



ATID Co.,Ltd

RFID Programming Guide

Android Developer Guide

SDK Team
2017-02-22

		RFID Programming Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

개정 이력

버전	개정일자	개정사유 ¹	개정내역 ²	작성자
V1.0	2015-07-07	초안	신규 생성	박영호
V2.0	2017-02-22	수정	변경 내역 적용	조영진

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

² 개정내역 : 개정이 발생하는 페이지 번호와 변경 내용을 기술

		RFID Programming Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

목차

목차	3
1. Intro	4
2. Getting Started	5
2.1. Create Project for Android.....	5
2.2. Development Environment	11
3. Programing Guide.....	15
3.1. 초기화	15
3.1.1. 객체 생성	15
3.1.2. EventListener 등록/해제	15
3.1.3. Sleep/Wakeup 관리.....	16
3.2. Event Handler.....	17
3.3. Action Operation.....	18
3.3.1. Start Operation.....	18
3.3.2. Stop Operation	19
3.4. 종료	20

		RFID Programming Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

1. Intro

본 문서는 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 4.2.2을 지원한다.

		RFID Programming Guide					
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

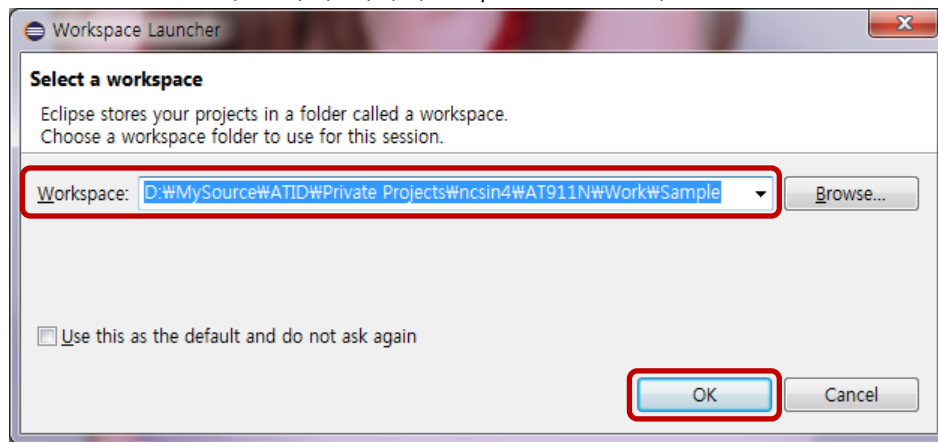
2. Getting Started

Getting Started에서는 SDK Library를 사용하여 간단하게 Android Sample을 작성해 봄으로써, RFID 응용프로그램을 개발하는 방법에 대하여 설명한다.

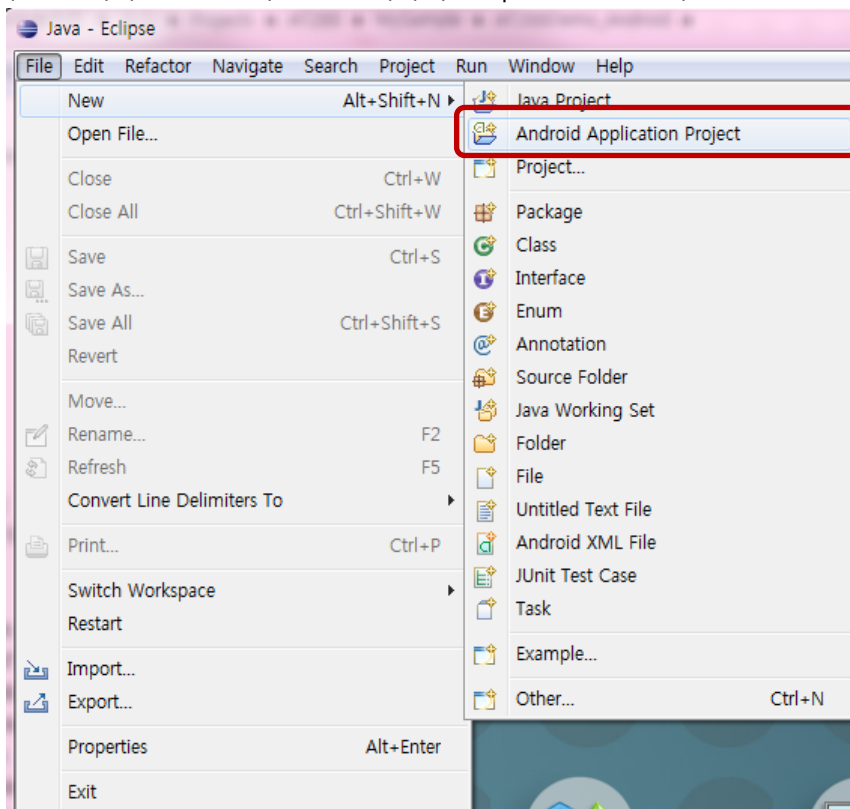
설명에 사용되는 개발 툴은 Eclipse Juno를 사용하였다.

2.1. Create Project for Android

Android용 응용프로그램을 개발 하기 위하여 Eclipse를 실행한다.

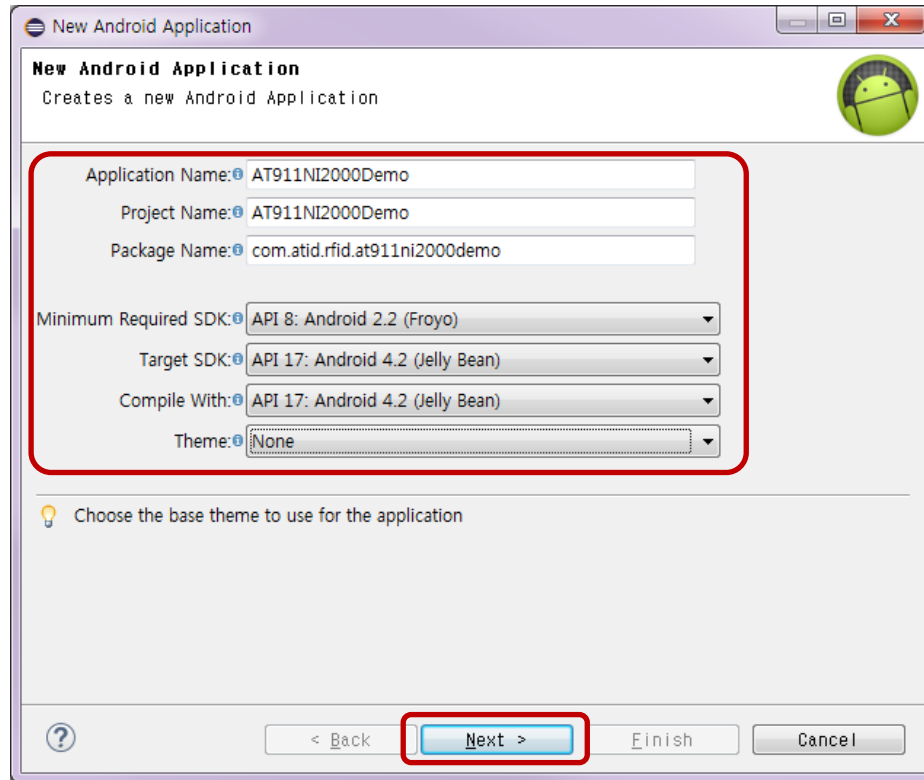


Eclipse를 실행하면 프로젝트를 작업 할 폴더를 선택하기 위한 대화상자가 나타난다. 프로젝트를 작업을 할 폴더를 입력하고 "OK" 버튼을 클릭하여 Eclipse를 진행한다.

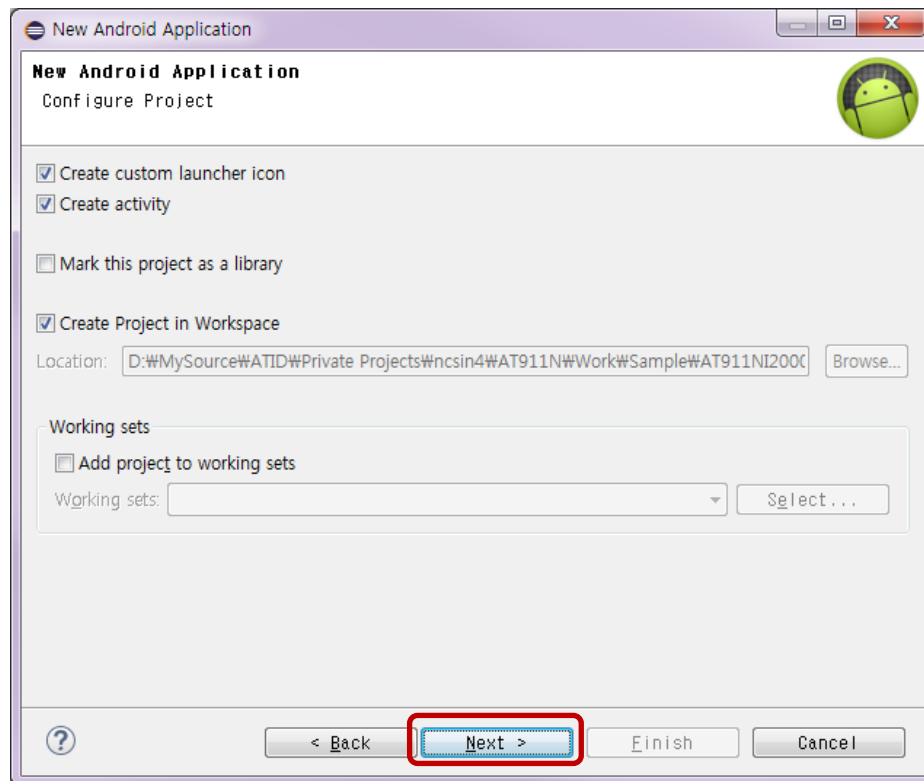


		RFID Programming Guide					
All That Identification		Android Developer Guide			회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

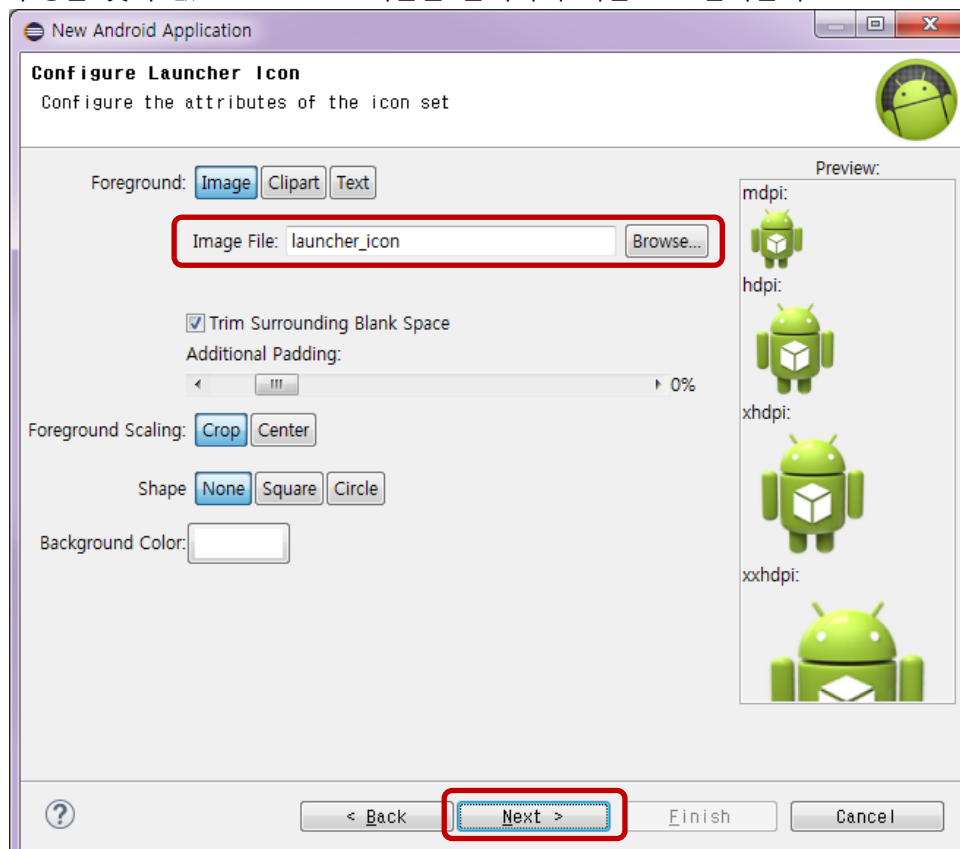
Eclipse가 실행되면 메뉴에서 File > New > Android Application Project를 선택하여 프로젝트를 생성하기 위한 대화상자를 나타나게 한다.



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.

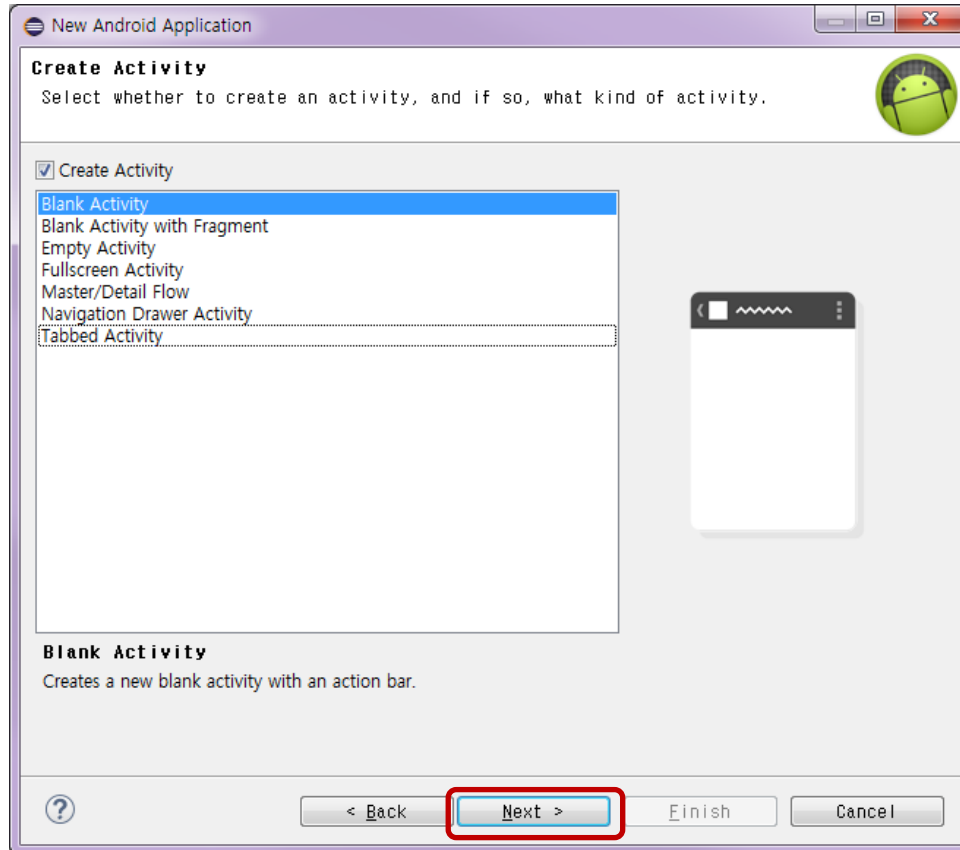


특별하게 수정할 것이 없으므로 "Next"버튼을 클릭하여 다음으로 넘어간다.

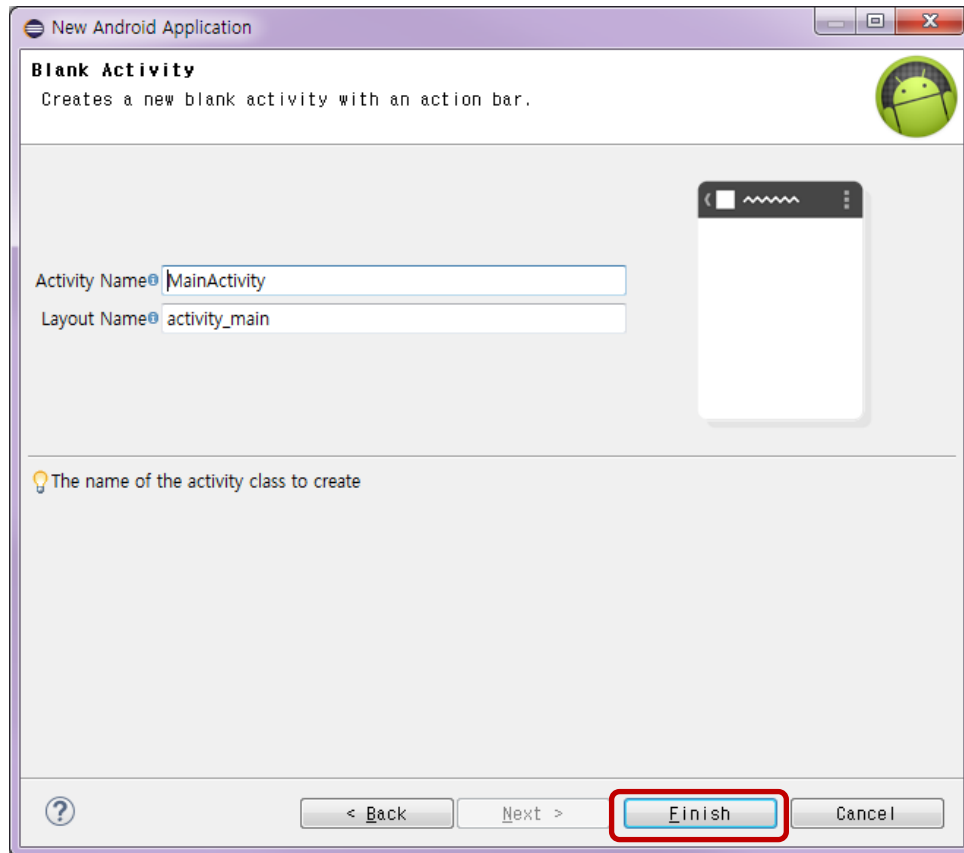


		RFID Programming Guide					
All That Identification		Android Developer Guide			회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

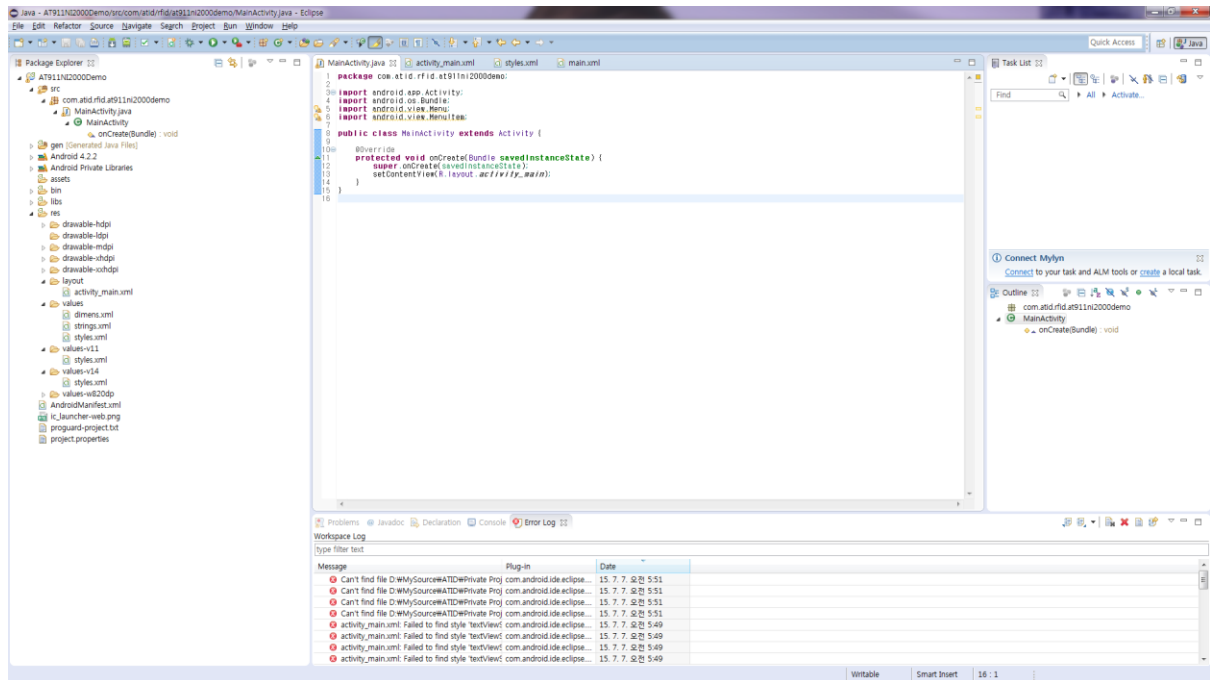
Application에서 사용할 Icon이 될 이미지를 입력하고 "Next"버튼을 클릭하여 다음으로 넘어간다.



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



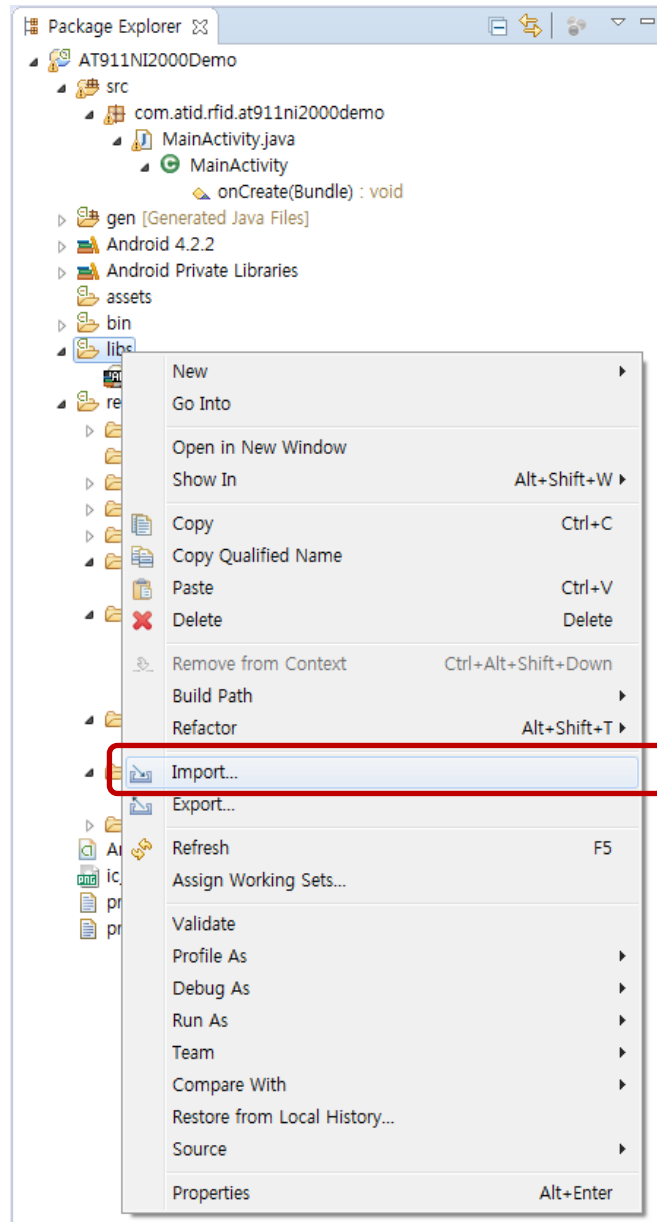
Activity Name에 Main Activity의 이름을 입력하고 Layout Name에도 개발에 사용할 Main Layout의 이름을 입력한 후 "Finish" 버튼을 클릭하여 프로젝트의 생성을 종료한다.



프로젝트가 생성된 Eclipse의 화면이다.

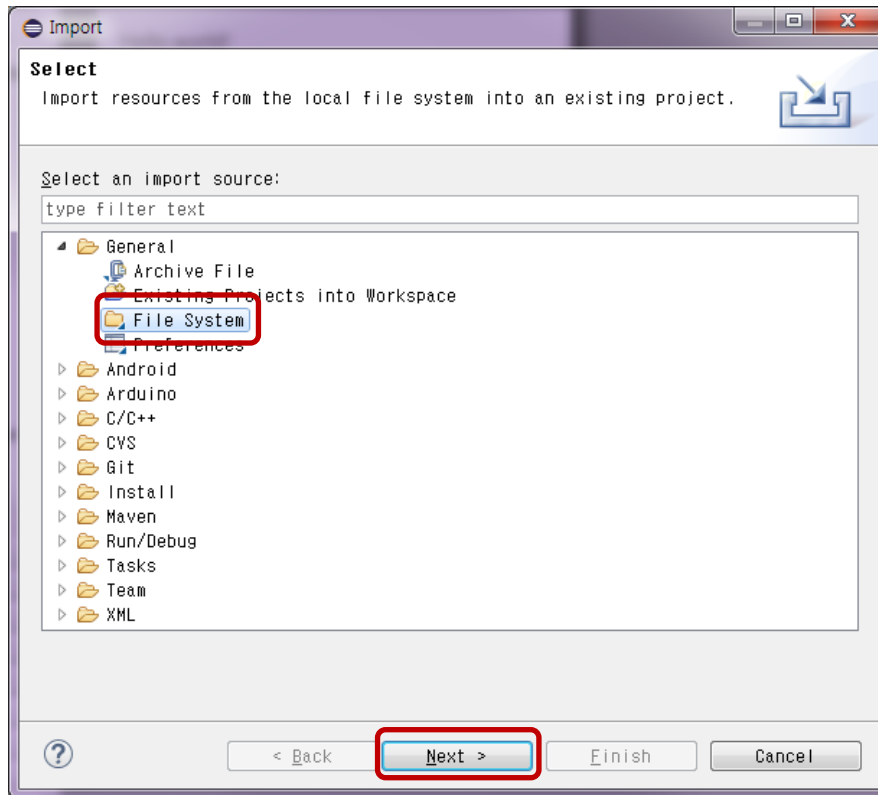
2.2. Development Environment

RFID Android 응용 프로그램을 개발하기 위해서는 RFID SDK Jar파일이 필요하다.

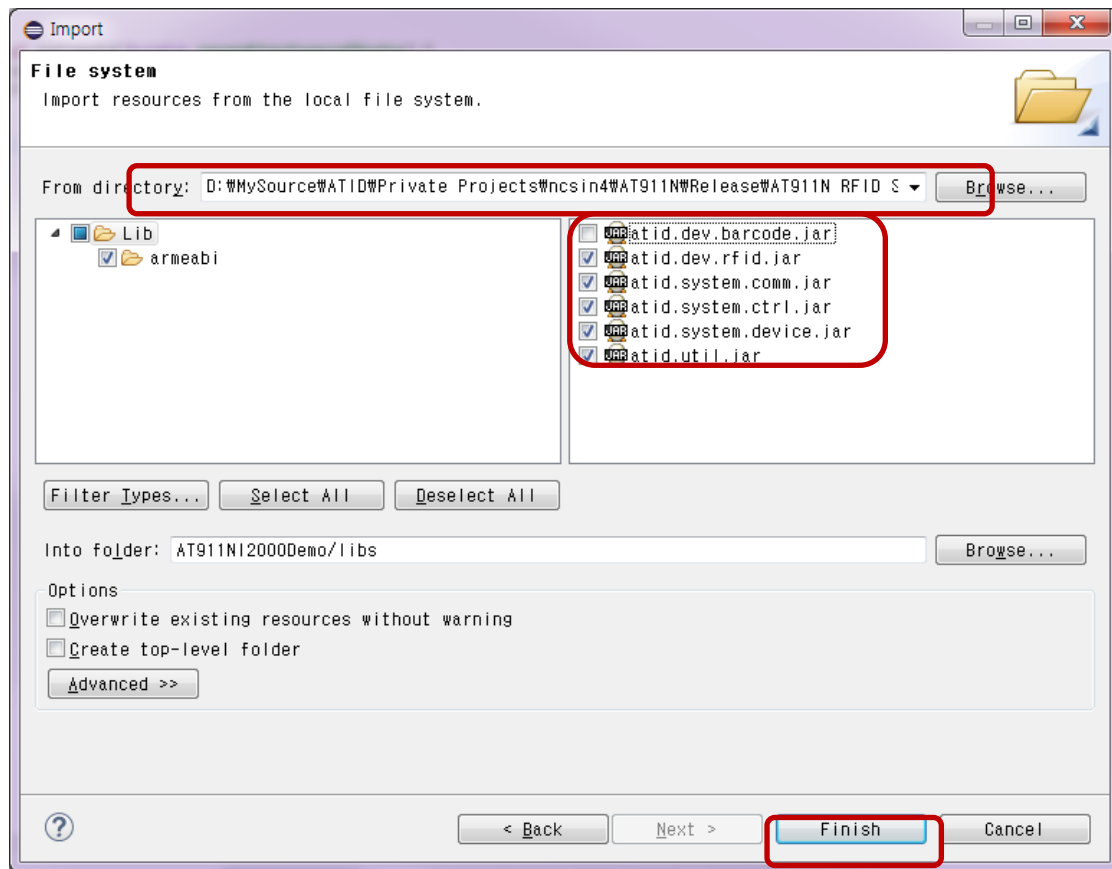


RFID SDK jar파일을 Import하기 위해 Package Explorer에서 libs폴더를 선택하고 마우스 오른쪽 버튼을 클릭하여 팝업 메뉴가 나타나도록 한다. 팝업 메뉴에서 "Import..."메뉴를 클릭한다.

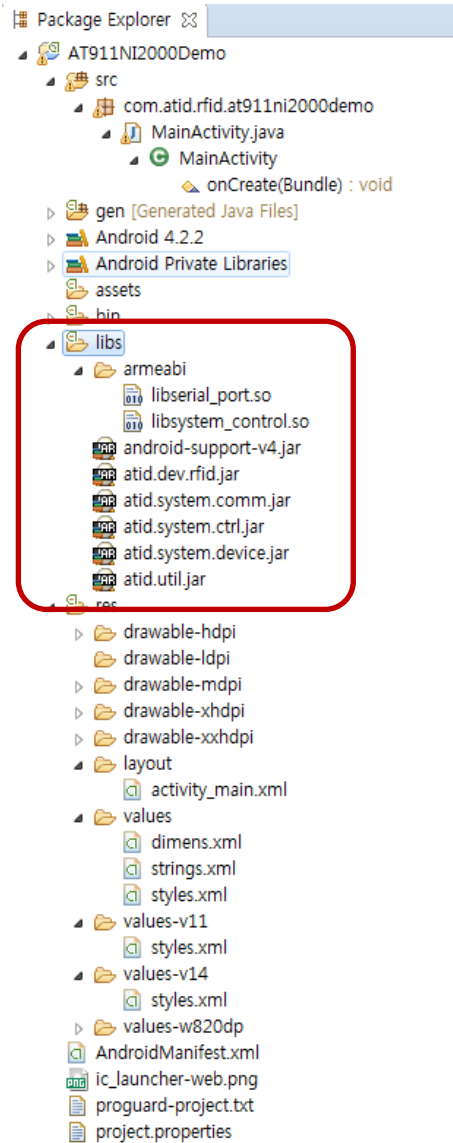
 All That Identification	RFID Programming Guide						
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0



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



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



jar파일이 Package Explorer로 복사되어 온 것을 확인할 수 있다.

3. Programing Guide

3.1. 초기화

3.1.1. 객체 생성

RFID SDK Library를 사용하여 RFID 를 사용하기 위해서는 ATRfidReader 클래스의 인스턴스를 생성하여야 한다. ATRfidReader 클래스의 인스턴스를 생성하는 방법은 Sample 소스의 MainActivity.java파일에 onCreate메서드에서 확인할 수 있다.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // 생략

    if ((mReader = ATRfidManager.getInstance()) == null) {
        // 생략
    }

    // 생략
}
```

3.1.2. EventListener 등록/해제

ATRfidReader 인스턴스로부터 응답을 수신하기 위해서는, 응답을 수신할 사용자 Activity에 RfidReaderEventListener interface를 구현해야 하며, interface가 구현된 Activity를 setEventListener를 통해서 listener를 등록하고 removeEventListener를 통해서 등록을 해제 한다.

Activity가 여러 개인 App에서는 이 과정을 여러 번 거쳐야 하므로, Activity의 onResume, onPause 메서드에서 등록/해제를 하도록 한다.

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

    if (mReader != null)
        mReader.setEventListener(this);

    ATLog.d(TAG, "INFO onResume()");
}
@Override
protected void onPause() {
    if (mReader != null)
        mReader.removeEventListener(this);

    ATLog.i(TAG, "INFO. onPause()");
    super.onPause();
}
```

3.1.3. Sleep/Wakeup 관리

Activity가 사용 중일 때와, 그렇지 않을 때를 구분하여 RFID Module의 전원 및 리소스를 관리 한다. Activity의 onStart에서 ATRfidManager.wakeup()을 Activity의 onStop에서 ATRfidManager.sleep()을 호출한다. wakeup과 sleep은 반드시 pair로 호출이 되어야 하며, sleep이 호출된 뒤에 wakeup이 호출되지 않는다면 Module은 정상 동작 하지 않을 수 있다.

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

    if (mReader != null) {
        ATRfidManager.wakeup();
    }

    ATLog.i(TAG, "INFO. onStart()");
}

@Override
protected void onStop() {

    ATRfidManager.sleep();

    ATLog.i(TAG, "INFO. onStop()");

    super.onStop();
}
```

 All That Identification	RFID Programming Guide						
Android Developer Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SDK Team	날자	2017-02-22	버전	V2.0

3.2. Event Handler

EventListener를 등록했다면, event가 발생할 때 마다 사용자가 구현한 interface 함수가 실행된다. 이벤트의 범주는
Module Operation (onReaderStateChanged), Inventory Operation (onReaderReadTag),
Access Operation (onReaderResult) 으로 구분된다.

onReaderStateChange 메서드는 Module의 connection 상태에 대하여 이벤트를 수신하기 위한 메서드로 아래와 같이 구현되어 있으며 자세한 내역은 샘플 프로그램을 참조한다.

```
@Override
public void onReaderStateChanged(ATRfidReader reader, ConnectionState state) {

    switch (state) {
        case Connected:
            // to do something
            break;
        case Disconnected:
            // to do something
            break;
        case Connecting:
            // to do something
            break;
        default:
            break;
    }

    ATLog.i(TAG, "EVENT. onReaderStateChanged(%s)", state);
}
```

onReaderReadTag 메서드는 Inventory operation으로 Tag를 인식 하였을 때 이벤트가 동작하며 아래와 같이 구현되어 있으며, 태그 데이터, RSSI, Phase 정보들을 전달 한다.

```
@Override
public void onReaderReadTag(ATRfidReader reader, String tag, float rssi, float phase) {

    ATLog.i(TAG, "EVENT. onReaderReadTag([%s], %.2f, %.2f)", tag, rssi, phase);
}
```

onReaderResult 메서드는 Read/Write/Lock/Kill등의 Access operation에 대한 결과를 확인하기 위한 Event입니다.

```
@Override
public void onReaderResult(ATRfidReader reader, ResultCode code, ActionState action, String epc, String data, float rssi, float phase) {
    ATLog.i(TAG, "EVENT. onReaderResult(%s, %s, [%s], [%s], %.2f, %.2f", code, action, epc, data, rssi, phase);
}
```

3.3. Action Operation

3.3.1. Start Operation

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

```
// Start Action
protected void startAction() {
    // 생략
    if(mReader.getModuleType() == RfidModuleType.I900MA) {
        mMAREader = (ATRfid900MAREader)mReader;
        if (chkContinuousMode.isChecked()) {

            // Multiple Reading
            if(tagType == TagType.Tag6B) {
                mMAREader.inventory6bTag()
            } else if(tagType == TagType.Tag6C) {
                mMAREader.inventory6cTag()
            } else if(tagType == TagType.TagRail) {
                mMAREader.inventoryRailTag()
            } else if(tagType == TagType.TagAny) {
                mMAREader.inventoryAnyTag()
            }
        } else {
            // Single Reading
            if(tagType == TagType.Tag6B) {
                mMAREader.readEpc6bTag()
            } else if(tagType == TagType.Tag6C) {
                mMAREader.readEpc6cTag()
            } else if(tagType == TagType.TagRail) {
                mMAREader.readEpcRailTag()
            } else if(tagType == TagType.TagAny) {
                mMAREader.readEpcAnyTag()
            }
        }
    } else {
        // Multiple Reading
        if (chkContinuousMode.isChecked()) {
            mReader.inventory6cTag()
        } else {
            // Single Reading
            mReader.readEpc6cTag()
        }
    }

    ATLog.i(TAG, "INFO. startAction()");
}
```

3.3.2. Stop Operation

진행중인 Inventory 또는 Access 명령을 중단 시키기 위해서는 stop을 호출한다.

ActionActivity.java 파일을 참조한다.

```
protected void stopAction() {

    if(mReader.getAction() == ActionState.Stop) {
        ATLog.e(TAG, "ActionState is not busy.");
        return;
    }
    ResultCode res;
    enableWidgets(false);

    if ((res = mReader.stop()) != ResultCode.NoError) {
        ATLog.e(TAG, "ERROR. stopAction() - Failed to stop operation [%s]",
res);
        enableWidgets(true);
        return;
    }

    ATLog.i(TAG, "INFO. stopAction()");
}
```

3.4. 종료

RFID의 사용이 끝나서 완전히 종료를 해야 한다면, `ATRfidManager.onDestroy()`를 호출하여 RFID와 관련된 자원을 해제 한다.

다음은 MainActivity.java의 `onDestroy()` 함수 구현 부 이다.

```
@Override
protected void onDestroy() {

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

    // Wake Unlock
    SysUtil.wakeUnLock();

    saveConfig();

    ATLog.d(TAG, "INFO. onDestroy");
    ATLog.shutdown();

    super.onDestroy();
}
```