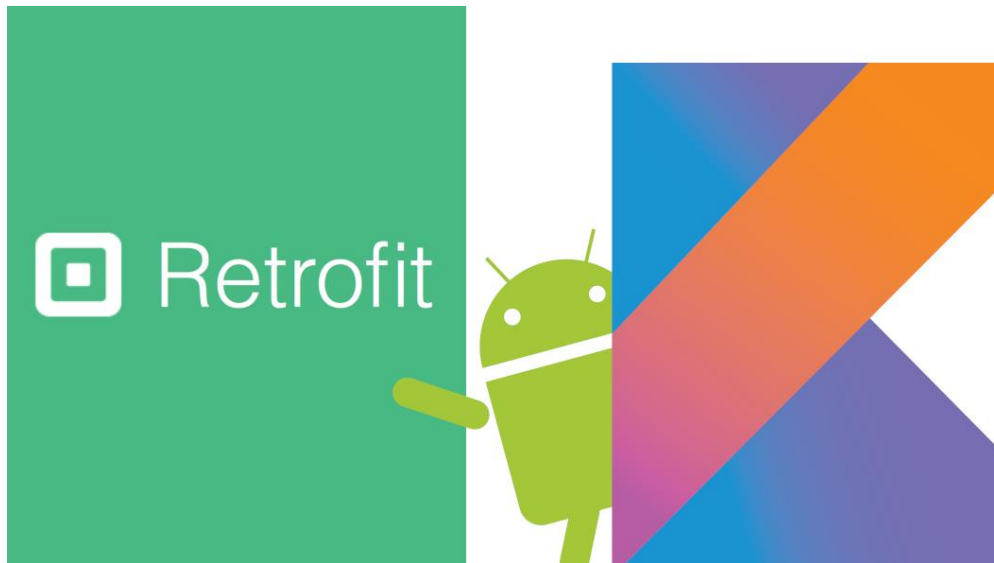
Để làm việc với Retrofit bạn cần triển khai cơ bản 3 lớp:

- Model class để ánh xạ với JSON data.
- Một interface dùng để định nghĩa các hàm và phương thức HTTP
- Retrofit.Builder Lớp để định nghĩa URL Endpoint cho các hoạt động liên quan tới HTTP



APIDeclaration

Annotations trong interface và các tham số của chúng chỉ ra cách mà request được thực hiện



Request Method

Mỗi method phải có một HTTP annotation. Có 5 annotations **GET**, **POST**, **PUT**, **DELETE**, và **HEAD**. Bên trong mỗi annotation là một đoạn của URL.

```
@GET("list-film.php")
```

Bạn cũng có thể chỉ định tham số truy vấn trong URL

```
@GET("users/list?sort=desc")
```

URL MANIPULATION

Dưới đây là ví dụ của interface dùng để định nghĩa các http request

```
interface MovieService {  
    @GET("list-film.php")  
    suspend fun getListFilms(): Response<List<MovieResponse>>  
  
    @GET("film-detail.php")  
    suspend fun getFilmDetail(@Query("id") id: String): Response<MovieResponse>  
}
```

Request Body

Một đối tượng có thể được chỉ định cho mục đích sử dụng làm phần body cho truy vấn với

@Body annotation

```
@POST("add-film.php")
suspend fun addFilm(@Body filmRequest: MovieRequest): Response<StatusResponse>
```

Đối tượng cũng sẽ được chuyển đổi bằng cách sử dụng trình chuyển đổi được chỉ định trong Retrofit instance. Nếu không có trình chuyển đổi nào được thêm vào, chỉ có thể sử dụng **RequestBody**.

Form Encoded and multipart

Các phương thức cũng có thể được khai báo để gửi dữ liệu được mã hóa theo mẫu và nhiều phần. Dữ liệu được mã hóa biểu mẫu được gửi khi có annotation **@FormUrlEncoding** . Mỗi cặp **key-value** được chú thích bằng **@Field** chứa tên và đối tượng cung cấp giá trị.

```
@FormUrlEncoded
@POST("user/edit")
suspend fun updateUser(
    @Field("first_name") first: String,
    @Field("last_name") last: String): Response<StatusResponse>
```

Form Encoded and multipart

Yêu cầu nhiều phần được sử dụng với **@Multipart** .Mỗi phần được khai báo bằng cách sử dụng chú thích **@Part**.

```
@Multipart
@PUT("user/photo")
suspend fun updateUser(
    @Part("photo") photo : MovieRequest,
    @Part("description") description : MovieRequest);
```

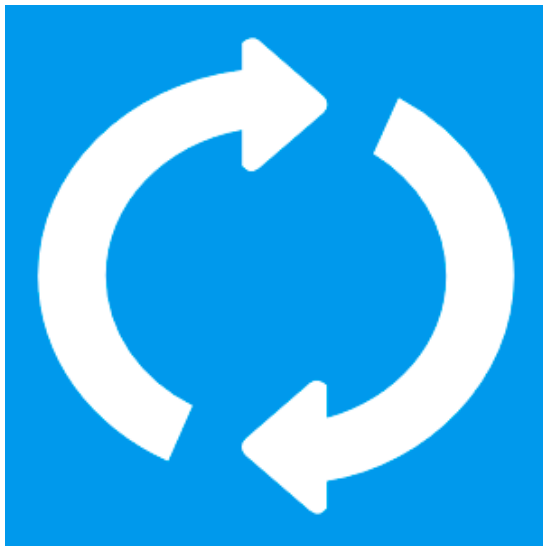
Header manipulation

Bạn có thể Header tĩnh cho phương thức sử dụng @Headers annotation

Lưu ý rằng các headers không ghi đè lên nhau. Tất cả các tiêu đề có cùng tên sẽ được bao gồm trong yêu cầu. Một tiêu đề yêu cầu có thể được cập nhật động bằng cách sử dụng chú thích **@Header**. Một tham số tương ứng phải được cung cấp cho **@Header**. Nếu giá trị là null, header sẽ bị bỏ qua. Nếu không, toString sẽ được gọi trên giá trị và kết quả được sử dụng.

Converters

Theo mặc định, Retrofit chỉ có thể giải tuần tự hóa (deserialize) các HTTPbodies thành ResponseBody của **OkHttp** và nó chỉ có thể chấp nhận loại RequestBody của nó cho **@Body**. Bộ chuyển đổi có thể được thêm vào để hỗ trợ các loại khác.



Converters

Dưới đây là 6 thư viện phổ biến sử dụng.

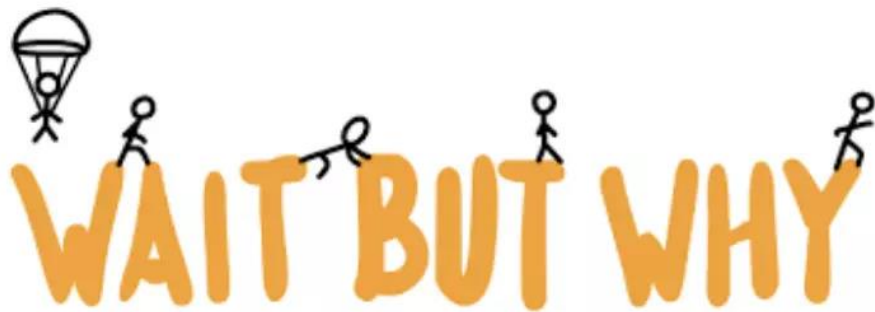
- **Gson:** com.squareup.retrofit:converter-gson
- **Jackson:** com.squareup.retrofit:converter-jackson
- **Moshi:** com.squareup.retrofit:converter-moshi
- **Protobuf:** com.squareup.retrofit2:converter-protobuf
- **Wire:** com.squareup.retrofit2:converter-wire

Và đối với XML, Retrofit hỗ trợ:

- **Simple Framework:** com.squareup.retrofit2:converter-simpleframework

Caching

Caching là cách lưu trữ tạm thời dữ liệu được tìm nạp từ mạng trên bộ lưu trữ của thiết bị, để chúng ta có thể truy cập vào lần sau khi thiết bị ngoại tuyến hoặc nếu chúng ta muốn truy cập lại cùng một dữ liệu.



Lợi ích của việc Caching

- Giảm tiêu thụ băng thông.
- Tiết kiệm cho bạn thời gian bạn dành thời gian chờ đợi máy chủ cung cấp cho bạn phản hồi mạng.
- Tiết kiệm cho máy chủ gánh nặng của lưu lượng bổ sung.
- Nếu bạn cần truy cập lại cùng một tài nguyên mạng sau khi đã truy cập gần đây, thiết bị của bạn đã giành được Yêu cầu phải gửi yêu cầu đến máy chủ; Thay vào đó, nó sẽ nhận được phản hồi lưu trữ

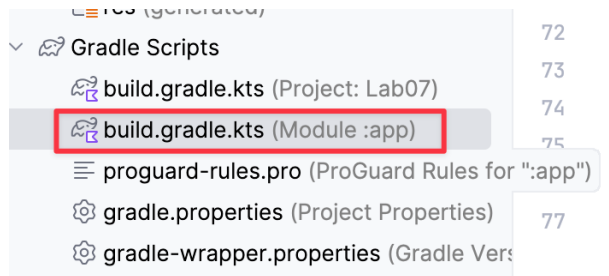
Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 1: Import thư viện retrofit

```
implementation ("com.squareup.retrofit2:retrofit:2.9.0")
```

```
implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
```



```
implementation(libs.androidx.lifecycle.runtime.ktx)
implementation("androidx.compose.runtime:runtime-livedata:1.6.0")

implementation ("com.squareup.retrofit2:retrofit:2.9.0")
implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
```

Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 2: Tạo **MovieResponse** khai báo các giá trị trả về khi gọi API

```
data class MovieResponse(  
    @SerializedName("filmId") val filmId: String,  
    @SerializedName("filmName") val filmName: String,  
    @SerializedName("duration") val duration: String,  
    @SerializedName("releaseDate") val releaseDate: String,  
    @SerializedName("genre") val genre: String,  
    @SerializedName("national") val national: String,  
    @SerializedName("description") val description: String,  
    @SerializedName("image") val image: String,  
)
```

Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 3: Tạo **MovieService** quản lý các API gọi trong ứng dụng

```
interface MovieService {  
    @GET("list-film.php")  
    suspend fun getListFilms(): Response<List<MovieResponse>>  
}
```

Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 4: Tạo **RetrofitService** để cấu hình Retrofit

```
open class RetrofitService() {  
  
    private val retrofit: Retrofit = Retrofit.Builder()  
        .baseUrl("<BASE_URL>")  
        .addConverterFactory(GsonConverterFactory.create())  
        .build()  
  
    val movieService: MovieService by lazy {  
        retrofit.create(MovieService::class.java)  
    }  
}
```


Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 5: Trong class **Movie** để quản lý các thuộc tính của phim

```
data class Movie(  
    val id: String,  
    val title: String,  
    val releaseDate: String,  
    val posterUrl: String,  
    val shotDescription: String,  
    val genre: String,  
    val duration: String,  
)
```

Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 6: Tạo class **Transform** để convert dữ liệu lấy từ API thành model

```
fun MovieResponse.toMovie(): Movie {  
    return Movie(  
        id = this.filmId,  
        title = this.filmName,  
        duration = this.duration,  
        releaseDate = this.releaseDate,  
        genre = this.genre,  
        shotDescription = this.description,  
        posterUrl = this.image  
    )  
}
```

Ví dụ minh họa

Gọi API để lấy thông tin danh sách phim

Bước 7: Tạo **MovieViewModel** để xử lý chức năng gọi API lấy danh sách

```
class MovieViewModel : ViewModel() {
    private val _movies = MutableLiveData<List<Movie>>()
    val movies: LiveData<List<Movie>> = _movies


    init {
        getMovies()
    }



    fun getMovies() {
        viewModelScope.launch {
            try {
                val response = RetrofitService().movieService.getListFilms()
                if (response.isSuccessful) {
                    _movies.postValue(response.body()?.map { it.toMovie() })
                } else {
                    _movies.postValue(emptyList())
                }
            } catch (e: Exception) {
                Log.e("TAG", "getMovies: " + e.message)
                _movies.postValue(emptyList())
            }
        }
    }
}
```


Ta được kết quả:



8:04


Thêm





GẶP LẠI CHỊ BẦU
Thời lượng: 130
Khởi chiếu: 31/01/2024
Thể loại: **Tình Cảm, Phim Hài**
Tóm tắt nội dung
"Gặp Lại Chị Bầu" xoay quanh Phúc, một thanh viên trẻ với đam mê diễn xuất nhưng phải trải qua cuộc sống muôn vàn khó khăn. Anh tình cờ lưu lạc đến xóm trọ của bà Lê và cùng những người bạn ở đây trải ...




MAI
Thời lượng: 145
Khởi chiếu: 01/02/2024
Thể loại: **Tình Cảm, Tâm Lý**
Tóm tắt nội dung
"Mai" xoay quanh cuộc đời của một người phụ nữ đẹp tên Mai (do Phương Anh Đào thủ vai) có số phận rất đặc biệt. Bởi làm nghề mát xa, Mai thường phải đối mặt với ánh nhìn soi mói, phán xét từ những người...




THANH GƯƠM DIỆT QUỶ: PHÉP MÀU TÌNH THÂN, CHO ĐẾN CHUYẾN ĐẶC ...
Thời lượng: 120
Khởi chiếu: 22/02/2024
Thể loại: **Hoạt hình**
Tóm tắt nội dung
Phim lấy bối cảnh ở làng thợ rèn, kể về hồi ...


Thanks!

