ANDROID

RETROFIT SESSION

Agenda

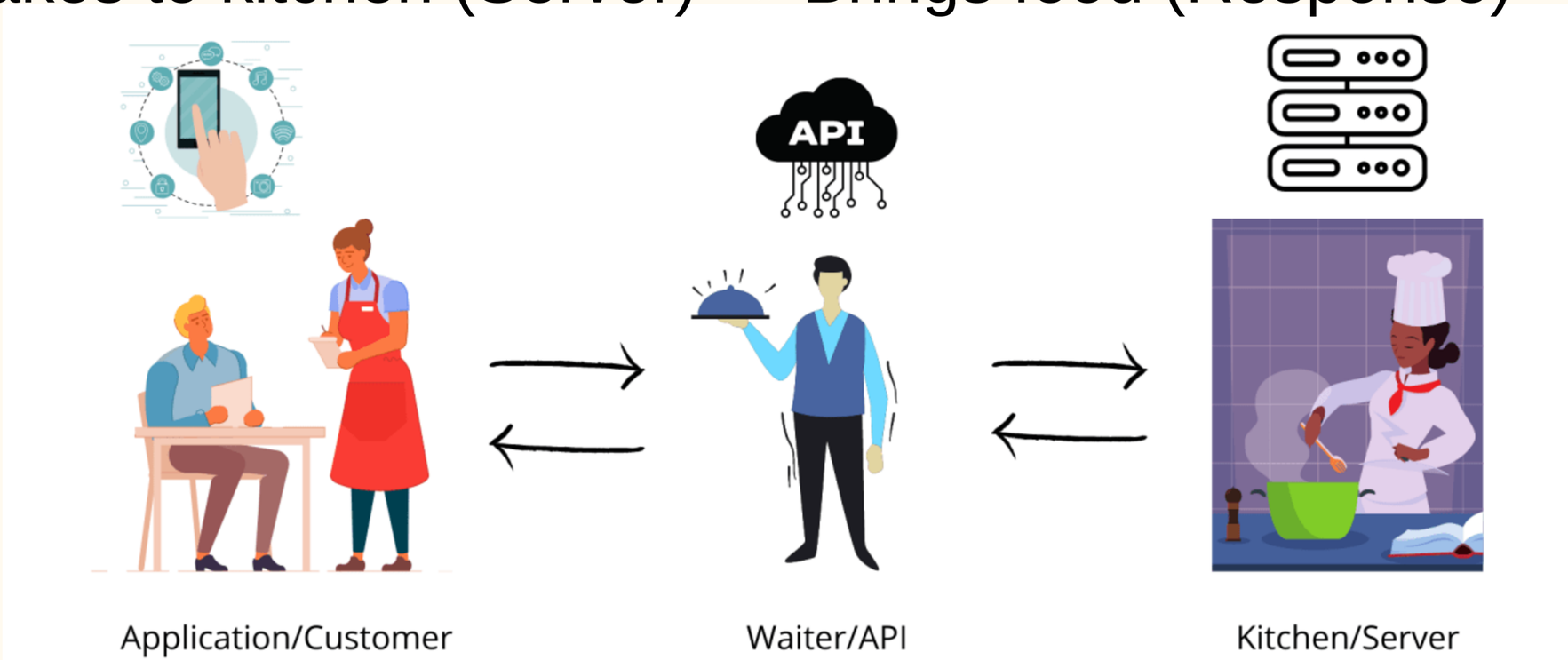
- API
- JSON
- Retrofit
- viewmodel (MVVM)
- Live Data

API

API

"API is like a waiter in a restaurant"

- You (Client App) → Give order (Request) → Waiter (API) → Takes to kitchen (Server) → Brings food (Response)



API

Why APIs Matter ?

- Most apps need real-time data (weather, social media, payments)

Without APIs:

- App would only work offline
- Limited functionality (e.g., no live weather updates)

With APIs:

- Fetch data from servers (e.g., Twitter feeds)
- Integrate third-party services (Google Maps, PayPal)

API

Types of API

- Public API: Open to the public, accessible with or without an API key.
- Key-based API: Requires an API key for authentication.
- OAuth API: Uses OAuth for secure, delegated access.

API

Types of API architectures

- REST (most common in mobile/web development)
- SOAP
- GraphQL
- WebSockets

But for mobile apps, REST APIs are the most popular and easiest to use.

REST API

- Uses HTTP methods (GET, POST, PUT, DELETE)
- Data usually returned as JSON could be(XML)

HTTP Protocol

what is HTTP

- HTTP is the language (or communication protocol) that your app and the API use to talk to each other.
- How the app asks for something (request)
- How the API answers (response)
- What actions the app can ask for (using HTTP methods like GET, POST, etc.)
- How success/failure (Error type) is reported (status codes)

HTTP Protocol

Steps to Send an HTTP Request:

1-Get the URL (Endpoint) :

The first step is to know the URL (also called the endpoint) to which you want to send the request.

Example: https://api.example.com/endpoint. (Try it)

HTTP Protocol

Steps to Send an HTTP Request:

2-Choose the HTTP Method (GET, POST, etc.)

GET: Used to retrieve data from the server.

POST: Used to send data to the server (usually when creating new resources).

PUT/PATCH: Used for updating existing resources.

DELETE: Used for deleting resources.

HTTP Protocol

Steps to Send an HTTP Request:

Add Query Parameters (if needed) optional

Example: https://api.example.com/users?page=1&limit=10.

Query Parameters are part of the URL and appear after the ? symbol.

They are used to modify or filter the data returned by the API. In this case, page=1 and limit=10 are used to request the first page of results with a limit of 10 users per page.

HTTP Protocol

Steps to Send an HTTP Request:

Headers are used to send metadata about the request or to authenticate the request. Unlike query parameters, headers do not appear in the URL path. Instead, they are included in the request's metadata, which is typically provided in the API documentation.

Common headers include:

- Authorization: Used for authentication, such as Bearer tokens or API keys.
- Accept: Specifies the response format you want, like application/json.

HTTP Protocol

API Examples :

path query example api key header example

⚠️Warning : Be careful with API rate limits!⚠️

Most APIs have a rate limit (maximum number of requests allowed per minute/hour/day).

If you exceed this limit, your API key or account may get temporarily blocked or permanently banned.

Always check the API documentation for the allowed request limits

HTTP Protocol

Handling API responses:

- Status code (e.g., 200, 404, 500): to know if the request succeeded or failed.
- Error messages: descriptive messages provided by the API in case of error.
- Response body (JSON, XML, etc.): where the actual data comes.

JSON

JSON

What is JSON ?

A JSON object is like a dictionary or map → it holds key-value pairs

The whole object is wrapped with curly braces { }.

Each key (name) is a string → always inside double quotes " "

“key” : Value

Each key has a value (data) → can be a number, String, List, etc.

Example: https://jsonplaceholder.typicode.com/posts. (Try it)

Example: https://www.themealdb.com/api.php. (Try it)

JSON

What is JSON ?

```
"userId": 1,
"id": 1,
"title": "sunt aut facere repellat provident occaecati exceptur
"body": "quia et suscipit suscipit recusandae consequuntur expe
autem sunt rem eveniet architecto"
```

RETROFIT

Retrofit

what is Retroft:

Retrofit is a type-safe HTTP client library for Android/Flutter and Java/Kotlin, developed by Square. It simplifies making REST API calls by converting HTTP APIs into Java/Kotlin interfaces using annotations.

Supports multiple data formats (JSON, XML)

Retrofit

Path to deal with API in android :

- INTERNET permission
- dependency
- Retrofit object
- Data class
- API Service Interface
- ViewModel
- UI observes ViewModel.

Retrofit

INTERNET permission

```
<uses-permission
android:name="android.permission.INTERNET">
</uses-permission>
```

Dependency :

```
gson="2.11.0"
retrofit="2.11.0"
gsonconverter="2.11.0"
```

```
gson={group="com.google.code.gson",name="gson",version.ref="gson"}
retrofit={group="com.squareup.retrofit2",name="retrofit",version.ref="retrofit"}
converter-gson = { group = "com.squareup.retrofit2", name = "converter-gson",
version.ref = "gsonconverter" }
```

Retrofit

Builder

oops new design pattern

Why Use It?

Avoids "telescoping constructors" (too many parameters).

Makes object creation more readable.

Allows optional parameters (no need to pass null).

Retrofit

Builder

oops new design pattern

```
class Burger(
    val cheese: Boolean,
    val lettuce: Boolean,
    val tomato: Boolean,
    val mayo: Boolean
)

// Usage: Unclear what `true/false` means!
val burger = Burger(true, false, true, false)
```

Retrofit

Builder

oops new design pattern

```
// Usage (Fluent API)
val myBurger = Burger.Builder()
    .addCheese()
    .addLettuce()
    .build()
```

Retrofit

Builder

oops new design pattern

```
class Burger private constructor(
    val cheese: Boolean,
    val lettuce: Boolean,
    val tomato: Boolean
) {
    class Builder {
        private var cheese: Boolean = false
        private var lettuce: Boolean = false
        private var tomato: Boolean = false

        fun addCheese() = apply { cheese = true }
        fun addLettuce() = apply { lettuce = true }
        fun addTomato() = apply { tomato = true }

        fun build() = Burger(cheese, lettuce, tomato)
    }
}
```

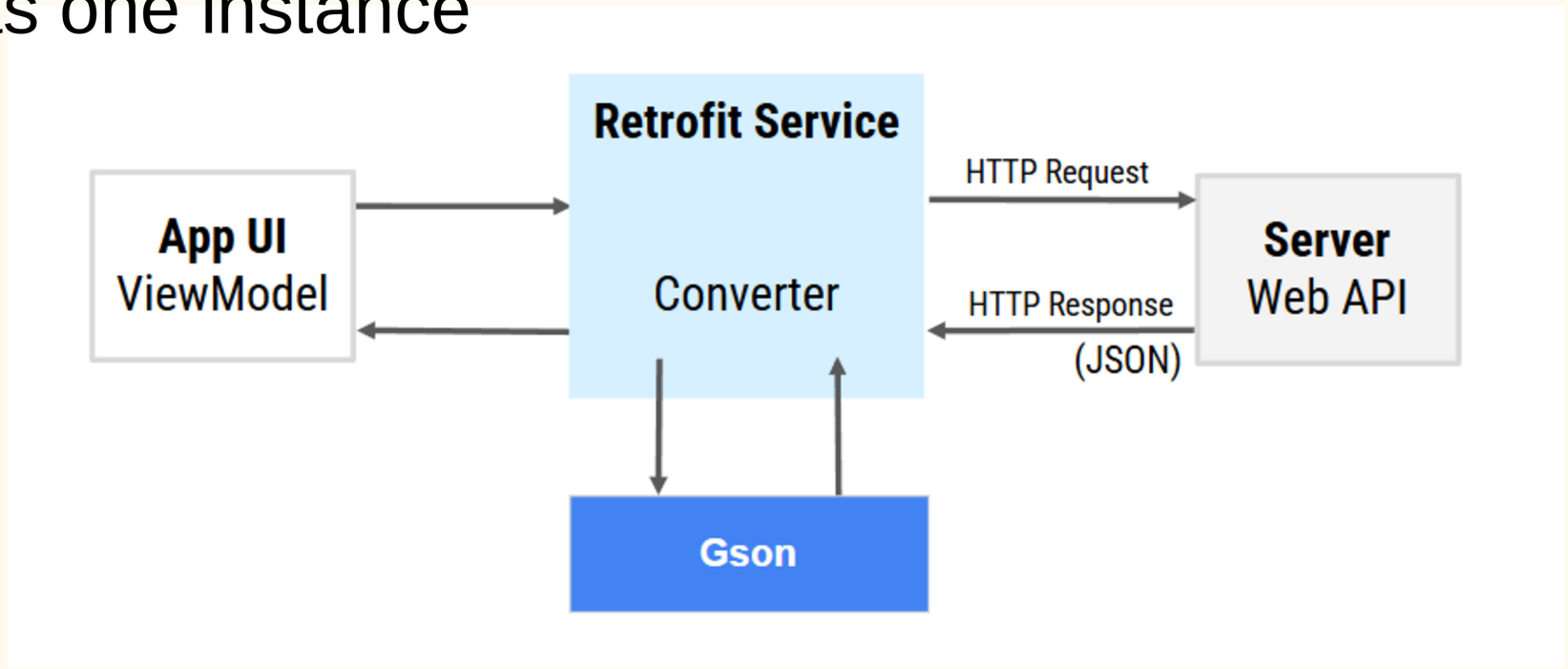
why constructor private ???
why there are variables inside class builder ??
what function build do ????

Retrofit

Retrofit object

first why object to be easy access

and has one instance



Retrofit

Gson

Gson is a JSON serialization/deserialization library from Google that converts:

Kotlin Data class → JSON (when sending requests)

JSON → Kotlin Data class (when receiving responses)

Retrofit

Retrofit object

```
object API {
    val gson=GsonBuilder().serializeNulls().create()
    val retrofit= Retrofit.Builder().
    baseUrl("https://api.rawg.io/api/developers/").
    addConverterFactory(GsonConverterFactory.create(gson))
        .build()

    val retrofitService:ApIservice by lazy {
        retrofit.create(ApIservice::class.java)
    }
}
```

side note

using lazy for the service means the instance is created only when you first access it, not when the app starts.

Retrofit

Data class

We need a Kotlin data class to map or convert this JSON into a Kotlin object that we can work with inside the app.

json

```json
{
  "id": 1,
  "name": "Chicken Curry",
  "category": "Dinner"
}
```

kotlin

```kotlin
data class Meal(
    val id: Int,
    val name: String,
    val category: String
)
```

Retrofit

Data class

We need a Kotlin data class to map or convert this JSON into a Kotlin object that we can work with inside the app.

json

```json
{
  "id": 1,
  "name": "Chicken Curry",
  "category": "Dinner"
}
```

kotlin

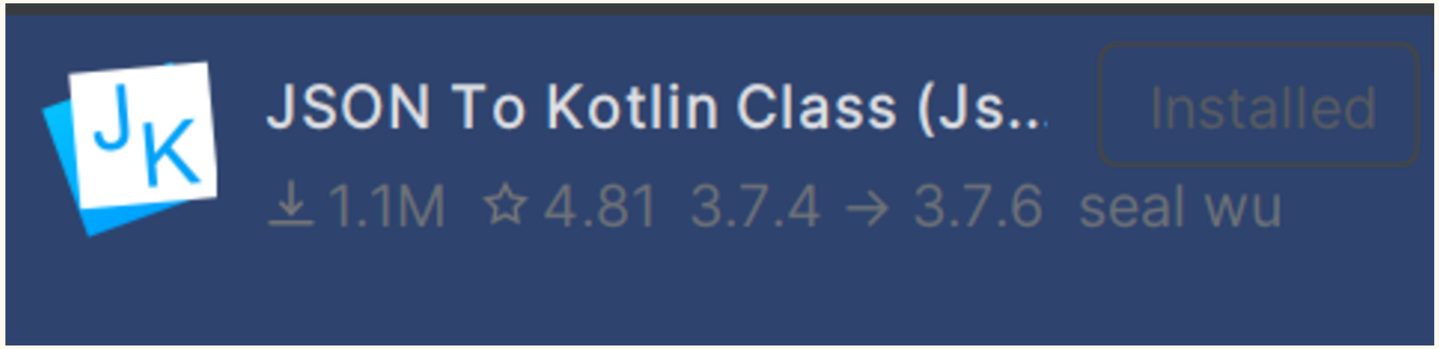
```kotlin
data class Meal(
    val id: Int,
    val name: String,
    val category: String
)
```

Retrofit

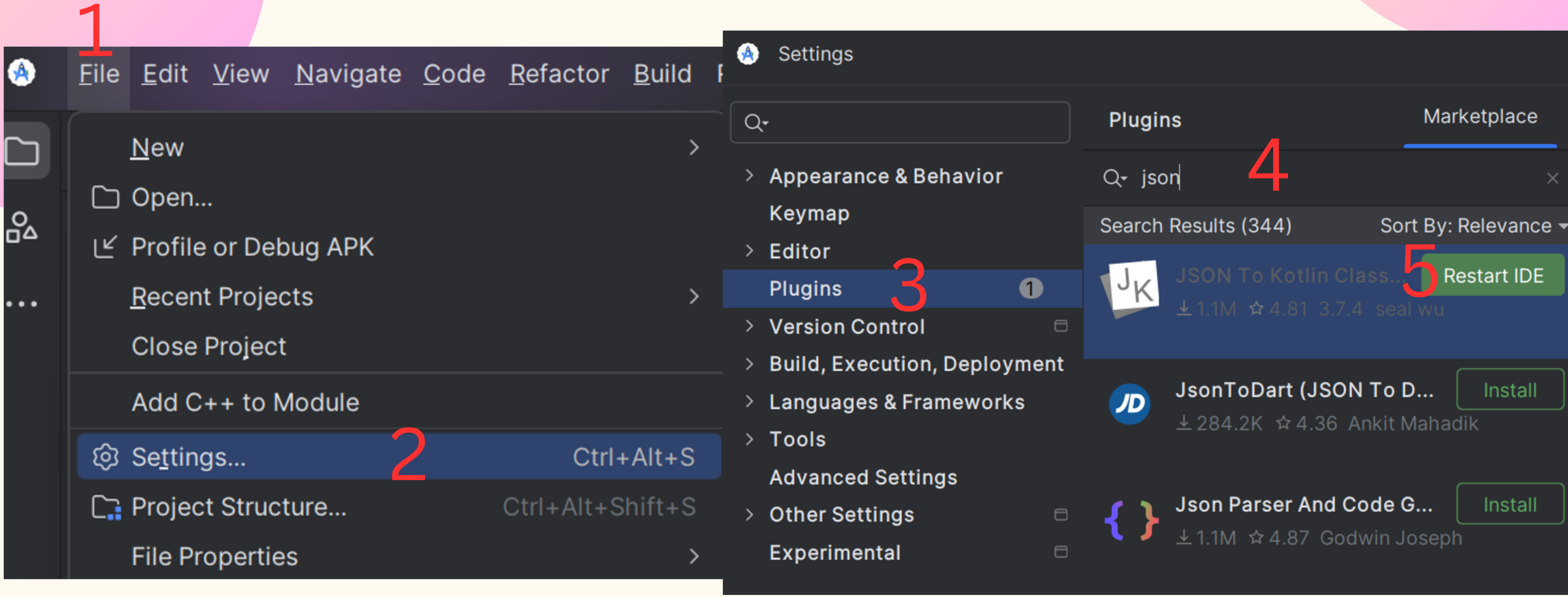
Data class

to automiate this we can donwload pluign make this automatic just add json object will create class for you

how to open plugin : menu bar → File →Settings→Select Plugins from the left panel.



Retrofit


1
File
Edit
View
Navigate
Code
Refactor
Build
New
Open...
Profile or Debug APK
Recent Projects
Close Project
Add C++ to Module
Settings...
Ctrl+Alt+S
2
Project Structure...
Ctrl+Alt+Shift+S
File Properties
Settings
Appearance & Behavior
Keymap
Editor
Plugins
1
3
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Other Settings
Experimental
Plugins
Marketplace
json
4
Search Results (344)
Sort By: Relevance
JSON To Kotlin Class...
Restart IDE
5
1.1M 4.81 3.7.4 seal wu
JsonToDart (JSON To D...
Install
284.2K 4.36 Ankit Mahadik
Json Parser And Code G...
Install
1.1M 4.87 Godwin Joseph

Retrofit

API Service Interface

Instead of manually writing networking logic, you just declare functions, and Retrofit takes care of:

- Making the request
- Parsing the response
- Handling the URL & parameters

Retrofit

API Service Interface

How we declare functions ?

- suspend
- the HTTP method (GET, POST…),
- the endpoint path (e.g., /meals/{id}),
- parameters (like @Query or @Path),
- the expected response type.

Retrofit

API Service Interface

```
interface ApiService {
    @GET("meals/{id}")
    suspend fun getMealById(@Path("id") id: Int): Response<Meal>
}
```

```
@GET("users")
suspend fun getUsers(@Query("page") page: Int): Response<List<User>>
```

Now i can fetch data :

return type

Response<Meal>

my fetch code :

```
val response=API.retrofitService.getmeal("100","534bb78d5aaf4b")
```

then response variable of type response object

Now i can fetch data :

then response variable of type response object

so it has attribut like

- response.code() → returns HTTP status code (200, 404).
- response.errorBody() → returns the error response body if the request failed.
- response.body() → returns the success response body (your data object of data class) if the request succeeded.

Now i can fetch data :

But where to write my fetch code ?

Activity

Now i can fetch data :

What happens if I write my API fetch code directly in the Activity (main thread)?

UI freezing, crash

Now i can fetch data :

Okay, what if I use a background thread (like Coroutine) inside the Activity to fetch data? **it works fine**

True—it works... BUT

What happens if I rotate the screen while fetching?”

Now i can fetch data :

What happens if I rotate the screen while fetching?”

The Activity is destroyed and recreated → so the Coroutine (or background thread) tied to the old Activity is lost.
The fetch process restarts → another API call triggers → data duplication, wasted network, inconsistent state.”

VIEW MODEL

view model

what is view model ?

“ViewModel is lifecycle-aware and survives configuration changes (like rotation), it is part of MVVM .

It keeps the background job and fetched data even when UI is recreated.

- Keeps fetching process alive across Activity/Fragment destruction
- Holds data → no duplicate calls
- UI can access same data without reloading

view model

To use viewModelScope with coroutines, you need to add this dependency—don’t forget it!

lifecycleViewModelKtx="2.8.4"

androidx-lifecycle-viewmodel-ktx={group ="androidx.lifecycle" , name="lifecycle-viewmodel-ktx" , version.ref="lifecycleViewModelKtx"}

view model

```
class myviewmodel(): ViewModel() {
    fun addone()
    {
        viewModelScope.launch {
            val response=API.retrofitService.getmeal( mealName: "100", apiKey: "534bb7
            if(response.code()==200)
            {
            }else
            {
                Log.d( tag: "tamer", msg: "error in fetch ${response.code()}")
            }
        }
    }
```

view model

first we make

lateinit var viewmodel : name of your view model

why? to allow you to use it in any place in activity or fargemnt

in activty

we create instance in oncreate

val viewModel =

ViewModelProvider(this).get(MyViewModel::class.java)

view model

in Activity

we create instance in oncreate

viewModel = ViewModelProvider(this).get(ViewModelName::class.java)

in Fragment

we create instance in oncreate

- viewModel = ViewModelProvider(requireActivity()).get(ViewModelName::class.java)
- ViewModelProvider(this).get(ViewModelName::class.java)

view model

What’s the difference?

- requireActivity() → shares ViewModel with the Activity (good for shared data between fragments). will live even fragment destroy
  Lives as long as the Activity lives.

- this → ViewModel is unique to that Fragment (not shared).
  Lives as long as the Fragment lives.

view model

But how i will return the Data use async with primative value ???

```
var nameResult: String = ""

fun addOne() {
    viewModelScope.async {
        val response = API.retrofitService.getmeal("100", "534bb78d5aaf4bb19fa40fd720c45781")
        if (response.code() == 200) {
            if (response.body() != null) {
                nameResult = response.body()?.name.toString()
            }
        } else {
            Log.d("tamer", "error in fetch ${response.code()}")
        }
    }
```

view model

Can anyone guess what will happen if I call addOne() in the Activity and then write textView.text = nameResult right after? what value will be inside it

```
var nameResult: String = ""


fun addOne() {
    viewModelScope.async {
        val response = API.retrofitService.getmeal("100", "534bb78d5aaf4bb19fa40fd720c457
        if (response.code() == 200) {
            if (response.body() != null) {
                nameResult = response.body()?.name.toString()
            }
```

LIVE DATA

Live Data

what is Live data ?

LiveData is a data holder that is observable and lifecycle-aware.

It automatically updates the UI when the data changes.

It avoids memory leaks because it respects the lifecycle of activities/fragments.

Live Data

what is Live data ?

```
rivate var _name=MutableLiveData<String?>()
val name: LiveData<String?> = _name
fun addone()
{
    viewModelScope.launch {
        val response=API.retrofitService.getmeal( mealName: "100", apiKey: "534bb78d5
        if(response.code()==200)
        {
            if(response.body()!==null)
            {
                _name.postValue(response.body()?.name.toString())
            }
        }else
```

Live Data

why we have 2 Live data ?

- LiveData → read-only → you can only observe it.

- MutableLiveData → can be changed → you can set/update the value. (should be private to follows (encapsulation & separation of concerns))

Live Data

what is observing ?

if name live data change the value in .it even same string it will do what inside observer .

it is iterator of name(read only)

```
viewmodel.name.observe(viewLifecycleOwner)
{
    text.text=it.toString()
}
```

Thank You