Android Programming
Lecture 7

9/23/2011

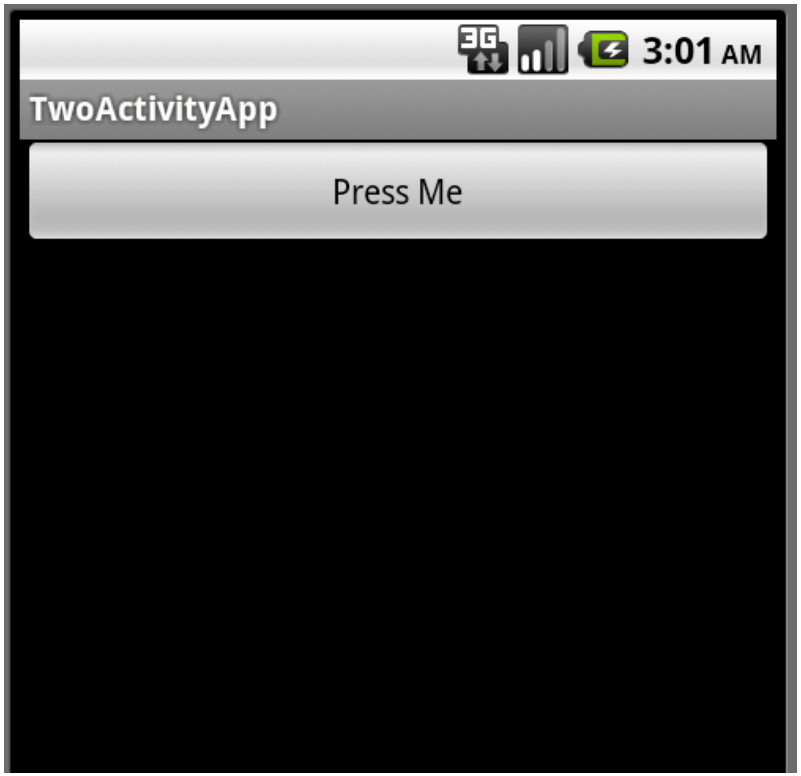
Multiple Activities

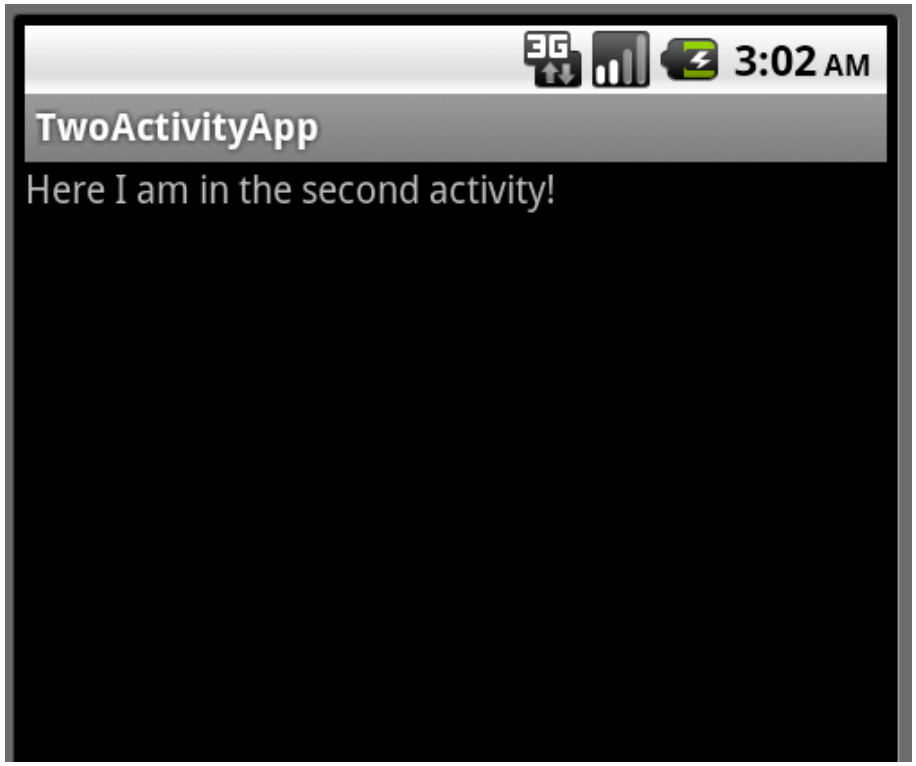
- So far, projects limited to one Activity

- Next step:
  - Intra-application communication
    - Having multiple activities within own application
  - Inter-application communication
    - Exploiting capabilities of other interactions

Multiple Activities: Code By Example

- First goal:
  - Press button in one Activity
  - Leads to opening of a second Activity





Multiple Activities

- All Activities within an application need to be specified in application AndroidManifest.xml
  - Only one is listed as the main launch Activity

First Activity (launched)

Second Activity

```
<?xml version Close encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="turkett.csc191"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FirstActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
         <activity android:name=".SecondActivity"
                  android:label="@string/app_name">
        </activity>
    </application>
</manifest>
```

Intents

Simplest intents are used to just trigger a specific known other Activity:

Have no specific description except target class

```
Intent intent = new Intent(MyActivity.this,
    MyOtherActivity.class);
startActivity(intent);
```

Intent parameters:

current context (current Activity)

target Activity

Second activity is an independent piece

(no return of information to first)

Two Activity Example: Two Layouts

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"
    >
<Button  android:id="@+id/first_activity_button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Press Me"
    />
</LinearLayout>
```

First Activity

second.xml

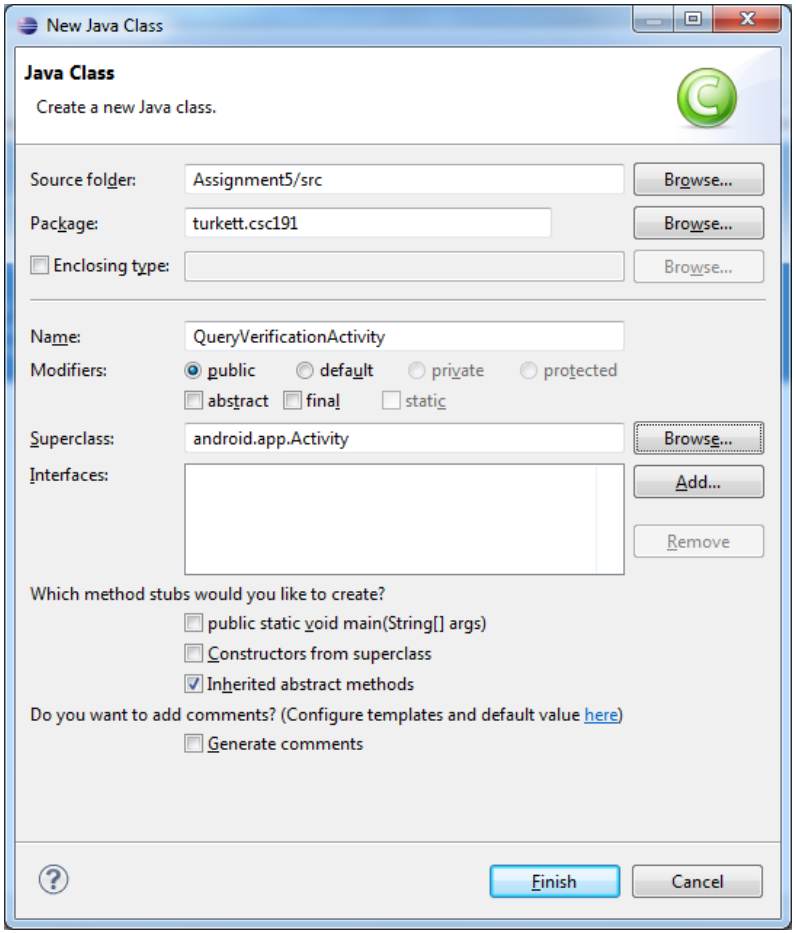
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
<TextView  android:id="@+id/second_activity_text_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Here I am in the second activity!"
    />
</LinearLayout>
```

Second Activity

Adding an Activity

- In *src/packagename*, right click and choose *New, Java Class*

- Verify package
- Choose an appropriate name
- Make superclass be *android.app.Activity*
- Select *Inherited abstract methods* if not already checked for *which method stubs to create*



Two Activity Example: Two Activities

Second Activity

SecondActivity.java

```
package turkett.csc191;

import android.app.Activity;

public class SecondActivity extends Activity {

    Button theButton;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // use the layout defined as "second" (with one big TextView)
        setContentView(R.layout.second);
    }
```

First Activity

FirstActivity.java

```
package turkett.csc191;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.view.View;
import turkett.csc191.SecondActivity;

public class FirstActivity extends Activity implements View.OnClickListener {

    Button theButton;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        theButton = (Button)findViewById(R.id.first_activity_button);
        theButton.setOnClickListener(this);
    }

    public void onClick(View arg0)
    {
        Intent intent = new Intent(FirstActivity.this, SecondActivity.class);
        startActivity(intent);
    }
}
```

Sending Parameter Values to Activities

We may want to use Activity-Activity interactions like function calls:

Passing in parameters

Receiving data back (return value)

So far, we have the equivalent of a function with a void return value and no parameters

Sending Parameter Values to Activities

Passing data to 2nd activity (parameters):

An intent can be associated with “Extras”

An Extra is a key-value pair:

key = string

value = data of essentially any basic type or array

Use *putExtra(key, value)* function on an *Intent* object

List of all possible *putExtra* functions is available here:
http://developer.android.com/reference/android/content/Intent.html

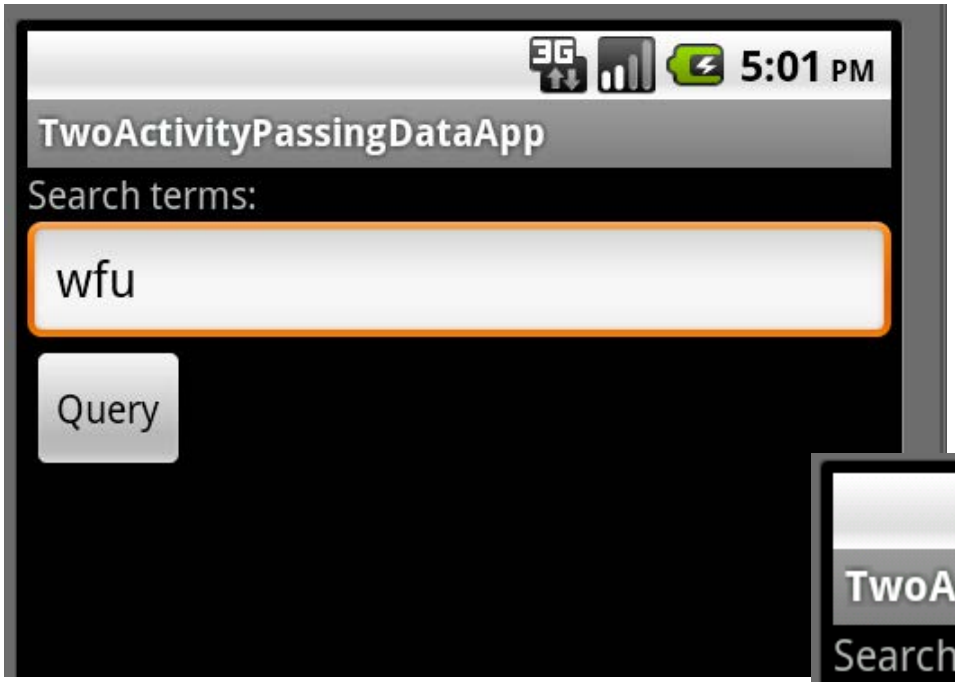
Sending Parameter Values to Activities

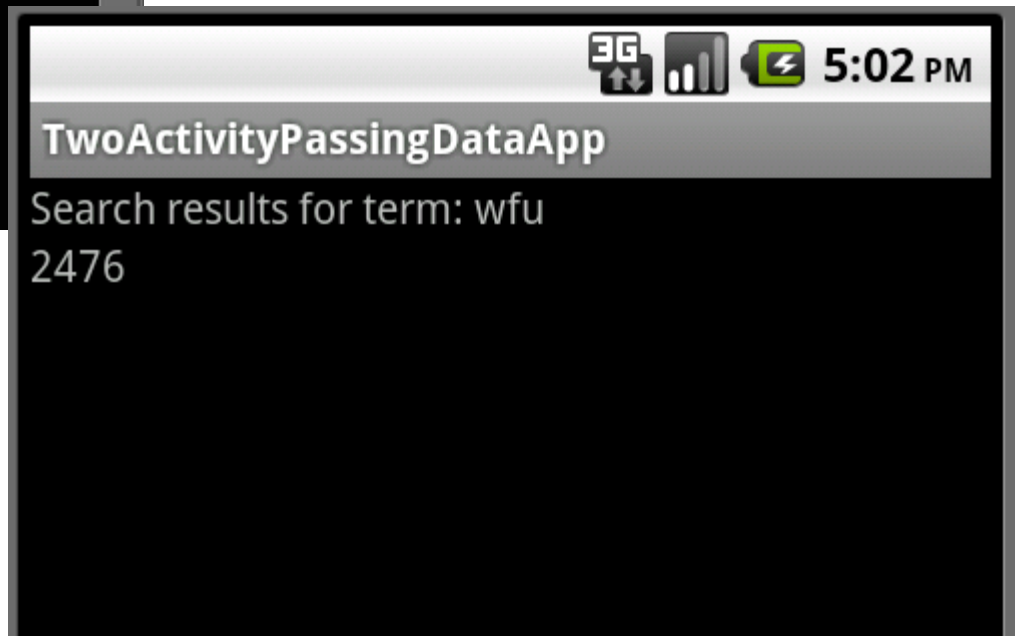
The notion of Extras is very similar to Hashtable datastructure/associative arrays:

Hashtable:

- Associates a key with a value based on a given hash function.
- When a query is made, a function is computed on the query key which says at what index in the table to find the value (or where to start searching)
- Typically employs *put/set* storage and *get* retrieval functions.

Sending Parameter Values to Activities





Parameters are search term string and “magic number”

Sending Parameter Values to Activities: Caller (Sender) Side

TwoActivityPassingDataAppActivity.java

```
package turkett.csc191;

import android.app.Activity;

public class TwoActivityPassingDataAppActivity extends Activity implements View.OnClickListener {

    Button searchButton;
    EditText searchTextField;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        searchTextField = (EditText)findViewById(R.id.search_field);
        searchButton = (Button)findViewById(R.id.search_button);
        searchButton.setOnClickListener(this);
    }

    public void onClick(View arg0) {
        Intent intent = new Intent(TwoActivityPassingDataAppActivity.this, SearchResultsActivity.class);
        intent.putExtra("searchTerm", searchTextField.getText().toString());
        intent.putExtra("magicValue", 2476);
        startActivity(intent);
    }


}
```

Sending Parameter Values to Activities: Callee (Receiver) Side

Receiver Activity can access the Intent that triggered the Activity by calling function:

Intent myReceivedIntent = getIntent();

With the Intent returned from *getIntent(),* requested Extras by using typed functions:

value = myReceivedIntent.getTypeExtra(key,…);

List of all possible *getTypeExtra* functions is available here:
http://developer.android.com/reference/android/content/Intent.html

Sending Parameter Values to Activities: Callee (Receiver) Side

SearchResultsActivity.java

```
package turkett.csc191;

import android.app.Activity;

public class SearchResultsActivity extends Activity {

        TextView labelText;
        TextView resultsText;

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

            labelText = (TextView)findViewById(R.id.search_results_label);
            resultsText = (TextView)findViewById(R.id.search_results);

            Intent callingIntent = getIntent();
            String searchTerm = callingIntent.getStringExtra("searchTerm");
            int magicNumber = callingIntent.getIntExtra("magicValue",-1);

            labelText.setText(labelText.getText() + " " + searchTerm);
            resultsText.setText(""+magicNumber);

        }

}
```

Return Values from Activities

Since we are considering inter-Activity communication like “function calls”, what are we still missing?

Receiving data back (return values)

Return Values from Activities

Two types of information that can be sent back to an Activity:

- A simple integer value result code
- Arbitrary data, stored in key-value pairs

Return Values from Activities

To have “return values” from an Activity, require a slightly different style of coding:

- Open the Activity as a Sub-Activity
- The Sub-Activity, upon closing, will trigger a listener in the parent (that initiated the activity) indicating that it was closed
- Info can be sent back through the function invoked when the listener is triggered

The Listener Pattern Strikes Again!

Return Values from Activities: Caller (Sender) Side

To open as a Sub-Activity:

- Create the Intent, as usual:

```
Intent intent = new Intent(MyActivity.this,
   MyOtherActivity.class);
```

- Generate a unique integer ID for that sub-activity

  (used to differentiate returns from multiple sub-activities)

```
int otherActivityID = 1;
```

- Start the activity, passing ID along

```
startActivityForResult(intent,otherActivityID);
```

Return Values from Activities: Caller Side

To get results from Sub-Activity in Activity:

- Listener function is triggered when the sub-activity finishes:

```
public void onActivityResult(int requestCode,
int resultCode, Intent data)
```

- Parameters:
  - requestCode – id of Sub-Activity that generated return value
  - resultCode – any integer value, two commonly used constants are Activity.RESULT_OK and Activity.RESULT_CANCELED
  - Intent data – all of the return data, bundled up in this Intent
    - Use *getTypeExtra* functions to extract data

Return Values from Activities: Caller Side

```
// unique id for sub-activity
private int RESULTS_ACTIVITY = 1;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    searchTextField = (EditText)findViewById(R.id.search_field);
    searchButton = (Button)findViewById(R.id.search_button);
    searchButton.setOnClickListener(this);
    magicValue = 2476;
}

public void onClick(View arg0) {
    // Make intent to call 2nd activity
    Intent intent = new Intent(TwoActivityReturnValueActivity.this, SearchResultsActivity.class);
    // Send a searchTerm string parameter
    intent.putExtra("searchTerm", searchTextField.getText().toString());
    // send a magic valud integer parameter
    intent.putExtra("magicValue", magicValue);
    // start the sub-activity waiting for a result, include sub-activity ID
    startActivityForResult(intent, RESULTS_ACTIVITY);
}

public void onActivityResult(int requestCode, int resultCode, Intent returnData) {
    // let parent handle the response as well
    super.onActivityResult(requestCode, resultCode, returnData);
    // if the return value is from desired sub-activity and has desired result code
    if ((requestCode == RESULTS_ACTIVITY) && (resultCode == RESULT_OK)) {
        // extract the returned magic value
        magicValue = returnData.getIntExtra("magicValue",-1);
        Log.v("TwoAct", "New Magic Value: " + magicValue);
    }
```

We don't have to indicate we are a special listener for this (i.e. we don't need to say *implements XYZ*) – we just have to write the function

Return Values from Activities: Callee Side

How does the Sub-Activity send back a response?

- Create an Intent to return
- Stuff the Intent with data using putExtra
- Call setResult(Intent, int) with the Intent to return and the returnCode integer
  - These values are given to the handler in the caller Activity
  - Activity.RESULT_OK and Activity.RESULT_CANCELED are two built-in return codes one can use
- Call finish() function
  - Finish kills the activity (pops it off the Activity stack)

Return Values from Activities: Callee Side

```
public class SearchResultsActivity extends Activity implements View.OnClickListener {

        TextView labelText;
        TextView resultsText;
        Button returnButton;
        int magicNumber;
        String searchTerm;

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

            labelText = (TextView)findViewById(R.id.search_results_label);
            resultsText = (TextView)findViewById(R.id.search_results);
            returnButton = (Button)findViewById(R.id.search_return_button);

            returnButton.setOnClickListener(this);

            Intent callingIntent = getIntent();
            searchTerm = callingIntent.getStringExtra("searchTerm");
            magicNumber = callingIntent.getIntExtra("magicValue",-1);

            labelText.setText(labelText.getText() + " " + searchTerm);
            resultsText.setText(""+magicNumber);
        }

        public void onClick(View args0) {
            Intent returnIntent = new Intent();
            returnIntent.putExtra("magicValue", magicNumber+1);
            setResult(Activity.RESULT_OK, returnIntent);
            finish();
        }
```

My example returns as its return value the magic number it was sent plus one

Return Values from Activities: Callee Side

If the user hits the physical “back” button on the phone, which “goes back an activity”:

- The caller’s *onActivityResult* function is called, but it receives a null Intent object and a return code of Activity.RESULT_CANCELED

Inter-Application Communication

- An application can also send out a general (instead of targeted) request for an action to be performed (an Intent)
  - Will need a different style of creating Intents
  - Need to specify:
    - General action to be performed
    - Data to be worked on

- Appropriate Activities will respond if they can perform the action
  - Need the ability to listen for these Intent requests

Inter-Application Communication

Use the Intent constructor which takes an action & data

Intent(String action, Uri dataURI)

An *URI* is a *uniform resource identifier* – usually a protocol and an address

http://www.cs.wfu.edu is a URL (and a URI) for web addresses

tel:1-337-758-4427 is a URI for telephone numbers

Inter-Application Communication

Intent Actions: Lots of built-in action descriptions, stored as constants in the Intent class

These are the current standard actions that Intent defines for launching activities (usually through `startActivity(Intent)`.

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CHOOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

Constant: Intent.ACTION_VIEW

Described in full here:
http://developer.android.com/reference/android/content/Intent.html

Inter-Application Communication

Making a request from your Activity to view a webpage:

Action: Intent.ACTION_VIEW

Data: Uri.parse(“http://www.cs.wfu.edu”)

(turns our string into a URI)

Inter-Application Communication Example

Making a request from your Activity to view a webpage:

Action: Intent.ACTION_VIEW

Data: Uri.parse(“http://www.cs.wfu.edu”)

(turns our string into a URI)

Inter-Application Communication Example

WebIntentExampleActivity.java

```
package turkett.csc191;

import android.app.Activity;

public class WebIntentExampleActivity extends Activity implements View.OnClickListener {

    Button goButton;
    EditText editText;

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

        goButton = (Button)findViewById(R.id.button);
        editText = (EditText)findViewById(R.id.edit_text);
        goButton.setOnClickListener(this);
    }

    public void onClick(View args0) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(editText.getText().toString()));
        startActivity(intent);
    }
}
```