ANDROID PROGRAMMING

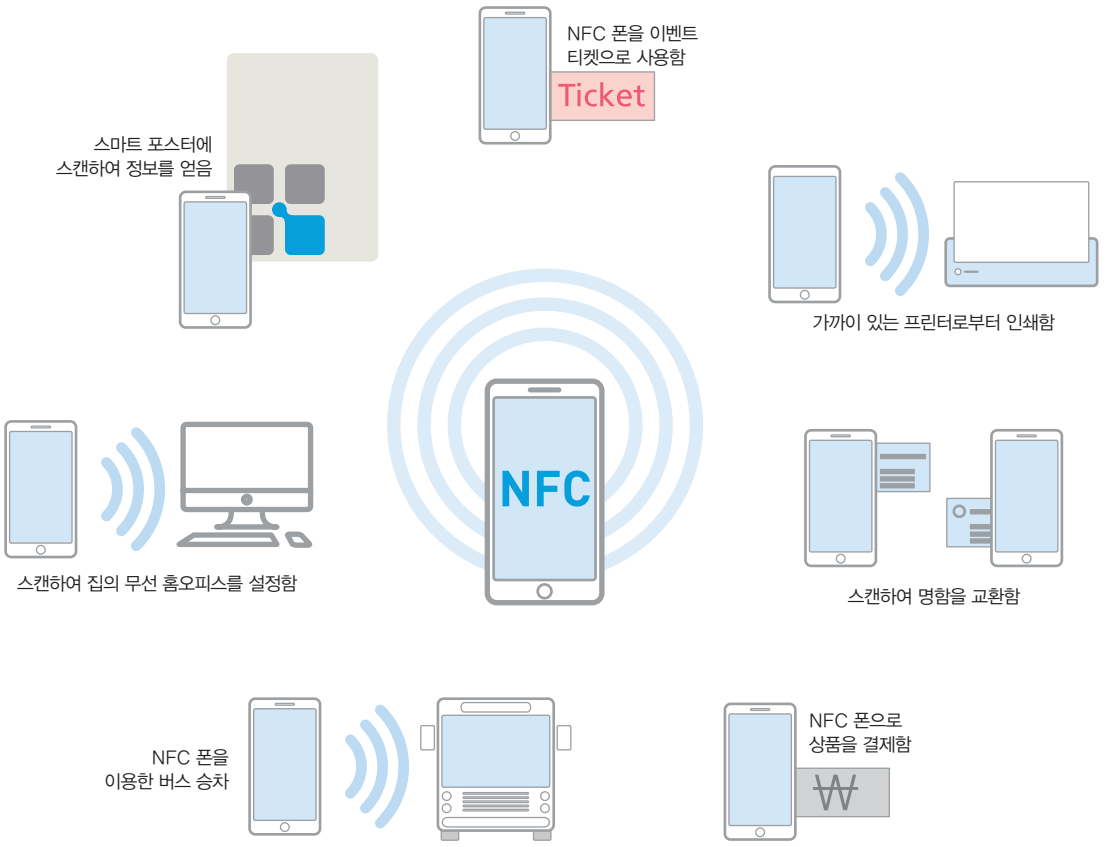
16 CHAPTER

NFC 활용

16 1 NFC 활용 원리

16.1.1 NFC 활용 앱의 예

NFC(Near Field Communication)는 근거리 무선 기술로서, 연결을 위해 4 cm 또는 그 이하 거리에서 가능하다. 안드로이 폰에는 NFC 센서가 있는데 NFC tag를 읽고 씀으로써 데이터를 공유할 수 있다. NFC는 소비자와 비즈니스에 다양한 혜택을 제공할 수 있다. 다음 그림은 그 예를 나타낸 것이다.

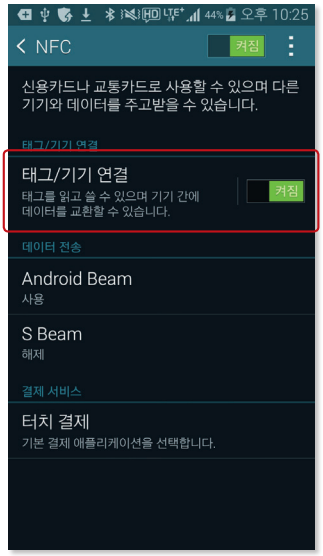


• NFC 활용 예(nfc.forum.org)

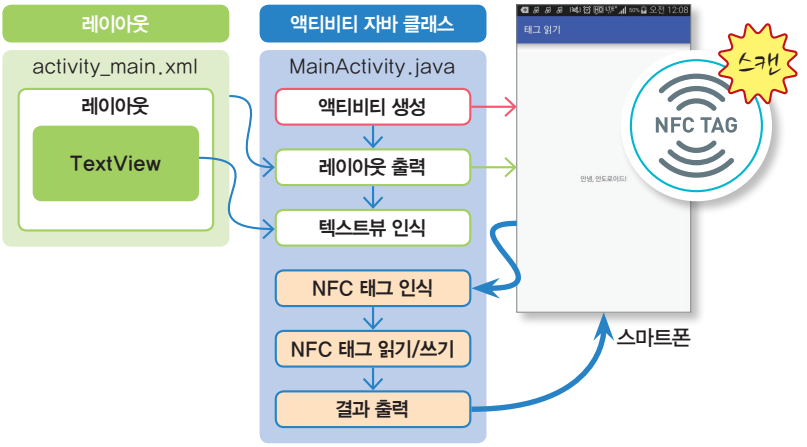
16.1.2 NFC 활용 원리

NFC 기능이 있는 안드로이드 디바이스는 세 가지 모드가 있다. ❶ NFC 디바이스가 수동형 NFC 태그를 읽거나 쓰는 **reader/writer mode**, ❷ NFC 디바이스가 다른 NFC 디바이스와 데이터를 교환하는 안드로이드 빔(android beam)과 같은 **P2P mode**, ❸ NFC 디바이스가 NFC 카드가 되어 다른 NFC 리더에 의해 읽혀지는 **card emulation mode**가 그것이다. 읽혀지는 card emulation mode는 전자 결제 등에 응용될 수 있다.

NFC 디바이스가 수동형 NFC 태그를 읽거나 쓰는 **reader/writer mode**를 기반으로 하는 NFC 활용 원리를 살펴보자. NFC 기능을 사용하기 위해서는 먼저 스마트폰이 NFC 기능을 사용할 수 있도록 설정해야 한다.



스마트폰이 NFC 태그를 스캔하면, NFC 태그를 인식하고 태그에 저장된 데이터를 추출하거나 문자를 태그에 저장 후에 레이아웃에 배치된 뷰에 결과를 출력한다.



다음 절에서 먼저 태그 내용을 읽는 방법을 살펴보자.

16.2 NFC 태그 읽기: NFC Reader 프로젝트 (태그 읽기)

16.2.1 프로젝트 개요 (NFC 태그 읽기)

초기화면에서 스마트폰으로 NFC 태그를 스캔하면 그 태그에 기록된 내용을 읽어 출력하는 NFC Reader 앱을 개발해보자. **어플리케이션 이름**은 『**NFC Reader**』, **어플리케이션 라벨**과 **액티비티 라벨**은 『**태그 읽기**』로 한다.

프로젝트 개요: 태그에 기록된 문자 읽기

Application Name: NFC Reader

어플리케이션 라벨: 태그 읽기

초기 화면

태그 스캔 후 태그 내용 출력

16.2.2 프로젝트 개발

STEP 1 프로젝트 생성

다음의 5단계 절차에 따라 프로젝트를 실행한다.

절차	내용
❶ 프로젝트 시작	메뉴에서 'File → New Project' 클릭
❷ 프로젝트 구성	Application Name: NFC Reader Company Domain: yschang.example.com
❸ 제품형태	Phone and Tablet
❹ 액티비티 유형	Empty Activity
❺ 파일 옵션	디폴트 값으로 설정

STEP 2 파일 편집

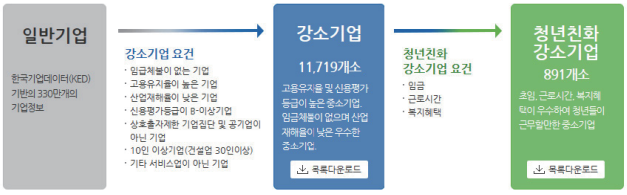
● **파일 구조와 편집 내용 (NFC 태그 읽기)**

모듈	폴더	소스 파일	편집 내용
manifests		AndroidManifest.xml	• NFC 사용 허용
java	com.example. yschang. nfcreader	MainActivity.java	• NFC 태그 스캔 시, 태그 내용 출력
res	drawable		
	layout	activity_main.xml	• NFC 태그 내용 출력을 위한 텍스트뷰 배치
	mipmap	ic_launcher.png	
	values	dimens.xml	
		strings.xml	• 어플리케이션 라벨 수정 • 태그사용 안내를 위한 속성 값 추가
		styles.xml	

수정 추가

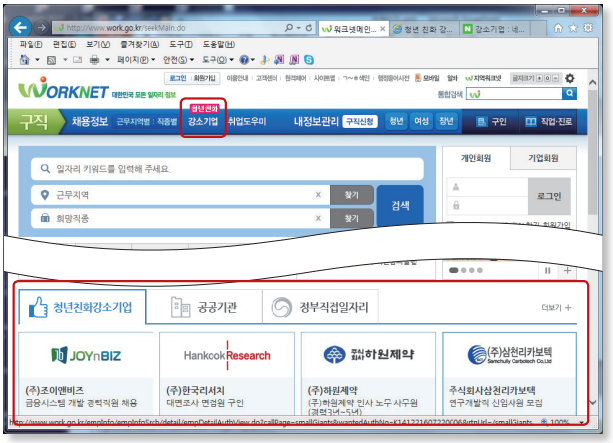
TOPIC 알고 있으면 좋은 소프트웨어 강소기업

강소기업은 작지만 강한 기업이다. 고용노동부에서는 한국기업데이터에 등록된 330만 개의 일반기업 중에서 고용유지율 및 신용평가 등급이 높고 임금체불이 없으며 산업 재해율이 낮은 만여 개의 우수한 중소기업을 강소기업으로 선정하고, 그 중에서 초임, 근로시간, 복지혜택이 우수하여 청년들이 근무할만한 중소기업을 청년 친화 강소기업으로 분류하고 있다.



• 중소기업 선정기준(고용노동부)

고용노동부와 한국고용정보원에서 공동으로 운영하는 워크넷(www.work.go.kr)에 가면 강소기업과 청년친화 강소기업 목록을 볼 수 있다. 청년친화 강소기업은 업종별로는 건설업, 도매및소매업, 전문과학및기술서비스업, 제조업, 출판영상방송통신및정보서비스업으로 분류하고 있는데, 소프트웨어 관련 회사는 출판영상방송통신및정보서비스업으로 분류되어 있다. 또한, 일자리 친화, 글로벌 역량, 기술력 우수, 지역선도 기업, 재무 건전성, 사회적 가치 측면에서 우수한 기업들을 분류하고 있으며, 기업 및 채용 정보 등 다양한 취업 관련 정보를 제공하고 있으니, 기업 및 채용 트렌드에 관심을 가져보자.



• 강소기업 현황(워크넷)

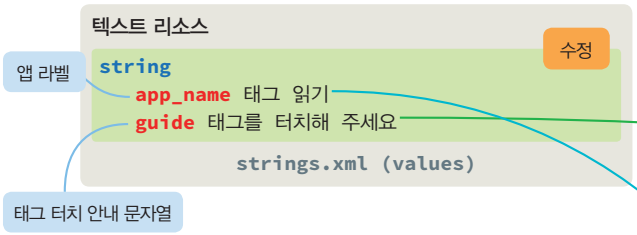
● 파일 간의 연관관계

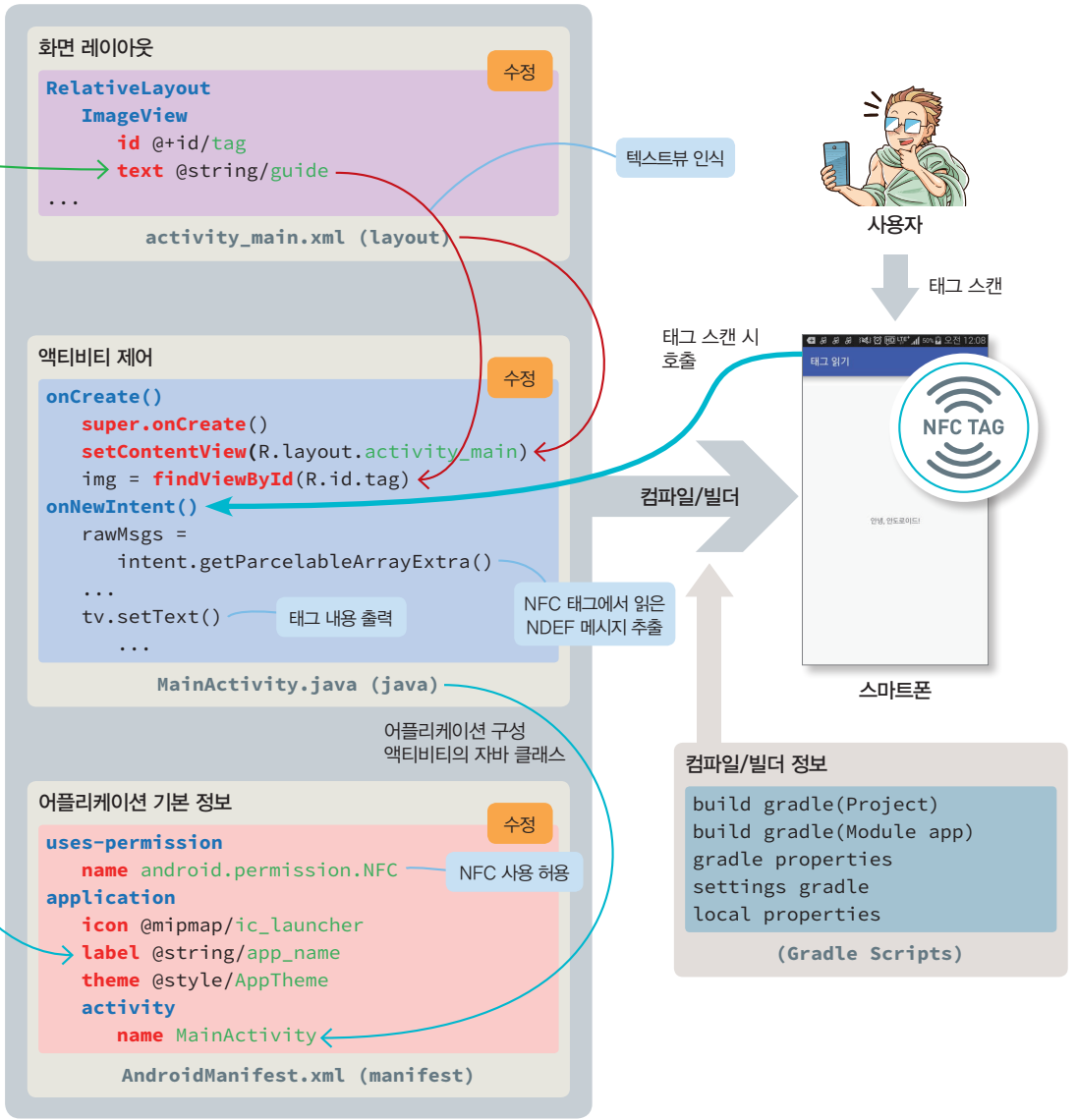
strings.xml에는 초기치로 설정되어 있는 어플리케이션 라벨을 '태그 읽기'로 수정하고, 태그사용 안내를 위한 속성 값을 추가한다.

activity_main.xml에는 태그를 읽은 값을 출력할 텍스트뷰를 배치한다.

MainActivity.java에는 태그를 읽고 출력하도록 작성한다.

AndroidManifest.xml에는 NFC 사용을 허용하도록 설정한다.




화면 레이아웃
수정
RelativeLayout
ImageView
id @+id/tag
text @string/guide
...
activity_main.xml (layout)
텍스트뷰 인식
액티비티 제어
수정
onCreate()
super.onCreate()
setContentView(R.layout.activity_main)
img = findViewById(R.id.tag)
onNewIntent()
rawMsgs =
intent.getParcelableArrayExtra()
...
tv.setText()
...
태그 내용 출력
NFC 태그에서 읽은
NDEF 메시지 추출
MainActivity.java (java)
어플리케이션 구성
액티비티의 자바 클래스
어플리케이션 기본 정보
수정
uses-permission
name android.permission.NFC
NFC 사용 허용
application
icon @mipmap/ic_launcher
label @string/app_name
theme @style/AppTheme
activity
name MainActivity
AndroidManifest.xml (manifest)
사용자
태그 스캔
태그 스캔 시
호출
컴파일/빌드
태그 읽기
NFC TAG
스마트폰
컴파일/빌더 정보
build gradle(Project)
build gradle(Module app)
gradle properties
settings gradle
local properties
(Gradle Scripts)

● **편집**

❶ 텍스트 자원

어플리케이션 이름을 수정하기 위해 **app_name** 속성값에 해당하는 데이터를 '**태그 읽기**'로 수정한다. 화면에 표시할 안내문구 문자열을 추가한다.

소스 | strings.xml

app_name의 데이터를 '태그 읽기'로 수정

```
01 <resources>
02     <string name="app_name">태그 읽기</string>
03     <string name="guide">태그를 터치해 주세요.</string>
04 </resources>
```

태그 읽기 안내 문자열

A

❷ 화면 설계

NFC 태그 내용 출력 영역을 **TextView**로 배치하고 **ID**를 부여한다. **ID**는 액티비티 클래스에서 NFC 태그를 읽고 출력할 때 사용된다.

소스 | activity_main.xml → B

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     xmlns:tools="http://schemas.android.com/tools"
04     android:layout_width="match_parent"
05     android:layout_height="match_parent"
06     android:paddingBottom="@dimen/activity_vertical_margin"
07     android:paddingLeft="@dimen/activity_horizontal_margin"
08     android:paddingRight="@dimen/activity_horizontal_margin"
09     android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="com.example.nfcreader.MainActivity">
11
12     <TextView
13         android:id="@+id/tag"
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:layout_centerInParent="true"
17         android:text="@string/guide" />
18
19 </RelativeLayout>
```

C

A

태그 스캔 후 태그 내용을 출력할 텍스트뷰

❸ 액티비티 제어

액티비티 간의 통신은 **인텐트**를 통해 실행한다. 액티비티가 호출되면 **onCreate()** 메소드로부터 시작하여 onStart(), onResume() 메소드가 순서대로 호출된다. **onCreate()** 메소드는 NFC 어댑터를 생성하고, onResume() 메소드에서는 NFC 어댑터가 foreground에서 사용 가능하도록 한다. 액티비티가 foreground로 작동 중인 경우에 NFC 태그의 스캔 등에 의해 인텐트를 받기 위해서는 **onNewIntent()** 메소드를 이용한다. 즉, NFC 태그가 스캔되면 **onNewIntent()** 메소드가 호출되는데 이때 인텐트가 전달된다. 인텐트가 생겼을 때 onNewIntent() 메소드가 호출되도록 하기 위해서는 인텐트 객체를 생성할 때 '**FLAG_ACTIVITY_SINGLE_TOP**'을 설정하면 된다.

소스 | MainActivity.java

```
01 package com.example.yschang.nfcreader;
02
03 import android.app.PendingIntent;
04 import android.content.Intent;
05 import android.nfc.NdefMessage;
06 import android.nfc.NdefRecord;
07 import android.nfc.NfcAdapter;
08 import android.os.Bundle;
09 import android.os.Parcelable;
10 import android.support.v7.app.AppCompatActivity;
11 import android.widget.TextView;
12
13 public class MainActivity extends AppCompatActivity {
14
15    private NfcAdapter nfcAdapter;
16    private PendingIntent pendingIntent;
17    TextView tv;
18
19    @Override
20    protected void onCreate(Bundle savedInstanceState) {
21       super.onCreate(savedInstanceState);
22       setContentView(R.layout.activity_main);
23
24       tv =(TextView)findViewById(R.id.tag);
25
26       nfcAdapter = NfcAdapter.getDefaultAdapter(this);
27       Intent intent = new Intent(this, getClass()).addFlags(Intent.
                                                  FLAG_ACTIVITY_SINGLE_TOP);
28       pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);
```

B

C

인텐트 객체 생성.
NFC 태그 스캔 시에 인텐트의 정보를
현재 액티비티에 전달하기 위해
onNewIntent() 메소드를 호출하도록 설정

NFC 어댑터를 생성.
NFC를 지원하지 않는
디바이스는 null 값 반환

태그 내용을 출력할
텍스트뷰 인식

전달받은 인텐트에 대한 PendingIntent 객체를 생성. NFC 어댑터를 포그라운드에서 사용하도록 설정할 때 사용

```
29    }
30
31    @Override
32    protected void onResume() {
33       super.onResume();
34       if(nfcAdapter != null) {
35          nfcAdapter.enableForegroundDispatch(
                                          this, pendingIntent, null, null);
36       }
37    }
38
39    @Override
40    protected void onPause() {
41       super.onPause();
42       if(nfcAdapter != null) {
43          nfcAdapter.disableForegroundDispatch(this);
44       }
45    }
46
47    @Override
48    protected void onNewIntent(Intent intent) {
49       super.onNewIntent(intent);
50
51       Parcelable[] rawMsgs =
             intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
52
53       if(rawMsgs != null) {
54          NdefMessage msgs =(NdefMessage) rawMsgs[0];
55          NdefRecord[] rec = msgs.getRecords();
56          byte[] bt = rec[0].getPayload();
57          String text = new String(bt);
58          tv.setText(text);
59       }
60    }
61 }
```

액티비티가 화면에 나타날 때 실행

NFC 기능이 포그라운드에서 사용 가능하게 함

다른 액티비티가 시작할 때 실행

NFC 기능이 포그라운드에서 사용 가능하지 않게 함

NFC 태그를 스캔하면 호출되며, 인텐트 정보가 전달됨

인텐트에서 NFC 어댑터가 NFC 태그에서 읽은 NDEF 메시지를 추출하여 Parcelable 객체에 저장함

첫 번째 Parcelable 객체를 NDEF 메시지로 변환

NDEF 메시지에 있는 레코드들을 객체 배열에 할당함

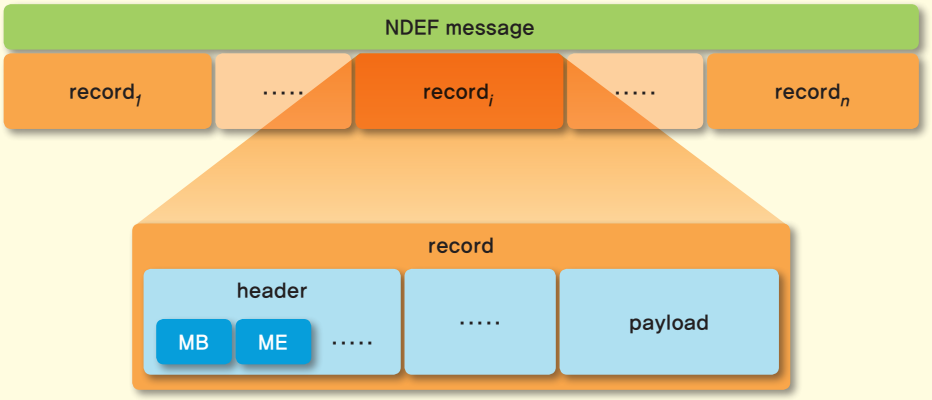
첫 번째 레코드의 페이로드(데이터)를 추출함

추출한 데이터를 출력함

바이트 코드를 문자열로 반환함

TIP NDEF message

NFC 태그의 데이터는 NFC forum(nfc.forum.org)에서 데이터를 교환하기 위해 정의한 NDEF(NFC data exchange format)를 사용한다. 데이터를 저장하는 NDEF message는 하나 또는 다수 개의 NDEF record로 구성된다. 각 record는 header, payload(data) 등이 있다. 첫 record와 마지막 record는 각각 MB(Message begin)와 ME(message end) flag가 1로 설정되어 있다.



클래스와 속성/메소드

● 클래스

클래스/인터페이스	설명
NdefMessage	54행. NDEF message를 나타냄
NdefRecord	55행. NDEF message 내의 NDEF Record를 나타냄
NfcAdapter	15,26행. NFC adapter(NFC controller)를 나타냄
PendingIntent	16,28행. Intent와 같음. 인텐트를 다른 액티비티로 전달할 때 사용
Parcelable	51행. 서로 다른 프로세스 간에 데이터를 쉽게 전달하기 위해 사용하는 인터페이스

● 상수

클래스	상수	설명
Intent	static final int FLAG_ACTIVITY_SINGLE_TOP	27행. 설정되면, 액티비티가 history stack의 top에서 실행 중이면 새로 시작하지 않음
NfcAdapter	static final String EXTRA_NDEF_MESSAGES	51행. 발견된 tag 내의 NdefMessage를 포함하는 정보

● **메소드**

클래스	메소드	설명
Activity	void onNewIntent(Intent intent)	48,49행. 현재 액티비티 실행 중에 인텐트에 정보가 생기면 자동으로 호출됨. 인텐트 객체 생성 시는 'FLAG_ACTIVITY_SINGLE_TOP' flag를 설정해야 함
NfcAdapter	static NfcAdapter getDefaultAdapter(Context context)	26행. 디폴트 NFC adapter(NFC controller)를 얻음. NFC가 지원되지 않는 디바이스에서는 null값을 반환함
	void enableForegroundDispatch(Activity activity, PendingIntent intent, IntentFilter[] filters, String[][] techLists)	35행. 주어진 액티비티에 대해 포그라운드에서 사용 가능하게 함
	void disableForegroundDispatch(Activity activity)	43행. 주어진 액티비티에 대해 포그라운드에서 사용 불가능하게 함
Object	final Class<?> getClass()	27행. 현재 Object 클래스를 나타내는 클래스의 인스턴스를 반환함
PendingIntent	static PendingIntent getActivity(Context context, int requestCode, Intent intent, int flags)	28행. 새로운 액티비티를 실행할 Pendingintent를 반환함
Intent	Intent addFlags(int flags)	27행. 인텐트에 flag를 추가함
	Parcelable[] getParcelableArrayExtra(String name)	51행. intent로부터 data를 추출함
NdefMessage	NdefRecord[] getRecords()	55행. NDEF 메시지 내의 레코드를 반환함
NdefRecord	byte[] getPayload()	56행. NDEF 레코드에 있는 payload(데이터)를 반환함

❹ 환경 설정

NFC 태그에 대한 read/write를 위한 **최소 API 레벨(10)**, **NFC 사용 허용**, **NFC 인텐트 필터** 등을 설정한다.

소스 | AndroidManifest.java

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03     package="com.example.yschang.nfcreader">
04
05     <uses-permission android:name="android.permission.NFC" />
06     <uses-feature android:name="android.hardware.nfc"
                                        android:required="true" />
07
```

NFC 사용 허용

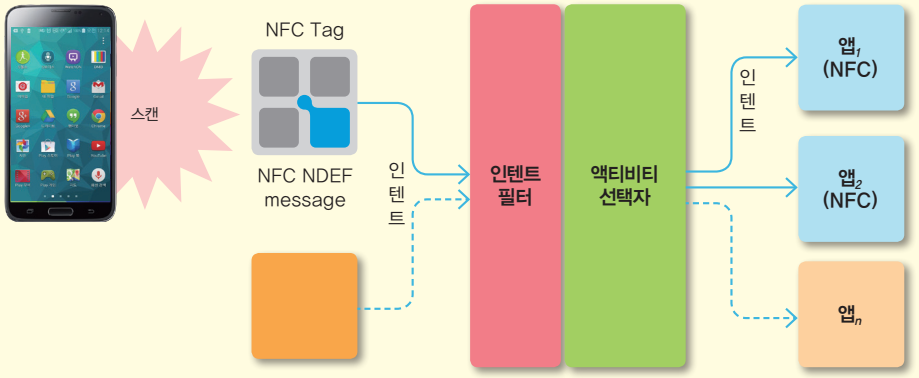
앱이 NFC 기능을 사용한다는 의미이며, 'true'로 설정하며 Google Play에서 NFC 기능을 가진 디바이스에서만 이 앱이 보여짐

```
08    <application
09        android:allowBackup="true"
10        android:icon="@mipmap/ic_launcher"
11        android:label="@string/app_name"
12        android:supportsRtl="true"
13        android:theme="@style/AppTheme">
14        <activity android:name=".MainActivity">
15            <intent-filter>
16                <action android:name="android.intent.action.MAIN" />
17
18                <category android:name="android.intent.category.LAUNCHER" />
19            </intent-filter>
20
21            <intent-filter>
22                <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
23                <category android:name="android.intent.category.DEFAULT"/>
24                <data android:mimeType="text/plain" />
25            </intent-filter>
26        </activity>
27    </application>
28
29 </manifest>
```

'text/plain' MIME 타입의 NFC 태그 스캔 후에 앱이 시작될 수 있도록 설정
(반대로 앱을 먼저 실행하고 NFC 태그를 스캔할 경우에는 intent-filter 부분을 설정하지 않아도 됨)

TIP Tag dispatch system

안드로이드 시스템은 여러 앱들이 사용하는 인텐트들이 있다. Tag dispatch system(태그 데이터 처리를 위한 작업 시간 할당 시스템)이 NFC 태그 정보를 가지고 있는 인텐트를 만들고 인텐트 필터를 통해 앱에 인텐트를 보낸다. 만약 하나 이상의 앱이 그 인텐트를 처리할 수 있다면, 액티비티 선택자(activity chooser)는 액티비티를 선택할 수 있도록 사용자에게 보여준다. 만일 NFC 태그로부터 'text/plain'의 MIME 타입으로 작성된 NDEF 메시지가 읽혀지면, 인텐트 필터는 AndroidManifest.xml에 action이 'android.nfc.action.NDEF_DISCOVERED'로 data가 'text/plain'으로 등록되어 있는 앱들에게 인텐트를 전달한다.



Tag dispatch system은 세 가지 인텐트를 정의하는데, 인텐트들은 우선순위가 있다.

우선순위	인텐트	설명
1	ACTION_NDEF_DISCOVERED	이 인텐트는 NDEF 데이터를 포함하는 태그가 스캔될 때 액티비티를 실행하기 위해 사용됨. 인텐트의 우선순위가 가장 높으며, Tag dispatch system은 다른 인텐트 이전에 이 인텐트를 가지는 액티비티를 실행하려고 함. 가능하면 아래 두 인텐트보다 구체적이기 때문에 이 인텐트 사용을 권장함
2	ACTION_TECH_DISCOVERED	ACTION_NDEF_DISCOVERED 인텐트가 없으면 tag dispatch system은 이 인텐트를 가지는 앱을 실행하려고 함. 만일, MIME 타입 또는 URI가 아닌 NDEF 데이터를 포함하거나 NDEF 데이터가 아닌 다른 알려진 태그 기술인 경우, ACTION_NDEF_DISCOVERED를 실행하지 않고 이 인텐트가 바로 실행됨
3	ACTION_TAG_DISCOVERED	ACTION_NDEF_DISCOVERED 또는 ACTION_TECH_DISCOVERED 인텐트를 가지는 액티비티가 없는 경우, 이 인텐트가 실행됨

STEP 3 프로젝트 실행

다음 2단계 절차에 따라 프로젝트를 실행하고 그 결과를 살펴보자.

절차	내용
실행메뉴 선택	'Run' 메뉴에서 'Run app' 클릭(또는 'Run app' 아이콘 클릭)
디바이스 선택	스마트폰 디바이스를 선택하고 'OK' 버튼 클릭

이제 태그에 문자를 기록하는 방법을 살펴보기로 하자.

16.3 NFC 태그 쓰기: NFC Writer 프로젝트 (태그 쓰기)

16.3.1 프로젝트 개요

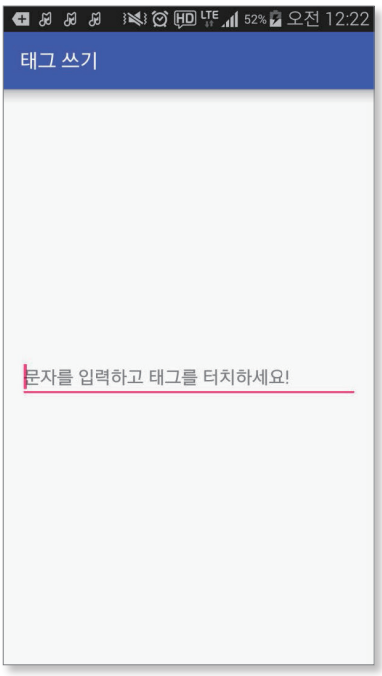
이제 문자를 입력하고 NFC 태그에 기록하는 앱을 개발해보자. 어플리케이션 이름은 『NFC Writer』, 어플리케이션 라벨과 액티비티 라벨은 『태그 쓰기』로 한다.

프로젝트 개요: 태그에 문자 쓰기

Application Name: NFC Writer

어플리케이션 라벨: 태그 쓰기

초기 화면



문자를 입력하고 태그를 터치하면 토스트에 결과가 나타남

16.3.2 프로젝트 개발

STEP 1 프로젝트 생성

다음의 5단계 절차에 따라 프로젝트를 실행한다.

절차	내용
❶ 프로젝트 시작	메뉴에서 'File → New Project' 클릭
❷ 프로젝트 구성	Application Name: NFC Writer Company Domain: yschang.example.com
❸ 제품형태	Phone and Tablet
❹ 액티비티 유형	Empty Activity
❺ 파일 옵션	디폴트 값으로 설정

STEP 2 파일 편집

● **파일 구조와 편집 내용**

모듈	폴더	소스 파일	편집 내용
manifests		AndroidManifest.xml	• NFC 사용 허용
java	com.example. yschang.nfc writer	MainActivity.java	• 입력 문자 추출 • 입력 문자의 태그에 기록
res	drawable		
	layout	activity_main.xml	• 태그 내용 입력을 위한 EditText 배치
	mipmap	ic_launcher.png	
	values	dimens.xml	
		strings.xml	• 어플리케이션 라벨 수정
		styles.xml	

수정 추가

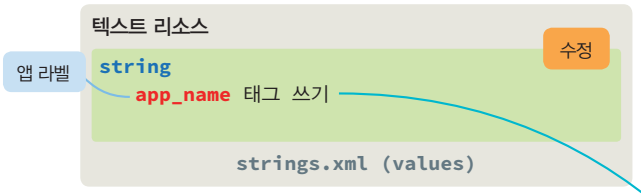
● **파일 간의 연관관계**

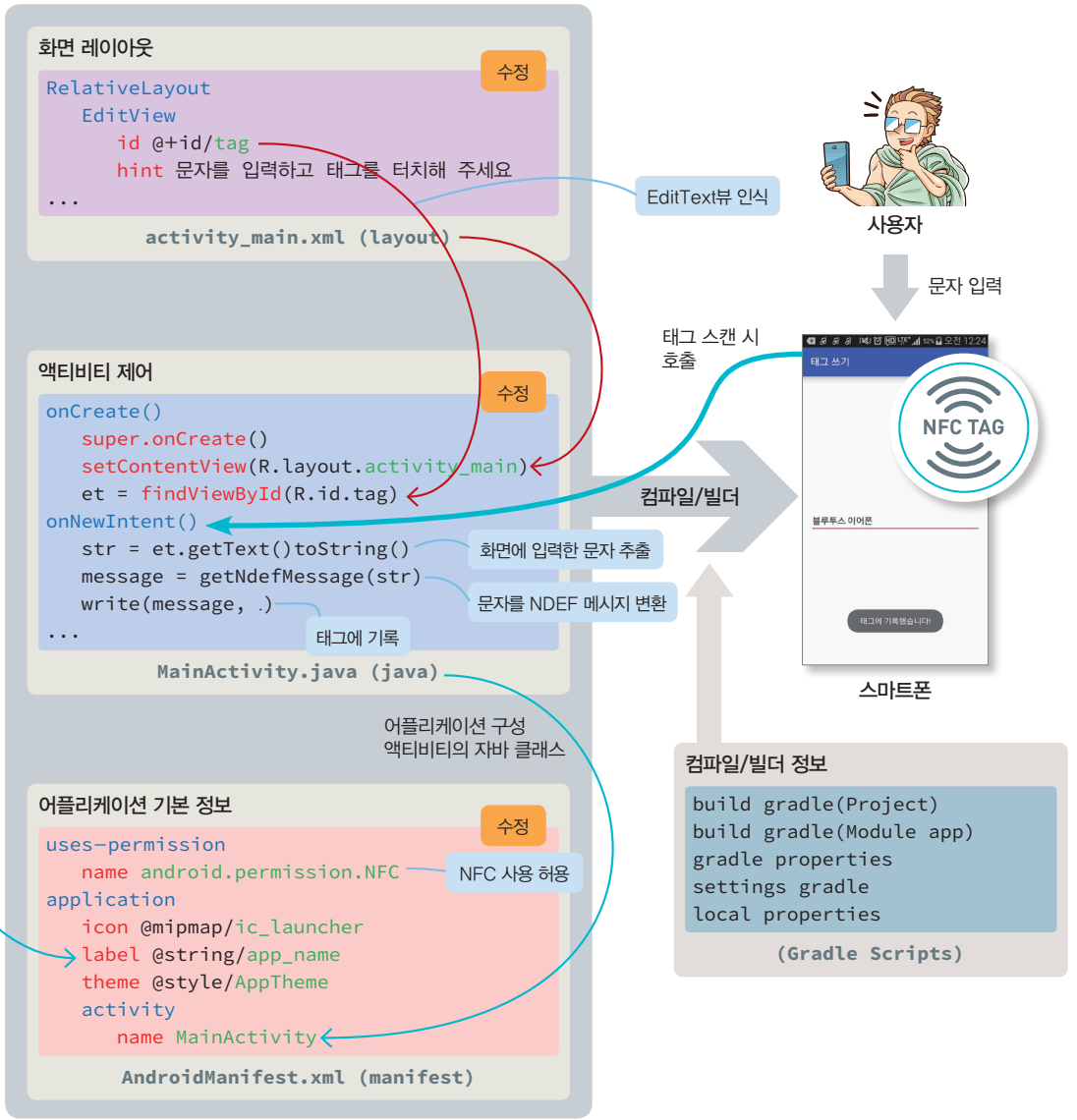
strings.xml에는 초기치로 설정되어 있는 어플리케이션 라벨을 '태그 쓰기'로 수정하고, 태그사용 안내를 위한 속성 값을 추가한다.

activity_main.xml에는 태그 값을 입력하기 위한 EditText뷰를 배치한다.

MainActivity.java에는 입력한 문자를 태그에 기록하도록 한다.

AndroidManifest.xml에는 NFC 사용을 허용하도록 설정한다.




화면 레이아웃
수정
RelativeLayout
EditView
id @+id/tag
hint 문자를 입력하고 태그를 터치해 주세요
...
activity_main.xml (layout)
EditText뷰 인식
사용자
문자 입력
태그 스캔 시 호출
액티비티 제어
수정
onCreate()
super.onCreate()
setContentView(R.layout.activity_main)
et = findViewById(R.id.tag)
onNewIntent()
str = et.getText()toString()
message = getNdefMessage(str)
write(message, .)
...
화면에 입력한 문자 추출
문자를 NDEF 메시지 변환
태그에 기록
MainActivity.java (java)
컴파일/빌더
태그 쓰기
NFC TAG
블루투스 이어폰
태그에 기록했습니다!
스마트폰
어플리케이션 구성
액티비티의 자바 클래스
컴파일/빌더 정보
build gradle(Project)
build gradle(Module app)
gradle properties
settings gradle
local properties
(Gradle Scripts)
어플리케이션 기본 정보
수정
uses-permission
name android.permission.NFC
NFC 사용 허용
application
icon @mipmap/ic_launcher
label @string/app_name
theme @style/AppTheme
activity
name MainActivity
AndroidManifest.xml (manifest)

● **편집**

❶ 텍스트 자원의 편집

어플리케이션 이름을 수정하기 위해 **app_name** 속성값에 해당하는 데이터를 '**태그 쓰기**'로 수정한다.

소스 | strings.xml

```
01 <resources>
02     <string name="app_name">태그 쓰기</string>
03 </resources>
```

app_name의 데이터를 '태그 쓰기'로 수정

❷ 화면 설계

NFC 태그에 쓸 문자를 입력할 수 있는 **EditText**로 배치하고 **ID**를 부여한다. **ID**는 NFC 태그에 쓰기 위해 입력한 문자를 추출할 때 사용된다.

소스 | activity_main.xml — A

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     xmlns:tools="http://schemas.android.com/tools"
04     android:layout_width="match_parent"
05     android:layout_height="match_parent"
06     android:paddingBottom="@dimen/activity_vertical_margin"
07     android:paddingLeft="@dimen/activity_horizontal_margin"
08     android:paddingRight="@dimen/activity_horizontal_margin"
09     android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="com.example.nfcwriter.MainActivity">
11
12     <EditText                                   B
13         android:id="@+id/tag"
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:layout_centerInParent="true"
17         android:hint="문자를 입력하고 태그를 터치하세요!" />
18
19 </RelativeLayout>
```

태그에 쓸 문자열을 입력 받는 EditText 뷰

클래스와 속성/메소드

● **클래스**

클래스	설명
EditText	12행. 편집 가능한 TextView

● **XML 속성**

클래스	메소드	설명
Textview	android:hint	17행. 문자열이 입력되지 않았을 때 출력하는 힌트 문자

❸ 액티비티 제어

onCreate() 메소드는 NFC 어댑터를 생성하고, onResume() 메소드에서는 NFC 어댑터가 foreground에서 사용 가능하도록 한다. 액티비티가 foreground로 작동 중인 경우에 NFC 태그가 스캔 등에 의한 인텐트를 받기 위해서는 **onNewIntent()** 메소드를 이용한다. EditText 뷰에 문자를 입력하고 NFC 태그를 스캔하면 **onNewIntent()** 메소드가 호출되는데 이때 인텐트가 전달된다.

소스 | MainActivity.java

```
01 package com.example.yschang.nfcwriter;
02
03 import android.app.PendingIntent;
04 import android.content.Intent;
05 import android.nfc.NdefMessage;
06 import android.nfc.NdefRecord;
07 import android.nfc.NfcAdapter;
08 import android.nfc.Tag;
09 import android.nfc.tech.Ndef;
10 import android.os.Bundle;
11 import android.support.v7.app.AppCompatActivity;
12 import android.widget.EditText;
13 import android.widget.Toast;
14
15 public class MainActivity extends AppCompatActivity {
16
17    private NfcAdapter nfcAdapter;
18    private PendingIntent pendingIntent;
19    Intent intent;
20    EditText et;
21    boolean mode = false;
22
23    @Override
24    protected void onCreate(Bundle savedInstanceState) {
25       super.onCreate(savedInstanceState);
```

```
26      setContentView(R.layout.activity_main);
27
28      et =(EditText)findViewById(R.id.tag);
29
30      nfcAdapter = NfcAdapter.getDefaultAdapter(this);
31      intent = new Intent(this, getClass()).addFlags(
                                          Intent.FLAG_ACTIVITY_SINGLE_TOP);
32      pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);
33  }
34
35  @Override
36  protected void onNewIntent(Intent intent) {
37      super.onNewIntent(intent);
38
39      Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
40      String str = et.getText().toString();
41      NdefMessage message = getNdefMessage(str);
42      write(message, tagFromIntent);
43  }
44
45  private NdefMessage getNdefMessage(String text) {
46      byte[] textBytes = text.getBytes();
47      NdefRecord textRecord = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
                        "text/plain".getBytes(), new byte[] {}, textBytes);
48      NdefMessage message = new NdefMessage(textRecord);
49      return message;
50  }
51
52  private boolean write(NdefMessage message, Tag tagFromIntent) {
53      try {
54          Ndef ndef = Ndef.get(tagFromIntent);
55          if(ndef != null) {
56              ndef.connect();
57              ndef.writeNdefMessage(message);
58              ndef.close();
59              Toast.makeText(this, "태그에 기록했습니다!", Toast.LENGTH_LONG).
                                                                  show();
60              return true;
61          }
62          return false;
63      } catch(Exception e) {
64          Toast.makeText(this, "태그에 쓰기 실패했습니다!", Toast.LENGTH_LONG).
                                                                  show();
65          return false;
66      }
```

태그에 쓸 내용을 입력할 EditText 뷰 인식

NFC 어댑터를 생성. NFC를 지원하지 않는 디바이스는 null 값 반환

인텐트 객체 생성. NFC 태그 스캔 시에 인텐트의 정보를 현재 액티비티에 전달하기 위해 onNewIntent() 메소드를 호출하도록 설정

전달받은 인텐트에 대한 PendingIntent 객체를 생성. NFC 어댑터를 포그라운드에서 사용하도록 설정할 때 사용

NFC 태그를 스캔하면 호출되며, 인텐트 정보가 전달됨

스마트폰이 스캔한 태그를 반환함

화면에 입력한 문자 추출

문자를 NDEF 메시지로 변환함

NDEF 메시지를 .스마트폰이 스캔한 태그에 기록함

'text'를 NDEF 메시지로 변환함

NDEF 메시지(message)를 태그(tagFromIntent)에 기록함

```
67      }
68
69      public void onResume() {
70          super.onResume();
71          if(nfcAdapter != null) {
72              nfcAdapter.enableForegroundDispatch(this, pendingIntent, null,
                                                                        null);
73          }
74      }
75
76      @Override
77      protected void onPause() {
78          super.onPause();
79          if(nfcAdapter != null) {
80              nfcAdapter.disableForegroundDispatch(this);
81          }
82      }
83  }
```

액티비티가 화면에 나타날 때 실행

다른 액티비티가 시작할 때 실행

클래스와 속성/메소드

● 클래스

클래스/인터페이스	설명
CharSequence	문자열의 순서와 탐색하는 방법을 정의한 인터페이스
EditText	20행. 편집 가능한 한 줄 형태의 텍스트뷰
Exception	63행. 복원 가능한 예외사항을 나타내는 모든 클래스들의 수퍼 클래스
Ndef	54행. 태그에 NDEF 데이터를 기록함
Tag	39,52행. 발견된 NFC 태그를 나타냄

● 상수

클래스	상수	설명
NdefRecord	short TNF_MIME_MEDIA	47행. NDEF 메시지의 record 내의 header(16.2절참조)에 있는 'type name filed'에 정의될 수 있는 값 중 하나로, 데이터가 인터넷 상에서 주고 받는 멀티미디어 데이터에 대한 형식임(MIME 타입)을 의미함
NfcAdapter	String EXTRA_TAG	39행. ACTION_NDEF_DISCOVERED, ACTION_TECH_DISCOVERED, ACTION_TAG_DISCOVERED 인텐트(16.2절 참조)에 대해 발견되는 태그를 포함하는 필수적인 추가적인 사항을 의미함

● **메소드**

클래스	메소드	설명
Intent	〈T extends Parcelable〉 T getParcelableExtra(String name)	39행. intent로부터 data를 추출함
Ndef	static Ndef get(Tag tag)	54행. 주어진 tag(NDEF 호환 가능한 태그)에 대한 Ndef 인스턴스를 반환함
	void close()	58행. 태그에 대해 입출력을 불가능하게 함
	void connect()	56행. Tag 객체에 입출력을 가능하게 함
	void writeNdefMessage(NdefMessage msg)	57행. NFC 태그에 NdefMessage 타입의 msg를 기록함
NdefMessage	NdefMessage(NdefRecord[] records)	41,48행. 하나 또는 여러 NDEF 레코드로부터 NDEF 메시지를 만드는 NdefMessage 클래스의 생성자
NdefRecord	NdefRecord(short tnf, byte[] type, byte[] id, byte[] payload)	47행. NDEF 레코드를 만드는 NdefRecord 클래스의 생성자
CharSequence	String toString()	40행. 문자열로 변환함
EditText	Editable getText()	40행. 텍스트뷰가 출력되는 문자열을 반환함
String	byte[] getBytes()	46,47행. 시스템의 디폴트 문자집합(안드로이드에서는 UTF-8)을 사용하는 암호화된 문자열에 대한 바이트 단위의 배열을 반환함

❺ 환경 설정

'16.2절의 NFC Reader 프로젝트'와 같이 NFC 태그에 대한 read/write를 위한 최소 API 레벨(10), NFC 사용 허용, NFC 인텐트 필터 등을 설정한다

소스 | AndroidManifest.java

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03     package="com.example.yschang.nfcwriter">
04
05     <uses-permission android:name="android.permission.NFC" />
06     <uses-feature android:name="android.hardware.nfc"
                                                android:required="true" />
07
08     <application
```

NFC 사용 허용

앱이 NFC 기능을 사용한다는 의미이며, 'true'로 설정하며 Google Play에서 NFC 기능을 가진 디바이스에서만 이 앱이 보여짐

```
09        android:allowBackup="true"
10        android:icon="@mipmap/ic_launcher"
11        android:label="@string/app_name"
12        android:supportsRtl="true"
13        android:theme="@style/AppTheme">
14        <activity android:name=".MainActivity">
15            <intent-filter>
16                <action android:name="android.intent.action.MAIN" />
17
18                <category android:name="android.intent.category.LAUNCHER" />
19            </intent-filter>
20
21            <intent-filter>
22                <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
23                <category android:name="android.intent.category.DEFAULT"/>
24                <data android:mimeType="text/plain" />
25            </intent-filter>
26        </activity>
27    </application>
28
29 </manifest>
```

'text/plain' MIME 타입의 NFC 태그 스캔 후에 앱이 시작될 수 있도록 설정(반대로 앱을 먼저 실행하고 NFC 태그를 스캔할 경우에는 intent-filter 부분을 설정하지 않아도 됨)

STEP 3 프로젝트 실행

다음 2단계 절차에 따라 프로젝트를 실행하고 그 결과를 살펴보자.

절차	내용
실행메뉴 선택	'Run' 메뉴에서 'Run app' 클릭(또는 'Run app' 아이콘 클릭)
디바이스 선택	스마트폰 디바이스를 선택하고 'OK' 버튼 클릭

다음에는 NFC 태그를 활용한 예로서, 태그를 스캔하면 웹페이지로 이동하는 방법을 살펴보자.

16.4 NFC 태그 읽기와 웹페이지 오픈: NFC Web(NFC 웹)

16.4.1 프로젝트 개요

홍보 포스터 등에 부착된 태그를 터치하면 태그에 기록된 URL(예: http://www.android.com)로 이동하는 앱을 개발해보자. **어플리케이션 이름**은 『**NFC Web**』, **어플리케이션 라벨**과 **액티비티 라벨**은 『**태그 웹**』으로 한다.

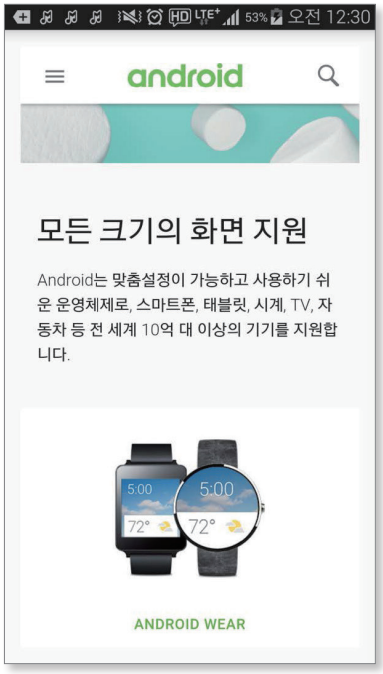
프로젝트 개요: 태그에 기록된 URL로 이동하기

Application Name: NFC Web

어플리케이션 라벨: 태그 웹

액티비티1(태그 스캔)

액티비티2(웹페이지 출력)



16.4.2 프로젝트 개발

STEP 1 프로젝트 생성

다음의 5단계 절차에 따라 프로젝트를 실행한다.

절차	내용
❶ 프로젝트 시작	메뉴에서 'File → New Project' 클릭
❷ 프로젝트 구성	Application Name: NFC Web Company Domain: yschang.example.com
❸ 제품형태	Phone and Tablet
❹ 액티비티 유형	Empty Activity
❺ 파일 옵션	디폴트 값으로 설정

STEP 2 파일 편집

● **파일 구조와 편집 내용**

모듈	폴더	소스 파일	편집 내용	관련 액티비티
manifests		AndroidManifest.xml	• NFC 사용 허용	태그 읽기, 웹사이트 출력
java	com.example. yschang. nfc web	MainActivity.java	• 태그 내용 읽기 • WebLoad 클래스 호출	태그 읽기
		Webload.java	• 웹페이지 출력	웹사이트 출력
res	drawable			
	layout	activity_main.xml	• 태그 내용 출력 텍스트뷰 배치 • 웹페이지 출력으로 이동하기 위한 버튼 배치	태그 읽기
		web.xml	• 웹뷰 배치	웹사이트 출력
	mipmap	ic_launcher.png		
	values	dimens.xml		
		strings.xml	• 어플리케이션 라벨 수정	태그 읽기
		styles.xml		

■ 수정 ■ 추가

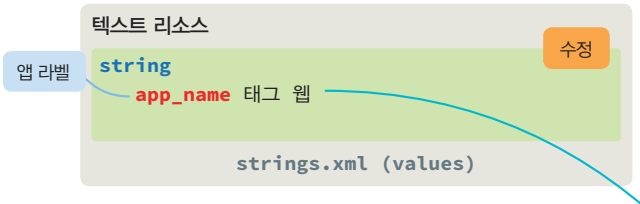
● 파일 간의 연관관계

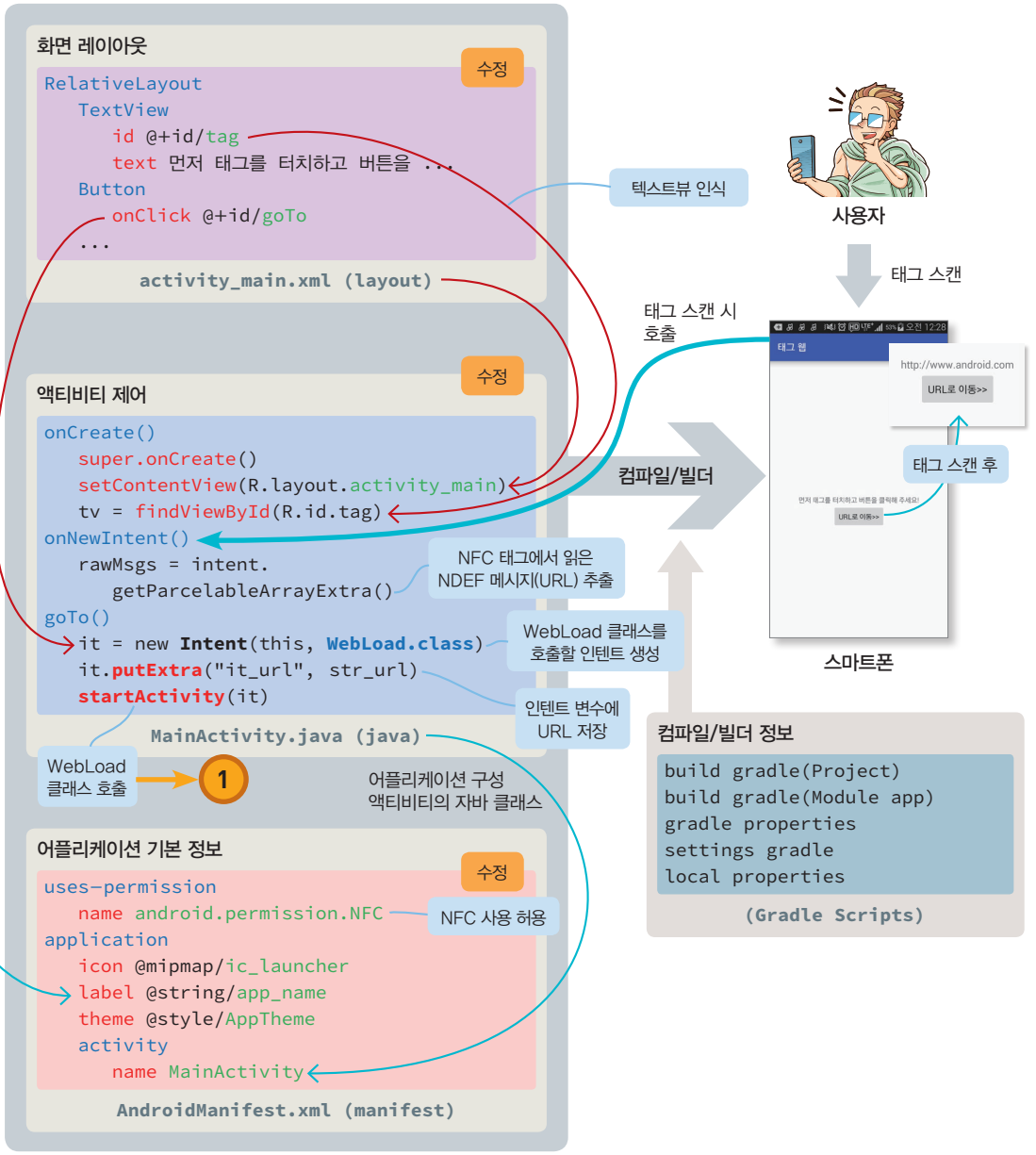
strings.xml에는 초기치로 설정되어 있는 어플리케이션 라벨을 '태그 웹'으로 수정한다.

activity_main.xml에는 태그를 읽은 값을 출력할 텍스트뷰와 WebLoad 클래스를 호출하기 위한 버튼을 배치한다.

MainActivity.java에는 버튼이 클릭되면 읽은 태그 값을 인텐트 변수에 저장하고 WebLoad를 호출한다.

AndroidManifest.xml에는 NFC 사용을 허용하도록 설정한다.




화면 레이아웃
수정
RelativeLayout
TextView
id @+id/tag
text 먼저 태그를 터치하고 버튼을 ...
Button
onClick @+id/goTo
...
activity_main.xml (layout)
텍스트뷰 인식
사용자
태그 스캔
태그 스캔 시 호출
액티비티 제어
수정
onCreate()
super.onCreate()
setContentView(R.layout.activity_main)
tv = findViewById(R.id.tag)
onNewIntent()
rawMsgs = intent.
getParcelableArrayExtra()
NFC 태그에서 읽은 NDEF 메시지(URL) 추출
goTo()
it = new Intent(this, WebLoad.class)
WebLoad 클래스를 호출할 인텐트 생성
it.putExtra("it_url", str_url)
startActivity(it)
인텐트 변수에 URL 저장
MainActivity.java (java)
WebLoad 클래스 호출
1
어플리케이션 구성 액티비티의 자바 클래스
컴파일/빌더
태그 웹
http://www.android.com
URL로 이동>>
태그 스캔 후
스마트폰
컴파일/빌더 정보
build gradle(Project)
build gradle(Module app)
gradle properties
settings gradle
local properties
(Gradle Scripts)
어플리케이션 기본 정보
수정
uses-permission
name android.permission.NFC
NFC 사용 허용
application
icon @mipmap/ic_launcher
label @string/app_name
theme @style/AppTheme
activity
name MainActivity
AndroidManifest.xml (manifest)

● **편집(태그 읽기 액티비티)**

❶ 텍스트 자원의 편집

어플리케이션 이름을 수정하기 위해 **app_name** 속성값에 해당하는 데이터를 '**태그 웹**'으로 수정한다.

소스 | strings.xml

```
01 <resources>
02     <string name="app_name">태그 웹</string>
03 </resources>
```

app_name의 데이터를 '태그 웹'으로 수정

❷ 화면 설계

NFC 태그에 쓸 문자를 입력할 수 있는 **EditText**로 배치하고 **ID**를 부여한다. **ID**는 NFC 태그에 쓰기 위해 입력한 문자를 추출할 때 사용된다.

소스 | activity_main.xml → A

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     xmlns:tools="http://schemas.android.com/tools"
04     android:layout_width="match_parent"
05     android:layout_height="match_parent"
06     android:paddingBottom="@dimen/activity_vertical_margin"
07     android:paddingLeft="@dimen/activity_horizontal_margin"
08     android:paddingRight="@dimen/activity_horizontal_margin"
09     android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="com.example.nfcweb.MainActivity">
11
12     <TextView                                          ← B
13         android:id="@+id/tag"
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:layout_centerInParent="true"
17         android:text="먼저 태그를 터치하고 버튼을 클릭해 주세요!" />
18
19     <Button
20         android:layout_width="wrap_content"
21         android:layout_height="wrap_content"
22         android:text="URL로 이동>>"
23         android:layout_below="@id/tag" />
```

태그 터치 안내 문자열

태그 내용(URL)을 전송하기 위한 버튼

ID가 tag인 뷰 아래에 배치

```
24      android:layout_centerHorizontal="true"
25      android:onClick="goTo" />
26
27  </RelativeLayout>
```

뷰를 수평 중앙에 배치

C

뷰를 클릭하면 goTo() 메소드를 호출함

클래스와 속성/메소드

● 클래스

클래스	설명
Button	19행. 푸쉬 버튼 위젯을 표현함

클래스	메소드	설명
RelativeLayout.LayoutParams	android:layout_below	23행. 뷰의 윗면을 지정된 ID의 뷰 아래에 배치

❸ 액티비티 제어

onCreate() 메소드는 NFC 어댑터를 생성하고, onResume() 메소드에서는 NFC 어댑터가 foreground에서 사용 가능하도록 한다. 액티비티가 foreground로 작동 중인 경우에 NFC 태그가 스캔 등에 의한 인텐트를 받기 위해서는 **onNewIntent()** 메소드를 이용한다. EditText 뷰에 문자를 입력하고 NFC 태그를 스캔하면 **onNewIntent()** 메소드가 호출되는데 이때 인텐트가 전달된다.

소스 | MainActivity.java

```
01 package com.example.yschang.nfcweb;
02
03 import android.app.PendingIntent;
04 import android.content.Intent;
05 import android.nfc.NdefMessage;
06 import android.nfc.NdefRecord;
07 import android.nfc.NfcAdapter;
08 import android.os.Bundle;
09 import android.os.Parcelable;
10 import android.support.v7.app.AppCompatActivity;
11 import android.view.View;
12 import android.widget.TextView;
```

```
13
14 public class MainActivity extends AppCompatActivity {
15
16    private NfcAdapter nfcAdapter;
17    private PendingIntent pendingIntent;
18    TextView tv;
19
20    @Override
21    protected void onCreate(Bundle savedInstanceState) {
22       super.onCreate(savedInstanceState);
23       setContentView(R.layout.activity_main);
24
25       tv =(TextView)findViewById(R.id.tag);
26
27       nfcAdapter = NfcAdapter.getDefaultAdapter(this);
28       Intent intent = new Intent(this, getClass()).addFlags(Intent.FLAG_
                                                    ACTIVITY_SINGLE_TOP);
29       pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);
30    }
31
32    @Override
33    protected void onResume() {
34       super.onResume();
35       if(nfcAdapter != null) {
36          nfcAdapter.enableForegroundDispatch(this, pendingIntent, null,
                                                                  null);
37       }
38    }
39
40    @Override
41    protected void onPause() {
42       super.onPause();
43       if(nfcAdapter != null) {
44          nfcAdapter.disableForegroundDispatch(this);
45       }
46    }
47
48    @Override
49    protected void onNewIntent(Intent intent) {
50       super.onNewIntent(intent);
51
52       Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.
                                                    EXTRA_NDEF_MESSAGES);
53
54       if(rawMsgs != null) {
```

```
55          NdefMessage msgs =(NdefMessage) rawMsgs[0];
56          NdefRecord[] rec = msgs.getRecords();
57          byte[] bt = rec[0].getPayload();
58          String text = new String(bt);
59          tv.setText(text);
60       }
61    }
62
63    public void goTo(View v) {
64
65       String str_url = tv.getText().toString();
66       Intent it = new Intent(this, WebLoad.class);
67       it.putExtra("it_url",    str_url);
68       startActivity(it);
69       finish();
70
71    }
72 }
```

태그 내용(URL)을 텍스트뷰에 할당

버튼 클릭 시 호출되는 콜백 메소드

텍스트뷰에 할당된 문자열(URL) 추출

WebLoad 클래스를 호출할 인텐트 생성

인텐트 변수에 URL 저장

WebLoad 클래스의 액티비티 호출

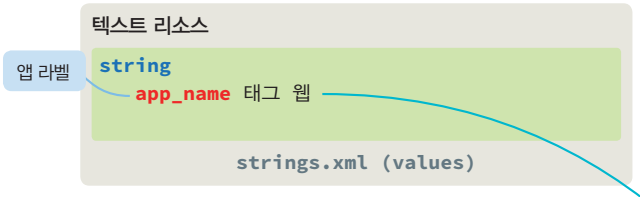
현재 액티비티 종료

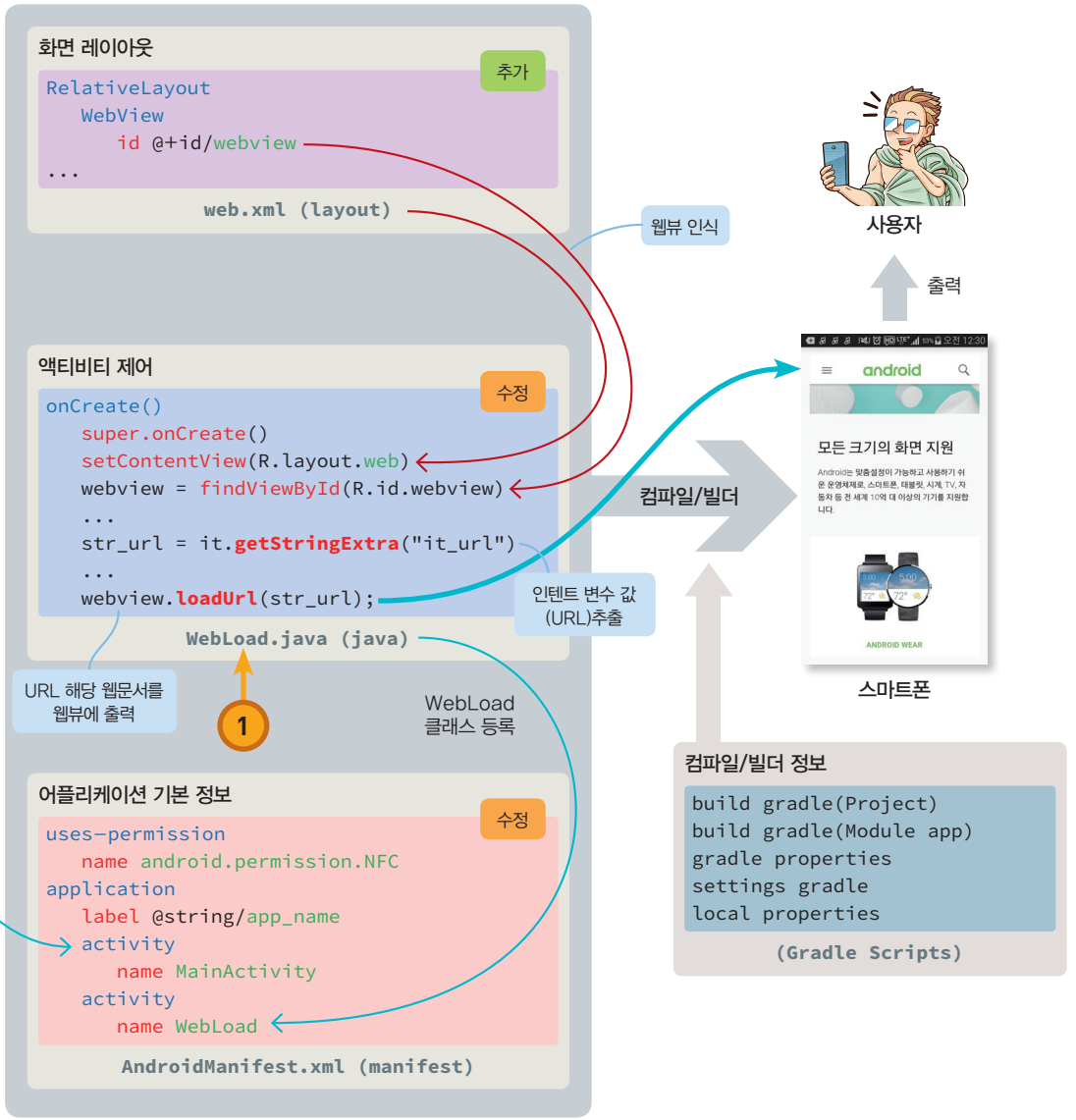
● **파일 간의 연관관계**

web.xml에는 웹문서를 출력할 웹뷰를 배치한다.

WebLoad.java는 인텐트 변수 값을 추출하고 웹문서를 출력한다.

AndroidManifest.xml에는 WebLoad 클래스를 등록한다.




화면 레이아웃
추가
RelativeLayout
WebView
id @+id/webview
...
web.xml (layout)
웹뷰 인식
액티비티 제어
수정
onCreate()
super.onCreate()
setContentView(R.layout.web)
webview = findViewById(R.id.webview)
...
str_url = it.getStringExtra("it_url")
...
webview.loadUrl(str_url);
인텐트 변수 값
(URL)추출
WebLoad.java (java)
URL 해당 웹문서를
웹뷰에 출력
1
WebLoad
클래스 등록
어플리케이션 기본 정보
수정
uses-permission
name android.permission.NFC
application
label @string/app_name
activity
name MainActivity
activity
name WebLoad
AndroidManifest.xml (manifest)
사용자
출력
android
모든 크기의 화면 지원
ANDROID WEAR
스마트폰
컴파일/빌더
컴파일/빌더 정보
build gradle(Project)
build gradle(Module app)
gradle properties
settings gradle
local properties
(Gradle Scripts)

● **편집(웹페이지 출력 액티비티)**

❶ 화면 설계

웹페이지 출력을 위해 웹뷰를 배치한다.

소스 | web.xml → D

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     xmlns:tools="http://schemas.android.com/tools"
04     android:layout_width="match_parent"
05     android:layout_height="match_parent"
06     android:paddingBottom="@dimen/activity_vertical_margin"
07     android:paddingLeft="@dimen/activity_horizontal_margin"
08     android:paddingRight="@dimen/activity_horizontal_margin"
09     android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="com.example.nfcweb.MainActivity">
11
12     <WebView                                    E
13         android:id="@+id/webview"
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content" />
16
17 </RelativeLayout>
```

웹문서를 출력할 웹뷰 배치

클래스와 속성/메소드

● **클래스**

클래스	설명
WebView	12행. 웹페이지를 출력하는 뷰

❷ 액티비티 제어

인텐트를 통해 전달된 데이터(URL)를 추출하고, WebView에 웹페이지를 출력한다.

소스 | WebLoad.java → F

```
01 package com.example.yschang.nfcweb;
02
03 import android.app.Activity;
```

```
04 import android.content.Intent;
05 import android.os.Bundle;
06 import android.webkit.WebSettings;
07 import android.webkit.WebView;
08 import android.webkit.WebViewClient;
09
10 public class WebLoad extends Activity {
11
12    protected void onCreate(Bundle savedInstanceState) {
13       super.onCreate(savedInstanceState);
14       setContentView(R.layout.web);
15
16       Intent it = getIntent();
17       String str_url   = it.getStringExtra("it_url");
18
19       WebView webview =(WebView)findViewById(R.id.webview);
20       webview.setWebViewClient(new WebViewClient());
21       WebSettings set = webview.getSettings();
22       set.setJavaScriptEnabled(true);
23       set.setBuiltInZoomControls(true);
24
25       webview.loadUrl(str_url);
26    }
27 }
```

D

현재 액티비티를 호출한 인텐트 객체 생성

인텐트 변수에 저장된 URL 추출

URL 해당 웹문서를 웹뷰에 출력 (17장 참조)

클래스와 속성/메소드

● 클래스

클래스/인터페이스	설명
WebView	19행. 웹페이지를 출력하는 뷰
WebViewClient	20행. 여러 알림과 요청을 받음
WebSettings	21행. WebView 설정 상태를 관리함. WebView가 처음 만들어질 때는 디폴트 설정 값을 가짐. WebSettings의 객체는 WebView.getSettings()에 의해 만들어짐

● 메소드

클래스	메소드	설명
webView	WebSettings getSettings()	21행. WebView 설정을 제어하기 위한 WebSettngs 객체를 가져옴
	void loadUrl(String url)	25행. 주어진 URL을 로딩함

WebSettings	abstract void setBuiltInZoomControls(boolean enabled)	23행. 웹뷰에 줌 기능 사용 여부를 설정함
	synchronized void setJavaScriptEnabled(boolean flag)	22행. WebView가 자바 스크립트 실행이 가능하도록 설정함
	void setWebViewClient(WebViewClient client)	20행. 여러 알림과 요청을 받게될 WebViewClient를 설정함

❸ 환경 설정

'16.2절의 NFC Reader 프로젝트'와 같이 NFC 태그에 대한 read/write를 위한 최소 API 레벨(10), NFC 사용 허용, NFC 인텐트 필터 등을 설정한다

소스 | AndroidManifest.java

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03     package="com.example.yschang.nfcweb">
04
05     <uses-permission android:name="android.permission.INTERNET" />
06     <uses-permission android:name="android.permission.NFC" />
07     <uses-feature android:name="android.hardware.nfc"
                  android:required="true" />
08
09     <application
10         android:allowBackup="true"
11         android:icon="@mipmap/ic_launcher"
12         android:label="@string/app_name"
13         android:supportsRtl="true"
14         android:theme="@style/AppTheme">
15         <activity android:name=".MainActivity">
16             <intent-filter>
17                 <action android:name="android.intent.action.MAIN" />
18
19                 <category android:name="android.intent.category.LAUNCHER" />
20             </intent-filter>
21
22             <intent-filter>
23                 <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
24                 <category android:name="android.intent.category.DEFAULT"/>
```

인터넷 사용 허용

NFC 사용 허용

```
25              <data android:mimeType="text/plain" />
26          </intent-filter>
27      </activity>
28
29      <activity
30          android:name=".WebLoad" >
31      </activity>
32  </application>
33
34 </manifest>
```

F

WebLoad 클래스에 의한 액티비티 등록.

STEP 3 프로젝트 실행

다음 2단계 절차에 따라 프로젝트를 실행하고 그 결과를 살펴보자.

절차	내용
실행메뉴 선택	'Run' 메뉴에서 'Run app' 클릭(또는 'Run app' 아이콘 클릭)
디바이스 선택	스마트폰 디바이스를 선택하고 'OK' 버튼 클릭

이제 네트워크 환경을 이용하는 방법으로, 조금 전에 살펴 본 웹페이지 출력을 포함하여 공공데이터와 구글맵을 활용하는 방법을 살펴보기로 하자.