



ATID Co.,Ltd

XC1003-1 RFID


Program

Developers Guide

Android Developer Guide

박영호

2015-04-23


		XC1003-1 RFID Program Developers Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

개정 이력

버전	개정일자	개정사유 ¹	개정내역 ²	작성자
V1.0	2015-04-23	초안	신규 생성	박영호


¹ 개정사유 : 제정 또는 개정 내용이 이전 문서에 대해 추가/수정/삭제인지 선택 기입

² 개정내역 : 개정이 발생하는 페이지 번호와 변경 내용을 기술

		XC1003-1 RFID Program Developers Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

목차


목차	3
1. Intro	4
2. Getting Started	5
2.1. Create Project for Android.....	5
2.2. Development Environment	12
3. Programing Guide.....	16
3.1. Create and Synchronization	16
3.1.1. Create Reader.....	16
3.1.2. Destroy Reader	19
3.2. Event Handler.....	20
3.3. Action Operation.....	22
3.3.1. Inventory and StopOperation.....	22

		XC1003-1 RFID Program Developers Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

1. Intro

본 문서는 XC1003-1 RFID SDK Library를 사용하여 응용 프로그램을 개발하고자 하는 Android개발자들을 위하여 SDK Library를 사용할 수 있는 개발환경 구축과 SDK Library 사용 방법을 기술 하는데 그 목적이 있다.

본 문서에서 사용되는 개발 도구는 Eclipse IDE for Java Developers가 사용되었고 개발 대상 플랫폼은 Android 4.4.2을 지원한다.

		XC1003-1 RFID Program Developers Guide					
All That Identification		Android Developer Guide			회사		ATID Co.,Ltd
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

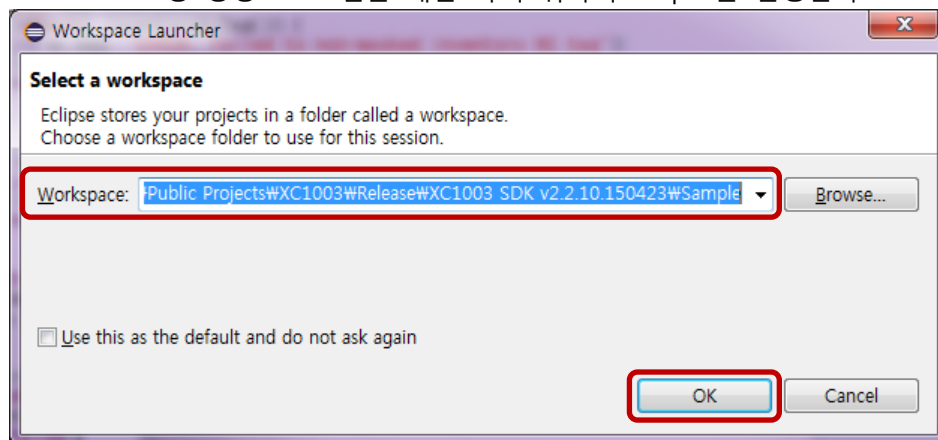
2. Getting Started

Getting Started에서는 XC1003-1 RFID SDK Library를 사용하여 간단하게 Android Sample을 작성하여 XC1003-1 RFID SDK Library 응용 프로그램을 개발하고, XC1003-1 RFID SDK Library를 사용하기 위하여 개발 환경을 구축하는 방법에 대하여 설명한다.

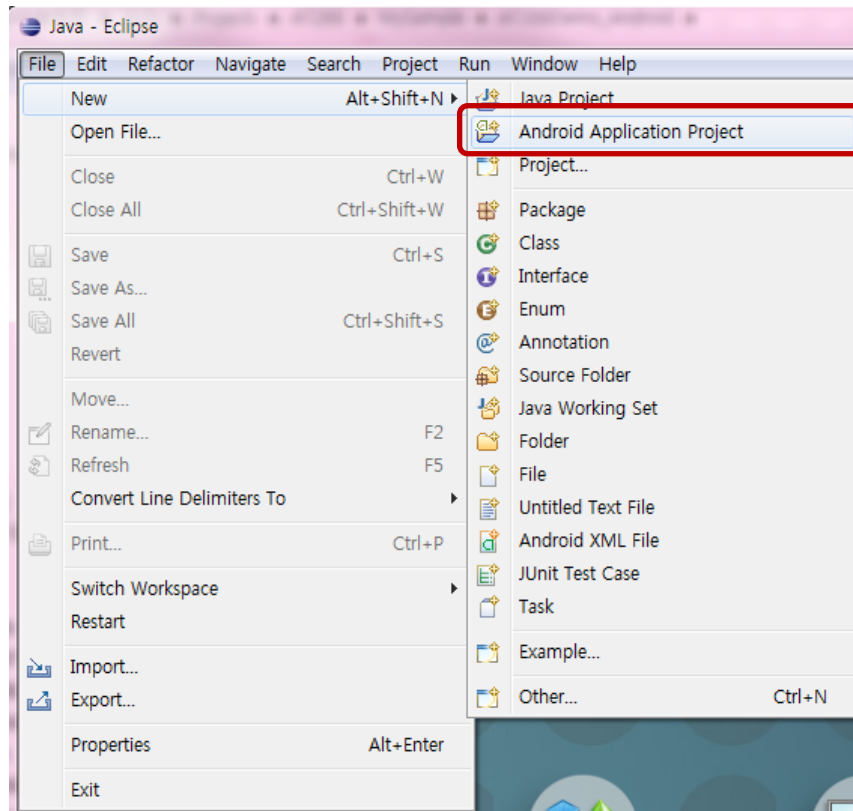
설명에 사용되는 개발 툴은 Eclipse Juno를 사용하였다.

2.1. Create Project for Android

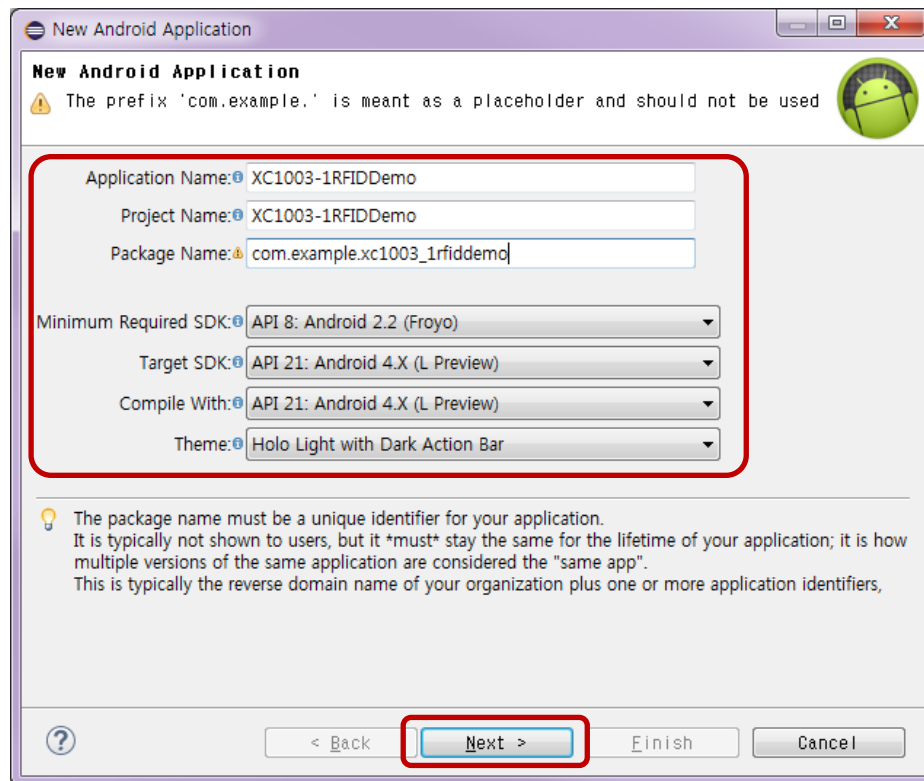
XC1003-1 RFID Android용 응용프로그램을 개발 하기 위하여 Eclipse를 실행한다.



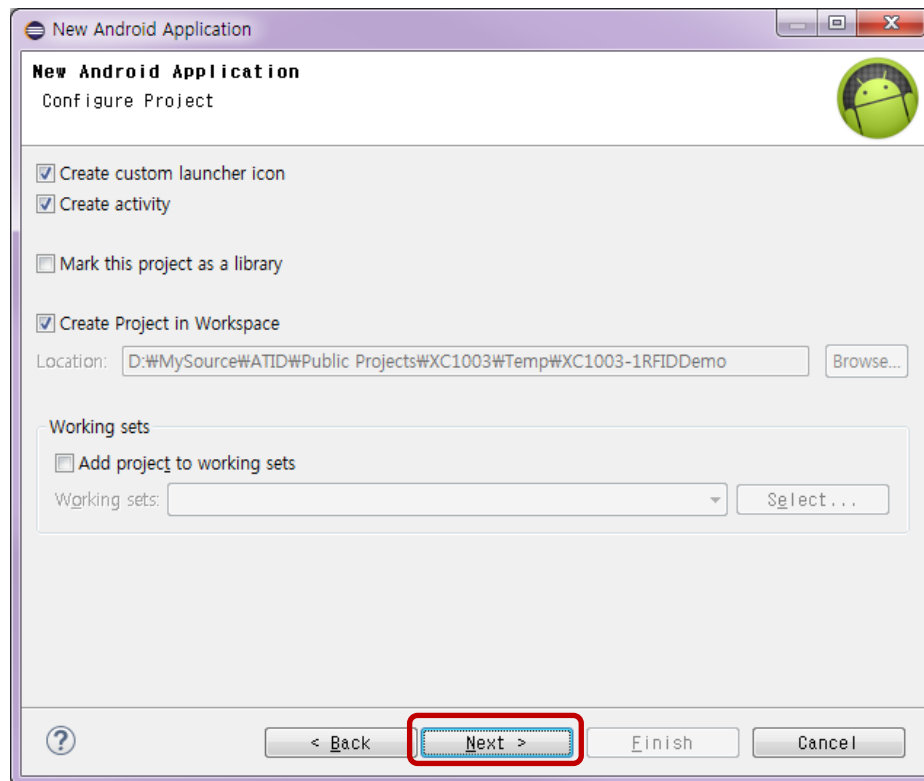
Eclipse를 실행하면 프로젝트를 작업 할 폴더를 선택하기 위한 대화상자가 나타난다. 프로젝트를 작업을 할 폴더를 입력하고 "OK" 버튼을 클릭하여 Eclipse를 진행한다.



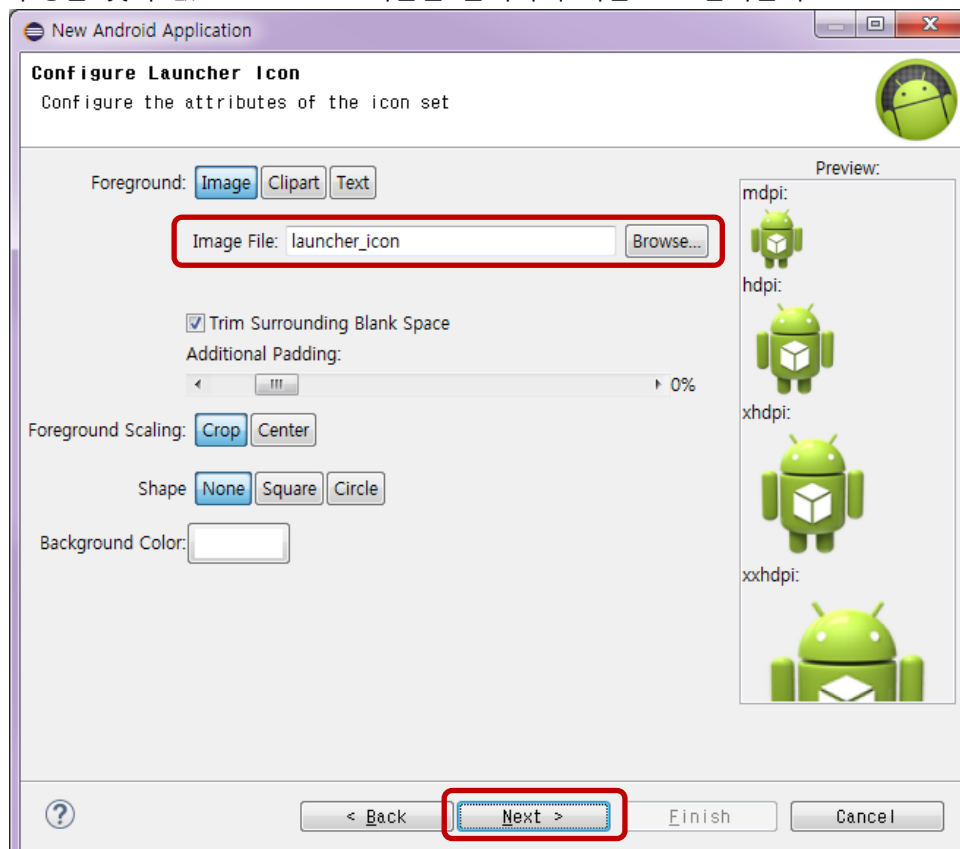
Eclipse가 실행되면 메뉴에서 File > New > Android Application Project를 선택하여 프로젝트를 생성하기 위한 대화상자를 나타나게 한다.




New Android Application 대화상자가 나타나면 Application Name과 Project Name을 입력하고 Package Name을 입력하고, Minimum Required SDK, Target SDK, Compile With 그리고 Theme를 선택하고 "Next" 버튼을 클릭하여 프로젝트 생성을 진행한다.

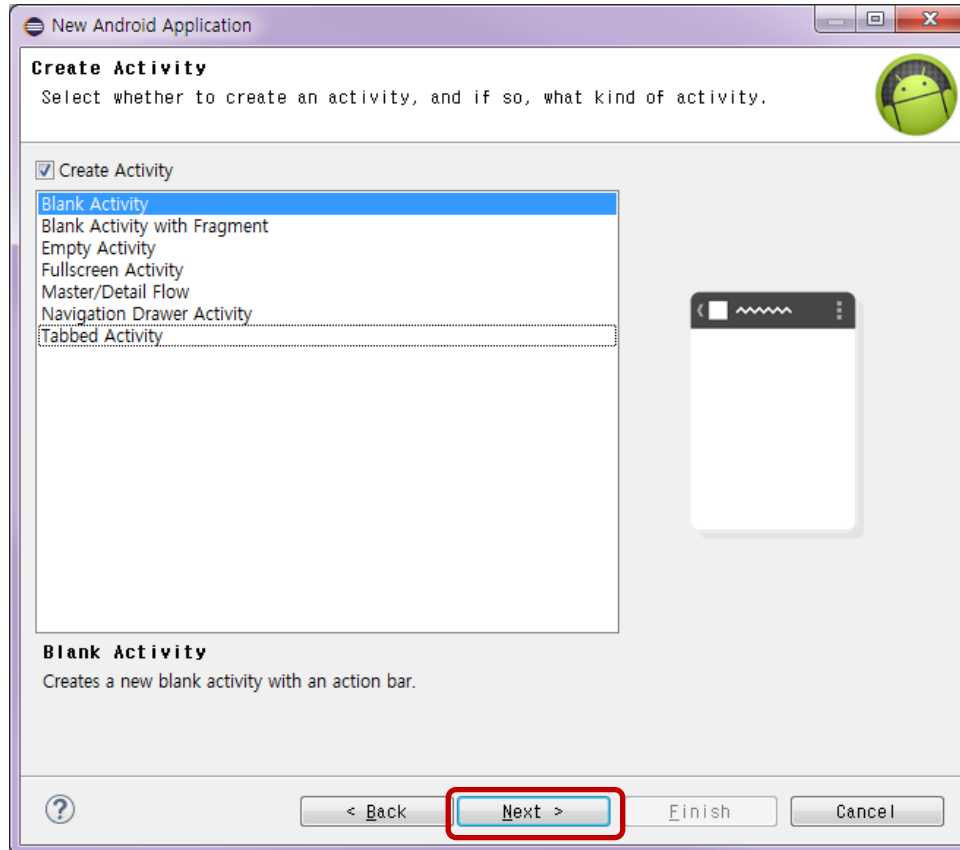


특별하게 수정할 것이 없으므로 "Next"버튼을 클릭하여 다음으로 넘어간다.

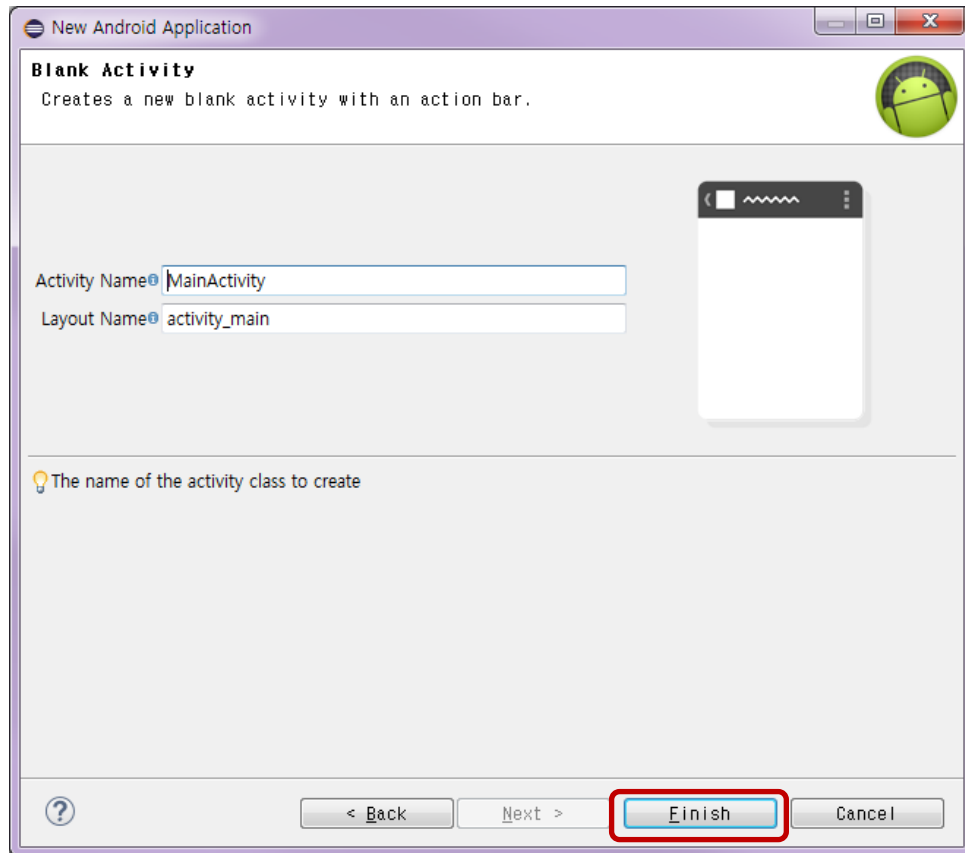


 All That Identification	XC1003-1 RFID Program Developers Guide						
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

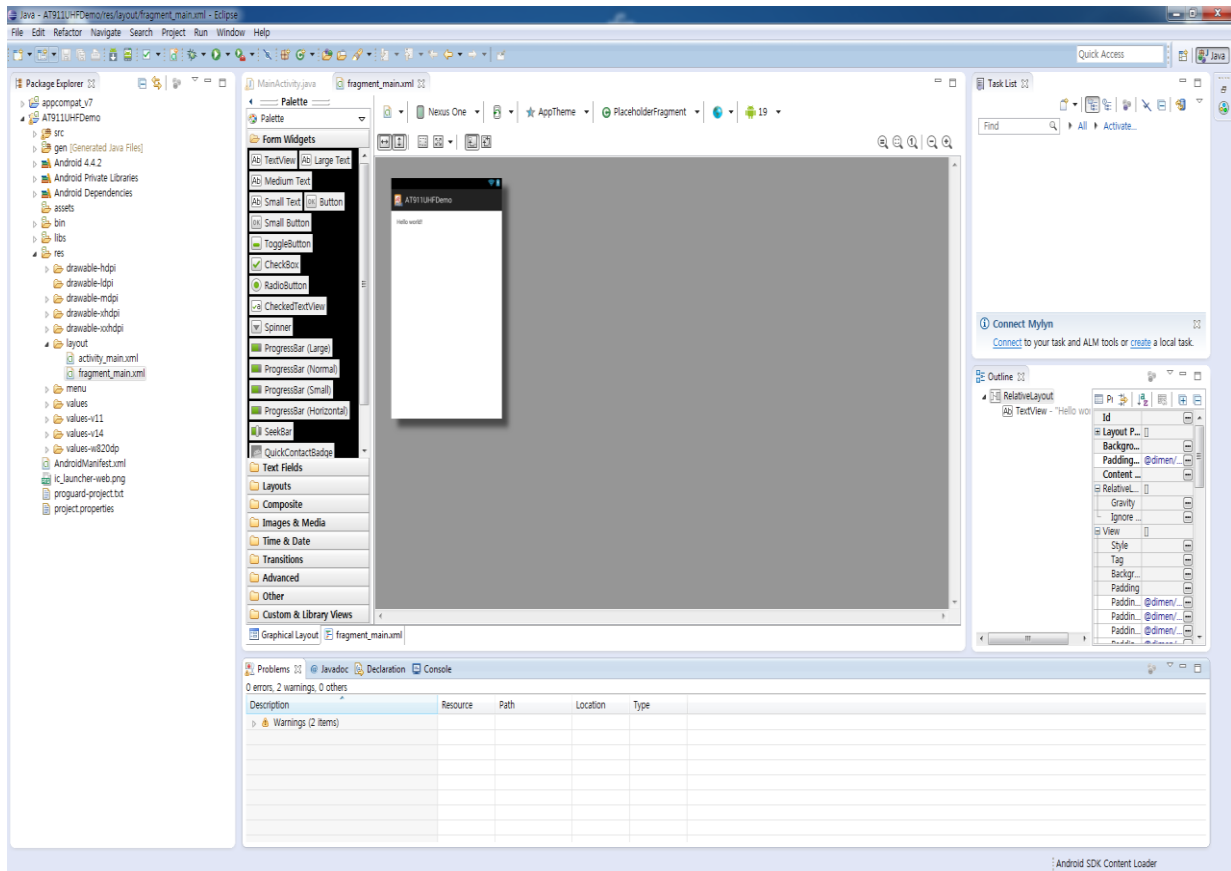
Application에서 사용할 Icon이 될 이미지를 입력하고 "Next"버튼을 클릭하여 다음으로 넘어간다.



Application의 Theme를 설정하는 화면이다. Theme를 선택한 후 "Next"버튼을 클릭하여 다음으로 넘어간다.



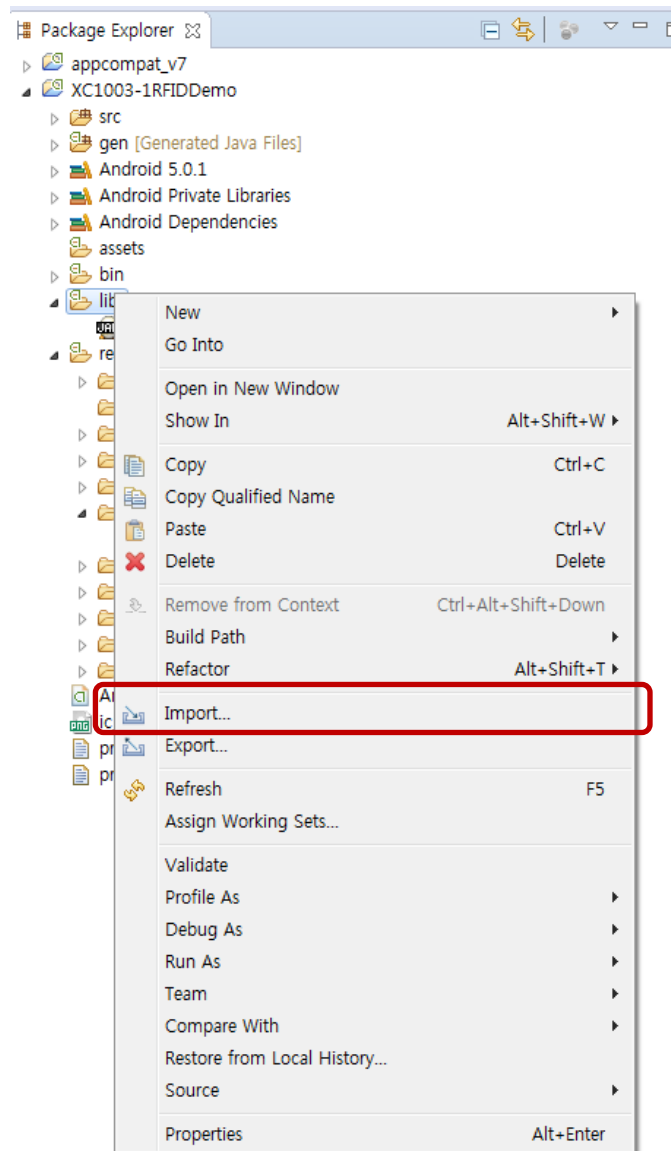
Activity Name에 Main Activity의 이름을 입력하고 Layout Name에도 개발에 사용할 Main Layout의 이름을 입력한 후 "Finish" 버튼을 클릭하여 프로젝트의 생성을 종료한다.



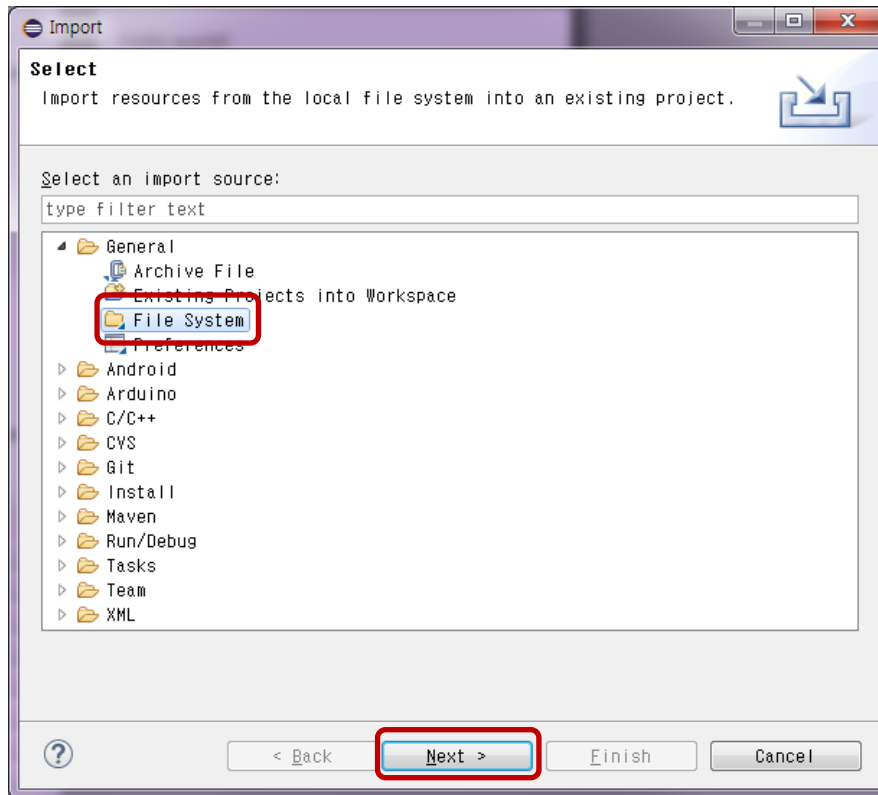
프로젝트가 생성된 Eclipse의 화면이다.

2.2. Development Environment

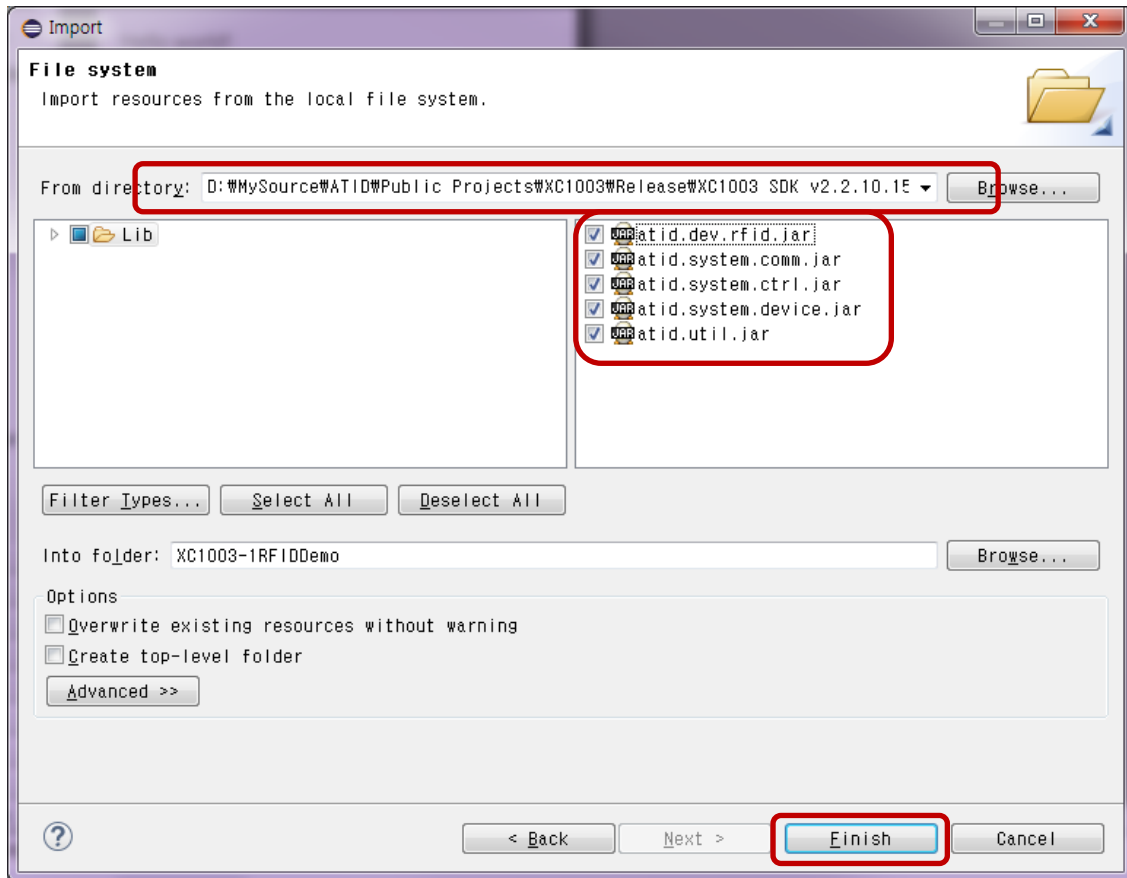
XC1003-1 RFID Android 응용 프로그램을 개발하기 위해서는 XC1003-1 RFID SDK Jar파일이 필요하다.



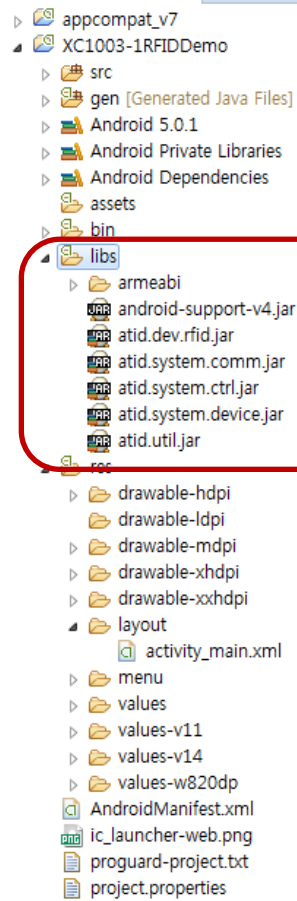
XC1003-1 RFID SDK jar파일을 Import하기 위해 Package Explorer에서 libs폴더를 선택하고 마우스 오른쪽 버튼을 클릭하여 팝업 메뉴가 나타나도록 한다. 팝업 메뉴에서 "Import..."메뉴를 클릭한다.



Import 대화상자가 나타나면 Select an import source에서 General > File System을 선택하고 "Next"버튼을 클릭한다.



From directory에 XC1003-1 RFID SDK Library가 설치된 폴더에 있는 Library폴더의 하위 폴더의 Android폴더를 선택하면 atid.dev.rfid.jar파일이 보일 것이다. 체크박스에 체크를 하고 "Finish"버튼을 클릭하면 AT288Lib.jar파일을 프로젝트가 생성한 폴더에 있는 libs폴더로 복사를 해온다.



jar파일이 Package Explorer로 복사되어 온 것을 확인할 수 있다.

3. Programing Guide

3.1. Create and Synchronization

3.1.1. Create Reader

XC1003-1 RFID SDK Library를 사용하여 XC1003-1 RFID 를 사용하기 위해서는 ATRfidReader 클래스의 인스턴스를 생성하여야 한다.

ATRfidReader클래스의 인스턴스를 생성하는 방법은 Sample 소스의 MainActivity.java파일에 onCreate메서드에서 확인할 수 있다.

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

    Log.i(TAG, "INFO. onCreate()");

    // Setup always wake up
    if (this.wakeLock == null) {
        PowerManager powerManager = (PowerManager)
getSystemService(POWER_SERVICE);
        this.wakeLock = powerManager.newWakeLock(
            PowerManager.FULL_WAKE_LOCK, APP_NAME);
        this.wakeLock.acquire();
    }

    // Display Version
    String packageName = getPackageName();
    String versionName = "";
    try {
        versionName =
getPackageManager().getPackageInfo(packageName,
            PackageManager.GET_META_DATA).versionName;
    } catch (NameNotFoundException e) {
    }
    this.txtDemoVersion.setText(versionName);

    enableButtons(true);

    txtLogo.setTextColor(getResources().getColor(R.color.connected_device));
}
```

AT911에서는 UHF Device의 Instance에 대하여 현재 제어를 하여야 하기 위해 onResume메서드에서 생성하는 것이 적당하다.

```
@Override
protected void onResume() {
    super.onResume();

    Log.i(TAG, "EVENT. onResume()");
}
```



```
// Initialize Scanner Instance
if ((this.reader = ATRfidManager.getInstance()) == null) {
    // Show Error Alert Dialog
    AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setOnCancelListener(new
DialogInterface.OnCancelListener() {

        @Override
        public void onCancel(DialogInterface dialog) {
            finish();
        }
    });
    alert.setPositiveButton(R.string.ok_button,
        new DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface
dialog, int which) {


                finish();
            }
        });
    alert.setIcon(android.R.drawable.ic_dialog_alert);
    alert.setTitle(R.string.device_error);
    alert.setMessage(R.string.filaed_to_device);
    alert.show();
    return;
}

this.txtFirmwareVersion.setText(this.reader.getFirmwareVersion());
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {

    switch (requestCode) {
        case INVENTORY_VIEW:
        case READ_MEMORY_VIEW:
        case WRITE_MEMORY_VIEW:
        case LOCK_MEMORY_VIEW:
        case OPTION_VIEW:
            enableButtons(true);
            break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
txtLogo.setTextColor(getResources().getColor(R.color.connected_device));
}
```

주의 할 것은 onActivityResult메서드를 반드시 재정의하여 Reader클래스의 onActivityResult메서드를 호출하여야 정상 동작을 할 수 있다. OpenDeviceListActivity메서드는 내부에서 Intent로

	XC1003-1 RFID Program Developers Guide						
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	박영호	날자	2015-04-23	버전	v1.0

Activity를 호출하며 그 결과값을 Main Activity의 onActivityResult를 통하여 반환 받는데, XC1003-1 RFID SDK Library 내부에서 이를 처리하기 위해서 Reader 개체의 onActivityResult를 호출해 주어야 하는 것이다. Reader와의 Synchronization을 위해 RfidReaderEventListener를 추가한다.

3.1.2. Destroy Reader

Main Activity와 Reader개체의 Life Cycle를 관리하기 위해 onCreate, onDestroy, onActivityResult등의 오버라이드된 메서드를 재정의하는 경우가 발생하는데 필요에 따라서 Reader개체를 동기화 하기 위해서 각 재정의된 메서드에서 Reader개체의 동일한 메서드를 호출할 수 있다.

```
@Override
protected void onDestroy() {

    Log.i(TAG, "EVENT. onDestroy()");

    // Setup basic wake up state
    if (this.wakeLock != null) {
        this.wakeLock.release();
        this.wakeLock = null;
    }
    super.onDestroy();
}
```

```
@Override
protected void onPause() {
    Log.i(TAG, "EVENT. onPause()");

    // Deinitialize RFID reader Instance
    ATRfidManager.onDestroy();

    super.onPause();
}
```

3.2. Event Handler

AT911 Device로부터 reader 이벤트를 수신하기 위해 클래스에 부분에 RfidReaderEventListener를 선언하여 Event Handler를 파라미터로 입력 받는다. Reader 개체는 Handler개체로 AT911 로부터 수신한 이벤트를 발생시킨다.

이벤트의 범주는 onReaderStateChanged, onReaderDetectTag , onReaderResult 이벤트로 나뉜다. 다음은 MainActivity.java에 구현되어 있는 Handler개체를 할당하는 예제이다.

```
public class InventoryView extends AccessView implements RfidReaderEventListener,
    OnClickListener, OnItemSelectedListener {
    @Override
    protected void onResume() {
        super.onResume();

        Log.i(TAG, "EVENT. onResume()");

        // Initialize RFID Reader
        getReader().setEventListener(this);

        // Initialize Power Gain
        int i = 0;
        RangeValue powerRange = getReader().getPowerRange();
        for (i = powerRange.max; i >= powerRange.min; i--) {
            this.adpPower.addItem(powerRange.max - i, "" + i + " dBm");
        }
        this.adpPower.notifyDataSetChanged();
        this.spnPower.setSelection(this.adpPower
            .indexOf(getReader().getPower()));

        // Enable Widgets
        enableWidgets(true);
    }

    @Override
    protected void onPause() {
        Log.i(TAG, "EVENT. onPause()");

        super.onPause();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        enableWidgets(true);

        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

onReaderStateChange 메서드는 AT911 동작중에 발생하는 Action에 대하여 이벤트를 수신하기

위한 메서드로 아래와 같이 구현되어 있으며 자세한 내역은 샘플 프로그램을 참조합니다.

```
@Override
public void onReaderStateChanged(ReaderState state) {
    enableWidgets(true);
}
```

onReaderDetectTag 메서드는 AT911 Tag를 검출 하였을 때 이벤트가 동작하며 아래와 같이 구현되어 있습니다..

```
@Override
public void onReaderDetectTag(String tag) {
    // Add Tag List
    this.adpTags.addItem(tag);
    // Display Tag Count
    this.txtCount.setText("" + this.adpTags.getCount());
    // Play Beep
    beep();
}
```

onReaderResult 메서드는 AT911 Memory Access, Tag Access 등 Action을 동작하였을 때에 Action에 대한 결과를 확인하기 위한 Event입니다.

```
@Override
public void onReaderResult(ResultCode code) {
    Log.e(TAG, "ERROR. Failed to onReaderResult(" + code + ")");
}
```

3.3. Action Operation

3.3.1. Inventory and StopOperation

AT911 로 Tag를 Inventory하기 위해서는 inventory6cTag, inventory6bTag, readEpc6cTag, readEpc6bTag 메서드를 사용한다. Inventory메서드를 사용하는 방법은 InventoryView.java파일에 구현되어 있다.

```
// Start Action
private void startAction() {

    Log.i(TAG, "INFO. Start Action()");

    if (getContinuousMode()) {
        // Multi Tag Inventory...
        if (!mReader.inventory6cTag()) {
            Log.e(TAG, "ERROR. Failed to non-masked inventory 6C tag");

            //enableWidgets(true);
            return;
        }
    } else {
        // Single Tag Inventory...
        if (!mReader.readEpc6cTag()) {
            Log.e(TAG, "ERROR. Failed to non-masked read EPC 6C tag");
            //enableWidgets(true);
            return;
        }
    }

    mAction=true;
    enableWidgets(false);
    //adpTags.start();
}

// Stop Action
private void stopAction() {
    Log.i(TAG, "INFO. Stop Action()");
    mAction=false;

    mReader.stop();
    //adpTags.shutdown();

    //enableWidgets(false);
    enableWidgets(true);
}
```