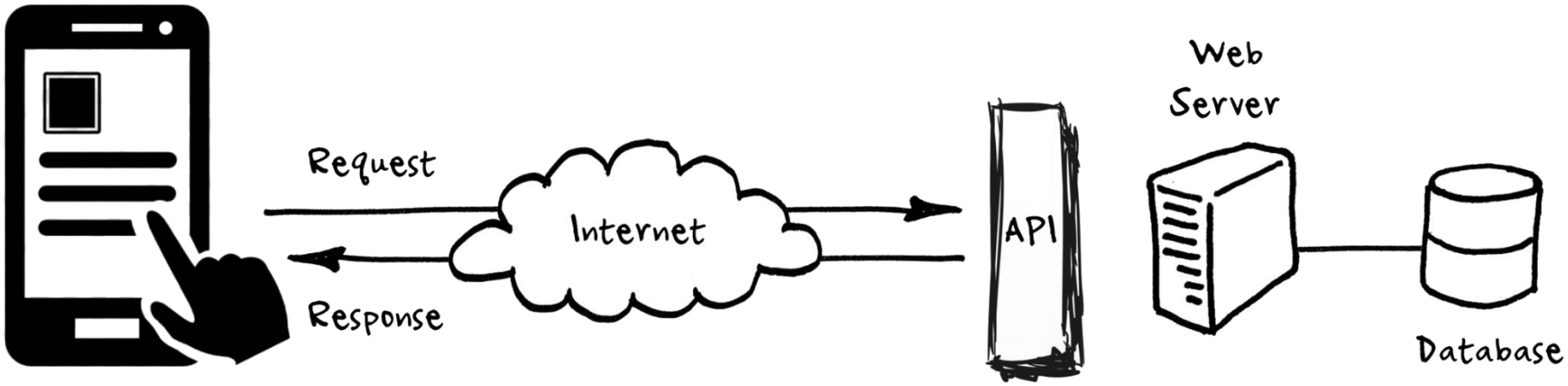
Calling Web API using Retrofit & Coroutines




WWW

Web and HTTP

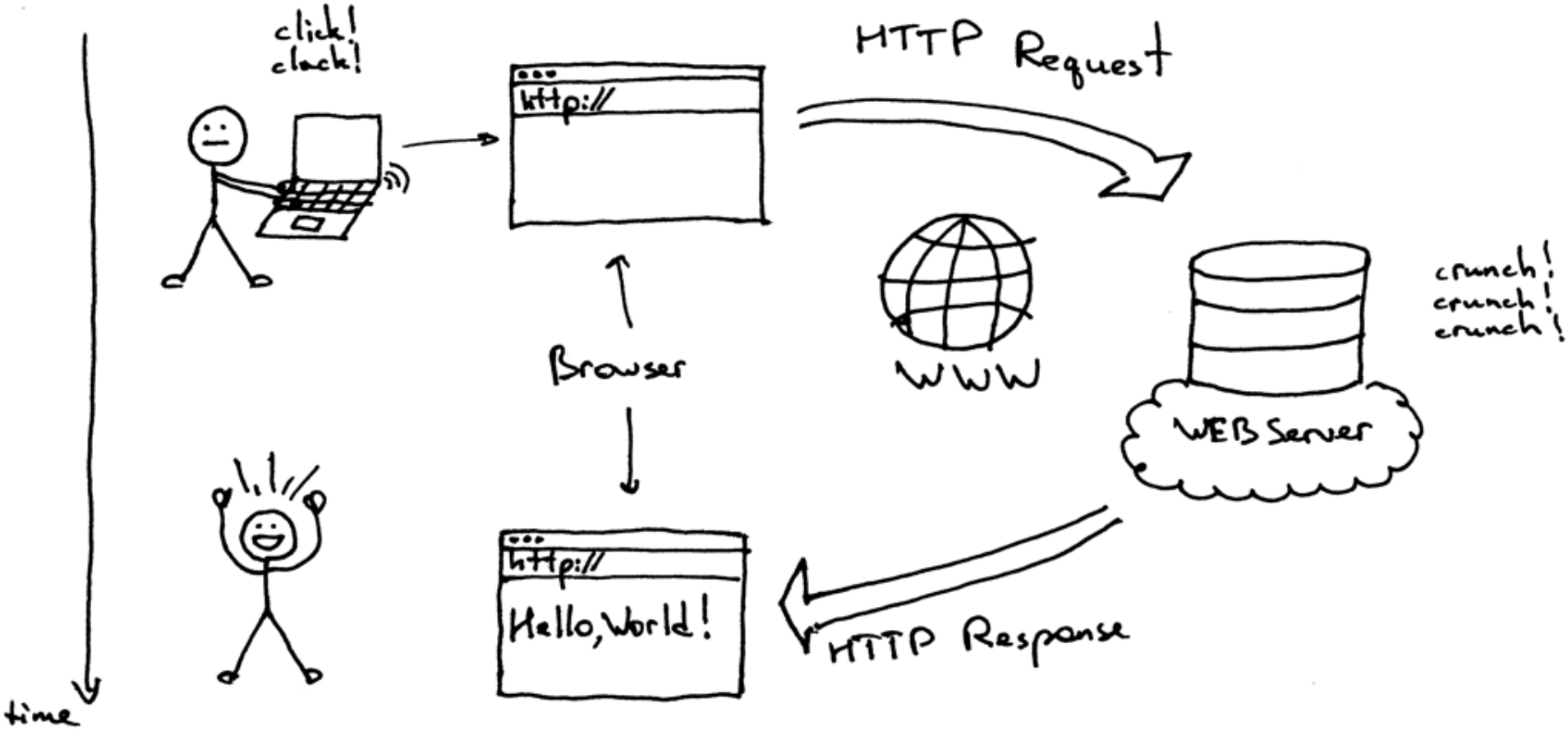

GET /index.html
HTTP/1.1
HTTP/1.1 200 OK
"Welcome to our
Web site!"

What is Web?

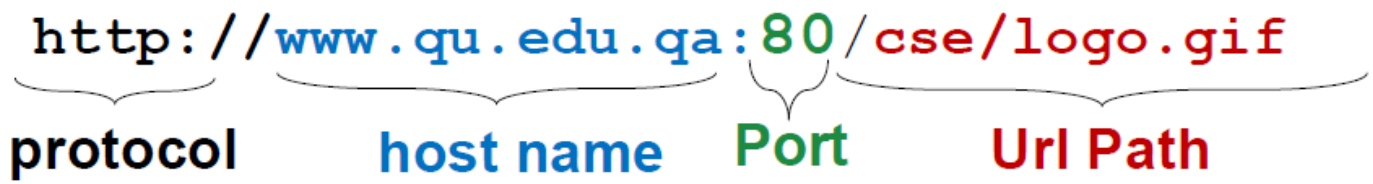
- Web = **global distributed system of interlinked resources accessed over the Internet using the HTTP protocol**
  - Consists of set of **resources** located on different servers:

    HTML pages, images, videos and other resources
  - Resources have unique **URL** (Uniform Resource Locator) address
  - Accessed through standard **HTTP** protocol
- The Web has a Client/Server architecture:
  - **Web browser** / **Mobile App** requests resources (using HTTP protocol) and displays them
  - **Web server** sends resources in response to requests (using HTTP protocol)

How the Web Works?



Uniform Resource Locator (URL)



- URL is a formatted string, consisting of:
  - **Protocol** for communicating with the server (e.g., http, https, ...)
  - **Name of the server or IP** address plus port (e.g. qu.edu.qa:80, localhost:8080)
  - **Path of a resource** (e.g. **/ceng/index.html**)
  - **Parameters** aka **Query String** (optional), e.g.

    https://www.google.com/search**?q=qatar%20university**

2

Web API
(aka Web Services)

Back

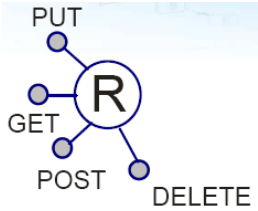
What is a Web API

What is a Web API?

- Web API = Web accessible Application Programming Interface accessible via HTTP to allow programmatic access to applications
  - Also known as Web Services.
  - Can be accessed by a broad range of clients including browsers and mobile devices
- Web API is a web service that accepts requests and returns **structured data** (JSON in most cases)

  - Programmatically accessible at a particular URL
  - You can think of it as a Web page returning JSON instead of HTML
- Major goal = **interoperability between heterogeneous systems**

Web Services Principles



- **Resources have unique address** (**nouns**) i.e., a URI

e.g., http://example.com/customers/123

- **Can use a Uniform Interface** (**verbs**) to access them:

  - HTTP verbs: GET, POST, PUT, and DELETE

- **Resource has representation(s)** (**data format**)

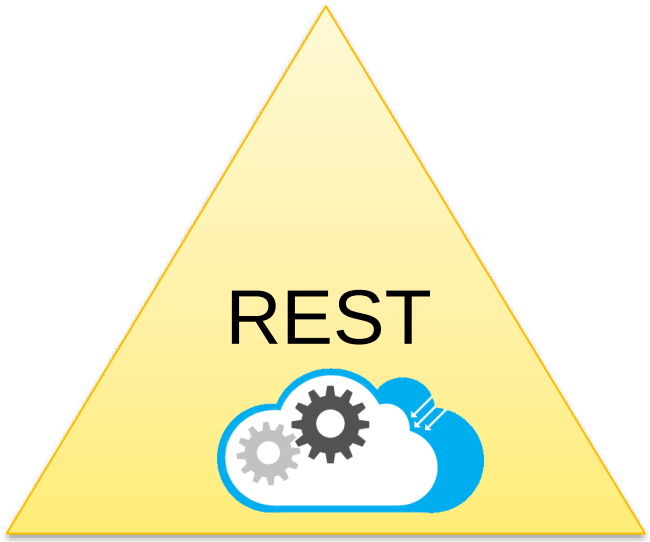
  - A resource can be in a variety of data formats: JSON, XML, RSS..

Resources

- The key abstraction in REST is a **resource**
- A resource is a conceptual mapping to a set of entities
  - Any **information that can be named can be a resource**: a document or image, a temporal service (e.g. "today's weather in Doha"), a collection of books and their authors, and so on

REST Services Main Concepts

Nouns (Resources)
e.g., http://example.com/employees/12345



Verbs
e.g., GET, POST

Representations
e.g., XML, JSON

Naming Resources

- REST uses URL to identify resources

> Dedicated **api** path is recommended for better organization

  - http://localhost/**api**/books/
  - http://localhost/api/books/ISBN-0011
  - http://localhost/api/books/ISBN-0011/authors

  - http://localhost/api/classes
  - http://localhost/api/classes/cmps356
  - http://localhost/api/classes/cs356/students

- As you traverse the **path** from more generic to more specific, you are navigating the data

Example CRUD (Create, Read, Update and Delete) API that manages books

- Create a new book
  - **POST** /books
- Retrieve all books
  - **GET** /books
- Retrieve a particular book
  - **GET** /books/:id
- Replace a book
  - **PUT** /books/:id
- Update a book
  - **PATCH** /books/:id
- Delete a book
  - **DELETE** /books/:id

Representations

Two main formats:

- **JSON**

```
{
    code: 'cmp123',
    name: 'Web Development'
}
```

- **XML**

```
<course>
    <code>cmp123</code>
    <name>Web Development</name>
</course>
```

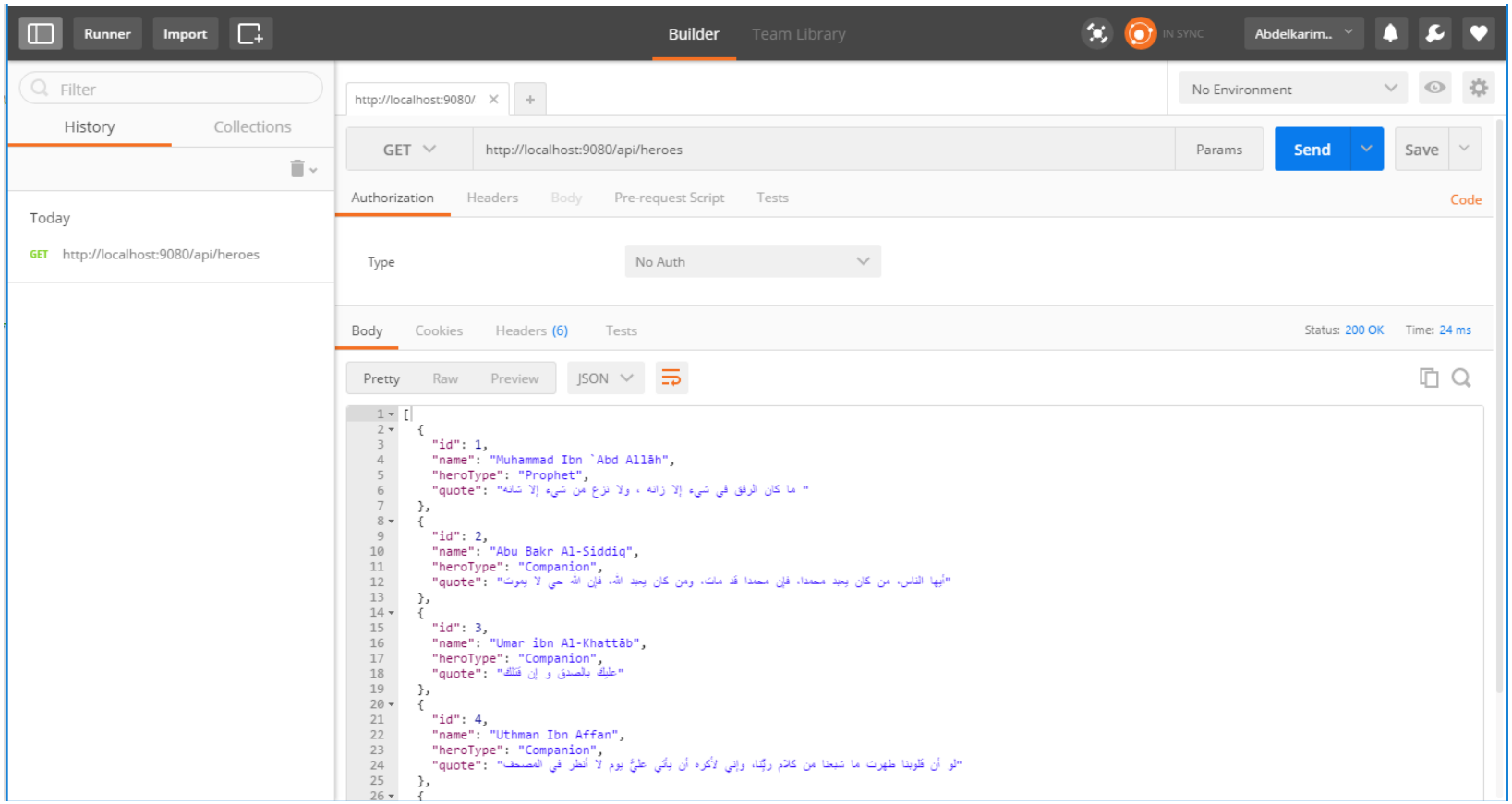
HTTP Verbs

- Represent the actions to be performed on resources
- Retrieve a representation of a resource: **GET**
- Create a new resource:
  - Use **POST** when the server decides the new resource URI
    - Post is not repeatable
  - Use **PUT** when the client decides the new resource URI
    - Put is repeatable
- **PUT** is typically used for update
- Delete an existing resource: **DELETE**
- Get metadata about an existing resource: **HEAD**
- Get which of the verbs the resource understands: **OPTIONS**

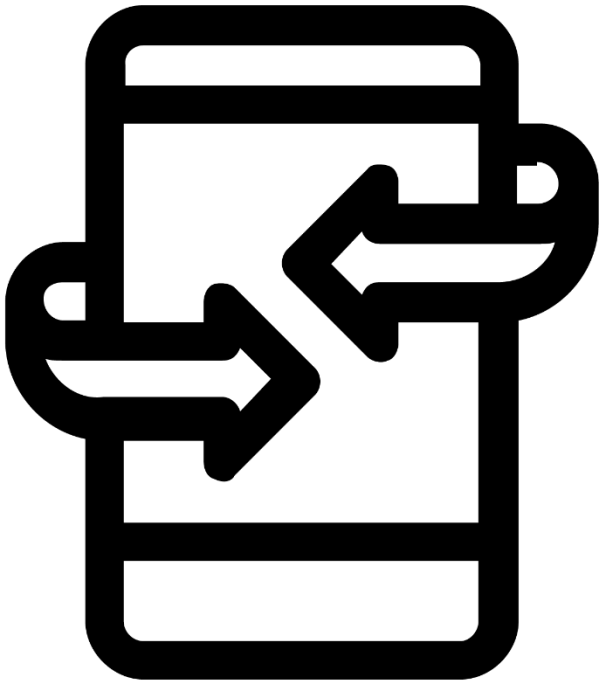
Testing REST Services

- Using Postman to test Web API

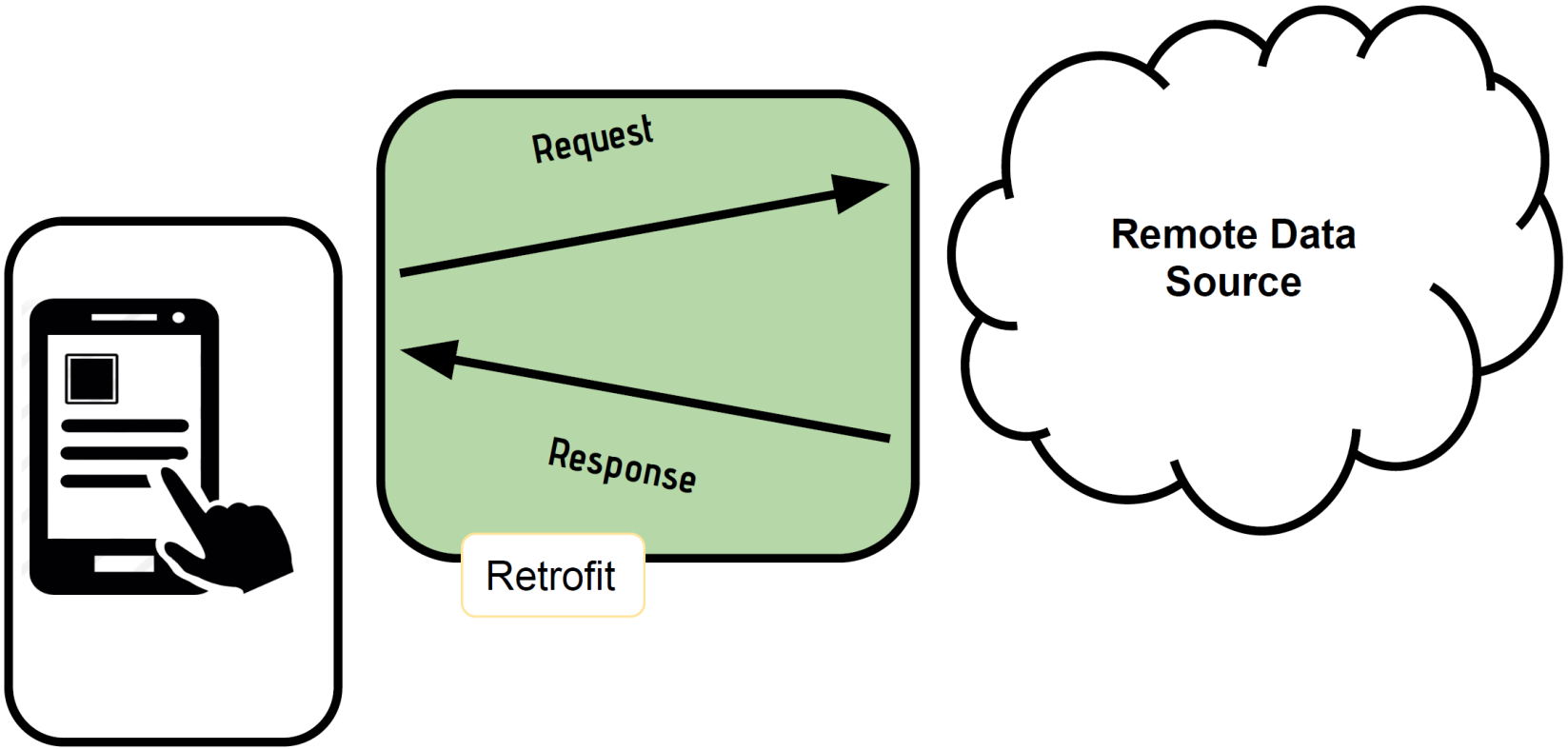
https://www.postman.com/downloads/



Retrofit



Retrofit Role



Retorfit

- HTTP client for Android & Java

1) Define your Service API using a Kotlin interface

  - Define how requests are created and sent and how their responses are read and parsed
  - Method and parameter annotation customize request

2) Use Retrofit.Builder to generate the client object to call the remove Web API

1. Define Service API

```
interface CountryService {
    @GET("all")
    suspend fun getCountries() : List<Country>

    @GET("name/{name}")
    suspend fun getCountry(@Path("name") name: String) : Country

    @POST("countries")
    suspend fun addCountry(@Body country: Country)
}
```

2. Generate the client object to call the remove Web API

```
private const val BASE_URL = "https://restcountries.eu/rest/v2/"
private val contentType = "application/json".toMediaType()

val jsonConverterFactory = Json { ignoreUnknownKeys = true
    coerceInputValues = true }.asConverterFactory(contentType)

val countryService by lazy {
    Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(jsonConverterFactory)
        .build()
        .create(CountryService::class.java)
}
```

Method Annotations

- Methods are annotated based on:
  - **HTTP Verb** – GET, POST, PUT, DELETE used to access the service
  - **URL Path** – e.g., /users
- Method annotations **maps** an **HTTP Verb** (e.g., GET or POST) + a **URI Path** (like /country/qatar) to a **method**
  - E.g., getAcount method maps to Get verb and /country/{name} Url path

```
@GET("countries/{name}")
suspend fun getCountry(@Path("name") name: String) : Country
```


22

Path Parameters

- Named **path parameters** can be added to the URL path. E.g., **/students/{id}**
  - E.g., if you have the path **/students/{id}**, then the "id" property is available to the method using @Path("id") studentId: Int

```
@GET("students/{id}")
suspend fun getStudents(@Path("id") studentId: Int) : Student
```


23

Query Parameters

- Named **query parameters** can be added to the URL path after a ? E.g., **/posts?sortBy=createdOnDate**
- Query parameters are often used for **optional** parameters (e.g., optionally specifying the property to be used to sort of results)
- **@Query** annotation is used to map method parameters to for each query parameter in the URL path
  - If you have the path /posts?sortBy=createdOnDate, then the "**sortBy**" property is available using:

```
@GET("posts")
suspend fun getPosts(@Query("sortBy") sortBy: String) : List<Post>
```


24

Working with a Request Body

- Annotate the method parameter with **@Body** annotation to send the send the parameter as request body in the HTTP request

```
@POST("countries")
suspend fun addCountry(@Body country: Country)
```