NFCDemo 예제 분석

2011.11.27

Café.naver.com/android21

김태성

NdefMessage구조

•NDEF란?
- •NFC Data Exchange Format
- •NFC 데이터 통신 프레임 형식

•NdefMessage구조
- 단일 Record 구조(Record 1개)

NdfRecord[0]

- 다중 Record 구조

NdefRecord[0]	NdefRecord[1]	NdefRecord[2]	…

•Record
- -내부 foramt에 따라 Uri, Text, SmartPoster등 규격에 맞게 여러 가지 정보 포함 가능
- -위 3가지 이외에도 규격서를 보면 더 다양한 format 존재

•url 참고
- http://www.nfc-forum.org/specs/

주요 클래스

- SmartPoster, TextRecord, UriRecord Class
  - Record에 담긴 정보의 형식을 나타내는 클래스
  - 소스코드에서 주요한 부분
    - parse 메소드
    - getView 메소드

AndroidManifest.xml 1/2

```
<?xml version="1.0" encoding="utf-8" ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.android.nfc">
<!-- NFC 사용 권한, 전화걸기 권한 허가 -->
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.CALL_PHONE" />

<application android:icon="@drawable/icon"
android:label="@string/app_name">
<!– 메인 액티비티 FakeTagsActivity 등록-->
<activity android:name=".simulator.FakeTagsActivity"
android:theme="@android:style/Theme.NoTitleBar">

<intent-filter>
   <action android:name="android.intent.action.MAIN" />
   <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>

</activity>
```

AndroidManifest.xml 2/2

```
<!– 태그 정보를 표시할 액티비티 TagViewer 등록-->
<activity android:name="TagViewer"
android:theme="@android:style/Theme.NoTitleBar">

<!– 태그를 인식 했을 때 표시되는 액티비티->
<intent-filter>
   <action android:name="android.nfc.action.TAG_DISCOVERED" />
   <category android:name="android.intent.category.DEFAULT" />
</intent-filter>

</activity>

</application>
<!– 2.3 진저브레드 이상에서 NFC 지원 -->
<uses-sdk android:minSdkVersion="9" />
<!– NFC 디바이스 사용 등록-->
<uses-feature android:name="android.hardware.nfc" android:required="true"
/>

</manifest>
```

res\layout\tag_divider.xml

```
<?xml version="1.0" encoding="utf-8" ?>

<!– 리스트 항목에 표시될 레이아웃 ->
<View xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="?android:attr/listDivider"
/>
```

res\layout\tag_text.xml

```
<?xml version="1.0" encoding="utf-8" ?>

<!– Text 태그를 인식 했을 때 표시할 레이아웃->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/text“
android:layout_width="match_parent"
android:minHeight="?android:attr/listPreferredItemHeight"
android:layout_height="wrap_content"
android:padding="4dip"
android:textAppearance="?android:attr/textAppearanceMedium"
android:maxLines="5"
android:ellipsize="end"
android:gravity="center_vertical"
/>
```

res\layout\tag_viewer.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<!– 태그 뷰어 액티비티의 레이아웃-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent" android:layout_height="match_parent"
android:orientation="vertical">


<!– 태그 타이틀 한줄 표시 -->
<TextView android:id="@+id/title" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:singleLine="true"
android:textAppearance="?android:attr/textAppearanceMedium"
android:textStyle="bold" android:shadowColor="#BB000000"
android:shadowRadius="2.75" android:gravity="center_vertical" />


 <!– 스크롤 가능하게 내용 표시 -->
<ScrollView android:layout_width="match_parent" android:layout_height="0dip"
android:layout_weight="1">
<LinearLayout android:id="@+id/list" android:layout_width="match_parent"
android:layout_height="wrap_content" android:orientation="vertical" />
</ScrollView>

</LinearLayout>
```

사운드 리소스 파일

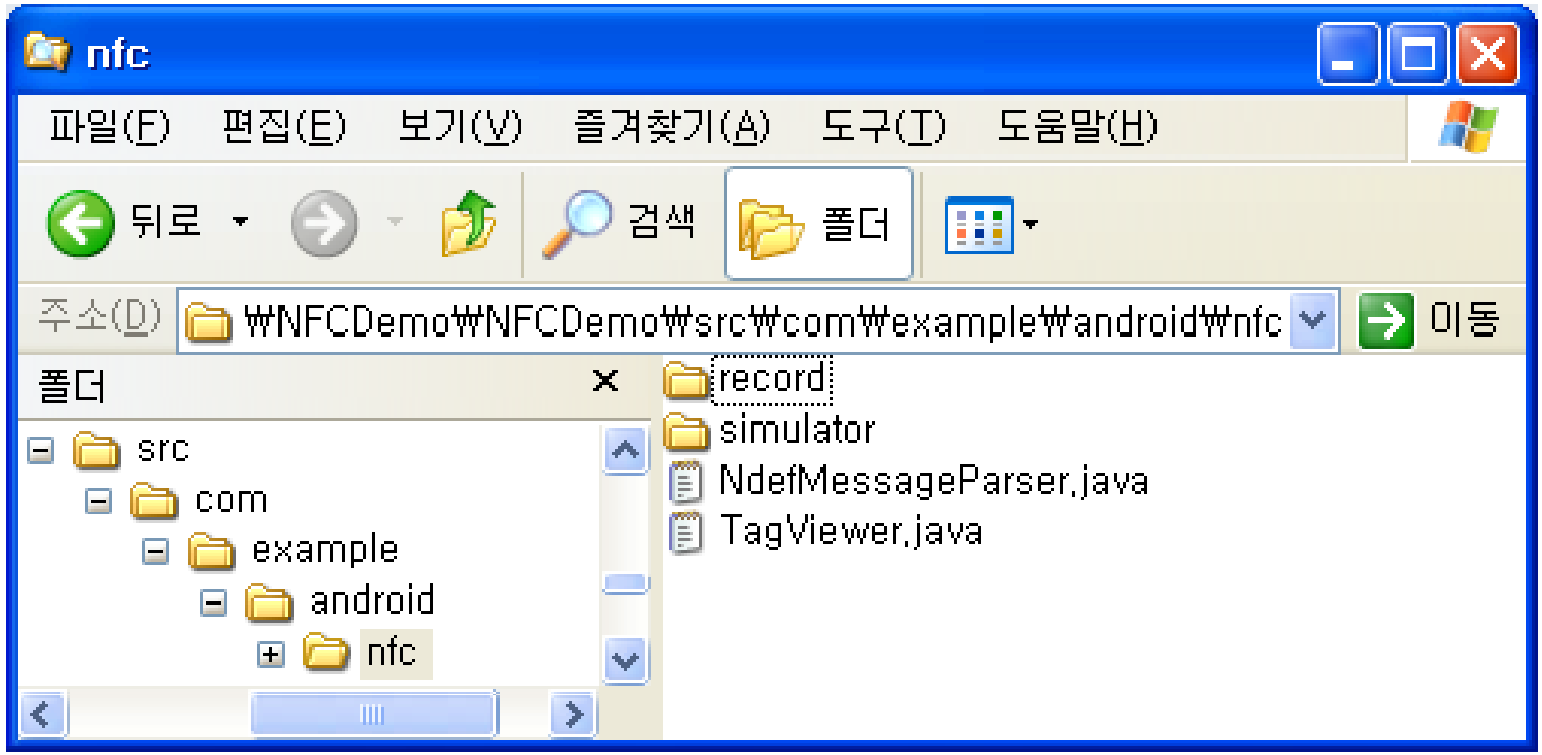
- 태그인식 사운드 파일
  - 태그를 인식 했을 때 플레이할 사운드
  - res₩raw₩discovered_tag_notification.ogg

\res\values\strings.xml

```
<?xml version="1.0" encoding="utf-8" ?>

<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">

<!– 앱의 이름   -->
<string name="app_name">NFCDemo</string>

<!– 태그가 인식 되었을 때 타이틀바에 표시할 액티비티의 이름    -->
<string name="title_scanned_tag">New tag collected</string>

<!– 텍스트형 태그의 제목   -->
<string name="tag_text">Text</string>

</resources>
```

\src\com\example\android\nfc\

- 소스 파일 및 디렉토리
  - record : 태그 데이터 형식 정보 포함 디렉토리
  - simulator : 메인 액티비티 포함
  - NdefMeeesgeParser : 인식된 Tag의 데이터 분석 클래스
  - TagViewer : 인식된 태그 표시 액티비티



NFCDemo

소스 코드 분석

인식된 태그 정보 표시 액티비티

TAGVIEWER.JAVA

TagViewer.java

```
016: package com.example.android.nfc;
017: // 태그가 인식 되면 정보를 보여주는 액티비티
018: import android.app.Activity;
019: import android.content.Intent;
020: import android.nfc.NdefMessage;
021: import android.nfc.NdefRecord;
022: import android.nfc.NfcAdapter;
023: import android.os.Bundle;
024: import android.os.Parcelable;
025: import android.util.Log;
026: import android.view.LayoutInflater;
027: import android.view.WindowManager;
028: import android.widget.LinearLayout;
029: import android.widget.TextView;
030:
031: import com.example.android.nfc.record.ParsedNdefRecord; // 사용자구현
032:
033: import java.util.List;
034:
```

TagViewer.java

```
035: /**
036:  * 스마트폰으로 방금 인식된 새 태그의 브로드 캐스트 관리 액티비티
038:  */
039: public class TagViewer extends Activity {
040:
041:     static final String TAG = "ViewTag";
042:
043:     /**
044:      * 이 액티비티는 사용자가 아무 것도 하지 않은 경우 자동 종료
046:      */
047:     static final int ACTIVITY_TIMEOUT_MS = 1 * 1000; // 1초 대기
048:
049:     TextView mTitle;   // 태그의 타이틀 표시
050:
051:     LinearLayout mTagContent;      // 태그의 내용 표시
052:
053:     @Override
054:     protected void onCreate(Bundle savedInstanceState) { // 생성자
055:         super.onCreate(savedInstanceState);
056:         setContentView(R.layout.tag_viewer);
057:         mTagContent   = (LinearLayout) findViewById(R.id.list); // 태그 내용 표시
058:         mTitle            = (TextView) findViewById(R.id.title); // 태그 타이틀 표시
059:         resolveIntent(getIntent());      // 이 액티비티가 실행될 때 받았던 인텐트 분석 후 처리 사용자 메소드
060:     }
061:
```

TagViewer.java

```
062:    void resolveIntent(Intent intent) {
063:        // 인텐트 파싱
064:        String action = intent.getAction();
065:        if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)) {
066:            // 태그가 인식 되면 저장하기 위해 서비스를 호출한다.
067:            // PendingIntent를 같이 보내서 다시 내가 불리게 한다.
068:            // 다시 불릴때는 onNewIntent() 호출되는데
069:            // 그때 DB에서 읽어서 화면에 표시한다.
070:            Parcelable[] rawMsgs = intent.getParcelableArrayExtra(
                                            NfcAdapter.EXTRA_NDEF_MESSAGES);
071:            NdefMessage[] msgs;
072:            if (rawMsgs != null) {
073:                msgs = new NdefMessage[rawMsgs.length];
074:                for (int i = 0; i < rawMsgs.length; i++) {   // 모든 태그 메시지 빼내기
075:                    msgs[i] = (NdefMessage) rawMsgs[i];
076:                }
```

TagViewer.java

```
077:            } else {
078:                // 알 수 없는 태그 형식일 경우 표시할 메시지 준비
079:                byte[] empty = new byte[] {};
080:                NdefRecord record = new NdefRecord(
                                    NdefRecord.TNF_UNKNOWN, empty, empty, empty);
081:                NdefMessage msg = new NdefMessage(new NdefRecord[] {record});
082:                msgs = new NdefMessage[] {msg};
083:            }
084:             //  뷰 설정 하기
085:            setTitle(R.string.title_scanned_tag);  // 인텐트에서 빼낸 정보 화면 표시
086:            buildTagViews( msgs );    // 뷰에 태그 표시 사용자 메소드
087:        } else {
088:            Log.e(TAG, "Unknown intent " + intent);
089:            finish();
090:            return;
091:        }
092:    }
093:
```

TagViewer.java

```
094:    void buildTagViews(NdefMessage[] msgs) {          // 태그 정보 화면 표시
095:        if (msgs == null || msgs.length == 0) {
096:            return;
097:        }
098:        LayoutInflater inflater = LayoutInflater.from(this);
099:        LinearLayout content = mTagContent;
100:        // 연속으로 두 개 태그를 인식 할 수 있기 때문에
101:        // 내용 표시 영역에 이전 내용을 지운다.
102:        content.removeAllViews();
103:        // 리스트에 첫 번째 메시지를 Parse
104:        // 모든 sub 레코드의 뷰들을 생성
105:        List<ParsedNdefRecord> records = NdefMessageParser.parse(msgs[0]);
106:        final int size = records.size();
107:        for (int i = 0; i < size; i++) {// 모든 나머지 레코드에 대해
108:            ParsedNdefRecord record = records.get(i);
109:            content.addView(record.getView(this, inflater, content, i));
110:            inflater.inflate(R.layout.tag_divider, content, true);        // 뷰 생성
111:        }
112:    }
```

TagViewer.java

```
113:
114:     @Override
115:     public void onNewIntent(Intent intent) { // 새인텐츠가 왔을때
116:         setIntent(intent); // 현재 인텐츠로 설정하고 처리
117:         resolveIntent(intent);
118:     }
119:
120:     @Override
121:     public void setTitle(CharSequence title) { // 타이틀 설정
122:         mTitle.setText(title);
123:     }
124: }
```

태그 메시지 파서 클래스

NDEF MESSAGE PARSER.JAVA

NdefMessageParser.java

```
016: package com.example.android.nfc;
017:
018: import android.nfc.NdefMessage;
019: import android.nfc.NdefRecord;
020:
021: import com.example.android.nfc.record.ParsedNdefRecord;
022: import com.example.android.nfc.record.SmartPoster;
023: import com.example.android.nfc.record.TextRecord;
024: import com.example.android.nfc.record.UriRecord;
025:
026: import java.util.ArrayList;
027: import java.util.List;
028:
029: /**
030:  * ParsedNdefMessage들을 만드는 유틸리티 클래스
031:  */
032: public class NdefMessageParser {
033:
034:     // 생성자에서 하는일은 없음
035:     private NdefMessageParser() {
036:
037:     }
038:
```

NdefMessageParser.java

```
039:     /** NdefMessage 파싱하기*/
040:     public static List<ParsedNdefRecord> parse(NdefMessage message) {
041:         return getRecords(message.getRecords());
042:     } // NDEF 메시지에 있는 모든 레코드를 해체해서 리스트로 반환한다.
043:
044:     public static List<ParsedNdefRecord> getRecords(NdefRecord[] records) {
045:         List<ParsedNdefRecord> elements = new ArrayList<ParsedNdefRecord>();
046:         for (NdefRecord record : records) {     // 모든 레코드에 대해
047:             if (UriRecord.isUri(record)) {                    // URI 인 경우
048:                 elements.add(UriRecord.parse(record));
049:             } else if (TextRecord.isText(record)) {           // 단순 텍스트인 경우
050:                 elements.add(TextRecord.parse(record));
051:             } else if (SmartPoster.isPoster(record)) {        // 스마트포스터 인 경우
052:                 elements.add(SmartPoster.parse(record));
053:             }
054:         }
055:         return elements;
056:     }
057: }
```

인터페이스

\RECORD\PARSEDNDEFRECORD.JAVA

\record\ParsedNdefRecord.java

```
017: package com.example.android.nfc.record;
018:
019: import android.app.Activity;
020: import android.view.LayoutInflater;
021: import android.view.View;
022: import android.view.ViewGroup;
023:
024:
025: public interface ParsedNdefRecord {          // URI든 Text든 Poster든 공통 구현
026:
027:     /**
028:      * 현재 레코드를 화면에 표시하기 위해 반환
029:      */
030:     public View getView(Activity activity, LayoutInflater inflater, ViewGroup parent,
031:             int offset);
032:
033: }
```

태그 레코드 형식

\RECORD\ SMART POSTER .JAVA

\record\SmartPoster.java

```
016: package com.example.android.nfc.record;
017:
018: import android.app.Activity;
019: import android.nfc.FormatException;  // NFC 패키지
020: import android.nfc.NdefMessage;
021: import android.nfc.NdefRecord;
022: import android.view.LayoutInflater;
023: import android.view.View;
024: import android.view.ViewGroup;
025: import android.view.ViewGroup.LayoutParams;
026: import android.widget.LinearLayout;
027:
028: import com.google.common.base.Charsets;       // 구아바 패키지
029: import com.google.common.base.Preconditions;
030: import com.google.common.collect.ImmutableMap;
031: import com.google.common.collect.Iterables;
032:
033: import com.example.android.nfc.NdefMessageParser;
034: import com.example.android.nfc.R;
035:
036: import java.util.Arrays;
037: import java.util.NoSuchElementException;
038:
```

\record\SmartPoster.java

```
039: /**
040:  * NFC 포럼의 "Smart Poster"에 대한 설명
041:  */
042: public class SmartPoster implements ParsedNdefRecord {
043:
044:     /**
045:      * NFC Forum의 Smart Poster 레코드 형식은 section 3.2.1. 에 정의
046:      *
047:      * 서비스에 대한 Title record인데, 서비스는 여러 가지 언어를 사용할 수 있지만
048:      * 언어 필드는 절대 반복되어 전송해서는 안됨.
049:      * 이 record는 옵션이므로 사용하지 않아도 됨.
050:      */
051:     private final TextRecord mTitleRecord;
052:
053:     /**
054:      * NFC Forum의 Smart Poster Record 형식은 section 3.2.1. 에 정의
055:      *
056:      * URI record는 Smart Poster의 핵심이다.
057:      * 모든 다른 레코드들은 이 레코드에 대한 meta(보조) 데이터이다.
058:      * 반드시 하나의 URI 레코드만 있어야 하며 하나를 넘어도 안 된다.
 059:      */
060:     private final UriRecord mUriRecord;
061:
```

\record\SmartPoster.java

```
062:     /**
063:      * NFC Forum의 Smart Poster Record 형식 section 3.2.1.에 정의
064:      *
065:      * Action record는 서비스가 어떻게 처리되어야 하는지는 나타낸다.
066:      * 예를 들면 action을 디바이스가 URI의 북 마크를 저장하거나 웹 브라우저를
067:      * 열어야 한다는 것을 나타낼 수 있다. Action record는 옵션이다.
068:      * 만약 없을 경우 디바이스는 서비스를 통해 어떻게 처리할지 결정한다.
069:      * 만약 액션 레코드가 있으면, 가장 우선적으로 처리한다.
070:      * UI 디자이너가 이를 무시할 수 있지만,
071:      * 그러면 디바이스 사이의 사용자 편의에 차이가 발생할 수 있다.
072:      */
073:     private final RecommendedAction mAction;
074:
075:     /**
076:      * NFC Forum의 Smart Poster Record 형식 section 3.2.1.에 정의
077:      *
078:      * Type record 변수인데, 만약 URI가 URL처럼 외부에 접근한다면
079:      * Type record는 그것을 가리키도록 선언할 수 있다.
080:      * 이것은 디바이스에게 연결을 하기 전에 어떤 종류의 객체인지 알려주는데 사용할 수 있다.
081:      * 이 Type record는 옵션이다.
082:      */
083:     private final String mType;
084:
```

\record\SmartPoster.java

```
085:     private SmartPoster(        // 생성자
             UriRecord uri, TextRecord title, RecommendedAction action, String type) {
086:         mUriRecord = Preconditions.checkNotNull(uri);
087:         mTitleRecord = title;
088:         mAction = Preconditions.checkNotNull(action);
089:         mType = type;
090:     }
091:
092:     public UriRecord getUriRecord() {
093:         return mUriRecord;
094:     }
095:
096:     /**
097:      * smart poster의 타이틀 반환. null 이 될 수도 있음. 옵션이니깐.
098:      */
099:     public TextRecord getTitle() {
100:         return mTitleRecord;
101:     }
102:
```

\record\SmartPoster.java

```
103:    public static SmartPoster parse(NdefRecord record) {

            // 파싱 전에 TNF_WELL_KNOWN 이거나 RTD_SMART_POSTER인지 체크
104:        Preconditions.checkArgument( record.getTnf() == NdefRecord.TNF_WELL_KNOWN );
            // checkArgument() 메소드는 필수 조건검사 후 부적합(false) 하면 예외 발생
105:        Preconditions.checkArgument(
                        Arrays.equals( record.getType(), NdefRecord.RTD_SMART_POSTER ) );

106:        try {

107:            NdefMessage subRecords = new NdefMessage(record.getPayload()); // 알맹이꺼내고
108:            return parse(subRecords.getRecords()); // 메시지 안에 레코드들 다시 파싱

109:        } catch (FormatException e) {
110:            throw new IllegalArgumentException(e);    // 형식 미지원 예외 처리
111:        }

112:    }
113:
```

\record\SmartPoster.java

```
// 원본 레코드 파싱
114:    public static SmartPoster parse(NdefRecord[] recordsRaw) {
115:        try {
                // 순회를 위해 순회자 얻기
116:            Iterable<ParsedNdefRecord> records = NdefMessageParser.getRecords(recordsRaw);
                // URI 형식 레코드만 걸러서 꺼내기
117:            UriRecord uri = Iterables.getOnlyElement( Iterables.filter(records, UriRecord.class) );
                // 옵션인 텍스트 레코드가 있으면 꺼내기
118:            TextRecord title = getFirstIfExists(records, TextRecord.class);
                // 액션 꺼내기
119:            RecommendedAction action = parseRecommendedAction( recordsRaw );
                // 타입 꺼내기
120:            String type = parseType(recordsRaw);
                // 스마트포스터 객체 생성 후 반환
121:            return new SmartPoster(uri, title, action, type); // 스마트포스터 변환 후 반환

122:        } catch (NoSuchElementException e) {
123:            throw new IllegalArgumentException(e);      // 예외 처리
124:        }
125:    }
126:
```

\record\SmartPoster.java

```
127:     public static boolean isPoster(NdefRecord record) { // 스마트포스터 인가?
128:         try {
129:             parse(record); // 파싱되면
130:             return true; // 기다
131:         } catch (IllegalArgumentException e) {
132:             return false;
133:         }
134:     }
135:
```

\record\SmartPoster.java

```
// 스마트포스터 표시 용 레이아웃을 만들어 반환, 인테페이스 필수 구현 메소드
136:    public View getView(Activity activity, LayoutInflater inflater, ViewGroup parent, int offset) {

137:        if (mTitleRecord != null) {             // 타이틀이 있으면

138:            // title과 URI를 가지는 레이아웃 만들기
139:            LinearLayout container = new LinearLayout(activity);
140:            container.setOrientation(LinearLayout.VERTICAL); // 세로 추가 방향
141:            container.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
142:                LayoutParams.WRAP_CONTENT)); // 가로 세로 크기 지정
143:            container.addView(mTitleRecord.getView(activity, inflater, container, offset));
144:            inflater.inflate(R.layout.tag_divider, container); // XML로 부터 객체 생성
145:            container.addView(mUriRecord.getView(activity, inflater, container, offset));
146:            return container;

147:        } else {
148:            // URI의 뷰만 반환
149:            return mUriRecord.getView(activity, inflater, parent, offset);
150:        }
151:    }
152:
```

\record\SmartPoster.java

```
153:     /**
154:      * 지정한 type 인스턴스들의 첫 번째 것 반환
155:      * 없으면 null 반환
156:      */
157:     private static <T> T getFirstIfExists(Iterable<?> elements, Class<T> type) {
              // 지정한 형식의 원소들만 있는 순회자 생성
158:         Iterable<T> filtered          = Iterables.filter(elements, type);
159:         T instance                    = null;       // 반환할 원소 포인터 생성

160:         if ( !Iterables.isEmpty(filtered) ) {       // 객체들이 있으면
161:             instance                  = Iterables.get(filtered, 0);       // 첫 번째 반환
162:         }

163:         return instance;
164:     }
165:
```

\record\SmartPoster.java

```
// Recommanded Action 설정 값, 변수 및 메소드 정의
166:        private enum RecommendedAction {              // enumeration 타입 정의
167:            UNKNOWN( (byte) -1 ),          DO_ACTION( (byte) 0 ),
168:            SAVE_FOR_LATER( (byte) 1 ),    OPEN_FOR_EDITING( (byte) 2 );
169:            // 액션 종류를 맵에 저장하기
170:             private static final ImmutableMap<Byte, RecommendedAction> LOOKUP;
171:             static {
172:                 ImmutableMap.Builder<Byte, RecommendedAction> builder
                            = ImmutableMap.builder();
173:                 for (RecommendedAction action : RecommendedAction.values()) {
174:                     builder.put(action.getByte(), action);    // 맵 빌더에 액션들 저장
175:                 }
176:                 LOOKUP = builder.build();        // 맵에 빌더로 생성한 액션들 저장
177:        }
178:

179:        private final byte mAction;              // RecommendedAction 보관용
180:
181:        private RecommendedAction(byte val) {// RecommendedAction 보관 메소드
182:            this.mAction = val;
183:        }
184:
```

\record\SmartPoster.java

```
185:        private byte getByte() { // Byte 타입의 액션 반환
186:            return mAction;
187:        }
188:    }
189:    // 지정한 타입의 레코드 검색 메소드
190:    private static NdefRecord getByType(byte[] type, NdefRecord[] records) {

          // records 배열에서 type이 같은 레코드를 꺼내 반환
191:        for ( NdefRecord record : records ) {
192:            if ( Arrays.equals( type, record.getType() ) ) { // 배열 비교
193:                return record;
194:            }
195:        }

196:        return null;
197:    }
198:
```

\record\SmartPoster.java

```
199:    private static final byte[] ACTION_RECORD_TYPE = new byte[] {'a', 'c', 't'};
200:      // recommended 액션 파싱
201:    private static RecommendedAction parseRecommendedAction(
                                                    NdefRecord[] records) {
          // act 라는 문자열의 타입을 가지는 액션 레코드 가져오기
202:        NdefRecord record = getByType(ACTION_RECORD_TYPE, records);

203:        if (record == null) {
204:            return RecommendedAction.UNKNOWN;      // 알 수 없음
205:        }

206:        byte action = record.getPayload()[0];  // 알맹이 꺼내기
207:        if (RecommendedAction.LOOKUP.containsKey(action)) {
208:            return RecommendedAction.LOOKUP.get(action);        // 액션 알려주기
209:        }

210:        return RecommendedAction.UNKNOWN;          // 알 수 없음
211:    }
212:
```

\record\SmartPoster.java

```
213:     private static final byte[] TYPE_TYPE = new byte[] {'t'}; // t는 타입
214:      // 타입 꺼내기
215:     private static String parseType( NdefRecord[] records ) {
          // 타입 레코드 가져오기
216:         NdefRecord type = getByType( TYPE_TYPE, records);

217:         if (type == null) {
218:             return null;  // 없음, 옵션이니까
219:         }
             // 타입은 문자열로 표시 가능
220:         return new String(type.getPayload(), Charsets.UTF_8);
221:     }
222: }
```

텍스트 레코드 클래스

\RECORD\ TEXT RECORD .JAVA

\record\TextRecord.java

```
016: package com.example.android.nfc.record;
017:
018: import android.app.Activity;
019: import android.nfc.NdefRecord;
020: import android.view.LayoutInflater;
021: import android.view.View;
022: import android.view.ViewGroup;
023: import android.widget.TextView;
024:
025: import com.google.common.base.Preconditions;
026:
027: import com.example.android.nfc.R;
028:
029: import java.io.UnsupportedEncodingException;
030: import java.util.Arrays;
031:
```

\record\TextRecord.java

```
032: /**
033:  * NFC Text Record
034:  */
035: public class TextRecord implements ParsedNdefRecord {
036:
037:     /** ISO/IANA 언어 코드 */
038:     private final String mLanguageCode;
039:
040:     private final String mText;
041:      // 생성자
042:     private TextRecord(String languageCode, String text) {
043:         mLanguageCode = Preconditions.checkNotNull(languageCode);
044:         mText = Preconditions.checkNotNull(text);
045:     }
046:
```

\record\TextRecord.java

```
            // 화면 표시용 뷰 생성 후 반환
047:    public View getView(Activity activity, LayoutInflater inflater, ViewGroup parent, int offset) {
048:        TextView text = (TextView) inflater.inflate(R.layout.tag_text, parent, false);
049:        text.setText(mText);
050:        return text;
051:    }
052:
053:    public String getText() {   // 텍스트 값 반환
054:        return mText;
055:    }
056:
057:    /**
058:     * 현재 글자 요소에 관련된 ISO/IANA 언어 코드 반환
059:     */
060:    public String getLanguageCode() {
061:        return mLanguageCode;
062:    }
063:
```

\record\TextRecord.java

```
064:    // TODO: deal with text fields which span multiple NdefRecords
065:    public static TextRecord parse(NdefRecord record) {          // 텍스트 레코드 변환
066:        Preconditions.checkArgument(record.getTnf() == NdefRecord.TNF_WELL_KNOWN);
067:        Preconditions.checkArgument(Arrays.equals(record.getType(), NdefRecord.RTD_TEXT));

068:        try {
069:            byte[] payload = record.getPayload();       // 레코드 알맹이 정보 빼내기
070:            /*
071:             * payload[0] 는 "Status Byte Encodings" 필드를 포함한다.
072:             * NFC Forum의 "Text Record Type Definition" section 3.2.1. 참조
073:             *
074:             * bit7 은 Text Encoding 필드
075:             *
076:             * if (Bit_7 == 0): 글자가 UTF-8 엔코딩
077:             * if (Bit_7 == 1): 글자가 UTF16 엔코딩
078:             *
079:             * Bit_6 은 예약비트이며, 0으로 반드시 초기화
080:             *
081:             * Bits 5 에서 0비트는 IANA 언어 코드의 길이
082:             */
```

\record\TextRecord.java

```
083:            String textEncoding = ((payload[0] & 0200) == 0) ? "UTF-8" : "UTF-16";
084:            int languageCodeLength = payload[0] & 0077;
085:            String languageCode = new String(payload, 1, languageCodeLength, "US-ASCII");
                // 알맹이 정보를이용해 텍스트 만들기
086:            String text =
087:                new String(payload, languageCodeLength + 1,
088:                    payload.length - languageCodeLength - 1, textEncoding);

089:            return new TextRecord(languageCode, text);          // 텍스트 레코드 생성 후 반환

090:        } catch (UnsupportedEncodingException e) {
091:            // 손상된 태그가 아닌 이상 절대 발생하지 않음
092:            throw new IllegalArgumentException(e);
093:        }
094:    }
```

\record\TextRecord.java

```
095:
096:     public static boolean isText(NdefRecord record) { // 텍스트 형식인가?
097:         try {
098:             parse(record); // 파싱되면
099:             return true; // 기다
100:         } catch (IllegalArgumentException e) {
101:             return false;
102:         }
103:     }
104: }
```

URI 형식 태그용 레코드 클래스

\RECORD\ URI RECORD .JAVA

\record\UriRecord.java

```
016: package com.example.android.nfc.record;
017:  // URI 형식의 레코드 타입 처리 클래스
018: import android.app.Activity;
019: import android.net.Uri;
020: import android.nfc.NdefRecord;
021: import android.view.LayoutInflater;
022: import android.view.View;
023: import android.view.ViewGroup;
024: import android.widget.TextView;
025:
026: import com.google.common.base.Preconditions;
027: import com.google.common.collect.BiMap;
028: import com.google.common.collect.ImmutableBiMap;
029: import com.google.common.primitives.Bytes;
030:
031: import com.example.android.nfc.R;
032:
033: import java.nio.charset.Charset;
034: import java.util.Arrays;
035:
```

\record\UriRecord.java

```
036: /**
037:  * Uri를 포함하는 파싱 된 레코드
038:  */
039: public class UriRecord implements ParsedNdefRecord {
040:
041:     private static final String TAG = "UriRecord";
042:
043:     public static final String RECORD_TYPE = "UriRecord";
044:
045:     /**
046:      * NFC Forum의 "URI Record 형식 정의"
047:      *
048:      * "URI Identifier Codes"를 URI 문자열 접두어로 매핑 시킨것
049:      * NFC Forum의 URI Record 형식 정의 문서 section 3.2.2
050:      */
051:     private static final BiMap<Byte, String> URI_PREFIX_MAP = ImmutableBiMap.<Byte,
        String>builder()
052:             .put((byte) 0x00, "")
053:             .put((byte) 0x01, "http://www.")
054:             .put((byte) 0x02, "https://www.")
```

\record\UriRecord.java

```
055:            .put((byte) 0x03, "http://")          // 지원할 여러가지 URI 레코드 형식
056:            .put((byte) 0x04, "https://")
057:            .put((byte) 0x05, "tel:")
058:            .put((byte) 0x06, "mailto:")
059:            .put((byte) 0x07, "ftp://anonymous:anonymous@")
060:            .put((byte) 0x08, "ftp://ftp.")
061:            .put((byte) 0x09, "ftps://")
062:            .put((byte) 0x0A, "sftp://")
063:            .put((byte) 0x0B, "smb://")
064:            .put((byte) 0x0C, "nfs://")
065:            .put((byte) 0x0D, "ftp://")
066:            .put((byte) 0x0E, "dav://")
067:            .put((byte) 0x0F, "news:")
068:            .put((byte) 0x10, "telnet://")
069:            .put((byte) 0x11, "imap:")
070:            .put((byte) 0x12, "rtsp://")
071:            .put((byte) 0x13, "urn:")
```

\record\UriRecord.java

```
072:                .put((byte) 0x14, "pop:")                // 여러가지 URI 레코드 형식
073:                .put((byte) 0x15, "sip:")
074:                .put((byte) 0x16, "sips:")
075:                .put((byte) 0x17, "tftp:")
076:                .put((byte) 0x18, "btspp://")
077:                .put((byte) 0x19, "btl2cap://")
078:                .put((byte) 0x1A, "btgoep://")
079:                .put((byte) 0x1B, "tcpobex://")
080:                .put((byte) 0x1C, "irdaobex://")
081:                .put((byte) 0x1D, "file://")
082:                .put((byte) 0x1E, "urn:epc:id:")
083:                .put((byte) 0x1F, "urn:epc:tag:")
084:                .put((byte) 0x20, "urn:epc:pat:")
085:                .put((byte) 0x21, "urn:epc:raw:")
086:                .put((byte) 0x22, "urn:epc:")
087:                .put((byte) 0x23, "urn:nfc:")
088:                .build();                                // 끝
089:
```

\record\UriRecord.java

```
090:     private final Uri mUri;
091:
092:     private UriRecord(Uri uri) {              // 생성자
093:         this.mUri = Preconditions.checkNotNull(uri);
094:     }
095:         // 레코드 정보 표시를 위한 뷰 생성, 인터페이스 구현
096:     public View getView(Activity activity, LayoutInflater inflater, ViewGroup parent, int offset) {
097:         TextView text = (TextView) inflater.inflate(R.layout.tag_text, parent, false);
098:         text.setText(mUri.toString());
099:         return text;
100:     }
101:
102:     public Uri getUri() {
103:         return mUri;
104:     }
105:
```

\record\UriRecord.java

```
106:     /**
107:      * android.nfc.NdefRecord를 android.net.Uri로 변환
108:      * TNF_WELL_KNOWN / RTD_URI and TNF_ABSOLUTE_URI 모두 처리
109:      *
110:      * @throws NdefRecord 가 URI를 포함하지 않으면
111:      * IllegalArgumentException 예외 발생
112:      */
113:     public static UriRecord parse(NdefRecord record) {

114:         short tnf = record.getTnf();

115:         if (tnf == NdefRecord.TNF_WELL_KNOWN) {  // 잘 알려진 레코드 형식
116:             return parseWellKnown(record);
117:         } else if (tnf == NdefRecord.TNF_ABSOLUTE_URI) {        // URI 절대 주소 형식
118:             return parseAbsolute(record);
119:         }

120:         throw new IllegalArgumentException("Unknown TNF " + tnf);        // 알수 없음
121:     }
122:
```

\record\UriRecord.java

```
123:     /* 순수 URI record 파싱*/
124:     private static UriRecord parseAbsolute(NdefRecord record) {
125:         byte[] payload = record.getPayload(); // 레코드의 알맹이 꺼내기
126:         Uri uri = Uri.parse(new String(payload, Charset.forName("UTF-8")));
127:         return new UriRecord(uri);
128:     }
129:
```

\record\UriRecord.java

```
130:     /** 잘 알려진 URI record 파싱*/
131:     private static UriRecord parseWellKnown(NdefRecord record) {
132:         Preconditions.checkArgument(Arrays.equals(record.getType(), NdefRecord.RTD_URI));
133:         byte[] payload = record.getPayload();
134:         /*
135:          * payload[0]는 URI Identifier Code를 포함한다.
136:          * NFC Forum의 "URI Record Type Definition" section 3.2.2. 참조
137:          *
138:          * payload[1]...payload[payload.length - 1] 는 URI의
139:          * 나머지 부분 포함
140:          */
141:         String prefix = URI_PREFIX_MAP.get(payload[0]);
142:         byte[] fullUri =
143:             Bytes.concat(prefix.getBytes(Charset.forName("UTF-8")), Arrays.copyOfRange(payload,
     1,
144:                 payload.length));
145:         Uri uri = Uri.parse(new String(fullUri, Charset.forName("UTF-8")));
146:         return new UriRecord(uri);
147:     }
```

\record\UriRecord.java

```
148:
149:     public static boolean isUri(NdefRecord record) { // URI 인가?
150:         try {
151:             parse(record);
152:             return true;
153:         } catch (IllegalArgumentException e) {
154:             return false;
155:         }
156:     }
157:
158:     private static final byte[] EMPTY = new byte[0];
159: }
```

메인 액티비티

\SIMULATOR\
FAKE TAGS ACTIVITY .JAVA

\simulator\FakeTagsActivity.java

```
016: package com.example.android.nfc.simulator;
017:  // 메인 액티비티
018: import android.app.ListActivity;
019: import android.content.Intent;
020: import android.nfc.NdefMessage;
021: import android.nfc.NdefRecord;
022: import android.nfc.NfcAdapter;
023: import android.os.Bundle;
024: import android.view.View;
025: import android.widget.ArrayAdapter;
026: import android.widget.ListView;
027:
028: import com.google.common.base.Charsets;
029: import com.google.common.base.Preconditions;
030: import com.google.common.primitives.Bytes;
031:
032: import java.nio.charset.Charset;
033: import java.util.Locale;
034:
```

\simulator\FakeTagsActivity.java

```
035: /**
036:  * 스캔 된 태그를 실행하는 메인 액티비티
037:  */
038: public class FakeTagsActivity extends ListActivity {
039:
040:     static final String TAG = "FakeTagsActivity";          // 디버깅용 문자열
041:
042:     static final byte[] UID = new byte[] {0x05, 0x00, 0x03, 0x08};  // 디바이스 UID
043:
044:     ArrayAdapter<TagDescription> mAdapter;          // 리스트 뷰 어댑터
045:
```

\simulator\FakeTagsActivity.java

```
//   새 태그 만드는 메소드
046:     public static NdefRecord newTextRecord(String text, Locale locale, boolean encodeInUtf8) {
047:         Preconditions.checkNotNull(text);                    // 필수 조건 체크
048:         Preconditions.checkNotNull(locale);

             // 레코드 값 준비
049:         final byte[] langBytes = locale.getLanguage().getBytes(Charsets.US_ASCII);
050:         final Charset utfEncoding = encodeInUtf8 ? Charsets.UTF_8 : Charset.forName("UTF-16");
051:         final byte[] textBytes = text.getBytes(utfEncoding);
052:         final int utfBit = encodeInUtf8 ? 0 : (1 << 7);
053:         final char status = (char) (utfBit + langBytes.length);
054:         final byte[] data = Bytes.concat(new byte[] {(byte) status}, langBytes, textBytes);

055:         return new NdefRecord(
                         NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_TEXT, new byte[0], data);
056:     }
057:
```

\simulator\FakeTagsActivity.java

```
// 가짜 NdefRecord 만들어 내기
058:     public static NdefRecord newMimeRecord(String type, byte[] data) {

059:         Preconditions.checkNotNull(type);
060:         Preconditions.checkNotNull(data);
061:         final byte[] typeBytes = type.getBytes(Charsets.US_ASCII);

062:         return new NdefRecord(NdefRecord.TNF_MIME_MEDIA, typeBytes, new
     byte[0], data);

063:     }
064:
```

\simulator\FakeTagsActivity.java

```
065:     static final class TagDescription { // 태그 설명 클래스
066:
067:         public String title;              // 제목
068:
069:         public NdefMessage[] msgs;              // NDEF 메시지
070:
071:         public TagDescription(String title, byte[] bytes) {// 생성자
072:             this.title = title;
073:             try {
074:                 msgs = new NdefMessage[] {new NdefMessage(bytes)};
075:             } catch (final Exception e) {
076:                 throw new RuntimeException("Failed to create tag description", e);
077:             }
078:         }
079:
```

\simulator\FakeTagsActivity.java

```
080:         @Override
081:         public String toString() {
082:             return title;
083:         }
084:     }
085:     // toString 호출하면 타이틀(태그의 이름) 돌려주기
```

\simulator\FakeTagsActivity.java

```
086:    @Override
087:    public void onCreate(Bundle savedState) { // 생성자 – 가짜(Mock) 태그 집어 넣기
088:        super.onCreate(savedState);
              // 리스트뷰에 표시할 기본 문자열들 준비
089:        final ArrayAdapter<TagDescription> adapter = new ArrayAdapter<TagDescription>(
090:            this, android.R.layout.simple_list_item_1, android.R.id.text1);

091:        adapter.add(
092:            new TagDescription("Broadcast NFC Text Tag", MockNdefMessages.ENGLISH_PLAIN_TEXT));

093:        adapter.add(new TagDescription(
094:            "Broadcast NFC SmartPoster URL & text", MockNdefMessages.SMART_POSTER_URL_AND_TEXT));

095:        adapter.add(new TagDescription(
096:            "Broadcast NFC SmartPoster URL", MockNdefMessages.SMART_POSTER_URL_NO_TEXT));

097:        setListAdapter(adapter);

098:        mAdapter = adapter;
099:    }
```

\simulator\FakeTagsActivity.java

```
100:   // 리스트뷰에서 태그 아이템 선택 했을 때, TagViwer 액티비티로 표시
101:     @Override
102:     public void onListItemClick(ListView l, View v, int position, long id) {
             // 태그 설명 가져오기
103:         final TagDescription description = mAdapter.getItem(position);
            // 태그 인식 액션 인텐트 생성
104:         final Intent intent = new Intent(NfcAdapter.ACTION_TAG_DISCOVERED);
             // 태스 설명을 인텐트에 집어 넣기
105:         intent.putExtra(NfcAdapter.EXTRA_NDEF_MESSAGES, description.msgs);
             // 인텐트 처리 액티비티 실행
106:         startActivity(intent);

107:     }

108: }
```

가짜 NDEF 메시지 클래스

\SIMULATOR\
MOCK NDEF MESSAGES.JAVA

\simulator\MockNdefMessages.java

```
016: package com.example.android.nfc.simulator;
017:  // 가짜 NDEF 포맷 메시지
018: /**
019:  * 이 클래스는 가짜 NFC Ndef 형식의 태그 목록을 제공한다.
020:  */
021: public class MockNdefMessages {
022:
023:     /**
024:      * URL을 포함하고 텍스트는 포함하지 않는 Smart Poster
025:      */
026:     public static final byte[] SMART_POSTER_URL_NO_TEXT =
027:         new byte[] {(byte) 0xd1, (byte) 0x02, (byte) 0x0f, (byte) 0x53, (byte) 0x70, (byte) 0xd1,
028:             (byte) 0x01, (byte) 0x0b, (byte) 0x55, (byte) 0x01, (byte) 0x67, (byte) 0x6f,
029:             (byte) 0x6f, (byte) 0x67, (byte) 0x6c, (byte) 0x65, (byte) 0x2e, (byte) 0x63,
030:             (byte) 0x6f, (byte) 0x6d};
031:
032:     /**
033:      * 영문으로 된 보통 텍스트 태그
034:      */
035:     public static final byte[] ENGLISH_PLAIN_TEXT =
036:         new byte[] {(byte) 0xd1, (byte) 0x01, (byte) 0x1c, (byte) 0x54, (byte) 0x02, (byte) 0x65,
037:             (byte) 0x6e, (byte) 0x53, (byte) 0x6f, (byte) 0x6d, (byte) 0x65, (byte) 0x20,
038:             (byte) 0x72, (byte) 0x61, (byte) 0x6e, (byte) 0x64, (byte) 0x6f, (byte) 0x6d,
039:             (byte) 0x20, (byte) 0x65, (byte) 0x6e, (byte) 0x67, (byte) 0x6c, (byte) 0x69,
040:             (byte) 0x73, (byte) 0x68, (byte) 0x20, (byte) 0x74, (byte) 0x65, (byte) 0x78,
041:             (byte) 0x74, (byte) 0x2e};
```

\simulator\MockNdefMessages.java

```
042:
043:     /**
044:      * URL과 텍스트를 포함하는 스마트 포스터
045:      */
046:     public static final byte[] SMART_POSTER_URL_AND_TEXT =
047:         new byte[] {(byte) 0xd1, (byte) 0x02, (byte) 0x1c, (byte) 0x53, (byte) 0x70, (byte) 0x91,
048:             (byte) 0x01, (byte) 0x09, (byte) 0x54, (byte) 0x02, (byte) 0x65, (byte) 0x6e,
049:             (byte) 0x47, (byte) 0x6f, (byte) 0x6f, (byte) 0x67, (byte) 0x6c, (byte) 0x65,
050:             (byte) 0x51, (byte) 0x01, (byte) 0x0b, (byte) 0x55, (byte) 0x01, (byte) 0x67,
051:             (byte) 0x6f, (byte) 0x6f, (byte) 0x67, (byte) 0x6c, (byte) 0x65, (byte) 0x2e,
052:             (byte) 0x63, (byte) 0x6f, (byte) 0x6d};
053:
054:     /**
055:      * 모든 가짜 Ndef 태그들
056:      */
057:     public static final byte[][] ALL_MOCK_MESSAGES =
058:         new byte[][] {SMART_POSTER_URL_NO_TEXT, ENGLISH_PLAIN_TEXT,
    SMART_POSTER_URL_AND_TEXT};
059: }
```