

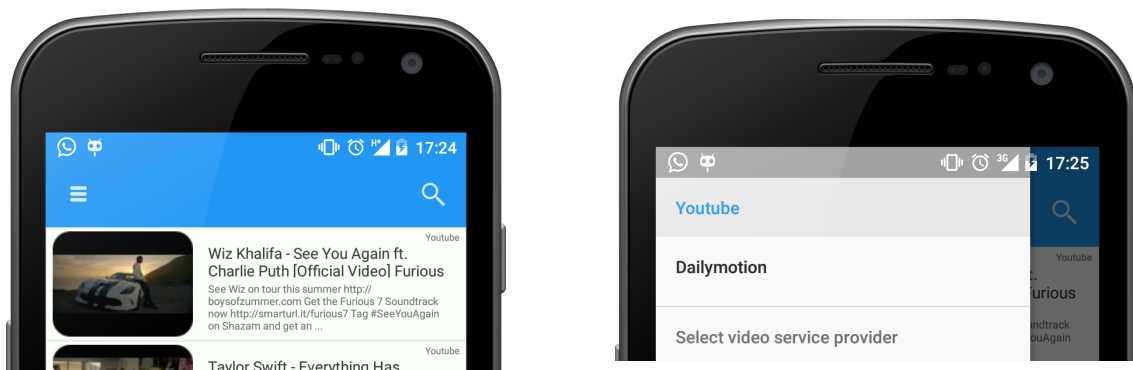
# Video ping

Video ping is an app developed for the CroApps 2015 contest.

## Lets go material

The application provides a simple interface for searching and viewing content from various video providers online.

Video ping takes into consideration googles official material design guidelines for providing a simple yet intuitive interface. Taking into consideration older users of the Android platform, Video ping is made using the AppCompat v7 library so It can provide compatibility among wide variety of Android versions. Among other thing is uses the new Toolbar widget which replaces the existing ActionBar - providing the same functionality but more customisability for the programmer. Animations, Drawers and Material designs - It's all there.



## Libraries used in the project

**AppCompat v7** - googles official library that allows backward compatibility for the new material design widget

**RecyclerView v7** - part of AppCompat v7

**MaterialDrawer 2.9.7** - excellent library for sweat-free implementation of the navigation drawer, with superb documentation and various pre-provided elements

**Google Youtube API v3** - API library needed for connecting to the Google servers

**Json** - lightweight data exchange format (simply a must)

**Google query 0.25.9** - great library providing all the code needed for grabbing images from links. It also provides great memory management(caching) system and view animations

**SmoothProgressBat 1.1.0** - visual sugar that is cross SDK version fully compatible and provides the user feedback about internet fetching and processing processes

So... what about the code? And how do I add new providers?

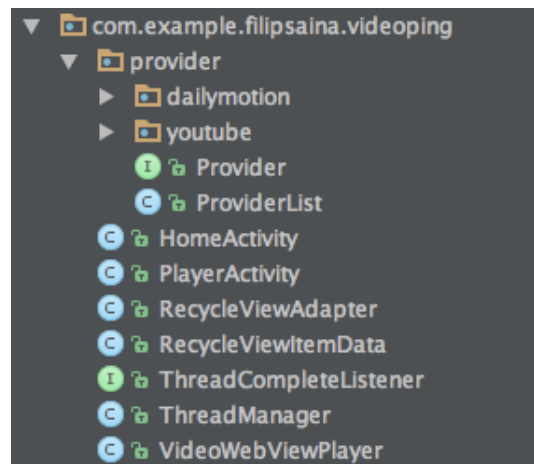


Figure 1.1

Video ping comes with 2 interfaces and 7 object classes (figure 1.1) + provider classes. For a programmer to add a new service to the application there are only two things to take into consideration:

- 1) your provider object needs to implement all methods from the 'Provider' interface
- 2) A new instance of the provider object should be enlisted in the 'ProviderList' object.

The 'Provider' interface (figure 1.2) requires the implementation of only the most necessary methods. Methods worth noting are the 'fetchDataFromServer' and 'getFullVideoUrl'. The first one should contain all the necessary code that handles the connecting (http) and parsing the return data.

```
package com.example.filipsaina.videoping.provider;

import ...

/**
 * Interface that every provider class must fulfill.
 * Created by filipsaina on 15/05/15.
 */
public interface Provider {

    public String getProviderName();
    public List<RecyclerViewItemData> fetchDataFromServer(String searchTerm);
    public String getFullVideoUrl(String videoId, String startTime);
}
```

Figure 1.2

The data returned is a List of RecyclerViewItemData items that should be declared within the method and put into a ArrayList(preferably). Necessary to mention is that this method(fetchDataFromServer) is going to be executed within its own Thread so the programmer does not need to worry about performance.

The later method provides the method caller with a String object containing the full link to the embedded video for the videoId(see already provided code for examples).

Currently the data required to create each RecyclerViewItemData Object(Figure 1.3) is quite simple, but Its easily extendable(providing some additional declarations in the .xml layout file and the RecyclerViewAdapter Class)

```
public class RecyclerViewItemData {  
  
    private String videoTitle;  
    private String videoId;  
    private String imageURL;  
    private String videoDescription;  
    private int providerIndex;  
}
```

Figure 1.3

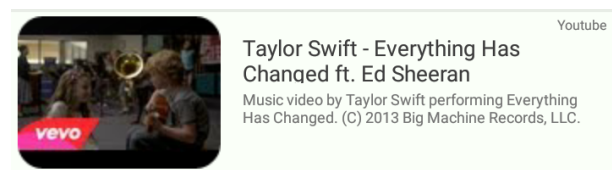


Figure 1.4 - an example of a search result in the RecyclerView

So now, the programmer has implemented the necessary methods and did all the code within. In order for the class to be executed within the code, it is necessary to include a new instance of the Object in the listOfAllProviders Providers array(Figure 1.5). And thats it, in the application now you should see a new item in your drawer and be able to perform search queries on your favourite provider.

```
//Every new provider needs to be included in this list  
private static Provider[] listOfAllProviders = {  
    new YoutubeProvider(),           //currentProviderIndex is currently set to Youtube(0)  
    new DailymotionProvider()  
};
```

Figure 1.5

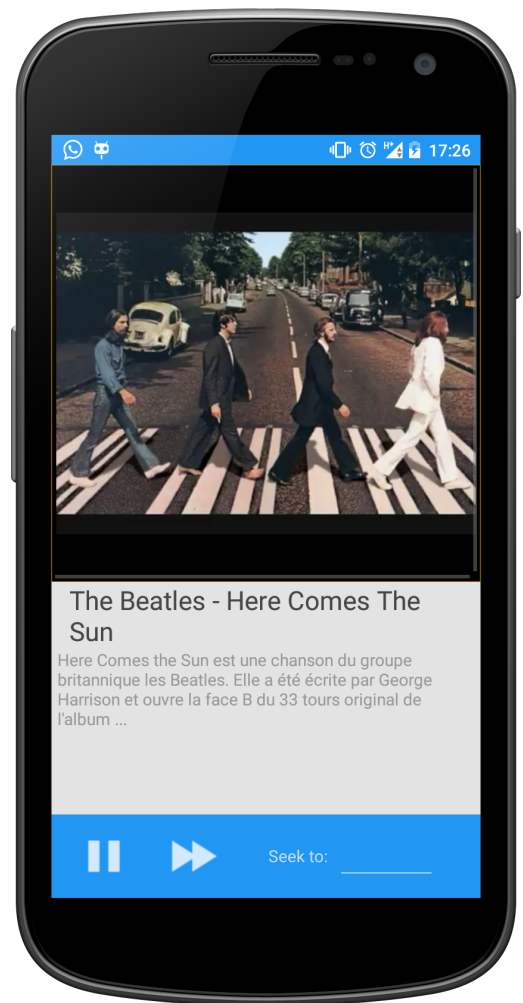
## Play. Stop. Pause.

Video ping uses an extended WebView component for previewing media content. For more information check out the VideoWebViewPlayer class. It is a preset WebView component that comes with controller methods.

Additionally, the player catches all user input and simply ignores it - now the user is obligated to use the outside controls.

Video should autostart on the activity call - this is achieved by simulating a user press every time the WebView completes loading the page (so is the play/pause calls). That way, regardless of what provider we implement, it is bound to provide the same playback features.

(note: The defined behaviour for the user interaction calls (onBackPressed, etc.) is defined in the PlayerActivity class, not in the VideoWebViewPlayer class)



### *Note:*

*Currently implemented providers are Youtube and Dailymotion. While Youtube provides full targeted functionality, the Dailymotion lacks the 'seek to' option. The reason is that Dailymotion does not provide a way of controlling media playback in such a manner and should be taken into consideration by the reviewers.*