

# Intro to Android Development

INSERT DATE HERE  
[acmucr.org/checkin](http://acmucr.org/checkin)



Hello!  
I'm Emma.

Contact Me:  
[Facebook](#) [LinkedIn](#)

1

# Install Android Studio!

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

# What are we doing?

• • •

## Movie application

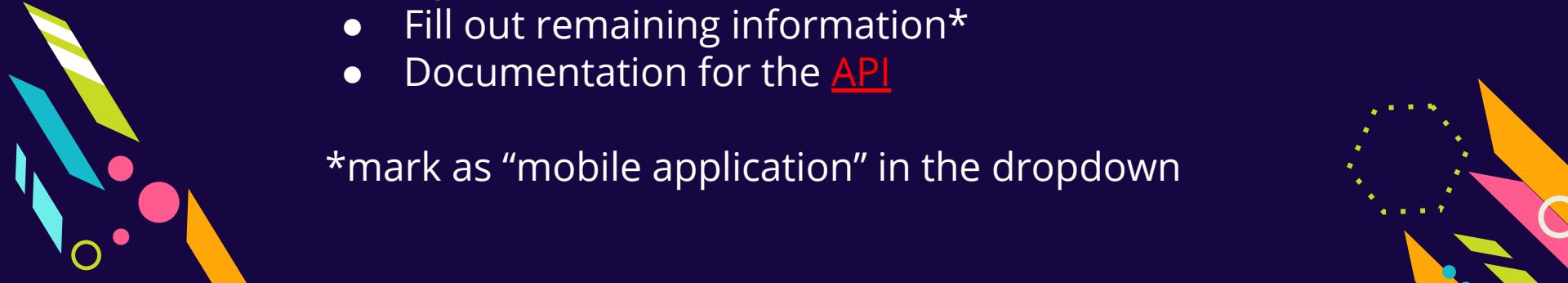
Allows users to browse current movies and view their descriptions utilizing the movie database [API.](#)

# Setting up the API

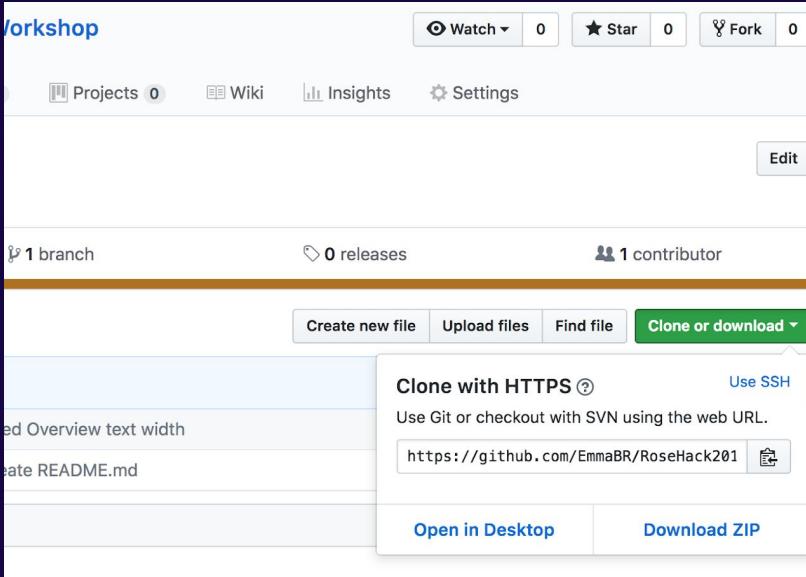


- Create an account at [TheMovieDB.org](https://TheMovieDB.org)
- Click on your profile (top right)
- In the drop down click “settings”
- In the far left column click “API”
- At the bottom of the page hit “click here” to generate a new API key
- Click “developer”
- Agree to terms and conditions
- Fill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

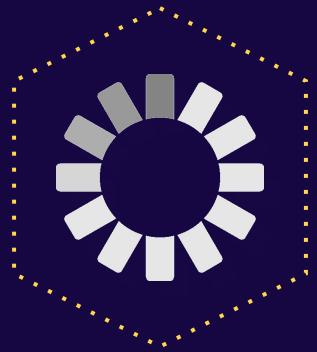
\*mark as “mobile application” in the dropdown



# Clone the Repository



- Go to:  
<https://github.com/EmmaBR/ACM2019-AndroidWorks>
- Clone the Repo or download the zip file.
- Open Android Studio.
- Click open a project and navigate to StartMovieApp and open it.



# Wait...

Your project is building!

# Before we start...

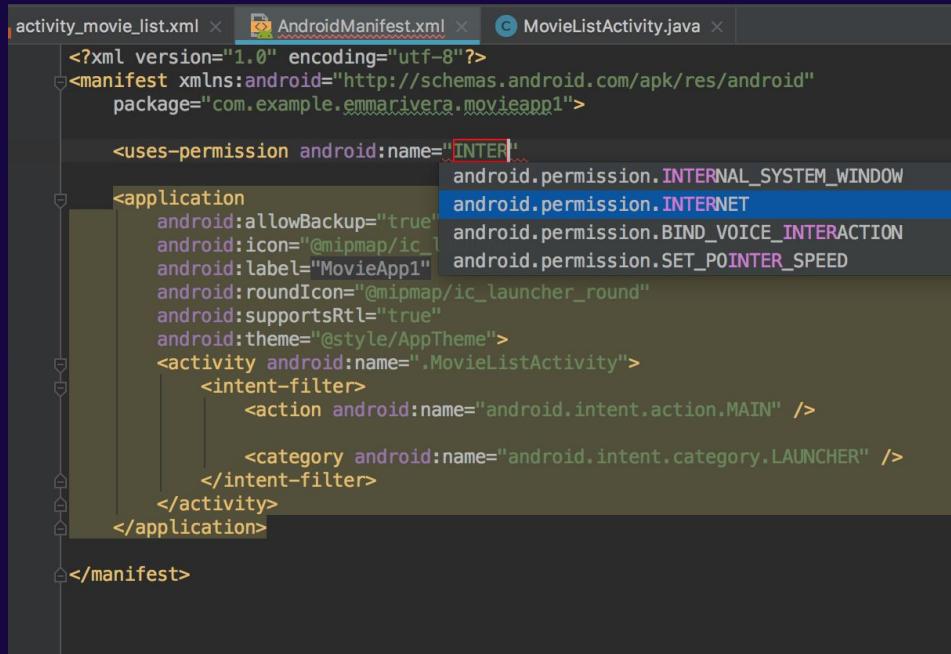


- Android Studio > Preferences > Editor > General > Auto Import : Make sure all the boxes are checked.
- Build, Execution, Deployment > Instant Run : Disable Instant Run by unchecking the first box

# 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



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.emmarivera.movieapp1">

    <uses-permission android:name="INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MovieApp1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MovieListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# Hiding Your API 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).

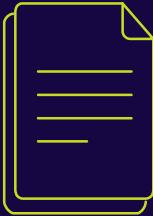
## Step 3

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



# Emulator

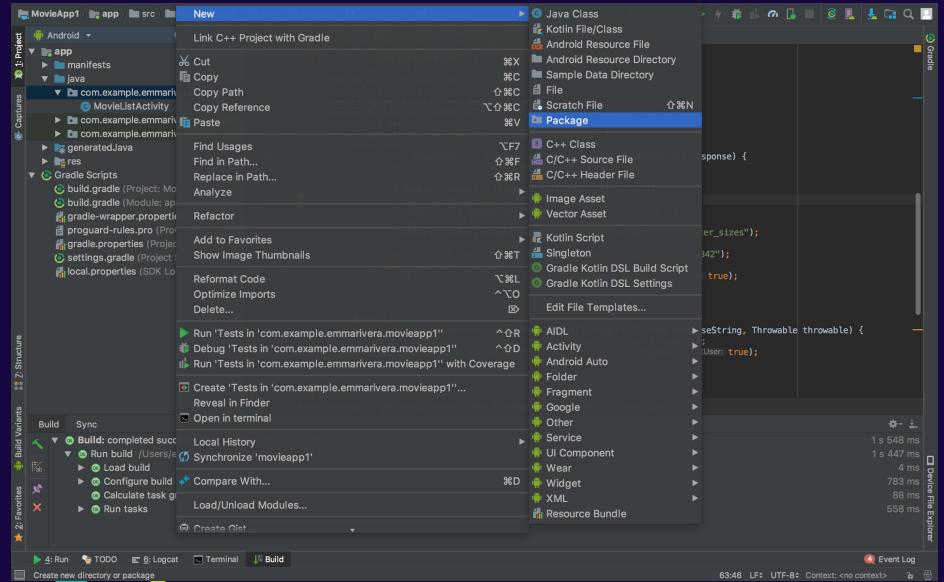
Run the Emulator to test to make sure your project builds and works correctly!!



# Let's add some files!

Creating a Model

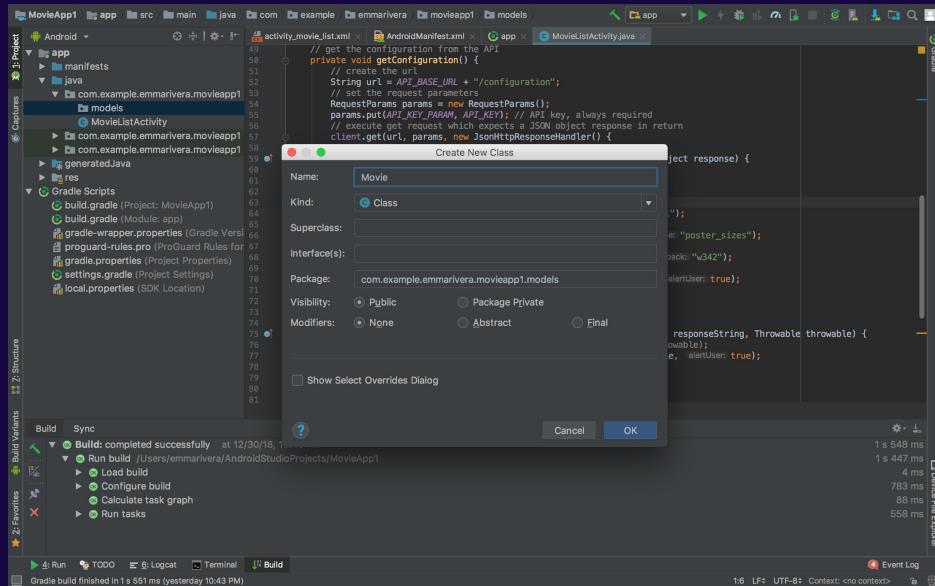
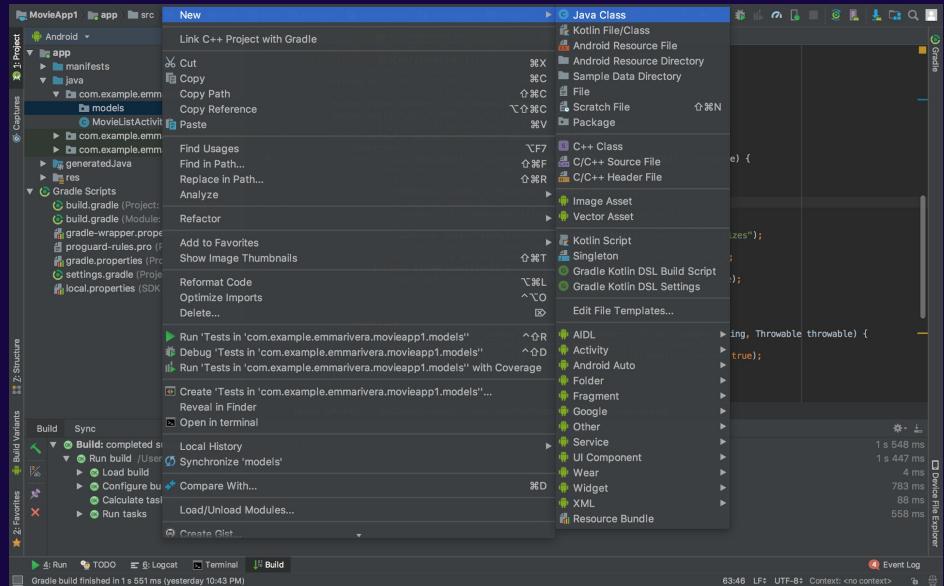
# Create the Model Package



The screenshot shows the Android Studio interface with a Java file open. A 'New Package' dialog box is displayed, prompting the user to enter a package name. The package name 'models' is typed into the input field. The background code shows a portion of the MovieListActivity.java file.

```
private void getConfiguration() {
    // get the configuration from the API
    String url = API_BASE_URL + "/configuration";
    // set the request parameters
    RequestParams params = new RequestParams();
    params.put("API_KEY", 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
                String posterSizeOption = images.getString("secure_base_url");
                // get the poster size
                JSONArray posterSizeOptions = images.getJSONArray("poster_sizes");
                posterSizeOptions.set(0, "w342");
                posterSizeOptions.set(1, "w500");
                posterSizeOptions.set(2, "w785");
                posterSizeOptions.set(3, "w1280");
                posterSizeOptions.set(4, "w342");
                posterSizeOptions.set(5, "w500");
                posterSizeOptions.set(6, "w785");
                posterSizeOptions.set(7, "w1280");
                posterSizeOptions.set(8, "w342");
                posterSizeOptions.set(9, "w500");
                posterSizeOptions.set(10, "w785");
                posterSizeOptions.set(11, "w1280");
                posterSizeOptions.set(12, "w342");
                posterSizeOptions.set(13, "w500");
                posterSizeOptions.set(14, "w785");
                posterSizeOptions.set(15, "w1280");
                posterSizeOptions.set(16, "w342");
                posterSizeOptions.set(17, "w500");
                posterSizeOptions.set(18, "w785");
                posterSizeOptions.set(19, "w1280");
                posterSizeOptions.set(20, "w342");
                posterSizeOptions.set(21, "w500");
                posterSizeOptions.set(22, "w785");
                posterSizeOptions.set(23, "w1280");
                posterSizeOptions.set(24, "w342");
                posterSizeOptions.set(25, "w500");
                posterSizeOptions.set(26, "w785");
                posterSizeOptions.set(27, "w1280");
                posterSizeOptions.set(28, "w342");
                posterSizeOptions.set(29, "w500");
                posterSizeOptions.set(30, "w785");
                posterSizeOptions.set(31, "w1280");
                posterSizeOptions.set(32, "w342");
                posterSizeOptions.set(33, "w500");
                posterSizeOptions.set(34, "w785");
                posterSizeOptions.set(35, "w1280");
                posterSizeOptions.set(36, "w342");
                posterSizeOptions.set(37, "w500");
                posterSizeOptions.set(38, "w785");
                posterSizeOptions.set(39, "w1280");
                posterSizeOptions.set(40, "w342");
                posterSizeOptions.set(41, "w500");
                posterSizeOptions.set(42, "w785");
                posterSizeOptions.set(43, "w1280");
                posterSizeOptions.set(44, "w342");
                posterSizeOptions.set(45, "w500");
                posterSizeOptions.set(46, "w785");
                posterSizeOptions.set(47, "w1280");
                posterSizeOptions.set(48, "w342");
                posterSizeOptions.set(49, "w500");
                posterSizeOptions.set(50, "w785");
                posterSizeOptions.set(51, "w1280");
                posterSizeOptions.set(52, "w342");
                posterSizeOptions.set(53, "w500");
                posterSizeOptions.set(54, "w785");
                posterSizeOptions.set(55, "w1280");
                posterSizeOptions.set(56, "w342");
                posterSizeOptions.set(57, "w500");
                posterSizeOptions.set(58, "w785");
                posterSizeOptions.set(59, "w1280");
                posterSizeOptions.set(60, "w342");
                posterSizeOptions.set(61, "w500");
                posterSizeOptions.set(62, "w785");
                posterSizeOptions.set(63, "w1280");
                posterSizeOptions.set(64, "w342");
                posterSizeOptions.set(65, "w500");
                posterSizeOptions.set(66, "w785");
                posterSizeOptions.set(67, "w1280");
                posterSizeOptions.set(68, "w342");
                posterSizeOptions.set(69, "w500");
                posterSizeOptions.set(70, "w785");
                posterSizeOptions.set(71, "w1280");
                posterSizeOptions.set(72, "w342");
                posterSizeOptions.set(73, "w500");
                posterSizeOptions.set(74, "w785");
                posterSizeOptions.set(75, "w1280");
                posterSizeOptions.set(76, "w342");
                posterSizeOptions.set(77, "w500");
                posterSizeOptions.set(78, "w785");
                posterSizeOptions.set(79, "w1280");
                posterSizeOptions.set(80, "w342");
                posterSizeOptions.set(81, "w500");
            } catch (JSONException e) {
                logError(message: "Failed getting configuration", throwable, alertUser: true);
            }
        }
    });
}
```

# Create the Model Class

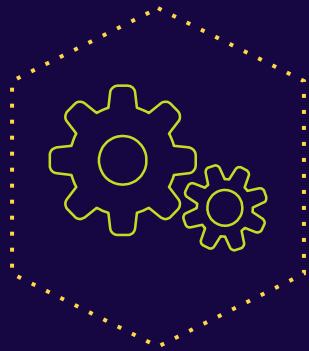


# Setting up the Movie Class

- 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

Show JSON\*

```
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 to be in quotes  
        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;  
    }  
}
```



# 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);
            }
        }

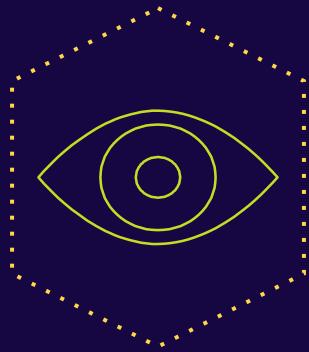
        @Override
        public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {
            logError( message: "Failed to get now playing movies", throwable, alertUser: true);
        }
    });
}

// get the configuration from the API
private void getConfiguration() {
    // ...
}
```

ListActivity > getConfiguration() > new JsonHttpResponseHandler > onSuccess()

```
// 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 throwable) {
            logError(message: "Failed to get now playing movies", throwable, alertUser: true);
        }
    });
}
```

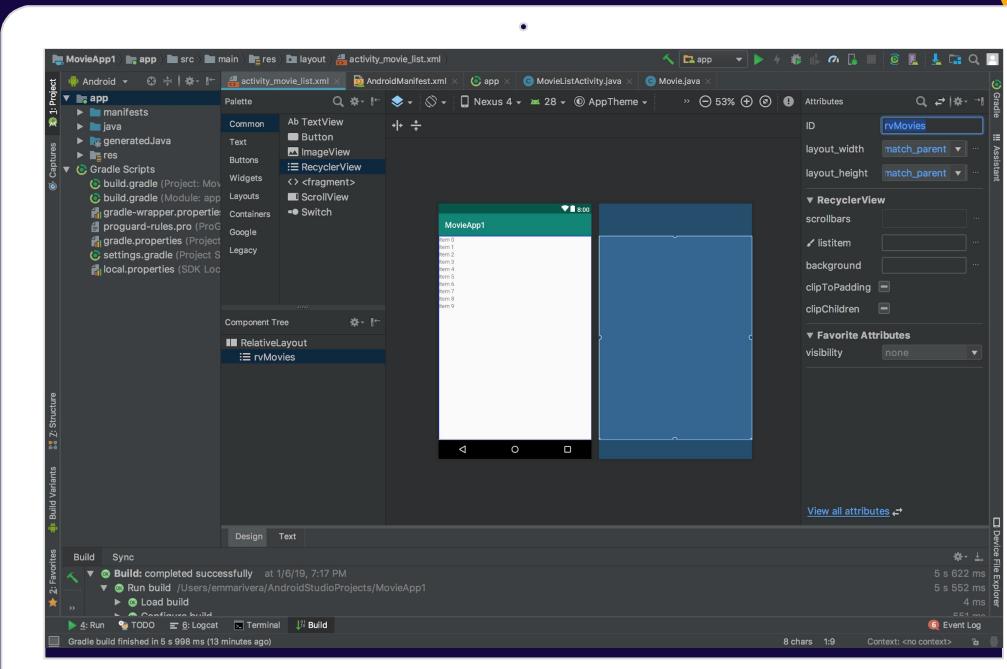


# Making the Data Visible to the User

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!

(for now)

# A 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

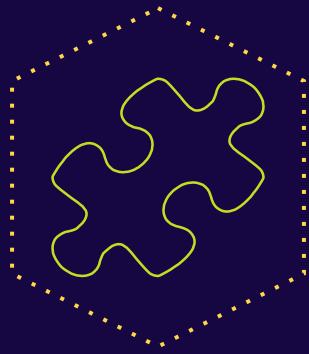
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="5dp"
    android:layout_width="match_parent" android:layout_height="wrap_content">

    <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>
```



# 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.Vie  
wHolder { }
```

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

1: Project 2: Structure 3: Favorites

activity\_movie\_list.xml item\_movies.xml MovieAdapter.java MovieListActivity.java Movie.java

Android manifests java com.example.emmarivera.movieapp1 models MovieAdapter MovieListActivity 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)

Build Sync

Build: completed successfully at 1/6/19, 7:17 PM

Run build /Users/emmarivera/AndroidStudioProjects/MovieApp1 Load build Configure build

Run TODO Logcat Terminal Build

5 s 622 ms 5 s 552 ms 4 ms 551 ms

Event Log

```
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);
        }
    }
}
```

Class 'MovieAdapter' must either be declared abstract or implement abstract method 'onBindViewHolder(VH, int)' in 'Adapter'

# Resolving 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

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

- Project Bar:** MovieApp1 > app > src > main > java > com.example.emmarivera.movieapp1 > MovieAdapter
- Toolbars:** Top bar includes icons for Run, Stop, Build, and Device.
- Side Navigation:** Includes sections for Project, Captures, Structure, Build Variants, Favorites, and Device File Explorer.
- Code Editor:** Displays the `MovieAdapter.java` file with the following code:

```
25 // creates and inflates a new view
26 @NonNull
27 @Override
28 public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
29     // get the context and create the inflater
30     Context context = viewGroup.getContext();
31     LayoutInflater inflater = LayoutInflater.from(context);
32     // create the view using the item_movie layout
33     View movieView = inflater.inflate(R.layout.item_movies, viewGroup, false);
34     // return that view wrapped by a viewHolder
35     return new ViewHolder(movieView);
36 }
37
38 // binds an inflated view to a new item
39 @Override
40 public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
41     // get the movie data at the specified position
42     Movie movie = movies.get(i);
43     // populate the view with the movie data
44     viewHolder.tvTitle.setText(movie.getTitle());
45     viewHolder.tvOverview.setText(movie.getOverview());
46     // TODO -- glide to add image
47 }
48
49 // returns the size of the data set
50 @Override
51 public int getItemCount() {
52     return movies.size();
53 }
54
55 // create view holder class as a static inner class
56 public static class ViewHolder extends RecyclerView.ViewHolder {
57
58     // track view objects
59     ImageView ivPosterImage;
60     TextView tvTitle;
61     TextView tvOverview;
62 }
```

The code editor highlights several methods and variables in orange, such as `onCreateViewHolder`, `onBindViewHolder`, `getItemCount`, and `ViewHolder`.

- Build Tab:** Shows a successful build log:

- Build completed successfully at 1/6/19, 7:17 PM
- Run build /Users/emmarivera/AndroidStudioProjects/MovieApp1
- Load build
- Configure build

- Event Log Tab:** Shows performance metrics:
- 5 s 622 ms
- 5 s 552 ms
- 4 ms
- Bottom Status Bar:** Shows "Gradle build finished in 5 s 622 ms (yesterday: 7:17 PM)"

# Connect Recycler in MovieListActivity

- 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 point
    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
    getConfig();
}
```

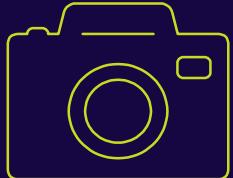
# Notify the Adapter the Dataset Changed



```
// 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);
                    // notify the adapter that a row was added
                    adapter.notifyItemInserted( position: movies.size() - 1);
                }
                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 throwable) {
            logError( message: "Failed to get now playing movies", throwable, alertUser: true);
        }
    });
}
```

MovieListActivity > getNowPlaying() > new JsonHttpResponseHandler > onSuccess()



# Adding Images Using Glide

[Glide](#)

# Create a 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

```
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");
    }
}
```

# Generate Getters



```
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 imageUrl;
    // 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
        imageUrl = 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 imageUrl;
    }

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

# Create Helper Functions for URLs



```
import org.json.JSONException;
import org.json.JSONObject;

public class Config {

    // base url for loading images
    String imageUrl;
    // 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
        imageUrl = 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 getUrl(String size, String path) {
        return String.format("%s%s%s", imageUrl, size, path); // concatenate all three urls
    }

    public String getImageBaseUrl() {
        return imageUrl;
    } Method 'getImageBaseUrl()' is never used

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

# 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

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

- Project Structure:** The project is named "MovieApp1". It contains an "app" module with "java", "res", and "Gradle Scripts" sections. The "java" section includes "Config", "MovieAdapter", and "MovieListActivity". The "res" section includes "drawable", "layout", "mipmap", and "values". The "Gradle Scripts" section includes "build.gradle" (Project: MovieApp1), "build.gradle" (Module: app), "gradle-wrapper.properties", "gradle.properties", "settings.gradle", and "local.properties".
- Code Editor:** The code editor displays the "MovieAdapter.java" file. The code is as follows:

```
Context context;
// initialize with list
public void setMovies(ArrayList<Movie> movies) {
    this.movies = movies;
}

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

// creates and inflates a new view
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
    Context context = viewGroup.getContext();
    Context context = viewGroup.getContext();
    View movieView = LayoutInflater.from(context).inflate(R.layout.item_movies, viewGroup, false);
    return new ViewHolder(movieView);
}

// binds an inflated view to a new item
@Override
public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
    Movie movie = movies.get(i);
    // populate the view with the movie data
    viewHolder.tvTitle.setText(movie.getTitle());
    viewHolder.tvOverview.setText(movie.getOverview());

    // build url for poster image
    String imageUrl = config.getImageUrl(config.getPosterSize(), movie.getPosterPath());
    // load the image using glide
    Glide.with(context).load(imageUrl).into(viewHolder.ivPosterImage);
}

// binds an inflated view to a new item
@Override
public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
    Movie movie = movies.get(i);
    // populate the view with the movie data
    viewHolder.tvTitle.setText(movie.getTitle());
    viewHolder.tvOverview.setText(movie.getOverview());

    // build url for poster image
    String imageUrl = config.getImageUrl(config.getPosterSize(), movie.getPosterPath());
    // load the image using glide
    Glide.with(context).load(imageUrl).into(viewHolder.ivPosterImage);
}
```

- Build Log:** The bottom pane shows the build log with the message "Build: completed successfully" at 1/7/19, 2:53 PM.

# Thanks! 🙌

**Any questions?**

You can find me at:

<https://www.linkedin.com/in/emma-rivera/>

<https://github.com/EmmaBR>