# Reading and Writing Files

Kanda Runapongsa Saikaew
Computer Engineering Department
Khon Kaen University
http://twitter.com/krunapon

# Access Files on Android App

- Android offers two models for accessing files
  - One for files prepackaged with your application
  - One for files created on-device by your application
- Location of files prepackaged within application
  - Place the file in the res/raw directory, so it will be put in the Android application APK file as part of the packaging process as a raw resource
  - To access this file, you need to get yourself a Resources object. From an activity, that is as simple as calling getResources()
  - A Resources object offers openRawResource() to get an InputStream on the file you specify

# Accessing Files in Your App

- Rather than a path, openRawResource() expects an integer identifier for the file as packaged
  - This works just like accessing widgets via findViewById()
  - For example, if you put a file named words.xml in res/raw, the identifier is accessible in Java as R.raw.words
- Since you can get only an InputStream, you have no means of modifying this file
- Hence, it is really useful just for static reference data
- Sample code

```
InputStream in = getResources().openRawResource(R.raw.words);
```

# File layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<TextView
android:id="@+id/selection"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<ListView
android:id="@android:id/list"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:drawSelectorOnTop="false"
/>
</LinearLayout>
```

# File res/raw/words.xml

```
<?xml version='1.0'?>
    <words>
        <word value="Facebook"/>
        <word value="Gmail"/>
        <word value="Google Docs"/>
        <word value="Slideshare"/>
        <word value="Dropbox"/>
        <word value="Eclipse"/>
    </words>
```

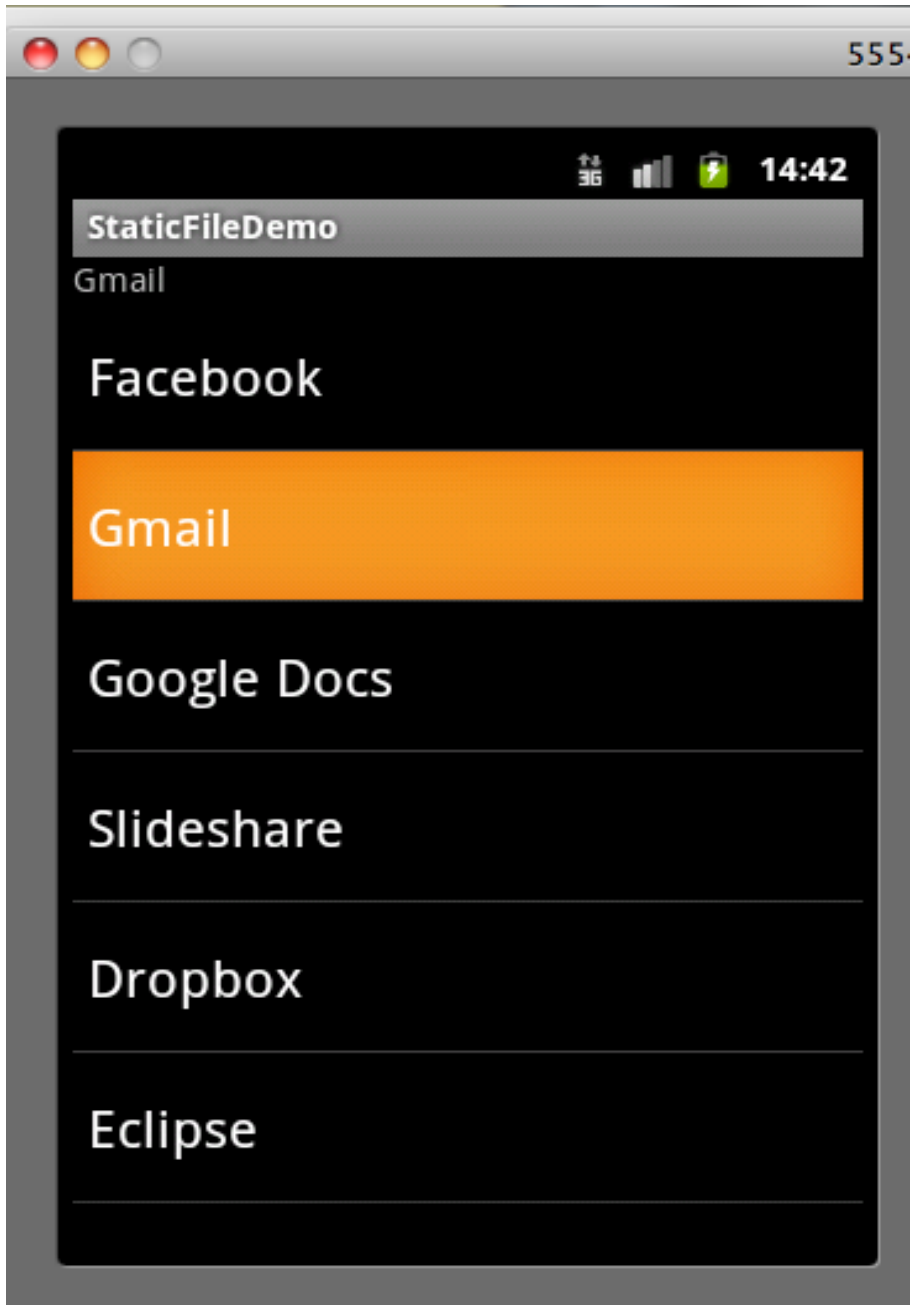
# StaticFileDemo (Java File 1/2)

```
import java.io.InputStream;
import java.util.ArrayList;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
public class StaticFileDemo extends ListActivity {
    TextView selection;
    ArrayList<String> items=new ArrayList<String>();
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        selection = (TextView)findViewById(R.id.selection);
        try {
            InputStream in = getResources().openRawResource(R.raw.words);
            DocumentBuilder builder = DocumentBuilderFactory.newInstance()
                    .newDocumentBuilder();
```

# StaticFileDemo (Java File 1/2)

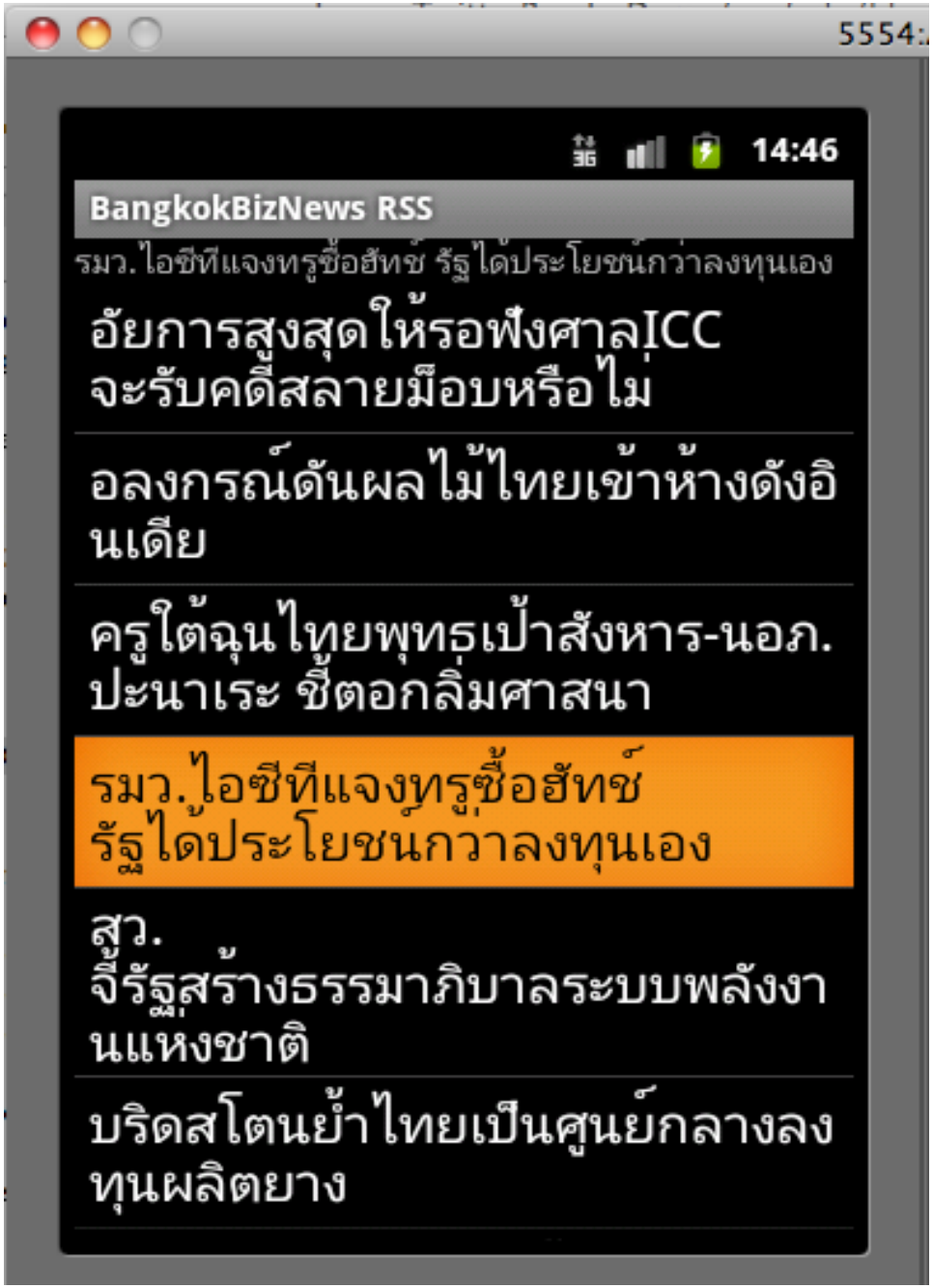
```
Document doc = builder.parse(in, null);
NodeList words=doc.getElementsByTagName("word");
for (int i = 0;i < words.getLength(); i++) {
    items.add(((Element)words.item(i)).getAttribute("value"));
}
in.close();
} catch (Throwable t) {
    Toast.makeText(this, "Exception: "+t.toString(), 2000).show();
}
setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, items));
}
public void onListItemClick(ListView parent, View v, int position,
long id) {
selection.setText(items.get(position).toString());
}
}
```

# StaticFileDemo Result




- Read content from a static file in res/raw folder of the project
- When the user clicks any item, the item content appears at the text view on the top below the program title.

# RSS Reader App




- Read content from Bangkok biz news RSS URL
- When the user clicks any item, the item content appears at the text view on the top below the program title

# Reading and Writing File

- Reading or writing your own, application-specific data files is nearly identical to what you might do in a desktop Java application
- The key is to use openFileInput() or openFileOutput() on your Activity or other Context to get an InputStream or OutputStream, respectively
- From that point forward, it is not much different from regular Java I/O logic
  - Wrap those streams as needed, such as using an InputStreamReader or OutputStreamWriter for text-based I/O.
  - Read or write the data
  - Use close() to release the stream when done

# Accessing Files from Applications

- If two applications both try reading a notes.txt file via openFileInput(), each will access its own edition of the file
- If you need to have one file accessible from many places, you probably want to create a content provider
- Note that openFileInput() and openFileOutput() do not accept file paths (e.g., path/to/file.txt), just simple filenames
  - private final static String NOTES="notes.txt";InputStream in=openFileInput(NOTES);
  - if (in!=null) {
  - InputStreamReader tmp=new InputStreamReader(in);
  - ...try {
  - OutputStreamWriter out=
  - new OutputStreamWriter(openFileOutput(NOTES, 0));

# Sample res/layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.
com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<Button android:id="@+id/close"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Close" />
<EditText
android:id="@+id/editor"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:singleLine="false"
```

# ReadWriteSample Java Code (1/3)

```
public class ReadWriteFileDemo extends Activity {
private final static String NOTES="notes.txt";
private EditText editor;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
editor=(EditText)findViewById(R.id.editor);
Button btn=(Button)findViewById(R.id.close);
btn.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v) {
finish();
}
});
}
```

# ReadWriteSample Java Code (2/3)

```
public void onPause() {
super.onPause();
try {
OutputStreamWriter out=
new OutputStreamWriter(openFileOutput(NOTES, 0));
out.write(editor.getText().toString());
out.close();
}
catch (Throwable t) {
Toast
.makeText(this, "Exception: "+t.toString(), 2000)
.show();
}
}
}
```

# ReadWriteSample Java Code (3/3)

```
public void onResume() {
super.onResume();
try {
InputStream in=openFileInput(NOTES);
if (in!=null) {
InputStreamReader tmp=new InputStreamReader(in);
BufferedReader reader=new BufferedReader(tmp);
String str;
StringBuffer buf=new StringBuffer();
while ((str = reader.readLine()) != null) {
buf.append(str+"\n"); }
in.close();
editor.setText(buf.toString());
}}
catch (java.io.FileNotFoundException e) {
// that's OK, we probably haven't created it yet }
catch (Throwable t) {
Toast.makeText(this, "Exception: "+t.toString(), 2000).show(); }}
```

# ReadWriteSample Result (1/2)


- Type a text in the text editor and then click close
- The program then writes the context in the editor to a file and then the program is closed

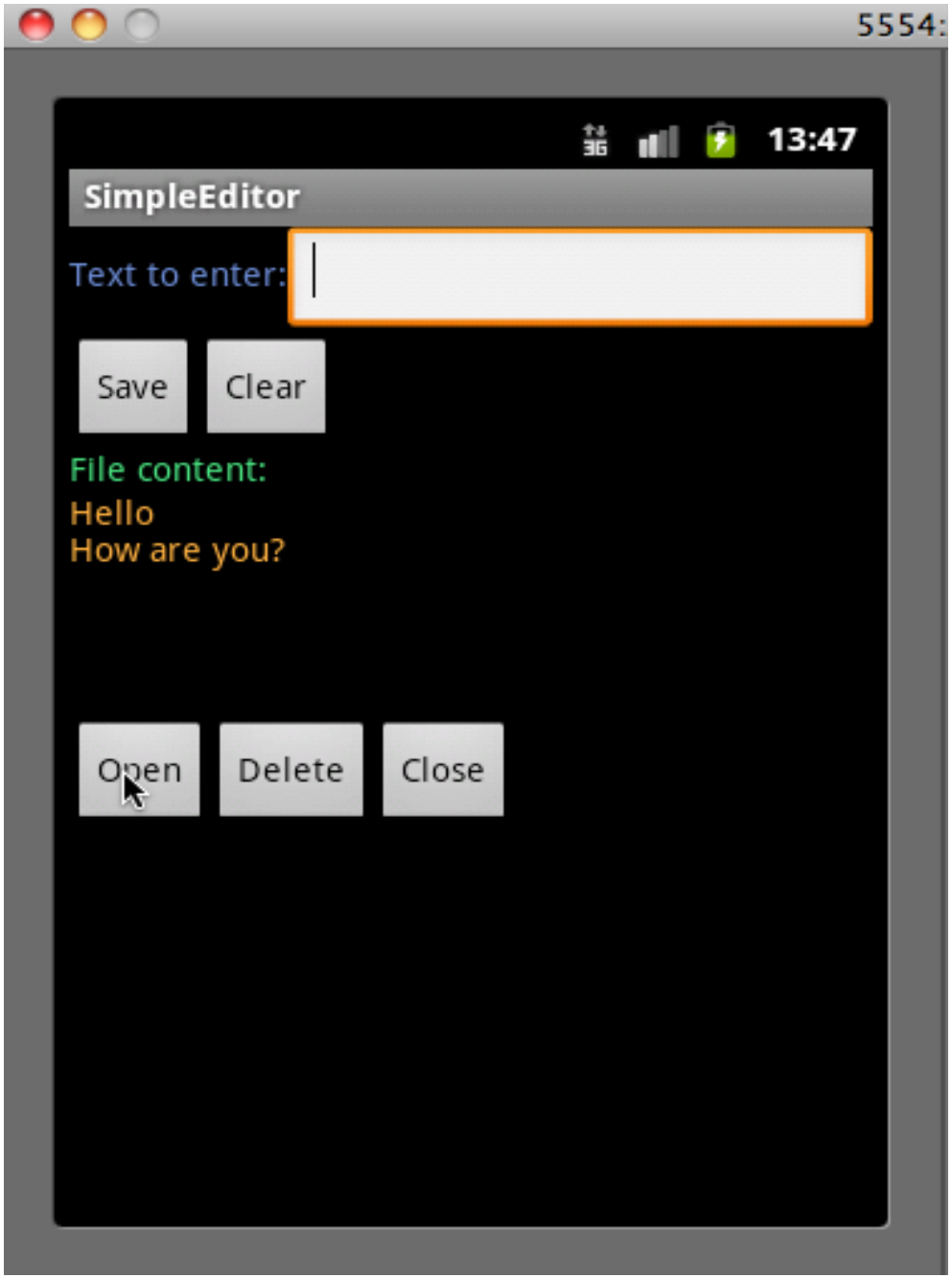
# ReadWriteSample Result (2/2)


- When the application is opened, the editor shows the content of the file

# Simple Editor (1/6)


- Type something in the text to enter filed and then click save
- That text will be saved in a file.

# Simple Editor (2/6)

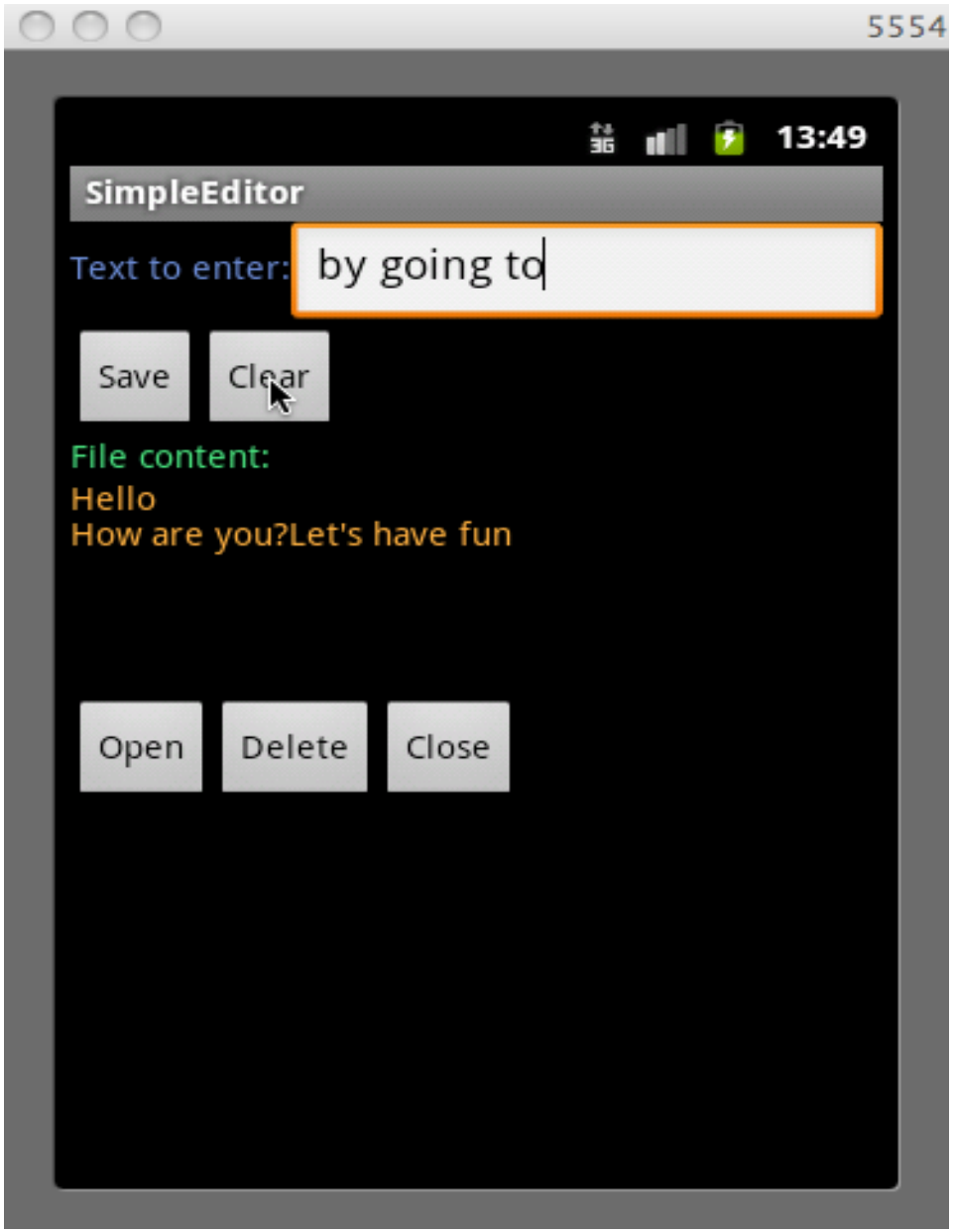



- After clicking save, the text field is clear
- Then, try to click open to see the updated content of the file

# Simple Editor (3/6)

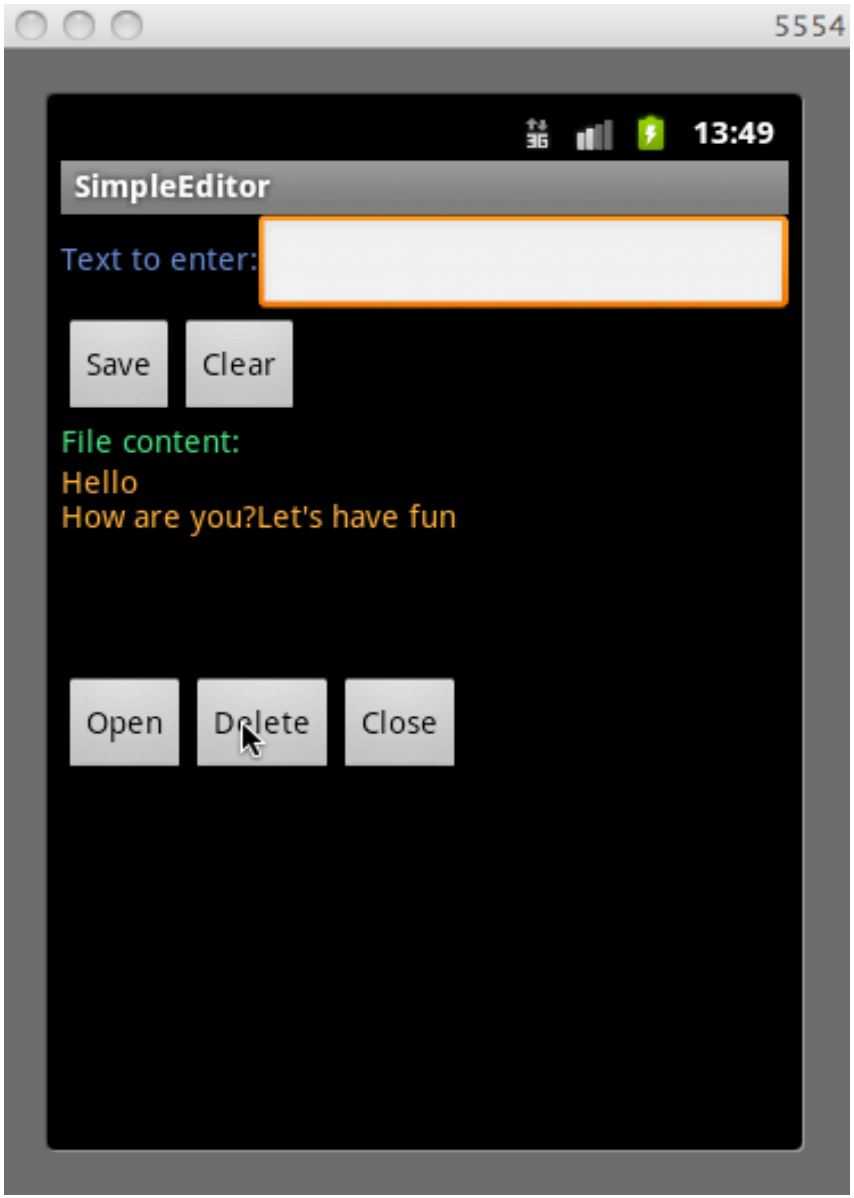

Now we see the newly entered text in the file content

# Simple Editor (4/6)




Let's type something and if we want to cancel, we can then just press clear

# Simple Editor (5/6)




- Now we can see that the text field is empty
- If we want to delete the file content, we can click button Delete

# Simple Editor (6/6)


After pressing Delete button, now file content is empty

# References


- Mark L Murphy, "Beginning Android 2", http://www.apress.com/book/downloadfile/4530