



# *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](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 Project*

- ▷ Go to: [rosehack.com/androidworkshop](http://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-AndroidWorkshop    New Tab    Rose-Hack-2019-Android Wor... +

GitHub, Inc. [US] | https://github.com/EmmaBR/RoseHack2019-AndroidWorkshop

Apps CS 120B: Introdu... 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

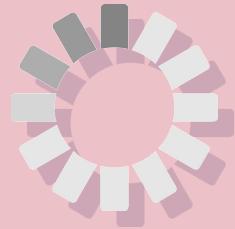
Branch: master ▾ New pull request Create new file Upload files Find file Clone or download

EmmaBR Fixed Overview text width MovieApp1 Fixed Overview text width README.md Create README.md README.md

Clone with HTTPS Use SSH  
Use Git or checkout with SVN using the web URL.  
<https://github.com/EmmaBR/RoseHack2019>

Open in Desktop Download ZIP

# RoseHack2019-AndroidWorkshop



*Wait...*

Your project is building!!

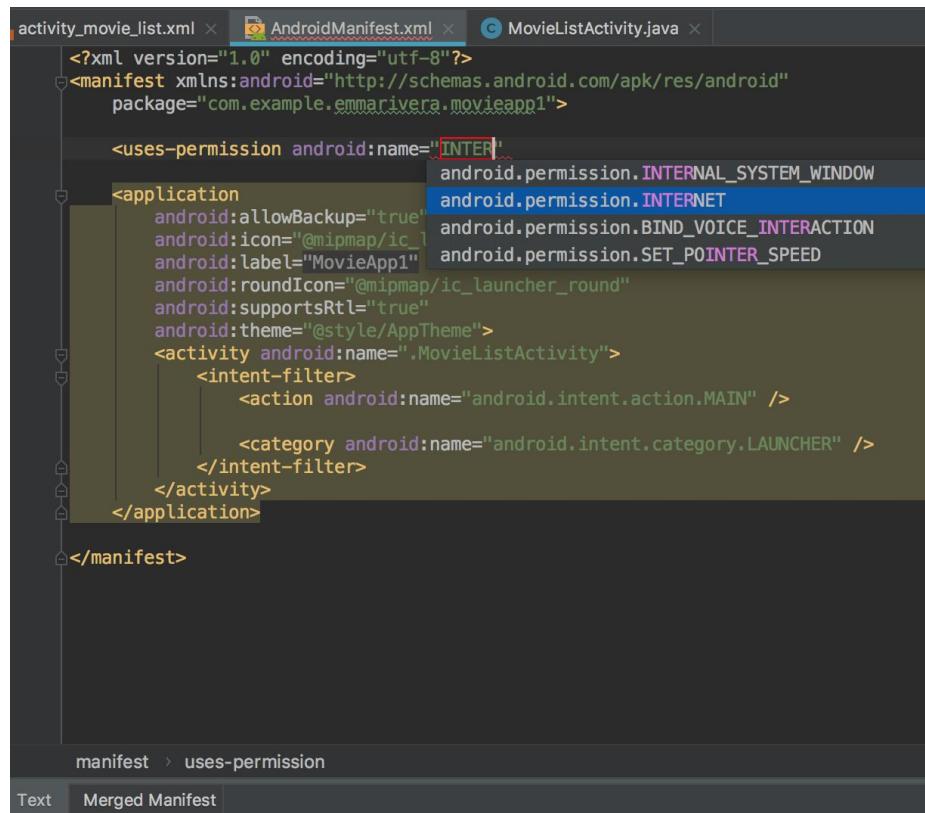
## *Before we Start...*

- ▷ 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



```
<?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>
```

manifest > uses-permission

Text Merged Manifest

## *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*

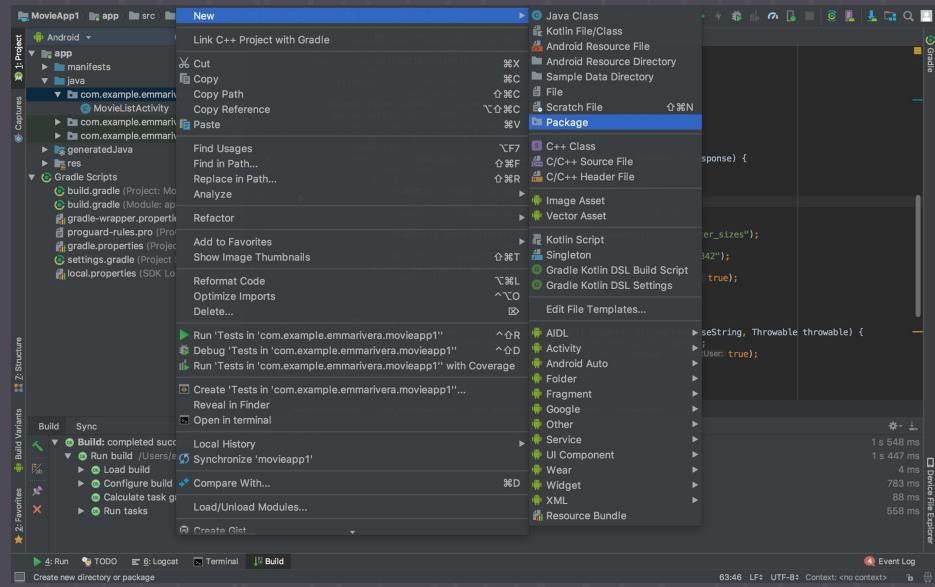
Run the emulator to test to make sure  
your project builds and works correctly!!



# *Let's Add Some Files!*

Creating a Model

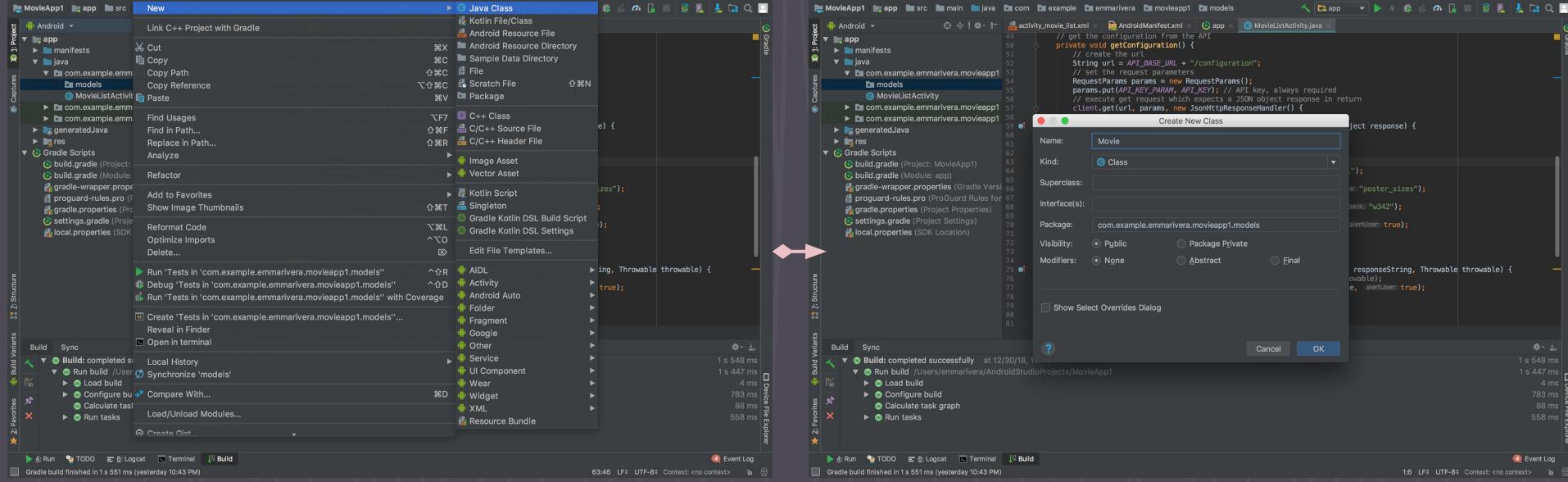
# Create the Model Package



The screenshot shows the Java code for 'MovieListActivity.java' with a call to 'client.get(url, params, new JsonHttpResponseHandler() { ... })'. A pink double-headed arrow points from the 'New Package' step in the previous screenshot to this code. A 'New Package' dialog is open, prompting for a package name, with 'models' typed in. The Java code includes imports for 'com.google.gson.JsonObject', 'com.google.gson.JsonArray', and 'com.google.gson.JsonObject'. The 'onSuccess' method handles JSON responses related to movie poster sizes.

```
private void getConfiguration() {
    // create the url
    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 base_url = images.getString("base_url");
                // get the poster size
                JSONArray posterSizeOptions = images.getJSONArray("poster_sizes");
                if (posterSizeOptions.length() > 0) {
                    poster_size = posterSizeOptions.getJSONObject(0).getString("url");
                } else {
                    poster_size = "w342";
                }
            } catch (JSONException e) {
                Log.e("MovieListActivity", "getConfiguration()", e);
            }
        }
        ...
    });
}
```

# 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

```
5  public class Movie {  
6  
7      // values from the API  
8      private String title;  
9      private String overview;  
10     private String posterPath; // only the path  
11  
12     // initialize from JSON data  
13     @ [ ] public Movie(JSONObject object) throws JSONException {  
14         title = object.getString( name: "title"); // remember key's have to  
15         overview = object.getString( name: "overview");  
16         posterPath = object.getString( name: "poster_path");  
17     }  
18  
19     /** NOTE: because these variables were declared private we  
20      * need to create getters to expose this data to expose this data  
21      * to the activity  
22      */  
23  
24     public String getTitle() {  
25         return title;  
26     }  
27  
28     public String getOverview() {  
29         return overview;  
30     }  
31  
32     public String getPosterPath() {  
33         return posterPath;  
34     }  
35 }  
36  
37 }
```



# *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("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() {
    // create the ...
    telListActivity > 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("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("Failed to parse now playing movies", e, alertUser: true);
            }
        }

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

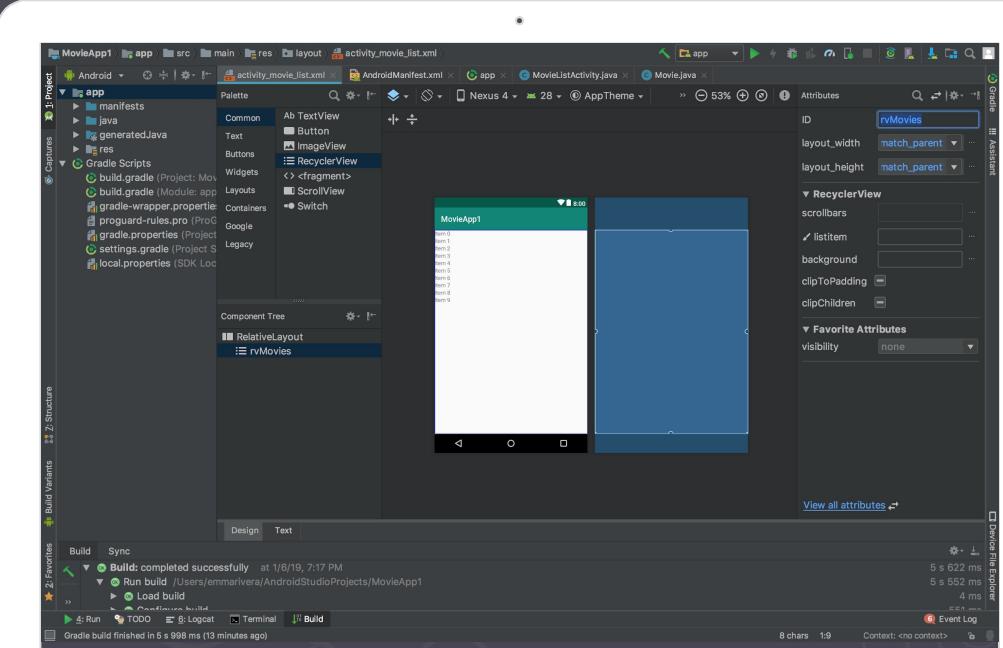


# *Making the data visible to the user*

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

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      android:layout_margin="5dp"
5      android:layout_width="match_parent" android:layout_height="wrap_content">
6
7      <ImageView
8          android:id="@+id/ivPosterImage"
9          android:layout_width="90dp"
10         android:layout_height="135dp"
11         android:layout_alignParentLeft="true"
12         android:layout_alignParentTop="true"
13         app:srcCompat="@android:color/black"
14         android:layout_marginRight="5dp"/>
15
16      <TextView
17          android:id="@+id/tvTitle"
18          android:layout_width="294dp"
19          android:layout_height="wrap_content"
20          android:layout_alignParentTop="true"
21          android:layout_marginTop="0dp"
22          android:layout_toRightOf="@+id/ivPosterImage"
23          android:text="Movie Title"
24          android:textAppearance="@android:style/TextAppearance.Large" />
25
26      <TextView
27          android:id="@+id/tvOverview"
28          android:layout_width="294dp"
29          android:layout_height="105dp"
30          android:layout_below="@+id/tvTitle"
31          android:layout_toEndOf="@+id/ivPosterImage"
32          android:text="TextView" />
33  </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.View  
Holder { }
```

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<MovieA  
dapter.ViewHolder>

MovieApp1 app src main java com example emmarivera movieapp1 MovieAdapter

1: Project 2: Favorites 3: Captures 4: Build Variants 5: Z: Structure 6: Build 7: Sync 8: 2: Favorites

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

1 2 import android.support.annotation.NonNull; 3 import android.support.v7.widget.RecyclerView; 4 import android.view.View; 5 import android.widget.ImageView; 6 import android.widget.TextView; 7 8 import com.example.emmarivera.movieapp1.models.Movie; 9 10 import java.util.ArrayList; 11 12 public class MovieAdapter extends RecyclerView.Adapter<MovieAdapter.ViewHolder> { 13 14 // list of movies 15 ArrayList<Movie> movies; 16 17 // initialize with list 18 public MovieAdapter(ArrayList<Movie> movies) { 19 this.movies = movies; 20 } 21 22 // create view holder class as a static inner class 23 public static class ViewHolder extends RecyclerView.ViewHolder { 24 25 // track view objects 26 ImageView ivPosterImage; 27 TextView tvTitle; 28 TextView tvOverview; 29 30 31 32 33 34 35 36 37 38 39 40 } } MovieAdapter

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

Run build /Users/emmarivera/AndroidStudioProjects/MovieApp1

Run build Load build Configure build

4: Run TODO 6: Logcat Terminal Build 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>

## *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

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

- Project Structure:** The left sidebar displays the project structure under "app".
- Code Editor:** The main area shows the `MovieAdapter.java` file. The code implements a RecyclerView adapter for movie data. It includes methods for creating view holders, binding data to view holders, and returning the item count.
- Build Tab:** The bottom-left tab bar shows the "Build" tab is selected, indicating the build process has completed successfully.
- Logcat Tab:** The "Logcat" tab is also visible in the bottom navigation.
- Bottom Status Bar:** The status bar at the bottom indicates the build took 5 seconds and 998 ms, and the current time is 47:38.

```
// creates and inflates a new view
@NotNull
@Override
public ViewHolder onCreateViewHolder(@NotNull ViewGroup viewGroup, int i) {
    // get the context and create the inflater
    Context context = viewGroup.getContext();
    LayoutInflator 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(@NotNull 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;
}
```

## *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 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
    getConfiguration();
}
```

```
// 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()

Finally Notify the Adapter that the Dataset Changed



# *Adding Images Using Glide*

Glide

## *Create New Model Class Called “Config”*

- ▷ Cut the imageUrl 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 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" );
    }
}
```

```
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.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 getImageUrl(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;
    }
}
```

## Create Helper Function for CreatingUrls

# *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 imageUrl %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

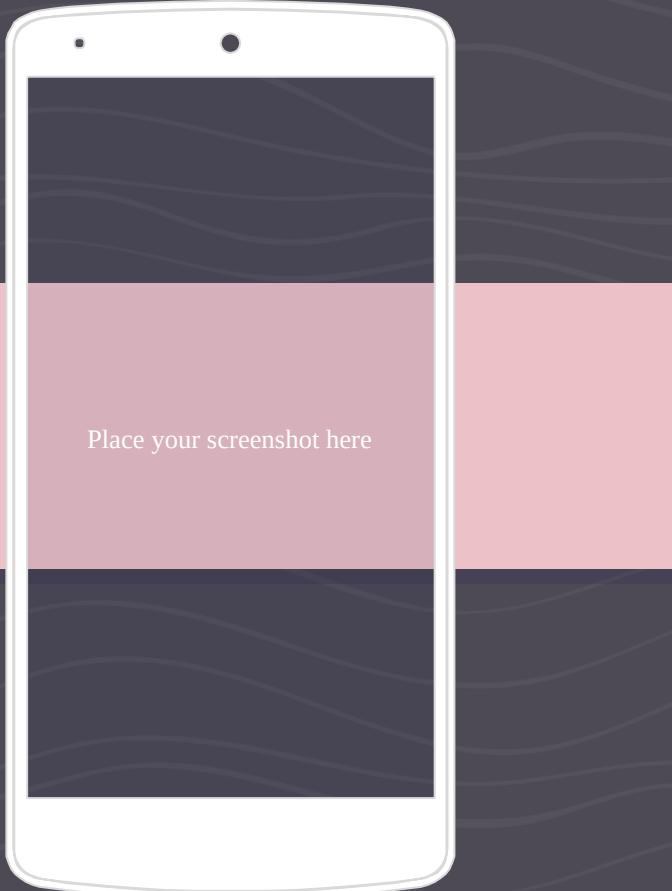
Load image with Glide

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

- Project Structure:** The project is named "MovieApp1". It contains an "app" module with "AndroidManifest.xml", "src/main/java/com/example/emmarivera/movieapp1/MovieAdapter.java", and "src/main/java/com/example/emmarivera/movieapp1/MovieListActivity.java". There are also "res" and "gradle" directories.
- Code Editor:** The code editor displays `MovieAdapter.java`. The code initializes a list of movies and creates a new view for each item using `LayoutInflater` and `View.inflate`. It then binds the inflated view to a new item using `onBindViewHolder`.
- Build Tab:** The build tab shows a successful build completed at 2:53 PM on 7/19. It includes a "Run build" entry for the "app" module.
- Logcat Tab:** The logcat tab shows no logs.
- Terminal Tab:** The terminal tab shows a command to "Configure build".
- Build Tab (Details):** Shows the build finished in 5s 827ms (34 minutes ago).
- Event Log Tab:** The event log tab shows no events.

***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](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

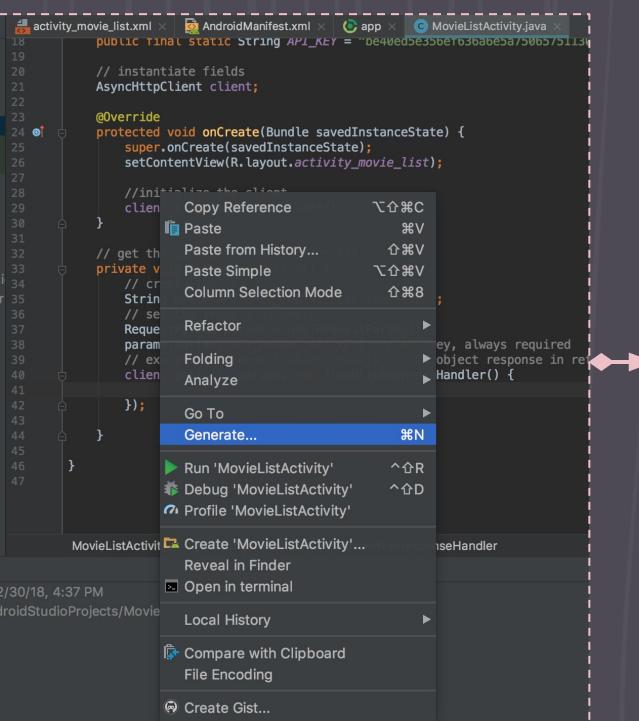
**Client** - the software that consumes or uses the API

**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.

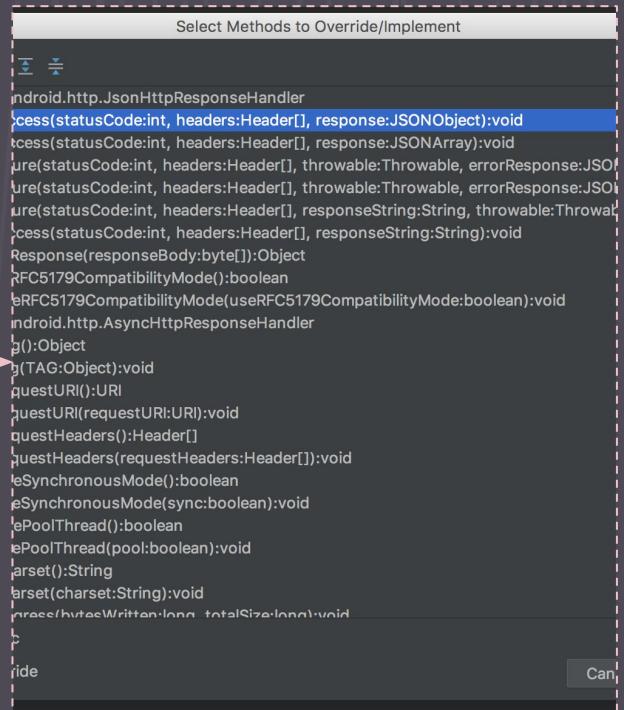
# Generating Override Methods



Right Click ->  
Generate

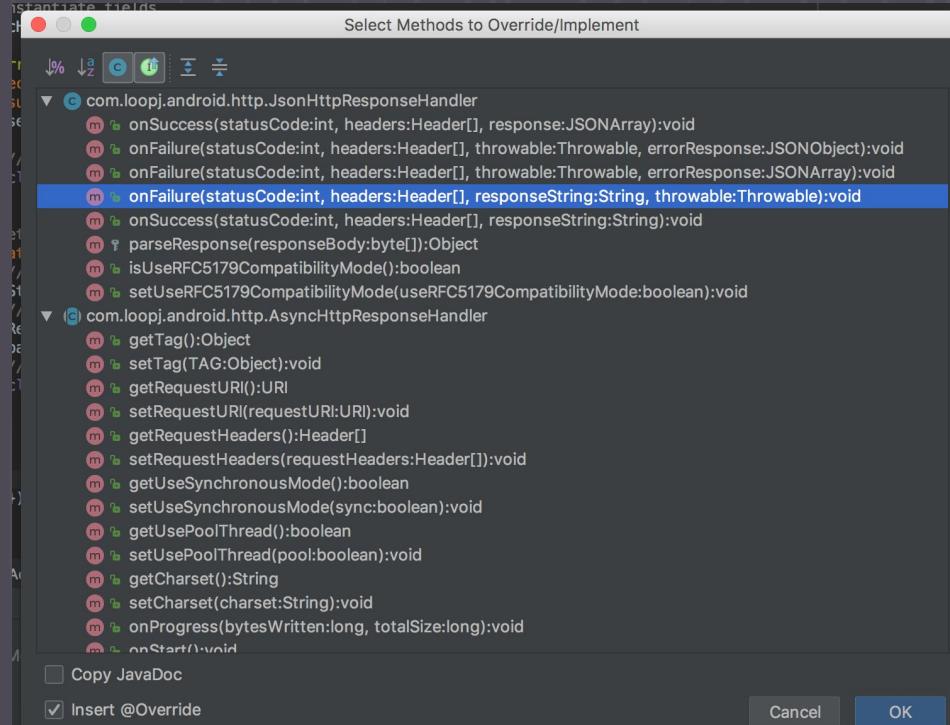
A screenshot of the Android Studio code editor showing the same code as the first image. The context menu is open again, and the 'Override Methods...' option is highlighted in blue. Other options in the menu are Generate, ToString(), Delegate Methods..., and Copyright.

Override  
Methods



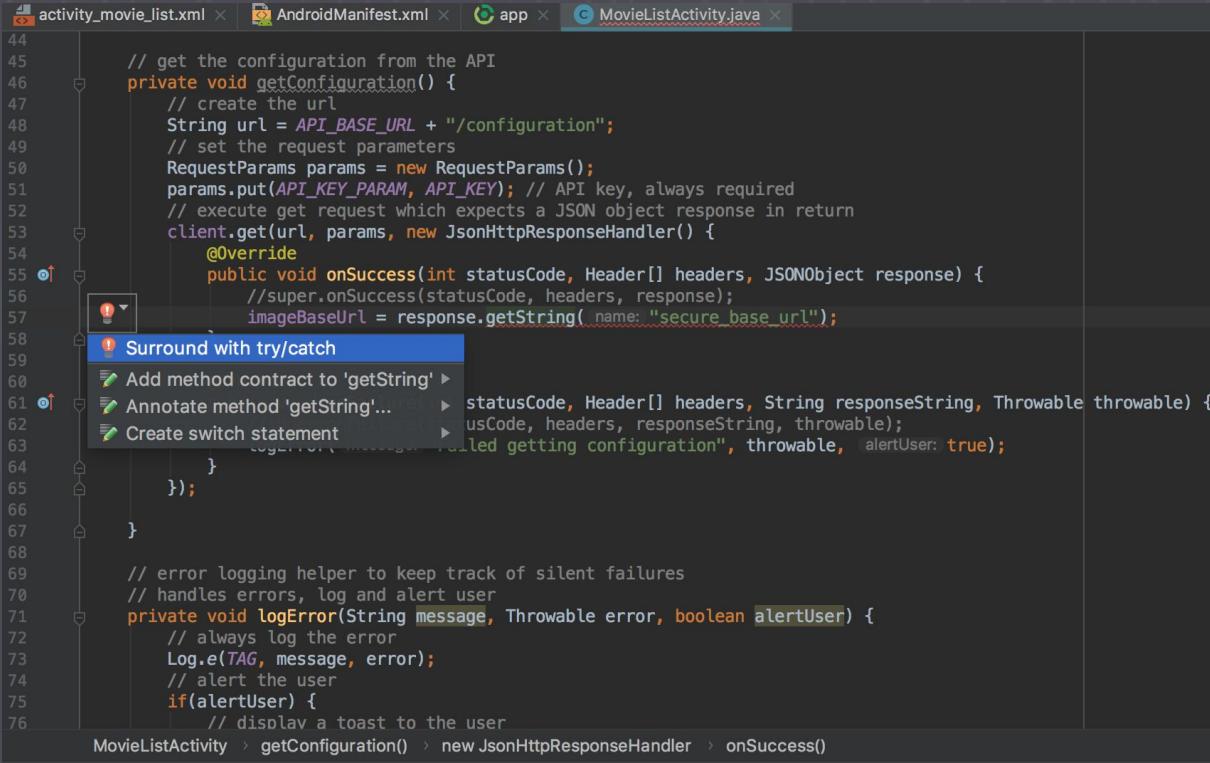
Override: onSuccess

# *Generating Override Methods cont...*



**Override: onFailure**

# Handling Errors



The screenshot shows the Android Studio interface with the tab bar at the top containing activity\_movie\_list.xml, AndroidManifest.xml, app, and MovieListActivity.java. The MovieListActivity.java tab is active, displaying Java code. A code completion dropdown menu is open over the line of code:

```
    imageBaseUrl = response.getString(name: "secure_base_url");
```

The dropdown menu has the following options:

- Surround with try/catch
- Add method contract to 'getString'
- Annotate method 'getString'...
- Create switch statement

The code in MovieListActivity.java is as follows:

```
44 // get the configuration from the API
45     private void getConfiguration() {
46         // create the url
47         String url = API_BASE_URL + "/configuration";
48         // set the request parameters
49         RequestParams params = new RequestParams();
50         params.put(API_KEY_PARAM, API_KEY); // API key, always required
51         // execute get request which expects a JSON object response in return
52         client.get(url, params, new JsonHttpResponseHandler() {
53             @Override
54             public void onSuccess(int statusCode, Header[] headers, JSONObject response) {
55                 //super.onSuccess(statusCode, headers, response);
56                 imageBaseUrl = response.getString(name: "secure_base_url");
57             }
58         });
59     }
60
61     // error logging helper to keep track of silent failures
62     // handles errors, log and alert user
63     private void logError(String message, Throwable error, boolean alertUser) {
64         // always log the error
65         Log.e(TAG, message, error);
66         // alert the user
67         if(alertUser) {
68             // display a toast to the user
69         }
70     }
71 }
```

At the bottom of the code editor, a navigation bar shows the path: MovieListActivity > getConfiguration() > new JsonHttpResponseHandler > onSuccess()

## *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
private 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,
                             Throwable throwable) {
            //super.onFailure(statusCode, headers, responseString, throwable);
            logError( message: "Failed getting configuration", throwable, alertUser: true)
        }
    })
}
```

tActivity > getConfiguration() > new JsonHttpResponseHandler > onSuccess()