

# NFCセミナー資料(2日目)

2013/10/06

早瀬

# Agenda

- “NFC歌留多”デモ サンプル解説
- NFCハッカソン
- アイディアソン+ハッカソンで投票

# Agenda

- “NFC歌留多”デモ サンプル解説
- NFCハッカソン
- アイディアソン+ハッカソンで投票

# “NFC歌留多” デモ(サンプル)の解説

※プロジェクト準備

# 主に変更を行う箇所

- AndroidManifest.xml

⇒ アプリケーションの構成

- Layout

⇒ **Activity**(画面)の部品配置

- 各Activityのソースコード

⇒ 処理

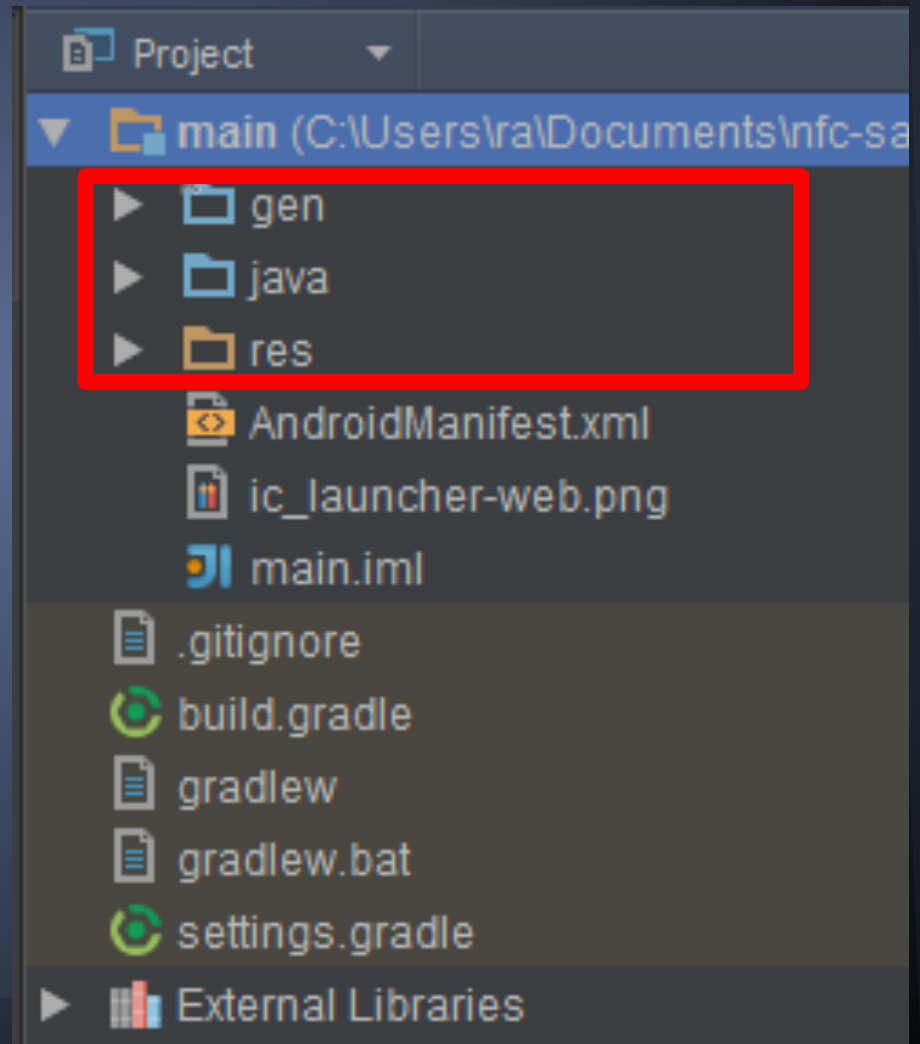
# プロジェクト構成(AndroidStudio)

## ■gen

自動生成されるファイル  
が配置される

## ■java(後述)

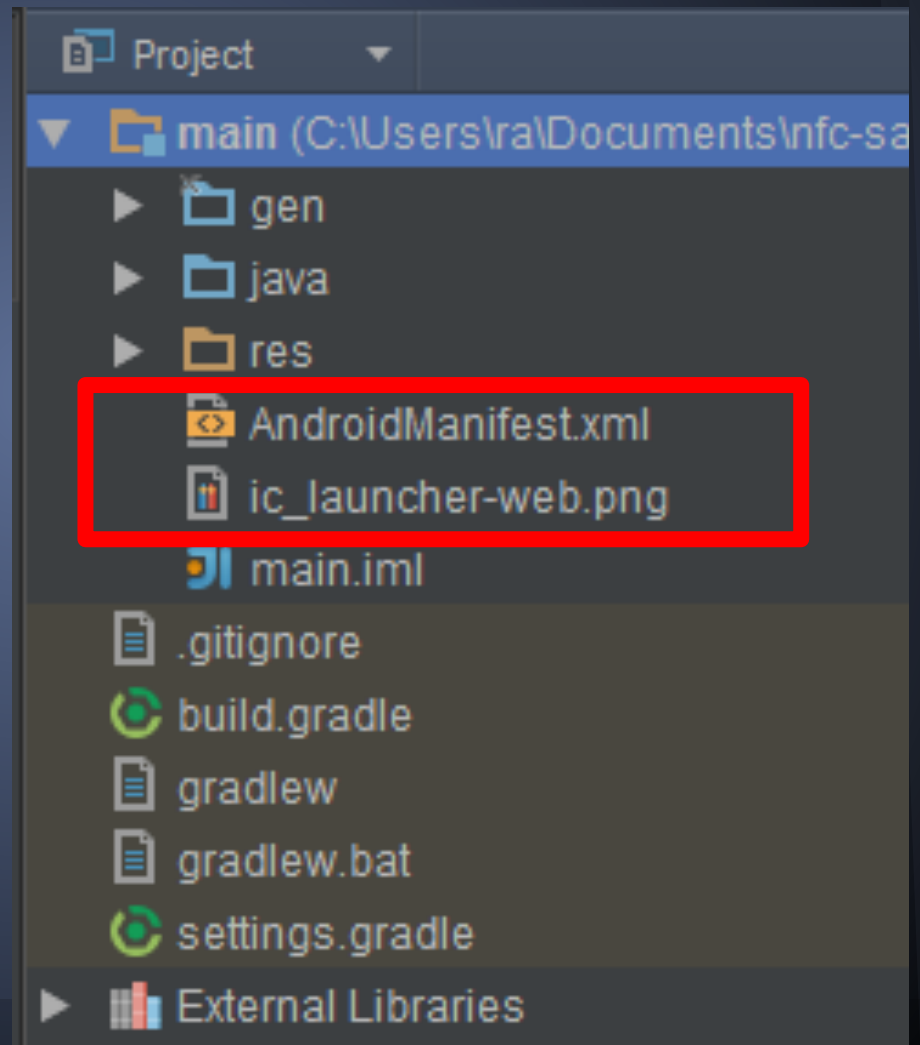
## ■res(後述)



# プロジェクト構成(AndroidStudio)

■AndroidManifest.xml  
プロジェクト構成ファイル

■ic\_launcher-web.png  
Google play store用の  
アイコン画像ファイル



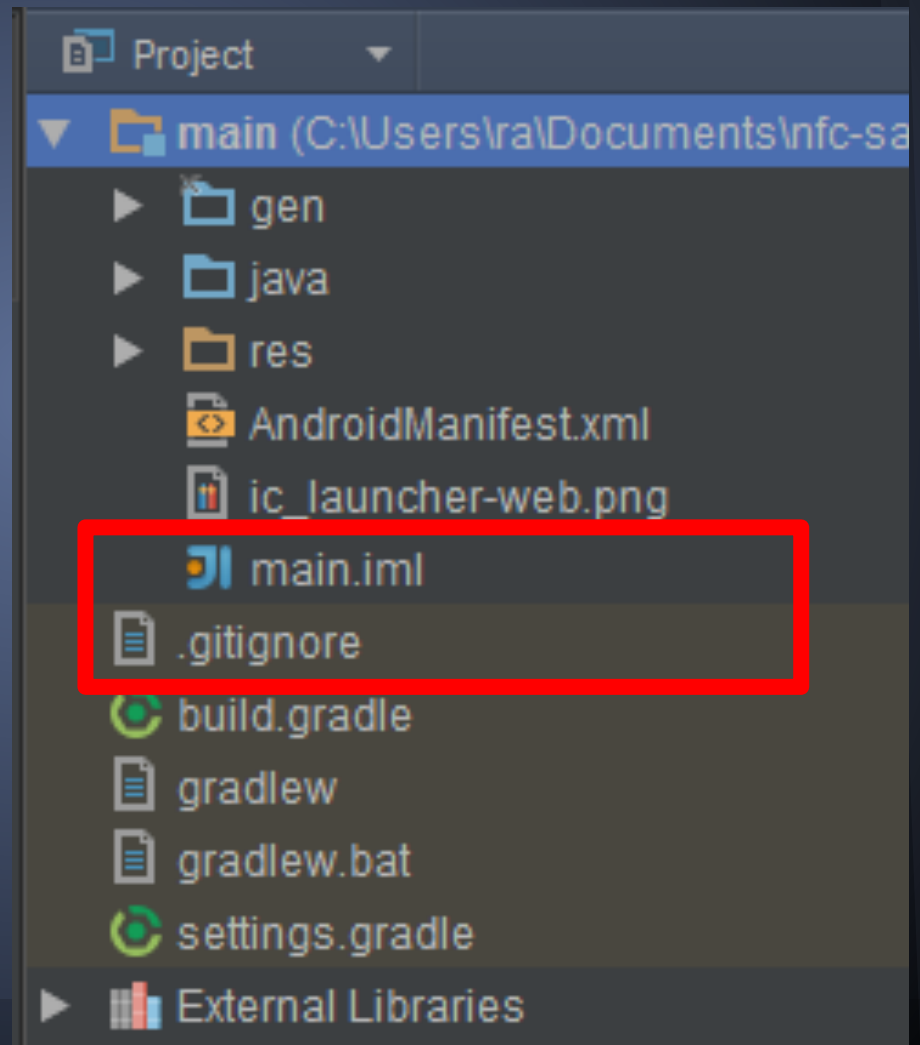
# プロジェクト構成(AndroidStudio)

## ■main.iml

AndroidStudioが  
自動生成した  
モジュール設定ファイル

## ■.gitignore

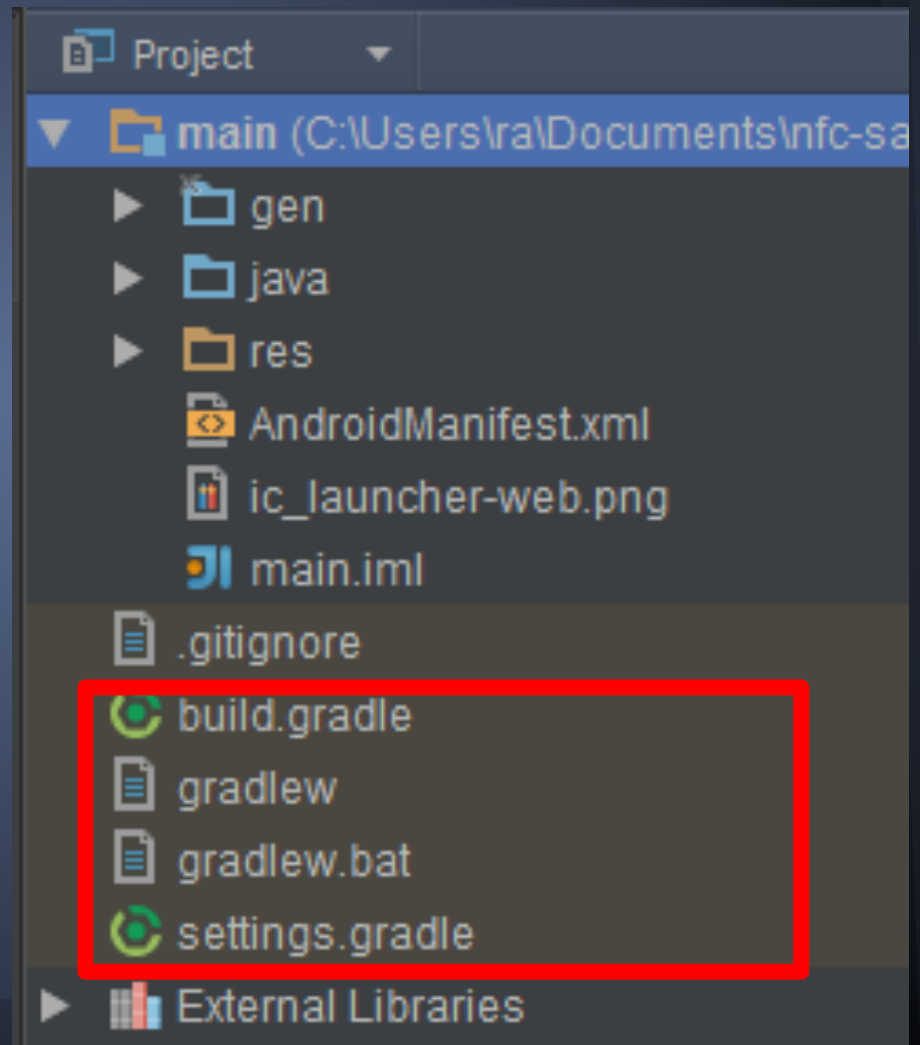
gitのバージョン管理  
対象外とするファイルを  
管理するファイル



# プロジェクト構成(AndroidStudio)

- build.gradle
- gradlew
- gradlew.bat
- settings.gradle

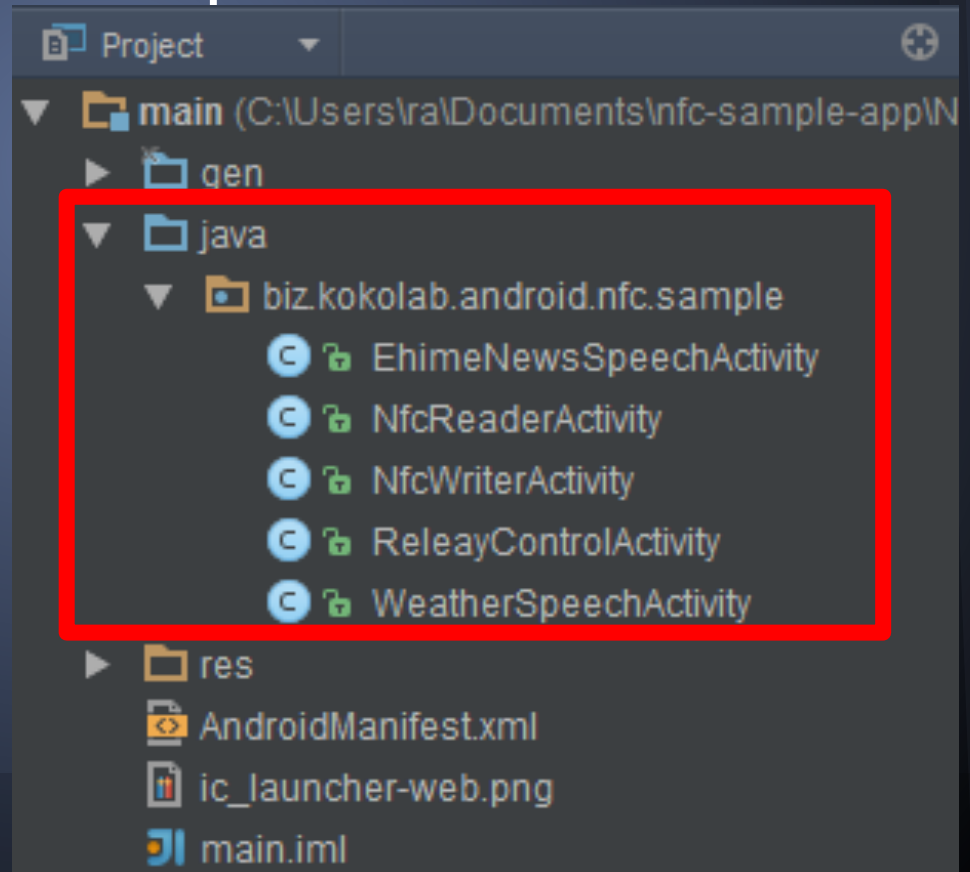
Gradle(ビルドツール)の  
設定/実行ファイル



# プロジェクト構成(AndroidStudio)

■名前空間(下記)毎に表示

`biz.kokolab.android.nfc.sample`



# プロジェクト構成(AndroidStudio)

## ■java

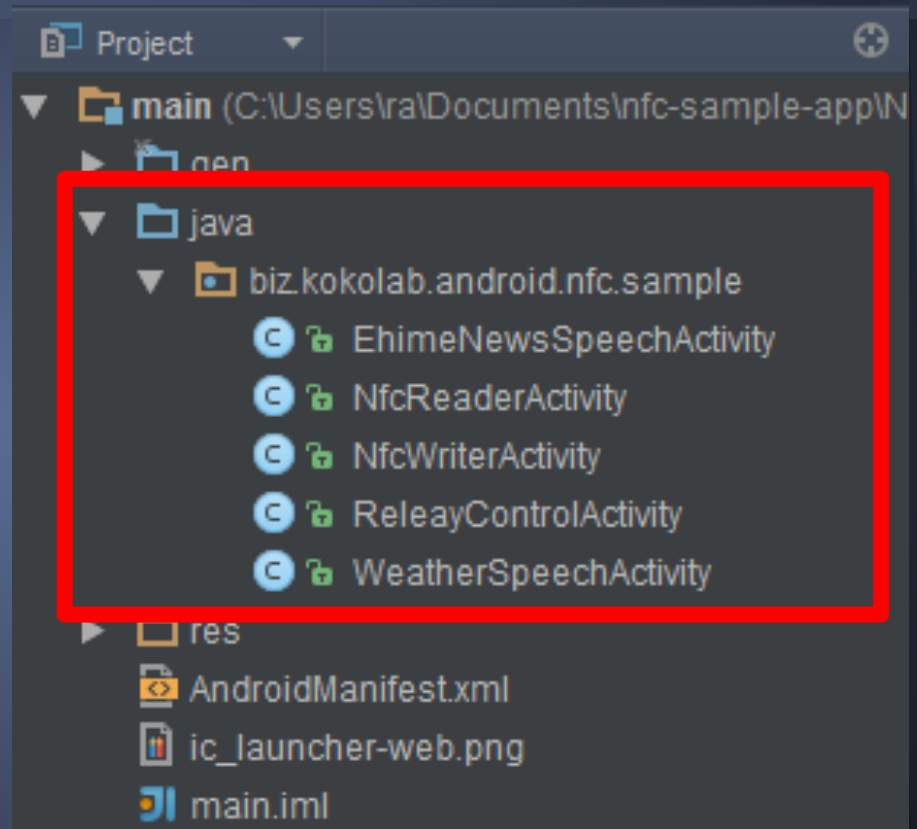
ソースコードを格納

今回のデモでは

**Activity**のファイルのみ

名前空間(下記)毎に表示

**biz.kokolab.android.nfc.sample**



# プロジェクト構成(AndroidStudio)

## ■drawable-XXX

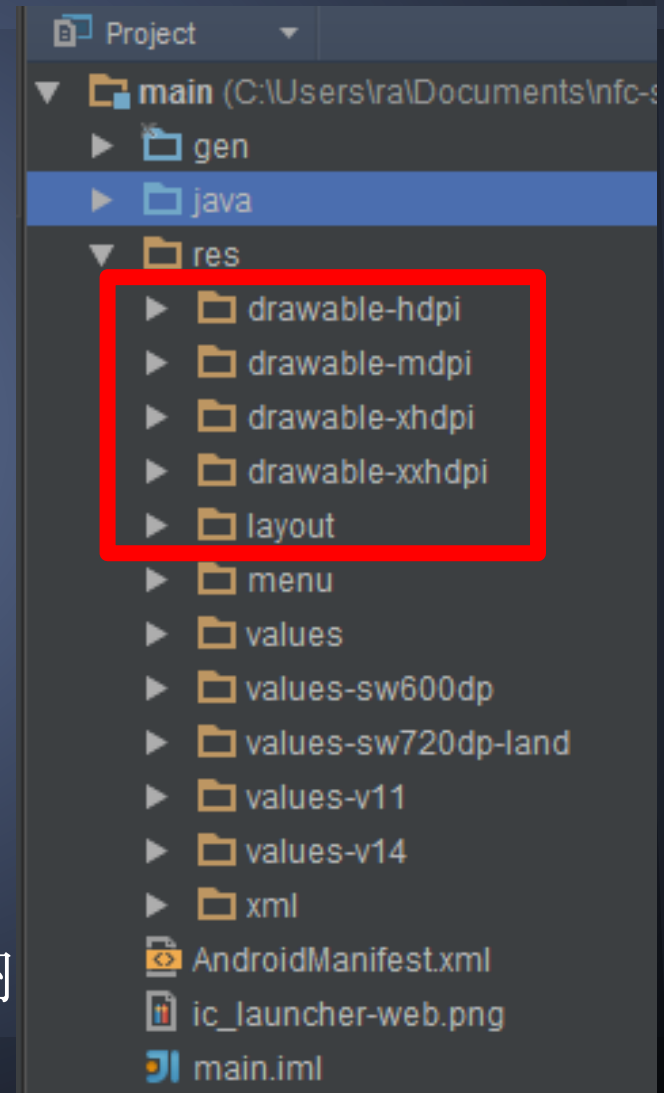
画像データを格納

XXXは解像度の違い。

(端末の解像度で切り替えられる  
ように用意できる)

## ■layout

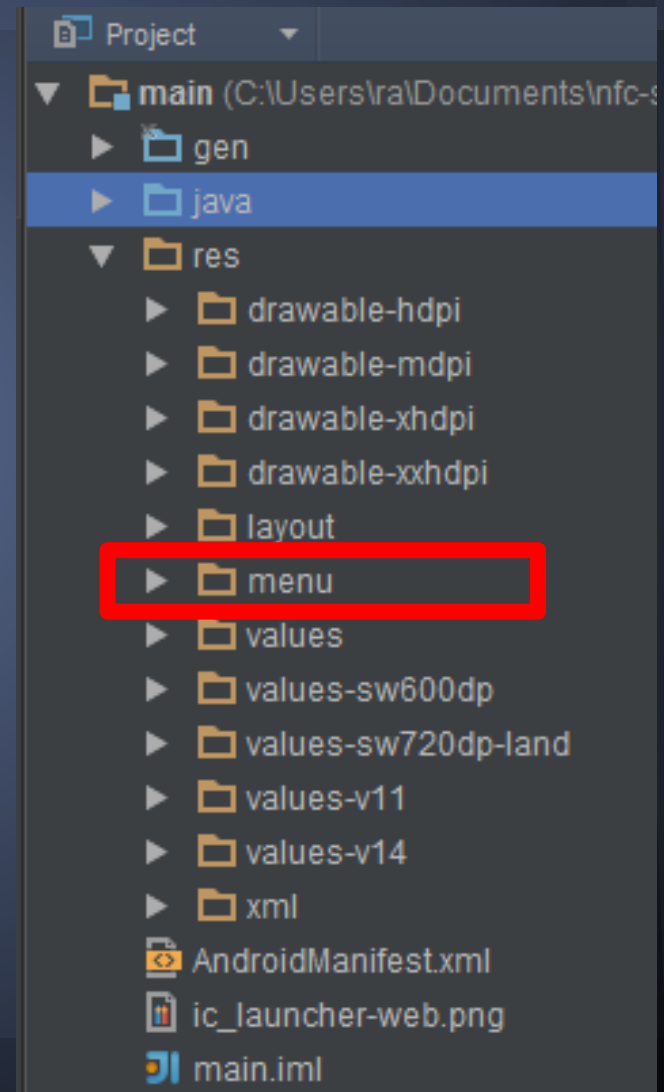
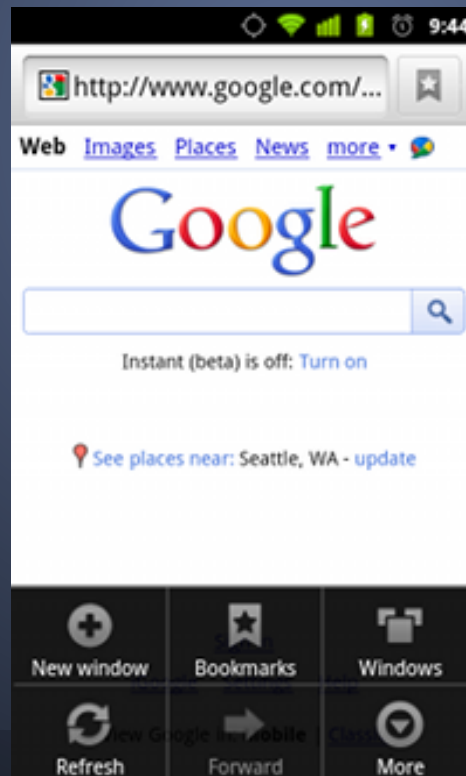
レイアウト(Activity上の部品など)  
に関する設定ファイル(xml)を格納



# プロジェクト構成(AndroidStudio)

## ■menu

メニューに関する設定ファイルを  
格納



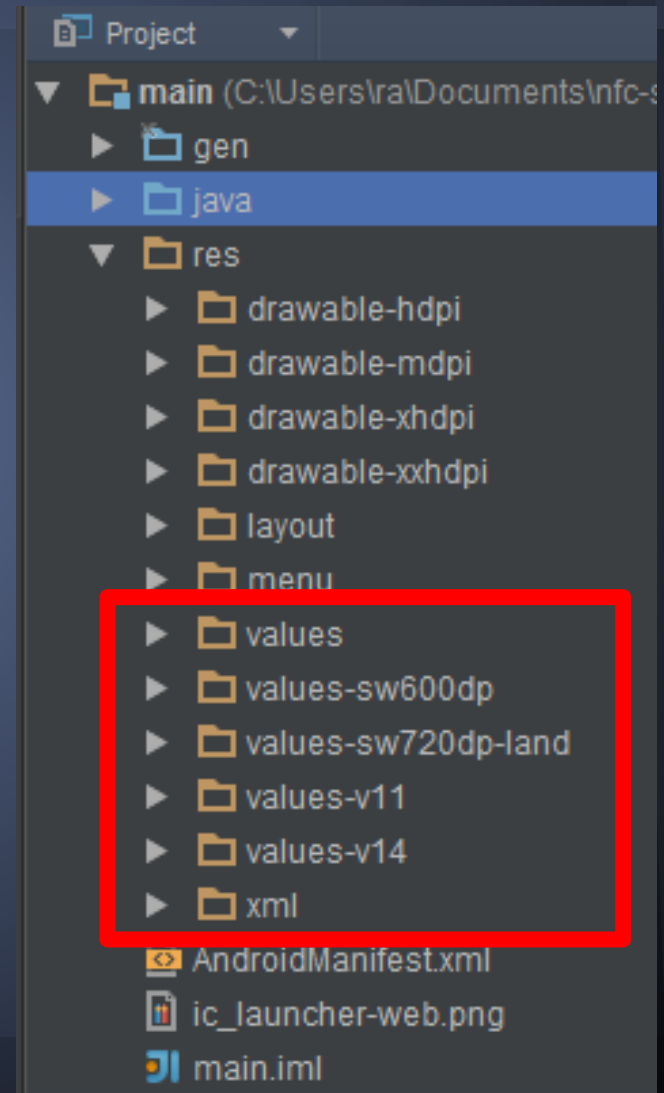
# プロジェクト構成(AndroidStudio)

## ■values

値を定義した設定ファイルを格納  
解像度やバージョンに応じた  
フォルダが用意されている

## ■xml

任意のXMLファイルを格納  
(固定データなどで利用)



# プロジェクト構成(AndroidStudio)

## ■values/dimens.xml

寸法に関する情報を定義

## ■values/strings.xml

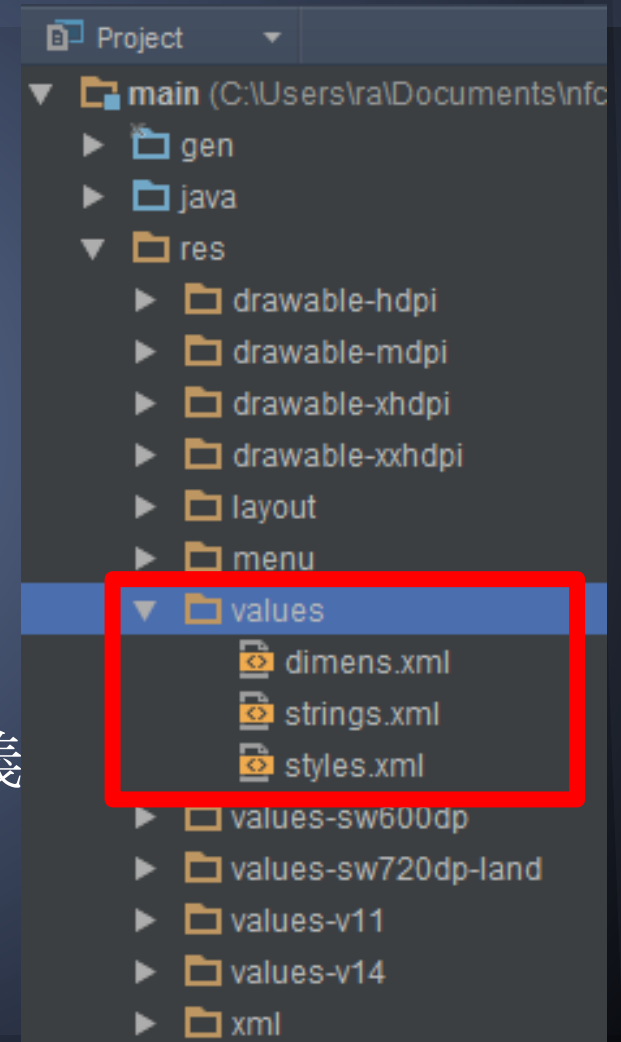
文字列情報を定義

## ■values/styles.xml

スタイル(表示テーマ)に関する情報を定義

## ■values/color.xml ※画像には無い

色に関する情報を定義



# AndroidManifest.xml

以下のような情報を定義する

- ・アプリケーションの基本情報(バージョンなど)
- ・実行に必要な権限(xxxを行いたい場合に記載)
- ・インストールに必要な条件(特定のバージョン)
- ・Activityの定義

# AndroidManifest.xmlの説明

1行目:XML宣言(xmlで記載されている宣言)

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

2-5行目:AndroidManifest.xmlのルート要素  
アプリケーションの基本情報を設定

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="biz.kokolab.android.nfc.sample"  
    android:versionCode="1"  
    android:versionName="1.0" >
```

# AndroidManifest.xmlの説明

7-9行目:Android APIレベルを指定する。

指定条件を満たさないデバイスには  
インストールさせない制御を行える。

```
<uses-sdk  
    android:minSdkVersion="14"  
    android:targetSdkVersion="16" />
```

# AndroidManifest.xmlの説明

11-15行目:アプリケーションが実行する処理に関する権限を定義する

- ・未定義の権限を要する処理は実行できない
- ・定義している権限についてはアプリケーションのインストール時にインストール確認画面で表示される

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.CALL_PHONE" />
```

```
<!-- NFC ハードウェアにアクセスするための宣言 -->  
<uses-permission android:name="android.permission.NFC" />
```

# AndroidManifest.xmlの説明

17行目:NFCハードウェアを持つデバイスのみに  
対し、Androidマーケットで表示されるよう  
にする宣言(コメントの通り)

```
<!-- NFC ハードウェアを持つデバイスのみに対し、Android マーケットでアプリケーションが表示されてくるようにするための宣言 -->  
<uses-feature android:name="android.hardware.nfc" android:required="true" />
```

# AndroidManifest.xmlの説明

19-23行目:Androidアプリケーションの定義  
アプリケーション名(設定画面で表示)  
アイコン、テーマなどを指定  
このタグ内に**Activity**の宣言を記載

```
<application  
    android:allowBackup="true"  
    android:icon="@drawable/ic_launcher"  
    android:label="Nfcサンプルアプリ"  
    android:theme="@style/AppTheme" >
```

# AndroidManifest.xmlの説明

26行目:利用する共有ライブラリの宣言  
(リレー制御でUSB接続するために利用)

```
<!-- USBアクセサリを利用するための宣言 -->  
<uses-library android:name="com.android.future.usb.accessory" />
```

# AndroidManifest.xmlの説明

26行目:利用する共有ライブラリの宣言  
(リレー制御でUSB接続するために利用)

```
<!-- USBアクセサリを利用するための宣言 -->  
<uses-library android:name="com.android.future.usb.accessory" />
```

# AndroidManifest.xmlの説明

28-37行目:書き込みActivityの宣言

このintent-filterを指定しておく  
とアプリ起動時に呼び出される

```
<activity
    android:name="biz.kokolab.android.nfc.sample.NfcWriterActivity"
    android:screenOrientation="portrait"
    android:label="Nfcサンプルアプリ" >

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

# AndroidManifest.xmlの説明

## 39-51行目:NFC読み込みActivityの宣言

このintent-filterを指定しておくと  
NFC検知時に呼び出される

```
<activity
    android:name="biz.kokolab.android.nfc.sample.NfcReaderActivity"
    android:label="Nfcサンプルアプリ"
    android:screenOrientation="portrait"
    android:launchMode="singleInstance">

    <!-- NDEFタグが見つかった時にこのアクティビティが呼ばれるように設定 -->
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

# AndroidManifest.xmlの説明

## 53-58行目:天気読みあげActivityの宣言

```
<activity
    android:name="biz.kokolab.android.nfc.sample.WeatherSpeechActivity"
    android:label="Nfcサンプルアプリ"
    android:screenOrientation="portrait"
    android:launchMode="singleInstance">
</activity>
```

## 60-65行目:愛媛新聞読みあげActivityの宣言

```
<activity
    android:name="biz.kokolab.android.nfc.sample.EhimeNewsSpeechActivity"
    android:label="Nfcサンプルアプリ"
    android:screenOrientation="portrait"
    android:launchMode="singleInstance">
</activity>
```

# AndroidManifest.xmlの説明

## 67-79行目:リレー制御Activityの宣言

このintent-filterを指定しておく  
と  
USB検知(接続)時に通知を受け取る  
meta-dataの記述により検知するUSBの  
製造元/モデル/バージョンを定義したファイルを  
指定する(res/xml/accessory\_filter.xml)

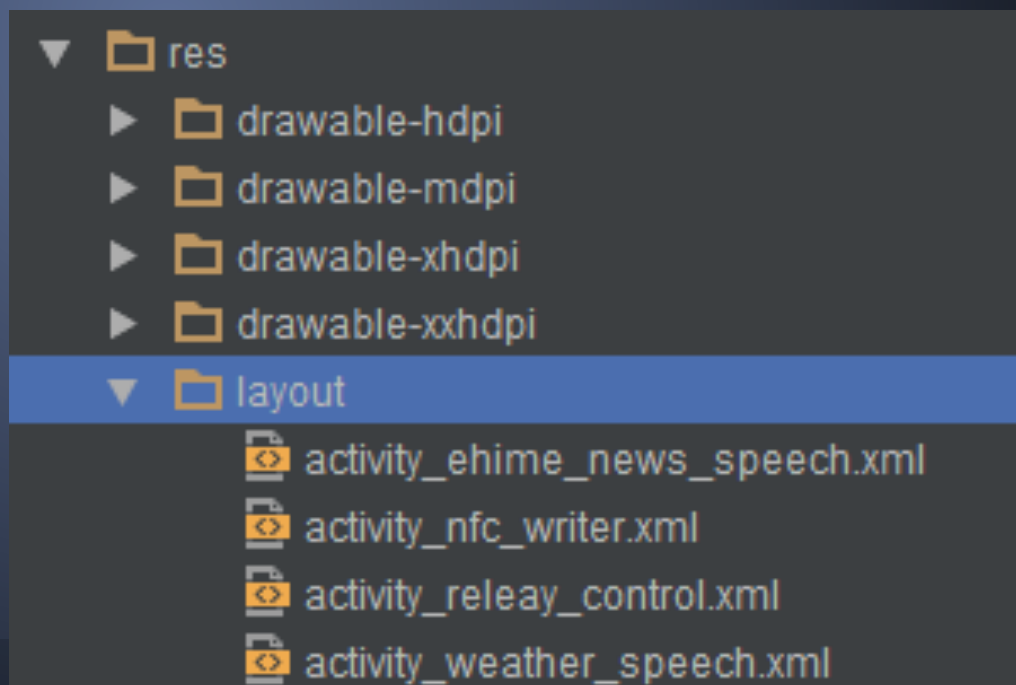
```
<activity
    android:name="biz.kokolab.android.nfc.sample.ReleayControlActivity"
    android:label="Nfcサンプルアプリ"
    android:screenOrientation="portrait"
    android:launchMode="singleTask">

    <!-- USBアクセサリが設定されたときに呼ばれる設定 -->
    <intent-filter>
        <action android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
    </intent-filter>
    <meta-data android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
        android:resource="@xml/accessory_filter" />
</activity>
```

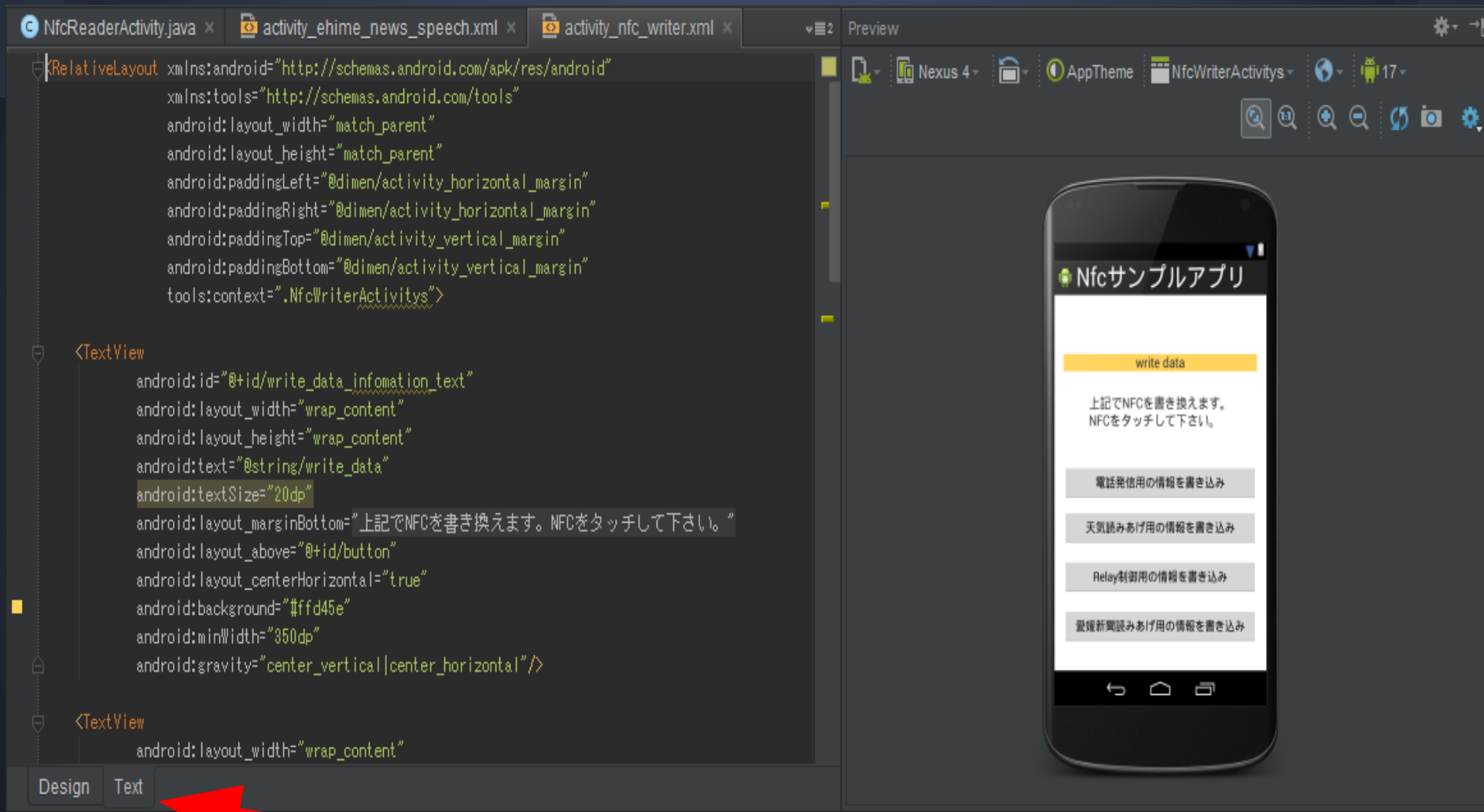
# Layoutの説明

- Activityの画面レイアウトを定義
- res/layoutの中にxmlファイルを配置

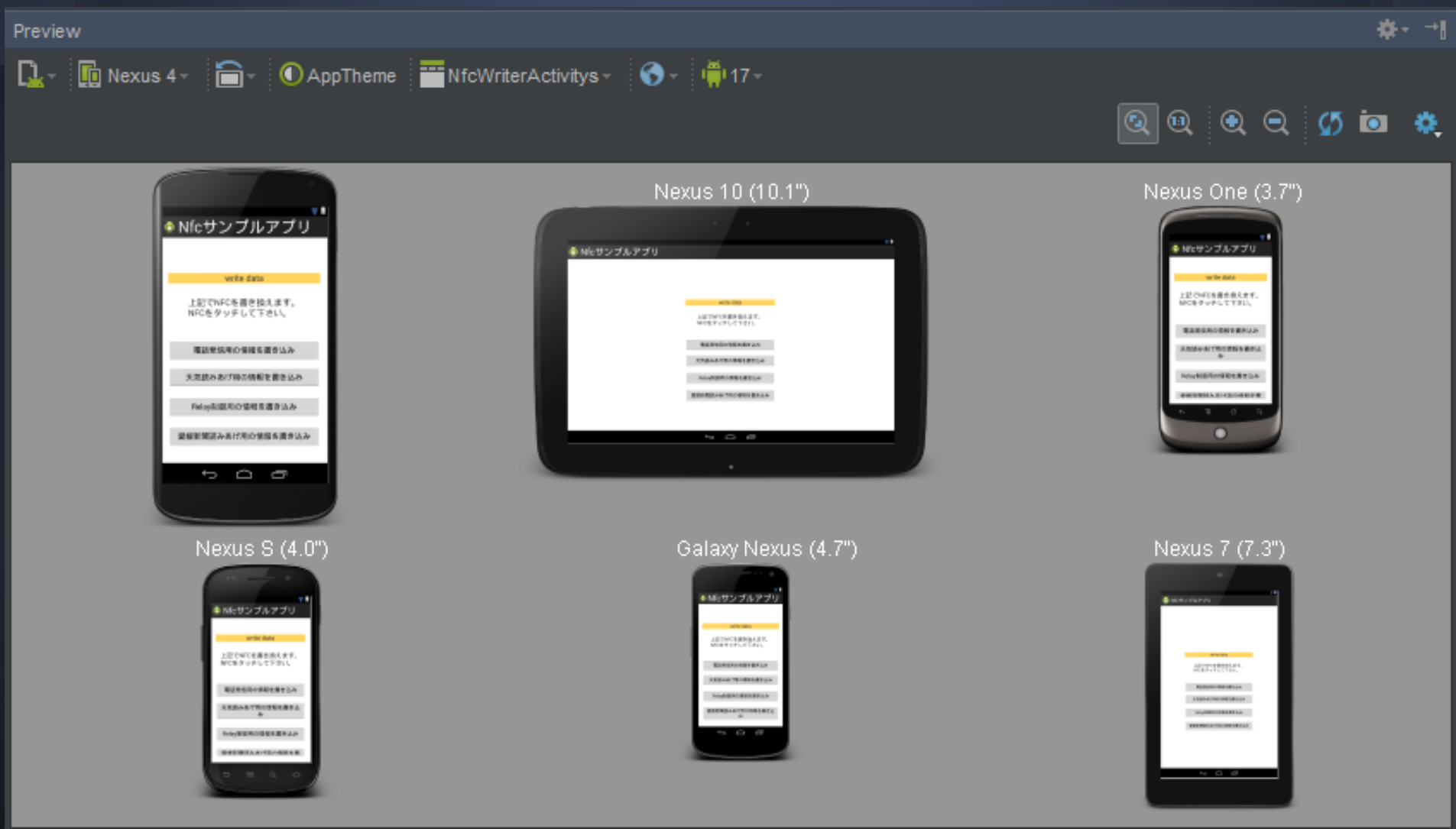
※全Activity用のファイルが存在するわけではない







# 複数デバイスによる表示確認



# Layoutの説明

- 大まかな配置は**Design**で行えます
- 詳細な配置、設定は**Text**で行うと良いです
- **android:onClick**でクリック時に呼び出されるメソッド(処理)を定義できます

```
<Button
    android:onClick="onClickTelephone"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="天気読みあげ用の情報を書き込み"
    android:id="@+id/button"
    android:minWidth="350dp"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:layout_alignLeft="@+id/button4"/>
```

# 各Activityのソースコード解説

各Activityのクラスは  
”Activity”クラスを継承して作成します。

```
/**  
 * NFCで書き込みを行うためのアクティビティ。  
 */  
public class NfcWriterActivity extends Activity {
```

# 各Activityのソースコード解説

処理内容に応じて分類分けすると  
基本(多く)は① ②のメソッド、  
多少③のメソッドがある感じになります。

①状態遷移系のメソッド

②イベントに関するメソッド(クリック時など)

③その他

# Activityの状態遷移図



# NfcWriterActivityの説明(状態遷移系)

- onCreate
- onResume
- onPause
- onNewIntent ✕
- writeTag
- onClickTelephone
- onClickWeatherSpeech
- onClickRelay
- onClickEhimeNewsSpeech

# NfcWriterActivityの説明(イベント系)

- onCreate
- onResume
- onPause
- onNewIntent
- writeTag
- onClickTelephone
- onClickWeatherSpeech
- onClickRelay
- onClickEhimeNewsSpeech

# NfcWriterActivityの説明(その他)

- writeTag

NFCタグ書き込み処理を行うメソッド

他メソッド(onNewIntent)より呼び出される。

# NfcWriterActivityの説明 - writeTag

・104-108行目:

書き込みデータが設定されていない場合終了する

mWriteDataは各ボタンクリック時に呼ばれる

メソッド(onClick～)内にて設定される。

Toastはポップアップのメッセージ

第二引数に表示文字列、第三引数に表示時間を指定する

```
if(mWriteData == null) {  
    // いずれかのボタンが押されるまでタグを書き込まない  
    Toast.makeText(this, "いずれかの書き込みボタンを押してください。", Toast.LENGTH_LONG).show();  
    return;  
}
```

# NfcWriterActivityの説明 - writeTag

・111-112行目:

NFCに書き込むNDEFデータを作成している。

```
// NDEFフォーマットでtext/plainのデータを書き込む。
```

```
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA, "text/plain".getBytes(), new byte[] {}, mWriteData.getBytes());
```

```
NdefMessage message = new NdefMessage(new NdefRecord[] { record });
```

# NfcWriterActivityの説明 - writeTag

・111-112行目:

NFCに書き込むNDEFデータを作成している。

```
// NDEFフォーマットでtext/plainのデータを書き込む。
```

```
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA, "text/plain".getBytes(), new byte[] {}, mWriteData.getBytes());
```

```
NdefMessage message = new NdefMessage(new NdefRecord[] { record });
```

# NDEF ?

- NFC Data Exchange Format

- NFCタグのデータフォーマット形式

NFCタグへの書き込み／読み込みはNDEF形式で行う。

- NDEFには下記が含まれる。
  - 1個のNDEF Message
  - 0個以上のNDEF Record

NDEF Message				
NDEF Record	NDEF Record	NDEF Record	NDEF Record	NDEF Record

# NDEF Recordフォーマット

MB(1bit)	ME(1bit)	CF(1bit)	SR(1bit)	IL(1bit)	TNF(3bit)
TYPE LENGTH (1Byte)					
PAYLOAD LENGTH(1Byte or 4Byte)					
ID LENGTH (0~1 Byte)					
TYPE (0~n Byte)					
ID (0~n Byte)					
PAYLOAD (0~n Byte)					

# NDEF Recordフォーマット - TNF

- TNF(Type Name Format)
- NDEF Recordの種類を示す。

値	種類
0x00	Empty
0x01	NFC Forum well-known-type
0x02	Media Type(RFC2046)
0x03	Absolute URI(RFC3986)
0x04	NFC Forum external type
0x05	Unknown
0x06	Unchanged
0x07	Reserved

MB (1bit)	ME (1bit)	CF (1bit)	SR (1bit)	IL (1bit)	TNF (3bit)
TYPE LENGTH (1Byte)					
PAYLOAD LENGTH(1Byte or 4Byte)					
ID LENGTH (0~1 Byte)					
TYPE (0~n Byte)					
ID (0~n Byte)					
PAYLOAD (0~n Byte)					

# NDEF Recordフォーマット - TYPE

PAYLOADの種類を示す。

例えばTNF=0x02(Media Type)の場合

“text/plain”はテキストデータ

“image/png”はPNG形式

“application/json”はJSONデータであることを示す。

MB (1bit)	ME (1bit)	CF (1bit)	SR (1bit)	IL (1bit)	TNF (3bit)
TYPE LENGTH (1Byte)					
PAYLOAD LENGTH(1Byte or 4Byte)					
ID LENGTH (0~1 Byte)					
TYPE (0~n Byte)					
ID (0~n Byte)					
PAYLOAD (0~n Byte)					

# NDEF Recordフォーマット - PAYLOAD

データ本体

PAYLOADの長さは  
PAYLOAD\_LENGTHに含める。

MB (1bit)	ME (1bit)	CF (1bit)	SR (1bit)	IL (1bit)	TNF (3bit)
TYPE LENGTH (1Byte)					
PAYLOAD_LENGTH (1Byte or 4Byte)					
ID LENGTH (0~1 Byte)					
TYPE (0~n Byte)					
ID (0~n Byte)					
PAYLOAD (0~n Byte)					

# NfcWriterActivityの説明 - writeTag

・111-112行目:

NFCに書き込むNDEFデータを作成している。

```
// NDEFフォーマットでtext/plainのデータを書き込む。  
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA, "text/plain".getBytes(), new byte[] {}, mWriteData.getBytes());  
NdefMessage message = new NdefMessage(new NdefRecord[] { record });
```

NdefRecordを作成する際に下記を設定している

- ・TNF(第一引数)
- ・TYPE(第二引数)
- ・PAYLOAD(第四引数)

# NfcWriterActivityの説明 - writeTag

・115-136行目:

Intentから検知したNFCタグ情報を取得し

下記条件に合致するNFCタグであるかを確認している。

- ・NFCタグがかざされた事によるIntentである
- ・NFCタグがNDEFフォーマットであるか
- ・NFCタグが書き込み可能であるか

# NfcWriterActivityの説明 - writeTag

・138-163行目:

NFCタグにNdefMessageを書き込んでいる。

基本的な流れとして、以下の流れで書き込みが行える。

Ndef.connect() ... NFCタグと通信を開始する



Ndef.writeNdefMessage() ... NdefMessageを書き込む



Ndef.close() ... NFCタグとの通信を終了する

# NfcReaderActivityの説明

- onCreate
- onNewIntent
- resolveIntent  
NDEF\_DISCOVEREDで検知したNFCタグから  
NDEFメッセージを取り出し、取得した文字列に応じて  
他のActivityを呼び出す(startActivity)
- extractMessageData  
NDEFメッセージを取り出す。  
(resolveIntentから呼び出される)

# NfcReaderActivityの説明 - readTag

resolveIntentメソッド内にて

NDEFデータの取得および解析を行っている。

# NfcReaderActivityの説明 - readTag

73行目:

intent(NDEF検知時のインテント)より、  
getParcelableArrayExtraメソッドを利用して  
NDEFメッセージ(配列)を取得

```
// タグからパースされた NDEF メッセージの配列を取得する。ACTION_NDEF_DISCOVEREDの場合は必須で送信される。  
Parcelable[] ndefMessageData = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
```

# NfcReaderActivityの説明 - readTag

79行目 + 102-118行目:

extractMessageDataにて、NDEFメッセージの1件目のデータ(NDEF Record)を取得する。

```
private String extractMessageData(Parcelable[] ndefMessageData) {  
    StringBuilder message = new StringBuilder();  
    for (int i = 0; i < ndefMessageData.length; i++) {  
        // NDEFメッセージを取り出す  
        NdefMessage ndefMessage = (NdefMessage)ndefMessageData[i];  
  
        for (NdefRecord record : ndefMessage.getRecords()) {  
            byte[] payload = record.getPayload();  
            if (payload != null) {  
                // 今回は1件目にデータを入れているので、それを取りだす。  
                return new String(payload);  
            }  
        }  
    }  
  
    return null;  
}
```

# NfcReaderActivityの説明 - readTag

80-91行目:

NDEF Recordのデータ(今回のデモでは文字列)の内容に応じて、処理を分岐させている。

```
String messageData = extractMessageData(ndefMessageData);
if ("biz.kokolab.telephone".equals(messageData)) {
    // 1.電話をかけるIntentを投げる
    startActivity(new Intent(MediaStore.ACTION_IMAGE_CAPTURE));
} else if ("biz.kokolab.weather".equals(messageData)) {
    startActivity(new Intent(this, WeatherSpeechActivity.class));
} else if ("biz.kokolab.relay".equals(messageData)) {
    startActivity(new Intent(this, RelayControlActivity.class));
} else if ("biz.kokolab.ehimenews".equals(messageData)) {
    startActivity(new Intent(this, EhimeNewsSpeechActivity.class));
} else {
    Log.i(LOG_TAG, "想定外のタグデータ受信です: data = " + messageData);
}
```

# NfcReaderActivityの説明 - 電話発信

```
// 1.電話をかけるIntentを投げる
```

```
startActivity(new Intent(Intent.ACTION_CALL, Uri.parse("tel:09099999999")));
```

暗黙的Intentを利用

属性(第一引数)と、URI(第二引数)を指定する。

# 暗黙的Intentの例

## ブラウザを開く

```
Intent bi = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(bi);
```

## Mapを開く

```
Intent mi = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:0,0?q=Tokyo"));
startActivity(mi);
```

# WeatherSpeechActivityの説明

- ・天気読み上げのActivity
- ・livedoorの「お天気Webサービス」からJSON情報取得

[http://weather.livedoor.com/weather\\_hacks/webservice](http://weather.livedoor.com/weather_hacks/webservice)

- ・Googleの「GoogleTTS」を利用して音声ファイルを取得  
(GoogleTTSはGoogle翻訳などに利用されている)

詳細は割愛します(明日必要であれば説明)

# RelayControlActivityの説明

- リレー制御のActivity
- AndroidとArduinoのUSB通信を行っている
- 色々処理を行っているように見えますが、  
ほとんどがAndroidとArduinoのUSB通信用の定型文  
⇒ 独自処理はSendRelayControlCommandクラスくらい。
- 対応するArduino側のソースコードも必要になります。

詳細は割愛します(明日必要であれば説明)

(時間があれば)ソースコード解説

# Agenda

- “NFC歌留多”デモ サンプル解説
- NFCハッカソン
- アイディアソン+ハッカソンで投票

# NFCハッカソン

- Hacking + marathon = Hackathon

- プログラミングばりばりかける方

⇒ アイデアソンのネタを実装してみよう！

- プログラミングいまいち...な方

⇒ デモに手を加えてみたり、

“Android SDK逆引きハンドブック”(書籍)を参照して

Androidアプリケーション開発をしてみよう！

# Agenda

- “NFC歌留多”デモ サンプル解説
- NFCハッカソン
- アイディアソン+ハッカソンで投票