International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 15 (2015) pp 35282-35286


# Application for sharing movie information based on open API and GCM

**Min-Ji Chu, Yun-Geol Jeong, Jaejoon Kim***
School of Computer and Communication, Daegu University
201 Daegudaero Jillyang, Gyeongsan, Gyeongbuk, 712-714 Korea
*jjkimisu@daegu.ac.kr (Corresponding author)

**Abstract**—Due to globalization, we often see foreign movies alongside domestic movies. As the choices become more diverse, people can have a difficult time choosing a movie to watch. People tend to choose a movie by its title rather than its genre or plot. A person could be unsatisfied with a movie choice because it was not what they expected it to be. To avoid such a situation, we made an application using Open API and GCM (Google Cloud Messaging). To use Open API, the user receives a project ID and API key. The ID of the mobile phone is sent to the GCM server and stored until the user acts upon it. Information related to the movie is then parsed and shown on the screen. If the information is to the user's liking, the application can also send the information to other people. By sharing information with others, the user can connect with them while expanding their knowledge about movies.

**Keyword**-Open API, Google cloud messaging, Client and server system, Information sharing

## 1. Introduction

The development of the internet has allowed us to share information with other people all across the world. There is a huge amount of interaction in areas of politics, economics, and culture. Watching foreign movies is one way we connect with foreign cultures. The number of imported movies available has greatly increased.

People tend to watch movies based on other people's recommendations or ratings. However, people may not know the genre or plot of the movie they watch. Movies selected in this manner may be unsatisfying due to differences in personal preferences. To avoid this, we need an application that can give information about a movie and send it to other people[1, 2]. Also, these days, there are many sequels to previous movies. If a person has not watched the previous movies, it may be difficult to understand the new movie. Our application can provide a simple summary so that the user can understand and enjoy the movie better. Also, the application can be useful for everybody because it can share movies that the user enjoyed or it can receive other movies that other people enjoyed.

This paper introduces an application that can search and share information on movies. The application is based on Android and uses Open API to retrieve information[3, 4, 5]. Many government agencies and companies are using Open API to provide information to the general public without restrictions. Once the user receives the company's key, the user can use Open API and retrieve information through parsing. While the user may be able to look up movies by accessing the Web, the user cannot send this information effectively. Our application shows the information using Open API and uses GCM (Google Cloud Messaging)[6] so that the user can share information about a movie with others by a simple push alarm.

## 2. Related works

### 2.1 Open API

An Open API (Application Programming Interface) is an open web service API [7]. It provides functions that can be difficult to implement, which makes it easier to build a system and provide a large amount of content. For example, if a search API is used, then one can build an application on search functions and provide images on personal blogs or forums to visitors. With the Open API, users can implement their own functions and become a developer. Also, we can reduce costs by using various services that Open API provides. Currently,the domestic Naver [8] and Daum [9]provide their own API to users. In this paper, we used the Open API of NAVER movies to develop our movie application.

Open API alone is not sufficient to implement an application. We need an XML parser to use the Open API. XML(eXtensible Markup Language)[10]can transform data into elements and values in any form that the user needs. Using an XML parser, we can retrieve data from the Open API and then implement it in Android to receive information on searched movies.

### 2.2 GCM (Google Cloud Messaging)

GCM allows the user to share information on movies with other people by sending push alarms [11, 12]. GCM is an imbedded push service provided by Google. GCM requires six items: the PROJECT ID, API KEY, Application ID, Registration ID, Google User Account, and server. We can use these interconnect an application and the GCM server to enable push services. Also, the developer can activate push services at any point.

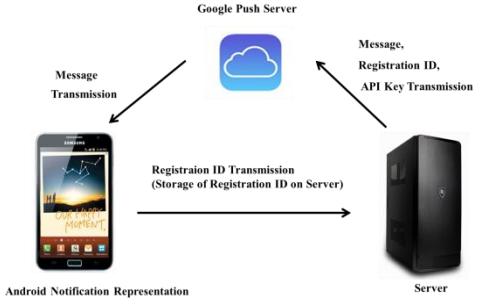



Fig. 1. GCM implementation scheme

Figure 1 illustrates the implementation of the GCM. The user saves the Registration ID of the phone to the server and then receives an API Key. Through the delivered API Key, the GCM sends a text to the user's phone. The API Key is only usable for a single project, since the developer may be working on other projects. An API Key is needed for each application and is allotted a GCM server. Through this process,a developer can make different GCM servers for different applications.

## 3. Proposed implementation method

This paper shows the implementation of an application that can provide information on various movies and use the push messaging system to share the information with various people. It improves upon simply searching for movies by sharingthe information found. Figure 2 shows a flow chart of the application's implementation.

### 3.1 Information search using Open API

We issuea key to obtain movie information from the Open API. Figure 3 shows the key issued from NAVER to use its Open API. This key is essential for using the data from the Open API.

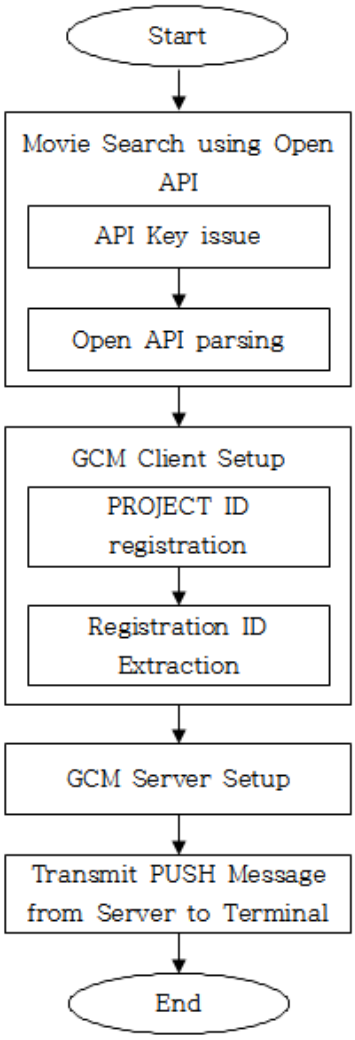



Fig. 2. Implementation flow chart

| Issued Date | Issued Information | Used Query | Management |
|---|---|---|---|
| 2015.04.25 | Identifier cinemaworld<br>Issued Key a7789234a18f1d13354073b6e50b75e1 | 0/25000 | Edit / Delete |

Fig. 3. Example of issued Open API Key from Naver

Figure 4 shows the objects created to obtain the data. The API key shown in Figure 3 is stored within "String key". Only one key may be issued per person, and as long as the developer does not share the key, the API key is unique. Using the information from the issued key allows us to use various API information.

Next,"xmlPullParser"obtains the data stored as XML. Parsing is done to convert the Open API into a form usable by Android. The xmlPullParser connects the XML information with the current application. We created a String function "getxmlData"and then input the queryURL value as shown in Figure 5. The API information is changed based on the URL and uses the search function. Also, we must set a request variable to indicate what information to obtain after setting the URL address. The request variable varies depending on the API used.

```
public class SecondActivity extends Activity {

    EditText edit;
    TextView text;
    ListView list;
    XmlPullParser xpp;
    String key="a7789234a18f1d13354073b6e50b75e1";
    String data;
    List<MovieInfo> movieInfoList;
    CustomListViewAdapter adapter;
```

Fig. 4.An object definition for displaying

```
String getXmlData(){
    StringBuffer buffer=new StringBuffer();
    String str= edit.getText().toString();
    String movie = URLEncoder.encode(str);
    String queryUrl="http://openapi.naver.com/search"
            +"?key="+key
            +"&target=movie"
            +"&query="+movie
            +"&display=10"
            +"&start=1";
```

Fig. 5.Inserting requested variables

The key is essential in applying the API Key stored as String Key in Figure 4. "Target = Movie" in Figure 5 indicates that the code will search for a movie. "query = +Movie" helps the screen bring up movie information by applying the Korean information in UTF-8 form. Lastly, the display and start variables define the number of movies listed and where to start the display. Figures 7 and 8 show the source code to show the movie information as text.

```
if(tag.equals("item")) {
    movieInfo = new MovieInfo();
}
else if(tag.equals("title")){
    buffer.append("검색제목 : ");
    xpp.next();
    movieInfo.setTitle(xpp.getText().
            replaceAll("<b>", "").replaceAll("</b>", ""));
    buffer.append(xpp.getText());
    buffer.append("\n");
}
```

Fig. 6.Removing a tag variable on a title

```
else if(tag.equals("image")){
    buffer.append("이미지 :");
    xpp.next();
    movieInfo.setImage(xpp.getText());
    buffer.append(xpp.getText());
    buffer.append("\n");
}

else if(tag.equals("director")){
    buffer.append("감독 :");
    xpp.next();
    movieInfo.setDirector(xpp.getText())
    buffer.append(xpp.getText());
    buffer.append("\n");
}
```

Fig. 7.Defining movie information on the display

Table 1 describes the request variables related to movie searching. Many functions can be applied if we connect to the API address indicated. However, Table 1 shows only a portion of the available request variables.

Table 1.Definition of useful requested variables

| VARIABLE | VALUE | DESCRIPTION |
|---|---|---|
| key | string | Write key string given through registration for use. |
| target | string | You must designate this to use the service. |
| query | string | These are questions that you want to search and UTF-8 encoding. |
| genre | string | This means the code of the genre that you want to search. |
| | | 1:drama 2:fantasy |
| | | 3:western 4:horror |
| | | 5:romance 6:adventure |
| | | 7:thriller 8:noir |
| | | 9:cult 10:documentary |
| | | 11:comedy 12:family |
| | | 13:mystery 14:war |
| | | 15:animation 16:crime |
| | | 17:musical 18:sci-fi |
| | | 19:action 20:martial arts |
| | | 21:erotic 22:suspence |
| | | 23:epic 24:black comedy |
| | | 25:experimental 26: cartoon |
| | | 27: music 28: poster of parody |

When searching for a movie, the search typically returns the movie title, director, and reviews. It can be difficult to figure out what the genre or plot is. Figure 8 shows the source code for obtaining particular details related to a movie being searched. The related information is brought to the phone in URL form after being linked. Also, the web view functions are automatically applied, so the information is shown in web view form on the screen.

```
webView = (WebView)findViewBy
webView.getSettings().setJava
webView.loadUrl(url);
webView.setWebViewClient(new
```

Fig. 8. Calling detailed movie information

### 3.2 GCM client

The PROJECT ID and API KEY needed to build the GCM client can be created on the Google homepage. Figures 9 and 10 show the PROJECT KEY and API KEY applied to the GCM for this application. Figure 11 shows the Registration ID extracted by the GCM client embedded in the application. The ID is transferred to the GCM server and awaits the push message.

| Project name | Project ID |
|---|---|
| API Project | api-project-175312085830 |
| CINEMA | decent-enjoy-89605 |
| TEST | blissful-link-89914 |

Fig. 9. An example of PROJECT ID for making a project

| Server application key | |
|---|---|
| API Key | AIzaSy08zSy59rVLxGtKvshzGTsSWZdffa00nCo |
| IP | All IPs allowed |
| Activation date | 2015.3.27. 4:18:00 PM |
| Activation standard | a891022@gmail.com |

Fig. 10. An example of API KEY for server application key

```
등록ID:APA91bExPquRd-Tyj8NozHiW0zTkfBwS6K6wPpnQ8X4U38Rh_zrNXcEfzLziLTbHa
VVpci0wdJqws7Munmndz_MFeAOGcCJxust_QZdEN3AdxCI1Zz8xpT4WNZ040LEte3nKgGq
rGytJ
```

Fig. 11.Extracting a Registration ID

The Registration ID is saved until the user needs to use it again. This process is carried out once the application is run and the checkbox is clicked.

### 3.3 GCM server and message transmission

The GCM client we have implemented thus far cannot send messages. We need a server that can receive messages andpass them on to the user's terminal. To connect the GCM client, we must use the API Key and the extracted Registration ID to interlock the server and client[13]. While managing the application, the Registration ID may change periodically. Therefore, we must confirm the ID with Log cat and add in String regId to set the path correctly.

```
public static void main(String[] args) {
    Sender sender = new Sender("AIzaSyD8zSy59rVLxGtKvshzGTs8WZdffaOOnCo");
                                //GCM API KEY

    String regId = "APA91bHY2FJcRlDLuJepnc9-E4IMQ7NvmOXdejEvXef41bSigGAwNjaA4Yu2AER-"
            + "jOIUE3QaRu4yX6nMwv5FvejjAJ1kkTf3QmGL0G2pCdxAZiCvm_"
            + "DvEd2AS0c2XvHi3EzLjxTdHZ0nCsQBkOXF6uzP7noCM5FbaQ";
                                //Registration ID
```

Fig. 12.Linkage for API Key, Registration ID

As illustrated in Figure 12, for a user to send a message using the GCM server, the message is defined as String[] args. The contents of the message can be shown in the “onMessage” section shown in Figure 13.

```
protected void onMessage(Context arg0, Intent arg1) {
    String msg = arg1.getStringExtra("msg");
    Log.e("getmessage", "getmessage:" + msg);
    generateNotification(arg0,msg);
```

Fig. 13.Setup for receiving a message

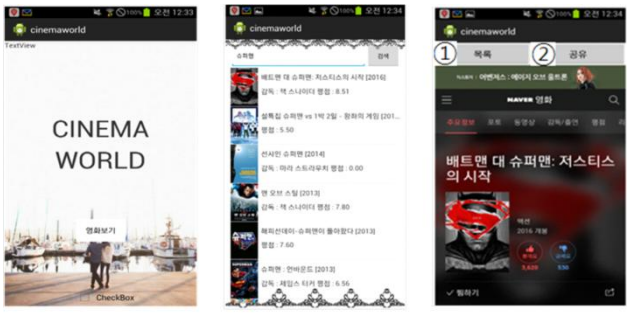
Figure 14 shows the source code for creating the list and share buttons on the Webview window. When the share button is pressed, the movie information is sent as a PUSH message to those who have the application installed.

```
Intent intent = getIntent();
url = intent.getStringExtra("url");

Button listButton = (Button)findViewById(R.id.goListBtn);
Button shareButton = (Button)findViewById(R.id.shareBtn);

listButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        finish();
```

Fig. 14.Creating "list" and "sharing" buttons on Web view window

## 4. Results

Once we start up the application, the screen will change to the view in Figure 15(a), with a CheckBox at the bottom of the screen. If the user checks this box, the Registration ID will be extracted by the GCM client imbedded in the application. This ID will be sent to the GCM server to await the push message, and when the user clicks "watch movie", a search engine will appear as in Figure 15(b). When the user types the title of a movie, information about that movie will be shown on the screen through the Open API parsed within the application.




(a) (b) ( c )

Fig. 15.Implementation results: (a) starting application screen, (b) searching movie list, (c) screen shot on web view.

In the screen shown in Figure 15(c)7, if the user presses the ① button, the user will be sent back to the list. If the user presses the ② button, the user can send the searched information to other people. The user can choose from their contacts to send a push message. Then, through Open API parsing, the movie information will be sent to the GCM server and then transferred automatically to contacts that have the application installed. After this process, the contact can click on the push message and read the movie information sent.

## 5. Conclusions

In this paper, we introduced a new movie application that can search and display movies of interest to the user. The API information is updated automatically when new movies are released, and this information can be shared between people. Through this process, the user can communicate with other people and receive much information.

An XML parser was used to interconnect the API and the application. By using an XML parser, we can interlock the Open API and the application and implement many functions that may be too difficult for beginners. The information used in the application is displayed on the application display based on the XML parser information. This means that depending on the developer, different types of Open APIs can be applied to develop various applications. Also, by using Open API, the developer can become familiarized with the technology by analyzing many different application source codes.

By using the theater ticket combined network and the Naver Movie Open API, developers are experiencing indirect marketing. GCM was developed by Google and has helped by providing the easy push message system. By using Google's GCM, many developers using Android can experience GCM easily, and this helps in maintaining user numbers and maintenance.

Future improvements can be suggested for our application. We can find a method to manage the Registration ID from each user more efficiently. We can also include popular movies of each decade as possible suggestions. We believe this will help different generations communicate more freely with each other. Lastly, if we can incorporate different arts such as music, opera, and musicals, we may be able to evolve our application as a method for users to experience and understand art and culture from different countries.

## References

[1] Shibahara, H.,Yamanouchi, M.,Sunahara, H.,2012, "A Proposal of the System for Automatic Sharing Information of Movements in Growing Vegetables,"Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on, Izmir, Turkey, pp. 394-398.

[2] Halder, S.,Samiullah, M.,Sarkar, A.M.J.,Young-Koo Lee, 2012, "Movie swarm: Information mining technique for movie recommendation system,"Electrical & Computer Engineering (ICECE), 2012 7th International Conference on, Dhaka, Bangladesh, pp. 462-465.

[3] Li, Zhang,Stover, C.,Lins, A.,Buckley, C.,Mohapatra, P., 2014, "Characterizing Mobile Open APIs in smartphone apps," Proc. of Networking Conference, 2014 IFIP, Trondheim, Norway, pp. 1-9.

[4] Michael Burton, 201, Android App Development For Dummies, For Dummies.

[5] Nishanthini, S.,Abinaya, M., Malathi, S., "Smart Video Surveillance system and alert with image capturing using android smart phones," 2014,Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on, Nagercoil,India, pp. 1714-1722.

[6] J. Hansen, T.M. Gronli, and G. Ghinea, 2012, "Towards cloud to device push messaging on android: Technologies, possibilities and challenges," Intl J. of Communications, Network and System Sciences, 5(12), pp. 839-849.

[7] Yoo-mi, P., Young-il, C., Sang-Ha, K., 2006, "Presence-based call reservation service using open API on the

Web-service architecture in broadband converged network,"The 8th International Conference on Advanced Communication Technology, Phoenix Park, Korea, pp. 270-273.

[8] http://movie.naver.com

[9] http://movie.daum.net

[10] Bill Evjen, Kent Sharkey, ThiruThangarathinam, Michael Kay, Alessandro Vernet, Sam Ferguson, 2007, Professional XML, Wrox.

[11] Penghui Li, Yan Chen,Taoying Li,Renyuan Wang,Junxiong Sun, 2013, "Implementation of Cloud Messaging System Based on GCM Service,"Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on, Shiyan,Hubei,China, pp. 1509-1512.

[12] Yilmaz, Y.S., Aydin, B.I., Demirbas, M., 2014, "Google cloud messaging (GCM): An evaluation,"Global Communications Conference (GLOBECOM), 2014 IEEE, Austin, TX, USA, pp. 2807-2812.

[13] Ing-Ray Chen,Ngoc AnhPhan, I-Ling Yen, 2002, "Algorithms for supporting disconnected write operations for wireless Web access in mobile client-server environments,"Mobile Computing, IEEE Transactions on, 1(1), pp. 46-58.