

Android y NFC



Android y NFC

Temas

- Conceptos

- NFC
- Tags
- NDEF

- Android + NFC

- Ejemplos

- Parte 1 : NDEF Writer
- Parte 2 : NDEF Reader
- Parte 3 : Auto-inicio de una aplicación que detecta un registro de la etiqueta NDEF



Lo básico: Modos de funcionamiento de NFC

- **Modo Peer-to-peer**

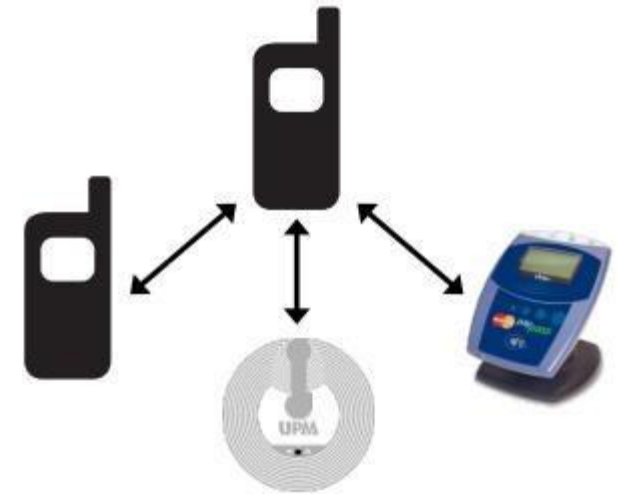
- La comunicación entre dos dispositivos NFC
- Android : Android Beam

- **Modo Reader/Writer**

- Dispositivos NFC para leer y escribir etiquetas NFC
- Algunos dispositivos NFC para leer y escribir con otras etiquetas y tarjetas inteligentes
- Android : Soporte para la mayoría de las etiquetas y tarjetas inteligentes compatibles con NFC

- **Modo de emulación de la tarjeta**

- El dispositivo NFC imita una tarjeta inteligente sin contacto
- Destinado a aplicaciones críticas de seguridad (pago y otros)
- No ha sido diseñado para la emulación de etiquetas NFC
 - En su lugar es el uso del modo peer- to-peer!
- Android : Sin apoyo oficial



Lo básico: Etiquetas NFC

- El NFC Forum define 4 plataformas de etiquetas estándar
 - Tipo 1
 - Innovision Jewel/Topaz
 - Tipo 2
 - NXP MIFARE Ultralight / Ultralight C
 - NXP NTAG203
 - Infineon my-d move / my-d NFC
 - Tipo 3
 - Sony FeliCa
 - Tipo 4
 - NXP MIFARE DESFire
 - Implantable en cualquier tarjeta inteligente sin contacto con soporte ISO 7816-4 (por ejemplo, en cualquier JavaCard)



Lo básico: Formato para el intercambio de datos con NFC (NDEF)

- Formato común para ...
 - Almacenamiento de datos en las etiquetas NFC
 - Transmisión de datos en modo peer-to-peer
- Capa de abstracción de datos
 - Idea : Las aplicaciones deben trabajar con cualquier plataforma de etiquetas NFC
 - Ventaja : La misma API es para leer y almacenar datos en cualquier etiqueta NFC
 - Los usuarios pueden elegir cualquier plataforma de etiquetas (con almacenamiento suficiente)!
- Formato
 - El registro NDEF es un contenedor de datos con la información del tipo de datos
 - El mensaje NDEF es una serie de uno o más registros NDEF
 - La etiqueta NFC contiene el mensaje NDEF

Lo básico: Los registros NDEF

- 4 grupos de tipos de registro

- Tipos conocidos del NFC Forum

- Definido por el NFC Forum
 - Por ejemplo, URI (**urn:NFC:wkt:U**) , Text (**urn:nfc:wkt:T**) , SmartPoster (**urn:nfc:wkt:Sp**)

- Tipos externos del NFC Forum

- Tipos de registro definibles por los desarrolladores de aplicaciones
 - Por ejemplo, **urn:nfc:ext:mroland.at:example**

- Los tipos de contenido MIME

- Los formatos de datos definidos por los tipos de contenido MIME
 - Por ejemplo, Tarjetas de Negocios (**text/x-vCard**)

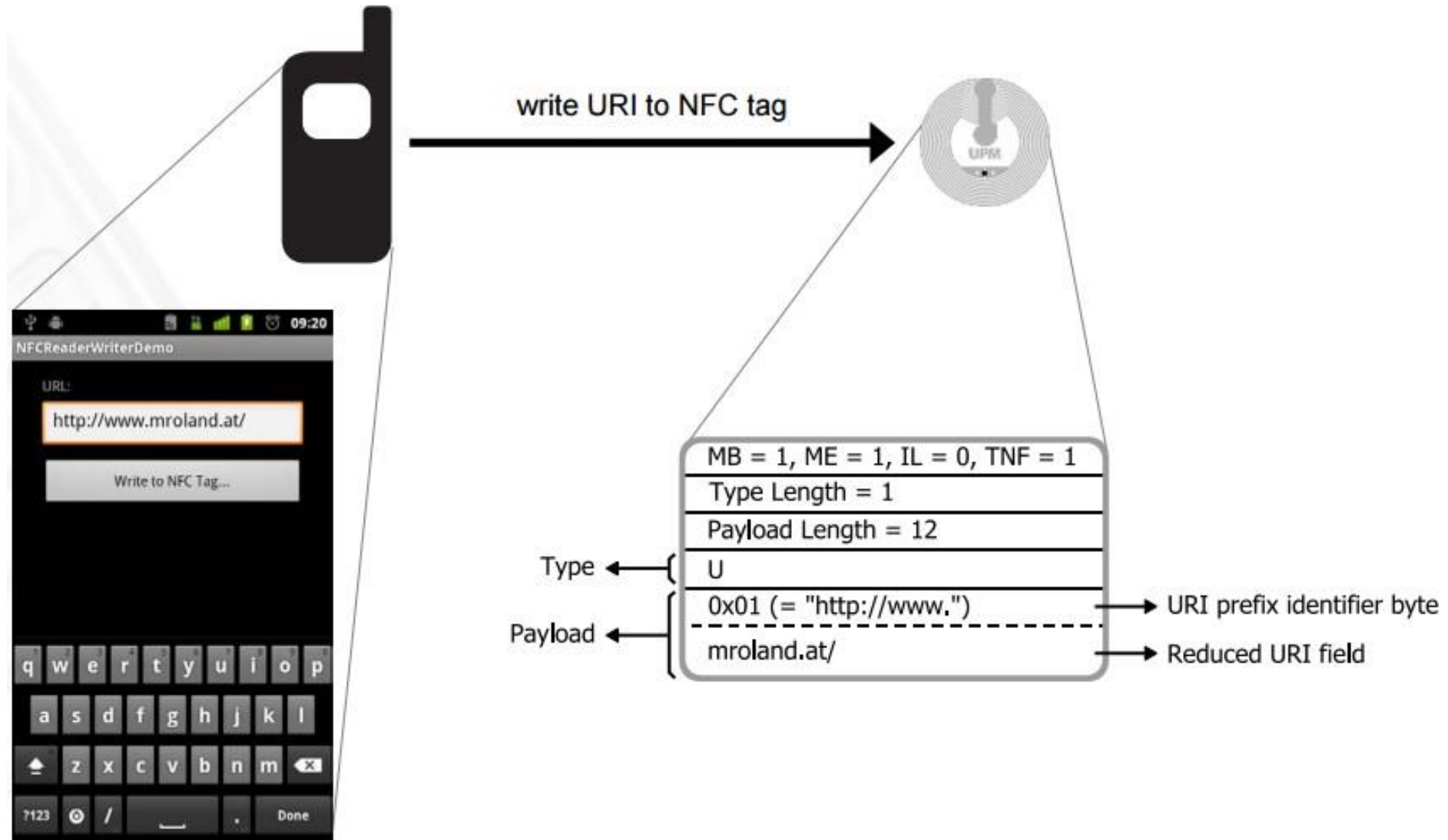
- Tipos absolutos de URI

- Los formatos de datos definidos por URIs
 - Importante : No se debe confundir esto con el tipo usual URI!

Características de NFC en Android

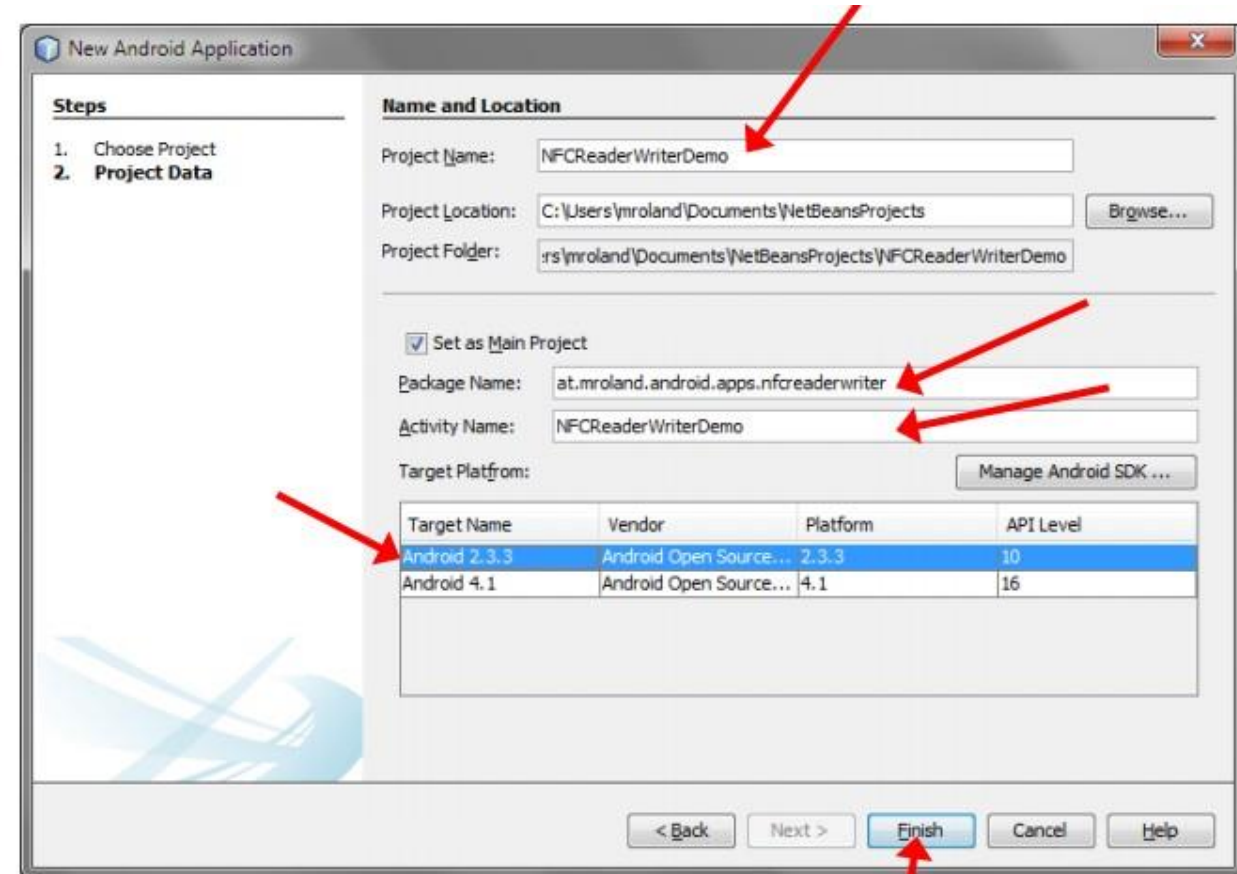
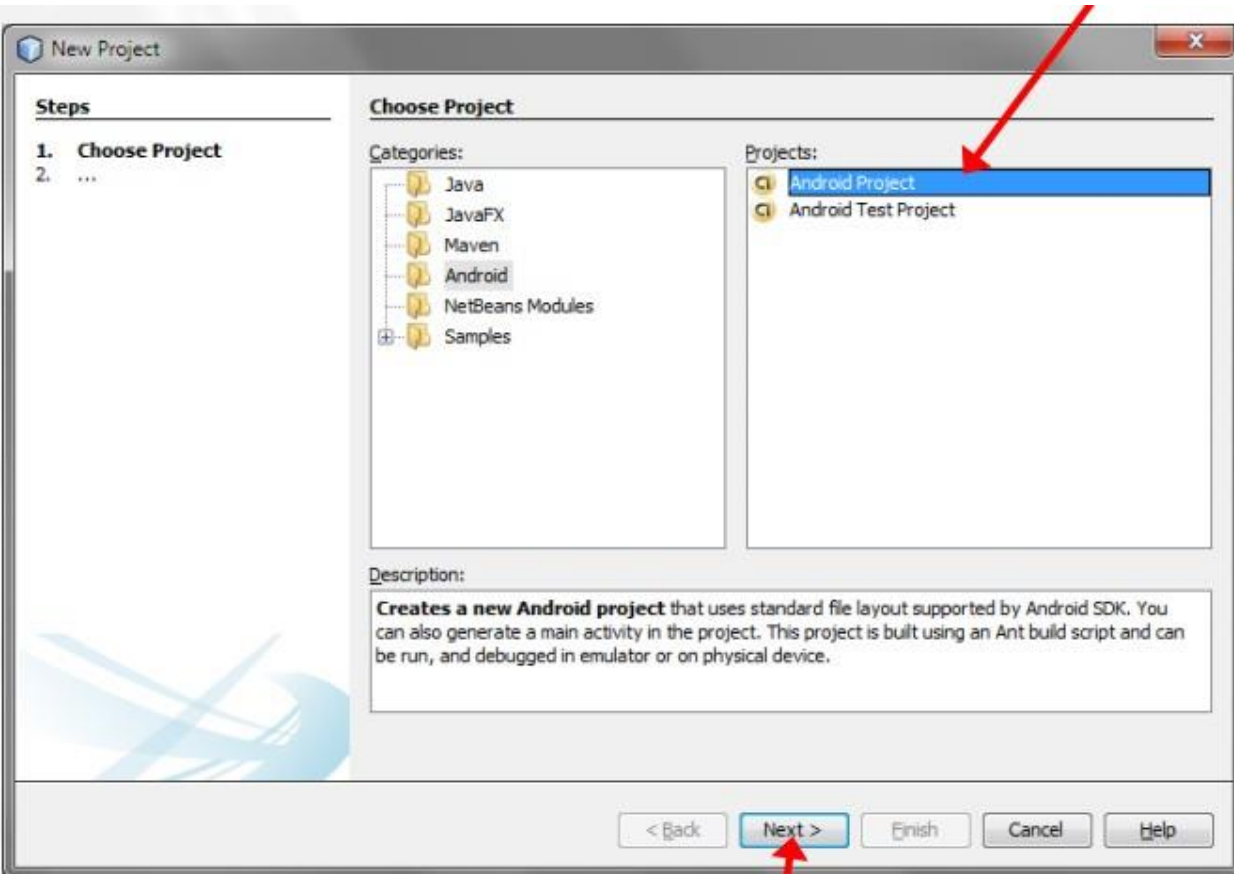
- **Peer- to-peer modo de**
 - Android Beam : Transferencia de mensajes NDEF y archivos (grandes)
- **Modo de lector / escritor**
 - Leer y escribir datos de NDEF en las etiquetas NFC
 - La comunicación con otras etiquetas y tarjetas inteligentes
 - Protocolos basados en ISO/IEC 14443-A
 - ISO/IEC 7816-4 APDUs
 - FeliCa (Lite)
 - ISO/IEC 15693 (sólo si el chipset NFC lo soporta)
 - MIFARE Classic (sólo si el chipset NFC lo soporta)
 - Aplicaciones de Auto-inicio al detectar
 - un determinado registro NDEF
 - una cierta tecnología de etiquetas
 - etiquetas no manejadas por otras aplicaciones

Ejemplo (Parte 1): Aplicación con NDEF Writer



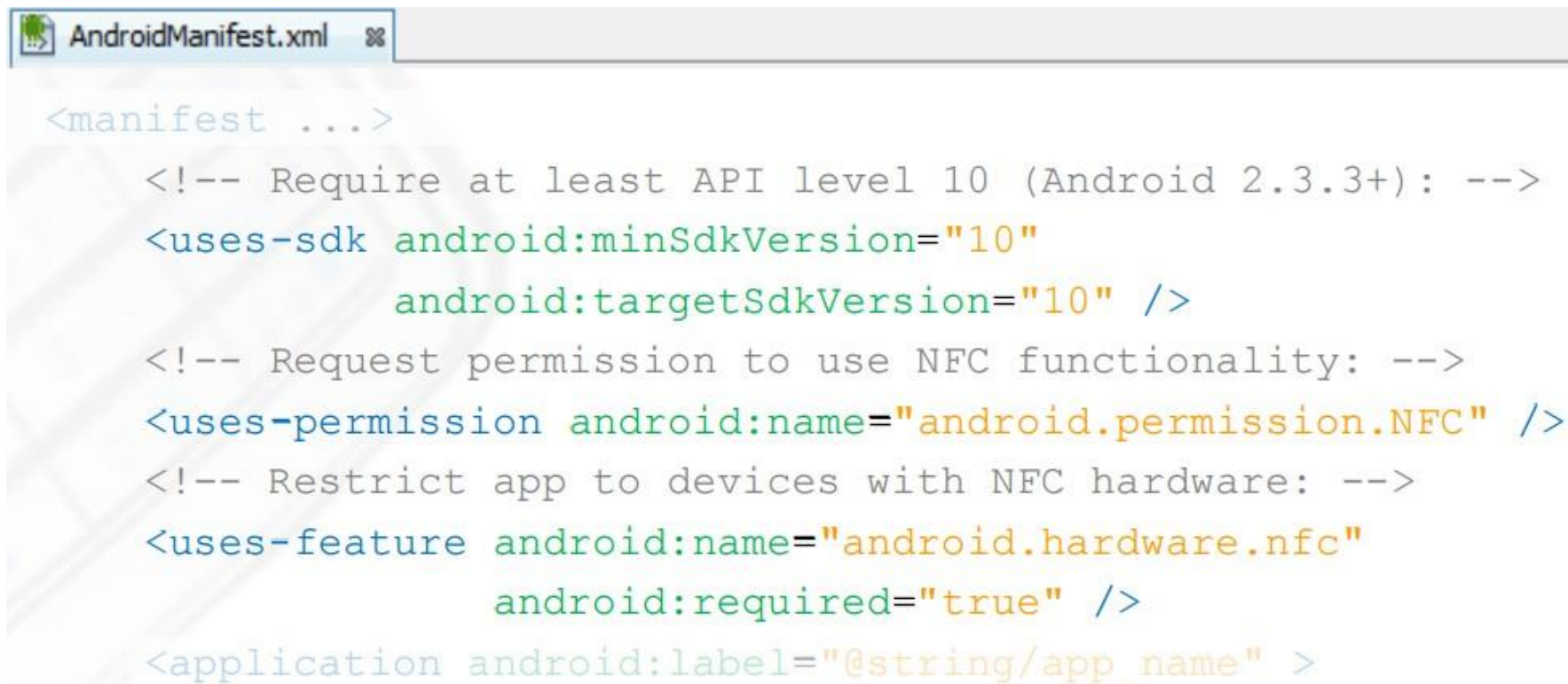
Paso 1: Crear un nuevo proyecto de Android

- File > New Project...

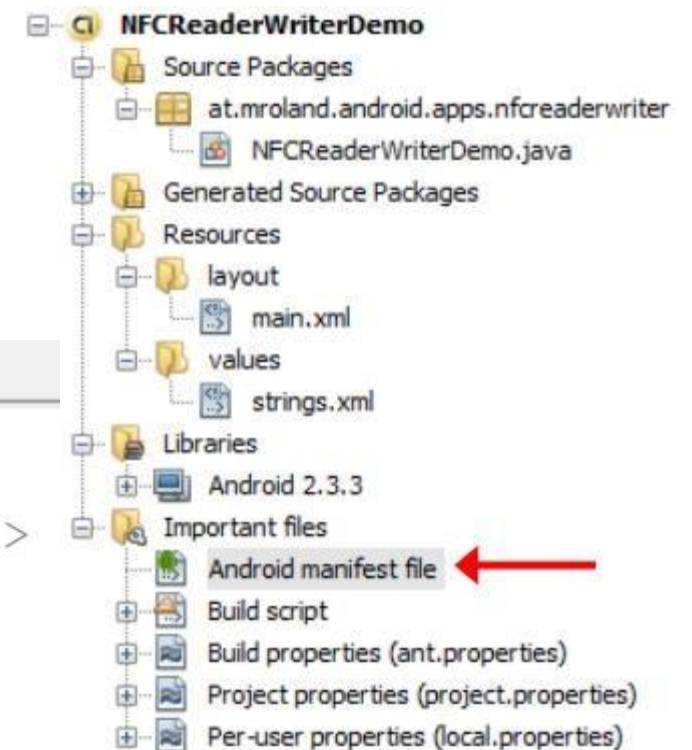


Paso 2: El AndroidManifest y NFC

- Se requiere Android 2.3.3 o posterior
- Solicitud de permiso para usar NFC
- Se requieren dispositivos con hardware NFC



```
<manifest ...>
  <!-- Require at least API level 10 (Android 2.3.3+): -->
  <uses-sdk android:minSdkVersion="10"
            android:targetSdkVersion="10" />
  <!-- Request permission to use NFC functionality: -->
  <uses-permission android:name="android.permission.NFC" />
  <!-- Restrict app to devices with NFC hardware: -->
  <uses-feature android:name="android.hardware.nfc"
                android:required="true" />
  <application android:label="@string/app_name" >
```



Paso 3: Agregar una interface de usuario

The image displays the XML layout code for an Android application's main interface, alongside a visual preview of the app running on a mobile device. The XML code is for a file named `main.xml` and defines a vertical `ScrollView` containing a vertical `LinearLayout`. Inside the `LinearLayout`, there are three UI components: a `TextView` for displaying a URL, an `EditText` for entering a URL, and a `Button` for writing to an NFC tag.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent" android:layout_height="match_parent" >
    <LinearLayout android:orientation="vertical"
        android:layout_width="match_parent" android:layout_height="wrap_content" >

        <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
            android:text="URL:"
            android:layout_marginLeft="30dp" android:layout_marginRight="30dp"
            android:layout_marginTop="15dp" android:layout_marginBottom="5dp" />

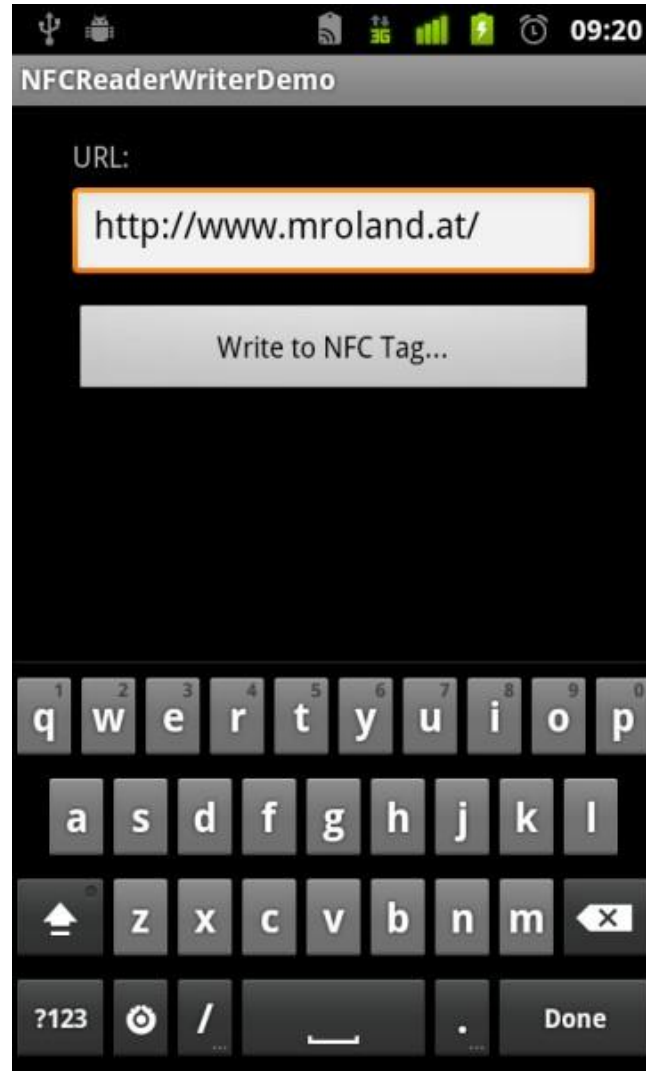
        <EditText android:id="@+id/myUrl"
            android:layout_width="match_parent" android:layout_height="wrap_content"
            android:text="http://www.mroland.at/"
            android:inputType="textUri"
            android:layout_marginLeft="30dp" android:layout_marginRight="30dp"
            android:layout_marginTop="0dp" android:layout_marginBottom="10dp" />

        <Button android:id="@+id/myWriteUrlButton"
            android:layout_width="match_parent" android:layout_height="wrap_content"
            android:text="Write to NFC Tag..."
            android:layout_marginLeft="30dp" android:layout_marginRight="30dp"
            android:layout_marginTop="0dp" android:layout_marginBottom="15dp"
            android:gravity="center" />

    </LinearLayout>
</ScrollView>
```

The visual preview on the right shows the application running on a mobile device. The title bar indicates the app is named "NFCReaderWriterDemo". The interface consists of a label "URL:" followed by a text input field containing the URL "http://www.mroland.at/". Below the input field is a button labeled "Write to NFC Tag...". A keyboard is visible at the bottom of the screen.

Lo que se tiene hasta el momento...



Paso 4: Creación de la actividad

NFCReaderWriterDemo.java

```
private static final int DIALOG_WRITE_URL = 1;
private EditText mMyUrl;
private Button mMyWriteUrlButton;
private boolean mWriteUrl = false;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mMyUrl = (EditText) findViewById(R.id.myUrl);
    mMyWriteUrlButton = (Button) findViewById(R.id.myWriteUrlButton);

    // Set action for "Write URL to tag..." button:
    mMyWriteUrlButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            mWriteUrl = true;
            NFCReaderWriterDemo.this.showDialog(DIALOG_WRITE_URL);
        }
    });
}
```

NFCReaderWriterDemo

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java ←
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)



Paso 5: Un cuadro de diálogo: Listo para escribir en una etiqueta

NFCReaderWriterDemo.java

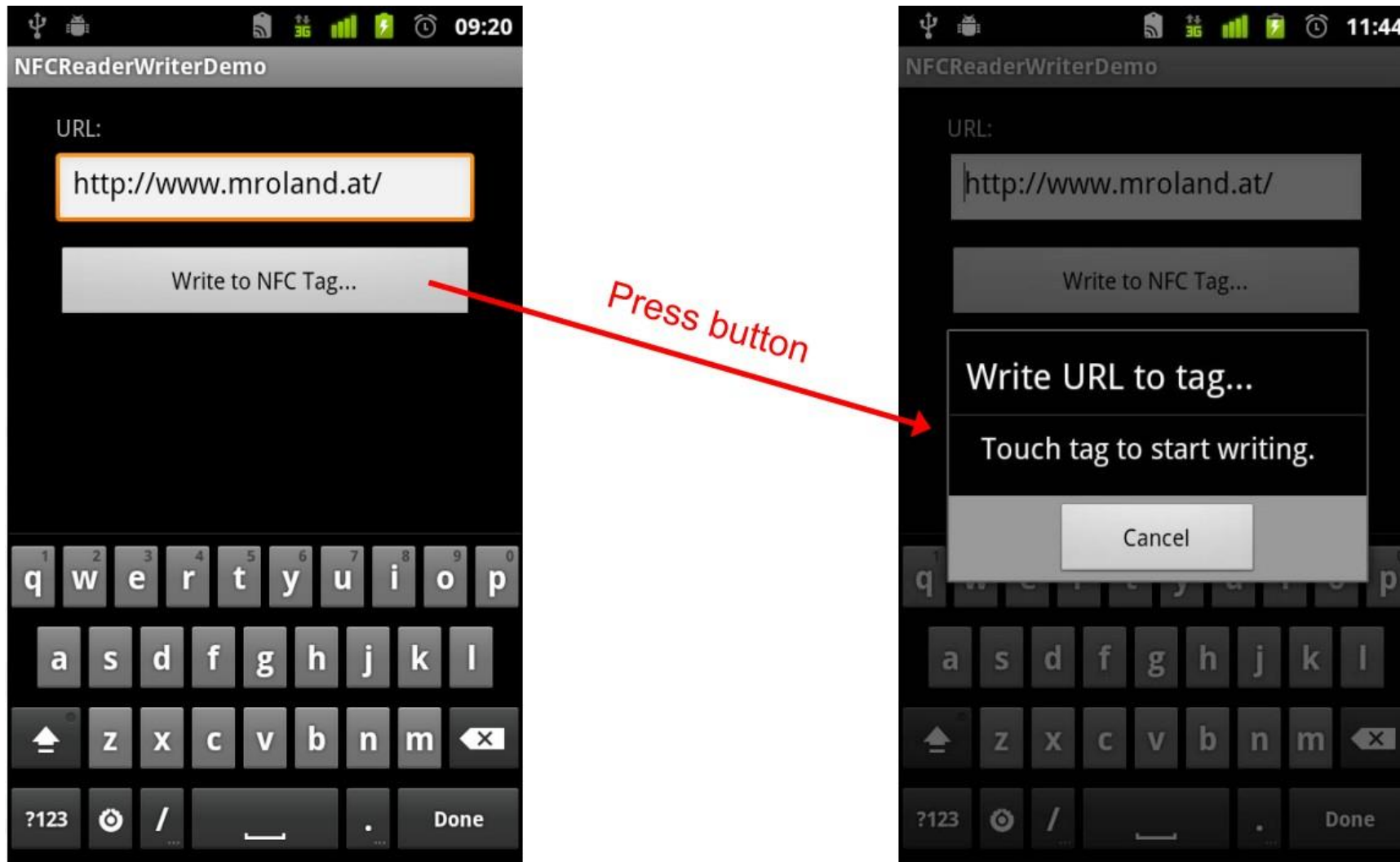
```
@Override
protected Dialog onCreateDialog(int id, Bundle args) {
    switch (id) {
        case DIALOG_WRITE_URL:
            return new AlertDialog.Builder(this)
                .setTitle("Write URL to tag...")
                .setMessage("Touch tag to start writing.")
                .setCancelable(true)
                .setNeutralButton(android.R.string.cancel,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface d, int arg) {
                            d.cancel();
                        }
                    })
                .setOnCancelListener(new DialogInterface.OnCancelListener() {
                    public void onCancel(DialogInterface d) {
                        mWriteUrl = false;
                    }
                })
                .create();

            return null;
    }
}
```

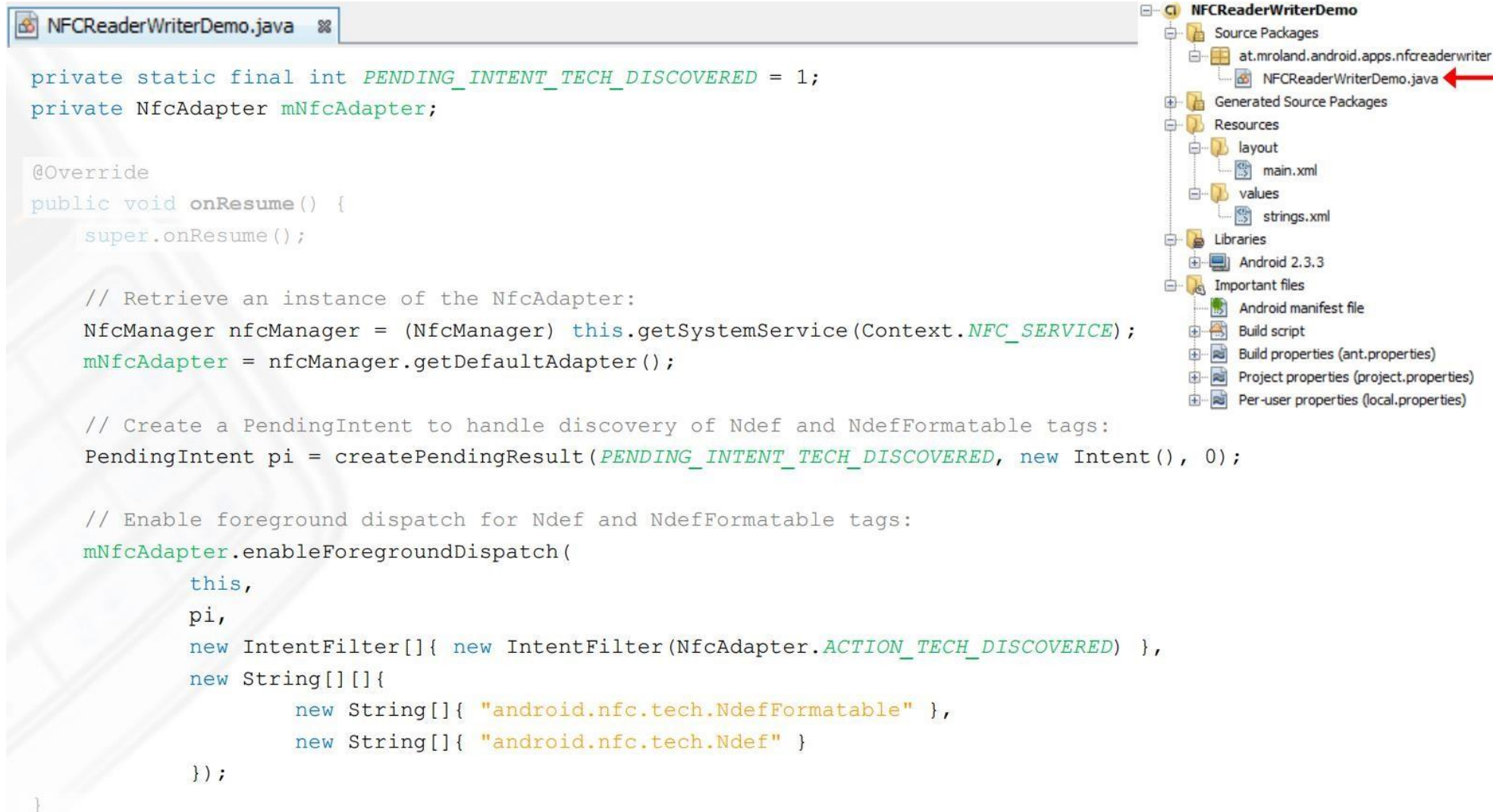
NFCReaderWriterDemo

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java ←
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Lo que se tiene hasta el momento...



Paso 6: Despacho en primer plano (Detección de etiquetas)



The screenshot displays the Android Studio interface. The main editor window shows the `NFCReaderWriterDemo.java` file with the following code:

```
private static final int PENDING_INTENT_TECH_DISCOVERED = 1;
private NfcAdapter mNfcAdapter;

@Override
public void onResume() {
    super.onResume();

    // Retrieve an instance of the NfcAdapter:
    NfcManager nfcManager = (NfcManager) this.getSystemService(Context.NFC_SERVICE);
    mNfcAdapter = nfcManager.getDefaultAdapter();

    // Create a PendingIntent to handle discovery of Ndef and NdefFormatable tags:
    PendingIntent pi = createPendingResult(PENDING_INTENT_TECH_DISCOVERED, new Intent(), 0);

    // Enable foreground dispatch for Ndef and NdefFormatable tags:
    mNfcAdapter.enableForegroundDispatch(
        this,
        pi,
        new IntentFilter[]{ new IntentFilter(NfcAdapter.ACTION_TECH_DISCOVERED) },
        new String[][]{
            new String[]{ "android.nfc.tech.NdefFormatable" },
            new String[]{ "android.nfc.tech.Ndef" }
        });
}
```

The right-hand pane shows the project structure for `NFCReaderWriterDemo`. The `Source Packages` folder is expanded, showing the package `at.mroland.android.apps.nfcreaderwriter` and the file `NFCReaderWriterDemo.java`, which is highlighted with a red arrow. Other folders visible include `Generated Source Packages`, `Resources` (containing `layout` with `main.xml` and `values` with `strings.xml`), `Libraries` (containing `Android 2.3.3`), and `Important files` (containing `Android manifest file`, `Build script`, `Build properties (ant.properties)`, `Project properties (project.properties)`, and `Per-user properties (local.properties)`).

Paso 7: Limpieza del despacho de primer plano



The screenshot displays the Android Studio interface. On the left, the 'NFCReaderWriterDemo.java' file is open, showing the following code:

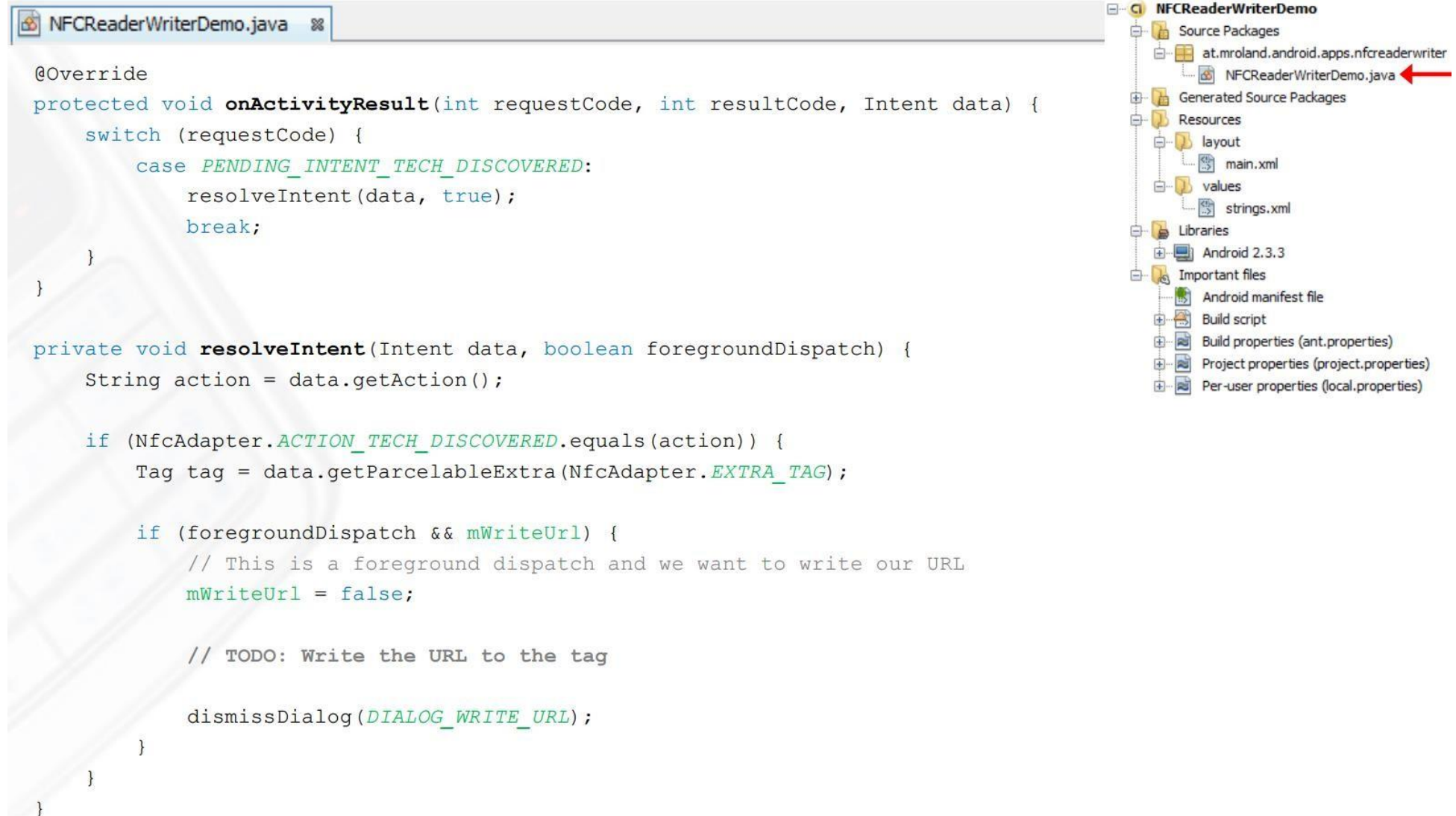
```
@Override
public void onPause() {
    super.onPause();

    // Disable foreground dispatch:
    mNfcAdapter.disableForegroundDispatch(this);
}
```

On the right, the 'Project' tab shows the project structure for 'NFCReaderWriterDemo':

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java (highlighted with a red arrow)
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Paso 8: Recibir el intento del despacho de primer plano



The screenshot displays the Android Studio interface. The main editor window shows the `NFCReaderWriterDemo.java` file with the following code:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case PENDING_INTENT_TECH_DISCOVERED:
            resolveIntent(data, true);
            break;
    }
}

private void resolveIntent(Intent data, boolean foregroundDispatch) {
    String action = data.getAction();

    if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {
        Tag tag = data.getParcelableExtra(NfcAdapter.EXTRA_TAG);

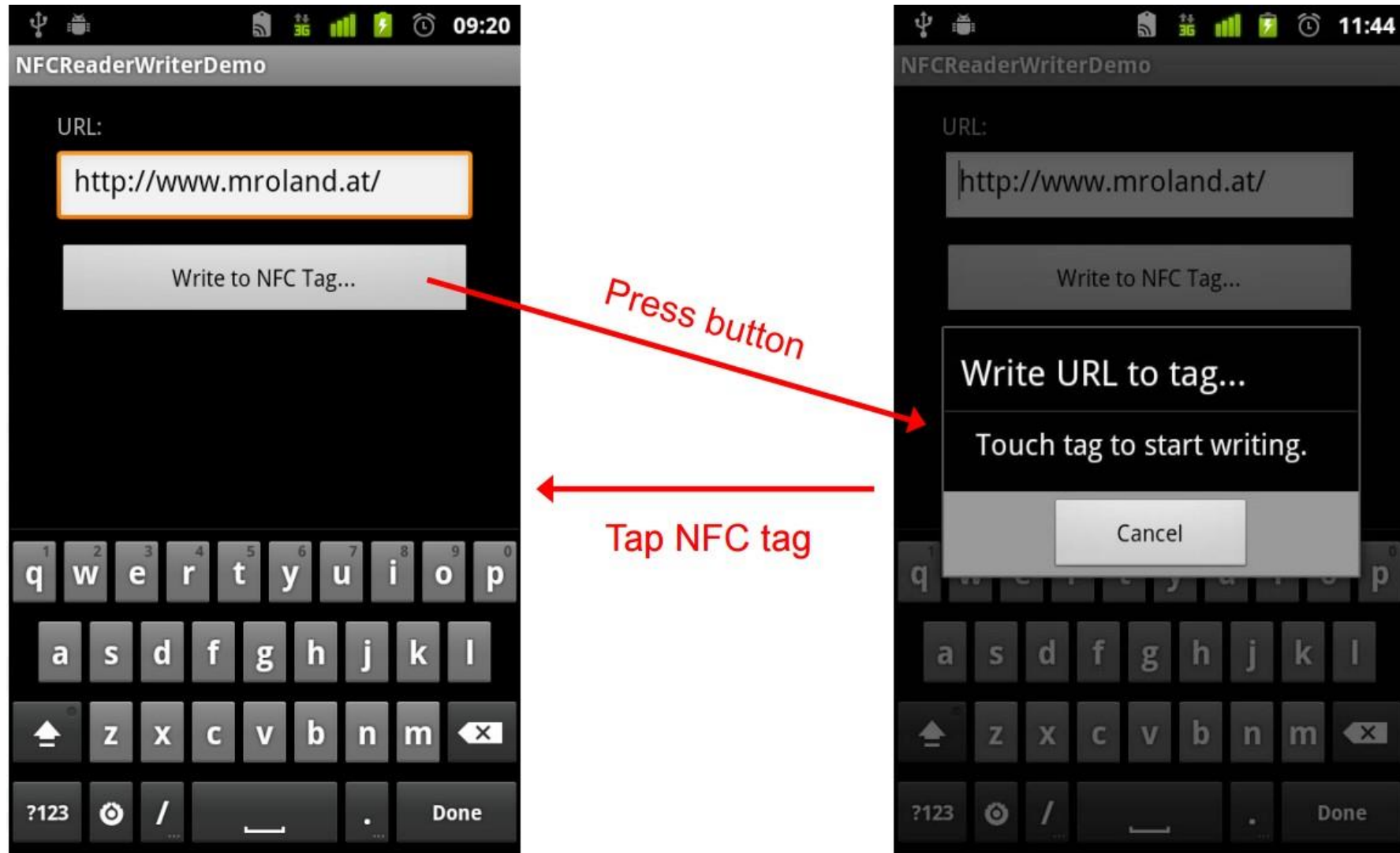
        if (foregroundDispatch && mWriteUrl) {
            // This is a foreground dispatch and we want to write our URL
            mWriteUrl = false;

            // TODO: Write the URL to the tag

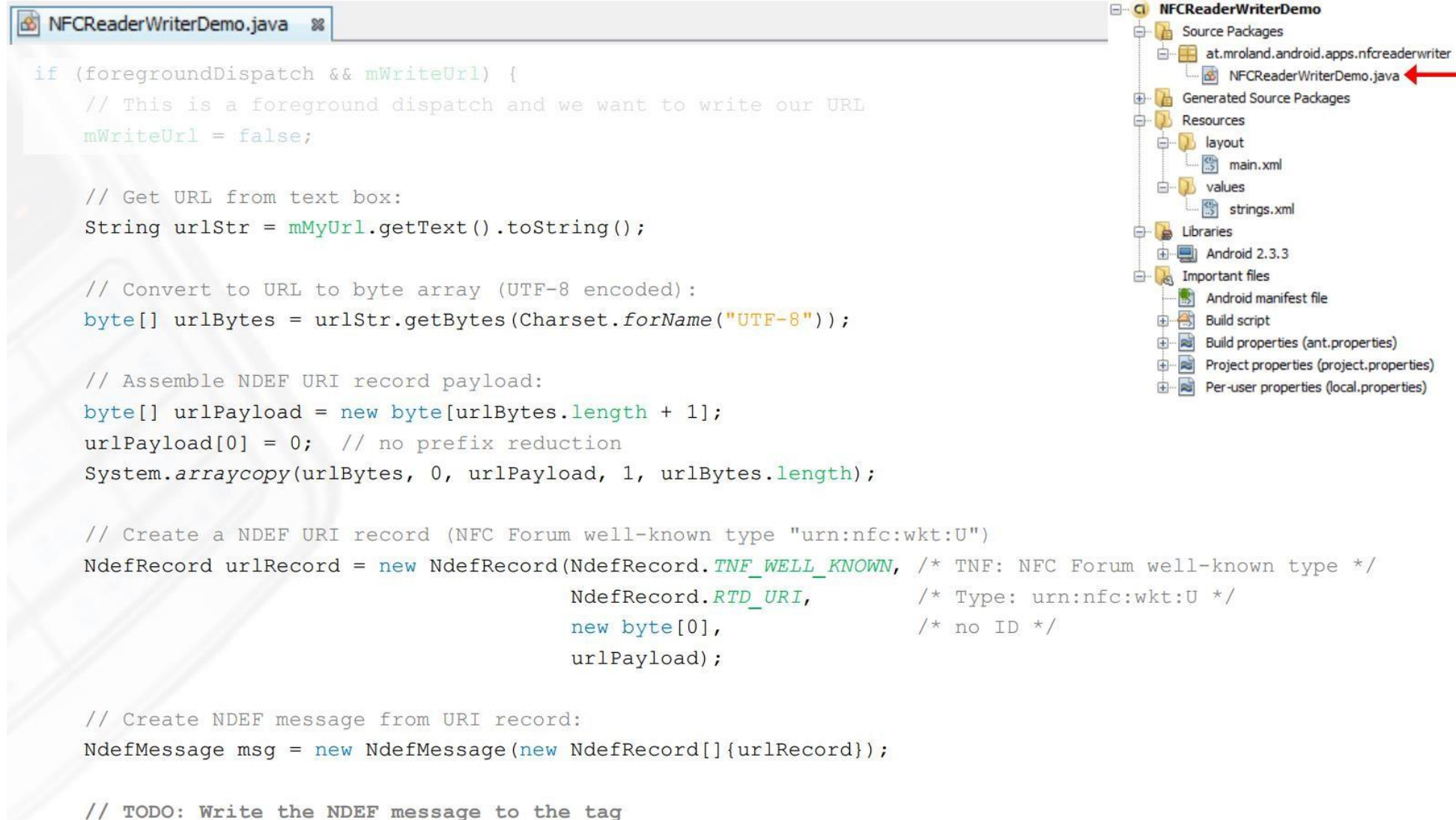
            dismissDialog(DIALOG_WRITE_URL);
        }
    }
}
```

The right-hand pane shows the project structure for `NFCReaderWriterDemo`. A red arrow points to the `NFCReaderWriterDemo.java` file located under the `Source Packages` folder, specifically within the `at.mroland.android.apps.nfcreaderwriter` package.

Lo que se tiene hasta el momento...



Paso 9: Preparar el mensaje URI NDEF



The screenshot displays the Android Studio interface. On the left, the `NFCReaderWriterDemo.java` file is open, showing the following code:

```
if (foregroundDispatch && mWriteUrl) {  
    // This is a foreground dispatch and we want to write our URL  
    mWriteUrl = false;  
  
    // Get URL from text box:  
    String urlStr = mMyUrl.getText().toString();  
  
    // Convert to URL to byte array (UTF-8 encoded):  
    byte[] urlBytes = urlStr.getBytes(Charset.forName("UTF-8"));  
  
    // Assemble NDEF URI record payload:  
    byte[] urlPayload = new byte[urlBytes.length + 1];  
    urlPayload[0] = 0; // no prefix reduction  
    System.arraycopy(urlBytes, 0, urlPayload, 1, urlBytes.length);  
  
    // Create a NDEF URI record (NFC Forum well-known type "urn:nfc:wkt:U")  
    NdefRecord urlRecord = new NdefRecord(NdefRecord.TNF_WELL_KNOWN, /* TNF: NFC Forum well-known type */  
                                           NdefRecord.RTD_URI, /* Type: urn:nfc:wkt:U */  
                                           new byte[0], /* no ID */  
                                           urlPayload);  
  
    // Create NDEF message from URI record:  
    NdefMessage msg = new NdefMessage(new NdefRecord[]{urlRecord});  
  
    // TODO: Write the NDEF message to the tag
```

On the right, the Project Explorer shows the project structure for `NFCReaderWriterDemo`. A red arrow points to the `NFCReaderWriterDemo.java` file located under `Source Packages` > `at.mroland.android.apps.nfcreaderwriter`.

Paso 10: Escribir el NDEF a una etiqueta preformateada

NFCReaderWriterDemo.java

```
Tag tag = data.getParcelableExtra(NfcAdapter.EXTRA_TAG);
[...]

// Create NDEF message from URI record:
NdefMessage msg = new NdefMessage(new NdefRecord[]{urlRecord});

Ndef ndefTag = Ndef.get(tag);
if (ndefTag != null) {
    // Our tag is already formatted, we just need to write our message

    try {
        // Connect to tag:
        ndefTag.connect();

        // Write NDEF message:
        ndefTag.writeNdefMessage(msg);
    } catch (Exception e) {
    } finally {
        // Close connection:
        try { ndefTag.close(); } catch (Exception e) { }
    }
} else {
    // Our tag is not NDEF formatted!
}

dismissDialog(DIALOG_WRITE_URL);
```

NFCReaderWriterDemo

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java ←
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)



Paso 11: Escribir el NDEF a la etiqueta no formateada

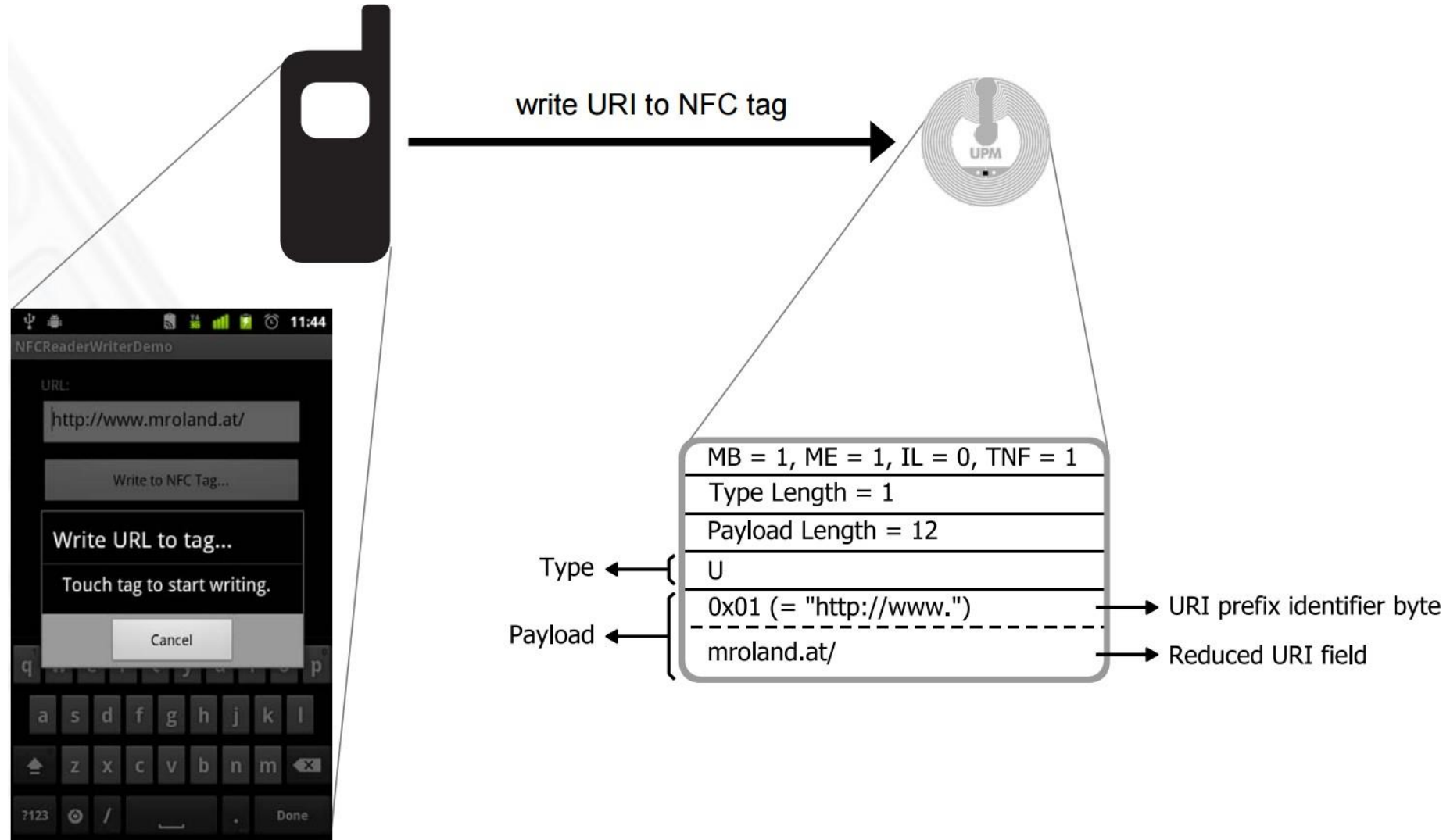
NFCReaderWriterDemo.java

```
if (ndefTag != null) {  
    [...]  
} else {  
    // Our tag is not NDEF formatted!  
    NdefFormatable ndefFormatableTag = NdefFormatable.get(tag);  
    if (ndefFormatableTag != null) {  
        // Our tag is not yet formatted, we need to format it with our message  
  
        try {  
            // Connect to tag:  
            ndefFormatableTag.connect();  
  
            // Format with NDEF message:  
            ndefFormatableTag.format(msg);  
        } catch (Exception e) {  
        } finally {  
            // Close connection:  
            try { ndefFormatableTag.close(); } catch (Exception e) { }  
        }  
    }  
}  
  
dismissDialog(DIALOG_WRITE_URL);
```

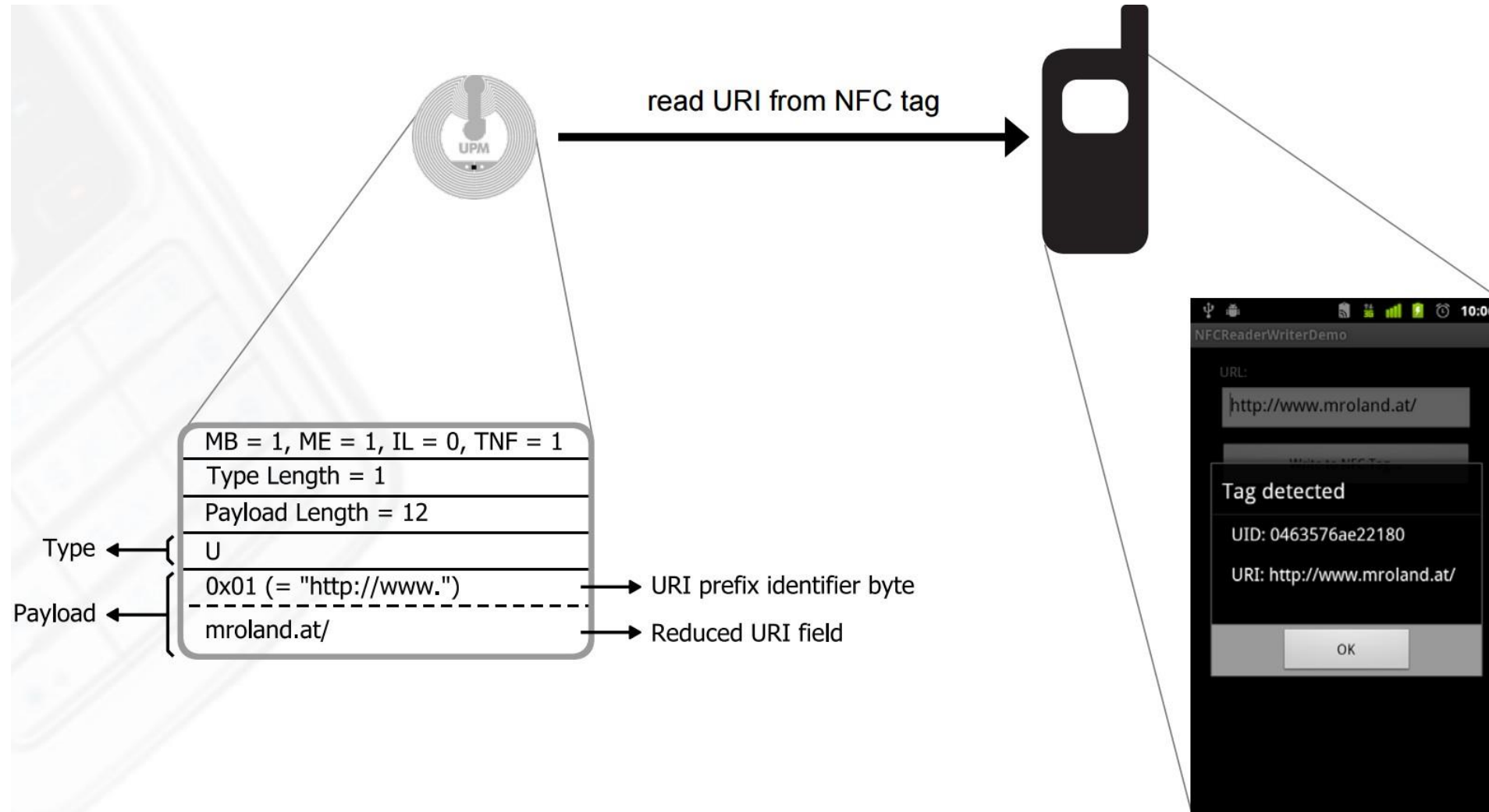
NFCReaderWriterDemo

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

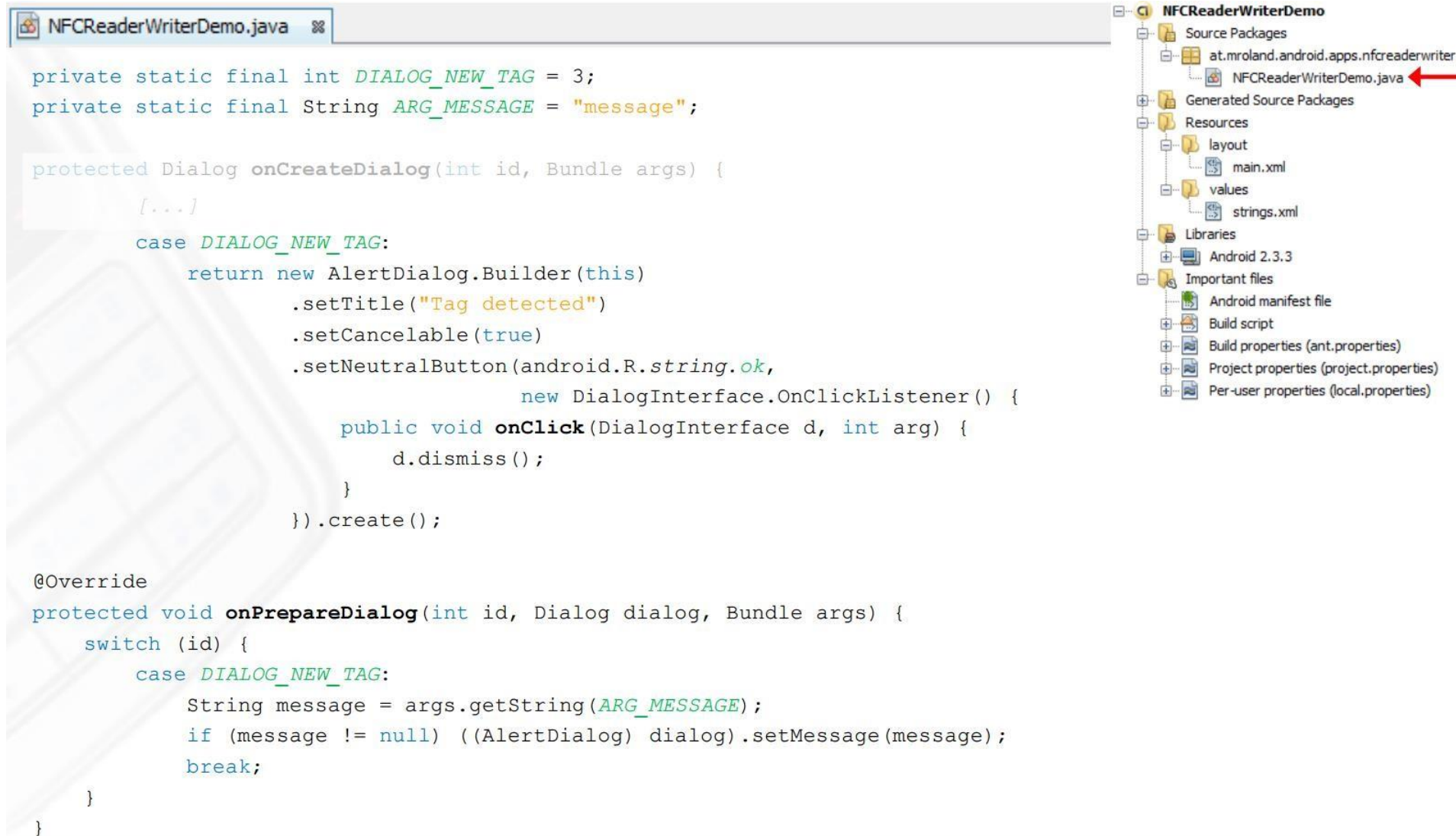
Lo que se tiene hasta el momento: [Aplicación NDEF Writer](#)



Ejemplo (Parte 2): Aplicación con NDEF Reader



Paso 1: Una caja de diálogo : Se detectó una etiqueta



The screenshot displays the Android Studio interface. The main editor window shows the `NFCReaderWriterDemo.java` file with the following code:

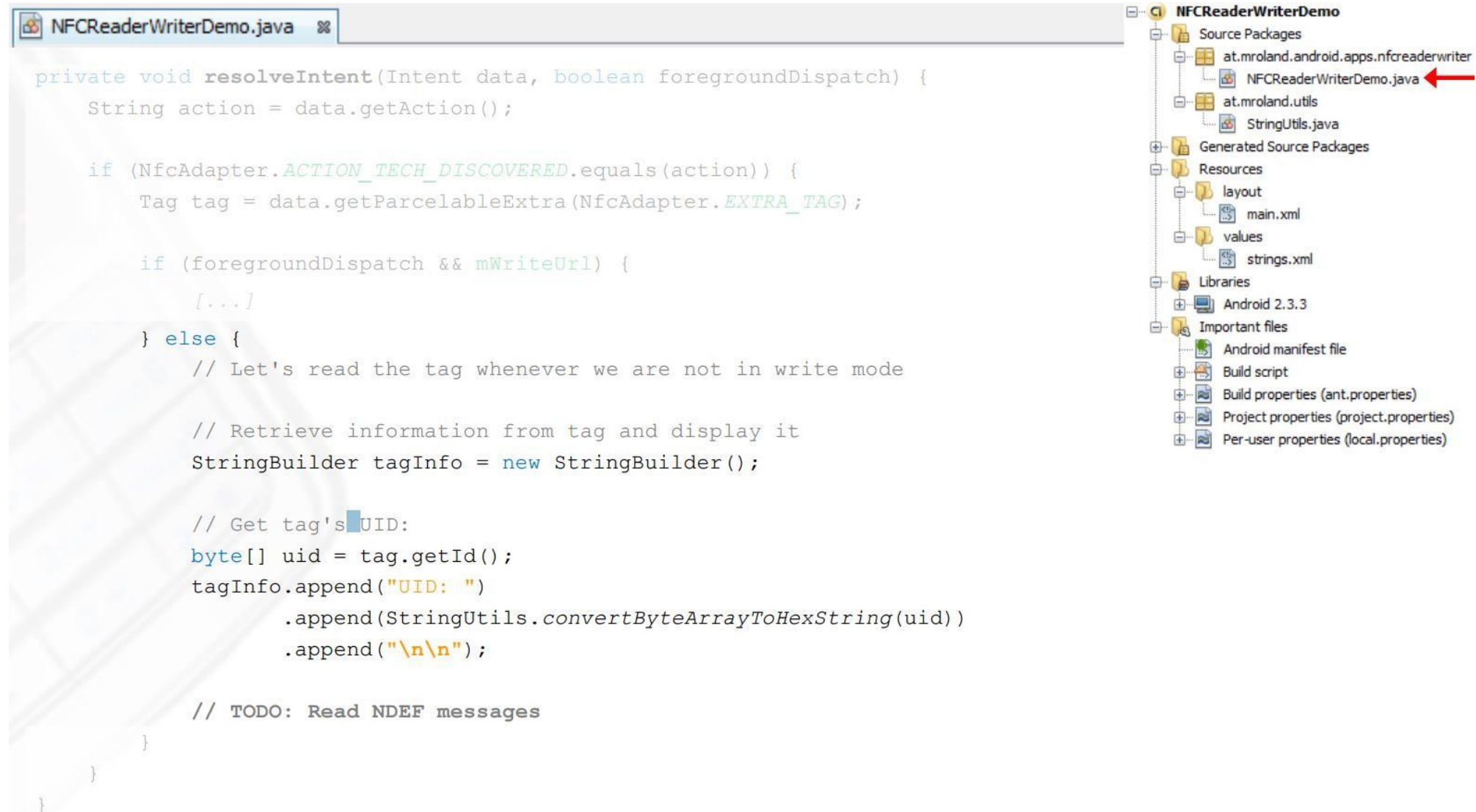
```
private static final int DIALOG_NEW_TAG = 3;
private static final String ARG_MESSAGE = "message";

protected Dialog onCreateDialog(int id, Bundle args) {
    [...]
    case DIALOG_NEW_TAG:
        return new AlertDialog.Builder(this)
            .setTitle("Tag detected")
            .setCancelable(true)
            .setNeutralButton(android.R.string.ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface d, int arg) {
                        d.dismiss();
                    }
                })
            .create();
}

@Override
protected void onPrepareDialog(int id, Dialog dialog, Bundle args) {
    switch (id) {
        case DIALOG_NEW_TAG:
            String message = args.getString(ARG_MESSAGE);
            if (message != null) ((AlertDialog) dialog).setMessage(message);
            break;
    }
}
```

The right-hand pane shows the project structure for `NFCReaderWriterDemo`. A red arrow points to the `NFCReaderWriterDemo.java` file in the `Source Packages` folder, which is located under `at.mroland.android.apps.nfcreaderwriter`. Other visible folders include `Generated Source Packages`, `Resources` (containing `layout` and `values`), and `Libraries` (containing `Android 2.3.3`). The `Important files` section lists the `Android manifest file`, `Build script`, `Build properties (ant.properties)`, `Project properties (project.properties)`, and `Per-user properties (local.properties)`.

Paso 2: Detección de la etiqueta



The screenshot displays the Android Studio interface. The main editor window shows the `NFCReaderWriterDemo.java` file with the following code:

```
private void resolveIntent(Intent data, boolean foregroundDispatch) {
    String action = data.getAction();

    if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {
        Tag tag = data.getParcelableExtra(NfcAdapter.EXTRA_TAG);

        if (foregroundDispatch && mWriteUrl) {
            [...]
        } else {
            // Let's read the tag whenever we are not in write mode

            // Retrieve information from tag and display it
            StringBuilder tagInfo = new StringBuilder();

            // Get tag's UID:
            byte[] uid = tag.getId();
            tagInfo.append("UID: ")
                .append(StringUtils.convertByteArrayToHexString(uid))
                .append("\n\n");

            // TODO: Read NDEF messages
        }
    }
}
```

The right-hand pane shows the project structure for `NFCReaderWriterDemo`:

- Source Packages
 - `at.mroland.android.apps.nfcreaderwriter`
 - `NFCReaderWriterDemo.java` (highlighted with a red arrow)
 - `at.mroland.utils`
 - `StringUtils.java`
- Generated Source Packages
- Resources
 - layout
 - `main.xml`
 - values
 - `strings.xml`
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Paso 3: Lectura del mensaje NDEF de la etiqueta

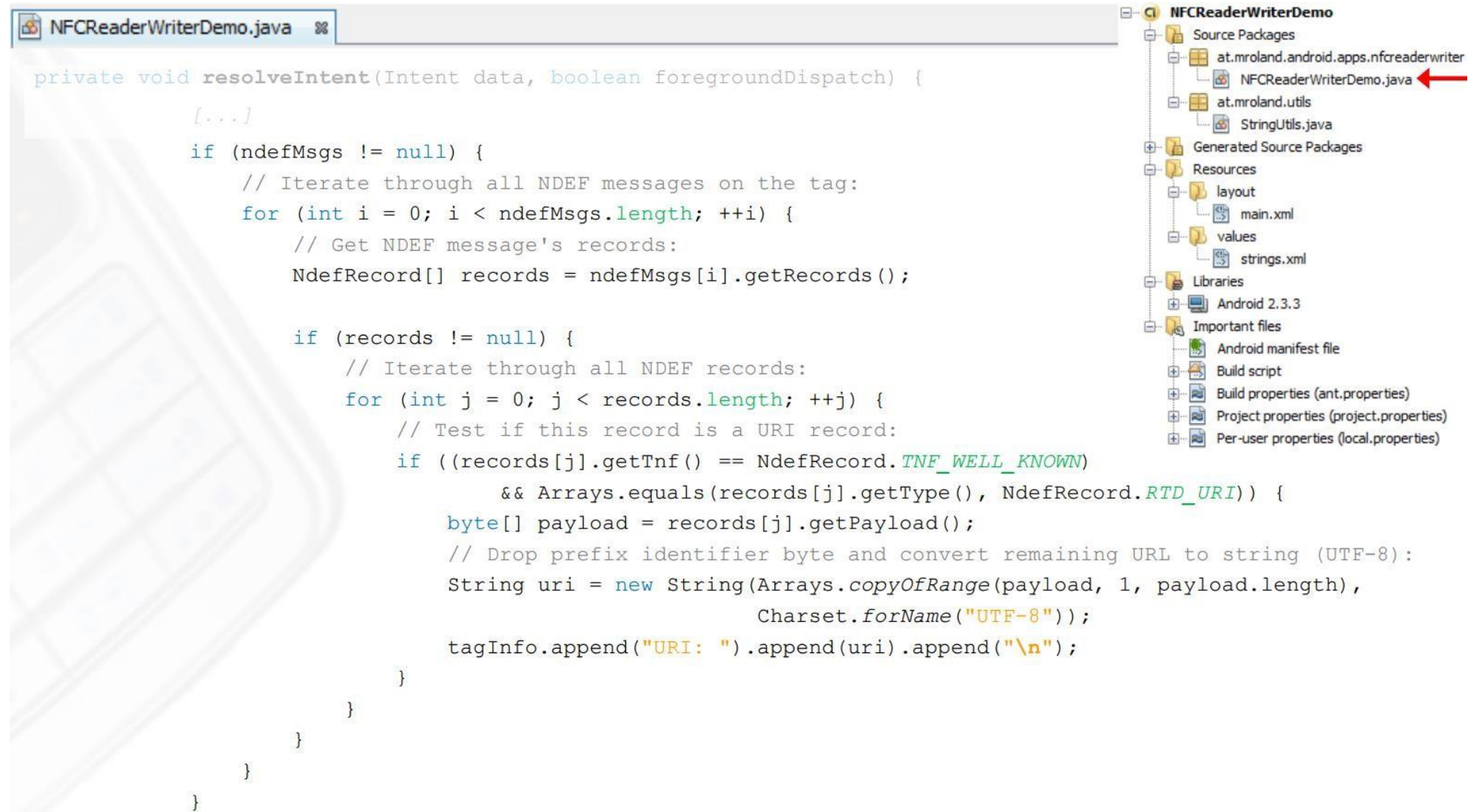


The screenshot displays the Android Studio IDE. The main editor window shows the `NFCReaderWriterDemo.java` file with the following code:

```
private void resolveIntent(Intent data, boolean foregroundDispatch) {  
    [...]  
    // Get tag's NDEF messages:  
    Parcelable[] ndefRaw =  
        data.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);  
    NdefMessage[] ndefMsgs = null;  
    if (ndefRaw != null) {  
        ndefMsgs = new NdefMessage[ndefRaw.length];  
        for (int i = 0; i < ndefMsgs.length; ++i) {  
            ndefMsgs[i] = (NdefMessage) ndefRaw[i];  
        }  
    }  
  
    // TODO: Find our URI record
```

The right-hand pane shows the project structure for `NFCReaderWriterDemo`. The `Source Packages` folder is expanded, showing the package `at.mroland.android.apps.nfcreaderwriter`. The file `NFCReaderWriterDemo.java` is highlighted with a red arrow, indicating its location within the project structure.

Paso 4: Encontrar el registro URI en los mensajes NDEF



The screenshot displays an IDE with the `NFCReaderWriterDemo.java` file open. The code implements the `resolveIntent` method, which iterates through NDEF messages and their records to find a URI. A red arrow in the project explorer points to the `NFCReaderWriterDemo.java` file.

```
private void resolveIntent(Intent data, boolean foregroundDispatch) {  
    [...]  
    if (ndefMsgs != null) {  
        // Iterate through all NDEF messages on the tag:  
        for (int i = 0; i < ndefMsgs.length; ++i) {  
            // Get NDEF message's records:  
            NdefRecord[] records = ndefMsgs[i].getRecords();  
  
            if (records != null) {  
                // Iterate through all NDEF records:  
                for (int j = 0; j < records.length; ++j) {  
                    // Test if this record is a URI record:  
                    if ((records[j].getTnf() == NdefRecord.TNF_WELL_KNOWN)  
                        && Arrays.equals(records[j].getType(), NdefRecord.RTD_URI)) {  
                        byte[] payload = records[j].getPayload();  
                        // Drop prefix identifier byte and convert remaining URL to string (UTF-8):  
                        String uri = new String(Arrays.copyOfRange(payload, 1, payload.length),  
                                                Charset.forName("UTF-8"));  
                        tagInfo.append("URI: ").append(uri).append("\n");  
                    }  
                }  
            }  
        }  
    }  
}
```

Project Structure:

- NFCReaderWriterDemo
 - Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java (highlighted with a red arrow)
 - at.mroland.utils
 - StringUtils.java
 - Generated Source Packages
 - Resources
 - layout
 - main.xml
 - values
 - strings.xml
 - Libraries
 - Android 2.3.3
 - Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Paso 5: Mostrar los datos de la etiqueta en un cuadro de diálogo



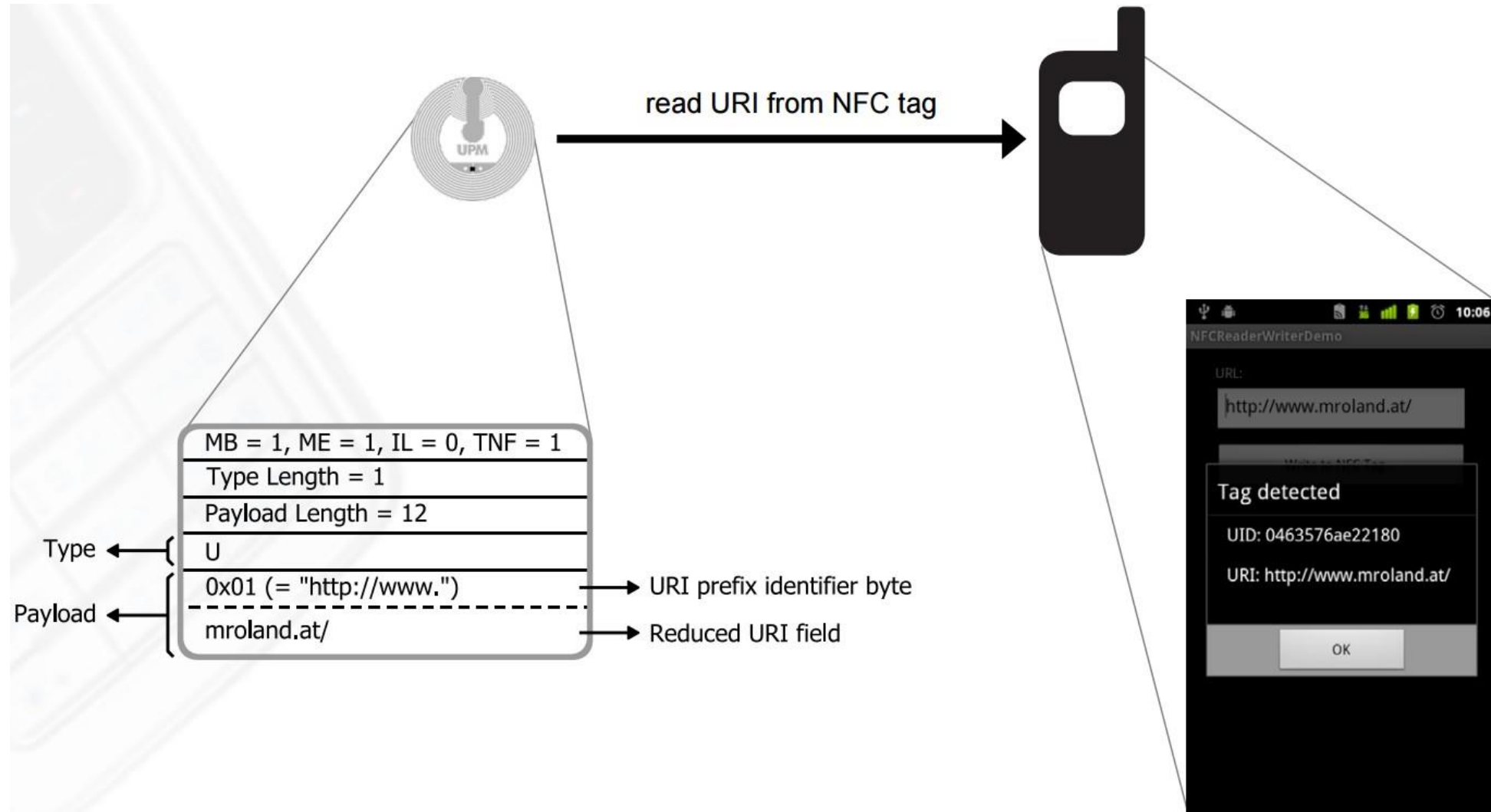
The screenshot displays an IDE with the `NFCReaderWriterDemo.java` file open. The code shows the `resolveIntent` method, which handles NFC intent data. It includes comments for `[...]` and `[...]` indicating omitted code. The method creates a `Bundle` object, adds the tag information as a string, and calls `showDialog` with the bundle and a new dialog tag.

```
private void resolveIntent(Intent data, boolean foregroundDispatch) {  
    [...]  
    if (ndefMsgs != null) {  
        [...]  
    }  
  
    Bundle args = new Bundle();  
    args.putString(ARG_MESSAGE, tagInfo.toString());  
    showDialog(DIALOG_NEW_TAG, args);  
}
```

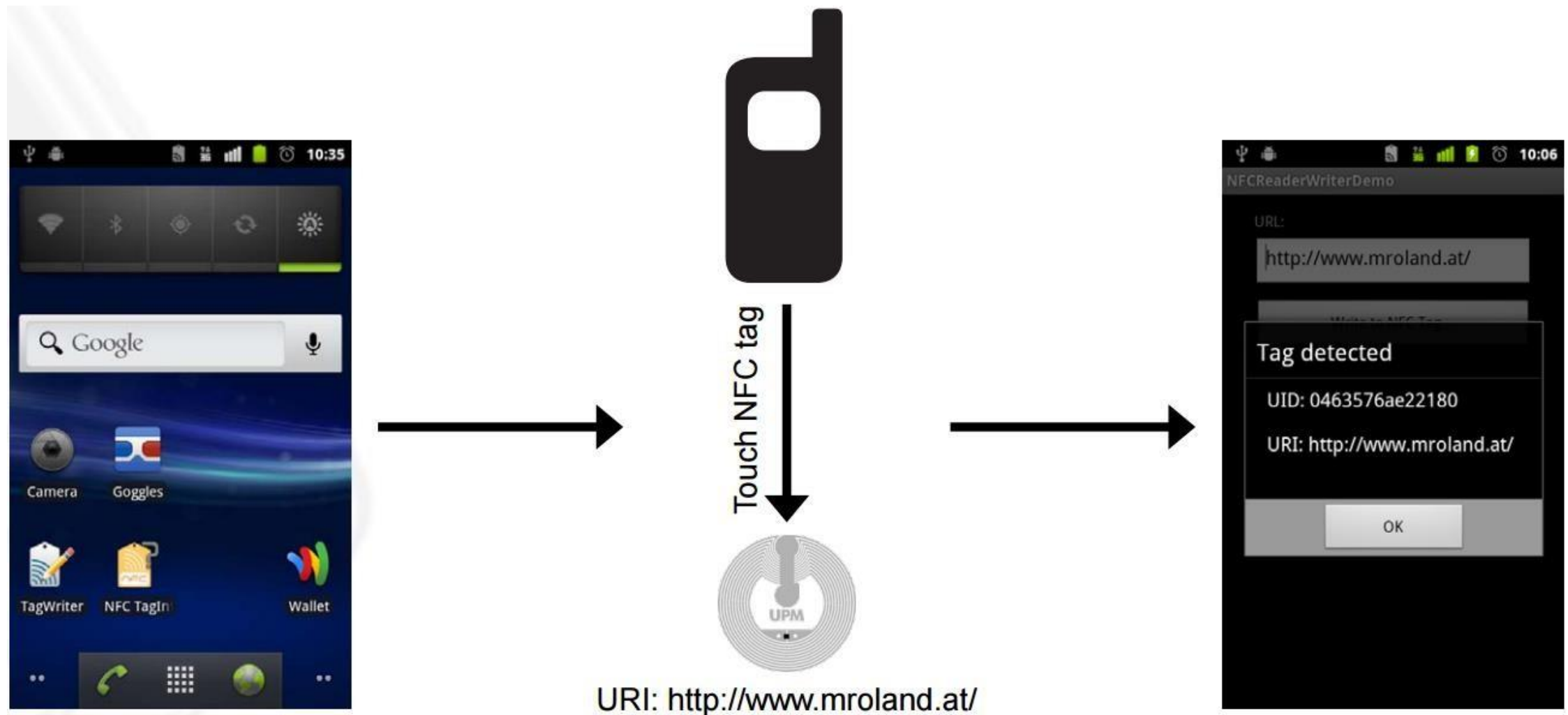
The project structure on the right shows the following hierarchy:

- NFCReaderWriterDemo
 - Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java (highlighted with a red arrow)
 - at.mroland.utils
 - StringUtils.java
 - Generated Source Packages
 - Resources
 - layout
 - main.xml
 - values
 - strings.xml
 - Libraries
 - Android 2.3.3
 - Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Lo que se tiene hasta el momento: [Aplicación NDEF Reader](#)



Ejemplo (Parte 3): Aplicación de Auto-inicio para la URI

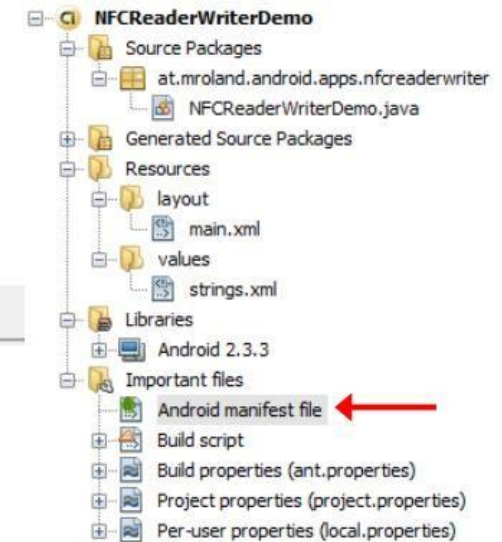


Paso 1: AndroidManifest y Auto-inicio en la etiqueta NFC

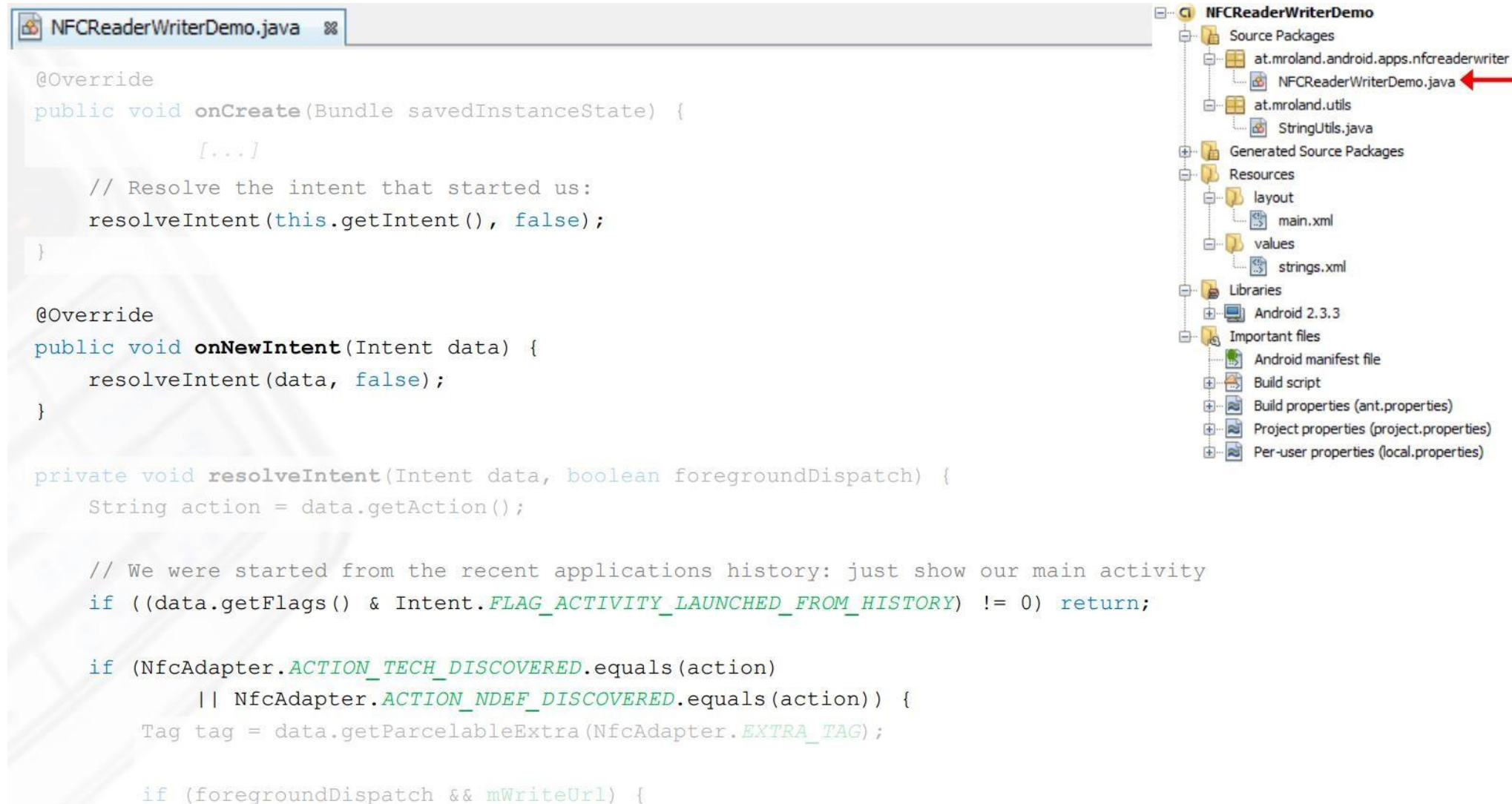
- Launch Mode: *singleTask*
- Handling configuration changes
- Intent filter: NDEF_DISCOVERED

AndroidManifest.xml

```
<activity android:name="NFCReaderWriterDemo"
    android:label="@string/app_name"
    android:launchMode="singleTask"
    android:configChanges="orientation|keyboardHidden" >
    <!-- Make our app startable through the app launcher: -->
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <!-- Make our app startable through our URL: -->
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http"
            android:host="www.mroland.at"
            android:pathPrefix="" />
    </intent-filter>
</activity>
```



Paso 2: Manejo del intento NDEF_DISCOVERED



The screenshot displays the Android Studio interface. On the left, the `NFCReaderWriterDemo.java` file is open, showing the following code:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    [...]

    // Resolve the intent that started us:
    resolveIntent(this getIntent(), false);
}

@Override
public void onNewIntent(Intent data) {
    resolveIntent(data, false);
}

private void resolveIntent(Intent data, boolean foregroundDispatch) {
    String action = data.getAction();

    // We were started from the recent applications history: just show our main activity
    if ((data.getFlags() & Intent.FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY) != 0) return;

    if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)
        || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
        Tag tag = data.getParcelableExtra(NfcAdapter.EXTRA_TAG);

        if (foregroundDispatch && mWriteUrl) {
```

On the right, the Project Explorer shows the project structure for `NFCReaderWriterDemo`. A red arrow points to the `NFCReaderWriterDemo.java` file within the `at.mroland.android.apps.nfcreaderwriter` package.

- Source Packages
 - at.mroland.android.apps.nfcreaderwriter
 - NFCReaderWriterDemo.java (highlighted with a red arrow)
 - at.mroland.utils
 - StringUtils.java
- Generated Source Packages
- Resources
 - layout
 - main.xml
 - values
 - strings.xml
- Libraries
 - Android 2.3.3
- Important files
 - Android manifest file
 - Build script
 - Build properties (ant.properties)
 - Project properties (project.properties)
 - Per-user properties (local.properties)

Las mejores aplicaciones NFC Android



NFC Task Launcher
Tagstand - August 27, 2013
Tools

Installed

This app is compatible with some of your devices.

★★★★★ (4,289)



NFC TagWriter by NXP
NXP Semiconductors - December 21, 2012
Productivity

Install **Add to Wishlist**

This app is compatible with all of your devices.

★★★★★ (473)



NFC Writer by Tagstand
Tagstand - June 28, 2013
Tools

Install **Add to Wishlist**

This app is compatible with all of your devices.

★★★★★ (128)



NFC Smart Tags
Manav Singhal - August 2, 2012
Productivity

Install **Add to Wishlist**

This app is compatible with all of your devices.

★★★★★ (13)




NFC Actions
Flomio - January 11, 2013
Productivity

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (97)



NFC Tag Writer & Reader
Connectthings - May 22, 2013
Tools

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (68)




NFC Launcher
Timree Lab - August 14, 2013
Tools

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (33)



Samsung TecTile US,Canada only
Samsung Telecommunication America - May 15, 2013
Personalization

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (620)



NFC Tag Manager
TapWise - April 26, 2013
Tools

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (3)



Xperia™ SmartTags
Sony Mobile Communications - June 13, 2012
Lifestyle

Install **Add to Wishlist**

This app is compatible with some of your devices.

★★★★★ (2,611)

Ejercicio: Con la aplicación NFCReaderWriterDemo

- Ejecutar el proyecto completo NFCReaderWriterDemo.
- Generar un reporte con todos los cambios realizados.
- Incluir las imágenes utilizadas durante el desarrollo.
- Enviar el reporte al sitio indicado por el profesor.