Android Lecture #9

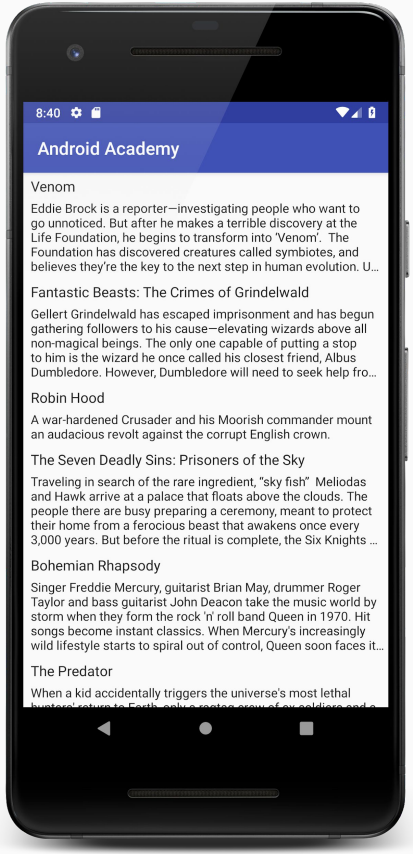
Networking

Agenda

- HTTP(s)
- Network requests
- 3rd party libraries
- Image loading
- Network Profiling

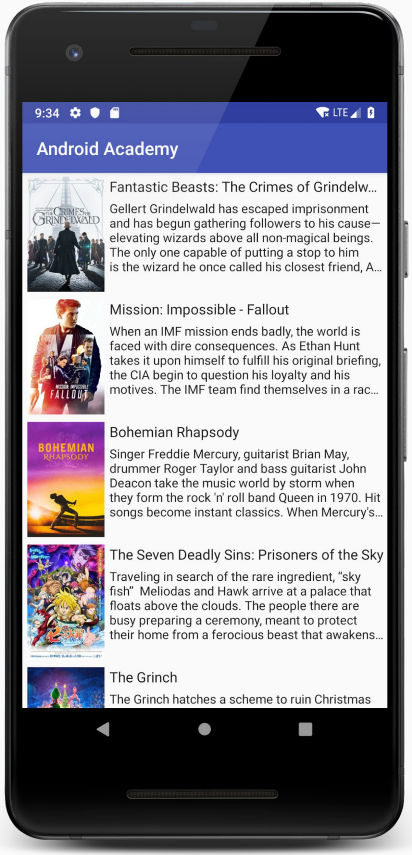
Goals:

1. Load data from the server



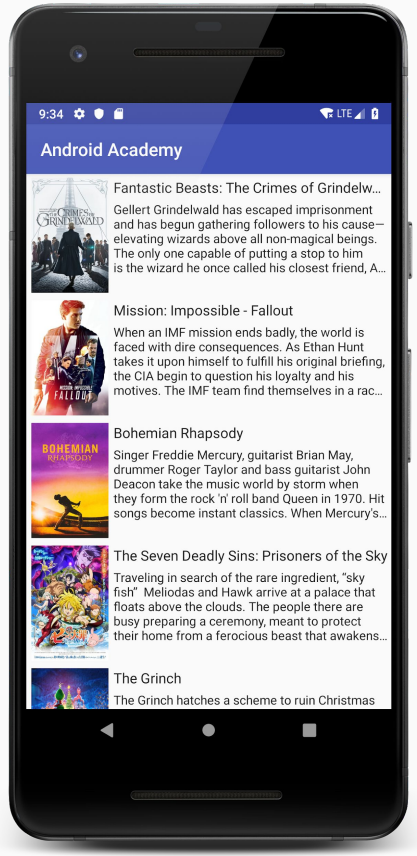
Goals:

1. Load data from the server
2. Load images



Goals:

1. Load data from the server
2. Load images
3. Understand what the hell we did in steps 1 and 2



How to perform a network request?

Permissions

Ask permission

Pleeeeeeeaaaase…..?

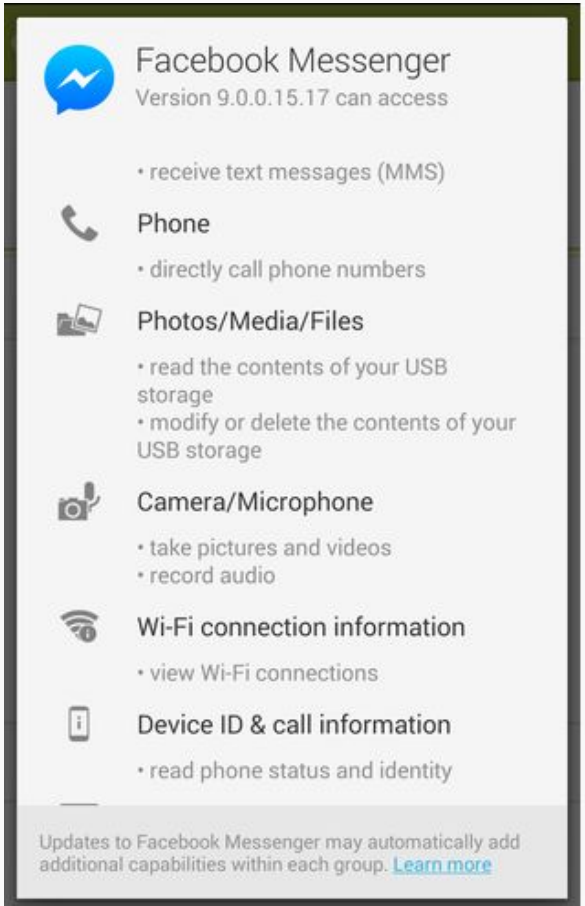
Permissions

Apps must explicitly **request access**
to resources and data outside their sandbox.

Permissions

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.academy.fundamentals">

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

    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />


</manifest>
```

API 22 : Upon Install



Normal

Internet
Bluetooth
Fingerprint

Granted by OS

Click for documentation

Permission Types

Dangerous

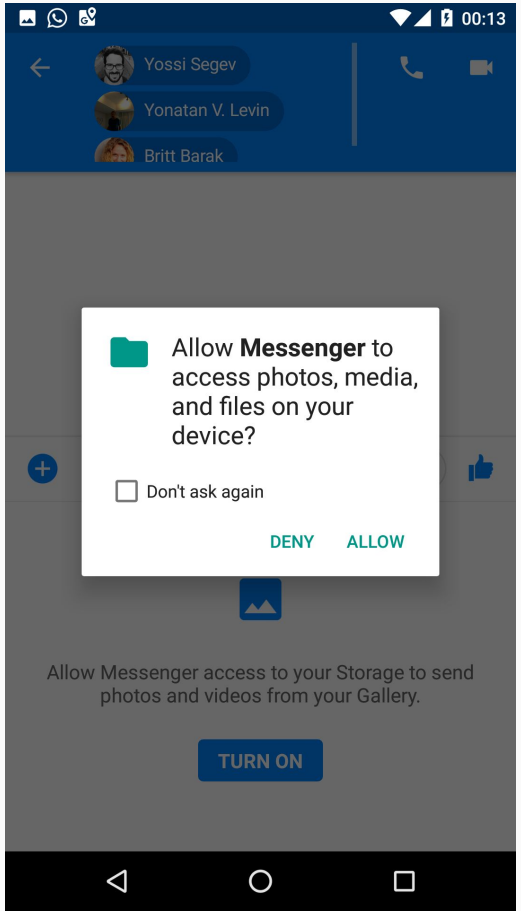
Contacts
Location
Storage

Granted by user

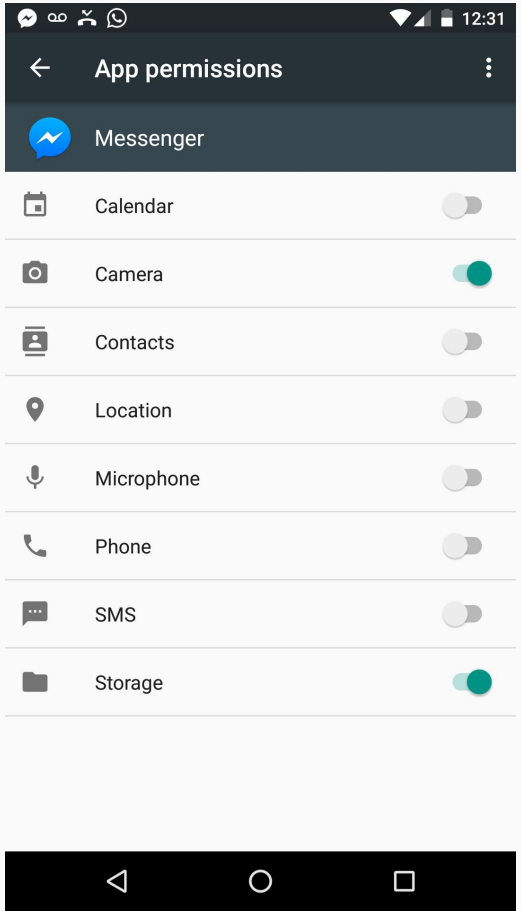
Click for documentation

API 23+ : Runtime

developer.android.com/training/permissions/requesting.html



API 23+ : Runtime



developer.android.com/training/permissions/requesting.html

Questions?

Client

Request

Response

Server

HTTP

- Hypertext Transfer Protocol.
- Stateless protocol.

HTTP Headers

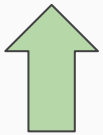
Headers

additional information about the request or response

```
Content-Type : application/json
User-Agent : Android
```

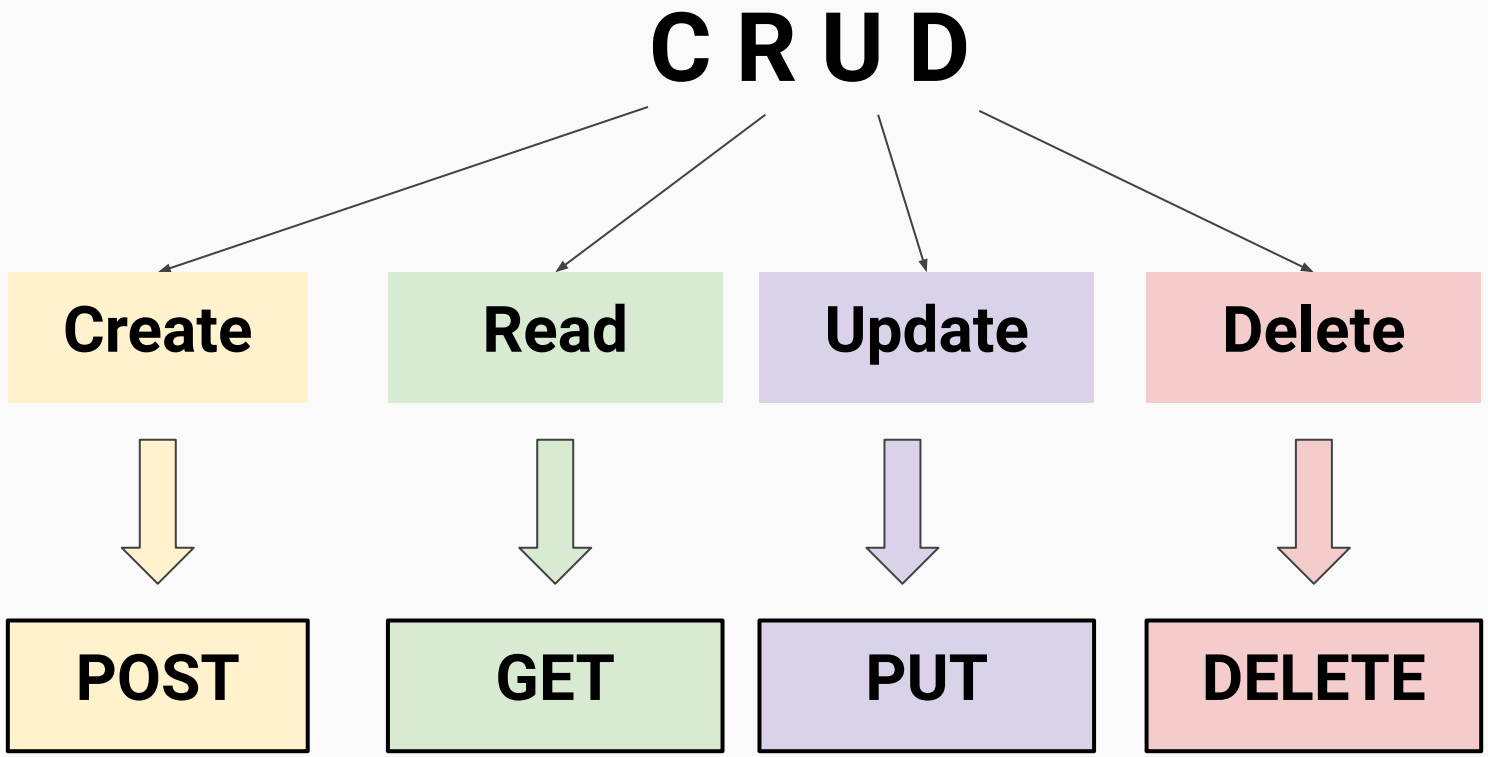
Request

Verb + URL + (Optional Payload)



What is this verb?

HTTP Request Verbs - CRUD concept



HTTP Verbs
(partial list)

HTTP Verbs

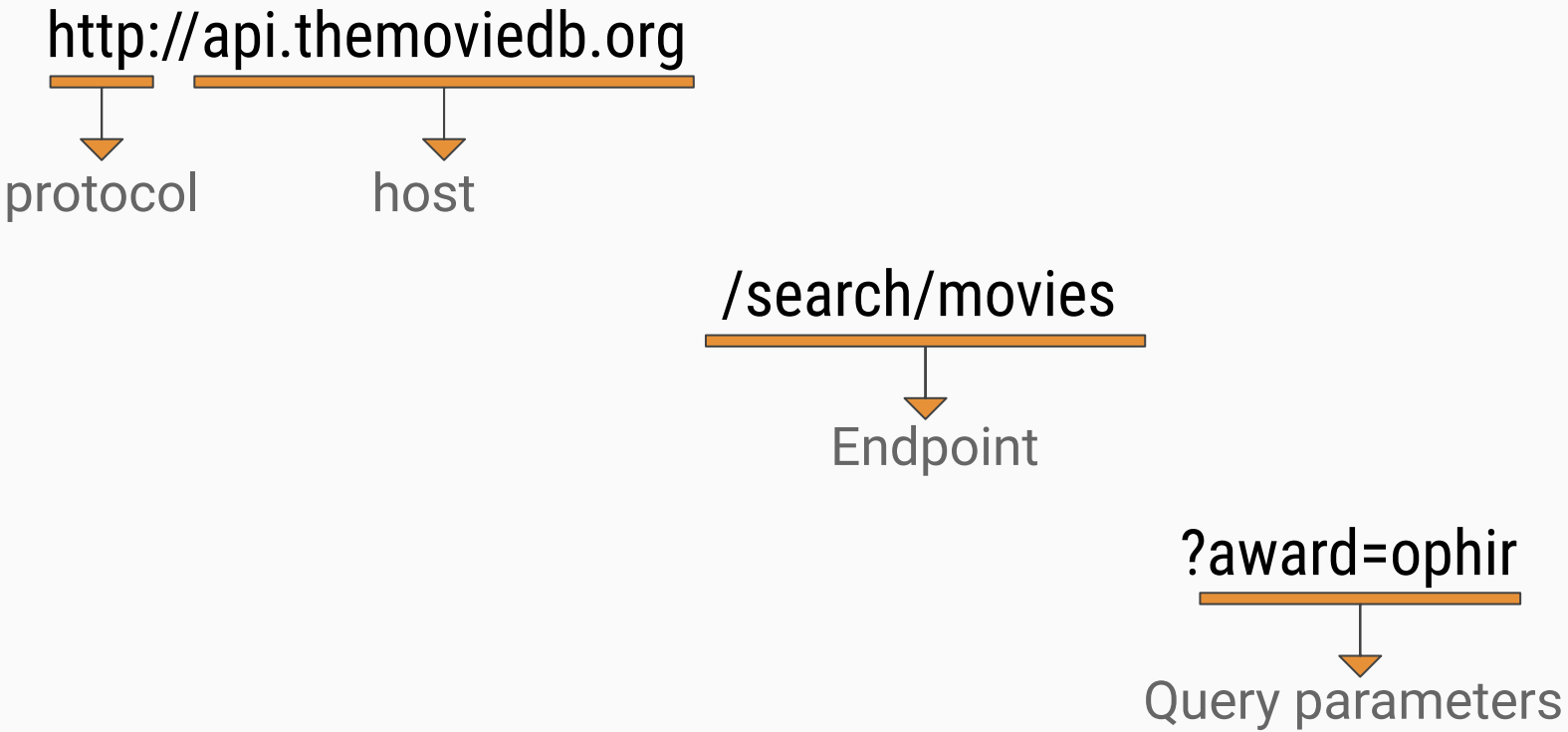
- **GET**: fetch an existing resource, by the URL's info.
- **POST**: create a new resource, by the request's payload.
- **PUT**: update an existing resource, by the request's payload.
- **DELETE**: delete an existing resource.

HTTP Request URL

URL (Universal Resource Locator)

http://api.themoviedb.org/search/movies?award=ophir

URL (Universal Resource Locator)



Response

Status Code + Payload (body)

Status Code

1xx: Informational Messages - (Not very useful)

2xx: Successful - 200 OK, 204 No Content

3xx: Redirection - Additional actions, Commonly go to another url.

4xx: Client Error - 404 Not Found, 400 Bad Request

5xx: Server Error - 503 Service Unavailable

What language does the server speak?

Payload

XML <>

JSON {}

HTML <>

XML <>

JSON {}

HTML <>

Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
   <movie>
      <language>English</language>
      <name>Mister Potato</name>
      <year>2017</year>
   </movie>
</result>
```

Payload

XML <>

JSON {}

HTML <>

```
<table>
<tr><td>language</td><td>name</td>
<td>year</td></tr>
<tr><td>English</td>
<td>Mister Potato</td>
<td>2017</td></tr>
</table>
```

Payload

XML <>

JSON {}

HTML <>

```
{
    "movie": {
        "name": "Mister Potato",
        "year": 2017,
        "language": "English"
    }
}
```

JSON

JSON


```
{
    "vote_count": 17,
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "poster_path": "/2hQEXyb6DXc3jAJzSBUxmIDf7jW.jpg",
    "overview": "Machines have taken over, but left humans",
    "spoken_language": {
        "iso": "en",
        "name": "English"
    },
    "release_date": "2006-06-24"
}
```


https://api.themoviedb.org/3/search/movie?api_key=some_key&language=en-US&query=android&page=1&include_adult=false

JSON

```
{
    "vote_count": 17,
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "poster_path": "/2hQEXyb6DXc3jAJzSBUxmIDf7jW.jpg",
    "overview": "Machines have taken over, but left humans",
    "spoken_language": {
        "iso": "en",
        "name": "English"
    },
    "release_date": "2006-06-24"
}
```

JS O N

JavaScript Object Notation

(Key : Value)

JSON

```
{
    "vote_count": 17,
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "poster_path": "/2hQEXyb6DXc3jAJzSBUxmIDf7jW.jpg",
    "overview": "Machines have taken over, but left humans",
    "spoken_language": {
        "iso": "en",
        "name": "English"
    },
    "release_date": "2006-06-24"
}
```

JS O N

JavaScript Object Notation

(Key : (Key : Value))

JSON - breakdown

{

}

JSON

Curly brackets marks

start and end of an Object.

https://api.themoviedb.org/3/search/movie?api_key=some_key&language=en-US&query=android&page=1&include_adult=false

```
{
    "vote_count": 17,
    "id": 17940,
}
```

JSON

Key (always a String)

:

Value

(number/String/boolean/null/

nested structure)

Key-value pairs are separated by

commas

https://api.themoviedb.org/3/search/movie?api_key=some_key&language=en-US&query=android&page=1&include_adult=false

JSON

- **Most widely used** data format for web
- **Fast** - lightweight syntax and easy parsing makes for faster responses
- **Diverse** - can represent multiple data types (even heavy ones as audio/video)
- **Readable** - easily consumed by javascript, human readable
- **Language independent** - text format, has two structures: key-value, array

What dictates payload format?

- **Client type:** Web browser, Smartphones , Hardware etc.
- **Client-Server Architectural style**

Server-Client Architecture/Model

Defines:

- **A set of rules** for server / client communication

Common Web Service Architectures

REST (json/xml)

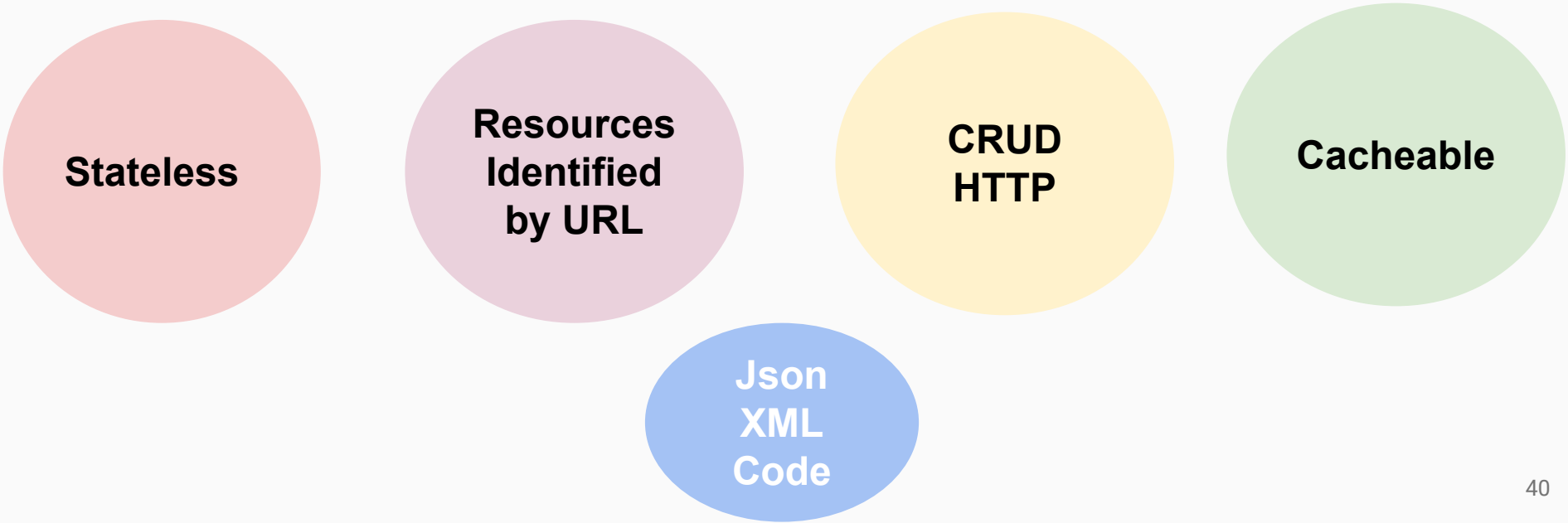
SOAP (xml)

GRAPHQL (json)

RPC [legacy]

REST - widely used in Web API's

REpresentation State Transfer



Questions?

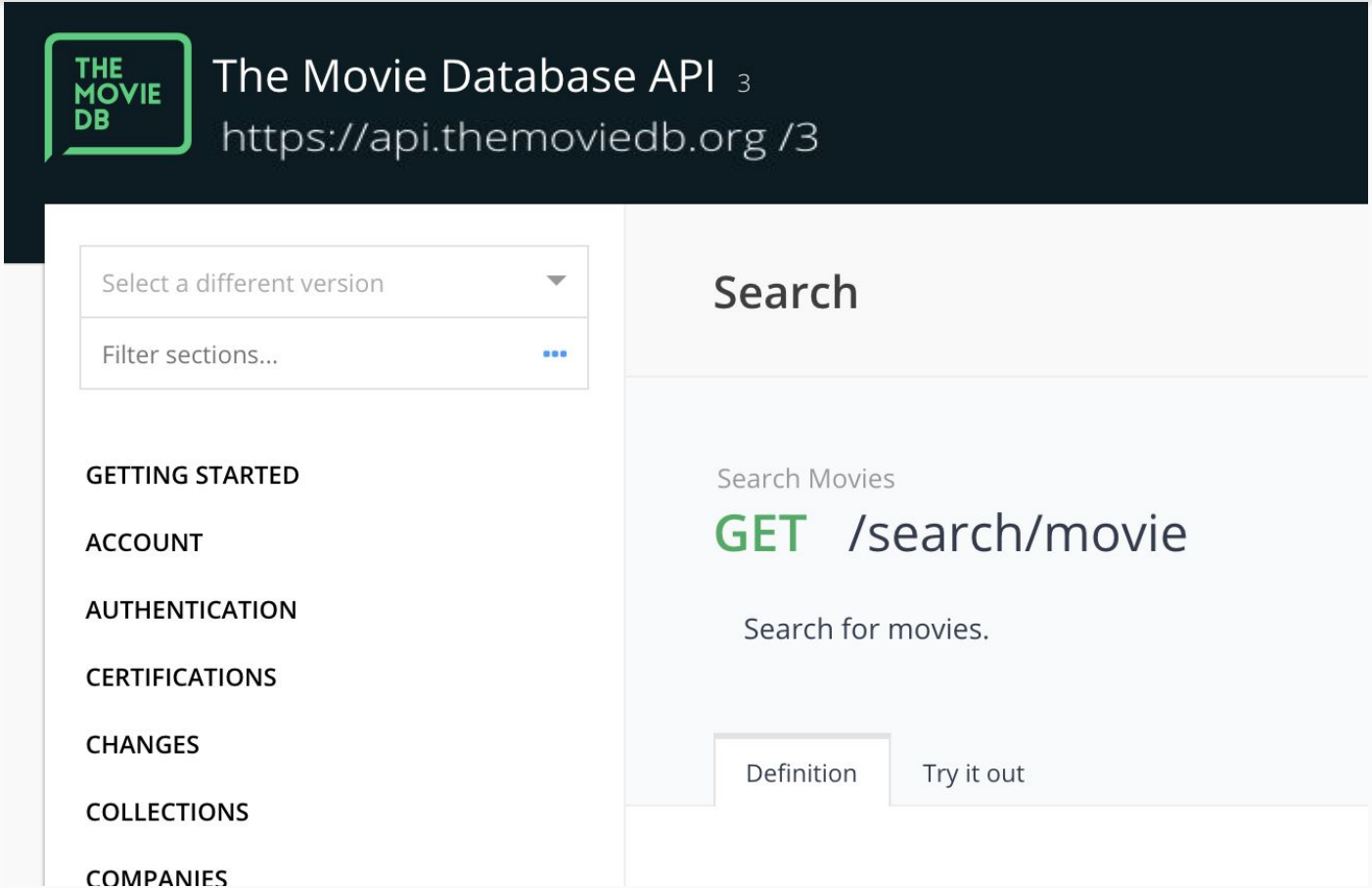
API

Where do we get
The data from?

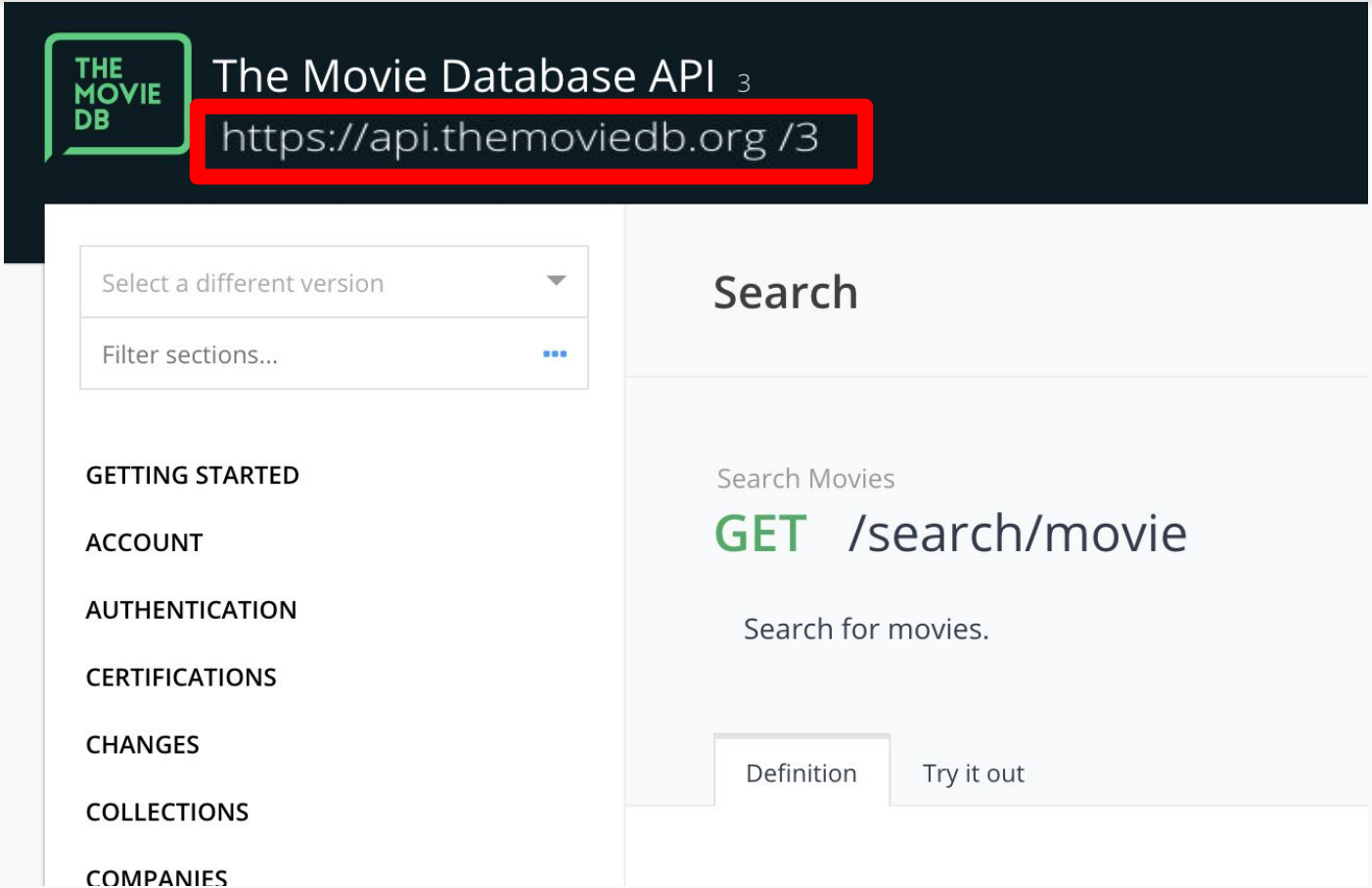
http://??????????????

API Services

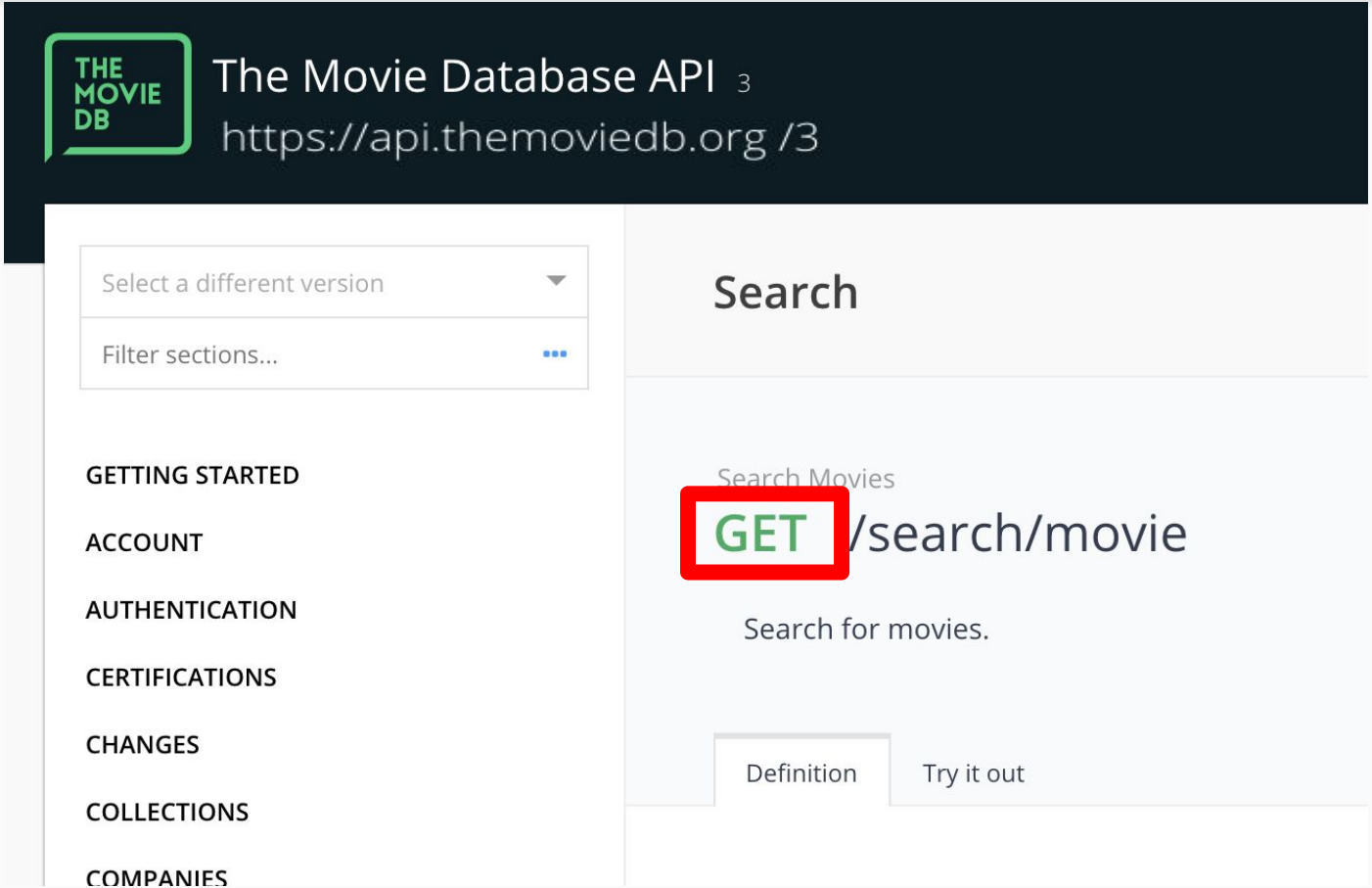
API - documentation



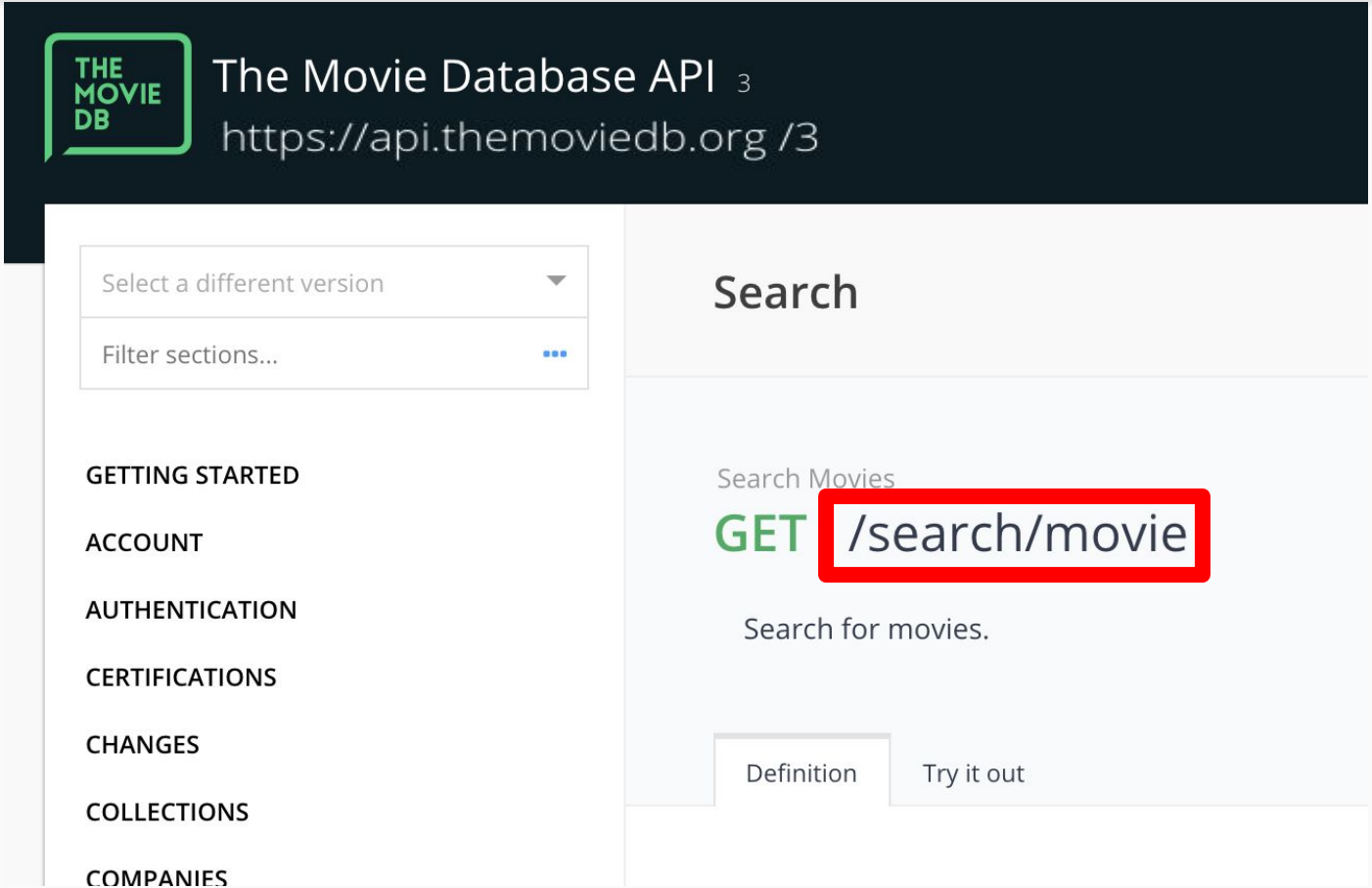
API - documentation



API - documentation



API - documentation



API - documentation

Query String

api_key	string	**default:** <<api_key>>	**required**
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. **minLength:** 2 **pattern:** ([a-z]{2})-([A-Z]{2}) **default:** en-US	optional
query	string	Pass a text query to search. This value should be URI encoded. **minLength:** 1	**required**
page	integer	Specify which page to query. **minimum:** 1 **maximum:** 1000 **default:** 1	optional
include_adult	boolean	Choose whether to inlcude adult (pornography) content in the results. **default**	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase. **pattern:** ^[A-Z]{2}$	optional
year	integer		optional
primary_release_year	integer		optional

API - documentation

Query String

api_key	string	default: <<api_key>>	required
		it. minLength: 2 pattern: ([a-z]{2})-([A-Z]{2}) default: en-US	
query	string	Pass a text query to search. This value should be URI encoded. minLength: 1	required
page	integer	Specify which page to query. minimum: 1 maximum: 1000 default: 1	optional
include_adult	boolean	Choose whether to inlcude adult (pornography) content in the results. default	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase. pattern: ^[A-Z]{2}$	optional
year	integer		optional
primary_release_year	integer		optional

Required

API - documentation

Query String

api_key	string	**default:** <<api_key>>	**required**
language	string	Pass a ISO 639-1 value to display translated data for the fields that support it. **minLength:** 2 **pattern:** ([a-z]{2})-([A-Z]{2}) **default:** en-US	optional
query	string	Pass a text query to search. This value should be URI encoded. **minLength:** 1	**required**
page	integer	Specify which page to query. **minimum:** 1 **maximum:** 1000 **default:** 1	optional
include_adult	boolean	Choose whether to inlcude adult (pornography) content in the results. **default**	optional
region	string	Specify a ISO 3166-1 code to filter release dates. Must be uppercase. **pattern:** ^[A-Z]{2}$	optional
year	integer		optional
primary_release_year	integer		optional

Required

API - Build our HTTP request URL

base url (host)

```
https://api.themoviedb.org/3
```

API - Build our HTTP request URL

base url (host)

```
https://api.themoviedb.org/3
```

+

path (endpoint)

```
/search/movies
```

API - Build our HTTP request URL

base url (host)

```
https://api.themoviedb.org/3
```

+

path (endpoint)

```
/search/movies
```

+

query

```
?year=2017&query=android&api_key=some_key
```

Let’s write our first request

What we will need:

- **Api service -** https://api.themoviedb.org/3 - check!
- **HTTP Client -** to consume the data on our device

Writing our client code - Pure Kotlin way

HttpUrlConnection

```
public VideosList getMovies() {
     String contentAsString = null




}
```

HttpUrlConnection

```
public VideosList getMovies() {
      String contentAsString = null
      URL url = new URL("https://api.themoviedb.org/3/search/movies");









}
```

HttpUrlConnection

```
public VideosList getMovies() {
     String contentAsString = null
     URL url = new URL("https://api.themoviedb.org/3/search/movies");
     HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();









}
```

HttpUrlConnection

```
public VideosList getMovies() {
     String contentAsString = null
     URL url = new URL("https://api.themoviedb.org/3/search/movies");
     HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
     urlConnection.setConnectTimeout(15000 /* milliseconds */);
```

```
}
```

HttpUrlConnection

```
public VideosList getMovies() {
     String contentAsString = null
     URL url = new URL("https://api.themoviedb.org/3/search/movies");
     HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
     urlConnection.setConnectTimeout(15000 /* milliseconds */);
     urlConnection.setRequestMethod("GET");









}
```

HttpUrlConnection

```
public VideosList getMovies() {
      String contentAsString = null
      URL url = new URL("https://api.themoviedb.org/3/search/movies");
      HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
      urlConnection.setConnectTimeout(15000 /* milliseconds */);
      urlConnection.setRequestMethod("GET");
      // Starts the query
      urlConnection.connect();









}
```

HttpUrlConnection

```
public VideosList getMovies() {
     String contentAsString = null
     URL url = new URL("https://api.themoviedb.org/3/search/movies");
     HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
     urlConnection.setConnectTimeout(15000 /* milliseconds */);
     urlConnection.setRequestMethod("GET");
     // Starts the query
     urlConnection.connect();
     int responseCode = urlConnection.getResponseCode();









}
```

HttpUrlConnection

```
public VideosList getMovies() {
      String contentAsString = null
      URL url = new URL("https://api.themoviedb.org/3/search/movies");
      HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
      urlConnection.setConnectTimeout(15000 /* milliseconds */);
      urlConnection.setRequestMethod("GET");
      // Starts the query
      urlConnection.connect();
      int responseCode = urlConnection.getResponseCode();
      if (response == 200) {

      }



}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();




        }




}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);


        }




}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();



        }


}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {

            }

        }

}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {
                stringBuilder.append(line);
            }

        }

}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {
                stringBuilder.append(line);
            }
            contentAsString = stringBuilder.toString();

        }
```

HttpUrlConnection

```
public VideosList getMovies() {
...
     if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {
                stringBuilder.append(line);
            }
            contentAsString = stringBuilder.toString();
            inputStreamReader.close();
            bufferedReader.close();
        }
```

HttpUrlConnection

```
public VideosList getMovies() {
...
    try {
        ...
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

HttpUrlConnection

```
public VideosList getMovies() {
...
    try {
        ...
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
    }

}
```

HttpUrlConnection

```
public VideosList getMovies() {
     URL url = new URL("https://api.themoviedb.org/3/search/movies");
     HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
     urlConnection.setConnectTimeout(15000 /* milliseconds */);
     urlConnection.setRequestMethod("GET");
     // Starts the query
     urlConnection.connect();
     int response = urlConnection.getResponseCode();
     InputStream is = urlConnection.getInputStream();
     // Convert the InputStream into a string -> to class
     String contentAsString = readIt(is);



}
```

HttpUrlConnection

```
public VideosList getMovies() {
    URL url = new URL("https://api.themoviedb.org/3/search/movies");
    HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
    urlConnection.setConnectTimeout(15000 /* milliseconds */);
    urlConnection.setRequestMethod("GET");
    // Starts the query
    urlConnection.connect();
    int response = urlConnection.getResponseCode();
    InputStream is = urlConnection.getInputStream();
    // Convert the InputStream into a string -> to class
    String contentAsString = readIt(is);
    // convert somehow to VideoList Object
    return new VideoList(contentAsString);
}
```

HttpUrlConnection

```
public VideoResult getMovies() {
    String contentAsString = null;
    HttpURLConnection urlConnection = null;
    try {
        URL url = new URL("https://api.themoviedb.org/3/search/movies");
        urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setConnectTimeout(15000 /* milliseconds */);
        urlConnection.setRequestMethod("GET");
        // Starts the query
        urlConnection.connect();
        int response = urlConnection.getResponseCode();
        if (response == 200) {
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String line;
            StringBuilder stringBuilder = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {
                stringBuilder.append(line);
            }
            contentAsString = stringBuilder.toString();
            inputStreamReader.close();
            bufferedReader.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
    }
    return new VideoResult(contentAsString);
}
```

Can we do better?


#SchittsCreek
JOHNNY

Third-party networking libraries

OkHttp - 3rd party networking library

An HTTP client for Android, Kotlin, and Java

OkHttp - 3rd party networking library

- Faster to code
- HTTP Connection optimizations
- Request optimizations (socket sharing, threading)
- Response caching to avoid repeated requests
- Easy Api supports both blocking and async calls

Add OKHttp

build.gradle

```
dependencies {
        implementation("com.squareup.okhttp3:okhttp:4.3.1")

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }


}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }


}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        return new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

OkHttp - 3rd party networking library

```
public VideosList getMovies() {

    OkHttpClient client = new OkHttpClient();
    Request.Builder builder = new Request.Builder();
    builder.url("https://api.themoviedb.org/3/search/movies");
    Request request = builder.build();
    try {
        Response response = client.newCall(request).execute();
        return new VideosList(response.body().string());
    }catch (Exception e){
        e.printStackTrace();
    }

}
```

HttpUrlConnection

```
public class MoviesActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movies);



    }
}
```

HttpUrlConnection

```
public class MoviesActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movies);

        getMovies()

    }
}
```

Pure Kotlin - HttpUrlConnection

#1 Rule in Android development

#1 Rule in Android development

Never **block** the UI Thread

Pure Kotlin - HttpUrlConnection

```
11112  11112  com.lookshot.splash   AndroidRuntime  FATAL EXCEPTION: main
11112  11112  com.lookshot.splash   AndroidRuntime  android.os.NetworkOnMainThreadException
11112  11112  com.lookshot.splash   AndroidRuntime  at android.os.StrictMode$AndroidBlockGuardPolicy.onNetwork(StrictM
                                                    ode.java:1099)
11112  11112  com.lookshot.splash   AndroidRuntime  at java.net.InetAddress.lookupHostByName(InetAddress.java:432)
11112  11112  com.lookshot.splash   AndroidRuntime  at java.net.InetAddress.getAllByNameImpl(InetAddress.java:258)
11112  11112  com.lookshot.splash   AndroidRuntime  at java.net.InetAddress.getAllByName(InetAddress.java:236)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:71)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:50)
11112  1111                                                                                          nection.
11112  1111                                                                                          ool.java
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpConnection.connect(HttpConnection.java:128
                                                    )
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpEngine.openSocketConnection(HttpEngine.jav
                                                    a:308)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpsURLConnectionImpl$HttpsEngine.makeSslConn
                                                    ection(HttpsURLConnectionImpl.java:460)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpsURLConnectionImpl$HttpsEngine.connect(Htt
                                                    psURLConnectionImpl.java:432)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpEngine.sendSocketRequest(HttpEngine.java:2
                                                    82)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpEngine.sendRequest(HttpEngine.java:232)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpURLConnectionImpl.getResponse(HttpURLConne
                                                    ctionImpl.java:273)
11112  11112  com.lookshot.splash   AndroidRuntime  at libcore.net.http.HttpURLConnectionImpl.getInputStream(HttpURLCo
```

NetworkOnMainThreadException

OMG!

Questions?

Response

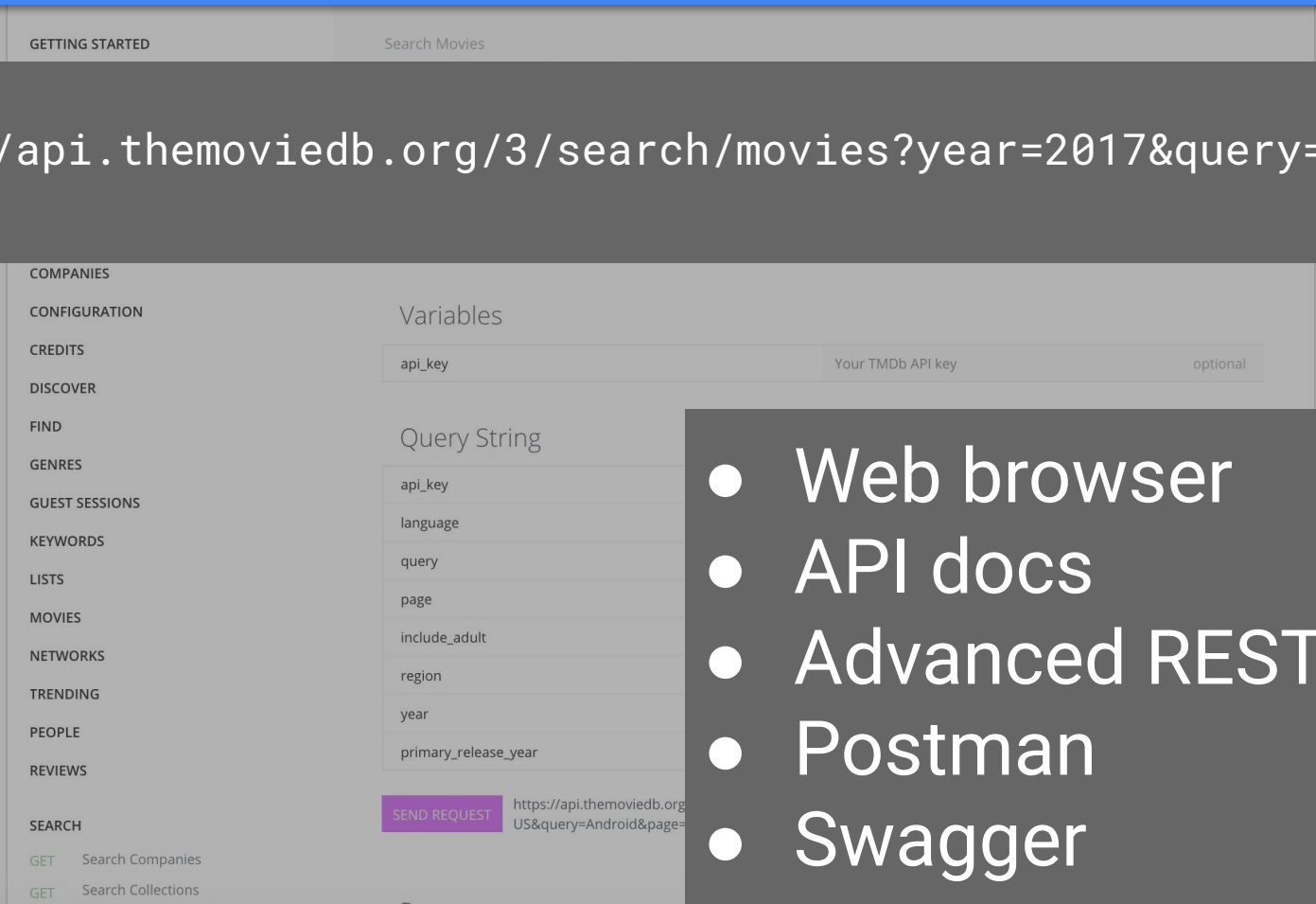
Parse our response

1. Test our api service to get expected response object model

2. Convert our JSON response string into Java class
   that will hold our parsed data

Test our api service to get expected response model

https://api.themoviedb.org/3/search/movies?year=2017&query=android



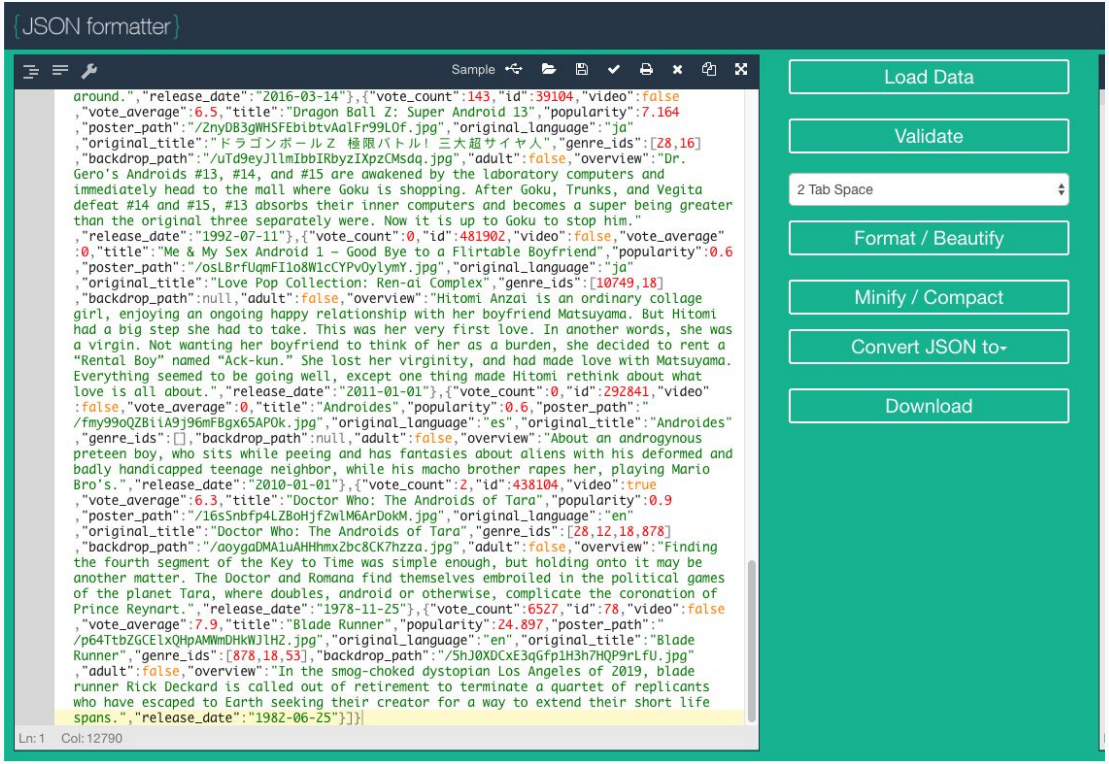
- Web browser
- API docs
- Advanced REST Client
- Postman
- Swagger

Result - JSON payload

{"page":1,"total_results":32,"total_pages":2,"results":[{"vote_count":26,"id":38849,"video":false,"vote_average":5.5,"title":"Android","popularity":2.171,"poster_path":"/jsRWN56VUbwLwSFhE1ler2C4tei.jpg","original_language":"en","original_title":"Android","genre_ids":[878,53],"backdrop_path":"/veTb0hmlRu1kOgAFujNVIbe2Qaj.jpg","adult":false,"overview":"Eccentric scientist Dr. Daniel and his shy assistant Max lead a quiet life on their space station, carrying out illegal research on androids, until they receive an unwelcome visit from three fugitives one of whom is female. Both Dr. Daniel and Max show an interest in her, but one of the other visitors has more sinister intentions.","release_date":"1982-10-15"},{"vote_count":34,"id":249021,"video":false,"vote_average":3.7,"title":"Android Cop","popularity":2.148,"poster_path":"/nju7Uzw9MpRNQxogDL7dMro9Qcs.jpg","original_language":"en","original_title":"Android Cop","genre_ids":[28,878],"backdrop_path":"/A72HB8E3oPjJm2GOBdbgN8diTa1.jpg","adult":false,"overview":"In the year 2045, a Los Angeles Police Department detective and his new android partner enter the Zone, a forbidden section of the city plagued with an unknown disease. There, they discover the source of the illness and attempt to stop it using the android's advanced technology and weaponry.","release_date":"2014-02-04"},{"vote_count":0,"id":429420,"video":false,"vote_average":0,"title":"Android","popularity":0.788,"poster_path":null,"original_language":"en","original_title":"Android","genre_ids":[878],"backdrop_path":null,"adult":false,"overview":"On a lonely spaceship orbiting Neptune, a man has recreated his dead wife and their son as androids. But when the androids begin demanding their freedom, he finds his life suddenly in danger.","release_date":"2018-12-31"},{"vote_count":17,"id":17940,"video":false,"vote_average":4.1,"title":"Android Apocalypse","popularity":1.483,"poster_path":"/2hQEXyb6DXc3jAJzSBUxmIDf7jW.jpg","original_language":"en","original_title":"Android Apocalypse","genre_ids":[878],"backdrop_path":"/9TkBA5IGglDR64b3nWLxNj4CMHy.jpg","adult":false,"overview":"Machines have taken over, but left humans thinking that they are still the ones in charge. The androids need humans because of the human brain fluid; without it the android brains can't work. Until the mad scientist finds out how to make this brain fluid artificially that is.","release_date":"2006-06-24"},{"vote_count":7,"id":111234,"video":false,"vote_average":1.6,"title":"Android Insurrection","popularity":1.13,"poster_path":"/dv5fCQWbYFjwd8wfRu2j3vDztHl.jpg","original_language":"en","original_title":"Android Insurrection","genre_ids":[878],"backdrop_path":"/a1xaL2CIDsy0YjkNWFmT7nHQyzL.jpg","adult":false,"overview":"In the distant future high-tech man colonized the world and leaves all the dirty work of an army of androids and robots."}]}

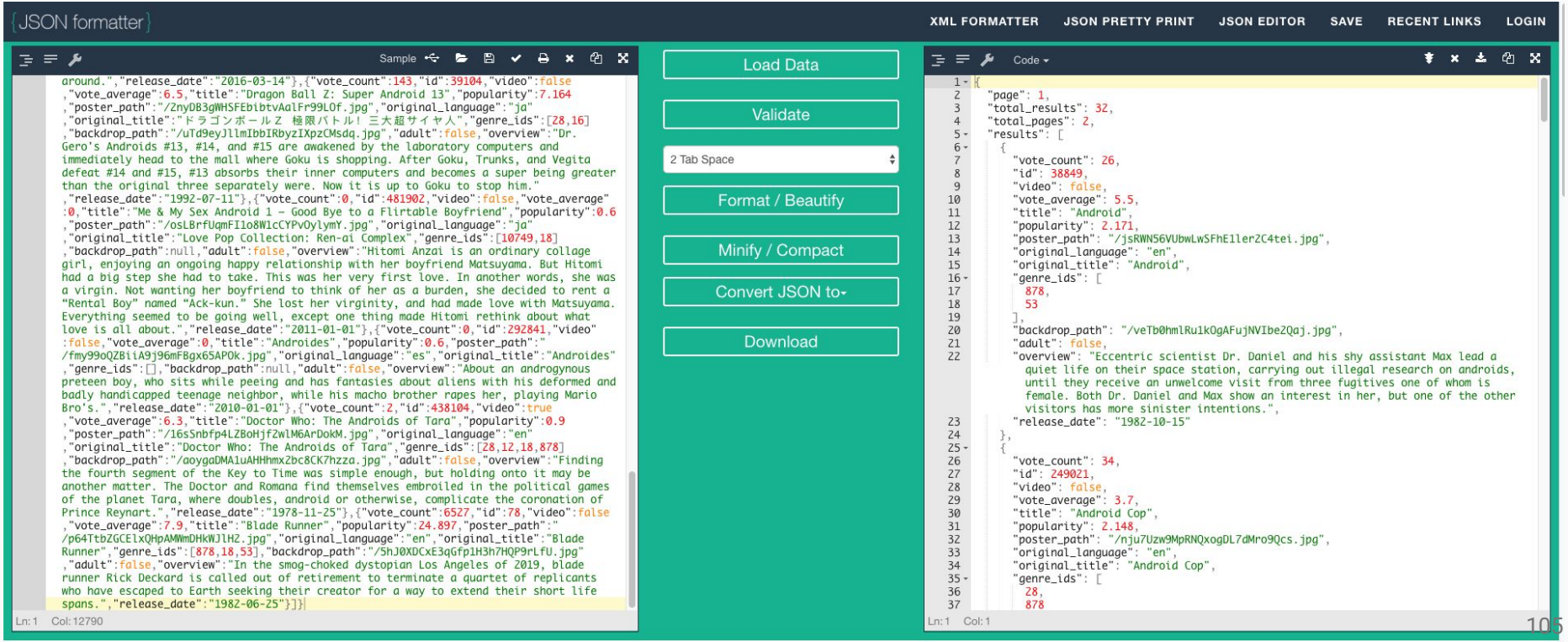
JSON formatting

Json Formatter



JSON formatting

Json Formatter



Creating Class model For JSON

JSON Movie Object

```
{
    "id" : 249021,
    "title" : "Android Cop",
    "overview" : "once upon...”
}
```

Kotlin class model For JSON content

JSON Movie Object

```
{
    "id" : 249021,
    "title" : "Android Cop",
    "overview" : "once upon...”
}
```

Java Movie class

```
class Movie {
    int id;
    String title;
    String overview;
}
```

Kotlin class model For JSON content

JSON Movie Object

```
{
    "id" : 249021,
    "title" : "Android Cop",
    "overview" : "once upon..."
}
```

Java Movie class

```
class Movie {
    int id;
    String title;
    String overview;
}
```

JSON key  Class Field

Kotlin class model For JSON content

JSON Movie Object

```
{
    "id" : 249021,
    "title" : "Android Cop",
    "overview" : "once upon..."
}
```

Java Movie class

```
class Movie {
    int id;
    String title;
    String overview;
}
```

JSON key  Class Field

Kotlin class model For JSON content

JSON Movie Object

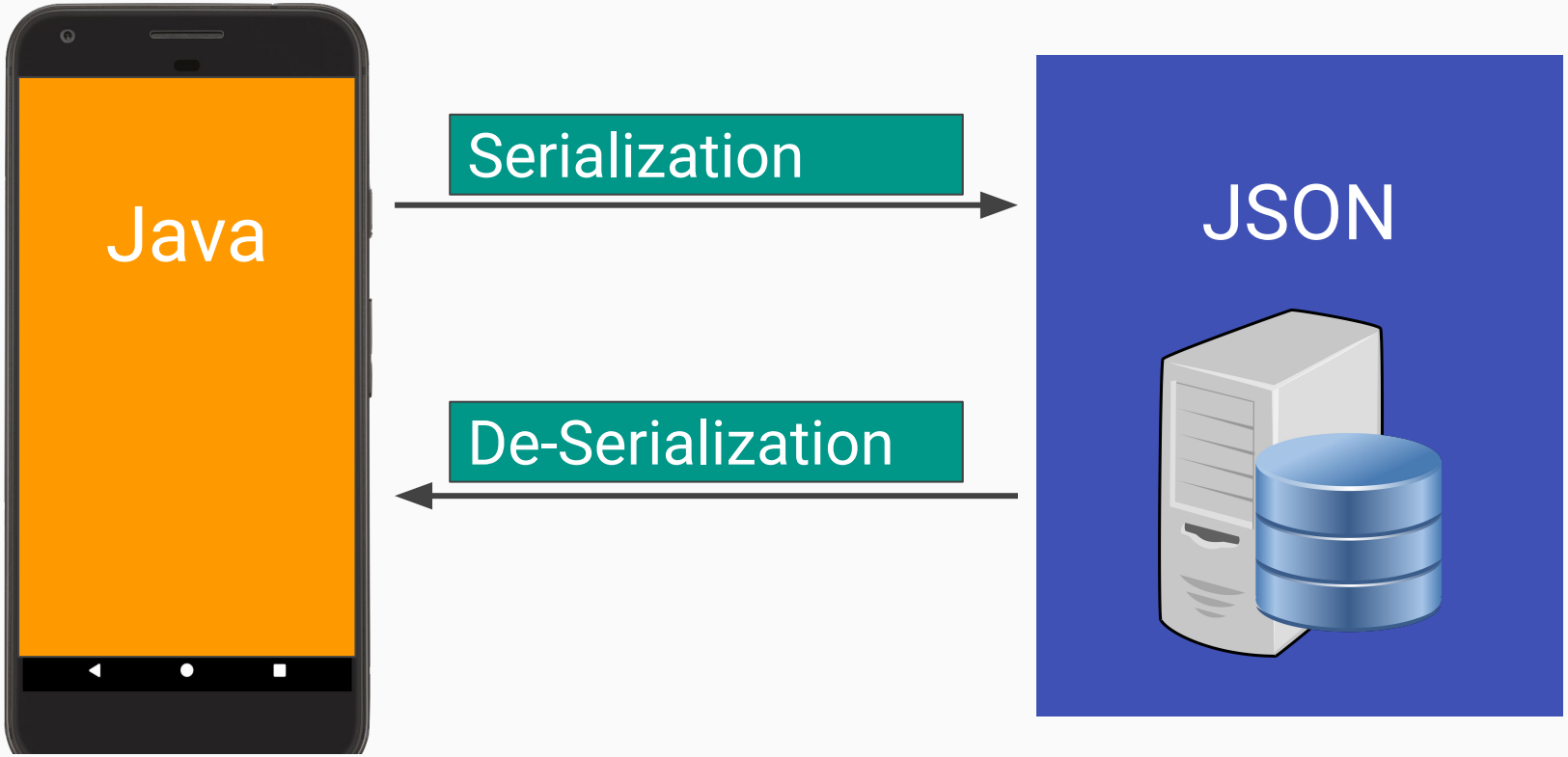
```
{
    "id" : 249021,
    "title" : "Android Cop",
    "overview" : "once upon..."
}
```

Java Movie class

```
class Movie {
    int id;
    String title;
    String overview;
}
```

JSON key  Class Field

Serialization/De-Serialization - from kotlin class to JSON String and vice versa



The Manual Way

The Manual Way

```
Movie movie = new Movie();
```

The Manual Way

```
Movie movie = new Movie();
```

```
JSONObject jsonObject = new JSONObject(jsonString);
```

The Manual Way

```
Movie movie = new Movie();

JSONObject jsonObject = new JSONObject(jsonString);

if (jsonObject.has("title")) {

}
```

The Manual Way

```
Movie movie = new Movie();

JSONObject jsonObject = new JSONObject(jsonString);

if (jsonObject.has("title")) {
    // throws JSONException if “title” does not exists
    movie.title = jsonObject.getString("title");
}
```

The Manual Way

```
Movie movie = new Movie();

JSONObject jsonObject = new JSONObject(jsonString);

if (jsonObject.has("overview")) {
    // throws JSONException if "title" does not exists

}
```

The Manual Way

```
Movie movie = new Movie();

JSONObject jsonObject = new JSONObject(jsonString);

if (jsonObject.has("overview")) {
    // throws JSONException if “overview” does not exists
    movie.overview = jsonObject.getString("overview");
}
```

The Manual Way

```
Movie movie = new Movie();

JSONObject jsonObject = new JSONObject(jsonString);

// returns “” if “overview” does not exists
movie.overview = jsonObject.optString("overview");
```

Parsing JSON Array

```
“results” : [{
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    },{
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }]
```

Parsing JSON Array

```
“results” : [{
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    },{
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }]
```

Parsing JSON Array

Objects array

```
“results” : [{
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English" }
    },{
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English" }
    }]
```

Parsing JSON Array

Objects array

```
“results” : [{
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English" }
    },{
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English" }
    }]
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();

for (int i = 0; i < result.length(); i++) {

}
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();

for (int i = 0; i < result.length(); i++) {
    Movie movie1 = new Movie();




}
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();

for (int i = 0; i < result.length(); i++) {
    Movie movie1 = new Movie();
    JSONObject jsonObject1 = (JSONObject) result.get(i);



}
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();

for (int i = 0; i < result.length(); i++) {
    Movie movie1 = new Movie();
    JSONObject jsonObject1 = (JSONObject) result.get(i);
    movie1.overview = jsonObject1.optString("overview");


}
```

Parsing JSON - array

```
JSONArray result = jsonObject.optJSONArray("results");
List<Movie> movies = new ArrayList<>();

for (int i = 0; i < result.length(); i++) {
    Movie movie1 = new Movie();
    JSONObject jsonObject1 = (JSONObject) result.get(i);
    movie1.overview = jsonObject1.optString("overview");
...
    movies.add(movie1);

}
```

Parsing JSON - JSONException

```
try {
    ...
    ...
    ...
} catch (JSONException e) {
   e.printStackTrace();
}
```

What if we have tens of fields in our JSON?

Serialization/Deserialization library to convert Kotlin Objects into JSON and back

Serialization/Deserialization library to convert Java Objects into JSON and back

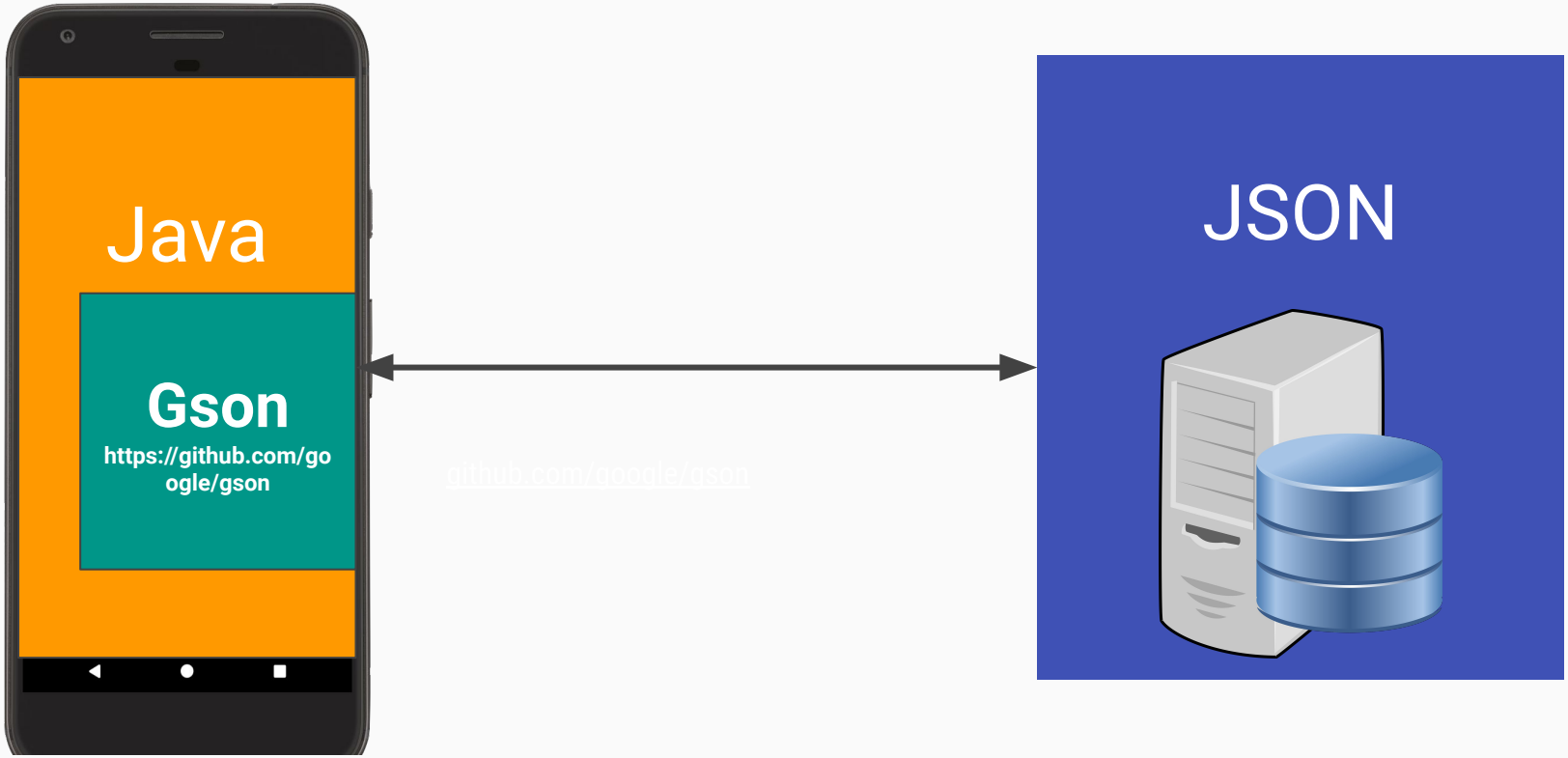
- GSON
- Moshi
- Jackson
- ……

Add GSON

build.gradle

```
dependencies {
        implementation 'com.google.code.gson:gson:2.8.6'
}
```

GSON - 3rd party library for Serialization/De-serialization



github.com/google/gson

JSON to data class model

```
{
    "id" : 249021,
    "title" : "Android Cop"
}
```

JSON to data class model

```
{
    "id" : 249021,
    "title" : "Android Cop"
}
```

```
class Movie {
    int id;
    String title;
}
```

JSON to data class model

```
{
    "id" : 249021,
    "title" : "Android Cop"
}
```

```
class Movie {
    int id;
    String title;
}
```

JSON to data class model

```
{
    "id" : 249021,
    "title" : "Android Cop"
}
```

```
class Movie {
    int id;
    String title;
}
```

GSON - JSON to data class model

```
{
    "id" : 249021,
    "title" : "Android Cop"
}
```

```
class Movie {
    int id;
    String title;
}
```

Questions?

Creating the data model

(data class)

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {

}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {

}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;

}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;

}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;
    String title;


}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;
    String title;


}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;
    String title;


}

class SpokenLanguage {
    String iso;
    String name;
}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;
    String title;
    SpokenLanguage language;

}

class SpokenLanguage {
    String iso;
    String name;
}
```

GSON

```
{
    "id": 17940,
    "title": "Android Apocalypse",
    "popularity": 1.96,
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }
}
```

```
class VideoResult {
    int id;
    String title;
    @SerializedName("spoken_language")
    SpokenLanguage language;
}

class SpokenLanguage {
    String iso;
    String name;
}
```

GSON

```
“results” : [{
    "id": 17940,
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    },{
    "id": 14142340,
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }]
}
```

```
class VideoResult {
    int id;
    String title;
    @SerializedName("spoken_language")
    SpokenLanguage language;
}
```

GSON

```
“results” : [{
    "id": 17940,
    "title": "Android Apocalypse",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    },{
    "id": 14142340,
    "title": "Master Android",
    "spoken_language":{
        "iso": "en",
        "name": "English"
    }]
}
```

```
class VideosListResult {
    @SerializedName("results")
    List<VideoResult> videos;
}

class VideoResult {
    int id;
    String title;
    @SerializedName("spoken_language")
    SpokenLanguage language;
}
```

JSON string -> Java Object

```
String resultJsonString = "{"results":[
    {"id":17940,"title":"Android Apocalypse",
    "spoken_language":{"iso":"en","name":"English"}},
    {"id":234432340,"title":"Master Android",
    "spoken_language":{"iso":"en","name":"English"}}]}";
```

JSON string -> Java Object

```
String resultJsonString = "{"results":[
    {"id":17940,"title":"Android Apocalypse",
    "spoken_language":{"iso":"en","name":"English"}},
    {"id":234432340,"title":"Master Android",
    "spoken_language":{"iso":"en","name":"English"}}]}";

Gson gson = new Gson();
```

JSON string -> Java Object

```
String resultJsonString = "{"results":[
    {"id":17940,"title":"Android Apocalypse",
    "spoken_language":{"iso":"en","name":"English"}},
    {"id":234432340,"title":"Master Android",
    "spoken_language":{"iso":"en","name":"English"}}]}";

Gson gson = new Gson();
VideosListResult videosListResult =
                    gson.fromJson(resultJsonString, VideosListResult.class);
```

Java Object -> JSON string

```
VideoResult videoResult; // initialized with cool data
Gson gson = new Gson();
```

Java Object -> JSON string

```
VideoResult videoResult; // initialized with cool data
Gson gson = new Gson();


String resultFromJavaObject = gson.toJson(videoResult);
```

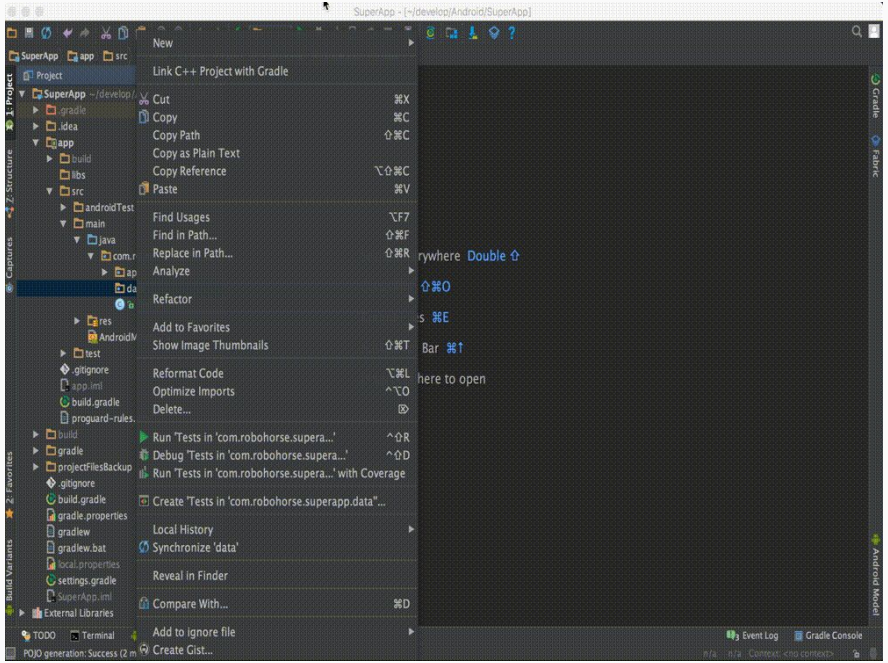
TIP!

Use
POJO generator
plugin!

TIP!

i.e. RoboPOJOGenerator

https://github.com/robohorse/RoboPOJOGenerator



Questions?

HTTP Request - Recap

1. Create HttpURLConnection
2. Convert the inputstream to string (or other type?)
3. Threading
4. Parse the string to Java object
5. Handle errors
6. What if we have 20 request?

Retrofit

REST Client for Android

REpresentational State Transfer

http://square.github.io/retrofit/
https://futurestud.io/blog/retrofit-getting-started-and-android-client

Retrofit

- Type safe OkHttp client
- Threading
- Caching
- Error handling
- Serialization / Deserialization
- And more...

How does Retrofit work?

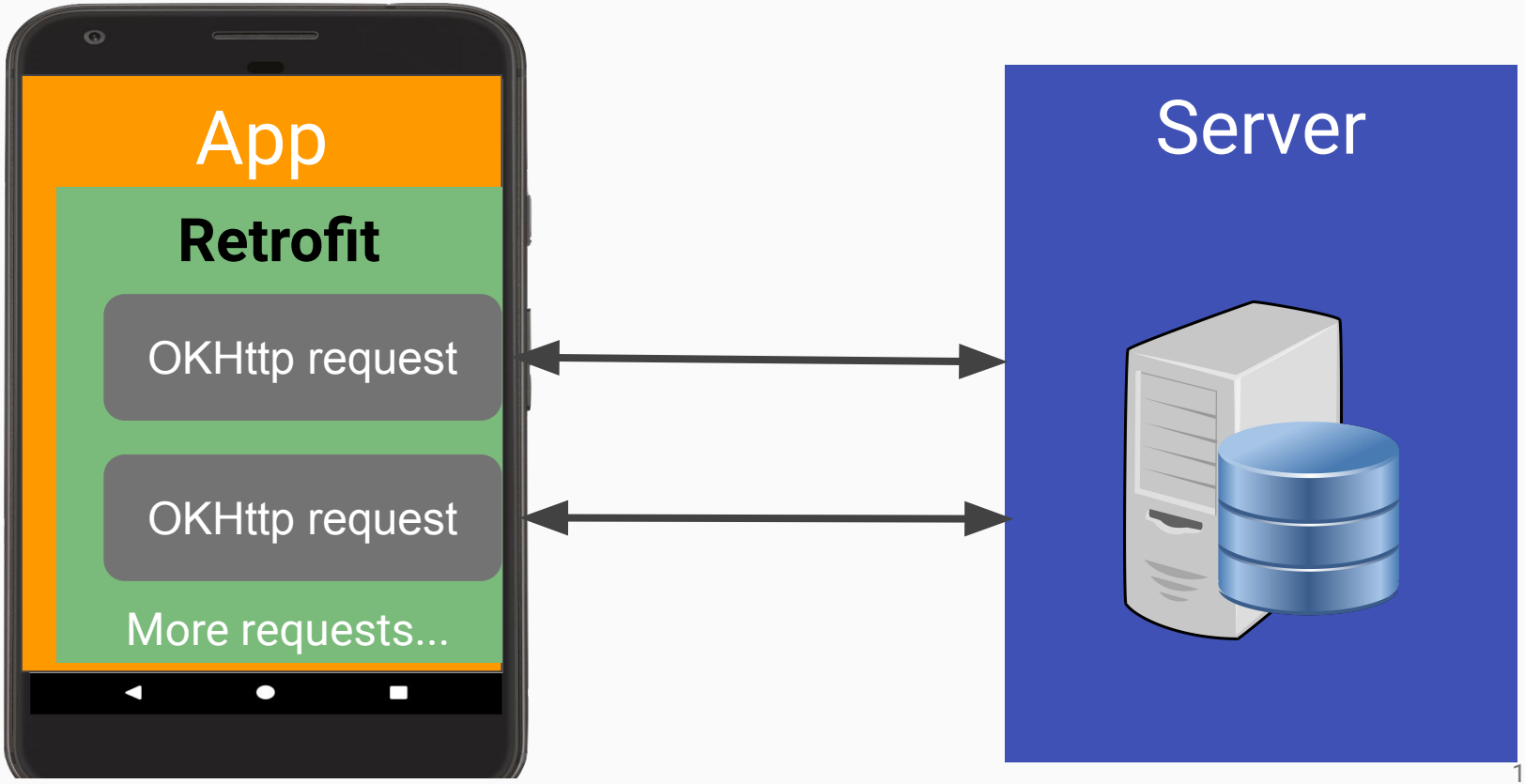
- **Shorter code** - Provides an abstraction layer to Okhttp library
- **Avoids repetitiveness** - no need to implement each Okhttp request
- **Auto-Generates** okhttp code in runtime using reflection
- **Converter.Factory** for Serialization/Deserialization, can be extended
- **Customize** Okhttp client

Retrofit

build.gradle

```
dependencies {
    implementation 'com.squareup.retrofit2:retrofit:2.7.1'
}
```

Retrofit


App
Retrofit
OKHttp request
OKHttp request
More requests...
Server

Rest Client with Retrofit

What retrofit needs in order to help us?

1. Create data models
2. Declare BaseURL for all requests
3. Create Retrofit Instance with BaseUrl
4. Add convertors
5. Http client (optional)
6. Interface representing server APIs
7. Expose retrofit.create() method (the interface)

What retrofit needs in order to help us?

1. Create data models 
2. Declare BaseURL for all requests

1. Declare BaseURL for all requests

```
public class RestClient
{









}
```

1. Declare BaseURL for all requests

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";









}
```

What retrofit needs in order to help us?

1. Create data models 
2. Declare BaseURL for all requests 
3. Create Retrofit Instance with BaseUrl

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder();


}
```

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL);


}
```

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .build();


}
```

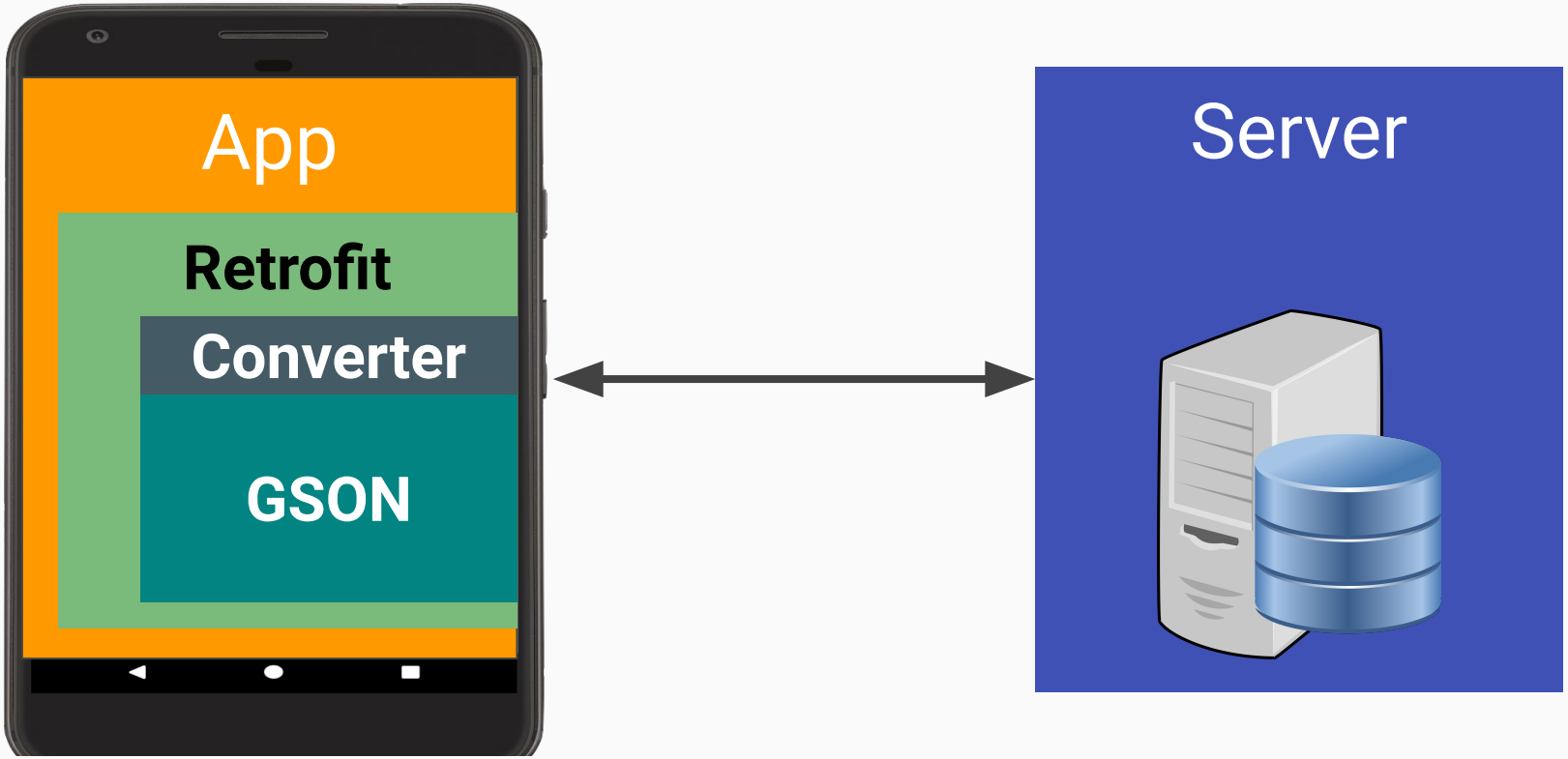
What retrofit needs in order to help us?

1. Create data models 
2. Declare BaseURL for all requests 
3. Create Retrofit Instance with BaseUrl 
4. Add convertors

Convertors

- Gson: com.squareup.retrofit2:converter-gson
- Jackson: com.squareup.retrofit2:converter-jackson
- Moshi: com.squareup.retrofit2:converter-moshi
- Protobuf: com.squareup.retrofit2:converter-protobuf
- Wire: com.squareup.retrofit2:converter-wire
- Simple XML: com.squareup.retrofit2:converter-simplexml
- Scalars: com.squareup.retrofit2:converter-scalars

Retrofit



Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .build();


}
```

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();


}
```

What retrofit needs in order to help us?

1. Create data models 
2. Declare BaseURL for all requests 
3. Create Retrofit Instance with BaseUrl 
4. Add convertors 
5. Http client (optional)

Setting up an Http Client (optional)

Setting up an Http Client (optional)

The default http engine in Retrofit is OKHttp.

Replaces the HttpUrlConnection.

Setting up an Http Client (optional)

The default http engine in Retrofit is OKHttp.

Replaces the HttpUrlConnection.

- Shrink data for better performance
- Cache requests
- Define timeouts
- Enable interceptors
- And much more

Retrofit - OKHttp Client

```
public class RestClient {
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
            .connectTimeout(40, TimeUnit.SECONDS)
            .build();

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    static MoviesService moviesService = retrofit.create(MoviesService.class);
}
```

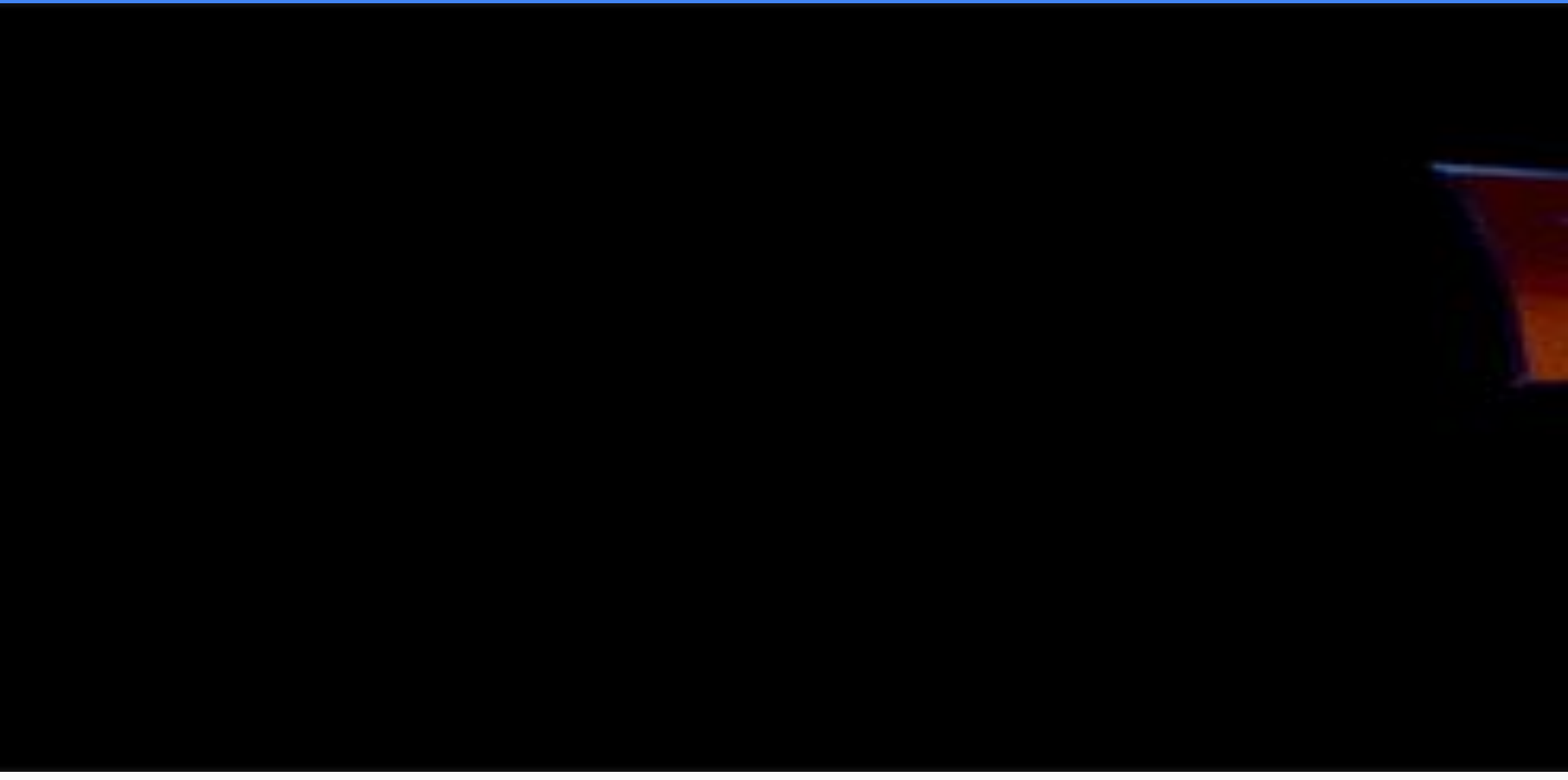
Retrofit - OKHttp Client

```
public class RestClient {
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
            .connectTimeout(40, TimeUnit.SECONDS)
            .build();

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .client(okHttpClient)
            .build();

    static MoviesService moviesService = retrofit.create(MoviesService.class);
}
```

What retrofit needs in order to help us?

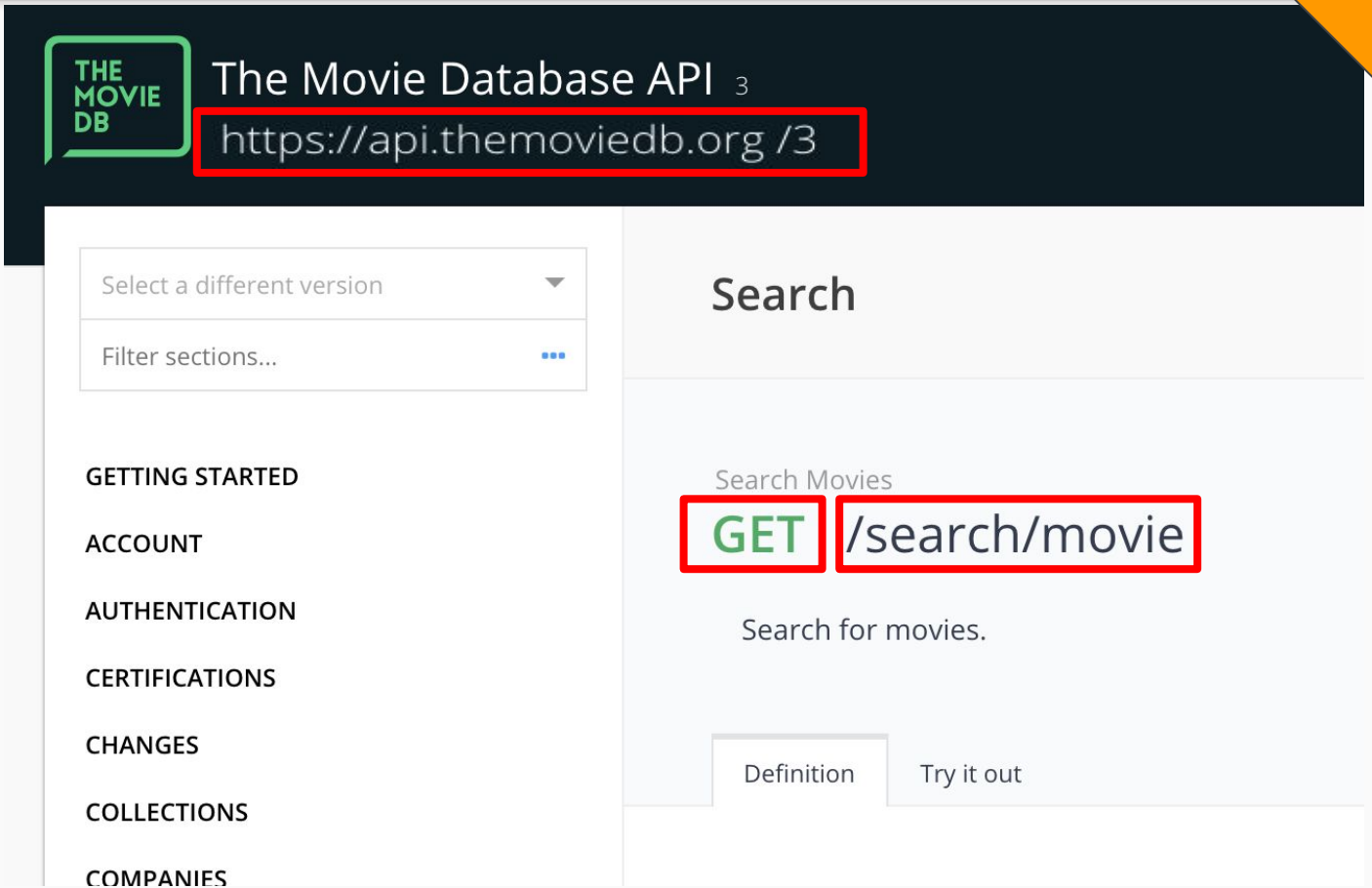
1. Create data models 
2. Declare BaseURL for all requests 
3. Create Retrofit Instance with BaseUrl 
4. Add convertors 
5. Http client (optional) 
6. Interface representing server APIs

Body-less functions -> here we come



API

REMINDER



Retrofit

```
public interface MoviesService {



}
```

Retrofit

```
public interface MoviesService {

    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {

    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {

    Call<VideosListResult> searchMoviesByTitle();

}
```

What is this call object?

Retrofit

- `Call<MoviesList>`

  Retrofit will take the response body and

  try to convert it to Java objects.

Retrofit

- `Call<MoviesList>`

  Retrofit will take the response body and

  try to convert it to Kotlin objects.

- `Call<ResponseBody>`

  This makes the raw response payload available,

  but skips the mapping to Java objects

Retrofit

- `Call<MoviesList>`

  Retrofit will take the response body and

  try to convert it to Kotlin objects.

- `Call<ResponseBody>`

  This makes the raw response payload available,

  but skips the mapping to Kotlin objects

- `Call<Void>`

  Skips response

Retrofit

```
public interface MoviesService {

    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {

    @GET
    Call<VideosListResult> searchMoviesByTitle();



}
```

```
public interface MoviesService {

    @GET
    Call<VideosListResult> searchMoviesByTitle();

}
```

HTTP Verb

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies
    @GET
    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle();


}
```

Endpoint

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle();



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle(String title);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle(@Query("query") String title);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle(@Query("query") String title);




}
```

Retrofit

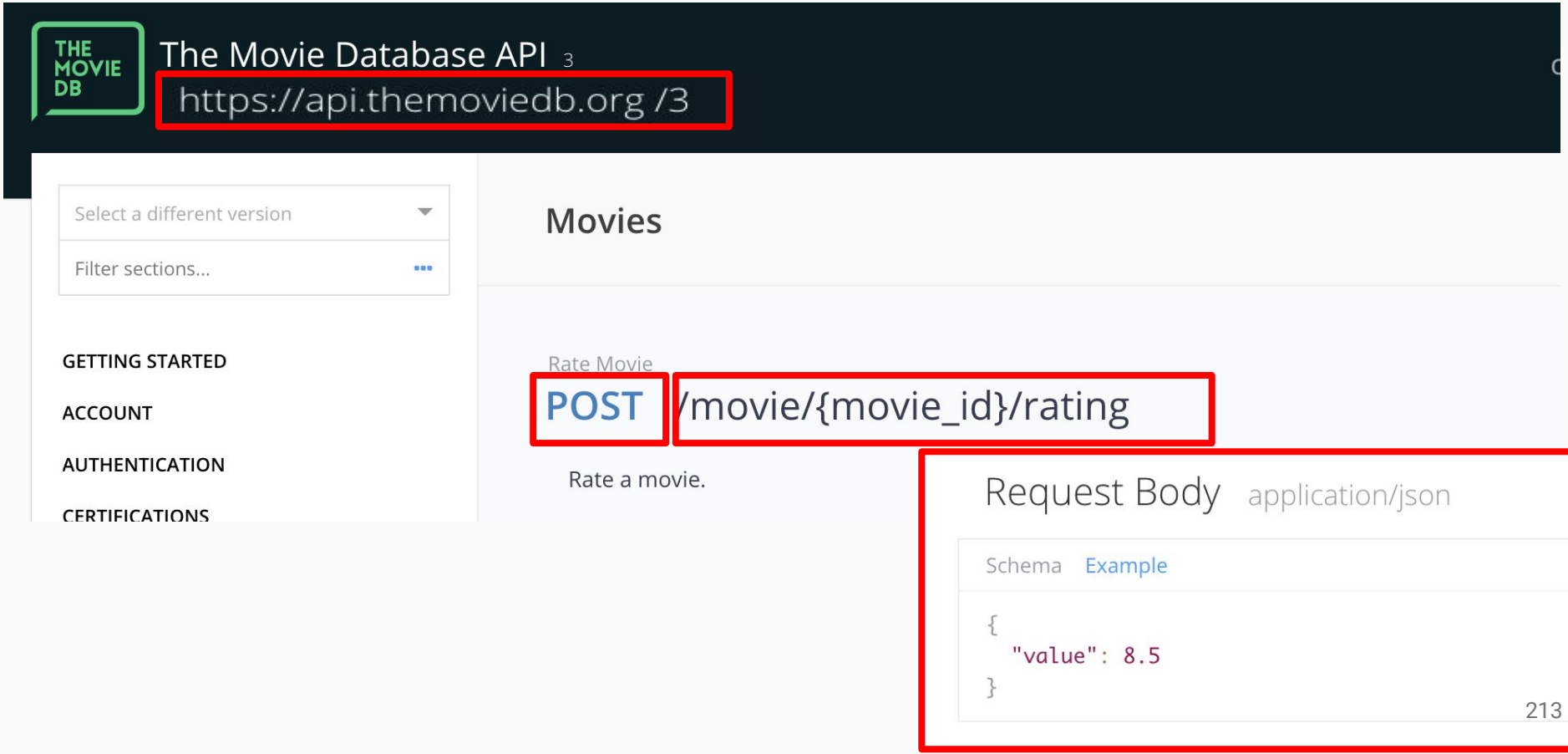
```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle(@Query("query") String title);


}
```

query

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/search/movies?query=android
    @GET("search/movies")
    Call<VideosListResult> searchMoviesByTitle(@Query("query") String title);

    // https://api.themoviedb.org/3/discover/movie?year=1993
    @GET("discover/movie")
    Call<VideosListResult> discoverMovies(@Query("year") int year);


}
```

API


THE MOVIE DB
The Movie Database API 3
https://api.themoviedb.org /3
Select a different version
Filter sections...
GETTING STARTED
ACCOUNT
AUTHENTICATION
CERTIFICATIONS
Movies
Rate Movie
POST /movie/{movie_id}/rating
Rate a movie.
Request Body application/json
Schema Example
{
"value": 8.5
}

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

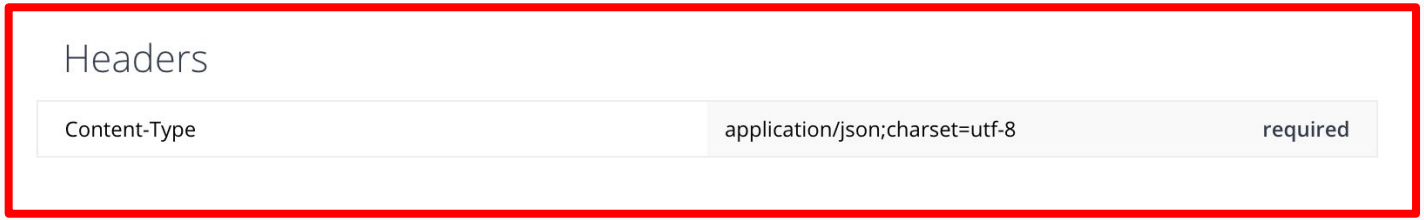
```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(String movieId);



}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);



}
```

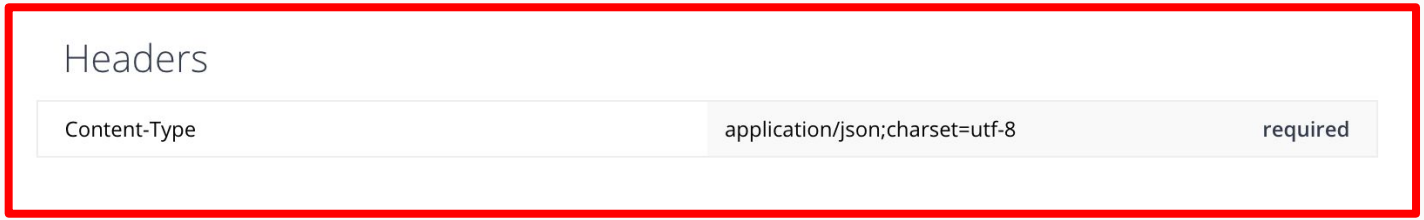
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);


}
```



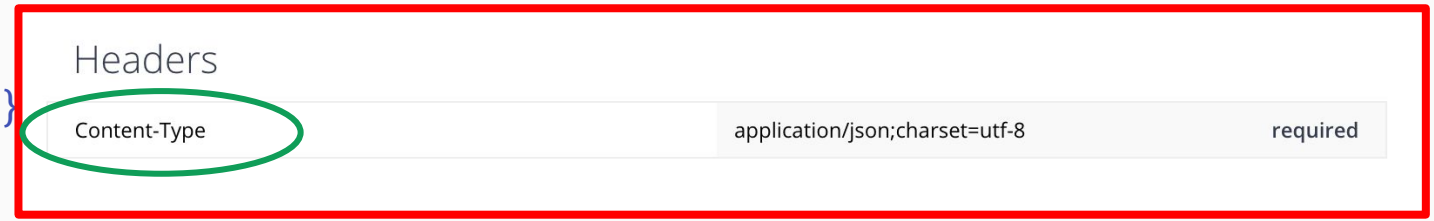
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);


}
```



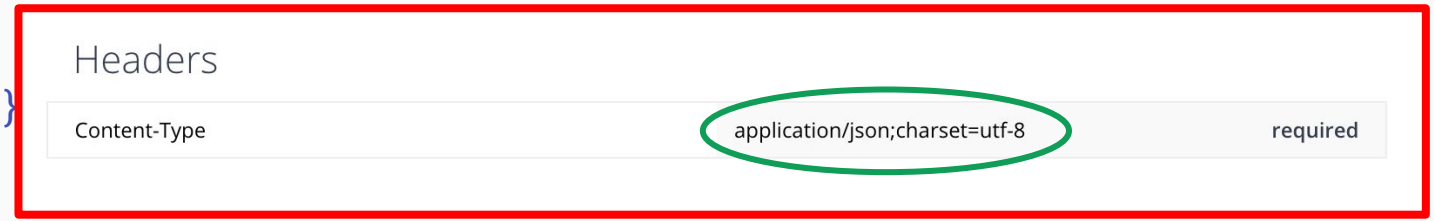
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);

}
```



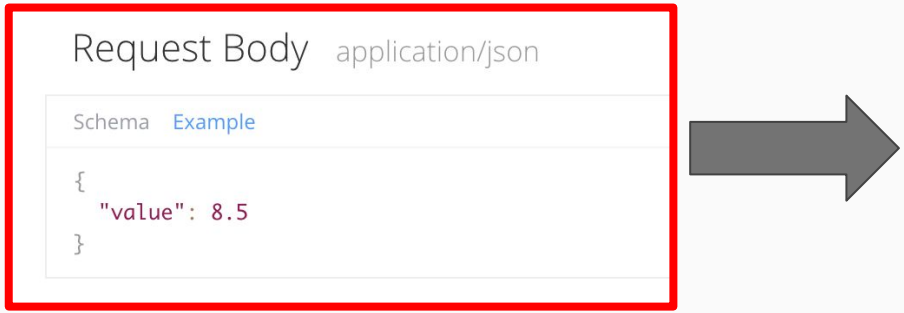
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);

}
```



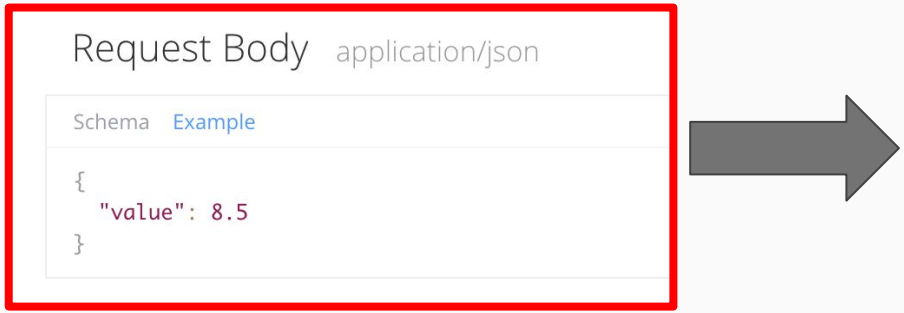
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId);
}
```



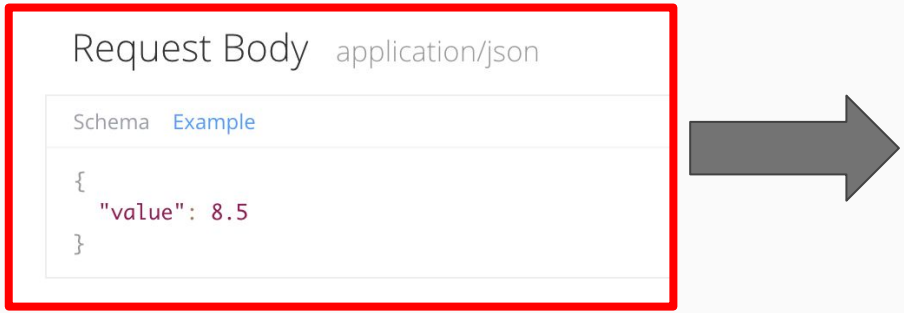
Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                         String rating);
}
```



Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                         @Body String rating);
}
```



Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                         @Body String rating);
```

Example Input:

```
String rating: ="{\"value\":8.5}"
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                         @Body String rating);
```

Example Input:

```
class Rating {
  Float value;
}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                         @Body Rating rating);
```

Example Input:

```
class Rating {
  Float value;
}
```

Retrofit

```
public interface MoviesService {
    // https://api.themoviedb.org/3/movies/{movie_id}/rating
    @Headers("content-type:application/json;charset=utf-8")
    @POST("movies/{movie_id}/rating")
    Call<Void> rateMovie(@Path("movie_id") String movieId,
                                                @Body Rating rating);



}
```

What retrofit needs in order to help us?

1. Create data models 
2. Declare BaseURL for all requests 
3. Create Retrofit Instance with BaseUrl 
4. Add convertors 
5. Http client (optional) 
6. Interface representing server APIs 
7. Expose retrofit.create() method (the interface)

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    static MoviesService moviesService
}
```

Retrofit

```
public class RestClient
{
    private static final String BASE_URL = "https://api.themoviedb.org";

    private static Retrofit retrofit = new Retrofit.Builder().
            baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    static MoviesService moviesService = retrofit.create(MoviesService.class);
}
```

Questions?

Make the call!

Retrofit - Asynchronous call

```
private void loadMovies(){









}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call




}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);




}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);
    call.enqueue(new Callback<VideosListResult>(){









    });
}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);
    call.enqueue(new Callback<VideosListResult>(){
        @Override
        public void onResponse(Call<VideosListResult> call, Response<VideosListResult> response){









        }









    });
}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);
    call.enqueue(new Callback<VideosListResult>(){
        @Override
        public void onResponse(Call<VideosListResult> call, Response<VideosListResult> response){
           if (response.isSuccessful()) {



           }
        }



    });
}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);
    call.enqueue(new Callback<VideosListResult>(){
        @Override
        public void onResponse(Call<VideosListResult> call, Response<VideosListResult> response){
           if (response.isSuccessful()) {
                adapter = new MoviesViewAdapter(response.getVideos(), MoviesActivity.this)
                recyclerView.setAdapter(adapter);
           }
        }




    });
}
```

Retrofit - Asynchronous call

```
private void loadMovies(){
    Call<VideosListResult> call = RestClient.moviesService.discoverMovies(2015);
    call.enqueue(new Callback<VideosListResult>(){
        @Override
        public void onResponse(Call<VideosListResult> call, Response<VideosListResult> response){
            if (response.isSuccessful()) {
                adapter = new MoviesViewAdapter(response.getVideos(), MoviesActivity.this)
                recyclerView.setAdapter(adapter);
            }
        }

        @Override
        public void onFailure(Call<VideosListResult> call, Throwable t){
           // show an error to the user
        }
    });
}
```

Load the data!

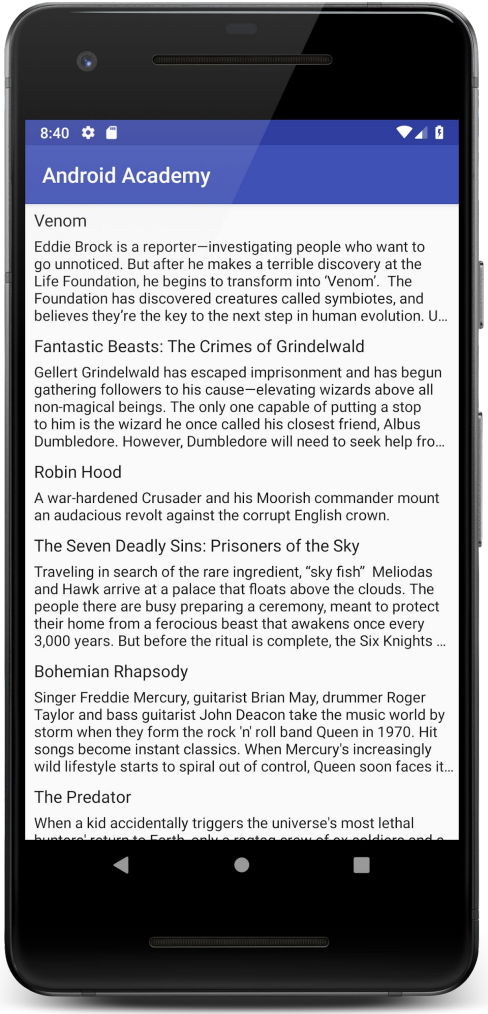
```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {









    }
}
```

Load the data!

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {
        VideoResult movie = movies.get(position)









    }
}
```

Load the data!

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {
        VideoResult movie = movies.get(position)
        tvTitle.setText(movie.getTitle());
        tvOverview.setText(movie.getOverview());




    }
}
```


8:40
Android Academy
Venom
Eddie Brock is a reporter—investigating people who want to go unnoticed. But after he makes a terrible discovery at the Life Foundation, he begins to transform into 'Venom'. The Foundation has discovered creatures called symbiotes, and believes they're the key to the next step in human evolution. U...
Fantastic Beasts: The Crimes of Grindelwald
Gellert Grindelwald has escaped imprisonment and has begun gathering followers to his cause—elevating wizards above all non-magical beings. The only one capable of putting a stop to him is the wizard he once called his closest friend, Albus Dumbledore. However, Dumbledore will need to seek help fro...
Robin Hood
A war-hardened Crusader and his Moorish commander mount an audacious revolt against the corrupt English crown.
The Seven Deadly Sins: Prisoners of the Sky
Traveling in search of the rare ingredient, "sky fish" Meliodas and Hawk arrive at a palace that floats above the clouds. The people there are busy preparing a ceremony, meant to protect their home from a ferocious beast that awakens once every 3,000 years. But before the ritual is complete, the Six Knights ...
Bohemian Rhapsody
Singer Freddie Mercury, guitarist Brian May, drummer Roger Taylor and bass guitarist John Deacon take the music world by storm when they form the rock 'n' roll band Queen in 1970. Hit songs become instant classics. When Mercury's increasingly wild lifestyle starts to spiral out of control, Queen soon faces it...
The Predator
When a kid accidentally triggers the universe's most lethal

Questions?

Load images!

Challenges

1. Heavy
2. Out of memory (Exception)
3. Caching
4. Resize
5. Image manipulation (round corners…)

Picasso

http://square.github.io/picasso/

https://github.com/bumptech/glide

Images loading

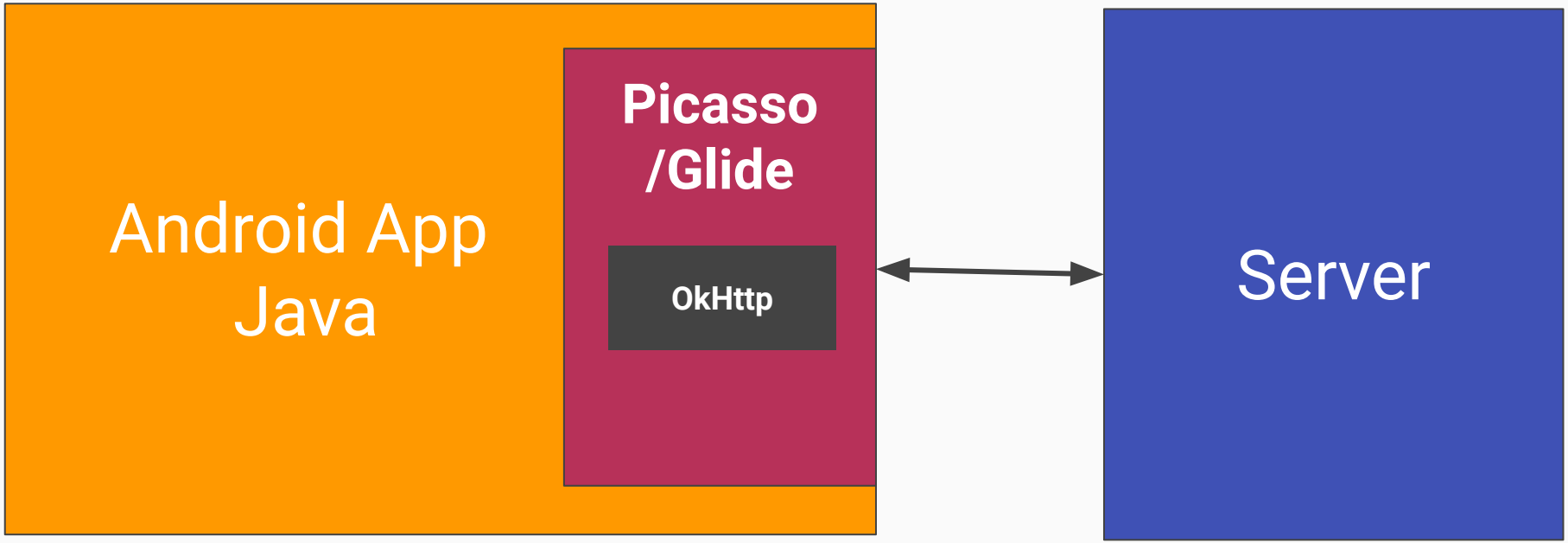


Image loading

```
build.gradle()

dependencies {
    implementation 'com.squareup.picasso:picasso:2.71828
}
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
        .load(imageUrl)
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
        .load(imageUrl)
        .into(videoImageView);
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
        .load(imageUrl)
        .placeholder(R.drawable.my_place_holder_image)
        .into(videoImageView);
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
        .load(imageUrl)
        .placeholder(R.drawable.my_place_holder_image)
        .error(R.drawable.error_image)
        .into(videoImageView);
```

Image loading

```
String imageUrl = “http://www.imgur.com/banana.png”;
ImageView videoImageView = findViewById(R.id.posterImageView);

Picasso.get()
        .load(imageUrl)
        .placeholder(R.drawable.my_place_holder_image)
        .error(R.drawable.error_image)
        .rotate(90)
        .into(videoImageView);
```

Image loading

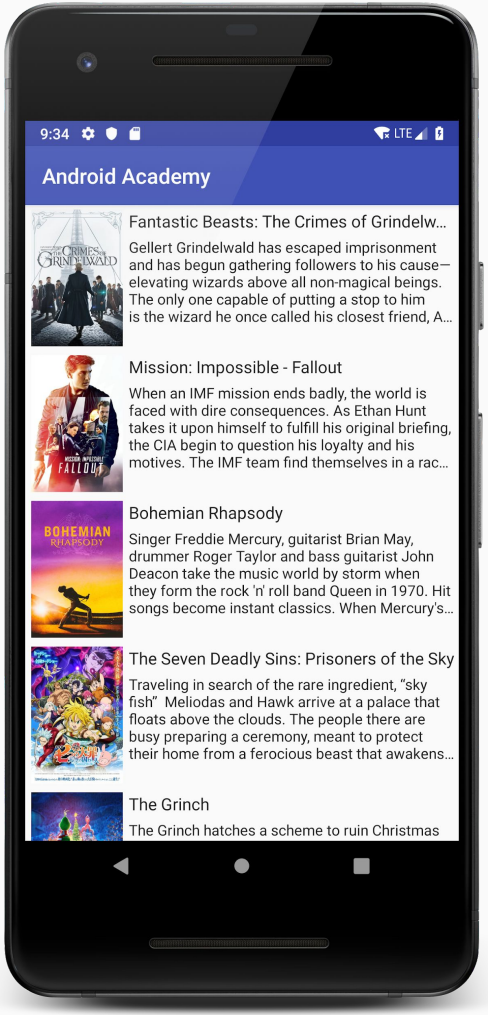
```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {
        MovieModel movie = movies.get(position)
        tvTitle.setText(movie.getTitle());
        tvOverview.setText(movie.getOverview());









    }
}
```

Image loading

```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {
        MovieModel movie = movies.get(position)
        tvTitle.setText(movie.getTitle());
        tvOverview.setText(movie.getOverview());
        if (!TextUtils.isEmpty(movie.getImageUrl())) {



        }
    }
}
```

Image loading

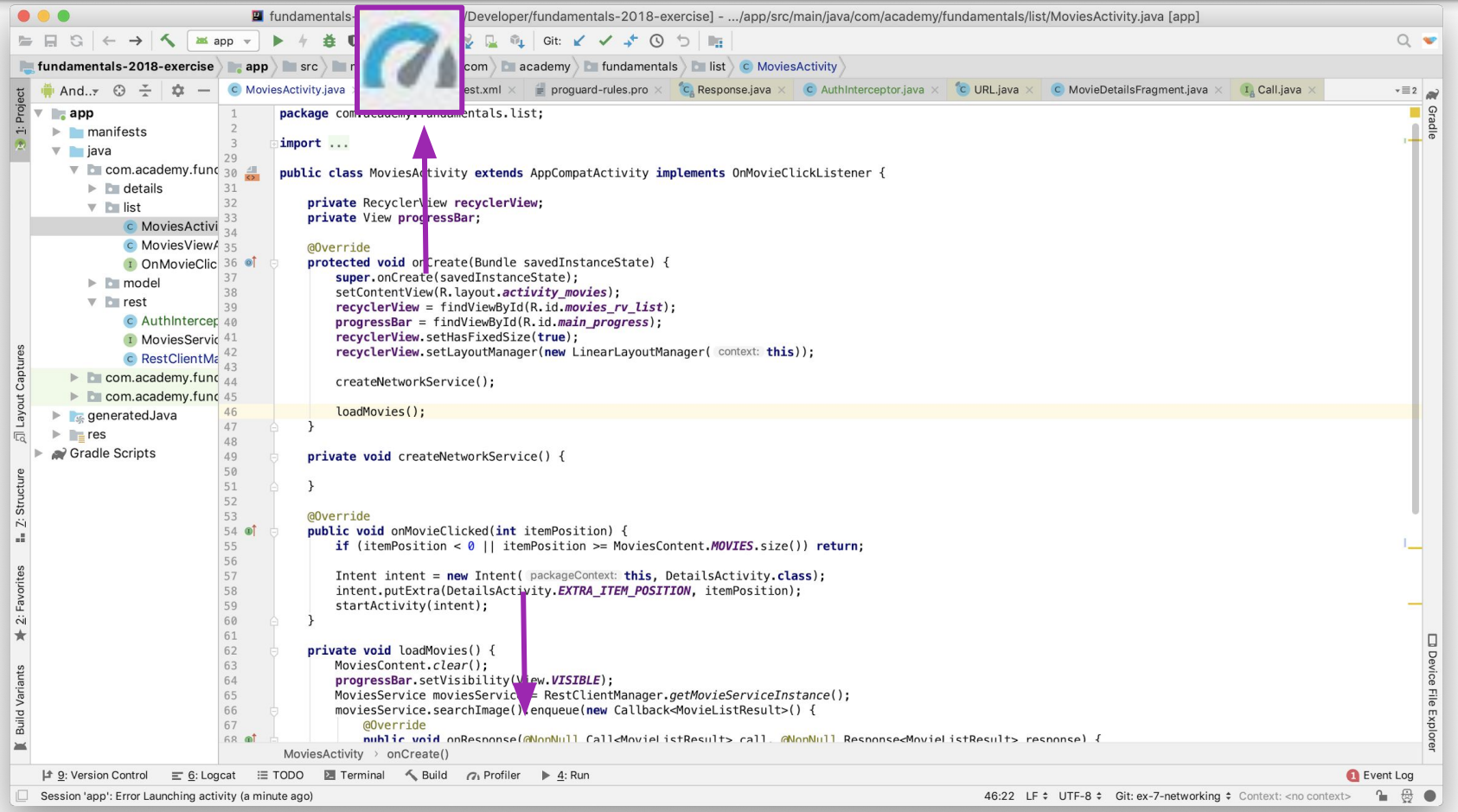
```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {

    public void onBindViewHolder(MovieViewHolder holder, int position) {
        MovieModel movie = movies.get(position)
        tvTitle.setText(movie.getTitle());
        tvOverview.setText(movie.getOverview());
        if (!TextUtils.isEmpty(movie.getImageUrl())) {
            Picasso.get()
               .load(movie.getImageUrl())
               .into(imageView);
        }
    }
}
```


9:34
LTE
Android Academy
Fantastic Beasts: The Crimes of Grindelw...
Gellert Grindelwald has escaped imprisonment and has begun gathering followers to his cause—elevating wizards above all non-magical beings. The only one capable of putting a stop to him is the wizard he once called his closest friend, A...
Mission: Impossible - Fallout
When an IMF mission ends badly, the world is faced with dire consequences. As Ethan Hunt takes it upon himself to fulfill his original briefing, the CIA begin to question his loyalty and his motives. The IMF team find themselves in a rac...
Bohemian Rhapsody
Singer Freddie Mercury, guitarist Brian May, drummer Roger Taylor and bass guitarist John Deacon take the music world by storm when they form the rock 'n' roll band Queen in 1970. Hit songs become instant classics. When Mercury's...
The Seven Deadly Sins: Prisoners of the Sky
Traveling in search of the rare ingredient, "sky fish" Meliodas and Hawk arrive at a palace that floats above the clouds. The people there are busy preparing a ceremony, meant to protect their home from a ferocious beast that awakens...
The Grinch
The Grinch hatches a scheme to ruin Christmas

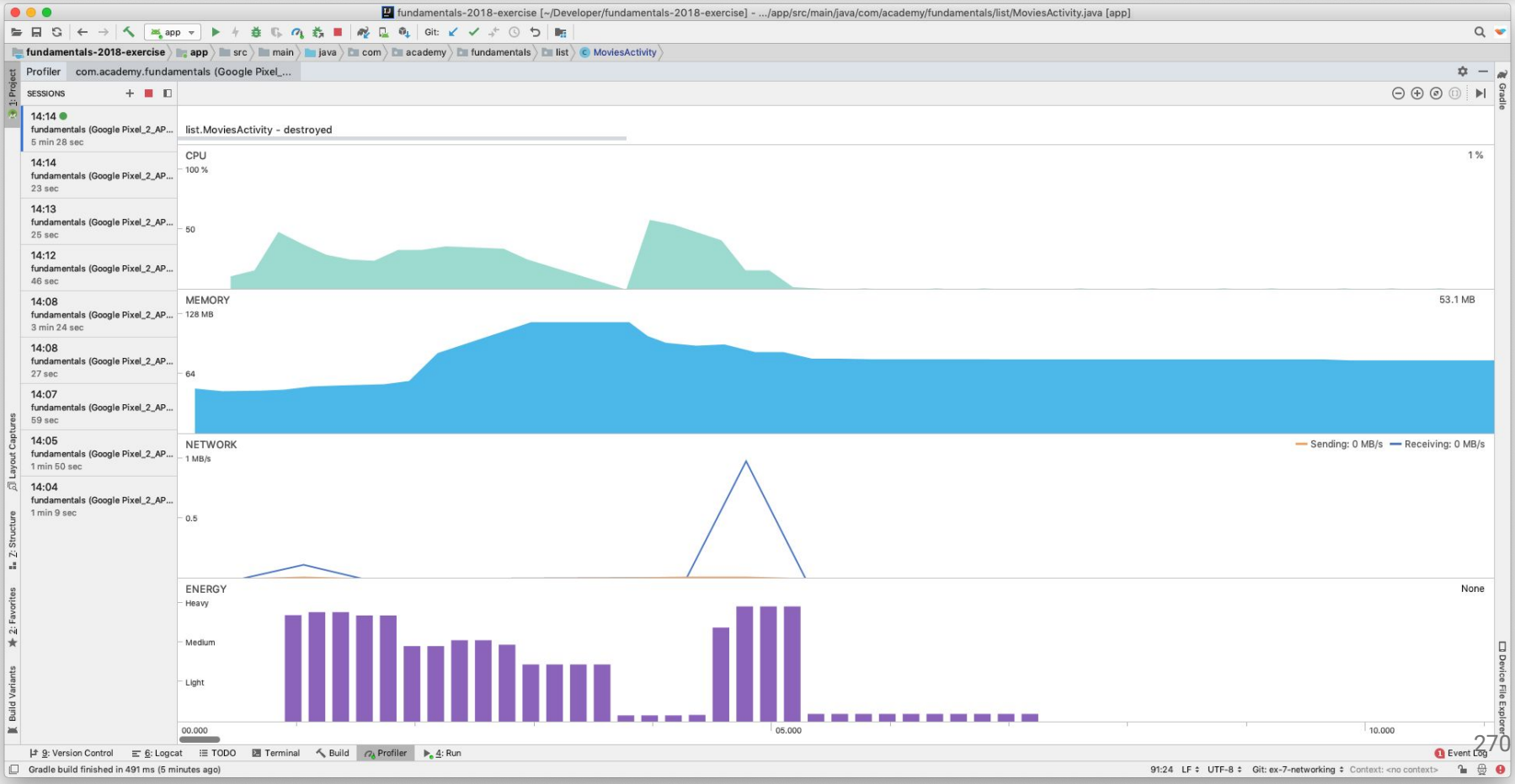

We got this

Network Profiler

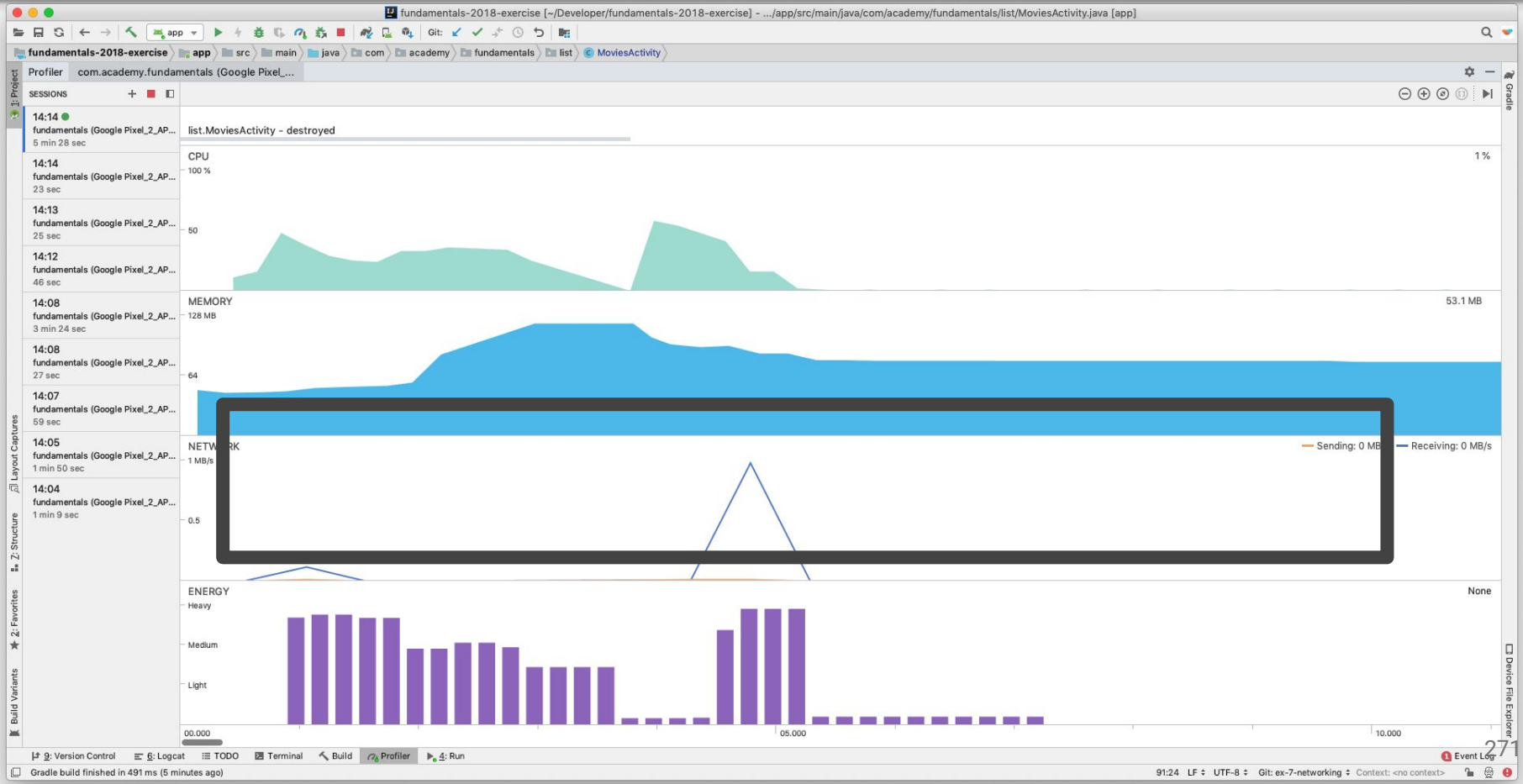
Network profiler



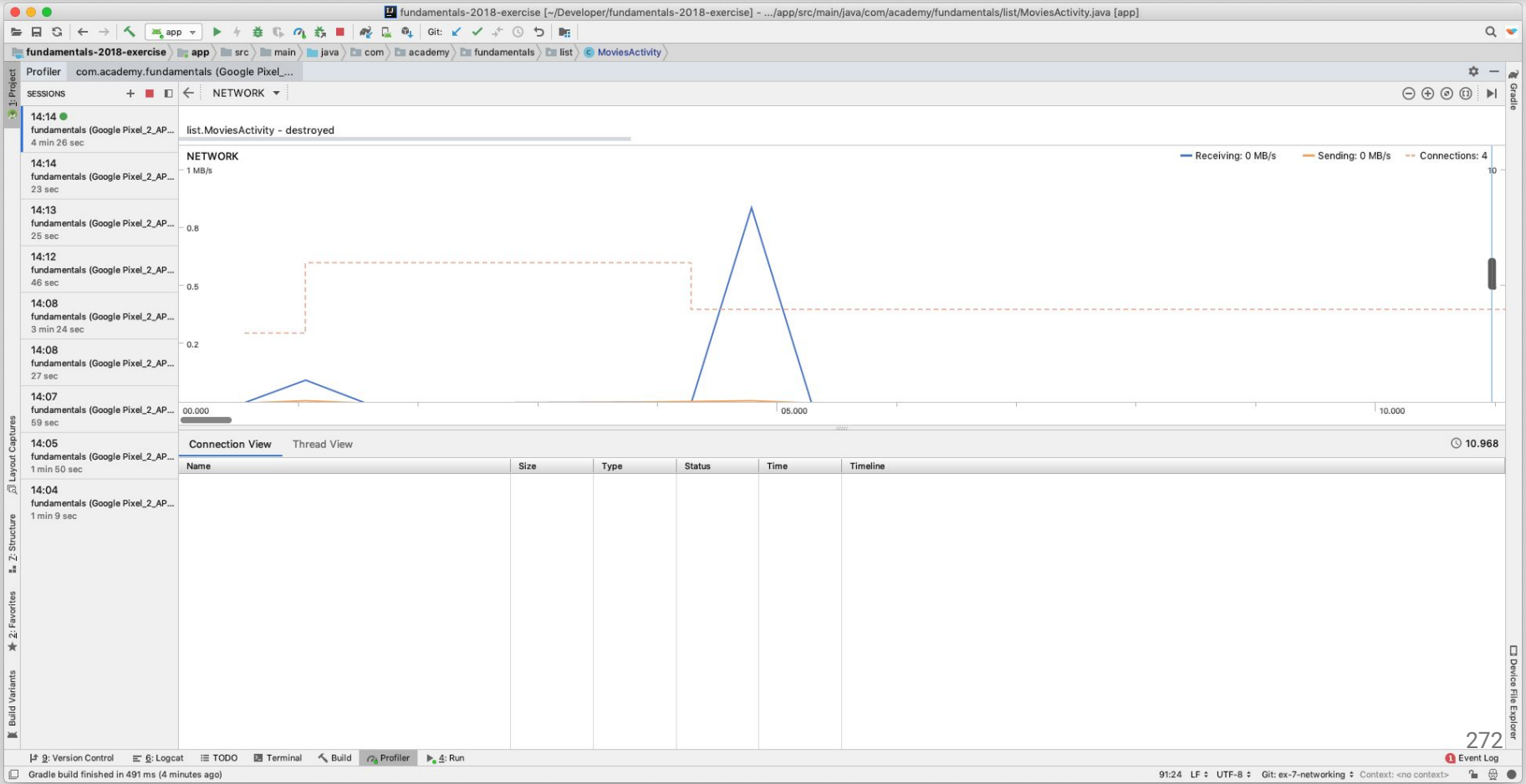
Network profiler



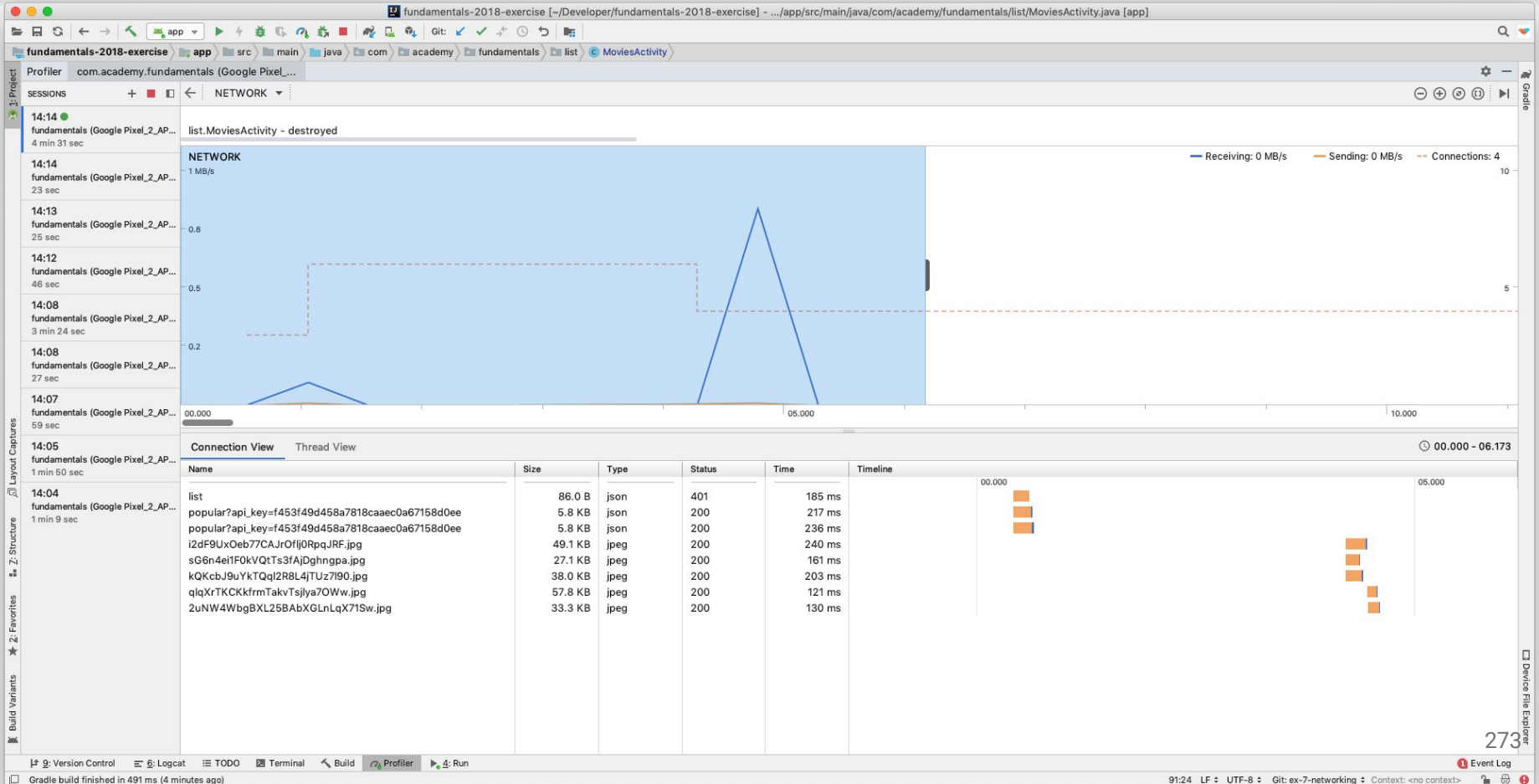
Network profiler



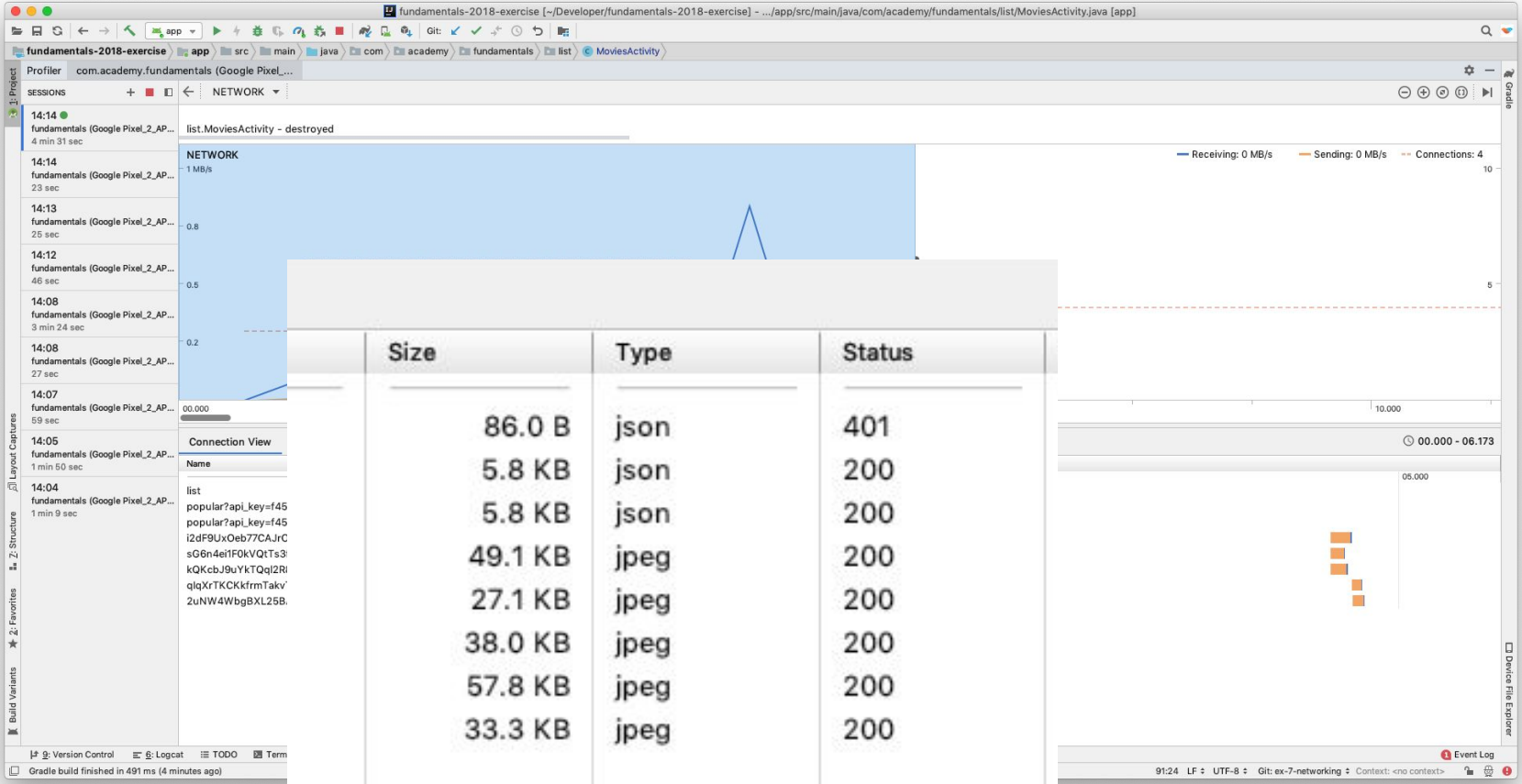
Network profiler



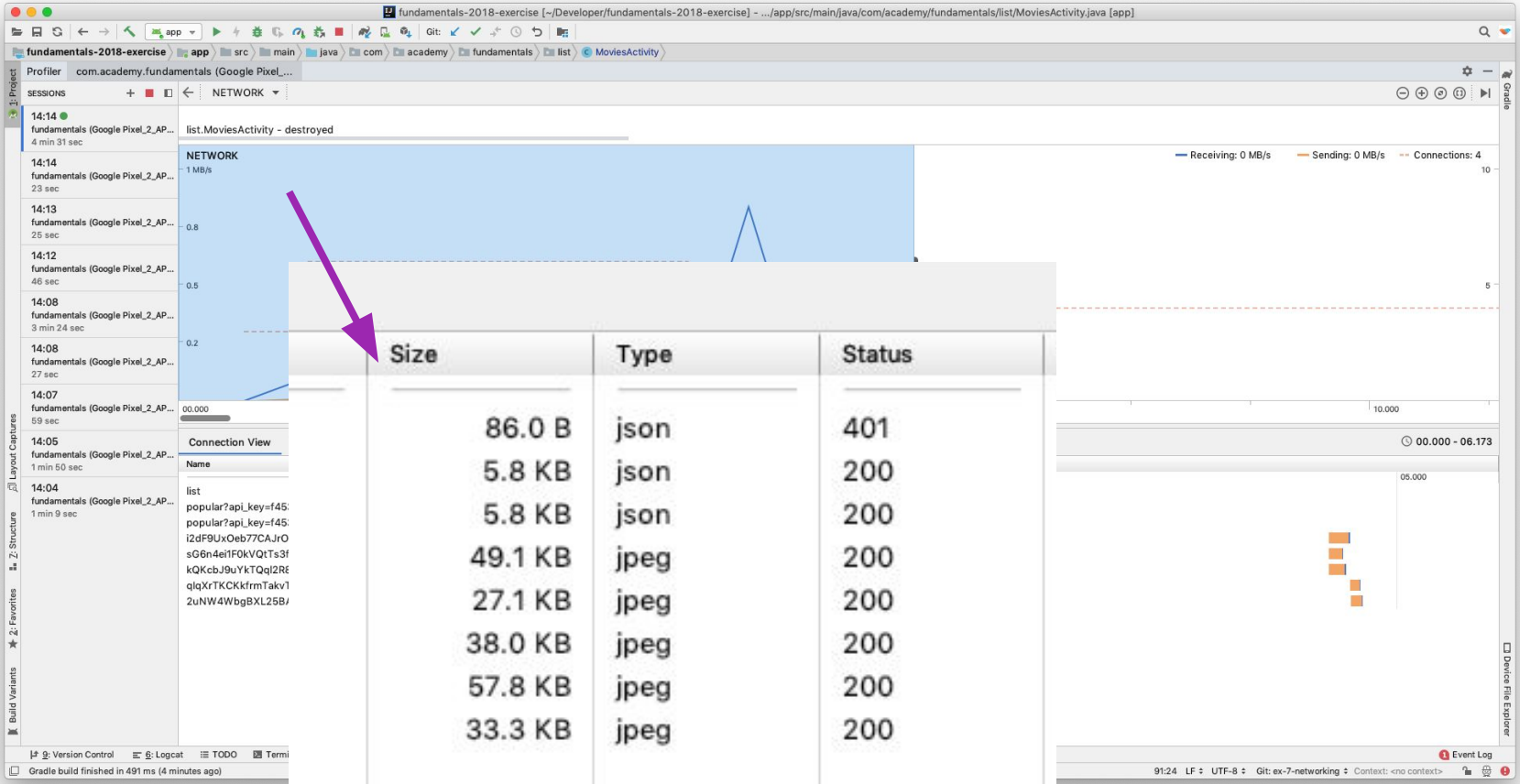
Network profiler



Network profiler

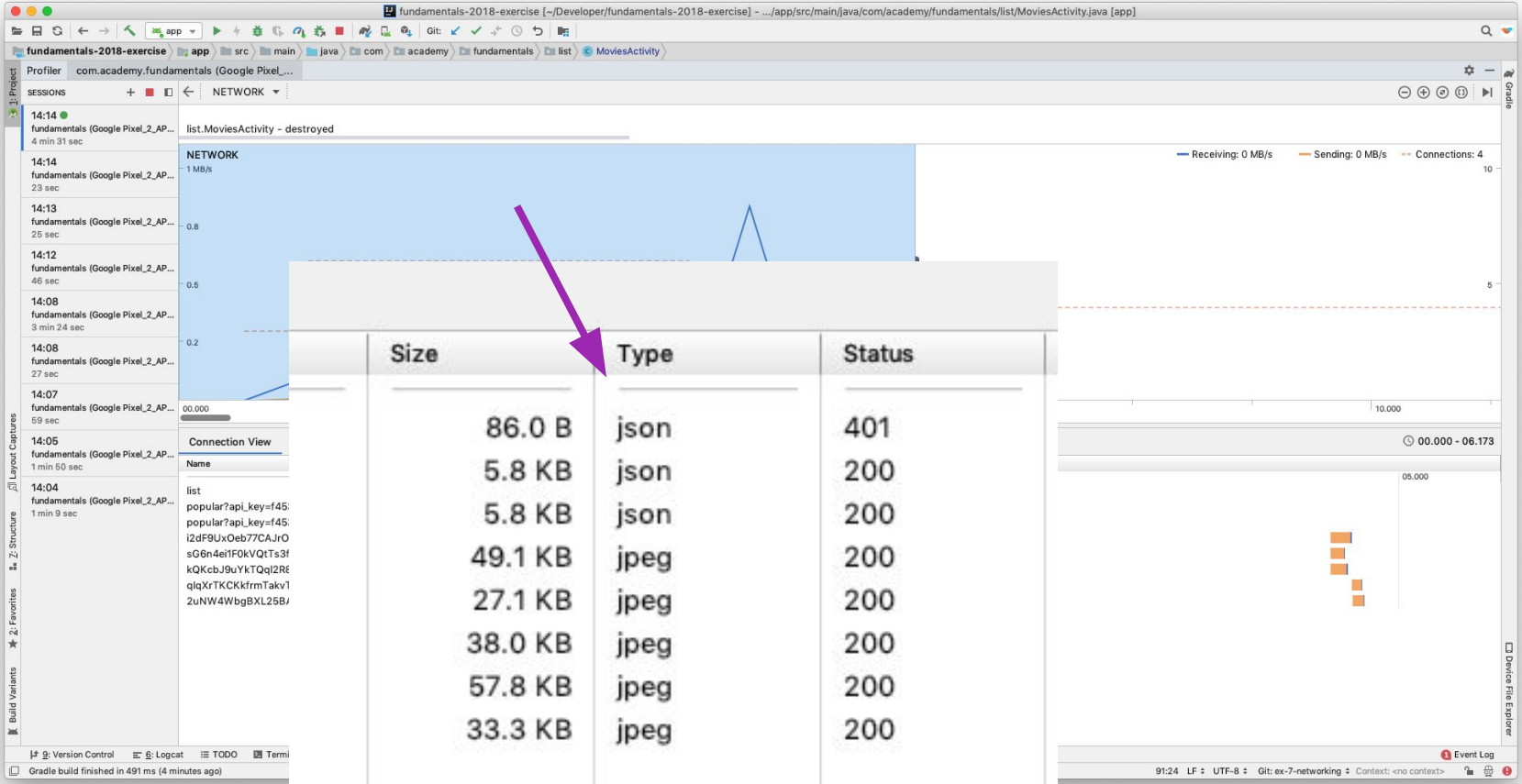


Network profiler

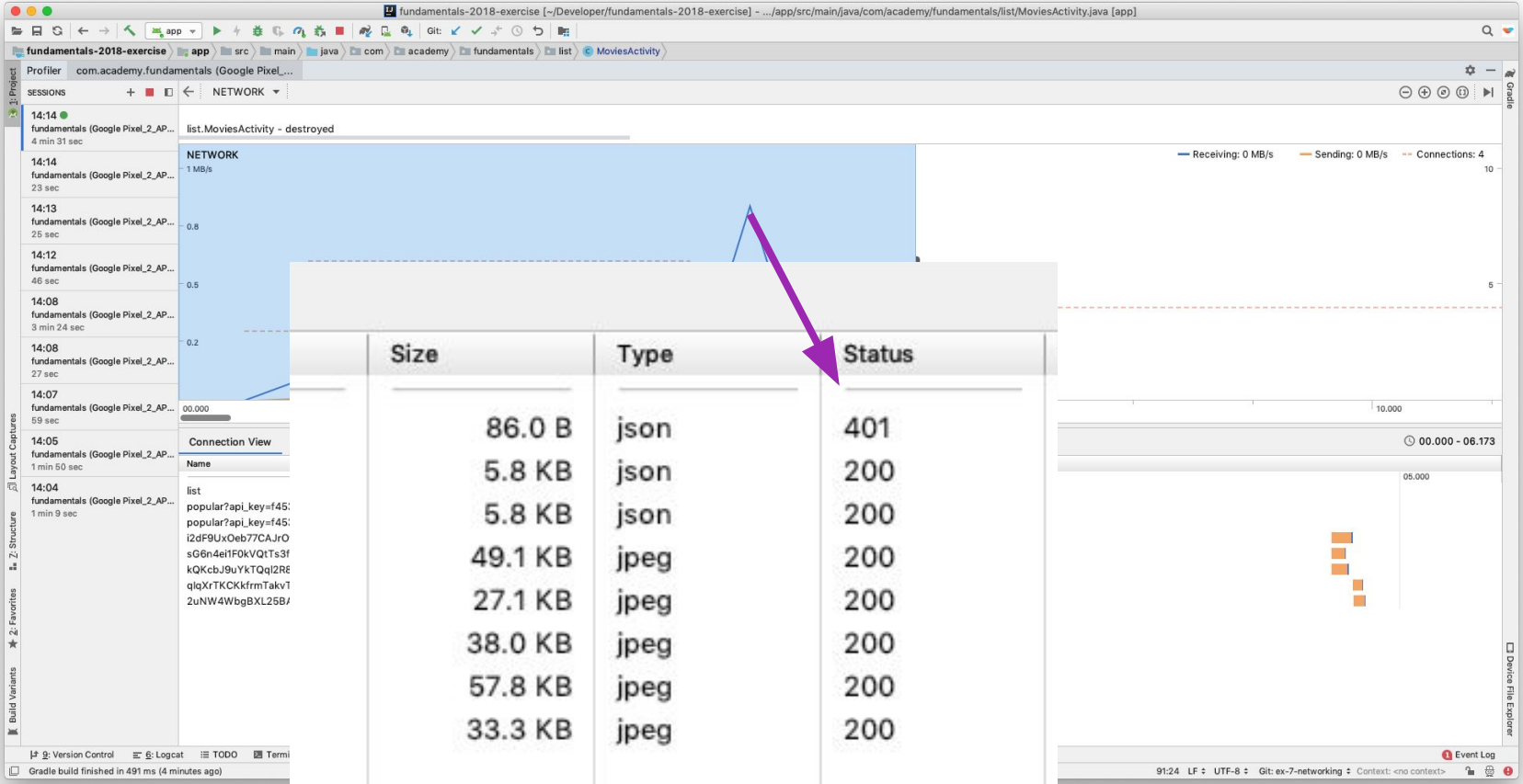


Size	Type	Status
86.0 B	json	401
5.8 KB	json	200
5.8 KB	json	200
49.1 KB	jpeg	200
27.1 KB	jpeg	200
38.0 KB	jpeg	200
57.8 KB	jpeg	200
33.3 KB	jpeg	200

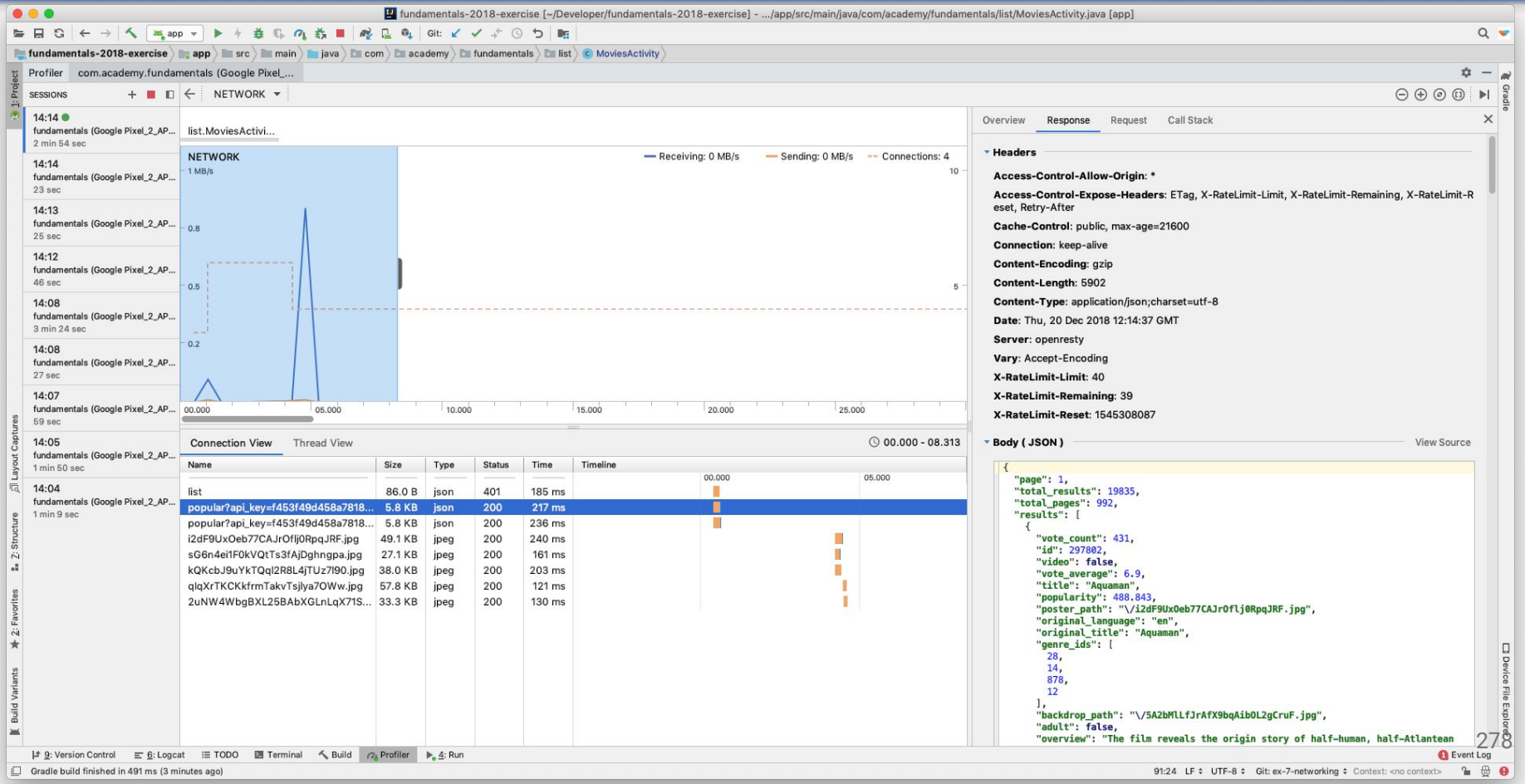
Network profiler



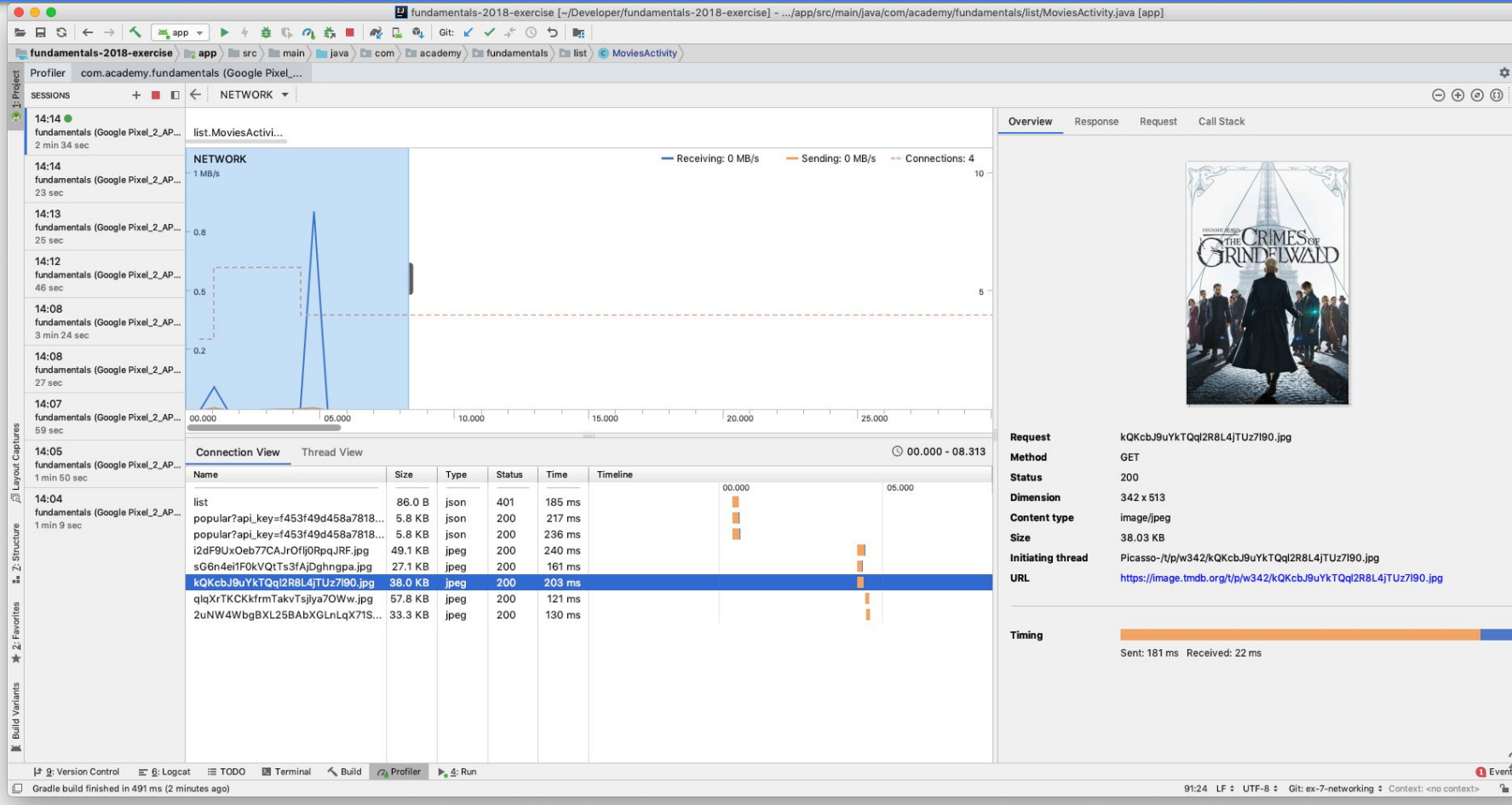
Network profiler



Network profiler



Network profiler



Questions?

Exercise