# *Android Development*

ROSE HACK 2019

*Hello!*

# I am Emma Rivera

If you have any questions you can find me at any of the links below:

Facebook LinkedIn

1

# *Install Android Studio!*

https://developer.android.com/studio/

# *What are we doing?*

Movie application

Allows users to browse current movies and view their descriptions utilizing the movie database API.

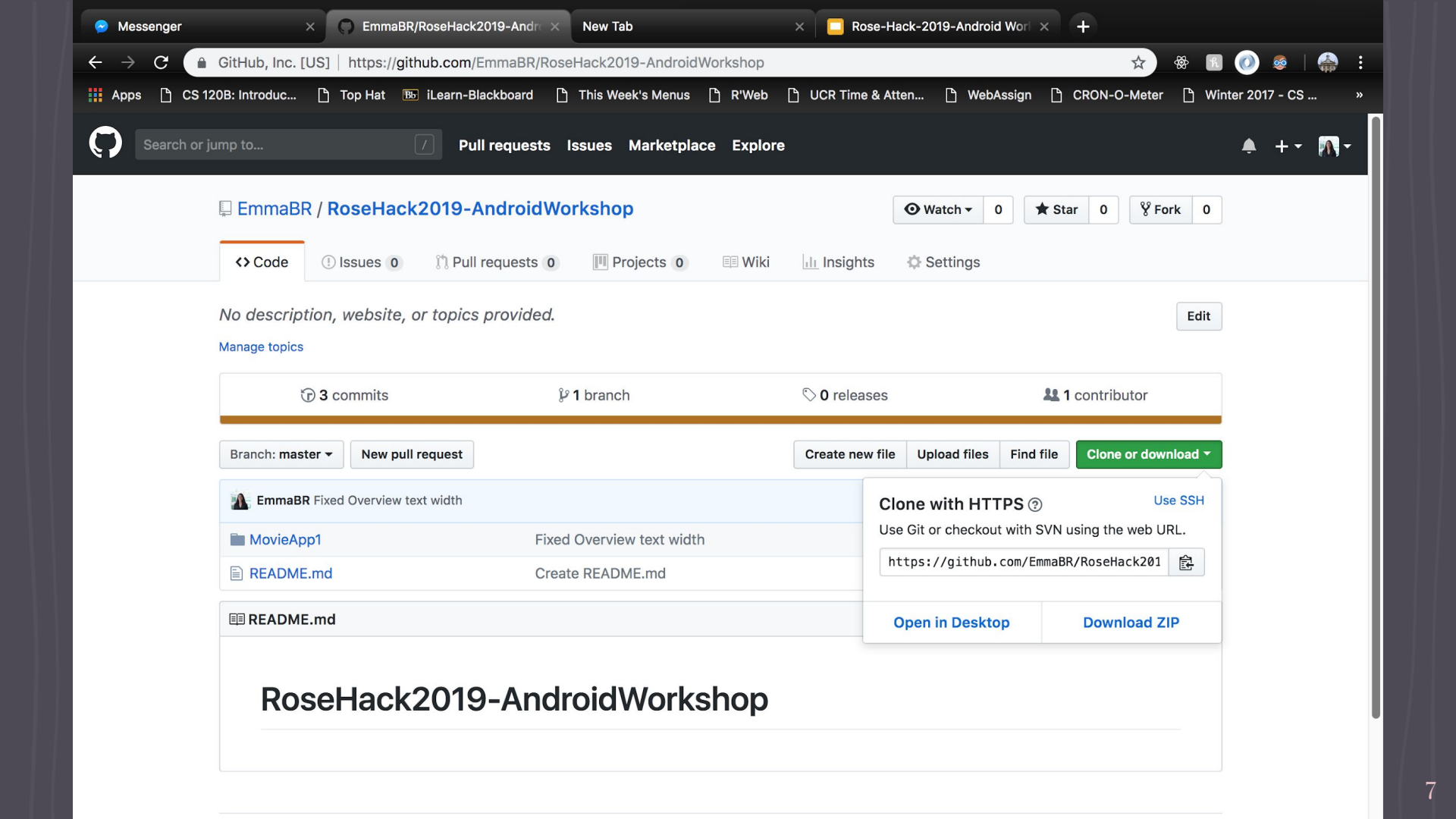
# *Setting up your API*

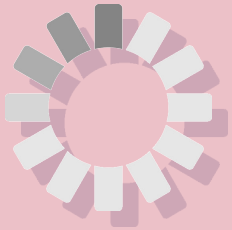
- Create an account at TheMovieDB.org
- Click on your profile (top right)
- In the drop down click “settings”
- In the far left column click “API”
- At the bottom of the page hit “click here” to generate a new API key
- Click “developer”
- Agree to terms and conditions
- Fill out remaining information*
- Documentation for the API

*mark as “mobile application” in the dropdown

# *Clone the Project*

- Go to: rosehack.com/androidworkshop
- Clone the Repo or download the zip file.
- Open Android Studio.
- Click open a project and navigate to StartMovieApp and open it.


Messenger
EmmaBR/RoseHack2019-Andr
New Tab
Rose-Hack-2019-Android Wor
GitHub, Inc. [US] | https://github.com/EmmaBR/RoseHack2019-AndroidWorkshop
Apps
CS 120B: Introduc...
Top Hat
iLearn-Blackboard
This Week's Menus
R'Web
UCR Time & Atten...
WebAssign
CRON-O-Meter
Winter 2017 - CS ...
Search or jump to...
Pull requests
Issues
Marketplace
Explore
EmmaBR / RoseHack2019-AndroidWorkshop
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
3 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
EmmaBR Fixed Overview text width
MovieApp1
Fixed Overview text width
README.md
Create README.md
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/EmmaBR/RoseHack201
Open in Desktop
Download ZIP
README.md
RoseHack2019-AndroidWorkshop



# *Wait...*

Your project is building!!

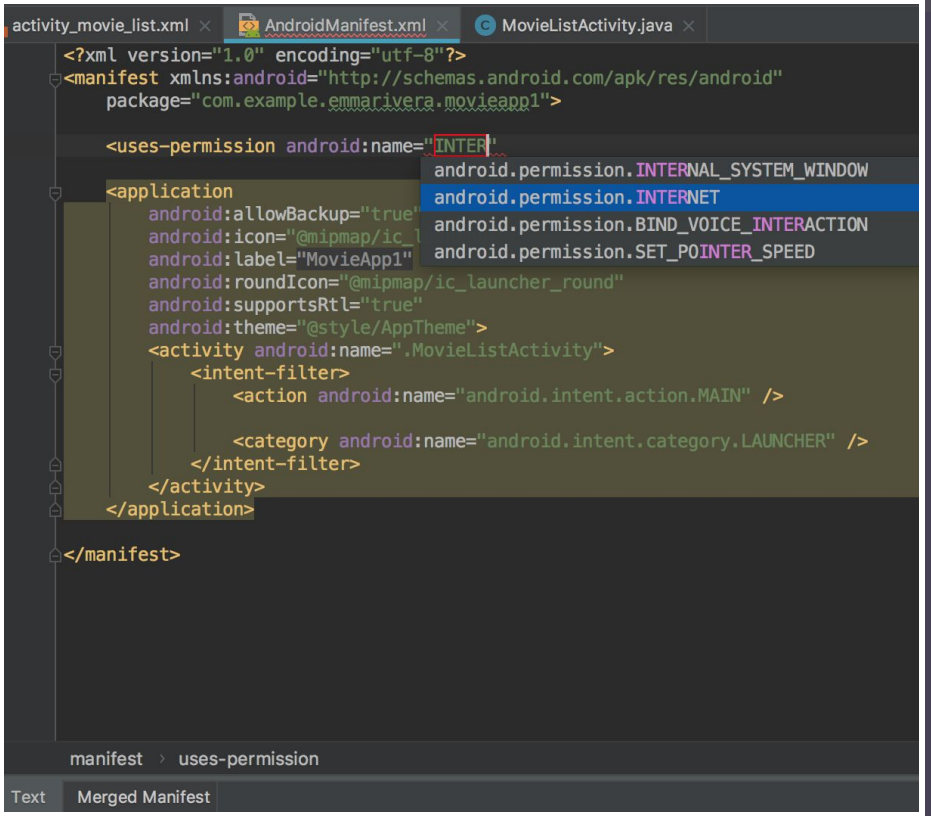
## *Before we Start...*

- Android Studio > Preferences > Editor > General > Auto Import :  Make sure all the boxes are checked.
- Build, Execution, Deployment > Instant Run : Disable Instant Run by unchecking the first box

# *Add Internet to Your Application*

- AndroidManifest.xml
- Outside of application tags type uses permission tags*
- Type internet* (all caps)

*will populate on it's own



# *Hiding API your key*

## Step 1

Under res find values.

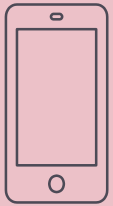
Right click on values. In the pop up select “new” then “values resource file”. Name it: secrets.xml.

## Step 2

In secrets.xml create a string with a name “api_key” and cut and paste the key as the value of the string. Now when referencing the string in the java file use getString(R.string.api_string).

## Step 3

Add this file (secrets.xml) to your gitignore.



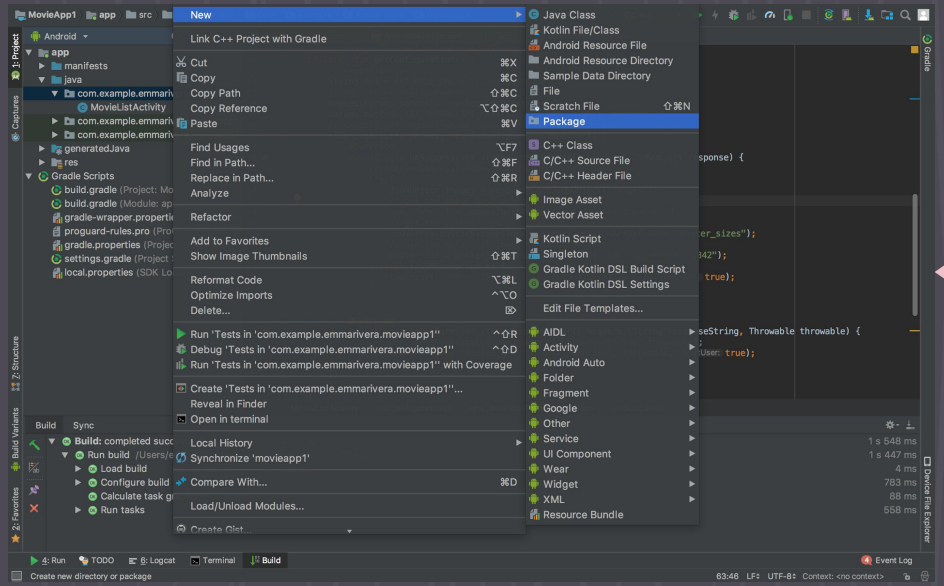
# *Emulator*

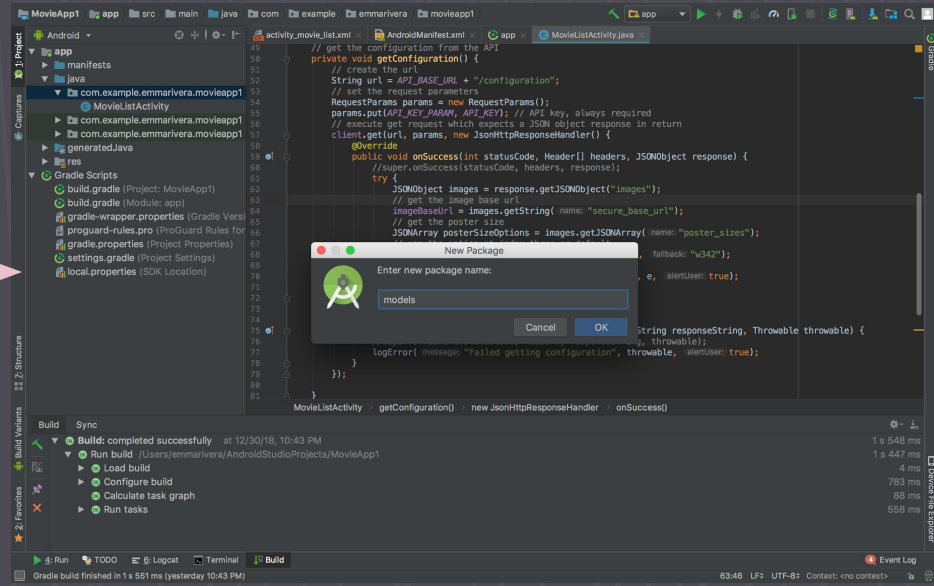
Run the emulator to test to make sure your project builds and works correctly!!

# *Let's Add Some Files!*

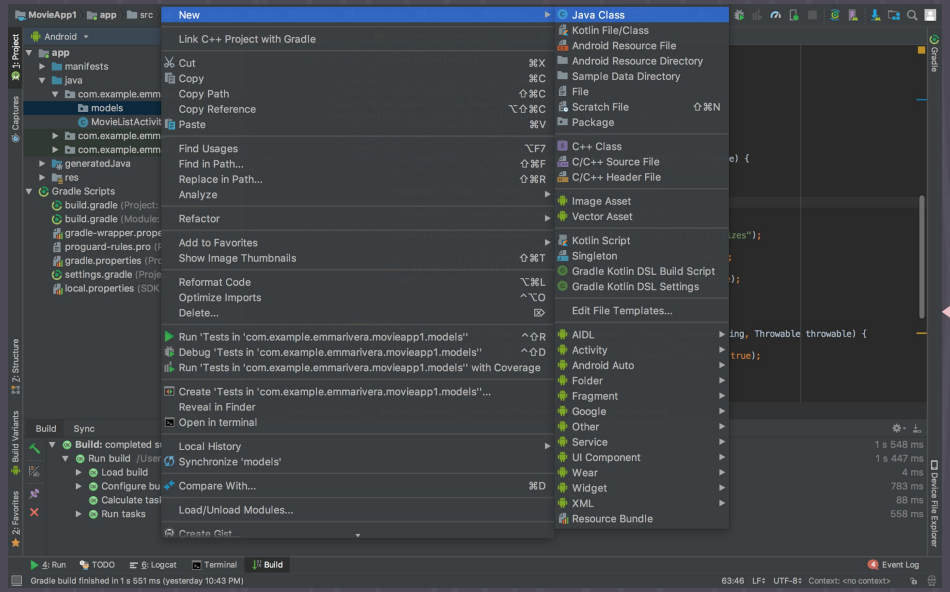
Creating a Model

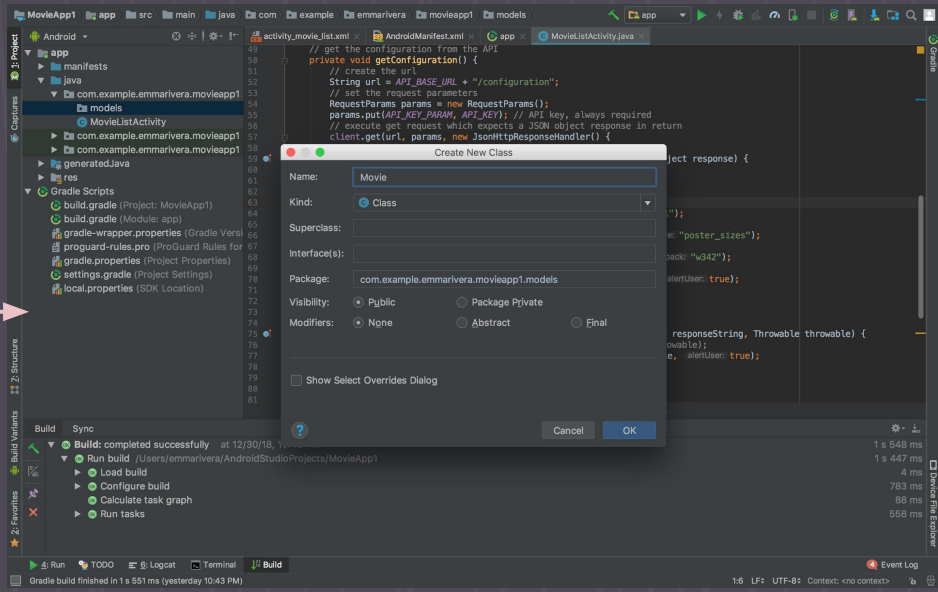
# Create the Model Package



# Create the Model Class



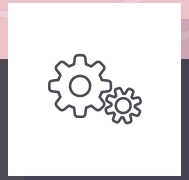



## *Setting Up the Movie Class to Obtain Data*

- Create variables for the data you want to receive
- Create a custom Movie constructor and initialize variables inside.
- Generate getters to allow the data to be accessed in the Activity class

```
public class Movie {

    // values from the API
    private String title;
    private String overview;
    private String posterPath; // only the path

    // initialize from JSON data
    public Movie(JSONObject object) throws JSONException {
        title = object.getString( name: "title"); // remember key's have t
        overview = object.getString( name: "overview");
        posterPath = object.getString( name: "poster_path");
    }

    /** NOTE: because these variables were declared private we
     *  need to create getters to expose this data to expose this data
     *  to the activity
     **/

    public String getTitle() {
        return title;
    }

    public String getOverview() {
        return overview;
    }

    public String getPosterPath() {
        return posterPath;
    }
}
```

Show movie JSON

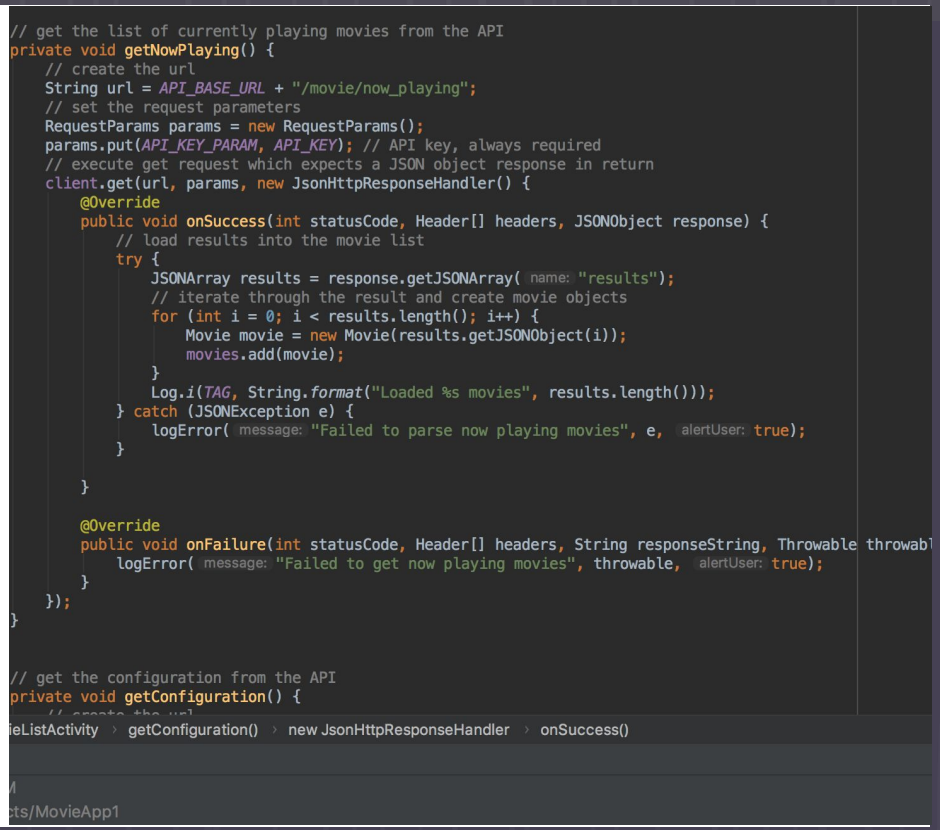


# *Accessing our data in the Activity Class*

Traversing the JSON

# *Getting Now Playing Movies*

- Create variables for the data you want to receive (ArrayList)
- Write the getNowPlaying function to retrieve the data
- Call get now playing function in the onSuccess of getConfiguration(); (must execute after get configuration)
- Call getConfiguration(); in the onCreate

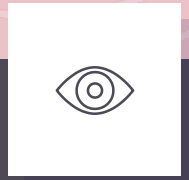


```
// get the list of currently playing movies from the API
private void getNowPlaying() {
    // create the url
    String url = API_BASE_URL + "/movie/now_playing";
    // set the request parameters
    RequestParams params = new RequestParams();
    params.put(API_KEY_PARAM, API_KEY); // API key, always required
    // execute get request which expects a JSON object response in return
    client.get(url, params, new JsonHttpResponseHandler() {
        @Override
        public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
            // load results into the movie list
            try {
                JSONArray results = response.getJSONArray( name: "results");
                // iterate through the result and create movie objects
                for (int i = 0; i < results.length(); i++) {
                    Movie movie = new Movie(results.getJSONObject(i));
                    movies.add(movie);
                }
                Log.i(TAG, String.format("Loaded %s movies", results.length()));
            } catch (JSONException e) {
                logError( message: "Failed to parse now playing movies", e,  alertUser: true);
            }

        }

        @Override
        public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwabl
            logError( message: "Failed to get now playing movies", throwable,  alertUser: true);
        }
    });
}


// get the configuration from the API
private void getConfiguration() {
    // create the url
```

ieListActivity › getConfiguration() › new JsonHttpResponseHandler › onSuccess()

cts/MovieApp1

```java
// get the list of currently playing movies from the API
private void getNowPlaying() {
    // create the url
    String url = API_BASE_URL + "/movie/now_playing";
    // set the request parameters
    RequestParams params = new RequestParams();
    params.put(API_KEY_PARAM, API_KEY); // API key, always required
    // execute get request which expects a JSON object response in return
    client.get(url, params, new JsonHttpResponseHandler() {
        @Override
        public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
            // load results into the movie list
            try {
                JSONArray results = response.getJSONArray( name: "results");
                // iterate through the result and create movie objects
                for (int i = 0; i < results.length(); i++) {
                    Movie movie = new Movie(results.getJSONObject(i));
                    movies.add(movie);
                }
                Log.i(TAG, String.format("Loaded %s movies", results.length()));
            } catch (JSONException e) {
                logError( message: "Failed to parse now playing movies", e,  alertUser: true);
            }

        }

        @Override
        public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwabl
            logError( message: "Failed to get now playing movies", throwable,  alertUser: true);
        }
```



# *Making the data visible to the user*

XML files and UI/UX

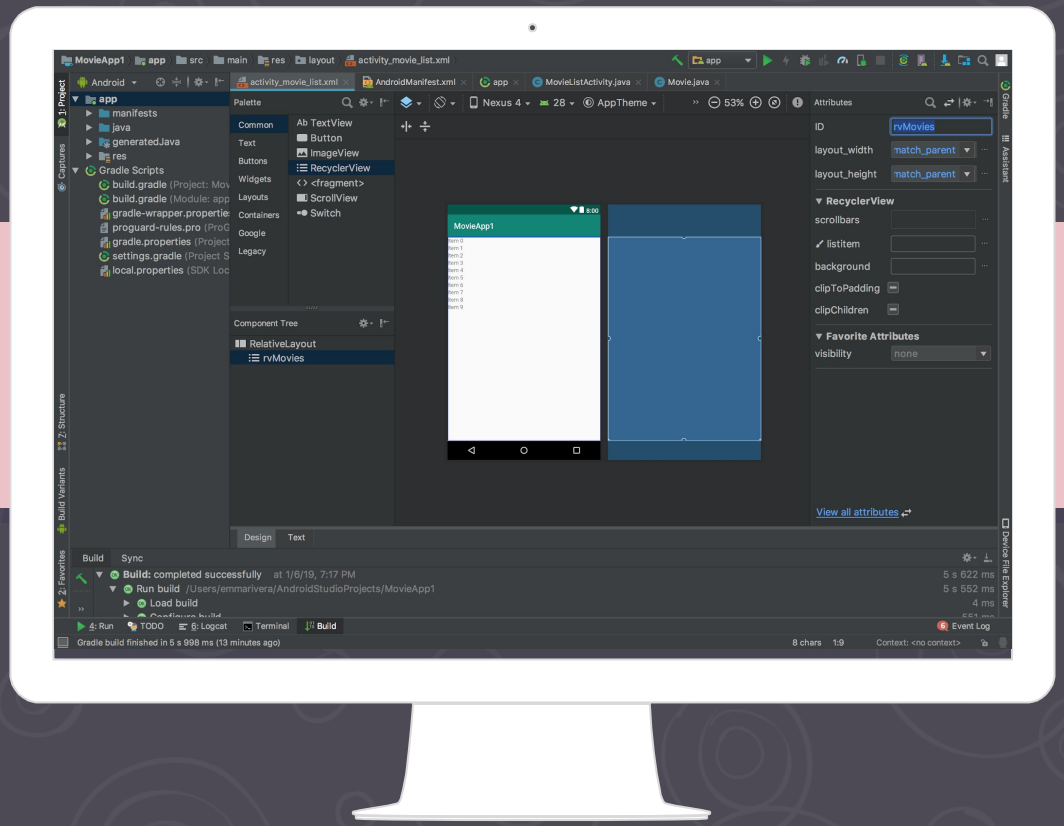
# *Adding a Recyclerview*

Delete “Hello World” textView

Click recyclerview and add dependency

Drag and drop recycler to top left (it will expand)

Set ID to “rvMovies”



# *Adding Items to the Recyclerview*

## Create Resource File

Res -> layout

Right Click Layout

New -> Layout Resource File

## Add Textviews to Items

Drag and drop the text view to the right of the image.

Change ID to “tvTitle”

Change text appearance to large

Drag to align with right

## Name the File

Name the file “item_movies”

Make sure to set the Root Element to Relativelayout

Click “ok”

## Add another Text View

This one will be for the overview/synopsis of the movie

Make sure to give it a unique ID

## Add Images to Items

Drag in an ImageView

Choose the image to be a black background (select android > black)

Change the width to 90dp and height to 135dp

Set ID to ivPosterImage

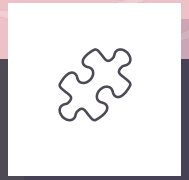
## We’re Done!

## *Few more things...*

- Change the layout height to wrap content
- Add a margin between items (5dp)
- Add a margin right to the poster image ImageView (5dp)
- Remove the line that aligns the overview to the end of the poster image
- Make the overview textView height wrap content

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="5dp"
    android:layout_width="match_parent" android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/ivPosterImage"
        android:layout_width="90dp"
        android:layout_height="135dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        app:srcCompat="@android:color/black"
        android:layout_marginRight="5dp"/>

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="294dp"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="0dp"
        android:layout_toRightOf="@+id/ivPosterImage"
        android:text="Movie Title "
        android:textAppearance="@android:style/TextAppearance.Large" />

    <TextView
        android:id="@+id/tvOverview"
        android:layout_width="294dp"
        android:layout_height="105dp"
        android:layout_below="@+id/tvTitle"
        android:layout_toEndOf="@+id/ivPosterImage"
        android:text="TextView" />
</RelativeLayout>
```

RelativeLayout › TextView



# *Creating the Adapter*

Connects the xml to java file.

# *Setting up the Adapter*

Create MovieAdapter.java

Create an array of movies.

Auto generate constructor.

Initialize with list by generating the constructor.

Create a view holder class as a static inner class

public static ViewHolder extends RecyclerView.ViewHolder { }

Click on red light bulb to solve issue.

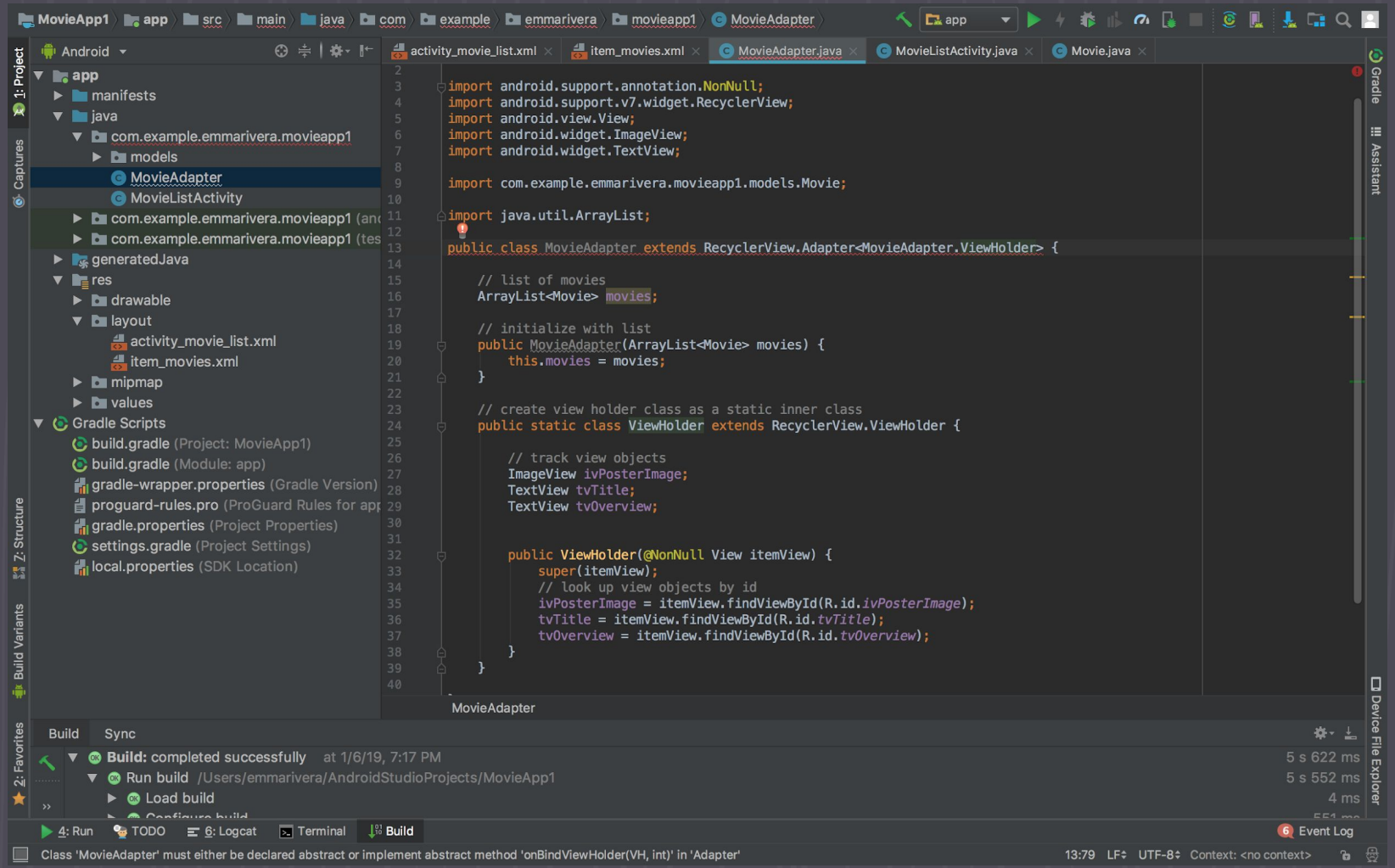
Add variables to track view objects.

Look up the items by id in public ViewHolder(View itemView) {

// here

}

Extend MovieAdapter with RecyclerView.Adapter<MovieAdapter.ViewHolder>


MovieApp1 app src main java com example emmarivera movieapp1 MovieAdapter
app
Android
activity_movie_list.xml
item_movies.xml
MovieAdapter.java
MovieListActivity.java
Movie.java
app
manifests
java
com.example.emmarivera.movieapp1
models
MovieAdapter
MovieListActivity
com.example.emmarivera.movieapp1 (and
com.example.emmarivera.movieapp1 (tes
generatedJava
res
drawable
layout
activity_movie_list.xml
item_movies.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MovieApp1)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.emmarivera.movieapp1.models.Movie;
import java.util.ArrayList;
public class MovieAdapter extends RecyclerView.Adapter<MovieAdapter.ViewHolder> {
// list of movies
ArrayList<Movie> movies;
// initialize with list
public MovieAdapter(ArrayList<Movie> movies) {
this.movies = movies;
}
// create view holder class as a static inner class
public static class ViewHolder extends RecyclerView.ViewHolder {
// track view objects
ImageView ivPosterImage;
TextView tvTitle;
TextView tvOverview;
public ViewHolder(@NonNull View itemView) {
super(itemView);
// look up view objects by id
ivPosterImage = itemView.findViewById(R.id.ivPosterImage);
tvTitle = itemView.findViewById(R.id.tvTitle);
tvOverview = itemView.findViewById(R.id.tvOverview);
}
}
MovieAdapter
Build
Sync
Build: completed successfully at 1/6/19, 7:17 PM
5 s 622 ms
Run build /Users/emmarivera/AndroidStudioProjects/MovieApp1
5 s 552 ms
Load build
4 ms
4: Run
TODO
6: Logcat
Terminal
Build
6 Event Log
Class 'MovieAdapter' must either be declared abstract or implement abstract method 'onBindViewHolder(VH, int)' in 'Adapter'
13:79
LF
UTF-8
Context: <no context>
1: Project
Captures
Z: Structure
Build Variants
2: Favorites
Gradle
Assistant
Device File Explorer

# *Resolve the Error*

**Generate Methods**

Click the Red Light bulb.

Implement Methods.

Make sure “Copy JavaDoc” is unchecked.

Click ok. (There should be three)

**Implement Functions**

getItemCount: return the size of the array.

onCreateViewHolder:

Get context & create inflater. Create the view using item_movies layout. Return the item wrapped by a ViewHolder*

**Implement Functions cont.**

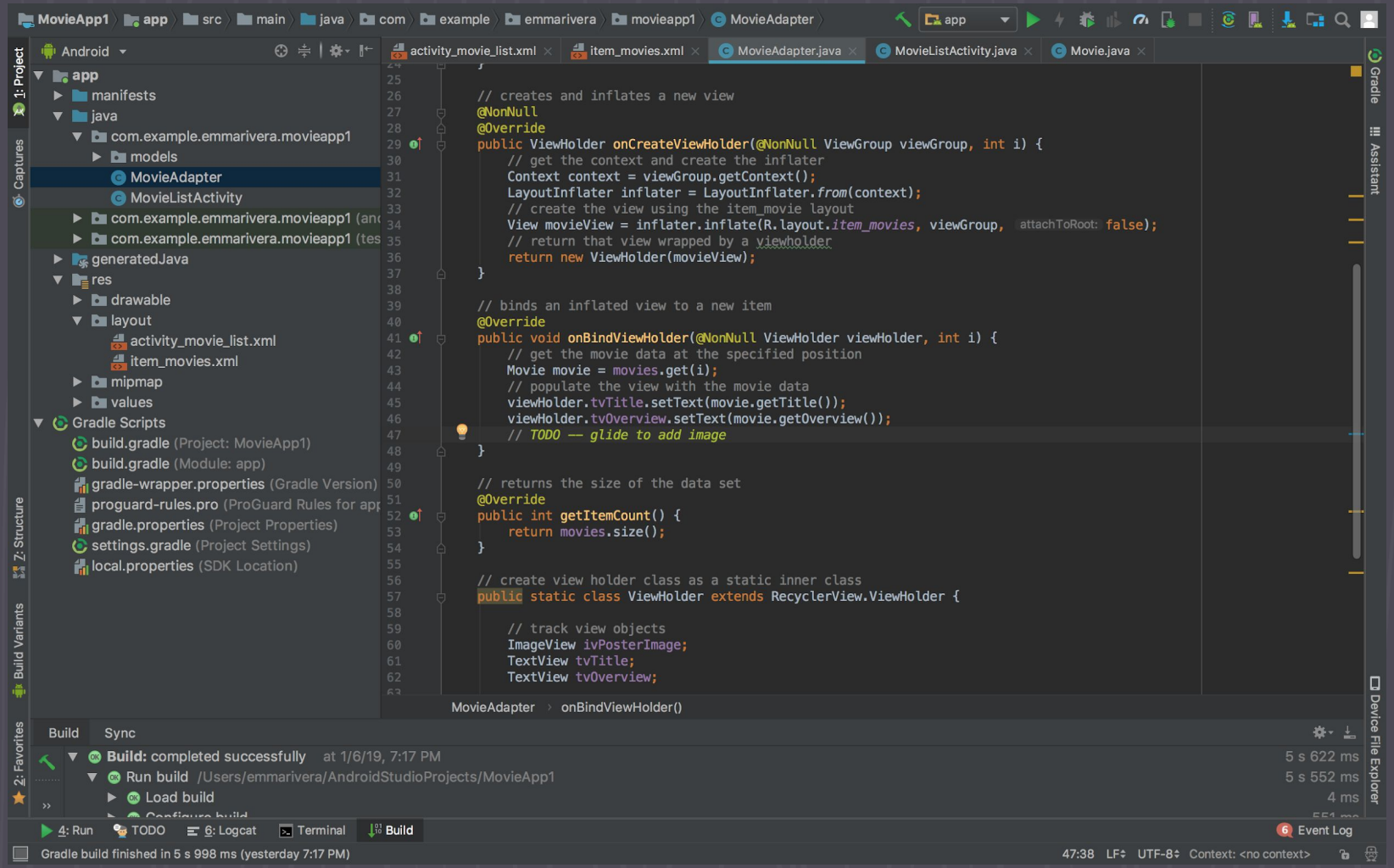
onBindViewHolder:

Get the data at the specified position

Load the viewHolder with the data at that point.

// we will come back to adding images later

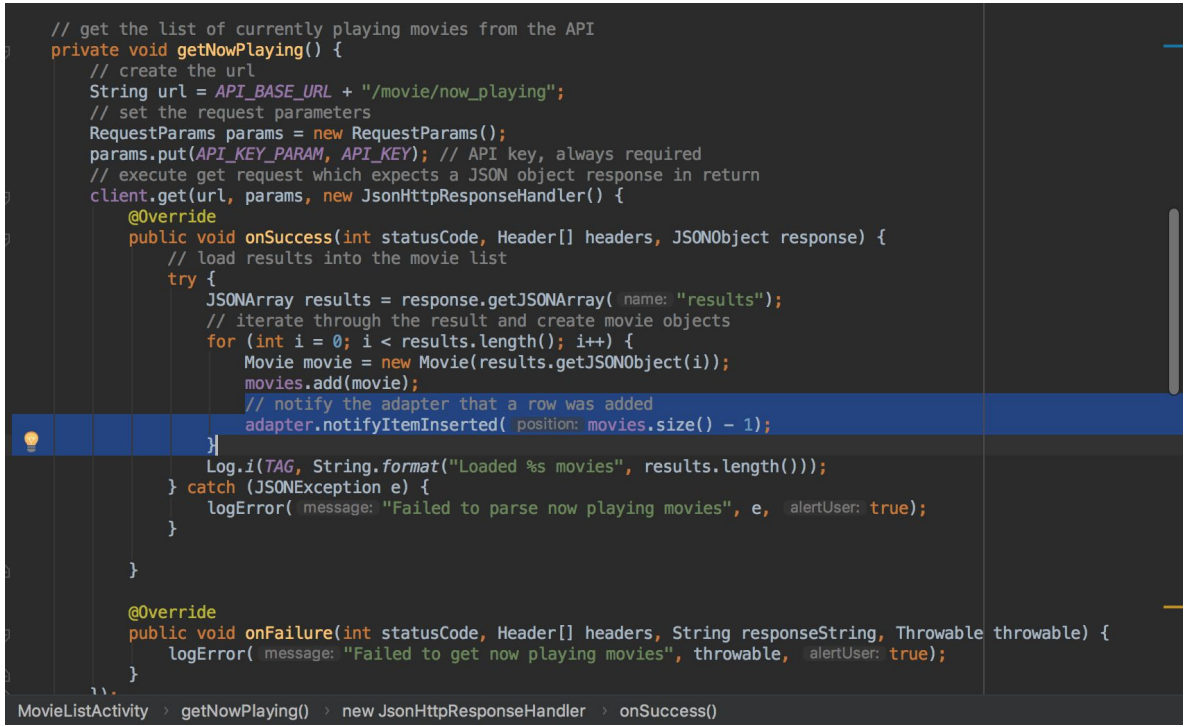
*don’t forget to add “new” before


MovieApp1 app src main java com example emmarivera movieapp1 MovieAdapter
app
Android
app
manifests
java
com.example.emmarivera.movieapp1
models
MovieAdapter
MovieListActivity
com.example.emmarivera.movieapp1 (and
com.example.emmarivera.movieapp1 (tes
generatedJava
res
drawable
layout
activity_movie_list.xml
item_movies.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MovieApp1)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_movie_list.xml
item_movies.xml
MovieAdapter.java
MovieListActivity.java
Movie.java
// creates and inflates a new view
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
// get the context and create the inflater
Context context = viewGroup.getContext();
LayoutInflater inflater = LayoutInflater.from(context);
// create the view using the item_movie layout
View movieView = inflater.inflate(R.layout.item_movies, viewGroup, attachToRoot: false);
// return that view wrapped by a viewholder
return new ViewHolder(movieView);
}
// binds an inflated view to a new item
@Override
public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
// get the movie data at the specified position
Movie movie = movies.get(i);
// populate the view with the movie data
viewHolder.tvTitle.setText(movie.getTitle());
viewHolder.tvOverview.setText(movie.getOverview());
// TODO — glide to add image
}
// returns the size of the data set
@Override
public int getItemCount() {
return movies.size();
}
// create view holder class as a static inner class
public static class ViewHolder extends RecyclerView.ViewHolder {
// track view objects
ImageView ivPosterImage;
TextView tvTitle;
TextView tvOverview;
MovieAdapter › onBindViewHolder()
Build
Sync
Build: completed successfully at 1/6/19, 7:17 PM
5 s 622 ms
Run build /Users/emmarivera/AndroidStudioProjects/MovieApp1
5 s 552 ms
Load build
4 ms
4: Run
TODO
6: Logcat
Terminal
Build
6 Event Log
Gradle build finished in 5 s 998 ms (yesterday 7:17 PM)
47:38
LF
UTF-8
Context: <no context>
1: Project
Captures
2: Favorites
Build Variants
7: Structure
Gradle
Assistant
Device File Explorer

## *Connect Recycler in MovieListActivity.java*

- Create the RecyclerView and MovieAdapter variables.
- Instantiate the variables after the initialization of the movie array.

```
// instantiate fields
AsyncHttpClient client;
// base url for loading images
String imageBaseUrl;
// the poster size to use when fetching the images
String posterSize;
// list of currently playing movies
ArrayList<Movie> movies;
// the recycler view
RecyclerView rvMovies;
// the adapter wired to the recycler view
MovieAdapter adapter;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_movie_list);

    //initialize the client
    client = new AsyncHttpClient();
    // initialize the list of movies
    movies = new ArrayList<>();
    // initialize the adapter -- movies adapter cannot be reinitialized after this po
    adapter = new MovieAdapter(movies);

    // resolve the recycler view and connect layout manager
    rvMovies = findViewById(R.id.rvMovies);
    rvMovies.setLayoutManager(new LinearLayoutManager( context: this));
    rvMovies.setAdapter(adapter);


    // get configuration on creation
    getConfiguration();
}
```




Finally Notify the Adapter that the Dataset Changed

# *Adding Images Using Glide*

Glide

## *Create New Model Class Called "Config"*

- Cut the imageBaseUrl and posterSize out of MovieListActivity.java and past into Config.java.
- Create a constructor
- Cut and paste the logic for parsing the data into the constructor of Config.java
- Solve error w/exception

```java
package com.example.emmarivera.movieapp1.models;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Config {

    // base url for loading images
    String imageBaseUrl;
    // the poster size to use when fetching the images
    String posterSize;

    public Config(JSONObject object) throws JSONException {
        JSONObject images = object.getJSONObject("images");
        // get the image base url
        imageBaseUrl = images.getString( name: "secure_base_url");
        // get the poster size
        JSONArray posterSizeOptions = images.getJSONArray( name: "poster_sizes");
        // use the option at index three or default
        posterSize = posterSizeOptions.optString( index: 3,  fallback: "w342");
    }

}
```

```java
package com.example.emmarivera.movieapp1.models;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Config {

    // base url for loading images
    String imageBaseUrl;
    // the poster size to use when fetching the images
    String posterSize;

    public Config(JSONObject object) throws JSONException {
        JSONObject images = object.getJSONObject("images");
        // get the image base url
        imageBaseUrl = images.getString( name: "secure_base_url");
        // get the poster size
        JSONArray posterSizeOptions = images.getJSONArray( name: "poster_sizes");
        // use the option at index three or default
        posterSize = posterSizeOptions.optString( index: 3,  fallback: "w342");
    }

    public String getImageBaseUrl() {
        return imageBaseUrl;
    }

    public String getPosterSize() {
        return posterSize;
    }
}
```

# Generate Getters

```
import org.json.JSONObject;

public class Config {

    // base url for loading images
    String imageBaseUrl;
    // the poster size to use when fetching the images
    String posterSize;

    public Config(JSONObject object) throws JSONException {
        JSONObject images = object.getJSONObject("images");
        // get the image base url
        imageBaseUrl = images.getString( name: "secure_base_url");
        // get the poster size
        JSONArray posterSizeOptions = images.getJSONArray( name: "poster_sizes");
        // use the option at index three or default
        posterSize = posterSizeOptions.optString( index: 3,  fallback: "w342");
    }

    // helper function to create urls
    public String getImageUrl(String size, String path) {
        return String.format("%s%s%s", imageBaseUrl, size, path); // concatenate all three urls
    }

    public String getImageBaseUrl() {
        return imageBaseUrl;
    }

    public String getPosterSize() {
        return posterSize;
    }
}
```

Method 'getImageBaseUrl()' is never used

# Create Helper Function for Creating Urls

# *Fixing What we Broke*

- Go to MovieListActivity.java and create a Config variable.
- Instantiate config in the try {} of getConfiguration (config = new Config(response))
- Fix Log statement by using the getters.

```
try {
    config = new Config(response);
    Log.i(TAG, String.format("Loaded configuration with imageBaseUrl %s and posterSize %s",
            config.getImageBaseUrl(),
            config.getPosterSize()));
    // pass config to the adapter
```

- In MovieAdapter.java add the config
- Generate config setter

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

    // list of movies
    ArrayList<Movie> movies;
    // config needed for image urls
    Config config;

    // initialize with list
    public MovieAdapter(ArrayList<Movie> movies) {
        this.movies = movies;
    }

    public void setConfig(Config config) {
        this.config = config;
    }
```

# *Fixing What we Broke cont....*

- Go back to MovieListActivity.java
- Pass the config to the adapter

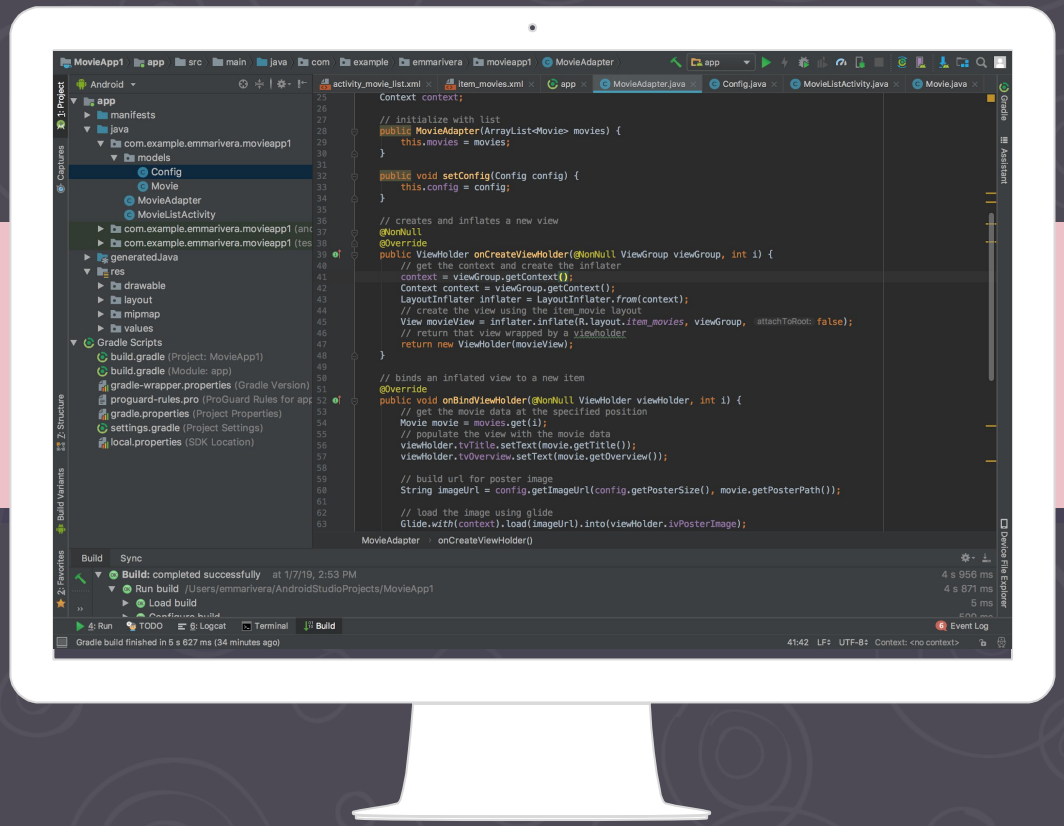
```
try {
    config = new Config(response);
    Log.i(TAG, String.format("Loaded configuration with imageBaseUrl %s and posterSize %s",
            config.getImageBaseUrl(),
            config.getPosterSize()));
    // pass config to the adapter
    adapter.setConfig(config);
    // get the now playing movies
    getNowPlaying();
} catch (JSONException e) {
    logError( message: "Failed Parsing configuration", e,  alertUser: true);
}
```

## *Movie Adapter TODO*

Create Context variable.

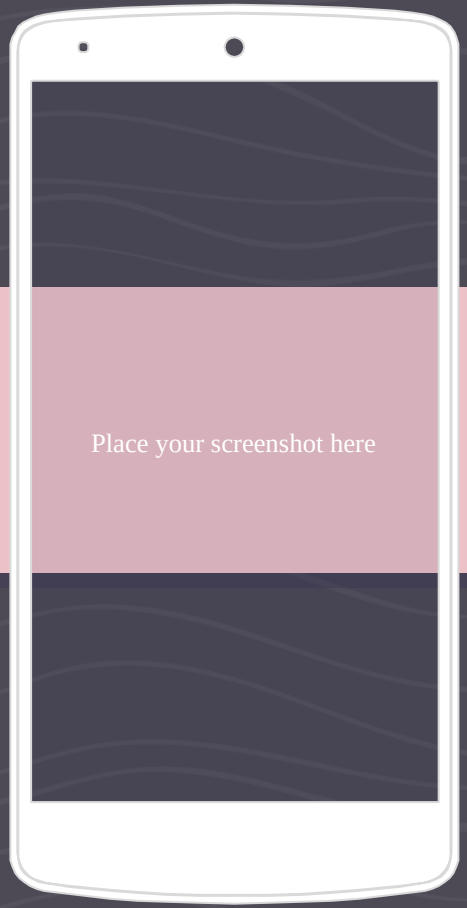
Build URL for poster image

Load image with Glide



## *And We're Done!!*

Your application should look something like this.



*Thanks!*

# Any questions?

You can find me at Facebook LinkedIn

**https://github.com/EmmaBR/flicks_movie_app**

# *The End!!*

Remaining slides are those that I omitted from the workshop due to lack of time but feel free to look over if you wish :)

# *Some Terms to Note*

**API -** Application Programming Interface

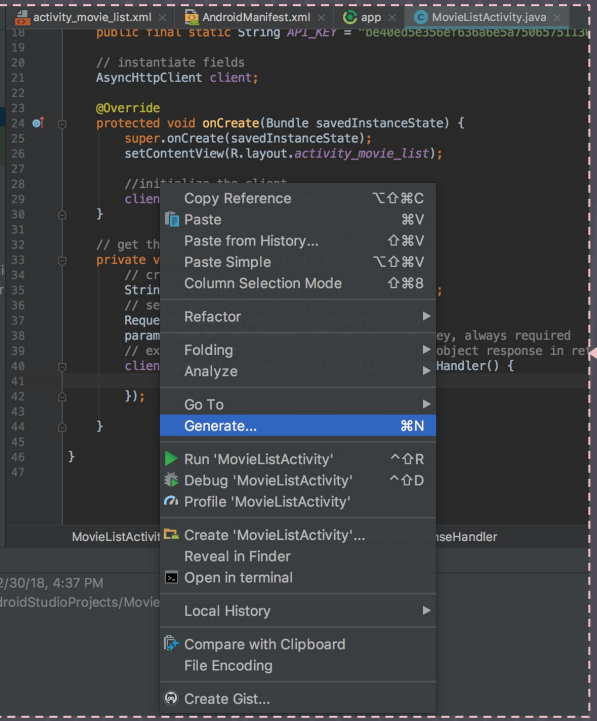
**Client** - the software that consumes or uses the API

**JSON**- JavaScript Object Notation

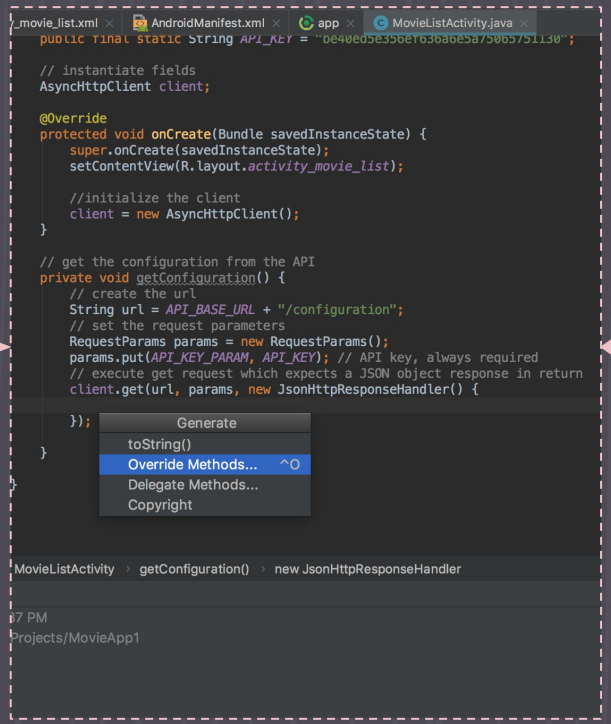
**Upper Camel Case** - naming convention where the first letter in every word is capitalized (eg: MovieListActivity)

**Async** - the response is not processed synchronously or at the same time as the request.

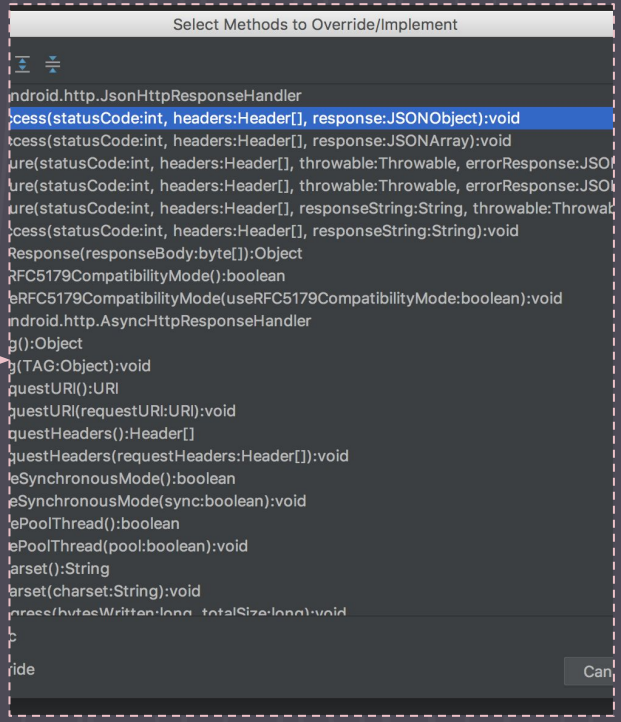
# *Generating Override Methods*




Right Click -> Generate

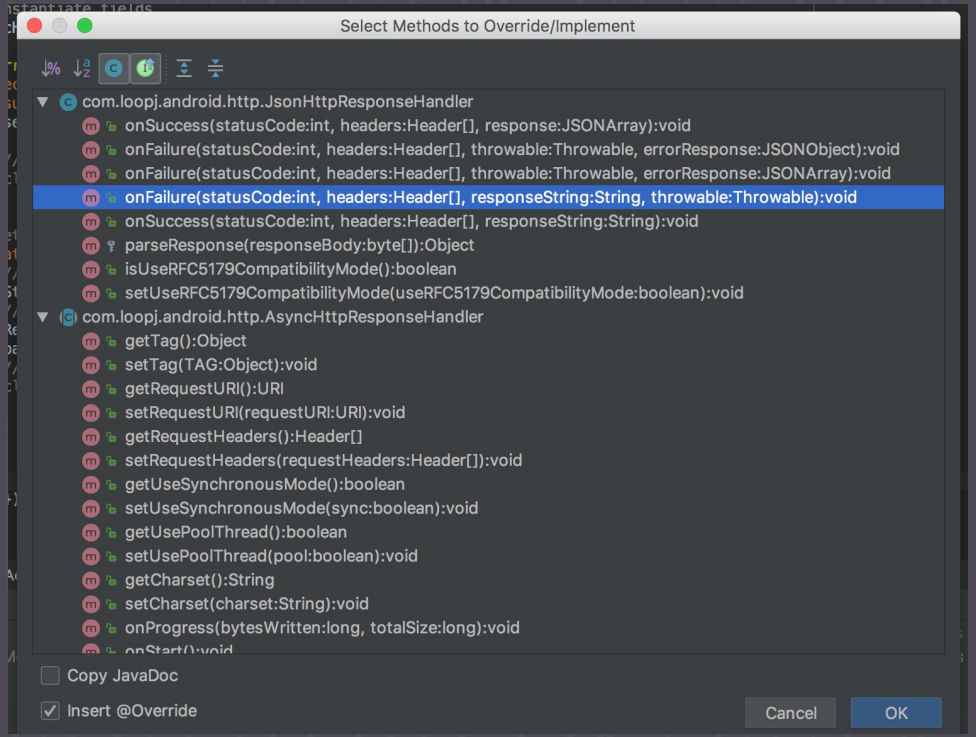



Override Methods



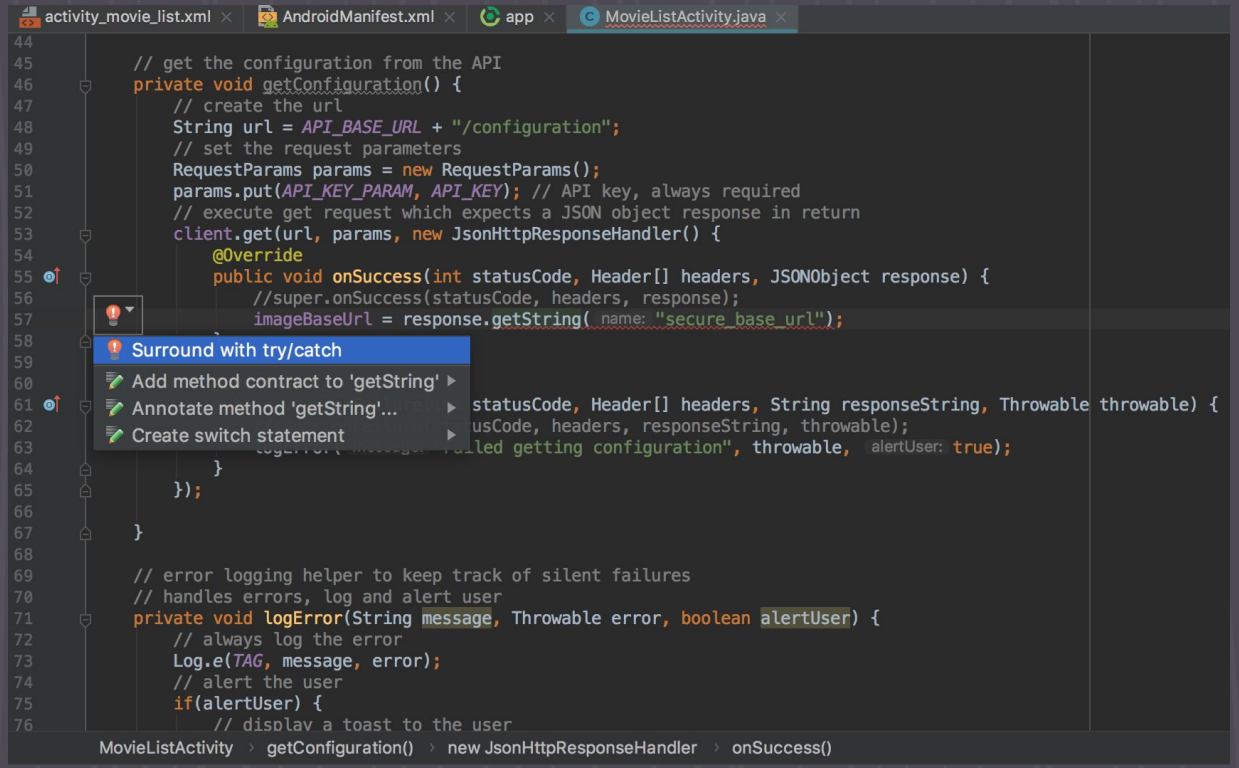

Override: onSuccess

# *Generating Override Methods cont...*




**Override: onFailure**

# *Handling Errors*



## *Getting Data from the API*

- In the onSuccess method add code to access the data in the API
- Make sure to name the key's exactly as stated in the API docs
- Travers data going key by key to get to the data you want

```
get the configuration from the API
vate void getConfiguration() {
// create the url
String url = API_BASE_URL + "/configuration";
// set the request parameters
RequestParams params = new RequestParams();
params.put(API_KEY_PARAM, API_KEY); // API key, always required
// execute get request which expects a JSON object response in return
client.get(url, params, new JsonHttpResponseHandler() {
    @Override
    public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
        //super.onSuccess(statusCode, headers, response);
        try {
            JSONObject images = response.getJSONObject("images");
            // get the image base url
            imageBaseUrl = images.getString( name: "secure_base_url");
            // get the poster size
            JSONArray posterSizeOptions = images.getJSONArray( name: "poster_sizes")
            // use the option at index three or default
            posterSize = posterSizeOptions.optString( index: 3,  fallback: "w342");
        } catch (JSONException e) {
            logError( message: "Failed Parsing configuration", e,  alertUser: true);
        }
    }

    @Override
    public void onFailure(int statusCode, Header[] headers, String responseString,
        //super.onFailure(statusCode, headers, responseString, throwable);
        logError( message: "Failed getting configuration", throwable,  alertUser: true)
    }
tActivity › getConfiguration() › new JsonHttpResponseHandler › onSuccess()
```