

5. Web Services

In this section we will learn how to use web services by making API call then reading the response, parsing data into custom objects.

We will use Open Movie Data Base API to make a web request and API returns JSON in response. We will parse the JSON data and create our own object of Movie which will have properties name, year released , genre and others. First we will see how OMDb API works

Go to the browser and enter this url

URL: <http://www.omdbapi.com/?t=titanic>

Response is generated in JSON format which contains movie details as shown in figure below.



```
{
  Title: "Titanic",
  Year: "1997",
  Rated: "PG-13",
  Released: "19 Dec 1997",
  Runtime: "194 min",
  Genre: "Drama, Romance",
  Director: "James Cameron",
  Writer: "James Cameron",
  Actors: "Leonardo DiCaprio, Kate Winslet, Billy Zane, Kathy Bates",
  Plot: "A seventeen-year-old aristocrat, expecting to be married to a rich claimant by her mother, falls in love with a kind but poor artist aboard the luxurious, ill-fated R.M.S. Titanic.",
  Language: "English, French, German, Swedish, Italian, Russian",
  Country: "USA",
  Awards: "Won 11 Oscars. Another 111 wins & 62 nominations.",
  Poster: "http://ia.media-imdb.com/images/M/MV5BMjEzMDM0N15BM15BanBnXkFtZTcwMzkyOTUwNw@@._V1_SX300.jpg",
  Metascore: "74",
  imdbRating: "7.7",
  imdbVotes: "606,110",
  imdbID: "tt0120338",
  Type: "movie",
  Response: "True"
}
```

Figure 15 Snapshot of JSON response by web service

In the search parameters we have requested “t=titanic” which is the title of the movie if we change it to “Batman” we will get results for batman. Now our objective is to make this URL request in android framework and get Title, Year and Genre from this data and send it to MovieDetailsView to display it using intents (which we have already done in section 4.2.3).

5.1 Permission

Since we will be using internet from user's device therefore we must ask user's permission to use internet. Add following line in your Application Manifest where parent tag is manifest.

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

5.2 API Call and Response

Following part of the url request will always remain same when searching for title.

URL: <http://www.omdbapi.com/?t=>

Now we need one input field for the user where title is typed and we add that to this url after "t=". Create a UI which has following three items

- EditText box
- Button to search
- TextView to display result.

Sample UI in the figure. Create a controller for this. Let's name it SearchActivity.

Note: In MainActivity add intent to go to search activity when Search Button is clicked also add newly created activity to your manifest file.

Code for Search Activity :

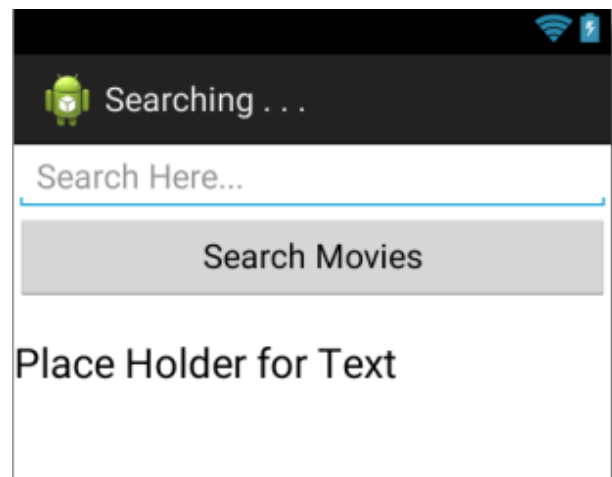


Figure 16 Search Activity Layout

```
EditText et_Search;  
Button btn_Search;  
TextView tv_Result;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    // TODO Auto-generated method stub  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_search);  
    et_Search = (EditText) findViewById(R.id.et_searchbox);  
    btn_Search = (Button) findViewById(R.id.BTN_SearchQueryButton);  
    tv_Result = (TextView) findViewById(R.id.tv_SearchResult);  
}
```

```
}

@Override
protected void onStart() {
    // TODO Auto-generated method stub
    super.onStart();
    btn_Search.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            if (et_Search.getText().toString().length() > 0) {
                searchTitle(et_Search.getText().toString());
            } else {
                Toast.makeText(SearchActivity.this, "Search Box empty",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

In `onClick()` method of Search Movies button (`btn_Search`) first we check if the `EditText` is not empty than calling our own method called `searchTitle(String)` which takes a string parameter containing the title user has entered.

In the `searchTitle` method first we will check if the title has some spaces then replace it with `%20` in order to get correct URL format. Then we will create the search URL string and pass the URL to our `SearchTask`, delegate this current activity so that results are sent back to `SearchActivity` and execute the task.

```
protected void searchTitle(String title) {
    title = title.replace(" ", "%20");

    String url = "http://www.omdbapi.com/?t=" + title;
    SearchTask searchTask = new SearchTask();
    searchTask.delegate = this;
    searchTask.execute(url);

    Toast.makeText(getBaseContext(), R.string.searching,
        Toast.LENGTH_LONG)
        .show();
}
```

In order to get the response back to `SearchActivity` we will create an interface which will be implemented in `SearchActivity` class. Now create a new java class and name it `AsyncResponse`. Copy the follow code to this class.

Note: Change package name to yours.

```
package fi.metropolia.qaisersiddique;

public interface AsyncResponse {
    void processFinish(String result);
}
```

After creating the interface we will implement it to SearchActivity by adding " **implements AsyncResponse** " in class definition.

```
public class SearchActivity extends Activity implements
AsyncResponse{
```

Final step is to write processFinish(String) method in the SearchActivity class.

```
@Override
    public void processFinish(String searchResults) {
        // TODO Auto-generated method stub
        tv_Result.setText(searchResults);
    }
```

In this method we are getting the search results in string format and displaying the raw JSON in the text view of Search Activity for now. Later we will change the code and in processFinish() method we will send raw JSON to our Parser which in return will give an object of Movie, to be used to send the details to MovieDetailsView.

Finally in order to have Search functionality complete we will write our SearchTask which is AsyncTask. Create a new java file in the project and copy the following code.

SearchTask is an AsyncTask which runs in the background and does the network operations so that main UI thread is not blocked. If we try to make network calls Android Framework will generate error message due to strict policy of not doing any network operations in your main thread.

```
package fi.metropolia.qaisersiddique;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URI;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.os.AsyncTask;

public class SearchTask extends AsyncTask<String, Void, String> {
    public AsyncResponse delegate = null;
    private StringBuffer sb;

    protected String doInBackground(String... params) {
        String url = params[0];
        try {
            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet();
            request.setURI(new URI(url));
            HttpResponse response = client.execute(request);
            BufferedReader in = new BufferedReader(new
InputStreamReader(
                response.getEntity().getContent()));
            sb = new StringBuffer("");
            String line = "";
            while ((line = in.readLine()) != null) {
                sb.append(line);
            }
            in.close();
            System.out.println(sb.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return sb.toString();
    }

    protected void onPostExecute(String result) {
        delegate.processFinish(result);
    }
}
```

doInBackground will be called, when searchTask is executed and we send a String url as the parameter. Using HttpClient and HttpGet we fetch the HttpResponse object. Then using the String builder response is converted to String and we return response to onPostExecute method in form of a string.

onPostExecute is called when background process is finished. In this method we take the result (a string) and send it processFinished() delegated to this call via interface.

At this point application should run without errors and display results as shown in screenshot below. Now we will make changes to our processFinish() method in SearchActivity and parse the JSON to create Movie Object.

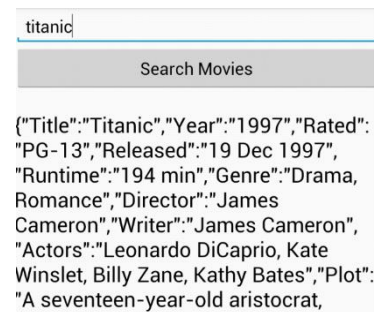


Figure 17 JSON response in TextView

5.3 Parsing JSON and Custom Objects

Let's first change processFinished method remove call to setText and add calls to 2 new methods.

```
@Override
public void processFinish(String searchResults) {

    Movie movie = createMovieObjectFromJSON(searchResults);
    displayMovieObject(movie);
}

private Movie createMovieObjectFromJSON(String searchResults) {
    return null;
}

private void displayMovieObject(Movie movie) {
}
```

Our target is to take Raw JSON, Parse it into Movie Object. Now we will create our Movie class with three properties name, year and genre. Create a new java class called Movie and copy this code.

```
package fi.metropolia.qaisersiddique;

public class Movie {

    public String name, year, genre;

    public Movie(String name, String year, String genre) {
        super();
        this.name = name;
        this.year = year;
        this.genre = genre;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getYear() {
        return year;
    }

    public void setYear(String year) {
        this.year = year;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

}
```

Now in the createMovieObjectFromJSON method we will send String and get a movie object in return.

```
private Movie createMovieObjectFromJSON(String searchResults)
{
    String name, year, genre;
    Movie movie = null;
    try {
        JSONObject jobject = new
JSONJSONObject(searchResults);
        if (jobject.has("Title")) {
            name = jobject.get("Title").toString();
        } else {
            name = "Name not found";
        }
        if (jobject.has("Year")) {
            year = jobject.get("Year").toString();
        } else {
            year = "Year not found";
        }
        if (jobject.has("Genre")) {
            genre = jobject.get("Genre").toString();
        } else {
            genre = "Genre not found";
        }

        movie = new Movie(name, year, genre);
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return movie;
}
```

The movie object returned from previous method will go to displayMovieObject() method which will take the information and print on the screen for user .

```
private void displayMovieObject(Movie movie) {
    String result;
    result = "Found " + movie.getName()+ "
(+movie.getYear()+)";
    tv_Result.setText(result);
}
```

Note: Use all strings from String.xml instead of hardcoding like in the snippets above. This was used just to make it easier to understand.

Task: Add a View Details Button to Search Activity which will be Invisible until the displayMovieObject() method is called. Once it's called then View Details button should be visible and onClick method should take user to MovieDetailsView Activity and send movie object's properties using intents as done in section 4.2.2.

Hint: You can hide widget in both xml and Java. Keywords are "Visibility GONE".

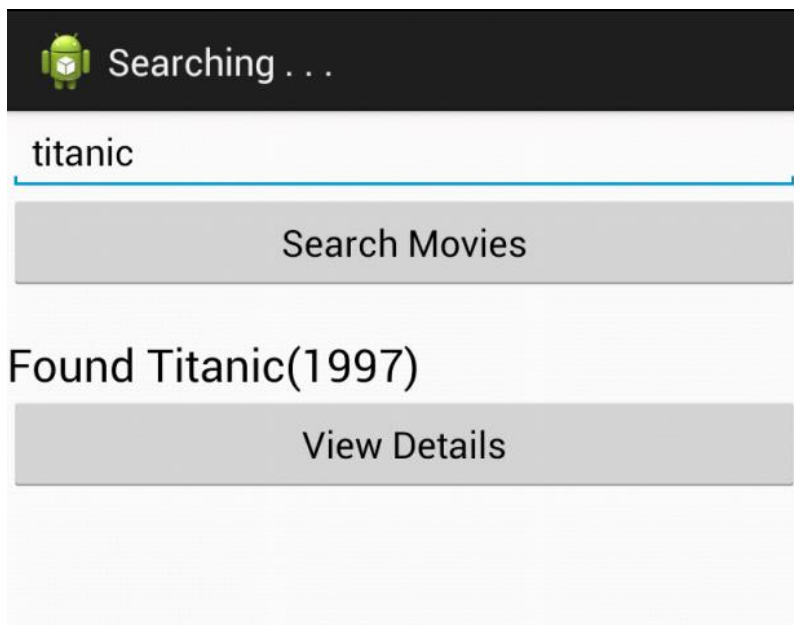


Figure 18 Search result shown

After finishing the TASK View Details button should take you to MovieDetailsView activity where name, year and genre is displayed.
Compile and Run.

TASK: Apart from name year and genre take another value from JSON data of your choice and display it.