NFC-connected Phone as a User Interface? There's an App For That! – Hands On

Class 4: Adding NFC Capability and Communications to Our App

September 26, 2019

Charles J. Lord, PE
President, Consultant, Trainer
Blue Ridge Advanced Design and Automation

Presented by:

DesignNews

Blue Ridge Advanced Design and Automation
Asheville, North Carolina

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

This Week's Agenda

9/23 Introduction to the Project and Development Environment

9/24 An NFC Primer and Introducing the NXP NTAG

9/25 Building an Android Application from Scratch

9/26 Adding NFC Capability and Communications to Our App

9/27 Putting it All Together

This Week's Agenda

9/23 Introduction to the Project and Development Environment

9/24 An NFC Primer and Introducing the NXP NTAG

9/25 Building an Android Application from Scratch

9/26 Adding NFC Capability and Communications to Our App

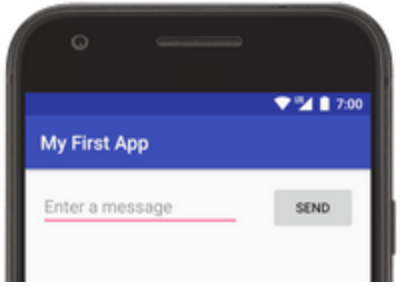
9/27 Putting it All Together

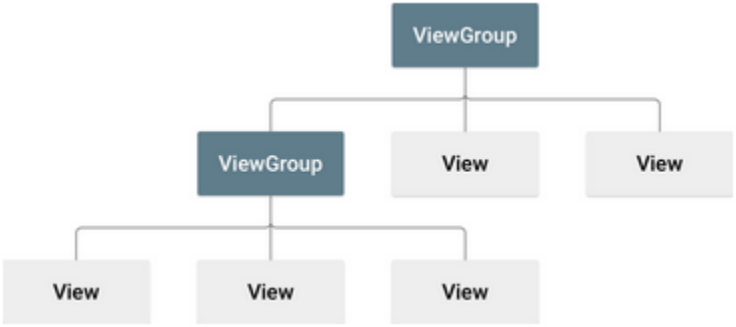
From Yesterday

- We will quickly review the homework from last night and the concepts included

- All code from yesterday and today is posted on my company github at https://github.com/bradatraining/CEC_NFC

Review of Homework



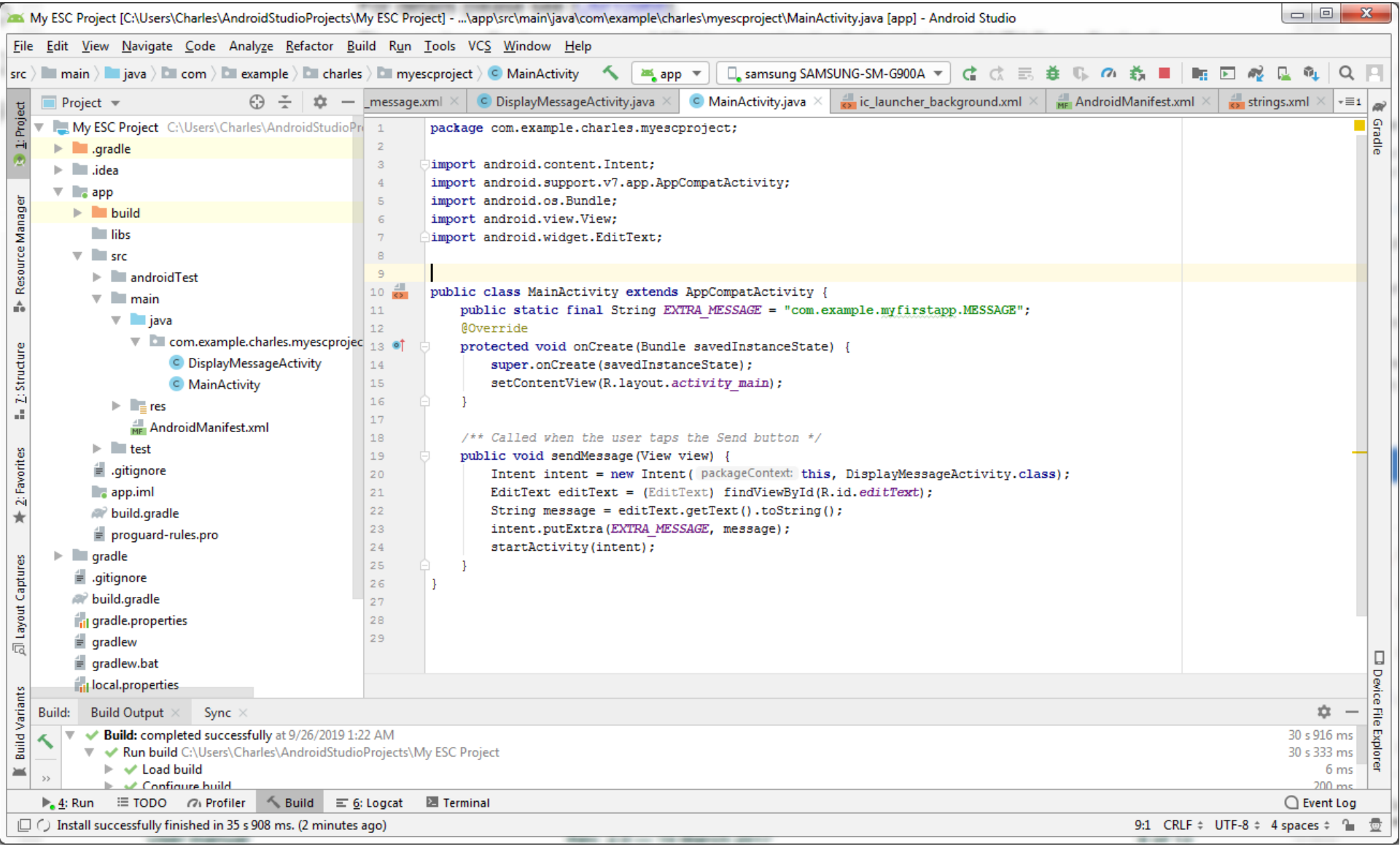


- We learned to build a hierarchy of viewgroups
- Using the layout editor, We built the XML tables to define these views without writing a line of code
- We also learned about string resources

In Part 2 (“Start another activity):

- We learned about ‘intents’ – ways of sharing data between activities
- We built a new activity for the “SEND” button and a new view for the resultant screen.
- Now let’s look at my code!

Question 1: Is an Intent a class or an object?


My ESC Project [C:\Users\Charles\AndroidStudioProjects\My ESC Project] - ...\app\src\main\java\com\example\charles\myescproject\MainActivity.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
src main java com example charles myescproject MainActivity app samsung SAMSUNG-SM-G900A
Project
My ESC Project C:\Users\Charles\AndroidStudioPr
.gradle
.idea
app
build
libs
src
androidTest
main
java
com.example.charles.myescprojec
DisplayMessageActivity
MainActivity
res
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
_message.xml
DisplayMessageActivity.java
MainActivity.java
ic_launcher_background.xml
AndroidManifest.xml
strings.xml
package com.example.charles.myescproject;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
/** Called when the user taps the Send button */
public void sendMessage(View view) {
Intent intent = new Intent(packageContext: this, DisplayMessageActivity.class);
EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
startActivity(intent);
}
}
Build: Build Output Sync
Build: completed successfully at 9/26/2019 1:22 AM 30 s 916 ms
Run build C:\Users\Charles\AndroidStudioProjects\My ESC Project 30 s 333 ms
Load build 6 ms
4: Run TODO Profiler Build 6: Logcat Terminal Event Log
Install successfully finished in 35 s 908 ms. (2 minutes ago)
9:1 CRLF UTF-8 4 spaces

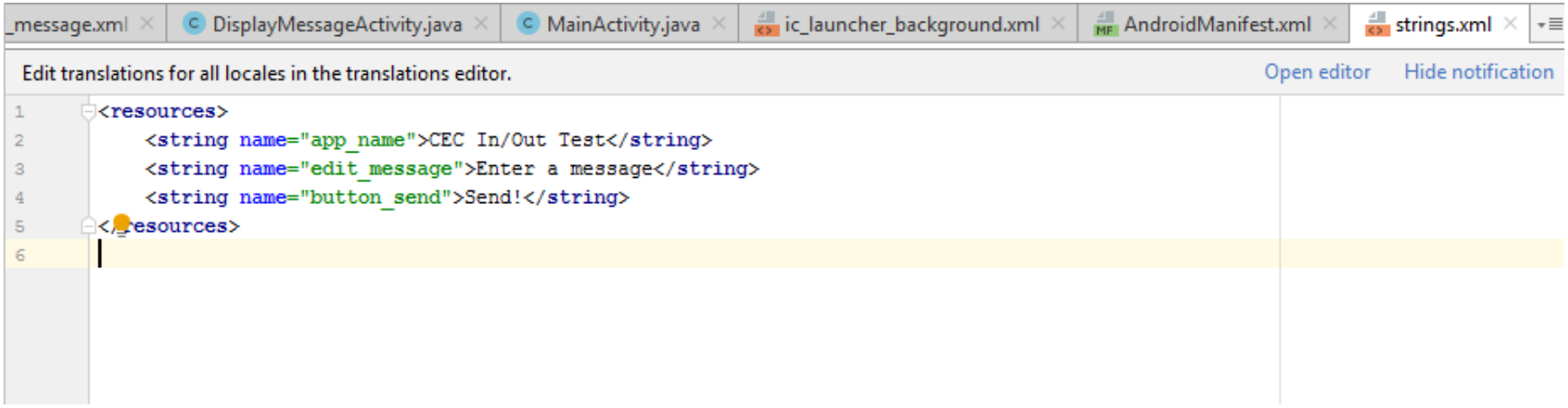
.xml | DisplayMessageActivity.java | MainActivity.java | ic_launcher_background.xml | AndroidMa

```java
package com.example.charles.myescproject;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;


public class MainActivity extends AppCompatActivity {
    public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent( packageContext: this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.editText);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

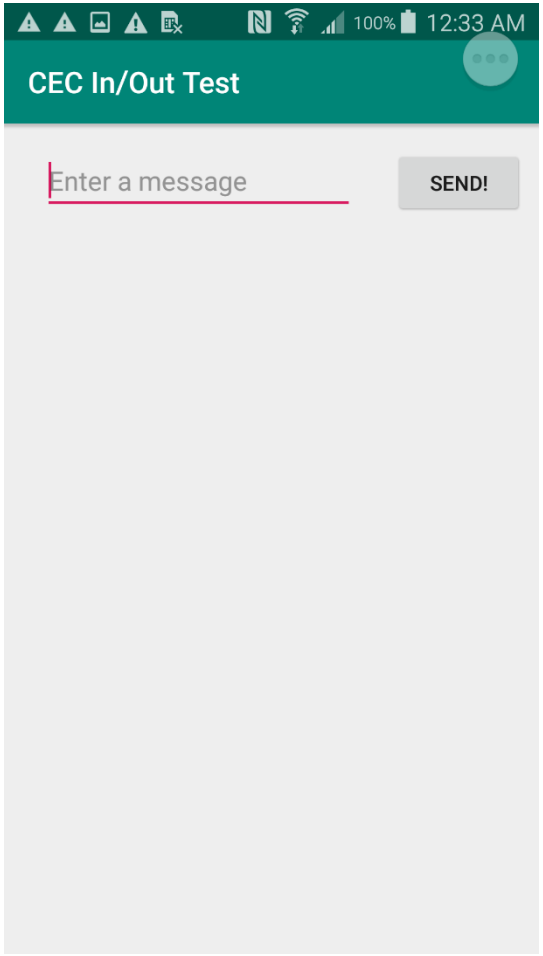
.xml | DisplayMessageActivity.java | MainActivity.java | ic_launcher_background.xml | And

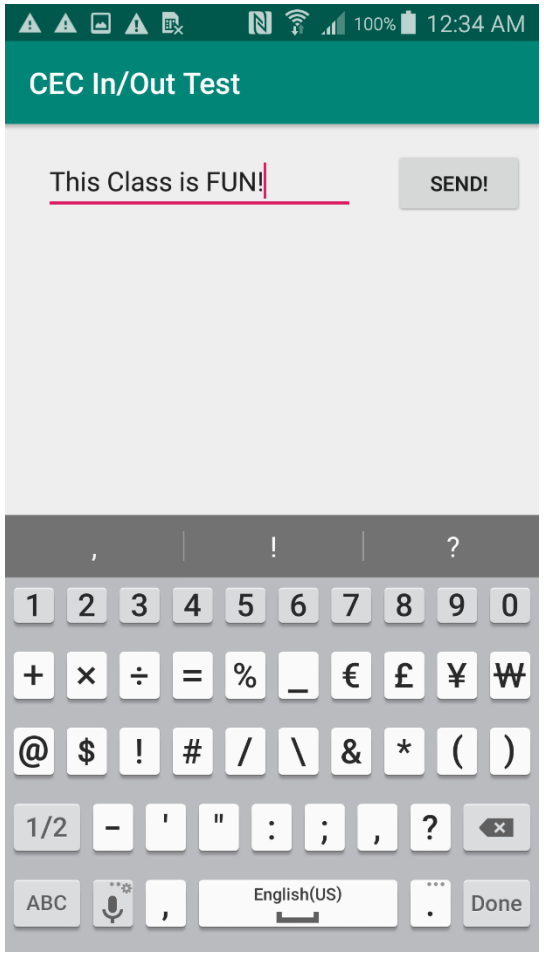
```
package com.example.charles.myescproject;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class DisplayMessageActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);

        // Get the Intent that started this activity and extract the string
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

        // Capture the layout's TextView and set the string as its text
        TextView textView = findViewById(R.id.textView);
        textView.setText(message);
    }
}
```

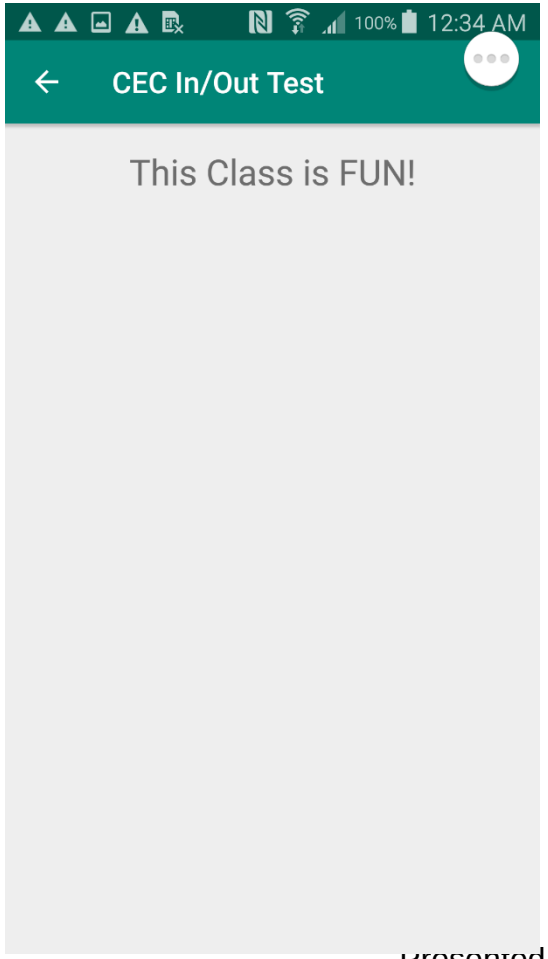
Note our strings



Our Screens


12:33 AM
CEC In/Out Test
Enter a message
SEND!


12:34 AM
CEC In/Out Test
This Class is FUN!
SEND!
English(US)
Done


12:34 AM
CEC In/Out Test
This Class is FUN!

OK, We know a little Android…

- We can define an edit box for entering data
- We can put that text in an instance for another activity
- We can switch to another activity and parse the instance for the desired text and display it

- So How do we send and receive with NFC?

Android NFC Basics 101

- We will need to include the android.nfc package
- We should check that the device has NFC capability
- We should define the NDEF messages that we will be reading and writing
  - MIME (includes text) **
  - URL
  - BT

Question 2: What does MIME stand for? Example?

Basic NFC Classes

NdefMessage	Represents an immutable NDEF Message.
NdefRecord	Represents an immutable NDEF Record.
NfcAdapter	Represents the local NFC adapter.
NfcEvent	Wraps information associated with any NFC event.
NfcManager	High level manager used to obtain an instance of an `NfcAdapter`.
Tag	Represents an NFC tag that has been discovered.

https://developer.android.com/reference/android/nfc

```
public static interface NfcAdapter.ReaderCallback
```

android.nfc.NfcAdapter.ReaderCallback

A callback to be invoked when the system finds a tag while the foreground activity is operating in reader mode.

Register your `ReaderCallback` implementation with `NfcAdapter#enableReaderMode` and disable it with `NfcAdapter#disableReaderMode`.

See also:

`NfcAdapter.enableReaderMode(Activity, NfcAdapter.ReaderCallback, int, Bundle)`

Summary

Public methods	
`abstract void`	`onTagDiscovered(Tag tag)`

Public methods

onTagDiscovered

Added in API level 19

```
public abstract void onTagDiscovered (Tag tag)
```

Parameters	
`tag`	`Tag`

```
public static interface NfcAdapter.OnTagRemovedListener
```

android.nfc.NfcAdapter.OnTagRemovedListener

A callback that is invoked when a tag is removed from the field.

See also:

`NfcAdapter.ignore(Tag, int, NfcAdapter.OnTagRemovedListener, Handler)`

Summary

Public methods	
`abstract void`	`onTagRemoved()`

Public methods

onTagRemoved

Added in API level 24

```
public abstract void onTagRemoved ()
```

Some Useful Methods

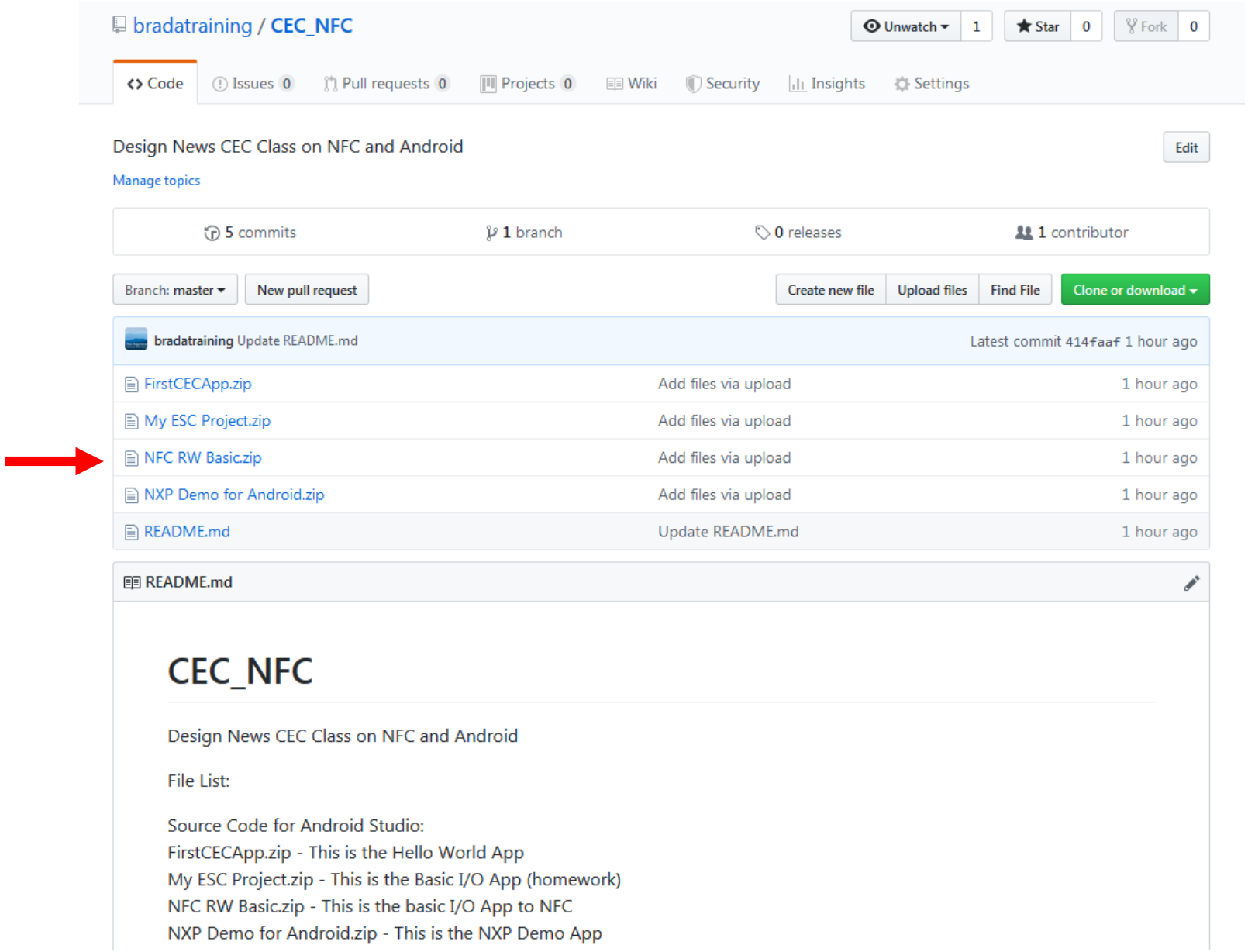
boolean	isSecureNfcEnabled() Checks Secure NFC feature is enabled.
boolean	isSecureNfcSupported() Checks if the device supports Secure NFC functionality.
boolean	isEnabled() Return true if this NFC Adapter has any features enabled.
void	enableReaderMode(Activity activity, NfcAdapter.ReaderCallback callback, int flags, Bundle extras) Limit the NFC controller to reader mode while this Activity is in the foreground.
static NfcAdapter	getDefaultAdapter(Context context) Helper to get the default NFC Adapter.
boolean	ignore(Tag tag, int debounceMs, NfcAdapter.OnTagRemovedListener tagRemovedListener, Handler handler) Signals that you are no longer interested in communicating with an NFC tag for as long as it remains in range.

Constants

String	ACTION_ADAPTER_STATE_CHANGED Broadcast Action: The state of the local NFC adapter has been changed.
String	ACTION_NDEF_DISCOVERED Intent to start an activity when a tag with NDEF payload is discovered.
String	ACTION_TAG_DISCOVERED Intent to start an activity when a tag is discovered.
String	ACTION_TECH_DISCOVERED Intent to start an activity when a tag is discovered and activities are registered for the specific technologies on the tag.
String	ACTION_TRANSACTION_DETECTED Broadcast Action: Intent to notify an application that an transaction event has occurred on the Secure Element.

So how do we build this?

- Will use an example with very similar function to our homework example but with NDEF string read / write capabilities


bradatraining / CEC_NFC
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
Design News CEC Class on NFC and Android
Edit
Manage topics
5 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
bradatraining Update README.md
Latest commit 414faaf 1 hour ago
FirstCECApp.zip Add files via upload 1 hour ago
My ESC Project.zip Add files via upload 1 hour ago
NFC RW Basic.zip Add files via upload 1 hour ago
NXP Demo for Android.zip Add files via upload 1 hour ago
README.md Update README.md 1 hour ago
README.md
CEC_NFC
Design News CEC Class on NFC and Android
File List:
Source Code for Android Studio:
FirstCECApp.zip - This is the Hello World App
My ESC Project.zip - This is the Basic I/O App (homework)
NFC RW Basic.zip - This is the basic I/O App to NFC
NXP Demo for Android.zip - This is the NXP Demo App

android_nfc_read_write-master | app | MainActivity.java

```java
package com.example.peng.nfcreadwrite;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.nfc.FormatException;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.os.Bundle;
import android.os.Parcelable;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.UnsupportedEncodingException;

public class MainActivity extends Activity {

    public static final String ERROR_DETECTED = "No NFC tag detected!";
    public static final String WRITE_SUCCESS = "Text written to the NFC tag successfully!";
    public static final String WRITE_ERROR = "Error during writing, is the NFC tag close enough to your device?";
    NfcAdapter nfcAdapter;
    PendingIntent pendingIntent;
    IntentFilter writeTagFilters[];
    boolean writeMode;
    Tag myTag;
    Context context;

    TextView tvNFCContent;
    TextView message;
    Button btnWrite;
```

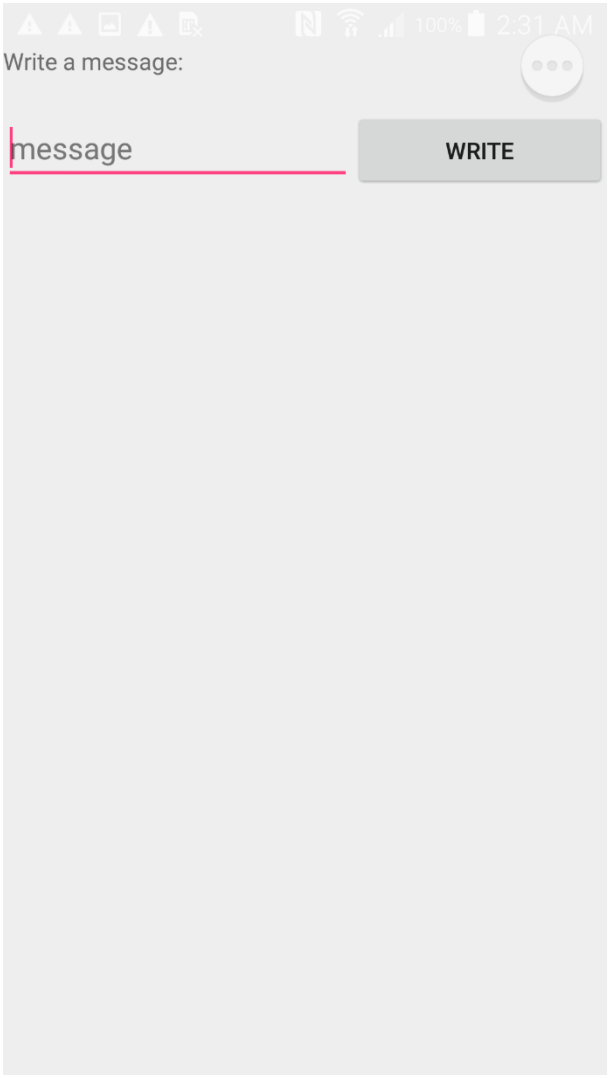
```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    context = this;

    tvNFCContent = (TextView) findViewById(R.id.nfc_contents);
    message = (TextView) findViewById(R.id.edit_message);
    btnWrite = (Button) findViewById(R.id.button);

    btnWrite.setOnClickListener((v) -> {
            try {
                if (myTag == null) {
                    Toast.makeText(context, ERROR_DETECTED, Toast.LENGTH_LONG).show();
                } else {
                    write(message.getText().toString(), myTag);
                    Toast.makeText(context, WRITE_SUCCESS, Toast.LENGTH_LONG ).show();
                }
            } catch (IOException e) {
                Toast.makeText(context, WRITE_ERROR, Toast.LENGTH_LONG ).show();
                e.printStackTrace();
            } catch (FormatException e) {
                Toast.makeText(context, WRITE_ERROR, Toast.LENGTH_LONG ).show();
                e.printStackTrace();
            }
    });
```

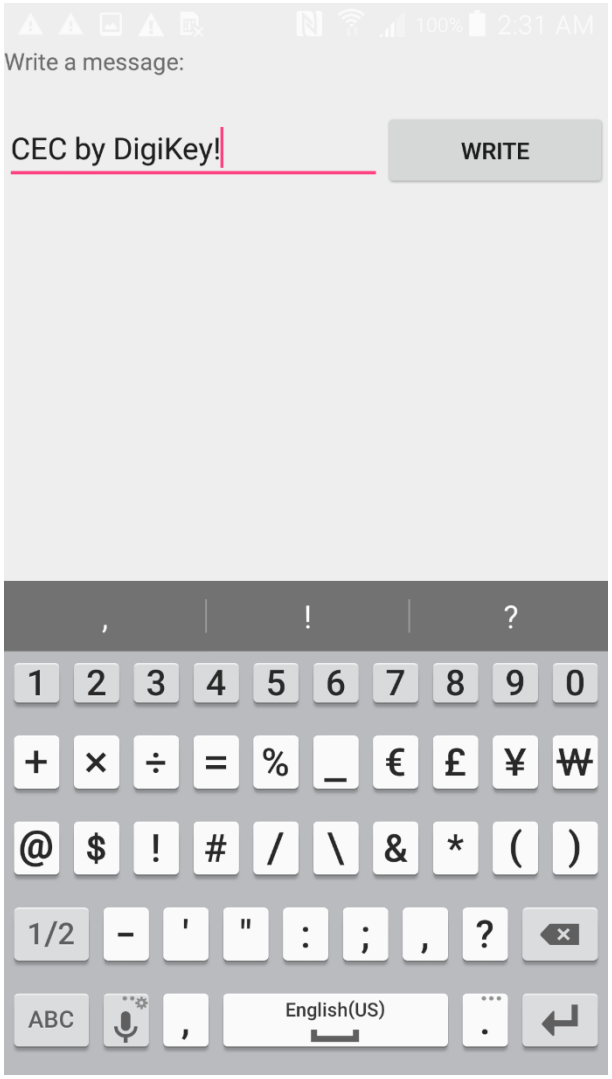
```
nfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (nfcAdapter == null) {
    // Stop here, we definitely need NFC
    Toast.makeText( context: this,  text: "This device doesn't support NFC.", Toast.LENGTH_LONG).show();
    finish();
}
readFromIntent(getIntent());

pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, new Intent( packageContext: this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP),  flags: 0);
IntentFilter tagDetected = new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
tagDetected.addCategory(Intent.CATEGORY_DEFAULT);
writeTagFilters = new IntentFilter[] { tagDetected };
```

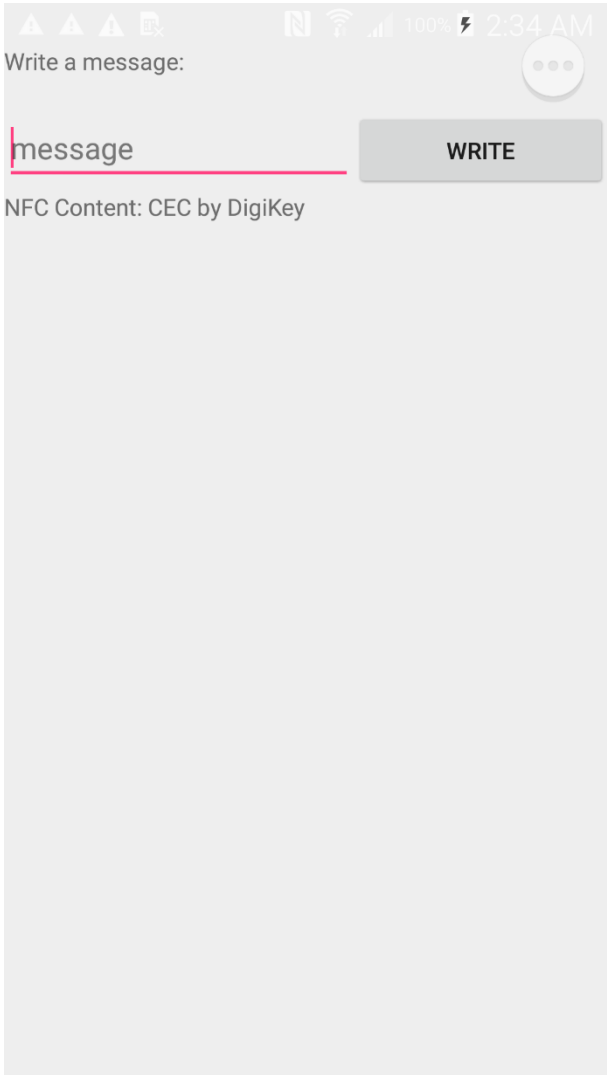
```
@Override
protected void onNewIntent(Intent intent) {
    setIntent(intent);
    readFromIntent(intent);
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())){
        myTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
    }
}

@Override
public void onPause(){
    super.onPause();
    WriteModeOff();
}

@Override
public void onResume(){
    super.onResume();
    WriteModeOn();
}
```

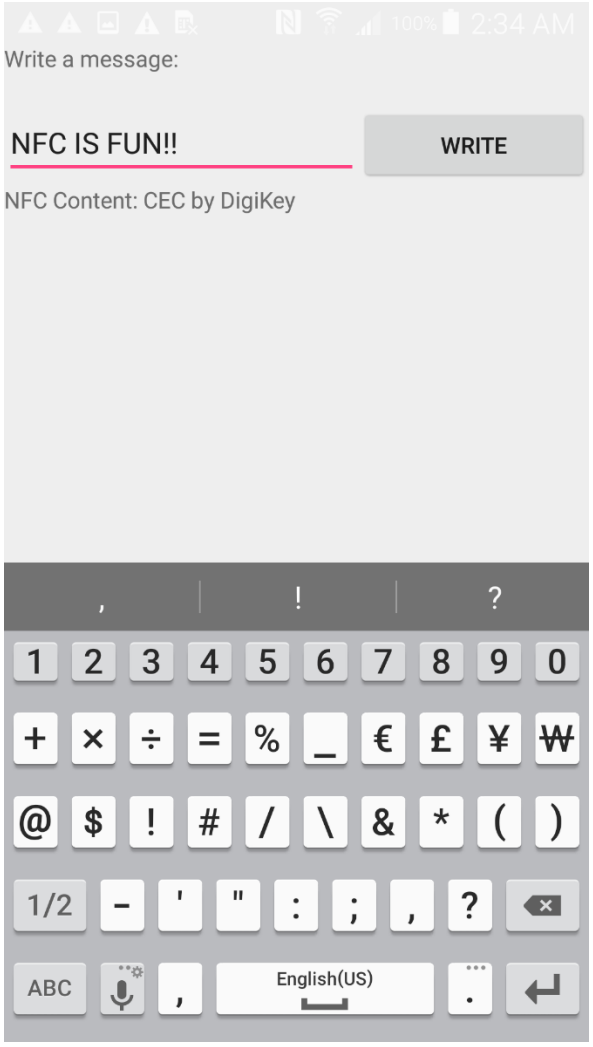
```
/*****************************************************************************
 ********************************Read From NFC Tag*****************************
 *****************************************************************************/
private void readFromIntent(Intent intent) {
    String action = intent.getAction();
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)
            || NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)
            || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
        Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
        NdefMessage[] msgs = null;
        if (rawMsgs != null) {
            msgs = new NdefMessage[rawMsgs.length];
            for (int i = 0; i < rawMsgs.length; i++) {
                msgs[i] = (NdefMessage) rawMsgs[i];
            }
        }
        buildTagViews(msgs);
    }
}
private void buildTagViews(NdefMessage[] msgs) {
    if (msgs == null || msgs.length == 0) return;

    String text = "";
      String tagId = new String(msgs[0].getRecords()[0].getType());
    byte[] payload = msgs[0].getRecords()[0].getPayload();
    String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16"; // Get the Text Encoding
    int languageCodeLength = payload[0] & 0063; // Get the Language Code, e.g. "en"
    // String languageCode = new String(payload, 1, languageCodeLength, "US-ASCII");

    try {
        // Get the Text
        text = new String(payload,  offset: languageCodeLength + 1,  byteCount: payload.length - languageCodeLength - 1, textEncoding);
    } catch (UnsupportedEncodingException e) {
        Log.e( tag: "UnsupportedEncoding", e.toString());
    }

    tvNFCContent.setText("NFC Content: " + text);
}
```

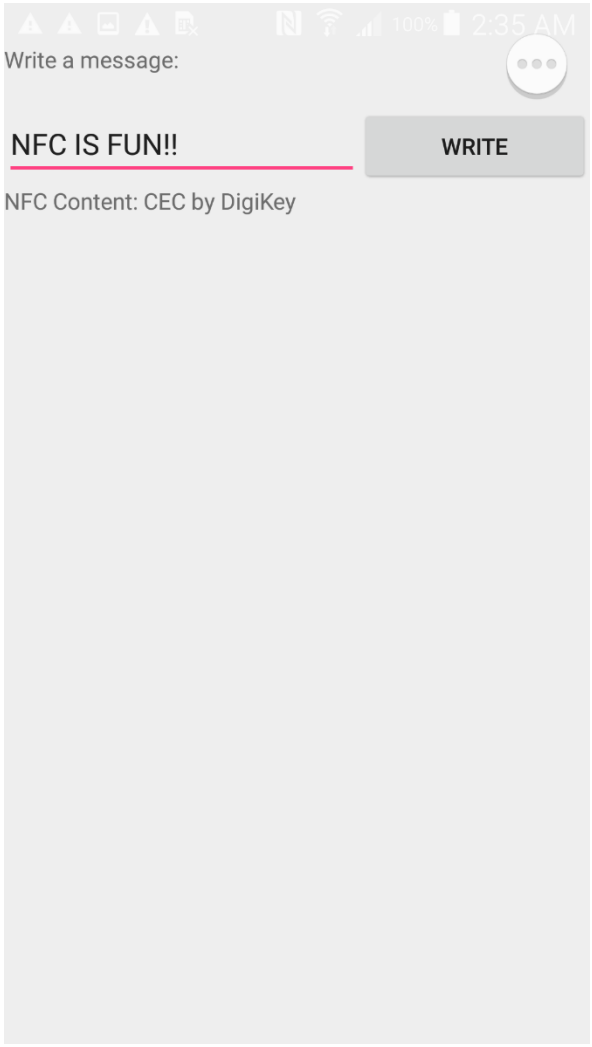
```
/*************************************************************************
 *********************************Write to NFC Tag*************************
 ************************************************************************/
private void write(String text, Tag tag) throws IOException, FormatException {
    NdefRecord[] records = { createRecord(text) };
    NdefMessage message = new NdefMessage(records);
    // Get an instance of Ndef for the tag.
    Ndef ndef = Ndef.get(tag);
    // Enable I/O
    ndef.connect();
    // Write the message
    ndef.writeNdefMessage(message);
    // Close the connection
    ndef.close();
}
private NdefRecord createRecord(String text) throws UnsupportedEncodingException {
    String lang       = "en";
    byte[] textBytes  = text.getBytes();
    byte[] langBytes  = lang.getBytes( charsetName: "US-ASCII");
    int    langLength = langBytes.length;
    int    textLength = textBytes.length;
    byte[] payload    = new byte[1 + langLength + textLength];

    // set status byte (see NDEF spec for actual bits)
    payload[0] = (byte) langLength;

    // copy langbytes and textbytes into payload
    System.arraycopy(langBytes,  i: 0, payload,  i1: 1,              langLength);
    System.arraycopy(textBytes,  i: 0, payload,  i1: 1 + langLength, textLength);

    NdefRecord recordNFC = new NdefRecord(NdefRecord.TNF_WELL_KNOWN,  NdefRecord.RTD_TEXT,  new byte[0], payload);

    return recordNFC;
}
```

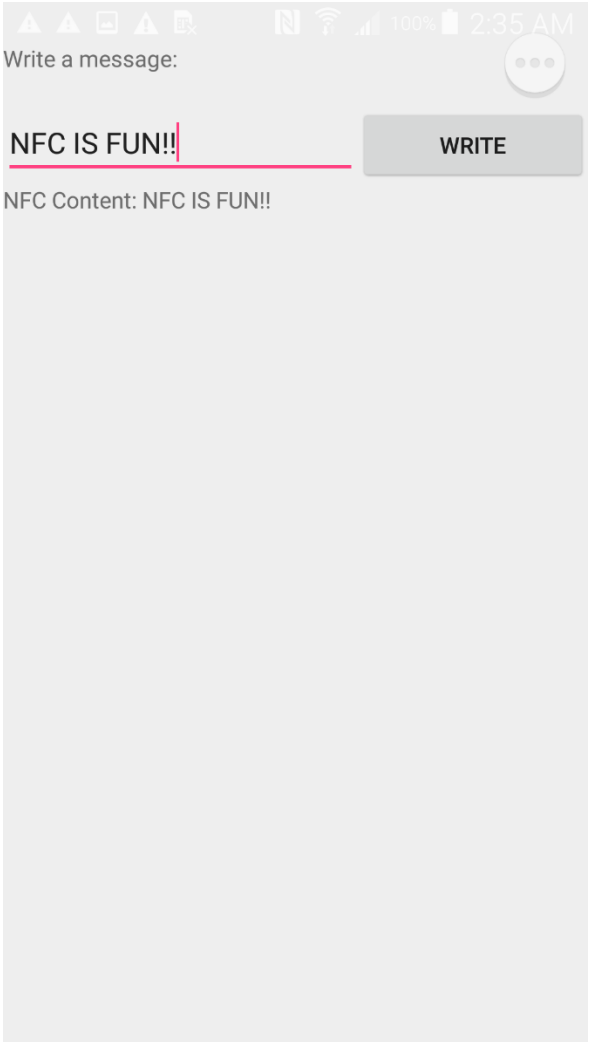

2:31 AM
Write a message:
message
WRITE


2:31 AM
Write a message:
CEC by DigiKey!
WRITE
,
!
?
1 2 3 4 5 6 7 8 9 0
+ × ÷ = % _ € £ ¥ ₩
@ $! # / \ & * ( )
1/2 - ' " : ; , ?
ABC
English(US)


2:34 AM
Write a message:
message
WRITE
NFC Content: CEC by DigiKey


2:34 AM
Write a message:
NFC IS FUN!!
WRITE
NFC Content: CEC by DigiKey
English(US)


2:35 AM
Write a message:
NFC IS FUN!!
WRITE
NFC Content: CEC by DigiKey


2:35 AM
Write a message:
NFC IS FUN!!
WRITE
NFC Content: NFC IS FUN!!

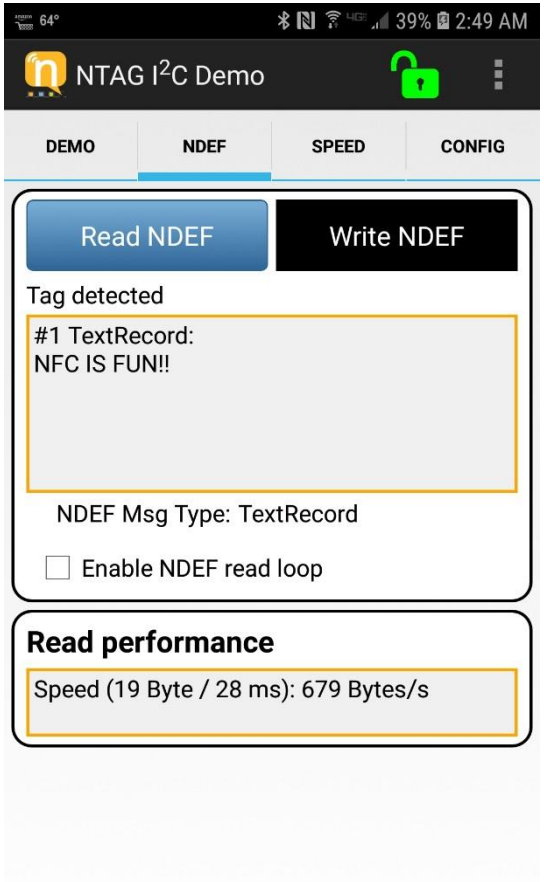
And it is indeed written to the first NDEF record!

Question 3: What does the “2.6V” on the display tell us?

We will look at the NXP App

For Homework (again)

- Load the source for the NXP NTAG Demo (from the Github)
- Look at the functions for the NDEF R/W tab
- How does it compare to the previous app?
- Also look for the security code – we will use that tomorrow!



This Week's Agenda

9/23 Introduction to the Project and Development Environment

9/24 An NFC Primer and Introducing the NXP NTAG

9/25 Building an Android Application from Scratch

9/26 Adding NFC Capability and Communications to Our App

9/27 Putting it All Together

Please stick around as I answer your questions!

- Please give me a moment to scroll back through the chat window to find your questions
- I will stay on chat as long as it takes to answer!
- I am available to answer simple questions or to consult (or offer in-house training for your company)
  c.j.lord@ieee.org
  http://www.blueridgetechnc.com
  http://www.linkedin.com/in/charleslord
  Twitter: @charleslord
  https://www.github.com/bradatraining