



Android Lecture #9



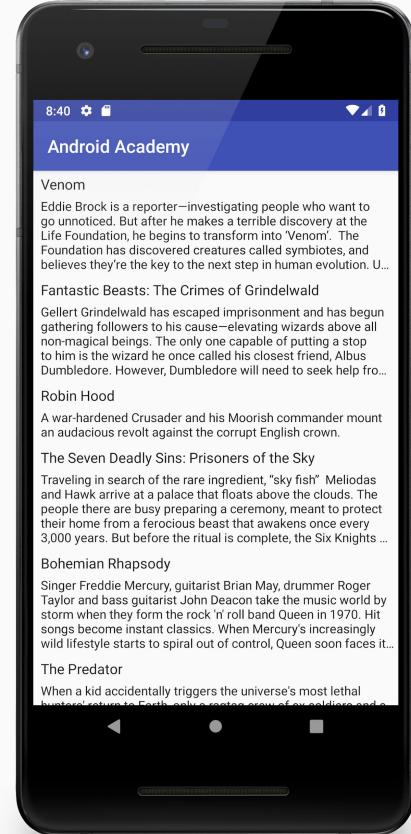
Networking

Agenda

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

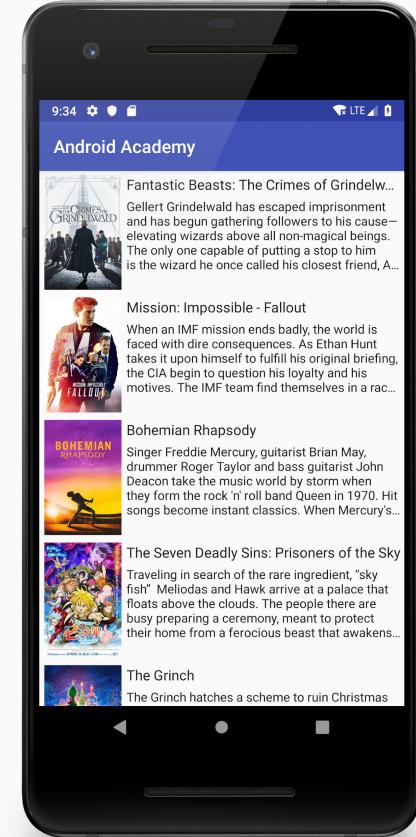


1. Load data from the server



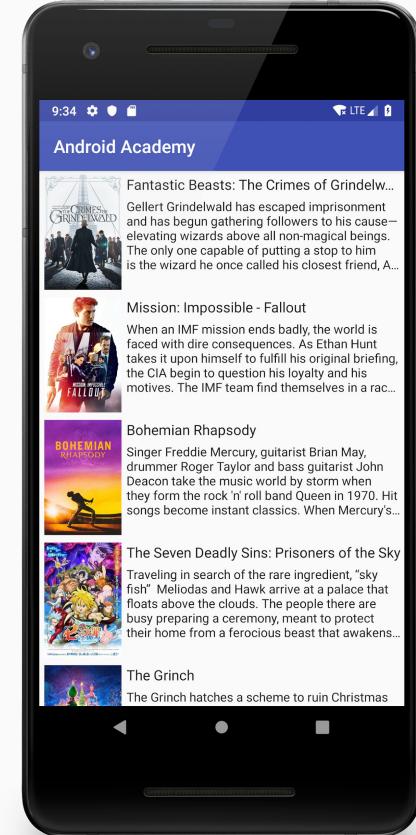
Goals:

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?



Ask permission

Pleeeeeaaaase.....?



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

Permissions

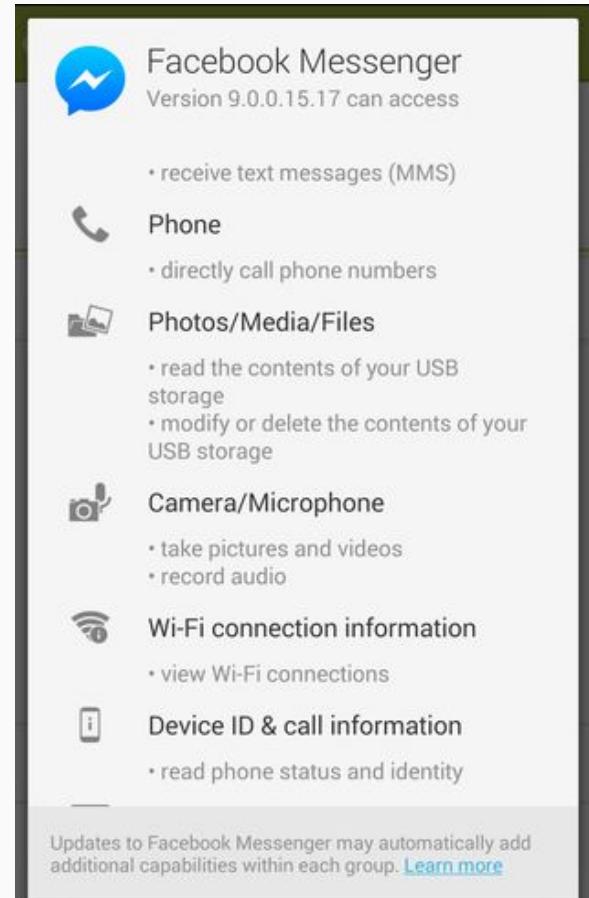
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.academy.fundamentals">

    <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

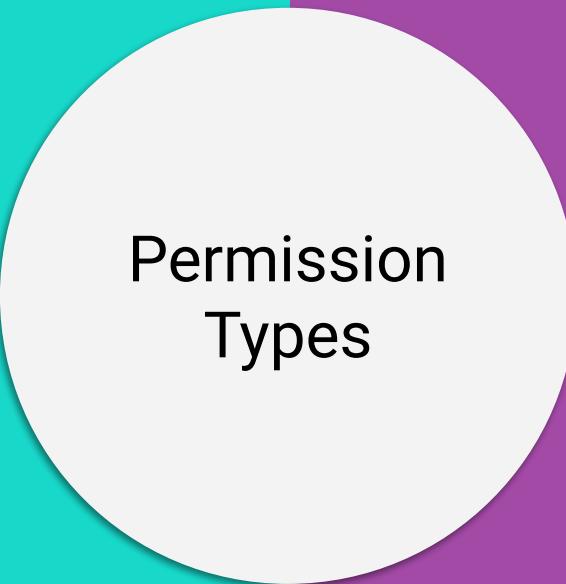
[Click for documentation](#)

Dangerous

Contacts
Location
Storage

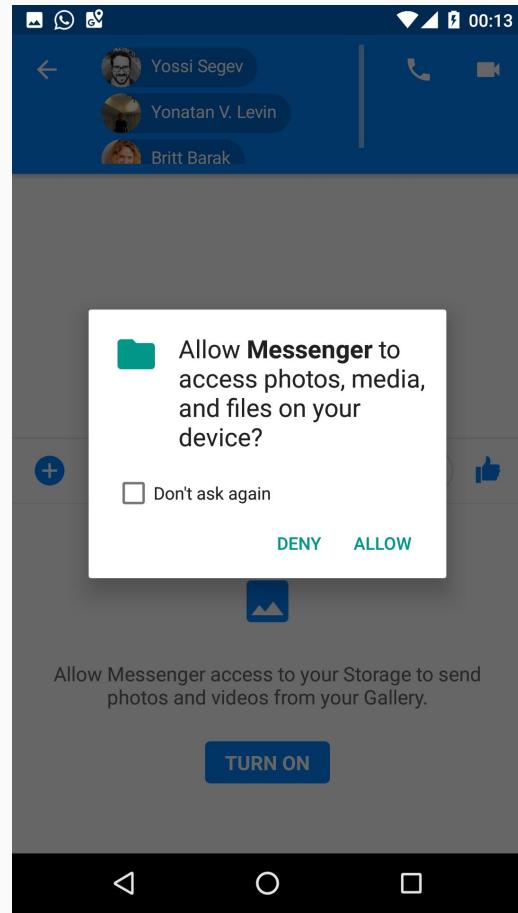
Granted by user

[Click for documentation](#)



Permissions

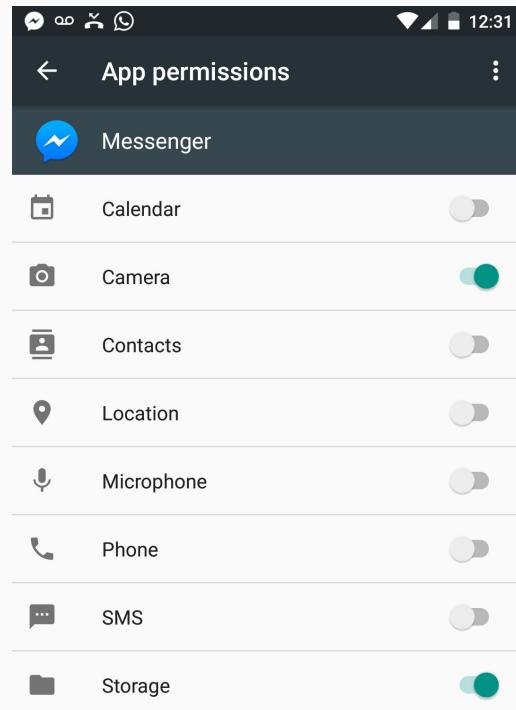
API 23+ : Runtime



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

Permissions

API 23+ : Runtime



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

Questions?

Client



Request



Server



Response



HTTP

- Hypertext Transfer Protocol.
- Stateless protocol.

Headers

additional information about the request or response

Content-Type : application/json

User-Agent : Android

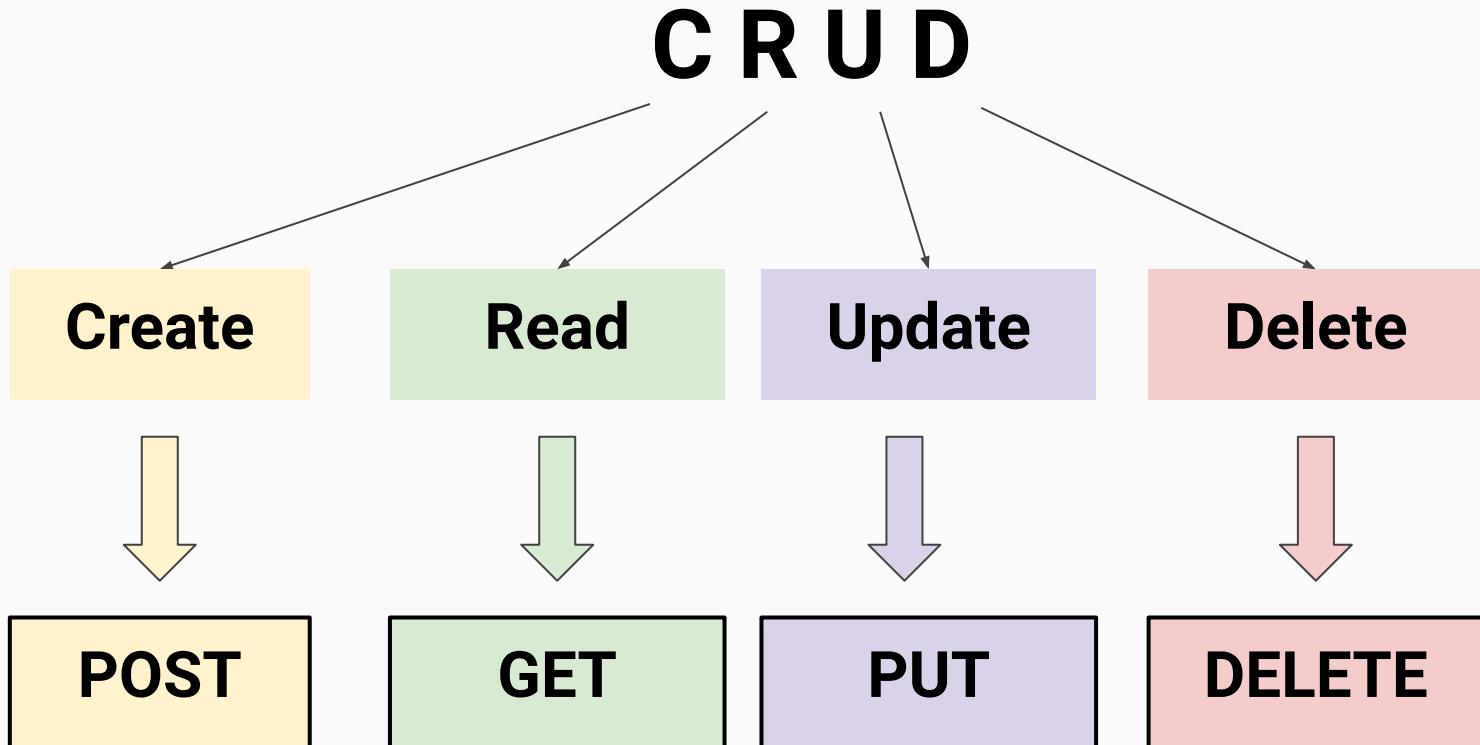
Request

Verb + URL + (Optional Payload)



What is this verb?

HTTP Request Verbs - CRUD concept



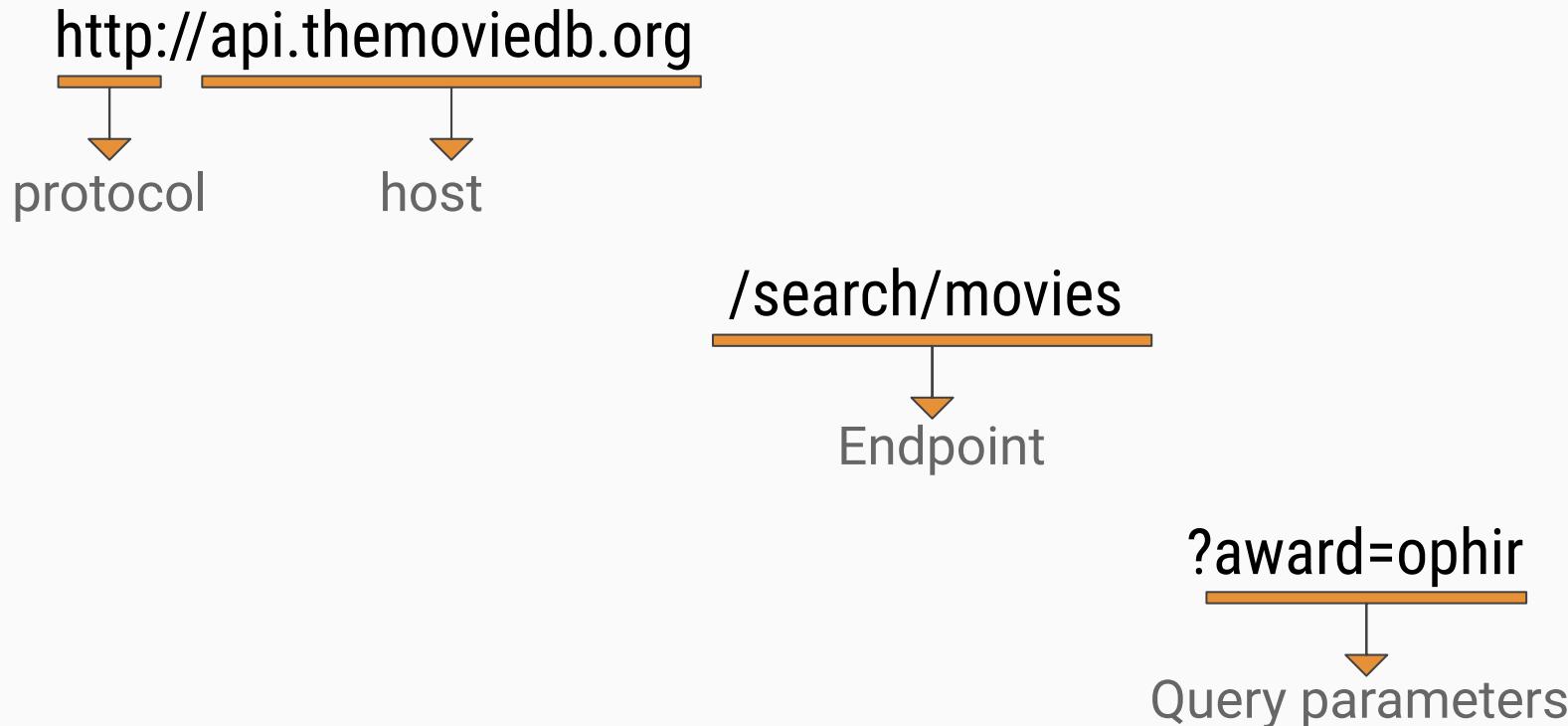
HTTP Verbs

- **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.

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 <>

Payload

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

XML <>

JSON { }

HTML <>

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 <>  
    "movie": {  
        "name": "Mister Potato",  
        "year": 2017,  
        "language": "English"  
    }  
JSON {}  
HTML <>
```

```
{  
    "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"
```

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 ON

JavaScript Object Notation

(Key : Value)

```
{  
    "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 ON

JavaScript Object Notation

(Key : (Key : Value))

{

JSON

Curly brackets marks
start and end of an Object.

}

JSON

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

JSON

Key (always a String)

:

Value

(number/String/boolean/null/
nested structure)

Key-value pairs are separated by
commas

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

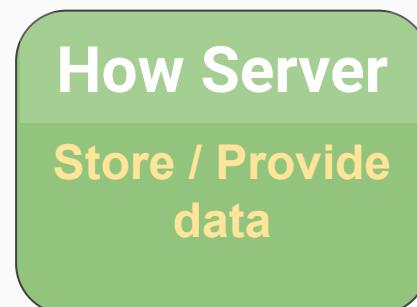
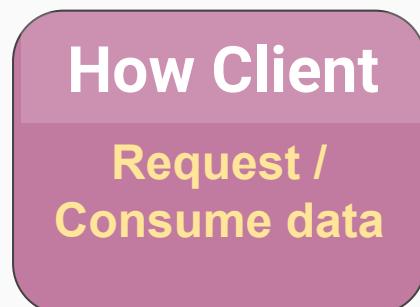
What dictates payload format?

- **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 **S**tate **T**ransfer

Stateless

Resources
Identified
by URL

CRUD
HTTP

Cacheable

Json
XML
Code

Questions?

Where do we get
The data from?

<http://?????????????>



API Services





The Movie Database API 3
[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

Select a different version ▾

Filter sections... ⋮

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES

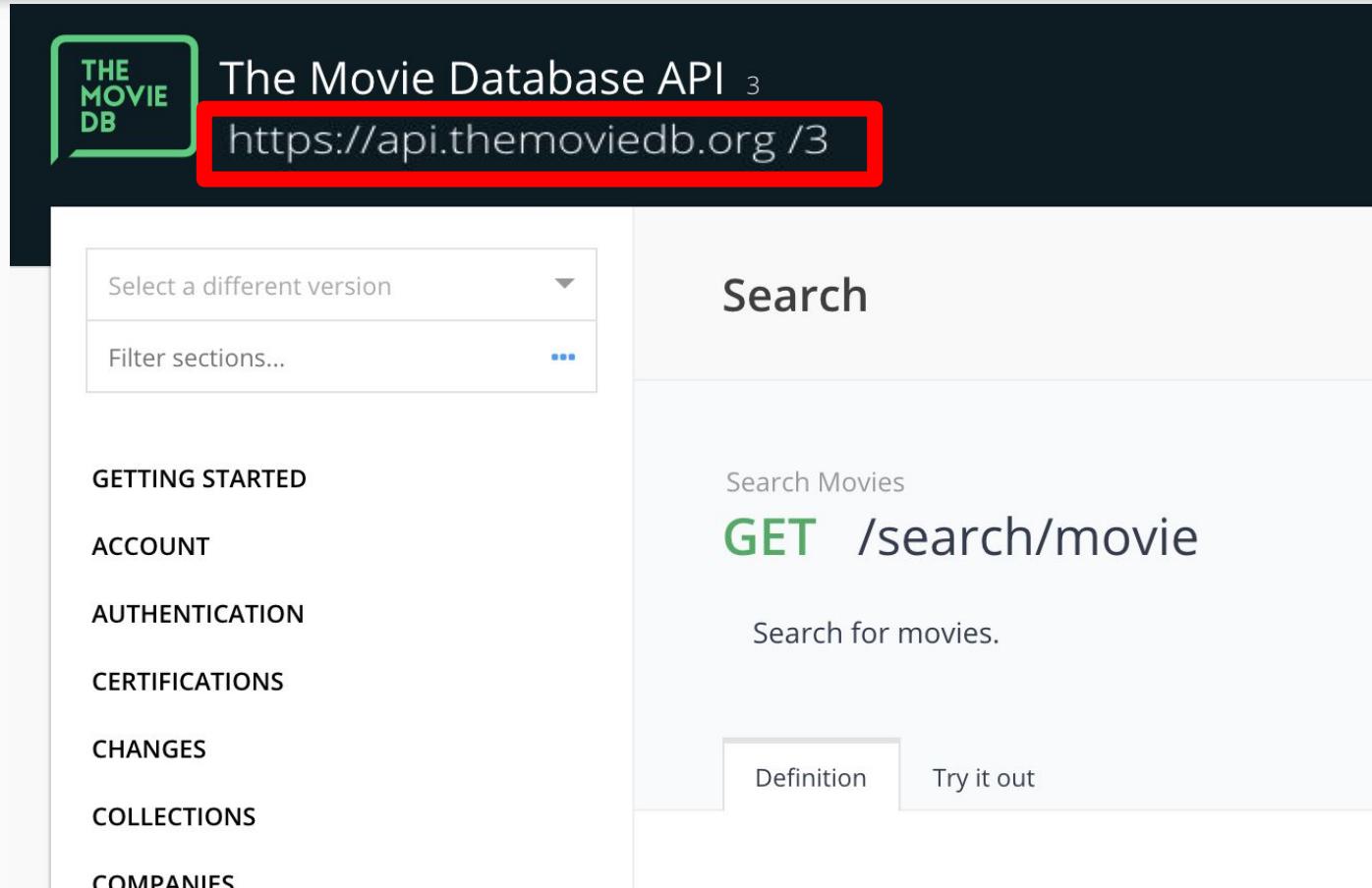
Search

Search Movies

GET /search/movie

Search for movies.

Definition Try it out



The Movie Database API 3

<https://api.themoviedb.org /3>

Select a different version ▾

Filter sections... ⚙

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES

Search

Search Movies

GET /search/movie

Search for movies.

Definition Try it out



The Movie Database API 3
[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

Select a different version ▾

Filter sections... ⚙

GET /search/movie

Search

Search Movies

Search for movies.

Definition Try it out

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES



The Movie Database API 3
[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

Select a different version ▾

Filter sections... ⋮

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES

Search

Search Movies

GET /search/movie

Search for movies.

Definition Try it out



The Movie Database API 3

[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

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 include 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
language	string	default: en	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 include 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



The Movie Database API 3

[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

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 include 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

Third-party networking libraries



An HTTP client for Android, Kotlin, and Java



- 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$AndroidBlockGuardPolicy.onNetwork(StrictM
ode.java:1099)
11112 11112 com.lookshot.splash    AndroidRuntime  at java.net.InetAddress.lookupHostName(InetAddress.java:432)
11112 11112 com.lookshot.splash    AndroidRuntime  at java.net.InetAddress.getAllByNameImpl(InetAddress.java:258)
11112 11112 com.lookshot.splash    AndroidRuntime  at java.net.InetAddress.getAllByName(InetAddress.java:236)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:71)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:50)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:49)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:48)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.<init>(HttpConnection.java:47)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpConnection.connect(HttpConnection.java:128 )
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpEngine.openConnection(HttpEngine.jav
a:308)
11112 11112 com.lookshot.splash    AndroidRuntime  at libcore.net.http.HttpsURLConnectionImpl$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

GETTING STARTED

Search Movies

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

COMPANIES

CONFIGURATION

CREDITS

DISCOVER

FIND

GENRES

GUEST SESSIONS

KEYWORDS

LISTS

MOVIES

NETWORKS

TRENDING

PEOPLE

REVIEWS

SEARCH

Variables

api_key

Your TMDb API key

optional

Query String

api_key

language

query

page

include_adult

region

year

primary_release_year

SEND REQUEST

https://api.themoviedb.org/3/search/movie?api_key=YOUR_API_KEY&language=en-US&query=Android&page=1

- 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":"/2hQEYyb6DXc3jAJzSBUxmIDf7jW.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":"/dv5fcQWbYFjwd8wfRu2j3vDztH1.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 formatter]

The screenshot shows a JSON Formatter application window. On the left is a large text area containing a complex JSON object. The JSON describes a scene from a movie or TV show, mentioning characters like Goku, Trunks, and Vegeta, and locations like a mall where Gokus are shopping. It includes details about the plot, such as characters being awoken by computers and becoming super beings. On the right is a sidebar with several buttons: "Load Data", "Validate", "Format / Beautify", "Minify / Compact", "Convert JSON to...", and "Download". Below these buttons is a dropdown menu set to "2 Tab Space". At the bottom left, there are status indicators: "Ln:1" and "Col:12790".

```
around.", "release_date": "2016-03-14"}, {"vote_count": 143, "id": 39104, "video": false, "vote_average": 6.5, "title": "Dragon Ball Z: Super Android 13", "popularity": 7.164, "poster_path": "/2nyDB3qWHSFEBibtvAlFr99L0f.jpg", "original_language": "ja", "original_title": "龍珠Z 極限バトルⅡ 三大超サイヤ人", "genre_ids": [28, 16], "backdrop_path": "/uTd9eyJlmIbbIRbyzIxpzMsdq.jpg", "adult": false, "overview": "Dr. Gero's Androids #13, #14, and #15 are awakened by the laboratory computers and immediately head to the mall where Gokus are shopping. After Goku, Trunks, and Vegeta defeat #14 and #15, #13 absorbs their inner computers and becomes a super being greater than the original three separately were. Now it is up to Goko to stop him.", "release_date": "1992-07-11"}, {"vote_count": 0, "id": 481902, "video": false, "vote_average": 0, "title": "Me & My Sex Android 1 - Good Bye to a Flirtatious Boyfriend", "popularity": 0.6, "poster_path": "/os1BrfUqmFIlo8WlcCYPvOlyM.jpg", "original_language": "ja", "original_title": "Love Pop Collection: Ren'ai Complex", "genre_ids": [10749, 18], "backdrop_path": null, "adult": false, "overview": "Hitomi Anzai is an ordinary collage girl, enjoying an ongoing happy relationship with her boyfriend Matsuyama. But Hitomi had a big step she had to take. This was her very first love. In another words, she was a virgin. Not wanting her boyfriend to think of her as a burden, she decided to rent a \"Rental Boy\" named \"Ack-kun.\" She lost her virginity, and had made love with Matsuyama. Everything seemed to be going well, except one thing made Hitomi rethink about what love is all about.", "release_date": "2011-01-01"}, {"vote_count": 0, "id": 292841, "video": false, "vote_average": 0, "title": "Androids", "popularity": 0.6, "poster_path": "/fmy990QZBiiA9j96mFBgx6AP0K.jpg", "original_language": "es", "original_title": "Androids", "genre_ids": [], "backdrop_path": null, "adult": false, "overview": "About an androgynous preteen boy, who sits while peeing and has fantasies about aliens with his deformed and badly handicapped teenage neighbor, while his macho brother rapes her, playing Mario Bros.", "release_date": "2010-01-01"}, {"vote_count": 2, "id": 438104, "video": true, "vote_average": 6.3, "title": "Doctor Who: The Androids of Tara", "popularity": 0.9, "poster_path": "/16sSnbfp4LBBohjf2wLM6ArDokN.jpg", "original_language": "en", "original_title": "Doctor Who: The Androids of Tara", "genre_ids": [28, 12, 18, 878], "backdrop_path": "/aoygadMALuAHHmxZbc8CK7hzza.jpg", "adult": false, "overview": "Finding the fourth segment of the Key to Time was simple enough, but holding onto it may be another matter. The Doctor and Romana find themselves embroiled in the political games of the planet Tara, where doubles, android or otherwise, complicate the coronation of Prince Reynart.", "release_date": "1978-11-25"}, {"vote_count": 6527, "id": 78, "video": false, "vote_average": 7.9, "title": "Blade Runner", "popularity": 24.897, "poster_path": "/p647tbZGCElxQHpAMmDHKNJLH2.jpg", "original_language": "en", "original_title": "Blade Runner", "genre_ids": [378, 18, 53], "backdrop_path": "/5hj0XDxCxE3agfp1H3n7HQ9P9rlfU.jpg", "adult": false, "overview": "In the smog-choked dystopian Los Angeles of 2019, blade runner Rick Deckard is called out of retirement to terminate a quartet of replicants who have escaped to Earth seeking their creator for a way to extend their short life spans.", "release_date": "1982-06-25"}]]
```

Ln:1 Col:12790

JSON formatting

Json Formatter

[JSON formatter]

```
around.", "release_date": "2016-03-14"}, {"vote_count": 143, "id": 39104, "video": false  
, "vote_average": 6.5, "title": "Dragon Ball Z: Super Android 13", "popularity": 7.164  
, "poster_path": "/2nyDB3qWHSFEBibtvAlFr99L0f.jpg", "original_language": "ja"  
, "original_title": "ドラゴンボールZ 極限バトル! 三大超サイヤ人", "genre_ids": [28, 16]  
, "backdrop_path": "/uTd9eyJ1mIbbIRbyzIxpzMsdq.jpg", "adult": false, "overview": "Dr.  
Gero's Androids #13, #14, and #15 are awakened by the laboratory computers and  
immediately head to the mall where Goku is shopping. After Goku, Trunks, and Vegeta  
defeat #14 and #15, #13 absorbs their inner computers and becomes a super being greater  
than the original three separately were. Now it is up to Goku to stop him."  
, "release_date": "1992-07-11"}, {"vote_count": 0, "id": 481902, "video": false, "vote_average":  
0, "title": "Me & My Sex Android 1 - Good Bye to a Flirtable Boyfriend", "popularity": 0.6  
, "poster_path": "/osBrfUqmFIlo8W1cCYPvOlymY.jpg", "original_language": "ja"  
, "original_title": "Love Pop Collection: Ren'ai Complex", "genre_ids": [10749, 18]  
, "backdrop_path": null, "adult": false, "overview": "Hitomi Anzai is an ordinary collage  
girl, enjoying an ongoing happy relationship with her boyfriend Matsuyama. But Hitomi  
had a big step she had to take. This was her very first love. In another words, she was  
a virgin. Not wanting her boyfriend to think of her as a burden, she decided to rent a  
"Rental Boy" named "Ack-kun." She lost her virginity, and had made love with Matsuyama.  
Everything seemed to be going well, except one thing made Hitomi rethink about what  
love is all about.", "release_date": "2011-01-01"}, {"vote_count": 0, "id": 292841, "video":  
false, "vote_average": 0, "title": "Androids", "popularity": 0.6, "poster_path": "  
/fm990QZBtA9j96mFBgx6AP0K.jpg", "original_language": "es", "original_title": "Androids"  
, "genre_ids": [], "backdrop_path": null, "adult": false, "overview": "About an androgynous  
preteen boy, who while peeing and has fantasies about aliens with his deformed and  
badly handicapped teenage neighbor, while his macho brother rapes her, playing Mario  
Bro's.", "release_date": "2010-01-01"}, {"vote_count": 2, "id": 438104, "video": true  
, "vote_average": 6.3, "title": "Doctor Who: The Androids of Tara", "popularity": 0.9  
, "poster_path": "/165Snbfp4LBBohjf2wLM6ArDokM.jpg", "original_language": "en"  
, "original_title": "Doctor Who: The Androids of Tara", "genre_ids": [28, 12, 18, 878]  
, "backdrop_path": "/aoygADMuAHHmxZbc8CK7hzza.jpg", "adult": false, "overview": "Finding  
the fourth segment of the Key to Time was simple enough, but holding onto it may be  
another matter. The Doctor and Romana find themselves embroiled in the political games  
of the planet Tara, where doubles, android or otherwise, complicate the coronation of  
Prince Reynart.", "release_date": "1978-11-25"}, {"vote_count": 6527, "id": 78, "video": false  
, "vote_average": 7.9, "title": "Blade Runner", "popularity": 24.897, "poster_path": "  
/p64TtbZGCElxQHPAMmDHkNjLH2.jpg", "original_language": "en", "original_title": "Blade  
Runner", "genre_ids": [878, 18, 53], "backdrop_path": "/5hj0XDcxE3qGfp1H3n7HQ9Prlfu.jpg"  
, "adult": false, "overview": "In the smog-choked dystopian Los Angeles of 2019, blade  
runner Rick Deckard is called out of retirement to terminate a quartet of replicants  
who have escaped to Earth seeking their creator for a way to extend their short life  
spans.", "release_date": "1982-06-25"}]]
```

XML FORMATTER JSON PRETTY PRINT JSON EDITOR SAVE RECENT LINKS LOGIN

Load Data Validate Format / Beautify Minify / Compact Convert JSON to... Download

```
1  {
2     "page": 1,
3     "total_results": 32,
4     "total_pages": 2,
5     "results": [
6         {
7             "vote_count": 26,
8             "id": 38849,
9             "video": false,
10            "vote_average": 5.5,
11            "title": "Android",
12            "popularity": 2.171,
13            "poster_path": "/jsRWNS6VublwSFhEller2C4tei.jpg",
14            "original_language": "en",
15            "original_title": "Android",
16            "genre_ids": [
17                878,
18                53
19            ],
20            "backdrop_path": "/veTb0hmlRu1kOgAFujNVlbe2Qaj.jpg",
21            "adult": false,
22            "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.",
23            "release_date": "1982-10-15"
24        },
25        {
26            "vote_count": 34,
27            "id": 249021,
28            "video": false,
29            "vote_average": 3.7,
30            "title": "Android Cop",
31            "popularity": 2.148,
32            "poster_path": "/njUuUzw9MpRNQxogDL7dMro9Qcs.jpg",
33            "original_language": "en",
34            "original_title": "Android Cop",
35            "genre_ids": [
36                28,
37                878
38        ]
```

Creating Class model For JSON

JSON Movie Object

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

JSON Movie Object

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

Java Movie class

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

JSON Movie Object

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

JSON key



Java Movie class

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

Class Field

JSON Movie Object

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

JSON key



Java Movie class

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

Class Field

JSON Movie Object

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

JSON key

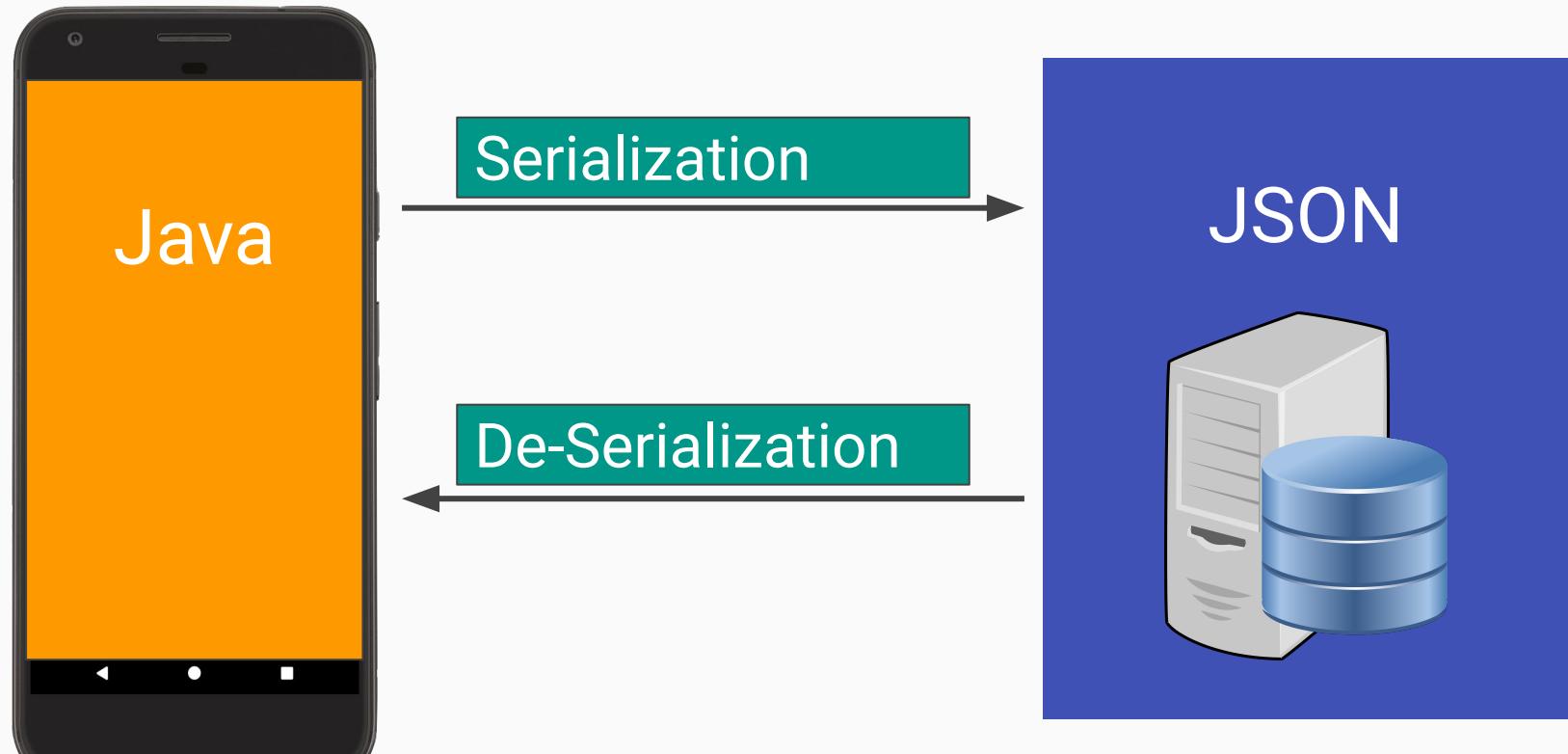


Java Movie class

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

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++) {  
    Movie movie = new Movie();  
    movie.setTitle(result.getJSONObject(i).getString("title"));  
    movie.setYear(result.getJSONObject(i).getLong("year"));  
    movie.setRating(result.getJSONObject(i).getDouble("rating"));  
    movie.setPlot(result.getJSONObject(i).getString("plot"));  
    movie.setGenre(result.getJSONObject(i).getString("genre"));  
    movies.add(movie);  
}  
return movies;
```

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

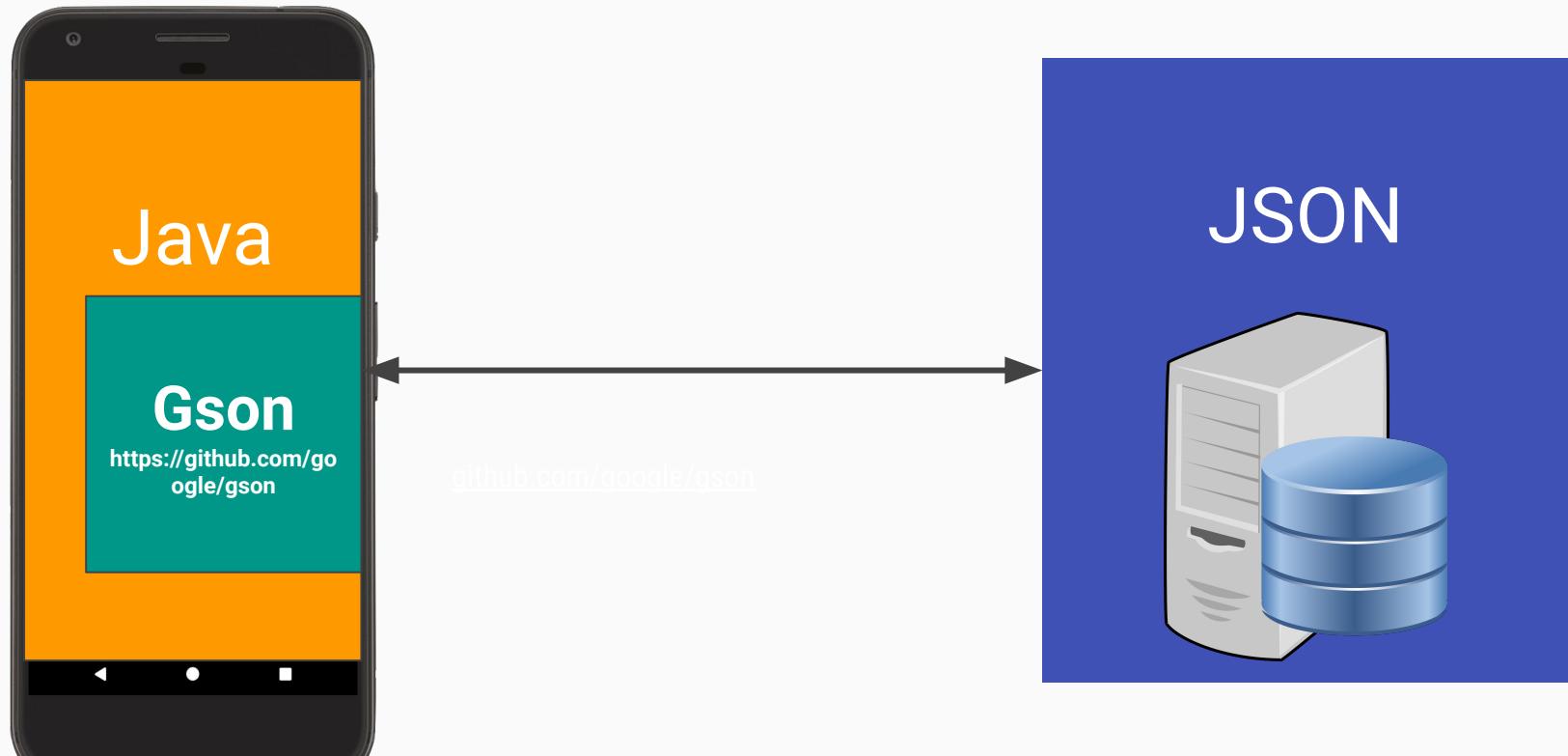
- GSON
- Moshi
- Jackson
-

Add GSON

build.gradle

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

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



JSON to data class model

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

JSON to data class model

```
{                                     class Movie {  
    "id" : 249021,                     int id;  
    "title" : "Android Cop"           String title;  
}  
}
```

JSON to data class model

```
{                                     class Movie {  
    "id" : 249021,                     int id;  
    "title" : "Android Cop"           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;  
}  
}
```

JSON

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

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

JSON

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

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

JSON

```
{  
    "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;  
}
```

JSON

```
{  
    "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;  
}
```

JSON

```
{  
    "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;  
}
```

JSON

```
"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;  
}  
}
```

JSON

```
"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();
```

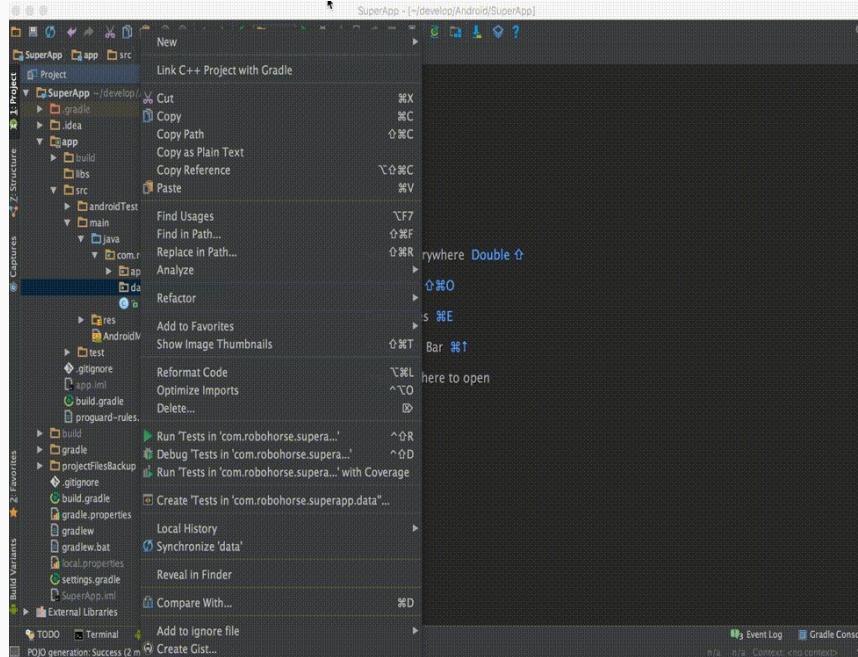
```
String resultFromJavaObject = gson.toJson(videoResult);
```

TIP!

Use
POJO generator
plugin!

TIP!

i.e. RoboPOJOGenerator
<https://github.com/robohorse/RoboPOJOGenerator>



Questions?

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

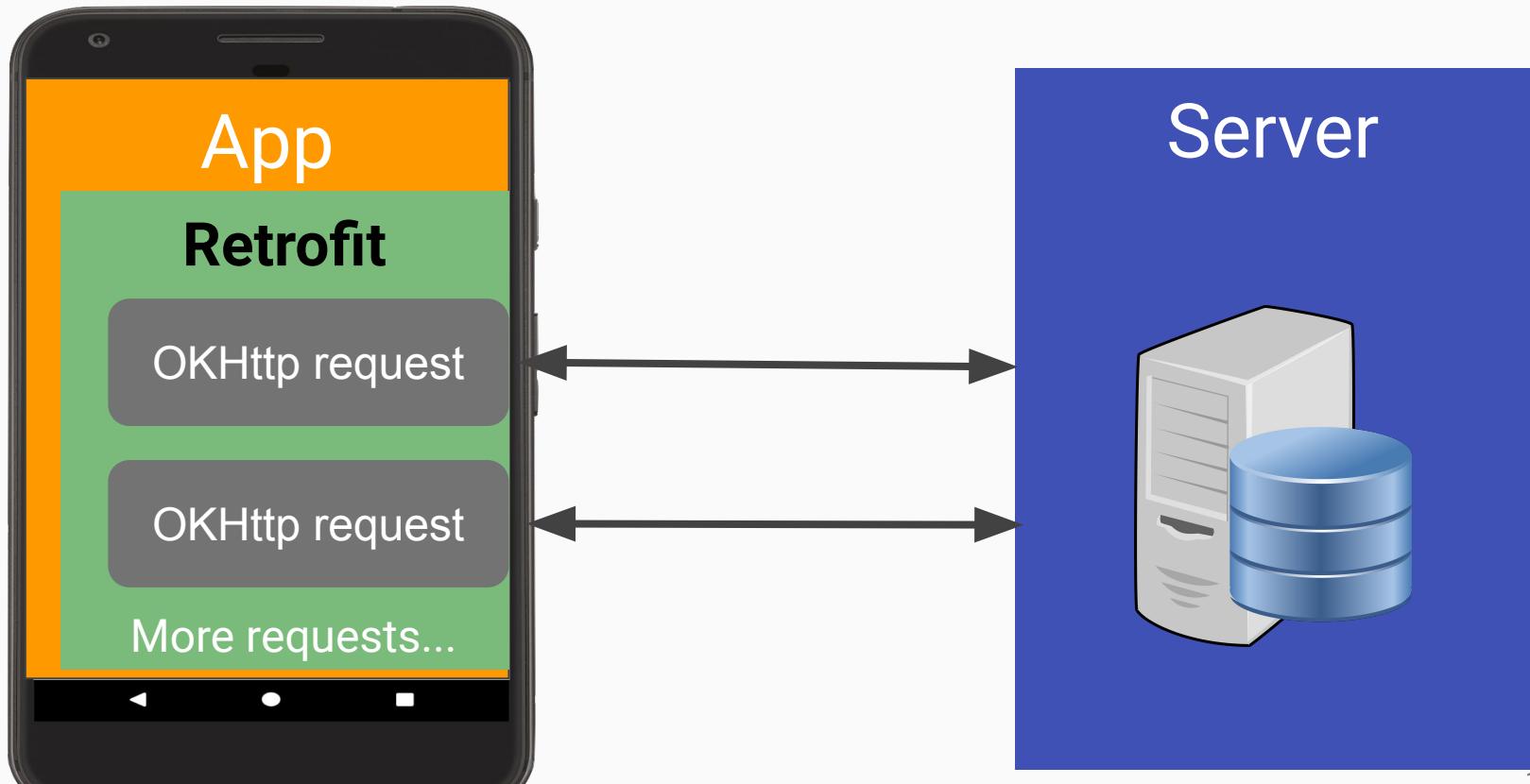
- Type safe OkHttp client
- Threading
- Caching
- Error handling
- Serialization / Deserialization
- And more...

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

build.gradle

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



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";

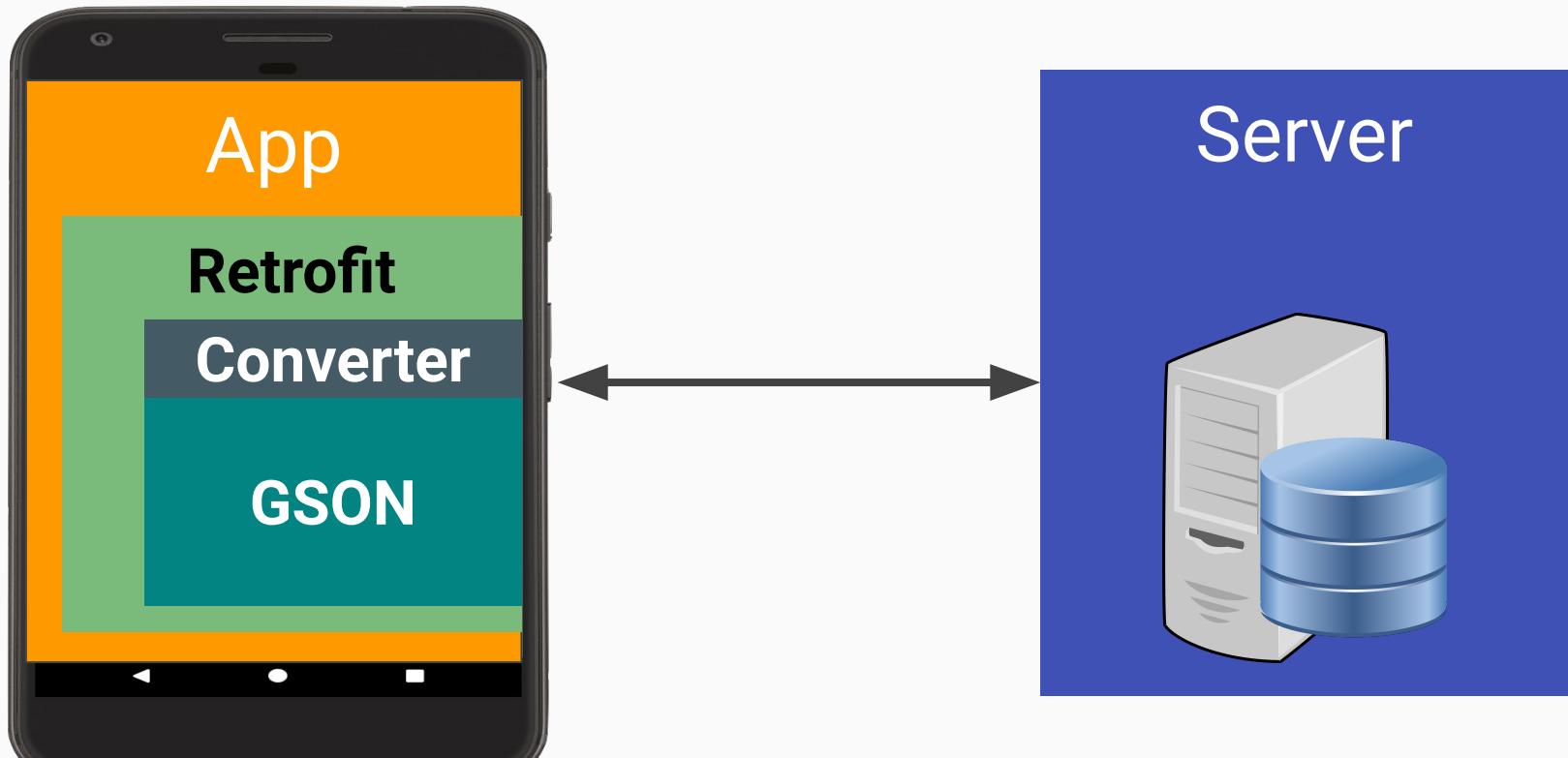
    private static Retrofit retrofit = new Retrofit.Builder().
        baseUrl(BASE_URL)
        .build();

}
```

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

- [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

```
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 converters 
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 - OkHttpClient

```
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 - OkHttpClient

```
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 converters 
5. Http client (optional) 
6. Interface representing server APIs

Body-less functions -> here we come

The Movie Database API 3

[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

Select a different version ▾

Filter sections... ⚙

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

CHANGES

COLLECTIONS

COMPANIES

Search

Search Movies

GET /search/movie

Search for movies.

Definition Try it out

```
public interface MoviesService {
```

```
}
```

Retrofit

```
public interface MoviesService {  
    Call<VideosListResult> searchMoviesByTitle();  
}
```

Retrofit

```
public interface MoviesService {  
  
    Call<VideosListResult> searchMoviesByTitle();  
  
}
```

```
public interface MoviesService {  
    Call<VideosListResult> searchMoviesByTitle();  
}
```

What is this call object?

- `Call<MoviesList>`

`Retrofit` will take the response body and try to convert it to `Java` objects.

- `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

- **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>**

Skip 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();  
}
```

```
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);  
}
```

```
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);  
}
```

The Movie Database API 3[https://api.themoviedb.org /3](https://api.themoviedb.org/3)

Select a different version

Filter sections...
...

Movies

GETTING STARTED

ACCOUNT

AUTHENTICATION

CERTIFICATIONS

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);  
}
```

Headers

Content-Type

application/json;charset=utf-8

required

```
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);  
}
```

Headers

Content-Type

application/json;charset=utf-8

required

```
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);  
}
```

Headers

Content-Type

application/json;charset=utf-8

required

```
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);  
}
```

Headers

Content-Type

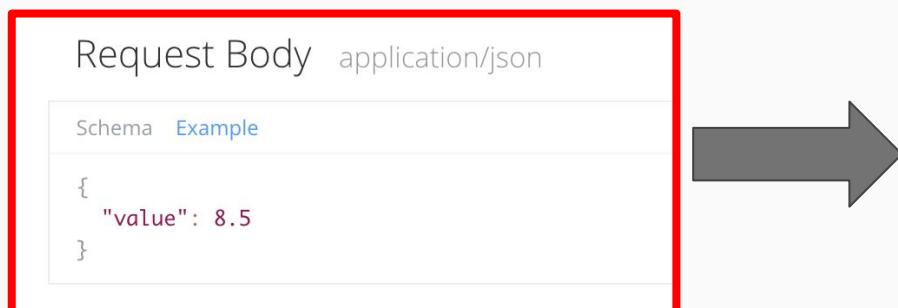
application/json;charset=utf-8

required

```
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);  
}
```

Request Body application/json

Schema	Example
{ "value": 8.5 }	



```
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);  
}
```

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(@Path("movie_id") String movieId,  
                           @Body String rating);  
}
```

Request Body application/json

Schema Example

```
{  
    "value": 8.5  
}
```



```
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);
```

Request Body application/json

Schema	Example
{ "value": 8.5 }	



Example Input:

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

```
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);
```

Request Body application/json

Schema	Example
{ "value": 8.5 }	



Example Input:

```
class Rating {  
    float value;  
}
```

```
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);
```

Request Body application/json

Schema	Example
{ "value": 8.5 }	



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 converters 
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!

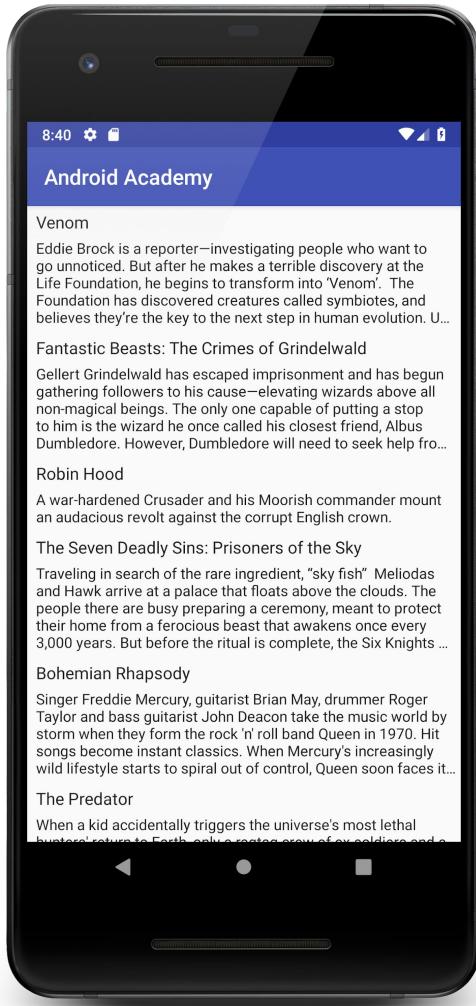
```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {  
  
    public void onBindViewHolder(MovieViewHolder holder, int position) {  
  
    }  
}
```

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());  
  
    }  
}
```



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/>



glide

<https://github.com/bumptech/glide>

Images loading

Android App
Java



Server

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

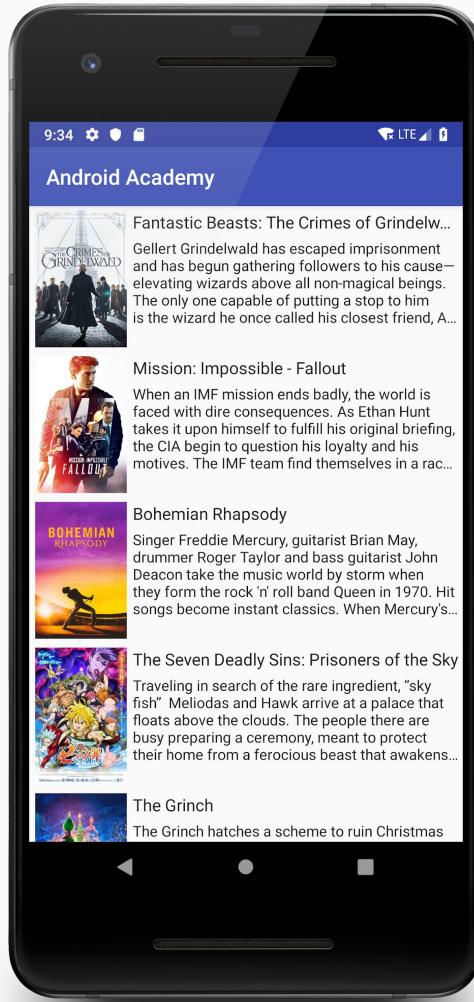
```
public class MoviesViewAdapter extends RecyclerView.Adapter<MoviesViewAdapter.ViewHolder> {  
  
    public void onBindViewHolder(MovieViewHolder holder, int position) {  
        MovieModel movie = movies.get(position)  
        tvTitle.setText(movie.getTitle());  
        tvOverview.setText(movie.getOverview());  
  
    }  
}
```

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);  
        }  
    }  
}
```

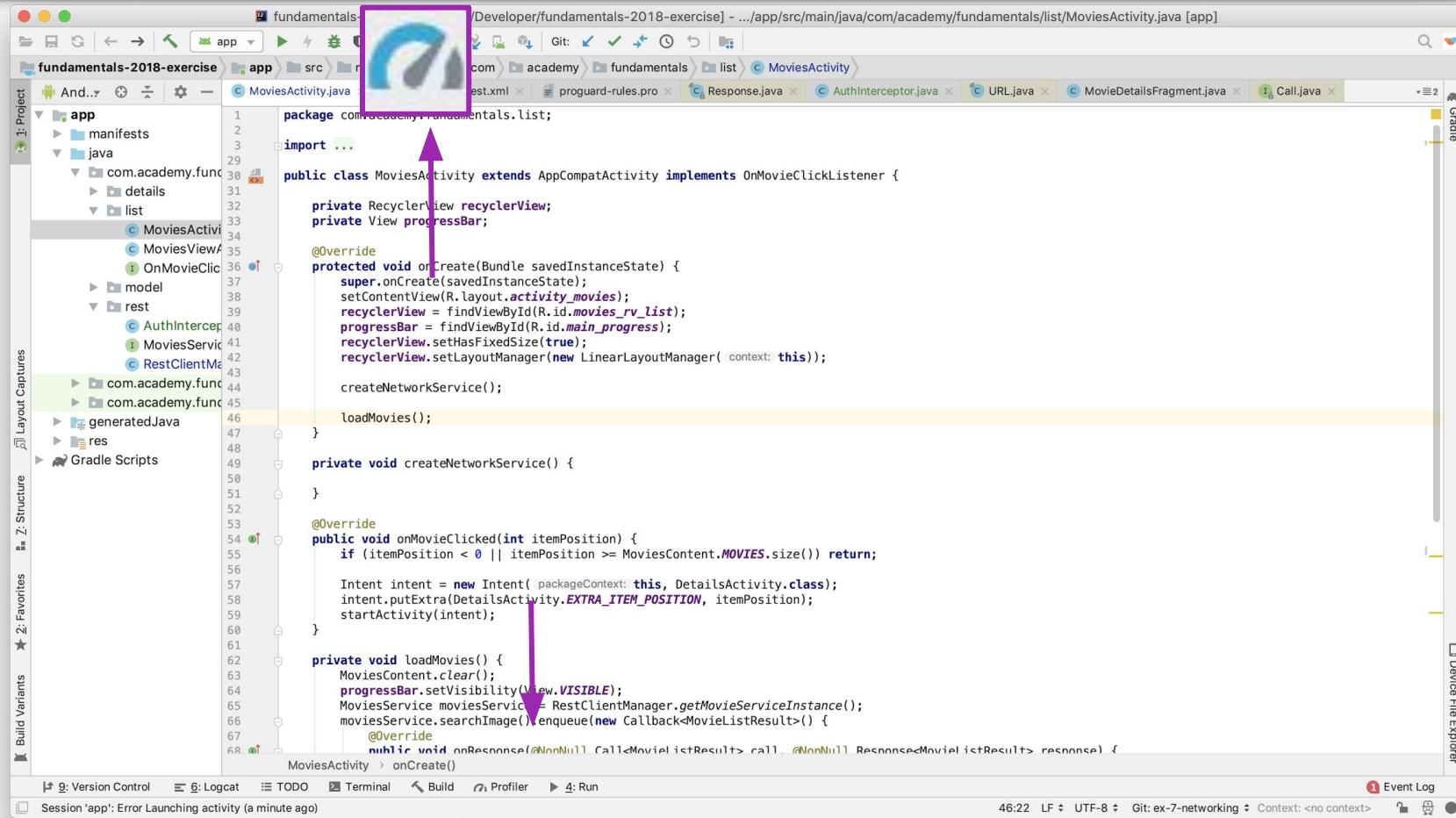


A composite image. On the right side, Snoop Dogg is shown from the chest up, looking slightly upwards and to his left with a serious expression. He has a mustache and is wearing a dark zip-up hoodie over a light-colored t-shirt. A silver chain necklace with a large, light-colored pendant hangs around his neck. On the left side of the image, there is a close-up of a vibrant bouquet of flowers, including red, yellow, and green blossoms, arranged in a circular pattern.

We got this

Network Profiler

Network profiler



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "fundamentals-2018-exercise". The "app" module contains Java files under "com.academy.fundamentals.list".
- MoviesActivity.java:** The current file is open, showing the following code:

```
package com.academy.fundamentals.list;

import ...

public class MoviesActivity extends AppCompatActivity implements OnMovieClickListener {

    private RecyclerView recyclerView;
    private View progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_movies);
        recyclerView = findViewById(R.id.movies_rv_list);
        progressBar = findViewById(R.id.main_progress);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(context: this));

        createNetworkService();

        loadMovies();
    }

    private void createNetworkService() {
    }

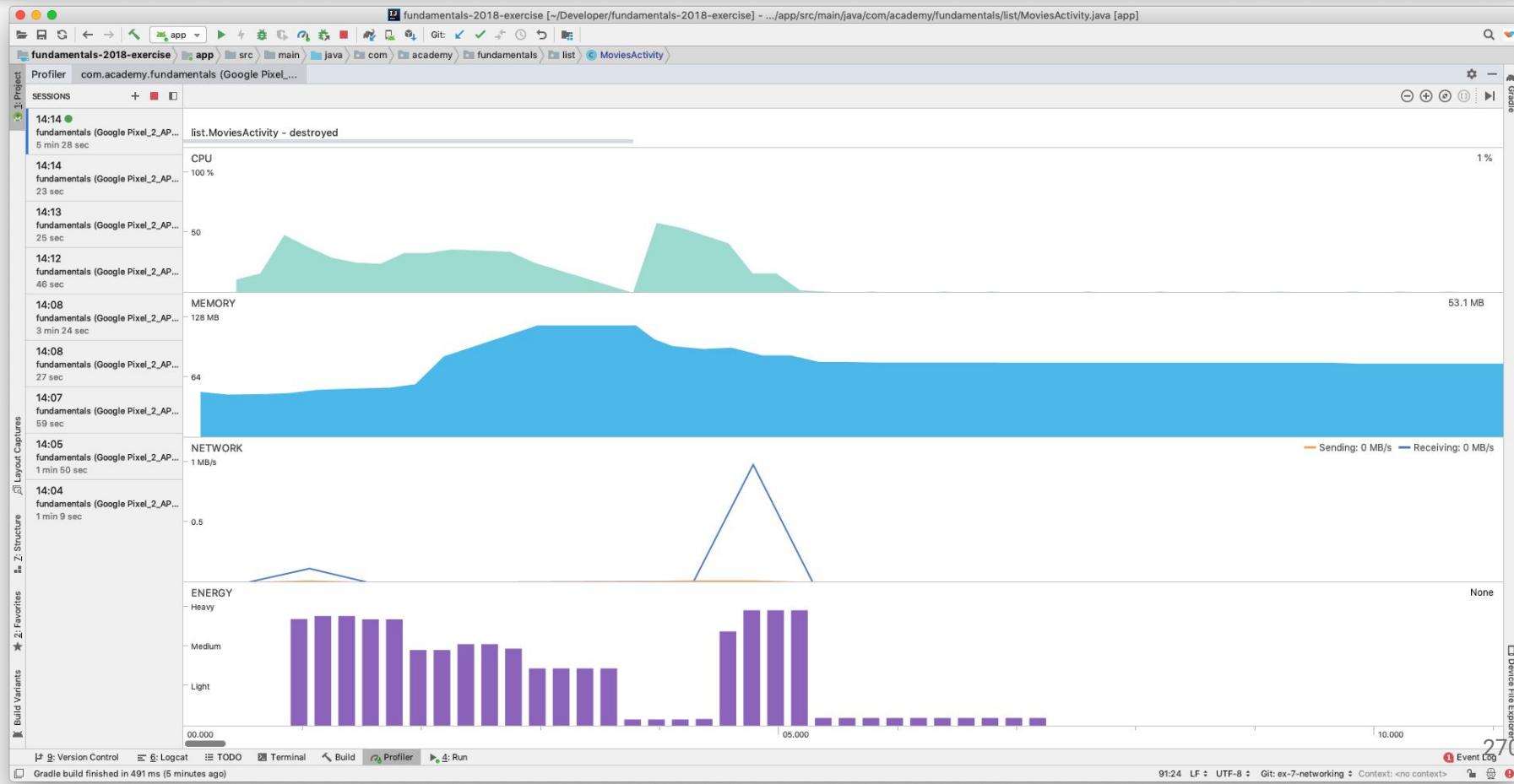
    @Override
    public void onMovieClicked(int itemPosition) {
        if (itemPosition < 0 || itemPosition >= MoviesContent.MOVIES.size()) return;

        Intent intent = new Intent(packageContext: this, DetailsActivity.class);
        intent.putExtra(DetailsActivity.EXTRA_ITEM_POSITION, itemPosition);
        startActivity(intent);
    }

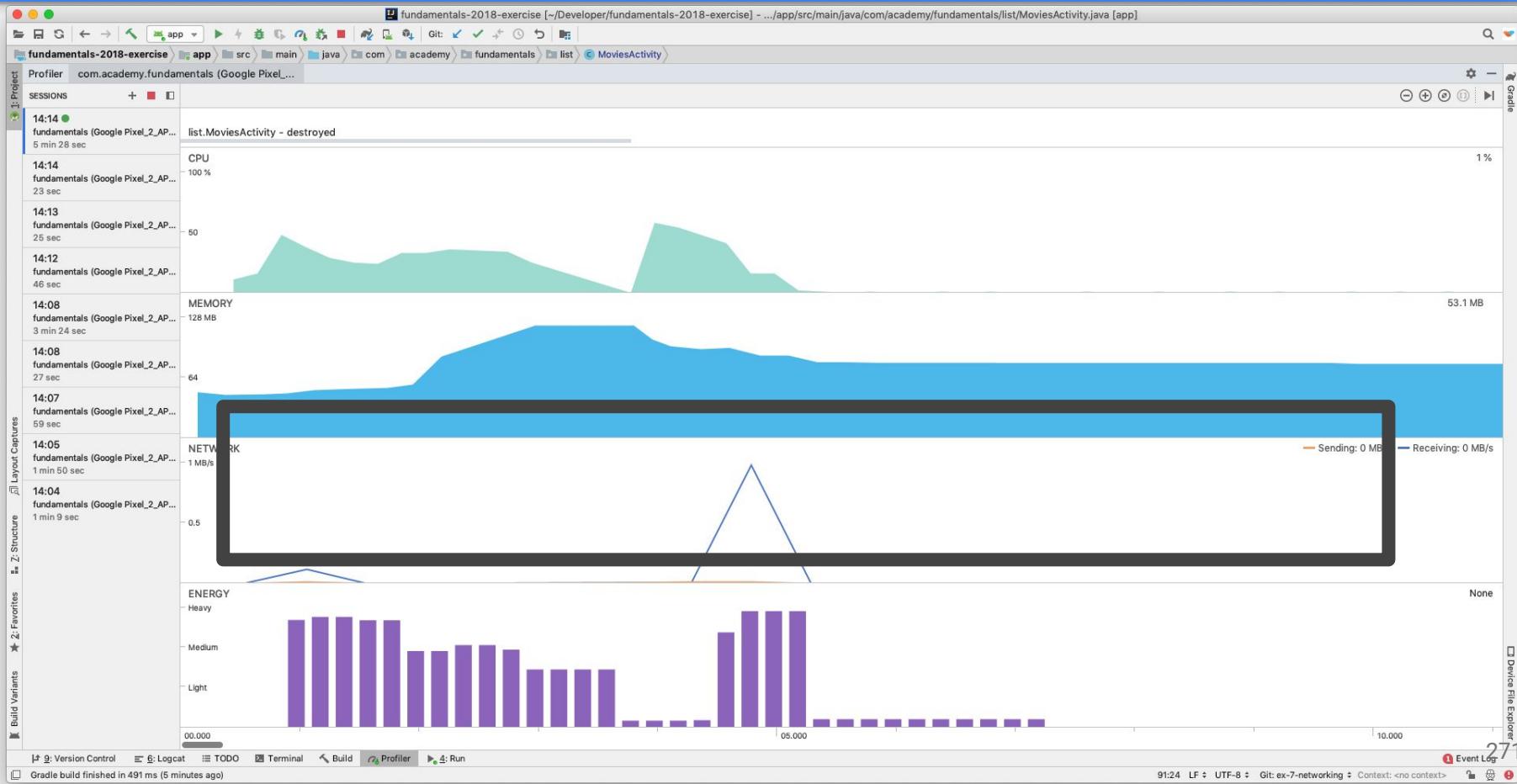
    private void loadMovies() {
        MoviesContent.clear();
        progressBar.setVisibility(View.VISIBLE);
        moviesService = RestClientManager.getMovieServiceInstance();
        moviesService.searchImage().enqueue(new Callback<MovieListResult>() {
            @Override
            public void onResponse(@NotNull Call<MovieListResult> call, @NotNull Response<MovieListResult> response) {
                MoviesActivity.this.onCreate();
            }
        });
    }
}
```

- Toolbar:** The "Profiler" button is highlighted with a purple arrow.
- Code Editor:** The code editor shows the "MoviesActivity.java" file with syntax highlighting and error markers.
- Sidebar:** The "Layout Captures" tab is selected.
- Bottom Bar:** The "Event Log" tab is visible.
- Page Number:** The page number 269 is in the bottom right corner.

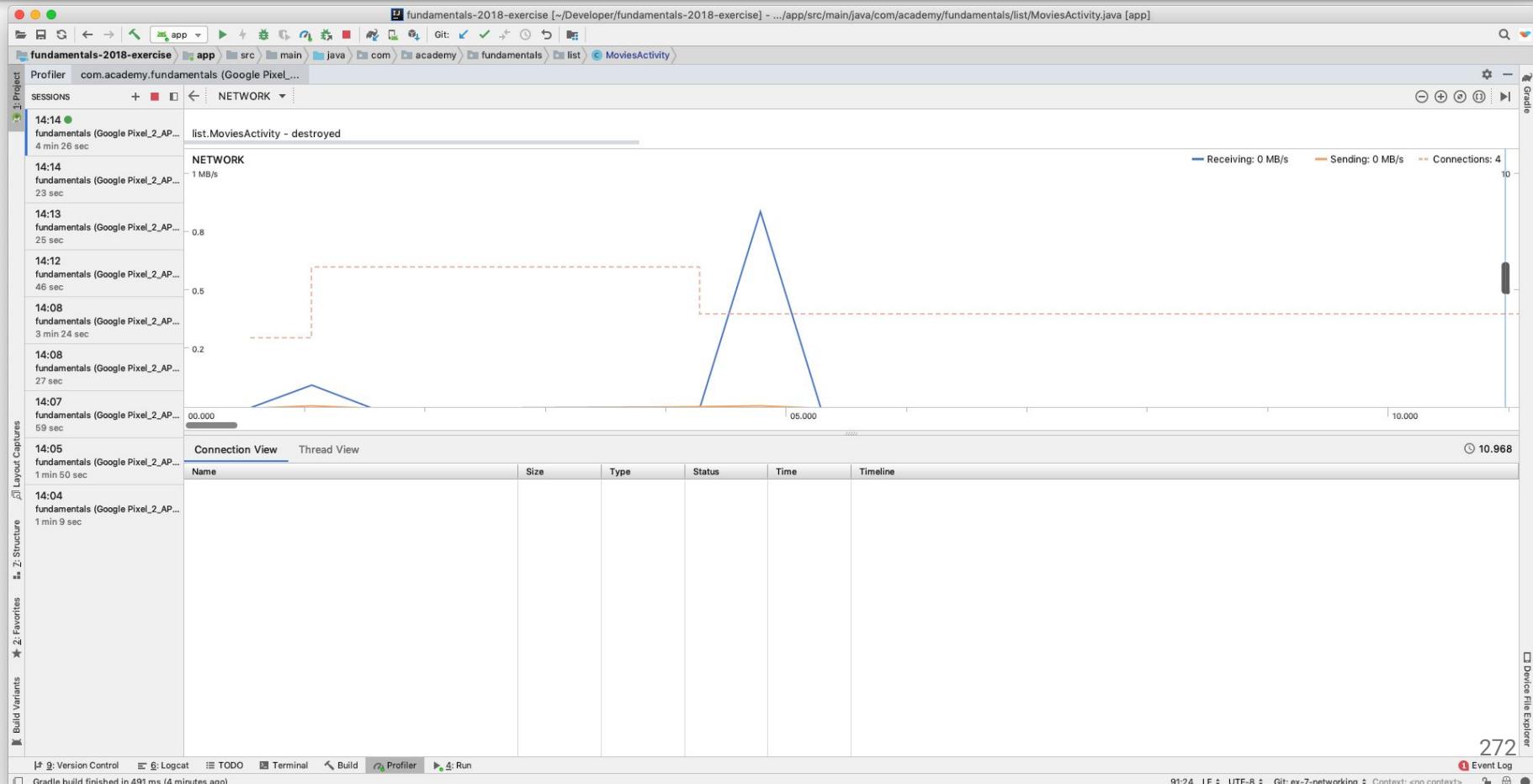
Network profiler



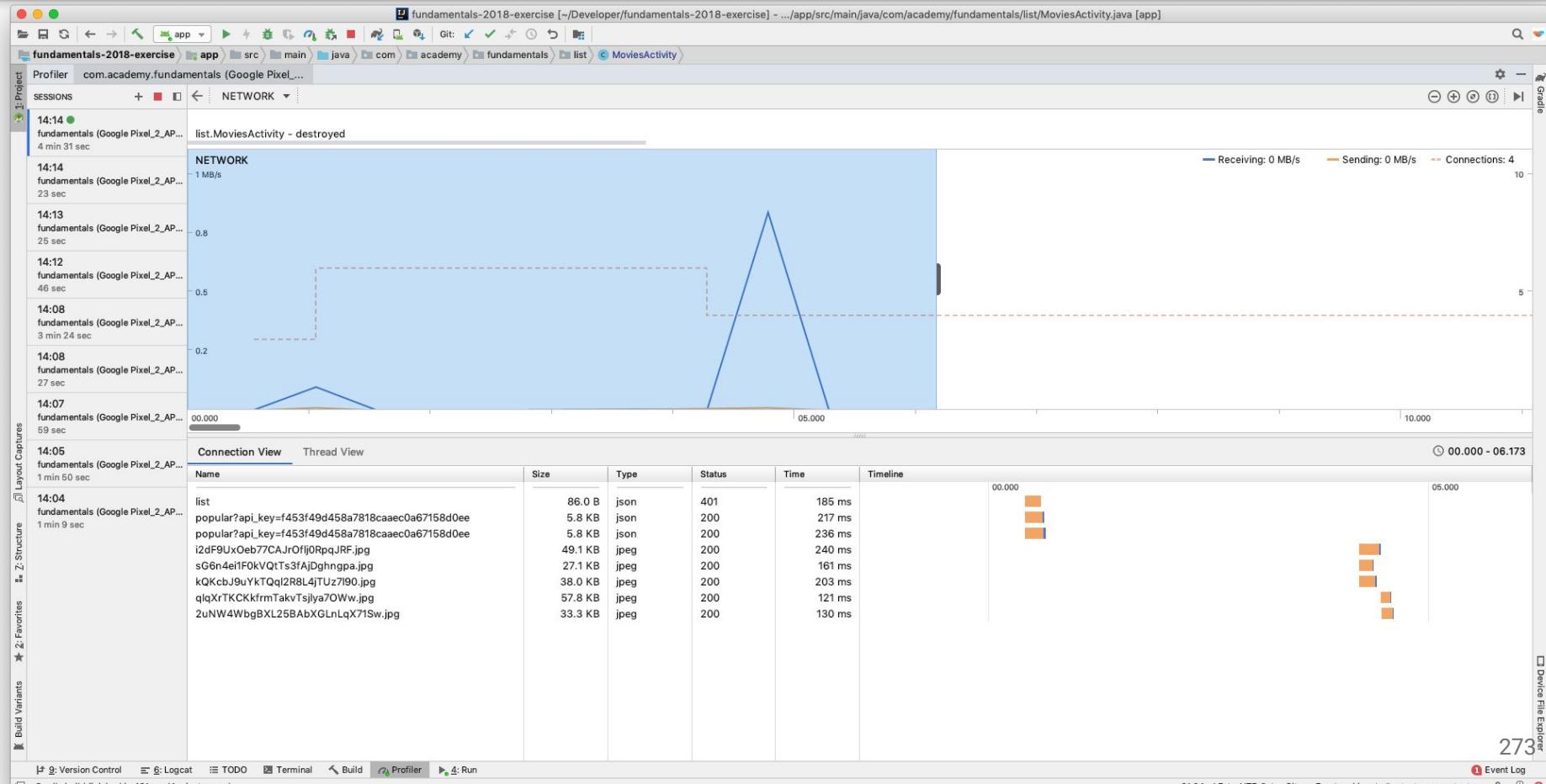
Network profiler



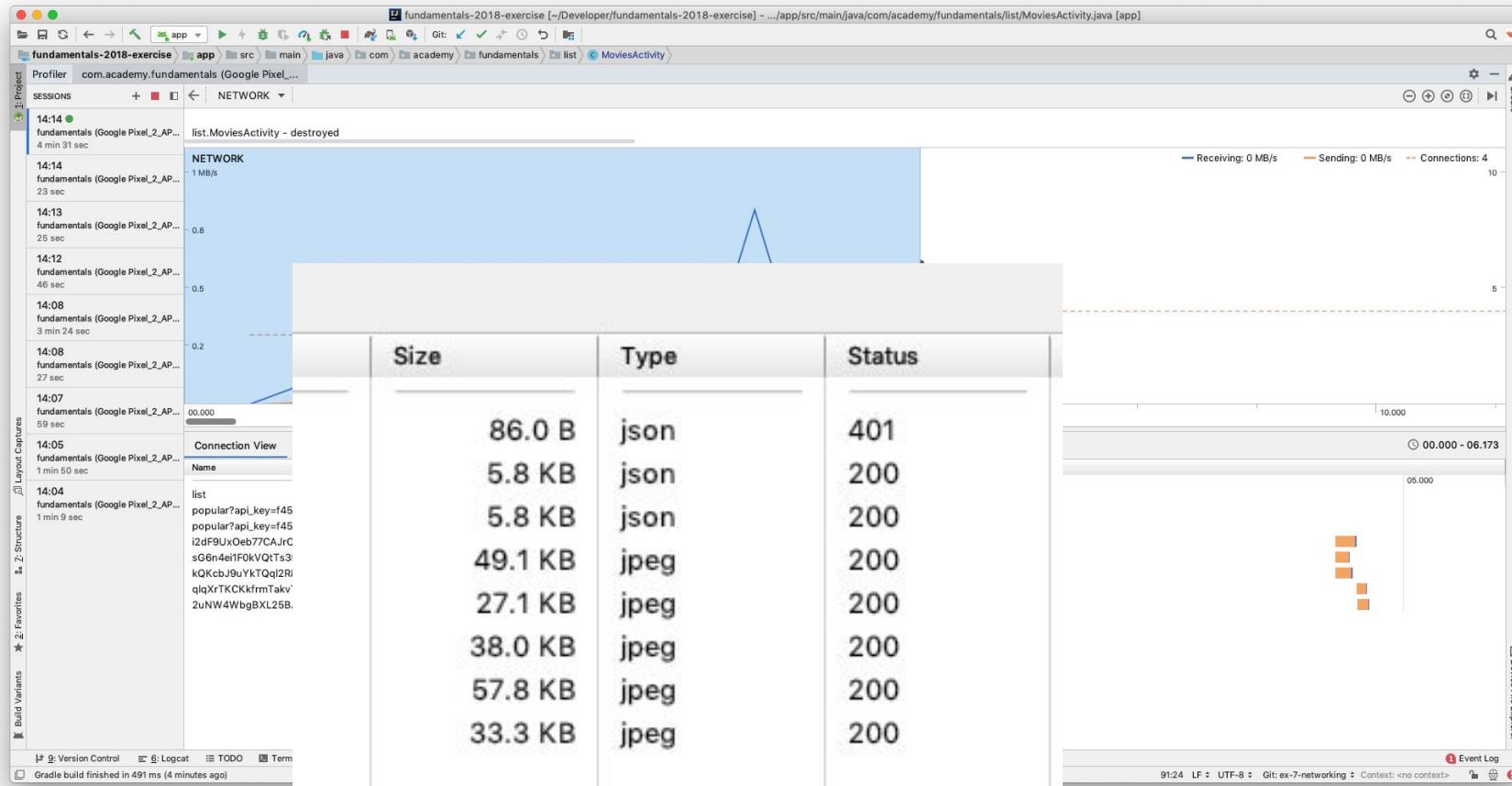
Network profiler



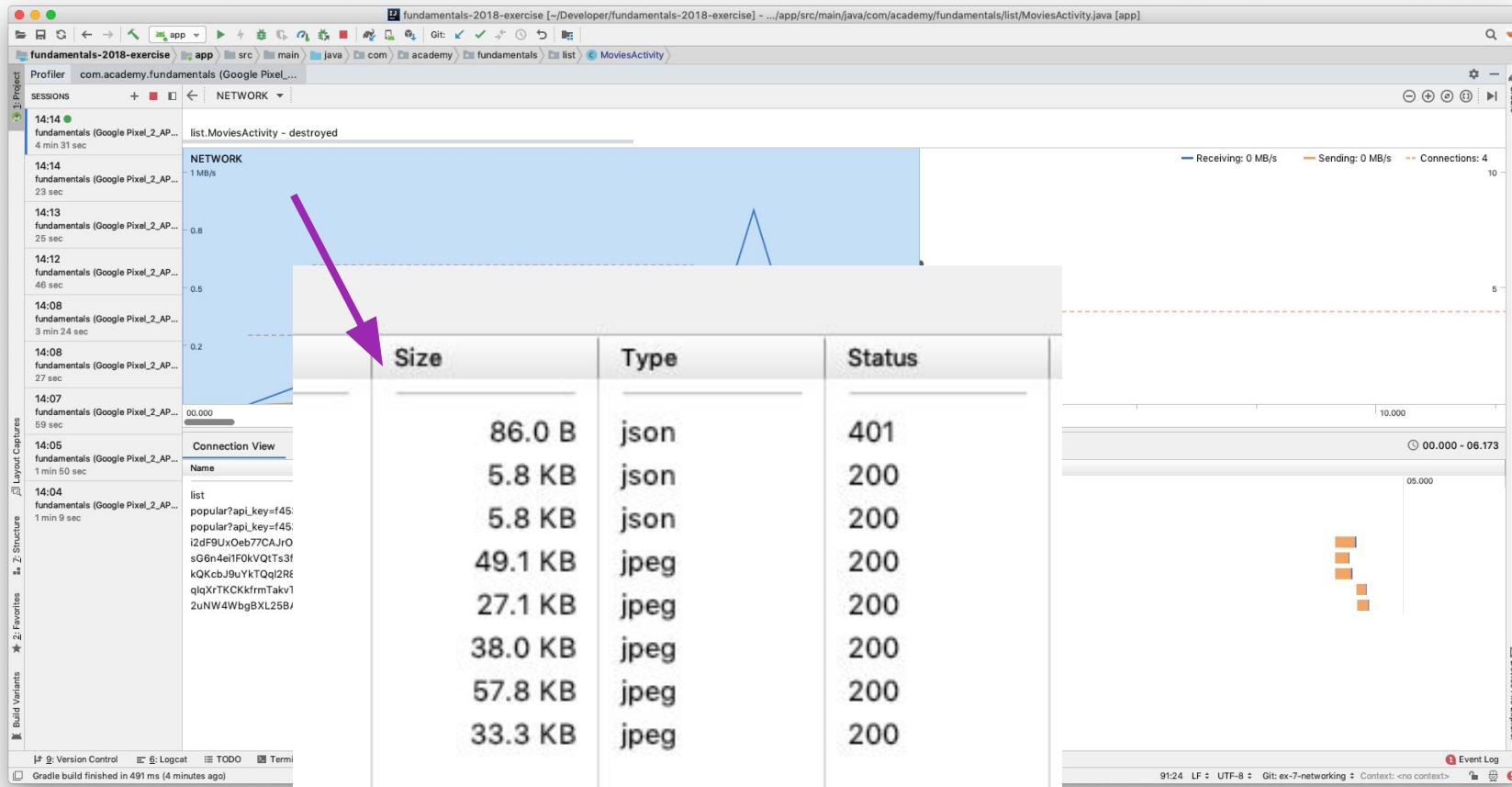
Network profiler



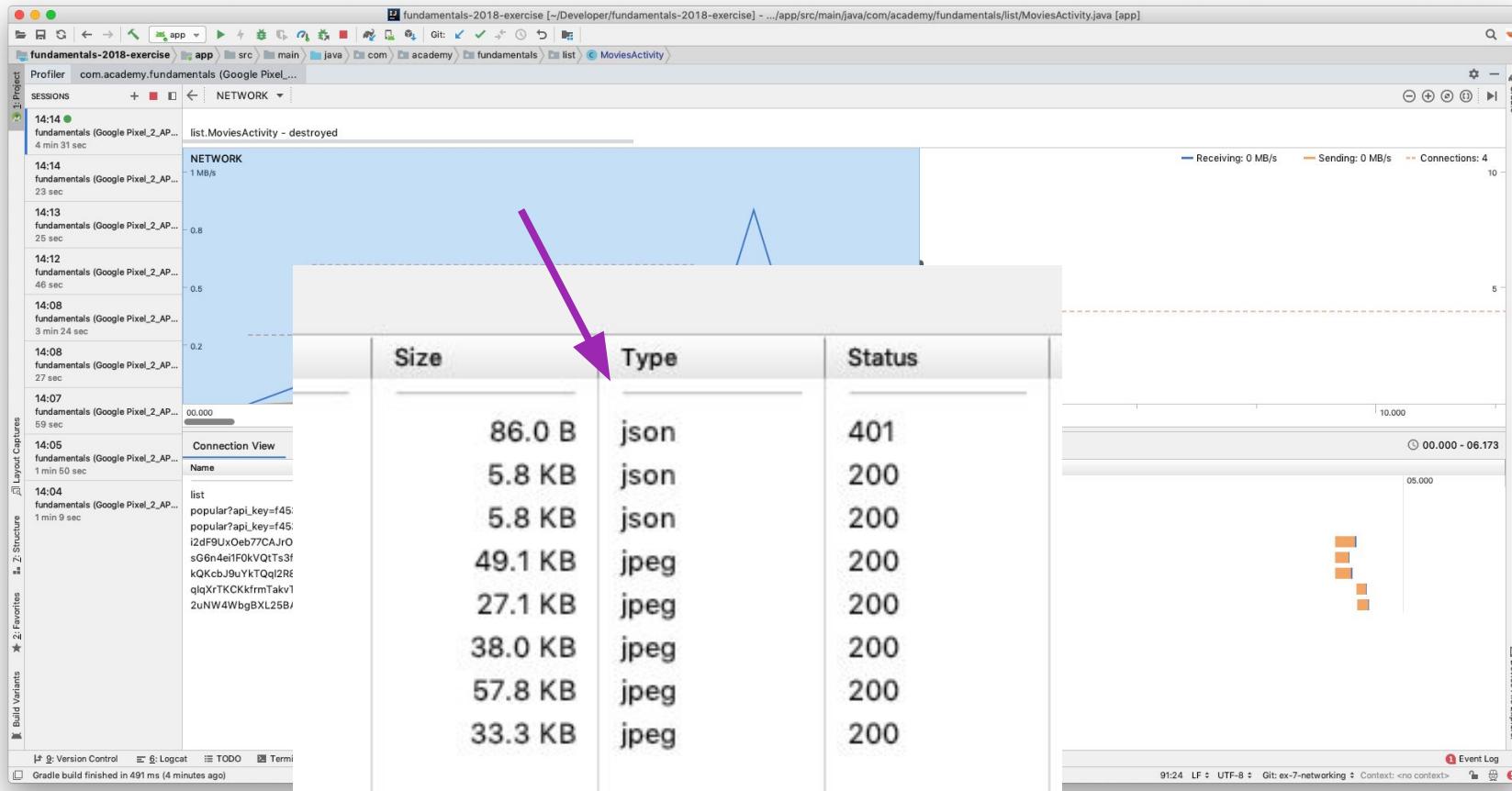
Network profiler



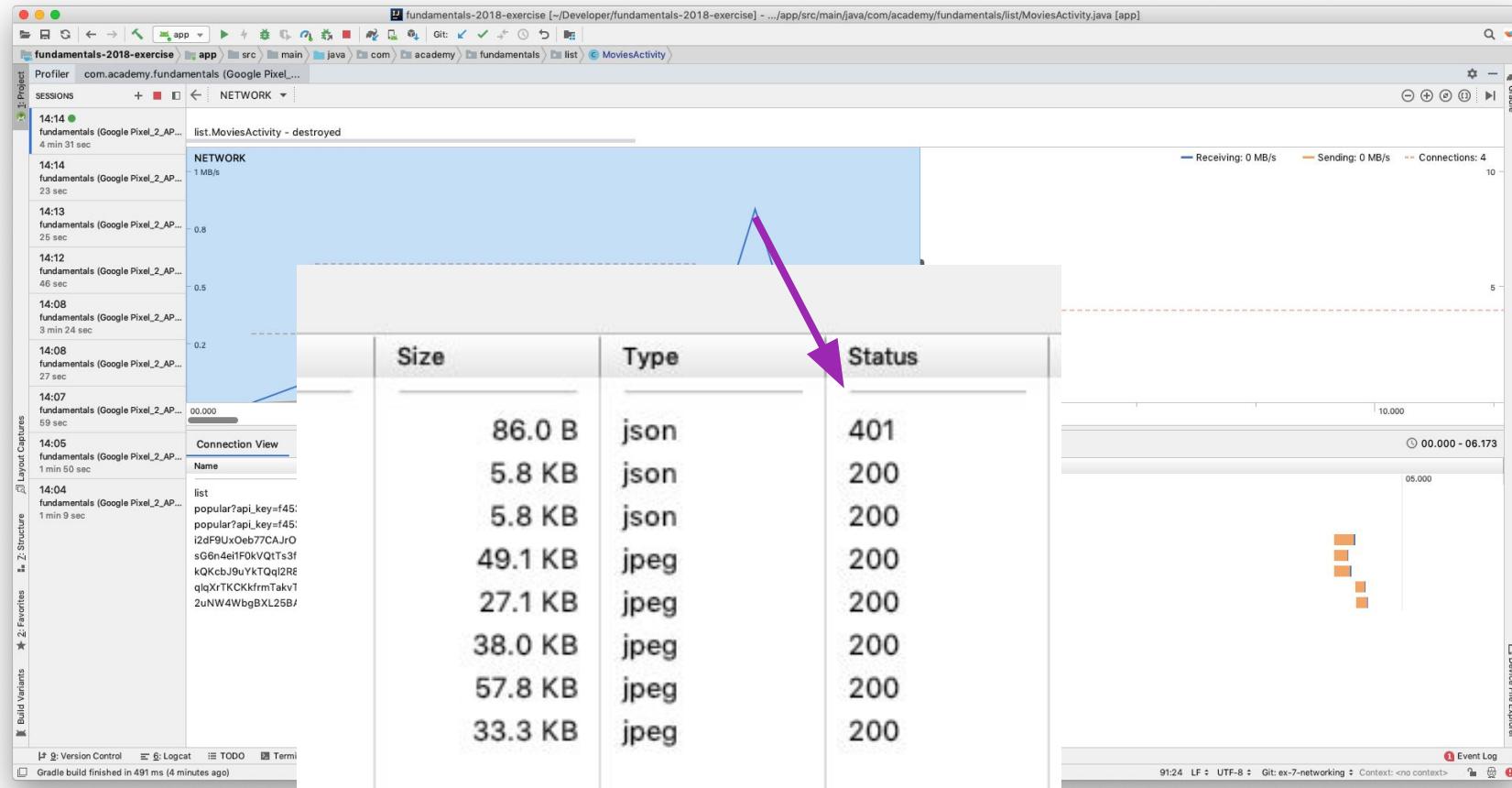
Network profiler



Network profiler



Network profiler



Network profiler

Sessions: 14:14 fundamentals (Google Pixel_2_AP... 2 min 54 sec)

NETWORK

Receiving: 0 MB/s Sending: 0 MB/s Connections: 4

list.MoviesActivi...

Headers

```
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: ETag, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, Retry-After
Cache-Control: public, max-age=21600
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 5902
Content-Type: application/json; charset=utf-8
Date: Thu, 20 Dec 2018 12:14:37 GMT
Server: openresty
Vary: Accept-Encoding
X-RateLimit-Limit: 40
X-RateLimit-Remaining: 39
X-RateLimit-Reset: 1545308087
```

Body (JSON)

```
{
  "page": 1,
  "total_results": 19835,
  "total_pages": 992,
  "results": [
    {
      "vote_count": 431,
      "id": 297802,
      "video": false,
      "vote_average": 6.9,
      "title": "Aquaman",
      "popularity": 488.843,
      "poster_path": "/i2dF9Ux0eb77CAJrOfljRpqJRF.jpg",
      "original_language": "en",
      "original_title": "Aquaman",
      "genre_ids": [
        28,
        14,
        878,
        12
      ],
      "backdrop_path": "/5A2bMllfJrAfX9hgAib0L2gCruF.jpg",
      "adult": false,
      "overview": "The film reveals the origin story of half-human, half-Atlantean"
    }
  ]
}
```

Event Log: Gradle build finished in 491 ms (3 minutes ago)

278

Network profiler

fundamentals-2018-exercise [~/Developer/fundamentals-2018-exercise] - .../app/src/main/java/com/academy/fundamentals/list/MoviesActivity.java [app]

Profiler com.academy.fundamentals (Google Pixel_...)

SESSIONS + ⌂ ⌂ NETWORK ⌂

1: Project

14:14 fundamentals (Google Pixel_2_AP... 2 min 34 sec

14:14 fundamentals (Google Pixel_2_AP... 23 sec

14:13 fundamentals (Google Pixel_2_AP... 25 sec

14:12 fundamentals (Google Pixel_2_AP... 46 sec

14:08 fundamentals (Google Pixel_2_AP... 3 min 24 sec

14:08 fundamentals (Google Pixel_2_AP... 27 sec

14:07 fundamentals (Google Pixel_2_AP... 59 sec

14:05 fundamentals (Google Pixel_2_AP... 1 min 50 sec

14:04 fundamentals (Google Pixel_2_AP... 1 min 9 sec

Layout Captures

2: Structure

2: Favorites

Build Variants

Version Control Logcat TODO Terminal Build Profiler Run

Gradle build finished in 491 ms (2 minutes ago)

Receiving: 0 MB/s Sending: 0 MB/s Connections: 4

list.MoviesActiv...

NETWORK 1 MB/s

0.8
0.5
0.2

00.000 05.000 10.000 15.000 20.000 25.000

Overview Response Request Call Stack

Request kQKcbJ9uYkTQqj2R8L4jTUz7I90.jpg

Method GET

Status 200

Dimension 342 x 513

Content type image/jpeg

Size 38.03 KB

Initiating thread Picasso-/p/w342/kQKcbJ9uYkTQqj2R8L4jTUz7I90.jpg

URL https://image.tmdb.org/t/p/w342/kQKcbJ9uYkTQqj2R8L4jTUz7I90.jpg

Timing Sent: 181 ms Received: 22 ms

Device File Explorer

Event Log 279

Name	Size	Type	Status	Time	Timeline
list	86.0 B	json	401	185 ms	00.000 05.000
popular?api_key=f453f49d458a7818...	5.8 KB	json	200	217 ms	00.000 05.000
popular?api_key=f453f49d458a7818...	5.8 KB	json	200	236 ms	00.000 05.000
i2df9UxOeb77CAJrOfj0RpqJRF.jpg	49.1 KB	jpeg	200	240 ms	00.000 05.000
sG6n4eIf0kVQtTs3Afjdghngpa.jpg	27.1 KB	jpeg	200	161 ms	00.000 05.000
kQKcbJ9uYkTQqj2R8L4jTUz7I90.jpg	38.0 KB	jpeg	200	203 ms	00.000 05.000
qlqXrTKCKfrmTakvTsJly7OWw.jpg	57.8 KB	jpeg	200	121 ms	00.000 05.000
2uNW4WbgBXL25BAbXGLnLqX71S...	33.3 KB	jpeg	200	130 ms	00.000 05.000

Questions?

Exercise

