

“Dynamic Push Message” 프로젝트

안드로이드 푸시 샘플

3100-03

Ver 1.0

JOYTUNE 프로젝트팀

Copyright © JOYTUNE

JOYTUNE의 사전 승인 없이 본 내용의 전부 또는 일부에 대한 복사, 배포, 사용을 금합니다.

목 차

1. DPUSH 를 이용한 안드로이드 푸시 샘플 만들기	4
1.1 시작 전에 필요한 작업들.....	4
1.2 샘플 어플의 내용	4
2. ANDROID APP(오늘의명언) 중요 설명	6
2.1 안드로이드 Service	6
2.2 안드로이드 Receiver.....	10
2.3 안드로이드 Manifest.....	11
3. SERVER API 연동	12
3.1 시작 전에 필요한 작업들.....	12
3.2 오늘의 명언 보내기	12

1. DPUSH 를 이용한 안드로이드 푸시 샘플 만들기

1.1 시작 전에 필요한 작업들

먼저 간단한 가입을 통해 PRODUCTKEY 를 받습니다.

[DPUSH PRODUCTKEY 받기](#)

기본적인 안드로이드 환경에 대한 설정은 사이트를 확인하세요.

[사이트 Document 확인](#)

생성한 Android 프로젝트의 libs 폴더에 JavaLibrary 를 첨부합니다.

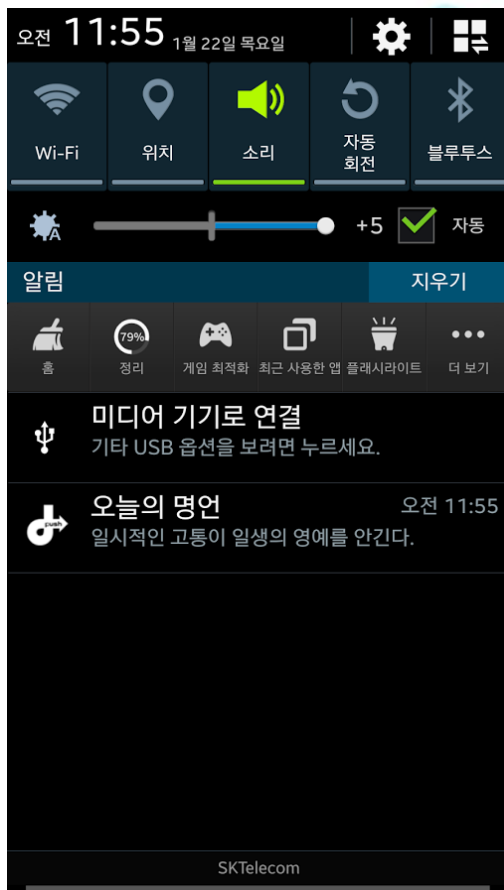
필요한 라이브러리는 "Java-WebSocket-1.3.0.jar" 와 "dpush-v0.x.jar" 입니다

[사이트 Download](#)

1.2 샘플 어플의 내용

DPUSH 를 통해 최초 작업된 안드로이드 앱은 "오늘의 명언"이라는 어플입니다.

이 어플은 간단히 서버에서 전송되는 메시지를 Notification 영역에 표시하고, 선택시 노출하는 기능만을 구현합니다.



서버에서 특정 메시지를 보낼 경우 안드로이드 폰에 상단 Notification 영역에 메시지가 왔음을 알립니다.



해당 어플은 스토어에서 DPUSH 로 검색하면 받을 수 있으며 스토어 주소는 다음과 같습니다.
<https://play.google.com/store/apps/details?id=kr.co.joytune.dpush>

2. Android APP(오늘의명언) 중요 설명

2.1 안드로이드 Service

전체 소스를 참고하시고 기본적으로 필요한 중요한 내용만을 설명합니다.

필요한 기능으로 정의된 내용은 아래와 같습니다.

1. 서버에서 메시지를 전송받을 수 있는 Client
2. 메시지 수신시 Notification 영역에 받은 메시지를 알림
3. 알림 확인시 화면에 해당 내용을 노출
4. 어플 종료 또는 스마트폰 시작시 자동으로 어플을 실행시킬수 있는 기능
5. 스마트폰이 Sleep Mode 일때도 메시지를 수신 할 수 있는 기능

우선 서버와의 접속을 위한 dpclient 는 서비스영역에 정의합니다.

```
private DPClient dpm;

// GRP_ID(그룹명) 와 ACT_ID(액션명) 는 property에서 읽어서 가져옵니다.

private void connect() {
    // 기 정의된 dpclient 가 없으면 생성한다
    if (this.client == null) {
        this.client = new DPClient('발급받은 PRODUCTKEY') {

            public void onConnected() {
                Log.d(TAG, "onconnect.....");
            }

            public void onDisconnected() {
                Log.d(TAG, "ondisconnect.....");
                // 종료시 재접속 요청(30초후)
                mHandler.sendEmptyMessageDelayed(0x8999, 30 * 1000);
            }

            public void onHeartbeat() {
                mHandler.sendEmptyMessage(0x7000);
            }
        };
    }

    // 서버와의 접속을 시도한다
    if (!client.isConnected()) {
        client.connect();
    }

    // 오픈된 그룹이면 정의된 ActionID 에 해당하는 callback만을 재정의한다
    if (client.isOpened(GRP_ID)) {
        Log.d(TAG, "is opened...");
        gi = client.getOpenedGroup(GRP_ID);
        gi.onReceive(ACT_ID, action(mHandler));
    }
    // 그룹을 오픈한다
    else {
```

```

        Log.d(TAG, "is not opened...");
        gi = client.openGroup(GRP_ID);
        gi.onReceive(ACT_ID, action(mHandler));
    }
}

```

메시지 도착시 Notification 영역에 노출하기 위한 기능을 추가합니다

```

/**
 * 서버에서 메시지가 수신되었을때의 callback을 정의한다
 * @param handler
 * @return
 */
private static Callback action(final Handler handler) {
    return new Callback() {
        @Override
        public void call(Object... args) {
            String message = (String) args[0];
            if (message != null && !"".equals(message)) {
                handler.sendMessage(handler.obtainMessage(ACTION_SHOW_NOTIFICATION, message));
            }
        }
    };
}

public static final int NOTIFICATION_ID = 0x1234;
private static final int ACTION_SHOW_NOTIFICATION = 0x5001;

/**
 * 알림영역에 Notification 을 노출한다
 * @param title
 * @param text
 */
private void showNotification(CharSequence title, CharSequence text) {

    // 알림 도착시 실행하고자 하는 작업(Intent)을 설정
    Intent intent = new Intent(DPushService.this.getApplicationContext(), DPushAlarm.class);
    // - 필요한 parameter 설정
    intent.putExtra("NM", title.toString());
    intent.putExtra("MG", text.toString());
    // - 기존에 호출된 Intent 가 있으면 제거한다
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    // - 새로운 Activity를 생성하도록 한다
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

    // PendingIntent 초기화
    PendingIntent contentIntent = PendingIntent.getActivity(getApplicationContext(), 0, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);

    // 메시지 도착을 알리기 위해 Notification을 사용한다
    // Notification Builder 객체생성
    NotificationCompat.Builder notibuilder = new
    NotificationCompat.Builder(getApplicationContext());
    // - 상태표시줄에 나타낼 아이콘
    notibuilder.setSmallIcon(R.drawable.dpush_logo);
    // - Notification 을 표시할 시각

```

```

        notibuilder.setWhen(System.currentTimeMillis());
        // - 상태표시줄에 나타낼 제목
        notibuilder.setTitle(title);
        // - 상태표시줄에 나타낼 내용
        notibuilder.setText(Html.fromHtml(text.toString()));
        // - 알림항목을 눌렀을때 해당 알림을 자동으로 해제시키도록 설정
        notibuilder.setAutoCancel(true);
        // - 알림항목을 눌렀을때 실행할 작업(Intent)
        notibuilder.setContentIntent(contentIntent);
        // - builder 를 통해 Notification 생성
        Notification notification = notibuilder.build();

        // - 알람도착시 소리
        notification.defaults |= Notification.DEFAULT_SOUND;

        // NotificationManager 객체 호출
        NotificationManager mNM = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        // - 고유아이디를가지는 notification 을 표시
        mNM.notify(DPUSH_NOTIFICATION_ID, notification);
    }

```

시스템에 의해 서비스가 강제 종료 되었을때 앱을 재시작 시키기 위한 Alarm 을 등록/해제 하는 기능을 추가합니다

```

@Override
public void onCreate() {
    super.onCreate();
    // 등록된 알람은 제거
    unregisterRestartAlarm();
}

@Override
public void onDestroy() {
    ...
    // 알람 등록
    registerRestartAlarm();

    super.onDestroy();
}

/**
 * 서비스가 시스템에 의해서 또는 강제로 종료되었을 때 호출되어
 * 알람을 등록해서 15초 후에 서비스가 실행되도록 한다.
 */
private void registerRestartAlarm() {
    Intent intent = new Intent(DPushService.this, DPushReceiver.class);
    intent.setAction(DPushReceiver.ACTION_DPUSH_RESTART);
    PendingIntent sender = PendingIntent.getBroadcast(DPushService.this, 0, intent, 0);

    long firstTime = SystemClock.elapsedRealtime();
    firstTime += 15 * 1000; // 15초 후에 알람이벤트 발생

    AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
    am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, firstTime, 60 * 1000, sender);
}

```



```

/**
 * 기존 등록되어있는 알람을 해제한다.
 */
private void unregisterRestartAlarm() {
    Intent intent = new Intent(DPushService.this, DPushReceiver.class);
    intent.setAction(DPushReceiver.ACTION_DPUSH_RESTART);
    PendingIntent sender = PendingIntent.getBroadcast(DPushService.this, 0, intent, 0);

    AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
    am.cancel(sender);
}

```

스마트폰이 SleepMode 로 진입했을때 서버와의 연결을 유지하기 위해 PowerManager 의 WakeLock 을 사용합니다. 이 코드는 샘플로 실 구현시는 battery 사용에 유의하셔야 합니다

```

private BroadcastReceiver screenoff;

@Override
public void onCreate() {

    super.onCreate();
    ...

    screenoff = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            // 스마트폰 화면이 켜졌을때
            if (Intent.ACTION_SCREEN_ON.equals(intent.getAction())) {
                WakeLocker.releaseWakeLock();
            }
            // 스마트폰 화면이 꺼졌을때
            else if (Intent.ACTION_SCREEN_OFF.equals(intent.getAction())) {
                WakeLocker.acquireWakeLock(context);
            }
        }
    };
    // Screen Off Receiver 를 등록
    IntentFilter offfilter = new IntentFilter(Intent.ACTION_SCREEN_OFF);
    offfilter.addAction(Intent.ACTION_SCREEN_ON);
    registerReceiver(screenoff, offfilter);
    ...
}

@Override
public void onDestroy() {
    ...
    // Screen Off Receiver 를 해제
    unregisterReceiver(screenoff);

    super.onDestroy();
}

static class WakeLocker {

    private static PowerManager.WakeLock sCpuWakeLock;

}

```

```

    * PowerManager.WakeLock 시작
    * @param context
    */
    static void acquireWakeLock(Context context) {
        PowerManager pm = (PowerManager) context.getSystemService(Context.POWER_SERVICE);
        if (!pm.isScreenOn()) {
            sCpuWakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
context.getClass().getSimpleName());
            sCpuWakeLock.acquire();
        }
    }

    /**
    * PowerManager.WakeLock 해제
    */
    static void releaseWakeLock() {
        if (sCpuWakeLock != null && sCpuWakeLock.isHeld()) {
            sCpuWakeLock.release();
        }
    }
}

```

2.2 안드로이드 Receiver

위와 같은 서비스를 생성한 이후, 부팅완료, 네트워크 변경등의 상황 발생시 서비스를 재시작하기 위한 Receiver 를 생성합니다

```

public class DPushReceiver extends BroadcastReceiver {

    private static final String TAG = DPushReceiver.class.getSimpleName();

    public static final String ACTION_DPUSH_RESTART = "kr.co.joytune.dpush.restart";

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        /* 서비스 재시작 */
        if (action.equals(DPushReceiver.ACTION_DPUSH_RESTART)) {
            Log.d(TAG, "Service dead, but resurrection");
            Intent i = new Intent(context, DPushService.class);
            context.startService(i);
        }

        /* 스마트폰 부팅완료 */
        if (action.equals(Intent.ACTION_BOOT_COMPLETED)) {
            Log.d(TAG, "ACTION_BOOT_COMPLETED");
            Intent i = new Intent(context, DPushService.class);
            context.startService(i);
        }

        /* 네트워크 변경 */
        if (action.equals(ConnectivityManager.CONNECTIVITY_ACTION)) {
            try {
                ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);

```

```

        NetworkInfo activeNetInfo = connectivityManager.getActiveNetworkInfo();
        if (activeNetInfo != null) {
            Intent i = new Intent(context, DPushService.class);
            context.startService(i);
        }
    }
    catch (Exception e) {
    }
}
}
}

```

2.3 안드로이드 Manifest

이제 위와 같은 기능을 사용하기 위해 Android Manifest file 을 수정합니다

생성된 위 Service 와 Receiver 를 manifest 에 추가합니다.

```

<service
    android:name=".DPushService"
    android:enabled="true"
    android:process=":remote" >
</service>

<receiver
    android:name=".DPushReceiver"
    android:enabled="true"
    android:exported="false"
    android:label="RestartService"
    android:process=":remote" >
    <intent-filter>
        <action android:name="kr.co.joytune.dpush.restart" />
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>

```

또한, 정의된 안드로이드 기능을 사용하기 위한 권한정보를 추가합니다

```

<!-- 인터넷 사용 -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- powermanager.wakelock 사용 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
<!-- 네트워크 상태 확인 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- 폰정보 확인 : Context.TELEPHONY_SERVICE -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- 부팅완료 확인 -->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

```

3. Server API 연동

3.1 시작 전에 필요한 작업들

Server API 의 경우 PHP 를 기본으로 설명합니다.

서버에 대한 부분은 BUSINESS SERVER API 에서 'DPServer' 의 선언'과 'send'를 참고하시면 됩니다.

[사이트 Document 확인](#)

Download 의 Server API(Business Server)에서 PHP 를 다운받아서 설치합니다.

[사이트 Download](#)

3.2 오늘의 명언 보내기

서버 API 에서의 작업은 간단합니다.

발급받은 PRODUCTKEY 를 넣고 send 를 호출하면 됩니다.

```
<?php
include "dpmessage.php";
$dpmmessage = new DPServer("발급받은 PRODUCTKEY");
$dpmmessage->send('twiase','tnoti',"A great secret of success is to go through life as
a man who never gets used up.<br>성공의 커다란 비결은 결코 지치지 않는 인간으로 인생을
살아가는 것이다.<br>알버트 슈바이처");
?>
```