

“Dynamic Push Message” 프로젝트

안드로이드 채팅 샘플

3100-04

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 채팅 샘플 만들기	4
1.1 시작 전에 필요한 작업들.....	4
1.2 샘플 어플의 내용	4
2. ANDROID 채팅 앱 중요 설명	6
2.1 앱 구조	6
2.2 메인 액티비티 (DPushActivity)	6
2.2.1 Product key 설정 및 변경	6
2.2.2 Login 정보 설정 및 변경	7
2.2.3 메시지 수신 및 화면 출력	8
2.2.4 사용자 메시지를 서버로 전송	9
2.2.5 다른 사용자의 로그인/로그아웃/정보 변경 내용을 공지	10
2.2.6 채팅방에 접속된 사용자들의 목록을 출력	12
2.2.7 스마트폰이 Sleep Mode 일 경우에도 메시지 수신	13
2.3 Product key 설정 액티비티 (DPushProdKeyCheck).....	13
2.4 로그인 설정 액티비티 (DPushLogin)	14
2.5 안드로이드 Manifest.....	16

1. DPUSH 를 이용한 안드로이드 채팅 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](#)

가입 후에 채팅을 위해서는 두가지 세팅이 필요합니다.

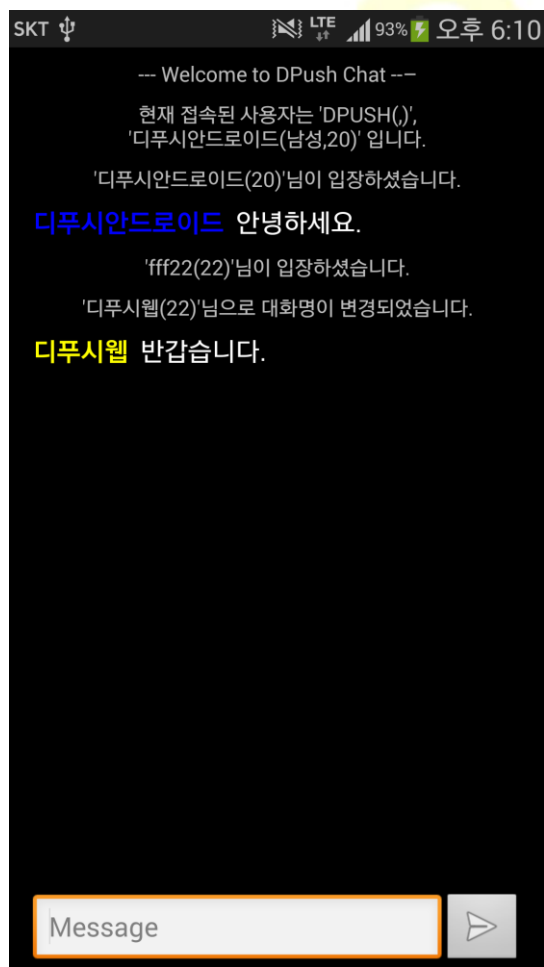
"Customer Info 사용 체크" 와 "클라이언트 메시지 체크" 입니다.

[설정방법 가이드 보기](#)

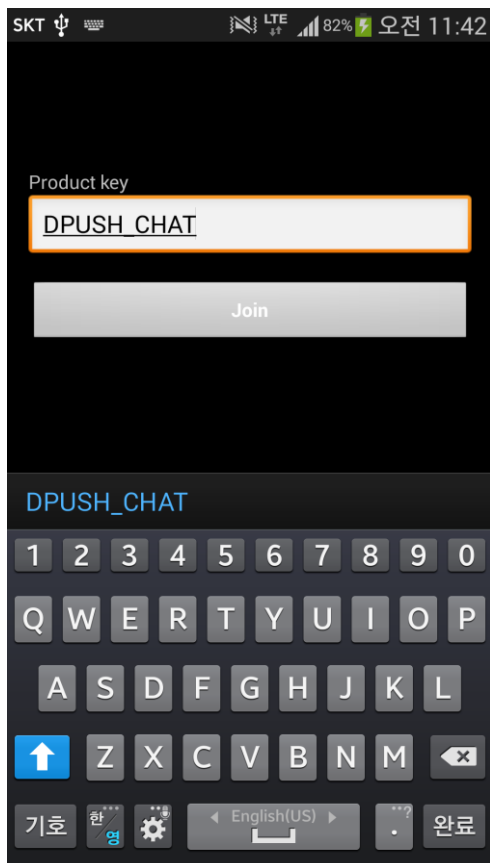
1.2 샘플 어플의 내용

본 문서는 DPUSH 기반 안드로이드 채팅앱에 대해 설명합니다.

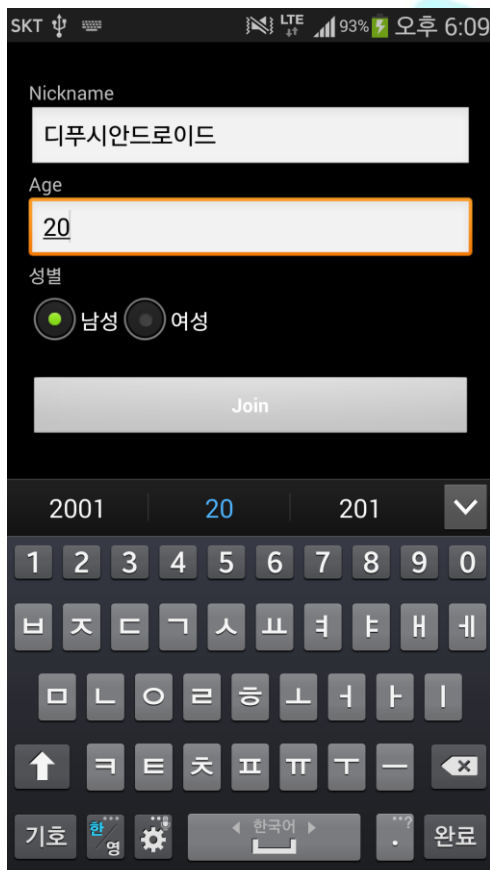
본 채팅앱은 사용자 로그인 후, 텍스트 메시지를 교환하는 기본적인 채팅 기능을 지원합니다. 또한, Product key 변경을 통해 채팅방 변경이 가능합니다.



“디푸시안드로이드” 라는 아이디로 로그인한 상태에서 메시지를 주고 받는 기능을 제공합니다.



Product key 을 설정/변경하는 기능을 제공합니다.



로그인 정보(아이디, 나이, 성별)를 입력/변경하는 기능을 제공합니다.

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

2. Android 채팅 앱 중요 설명

2.1 앱 구조

전체 소스를 참고하시기 바라며, 기본적으로 필요한 중요한 내용만을 설명합니다.

채팅앱은 아래와 같이 3 개의 Activity 들로 구성됩니다.

Activity 이름	내용
DPushActivity	채팅의 메인 Activity 로서 메시지를 입력받고 수신된 메시지를 보여주는 기능을 수행합니다. 실제 화면 구성 및 메시지 통신은 DPushMainFragment 클래스에서 수행합니다.
DPushProdKeyCheck	Product key 를 설정/변경하는 기능을 제공합니다. 설정된 key 는 다음번 앱 수행시에도 그대로 적용됩니다.
DPushLogin	로그인 정보를 받아 서버 접속을 기능을 제공합니다.

2.2 메인 액티비티 (DPushActivity)

DPushActivity 는 크게 아래와 같은 기능들을 수행합니다.

번호	기능	수행 클래스
1	Product key 설정/변경 및 서버 접속	DPushMainFragment
2	Login 정보 설정 및 변경	DPushMainFragment
3	메시지 수신 및 화면 출력	DPushMainFragment
4	사용자 메시지를 서버로 전송	DPushMainFragment
5	다른 사용자의 로그인/로그아웃/정보 변경 내용을 공지	DPushMainFragment
6	채팅방에 접속된 사용자들의 목록을 출력	DPushMainFragment
7	스마트폰이 Sleep Mode 일 경우에도 메시지 수신	DPushActivity

2.2.1 Product key 설정 및 변경

앱 수행시 이미 저장된 키가 있으면 그것을 사용하고 그렇지 않으면 Default 값인 "LIVEDEMOPUSH" 값을 사용합니다.

<pre> public void onCreate(Bundle savedInstanceState) { // 이전에 설정된 Product key를 찾는다. SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE); mProdKey = sharedPref.getString("PRODUCT_KEY", "LIVEDEMOPUSH"); } </pre>
--

설정된 Product key 를 가지고 서버에 접속합니다.

```

private void connectDPClient() {
    if (client != null) {
        client.close();
        client = null;
        gi = null;
    }

    mProdKeyChecked = false;

    DPOptions dpopts = new DPOptions();
    dpopts.setResultcallback(actionOnConnect(mHandler));

    client = new DPClient(mProdKey, dpopts) {

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

        public void onDisconnected() {
            Log.d(TAG, "onDisconnected.....");
            if (!client.isConnected()) connect();
        }
    };

    client.connect();
}

```

서버와 연결이 정상적으로 수행되면 다음번에 동일한 Product key 를 사용하기 위해 Activity 저장영역에 Key 를 저장합니다.

```

mHandler = new Handler() {
    public void handleMessage(android.os.Message msg) {
        switch (msg.what) {
            case ACTION_ON_CONNECT:
                b = msg.getData();
                String errMessage = b.getString("errmessage");
                if (errMessage.isEmpty()) {
                    // Product key를 store한다
                    SharedPreferences sharedPref =
                        getActivity().getPreferences(Context.MODE_PRIVATE);
                    SharedPreferences.Editor editor = sharedPref.edit();
                    editor.putString("PRODUCT_KEY", mProdKey);
                    editor.commit();

                    startSignIn();
                }
            }
        }
    };
};

```

2.2.2 Login 정보 설정 및 변경

Login 정보를 설정 혹은 변경 위해 DPushActivity 를 띄웁니다.

```

// Login 정보를 설정을 위해 Login 창을 띄운다.
private void startSignIn() {
    mUsername = "";
}

```

```

Intent intent = new Intent(getActivity(), DPushLogin.class);
intent.putExtra("requestcode", REQUEST_LOGIN);
startActivityForResult(intent, REQUEST_LOGIN);
}

// Login 정보 변경을 위해 Login 창을 띄운다.
private void startUpdateUserInfo() {
    Intent intent = new Intent(getActivity(), DPushLogin.class);
    intent.putExtra("username", mUsername);
    intent.putExtra("userage", "" + mUserage);
    intent.putExtra("usergender", "" + mUsergender);
    intent.putExtra("requestcode", REQUEST_UPDATE_USERINFO);
    startActivityForResult(intent, REQUEST_UPDATE_USERINFO);
}

```

로그인 창 Activity 인 DPushActivity 를 통해 받은 로그인 정보를 사용하여 Group 을 엽니다.

```

public void performOpenGroup() {
    . . . . .
    // 그룹을 엽니다
    else {
        GroupOptions gpopts = new GroupOptions();
        gpopts.setCustevent(true);
        gpopts.setSendevent(true);

        JSONObject cinfo = new JSONObject();
        try {
            cinfo.put("nickname", mUsername);
            cinfo.put("age", mUserage);
            if ("1".equals(mUsergender)) cinfo.put("gender", "M");
            else if ("2".equals(mUsergender)) cinfo.put("gender", "F");
            else cinfo.put("gender", "P");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        gpopts.setCustinfo(cinfo);

        gi = client.openGroup(GRP_ID, gpopts, null);
        // 메시지가 수신될때 수행되는 callback 함수 등록
        gi.onReceive(ACT_ID, action(mHandler));
        // Group 오픈 요청에 대한 결과를 처리하는 callback 함수 등록
        gi.setCallback(actionOnGroupOpened(mHandler));
        // 채팅방에 사용자가 들어올때 수행되는 callback 함수 등록
        gi.setOnUserIn(actionOnUserIn(mHandler));
        // 채팅방에서 사용자가 나갈때 수행되는 callback 함수 등록
        gi.setOnUserOut(actionOnUserOut(mHandler));
        // 채팅방내 사용자정보가 변경될때 수행되는 callback 함수 등록
        gi.setOnUserUpdated(actionOnUserUpdated(mHandler));
    }
}

```

2.2.3 메시지 수신 및 화면 출력

(2)에서 메시지가 수신될 때 호출되는 “action()” callback 함수를 통해 수신된 메시지를 화면에 출력합니다.

```
private static Callback action(final Handler handler) {
    return new Callback() {
        @Override
        public void call(Object... args) {
            String message = (String) args[0];
            JSONObject custInfo = (JSONObject) args[2];
            String nickname;
            try {
                nickname = custInfo.getString("nickname");
            }
            catch (JSONException e) {
                return;
            }

            Message msg = Message.obtain(null, ACTION_SHOW_TEXTMESSAGE, 0, 0);
            Bundle b = new Bundle();
            b.putString("nickname", nickname);
            b.putString("message", message);
            msg.setData(b);

            if (message != null) {
                handler.sendMessage(msg);
            }
        }
    };
}
```

화면 출력하는 부분은 아래와 같습니다.

```
public void onCreate(Bundle savedInstanceState) {
    .....
    case ACTION_SHOW_TEXTMESSAGE:
        b = msg.getData();
        addMessage(b.getString("nickname"), b.getString("message"));
        break;
    }
    .....
};
```

2.2.4 사용자 메시지를 서버로 전송

사용자가 메시지를 입력하고 전송 버튼을 누르면 입력된 메시지를 서버로 전송합니다.

```
private void attemptSend() {
    if (null == mUsername) return;
    mTyping = false;

    String message = mInputMessageView.getText().toString().trim();
    if (TextUtils.isEmpty(mInputMessageView.getText())) {
        mInputMessageView.requestFocus();
        return;
    }
    mInputMessageView.setText("");

    // 메시지를 전송한다.
```

```

        sendTextMessage(message);
    }

    public void sendTextMessage(String textMessage) {
        if (mProdKey.isEmpty()) {
            startGetProdKey("");
        }
        gi.send(ACT_ID, textMessage);
        return;
    }
}

```

2.2.5 다른 사용자의 로그인/로그아웃/정보 변경 내용을 공지

(2)에서 등록한 “actionOnUserIn()”, “actionOnUserOut()”, “actionOnUserUpdated()” callback 함수를 통해 다른 사용자의 로그인/로그아웃/정보변경 내용을 화면에 출력합니다.

```

private static Callback actionOnUserIn(final Handler handler) {
    return new Callback() {
        @Override
        public void call(Object... args) {

            JSONObject custInfo = (JSONObject) args[1];
            String nickname, age;
            try {
                nickname = custInfo.getString("nickname");
                age = custInfo.getString("age");
            }
            catch (JSONException e) {
                return;
            }

            Message msg = Message.obtain(null, ACTION_ON_USERIN, 0, 0);
            Bundle b = new Bundle();
            b.putString("nickname", nickname);
            b.putString("age", age);
            msg.setData(b);

            if (nickname != null && !nickname.isEmpty()) {
                handler.sendMessage(msg);
            }
        }
    };
}

private static Callback actionOnUserOut(final Handler handler) {
    return new Callback() {
        @Override
        public void call(Object... args) {

            JSONObject custInfo = (JSONObject) args[1];
            String nickname, age;
            try {
                nickname = custInfo.getString("nickname");
                age = custInfo.getString("age");
            }
            catch (JSONException e) {
                return;
            }
        }
    };
}

```

```

        Message msg = Message.obtain(null, ACTION_ON_USEROUT, 0, 0);
        Bundle b = new Bundle();
        b.putString("nickname", nickname);
        b.putString("age", age);
        msg.setData(b);

        if (nickname != null && !nickname.isEmpty()) {
            handler.sendMessage(msg);
        }
    }
};
}

private static Callback actionOnUserUpdated(final Handler handler) {
    return new Callback() {
        @Override
        public void call(Object... args) {

            JSONObject custInfo = (JSONObject) args[1];
            String nickname, age;
            try {
                nickname = custInfo.getString("nickname");
                age = custInfo.getString("age");
            }
            catch (JSONException e) {
                return;
            }

            Message msg = Message.obtain(null, ACTION_ON_USERUPDATED, 0, 0);
            Bundle b = new Bundle();
            b.putString("nickname", nickname);
            b.putString("age", age);
            msg.setData(b);

            if (nickname != null && !nickname.isEmpty()) {
                handler.sendMessage(msg);
            }
        }
    };
}
}

```

Callback 함수로부터 받은 메시지를 출력하는 부분은 아래와 같습니다.

```

public void onCreate(Bundle savedInstanceState) {
    .....
    case ACTION_ON_USERIN:
        b = msg.getData();
        addLog("'" + b.getString("nickname") + "(" + b.getString("age") +
        ")'님이 입장하셨습니다.");
        break;
    case ACTION_ON_USEROUT:
        b = msg.getData();
        addLog("'" + b.getString("nickname") + "(" + b.getString("age") +
        ")'님이 퇴장하셨습니다.");
        break;
    case ACTION_ON_USERUPDATED:
        b = msg.getData();
        addLog("'" + b.getString("nickname") + "(" + b.getString("age") +

```

```

        ")'님으로 대화명이 변경되었습니다.");
        break;
        .....
    };
};

```

2.2.6 채팅방에 접속된 사용자들의 목록을 출력

폰의 메뉴버튼을 통해 “Show userlist” 명령이 수행되면, 현재 접속된 사용자들의 목록을 출력합니다.

```

// 메뉴 버튼을 처리하는 함수
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    .....
    else if (id == R.id.action_show_userlist) showUserList();
    .....
}

// 현재 접속된 사용자들의 목록을 화면에 출력
private void showUserList() {
    JSONObject userList;
    userList = gi.getUserList();
    String userListResult = "";

    int i = 0;
    try {
        Iterator<String> keys = userList.keys();
        while (keys.hasNext()) {
            String key = keys.next();
            JSONObject custInfo = userList.getJSONObject(key);
            String genderResult = "";

            if (custInfo.has("gender")) {
                Log.d("DPushMainFragment", "gender" + custInfo.getString("gender"));
                genderResult = custInfo.getString("gender");
            }

            if ("M".equals(genderResult)) genderResult = "남성";
            else if ("F".equals(genderResult)) genderResult = "여성";
            else genderResult = "";

            userListResult += "'" + custInfo.getString("nickname") + "(" +
genderResult + "," + custInfo.getString("age") + ")'";
            if (i < userList.length() - 1) userListResult += ", ";
            i++;
        }
        addLog("현재 접속된 사용자는 " + userListResult + " 입니다.");
    }
    catch (JSONException e) {
        addLog("ERROR: 현재 접속된 사용자는 " + userListResult + " 입니다.");
    }
    return;
}

```

2.2.7 스마트폰이 Sleep Mode 일 경우에도 메시지 수신

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

```
PowerManager powerManager;
PowerManager.WakeLock wakeLock;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    powerManager = (PowerManager) getSystemService(POWER_SERVICE);
    wakeLock = powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
    "DPushChatWakelockTag");
    wakeLock.acquire();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    wakeLock.release();
}
```

2.3 Product key 설정 액티비티 (DPushProdKeyCheck)

Product key 를 입력 받는 액티비티이며, 메인(DPushMainFragment) Activity 로부터 이전 Product key 를 받고, 변경 key 를 전달합니다.

```
public class DPushProdKeyCheck extends Activity {

    private EditText mProdKeyView;
    private String mProdKey;
    private boolean mIsBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prodkeycheck);

        mProdKeyView = (EditText) findViewById(R.id.prodkey_input);

        Button prodKeyInButton = (Button) findViewById(R.id.prodkey_in_button);
        prodKeyInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                setProdKey();
            }
        });
        mProdKey = getIntent().getStringExtra("prodkey");
        mProdKeyView.setText(getIntent().getStringExtra("prodkey"));

        if (!getIntent().getStringExtra("errmessage").isEmpty())
            mProdKeyView.setError(getIntent().getStringExtra("errmessage"));
    }
}
```

```

private void setProdKey() {
    mProdKeyView.setError(null);
    String prodkey = mProdKeyView.getText().toString().trim();

    // 유저아이디가 적합한지를 체크한다.
    if (TextUtils.isEmpty(prodkey)) {
        mProdKeyView.setError(getString(R.string.error_field_required));
        mProdKeyView.requestFocus();
        return;
    }
    Intent intent = new Intent();
    intent.putExtra("prodkey", prodkey);
    setResult(RESULT_OK, intent);
    finish();
}
}

```

2.4 로그인 설정 액티비티 (DPushLogin)

로그인 정보를 설정/변경하는 입력 받는 Activity 이며, 메인(DPushMainFragment) Activity로부터 이전 로그인 정보를 받고, 변경된 로그인 정보를 전달합니다.

```

public class DPushLogin extends Activity {

    private EditText mUsernameView;
    private EditText mUserageView;
    private String mUsername;
    private int mUserage;
    private int mUsergender = 1; // 1: 남성(기본), 2: 여성
    private boolean mIsBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // 로그인 폼을 설정한다.
        mUsernameView = (EditText) findViewById(R.id.username_input);
        mUserageView = (EditText) findViewById(R.id.userage_input);

        Button signInButton = (Button) findViewById(R.id.sign_in_button);
        signInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });

        int requestCode = getIntent().getIntExtra("requestcode", -1);
        if (requestCode == DPushMainFragment.REQUEST_UPDATE_USERINFO) {
            mUsernameView.setText(getIntent().getStringExtra("username"));
            mUserageView.setText(getIntent().getStringExtra("userage"));

            String usergender = getIntent().getStringExtra("usergender");
            RadioButton rbMale = (RadioButton) findViewById(R.id.radio_male);
            RadioButton rbFemale = (RadioButton) findViewById(R.id.radio_female);

```

```
        if ("1".equals(usergender)) rbMale.setChecked(true);
        else if ("2".equals(usergender)) rbFemale.setChecked(true);
    }
}

public void onRadioButtonClicked(View view) {
    boolean checked = ((RadioButton) view).isChecked();

    switch (view.getId()) {
        case R.id.radio_male:
            if (checked) mUsergender = 1;
            break;
        case R.id.radio_female:
            if (checked) mUsergender = 2;
            break;
    }
}

private void attemptLogin() {
    mUsernameView.setError(null);
    String username = mUsernameView.getText().toString().trim();

    // 유저아이디가 적합한지를 체크한다.
    if (TextUtils.isEmpty(username)) {
        mUsernameView.setError(getString(R.string.error_field_required));
        mUsernameView.requestFocus();
        return;
    }

    String usage = mUsageView.getText().toString().trim();
    if (TextUtils.isEmpty(usage)) {
        mUsageView.setError(getString(R.string.error_field_required));
        mUsageView.requestFocus();
        return;
    }

    try {
        mUsage = Integer.parseInt(usage);
    }
    catch (Exception e) {
        mUsageView.setError(getString(R.string.error_age_number_required));
        mUsageView.requestFocus();
        return;
    }

    mUsername = username;
    Intent intent = new Intent();
    intent.putExtra("username", mUsername);
    intent.putExtra("usage", "" + mUsage);
    intent.putExtra("usergender", "" + mUsergender);
    setResult(RESULT_OK, intent);
    finish();
}

@Override
protected void onDestroy() {
    super.onDestroy();
}
```



```
}

```

2.5 안드로이드 Manifest

이제 위와 같은 기능을 사용하기 위해 Android Manifest file 을 수정합니다.

먼저 2.1 에서 설명한 3 개의 Activity 를 명시합니다.

```
<application
    android:allowBackup="true"
    android:debuggable="false"
    android:icon="@drawable/dpush_Logo"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar" >

    <activity
        android:name="kr.co.joytune.dpchat.DPushActivity"
        android:screenOrientation="portrait"
        android:label="DPush" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity
        android:name="kr.co.joytune.dpchat.DPushLogin"
        android:label="@string/title_activity_login"
        android:screenOrientation="portrait"
        android:theme="@android:style/Theme.Black.NoTitleBar" >
    </activity>

    <activity
        android:name="kr.co.joytune.dpchat.DPushProdKeyCheck"
        android:label="@string/title_activity_prodkey"
        android:screenOrientation="portrait"
        android:theme="@android:style/Theme.Black.NoTitleBar" >
    </activity>

</application>
```

또한, 정의된 안드로이드 기능을 사용하기 위한 권한정보를 추가합니다

```
<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" />
```