

CS 696 Emerging Web and Mobile Technologies
Spring Semester, 2011
Doc 19 Android Activity Life Cycle
Mar 24, 2011

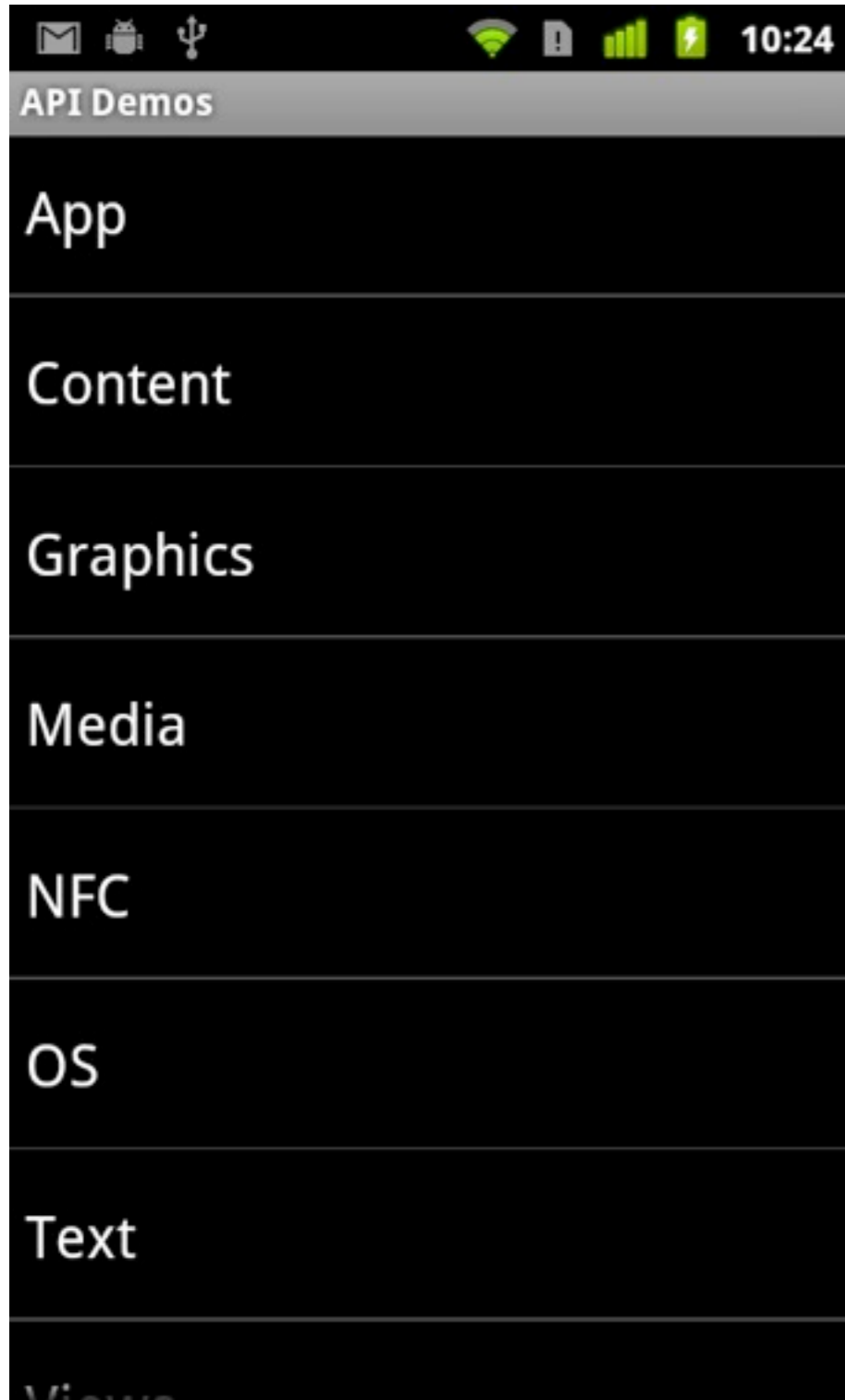
Copyright ©, All rights reserved. 2011 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

References

Android Developer's Guide, <http://developer.android.com/guide/index.html>

CS 696 Mobile Phone Application Development, Fall 2009,
<http://www.eli.sdsu.edu/courses/fall09/cs696/notes/index.html>

Examples



<http://developer.android.com/resources/browser.html>

Source

`androidInstallation/platforms/android-10/samples/ApiDemos`

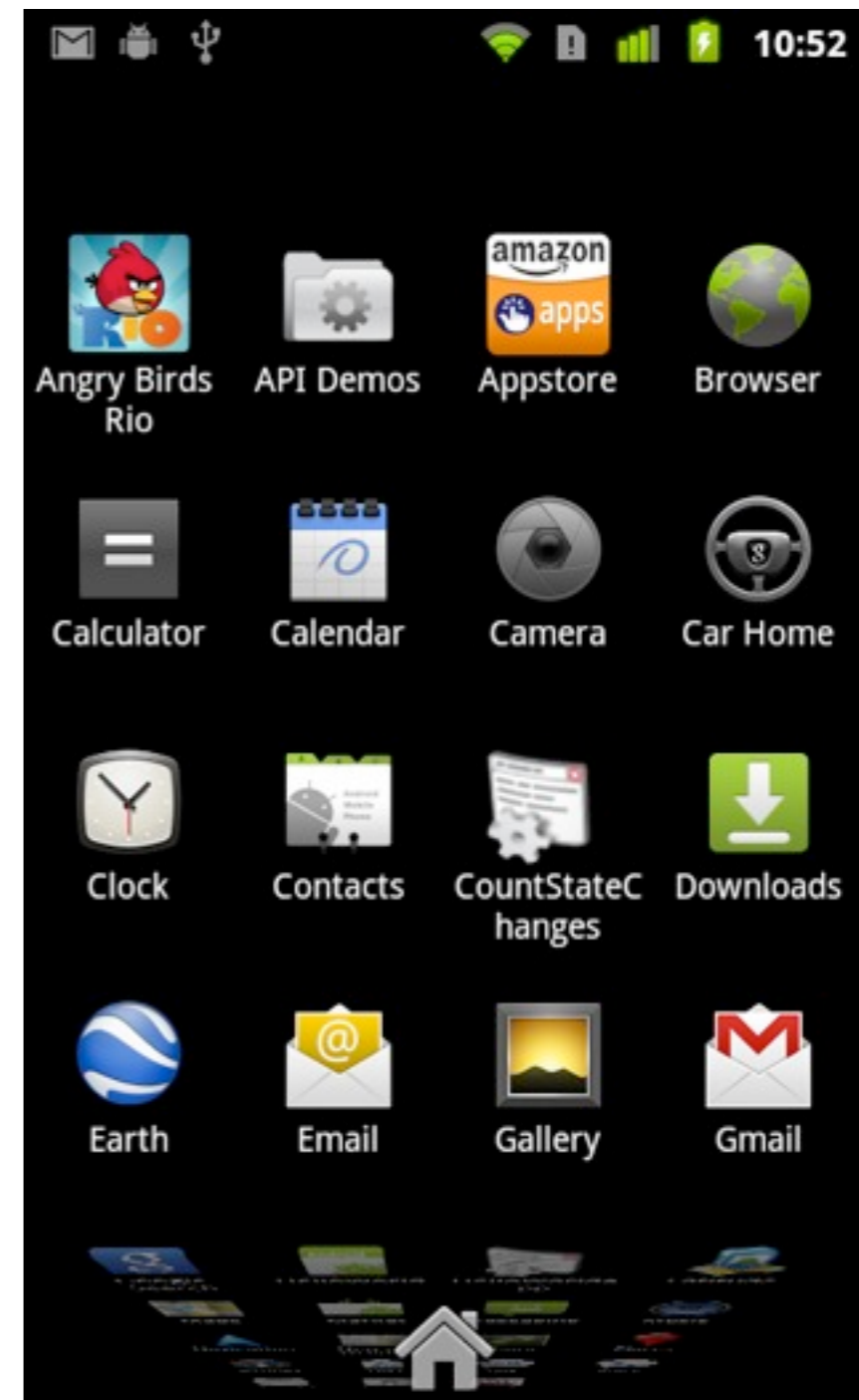
Activities & Tasks

Android Application

Set of related activities

Combined into one application file (.apk)

Launch-able from the home screen



Tasks

Sequence of activities the user follows to accomplish an objective

A user can

- Interrupt a task to start a new task

- Resume the first task where they left off

Tasks & Applications

Many applications are self contained

So task is sequence of activities from the application

Some applications use activities from other applications

Use phone

Show contacts

Use Web browser

Play music

So task is sequence of activities from multiple applications

Interrupting a Task

User presses Home and starts an application

Notifications

Activity Stack



Back Stack

History of activities used by user

May include activities of different applications

Back button

- Removes top of activity stack

- Makes next activity active

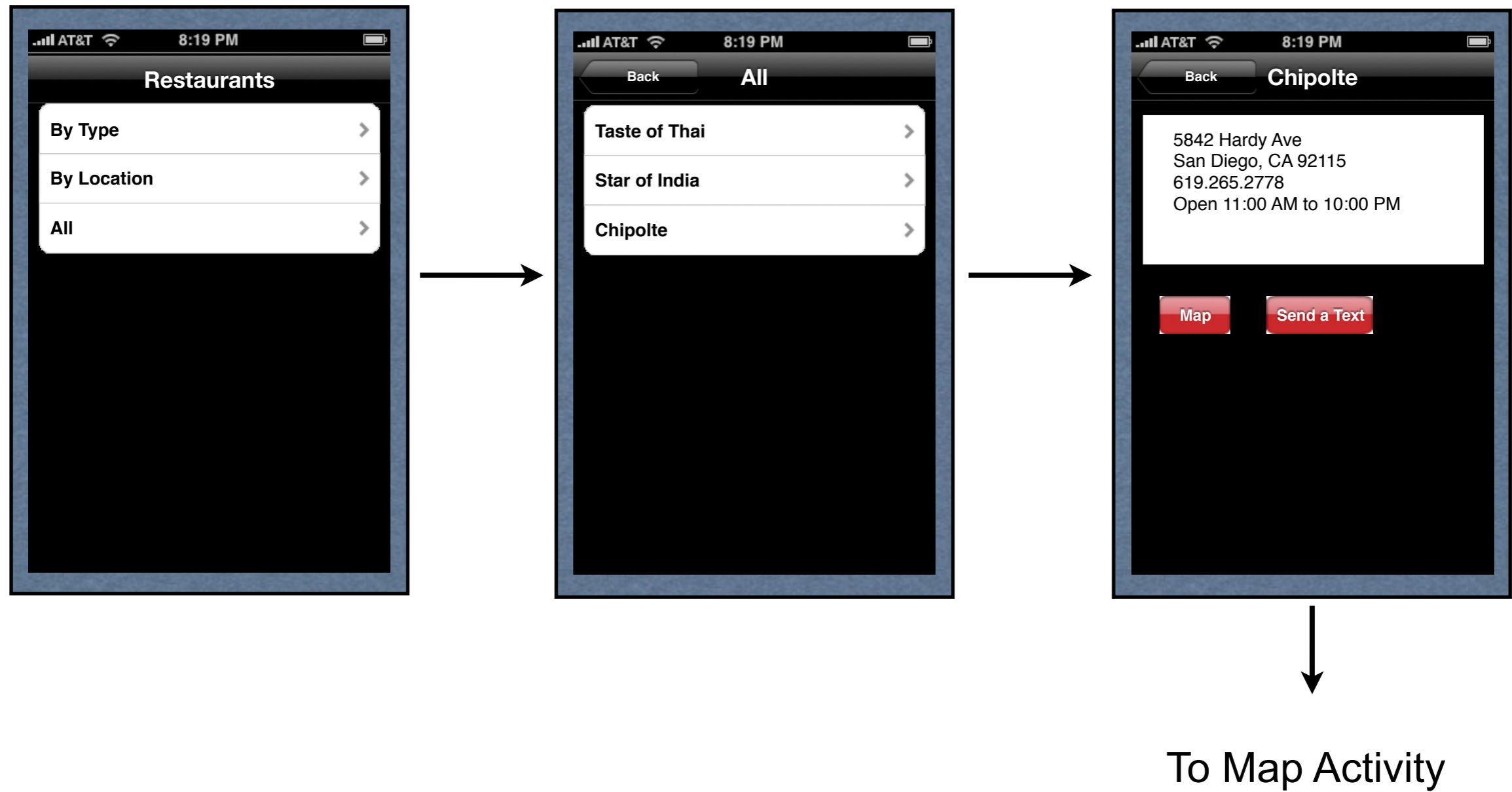
Home button

- Activity stack remains

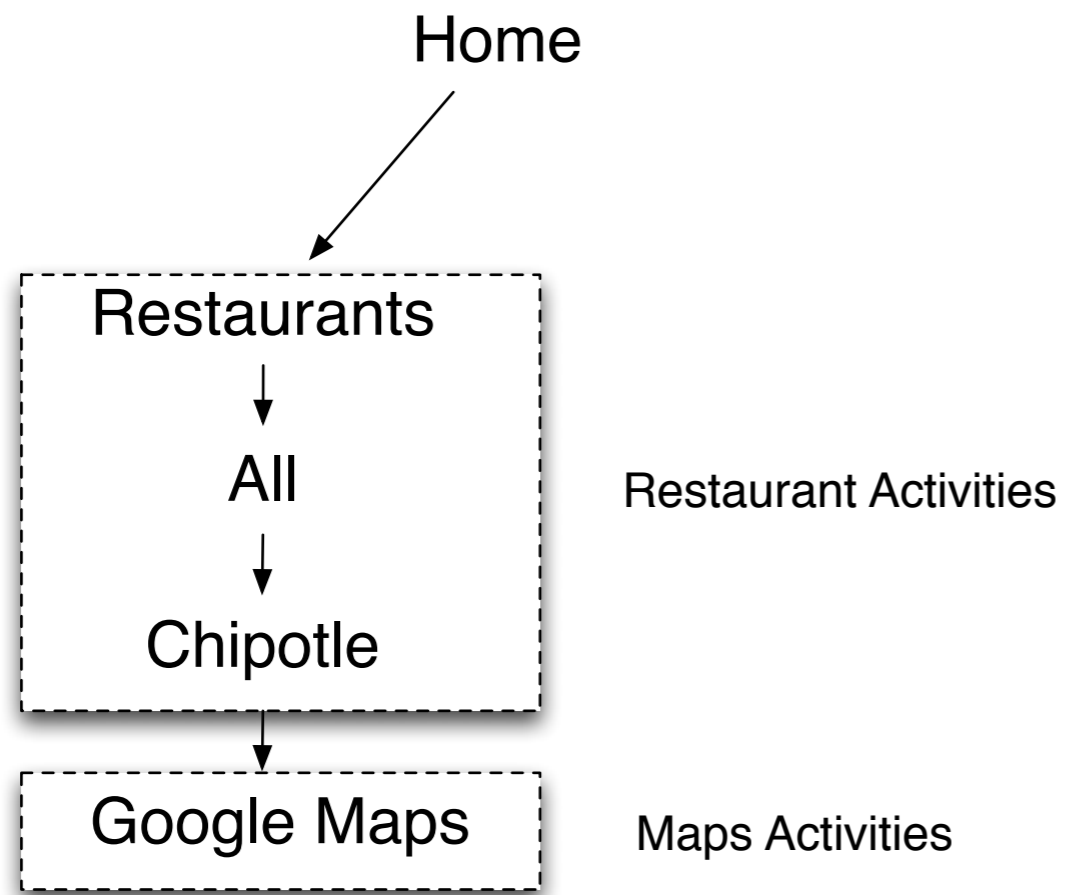
- Starting another application starts new activity stack

Stack only goes back to the start of the application at Home

Sample User Flow

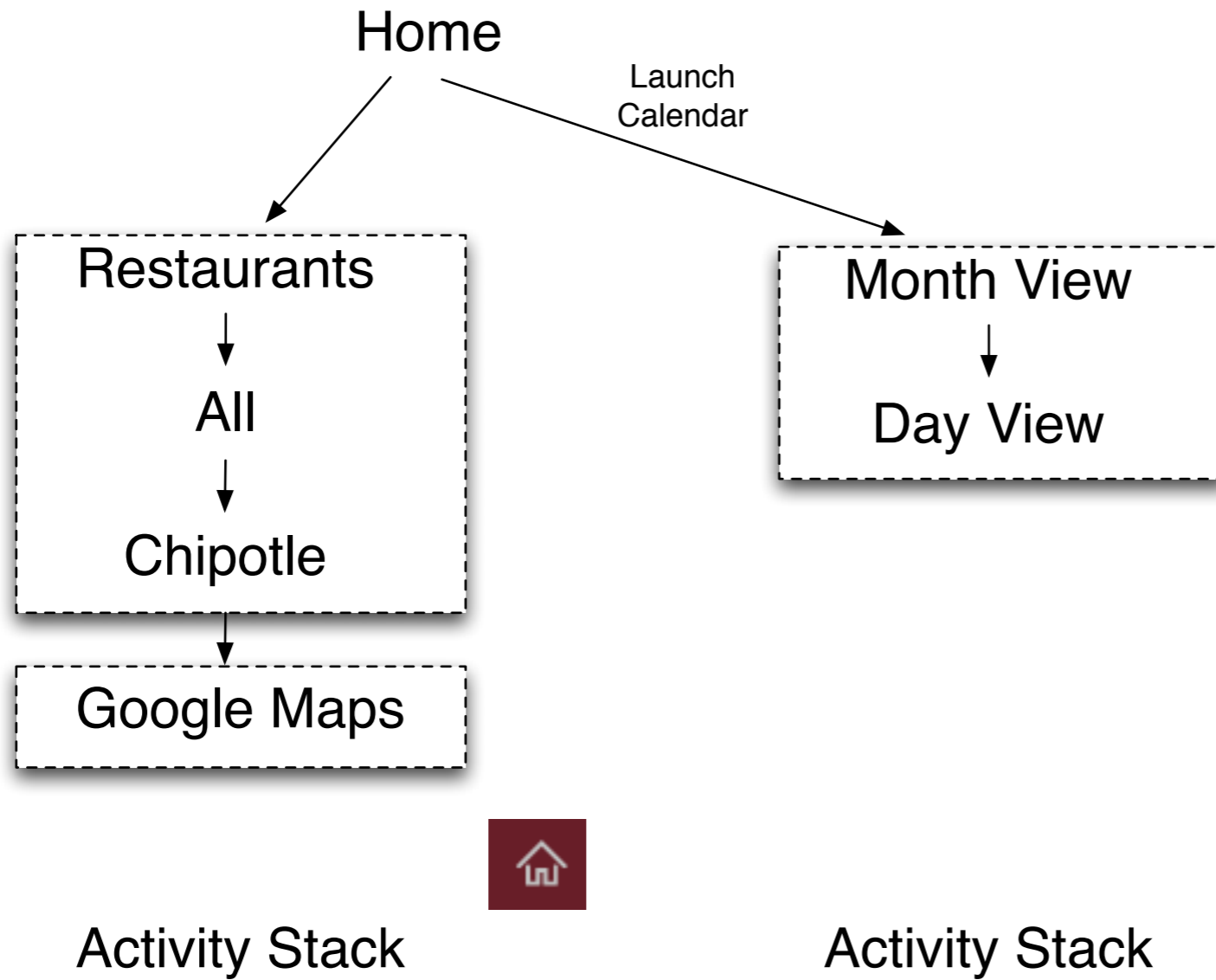


Activity Stack



Activity Stack

Multiple Activity Stacks



Applications & Activity Stacks

Launching a non-running application

- Create new activity stack

- Put application's beginning activity on stack

Launching a running application

- Show activity on top of applications activity stack

- That activity may be from another application

Exceptions

- Some background activities return to their initial screen

 - Contacts & Gallery

- Some activities continue to run while in the background

 - Music player

Activity Lifecycle States

Active (Resumed)

Running activity in foreground of screen

Paused

Lost focus, but still visible

Retains all state information

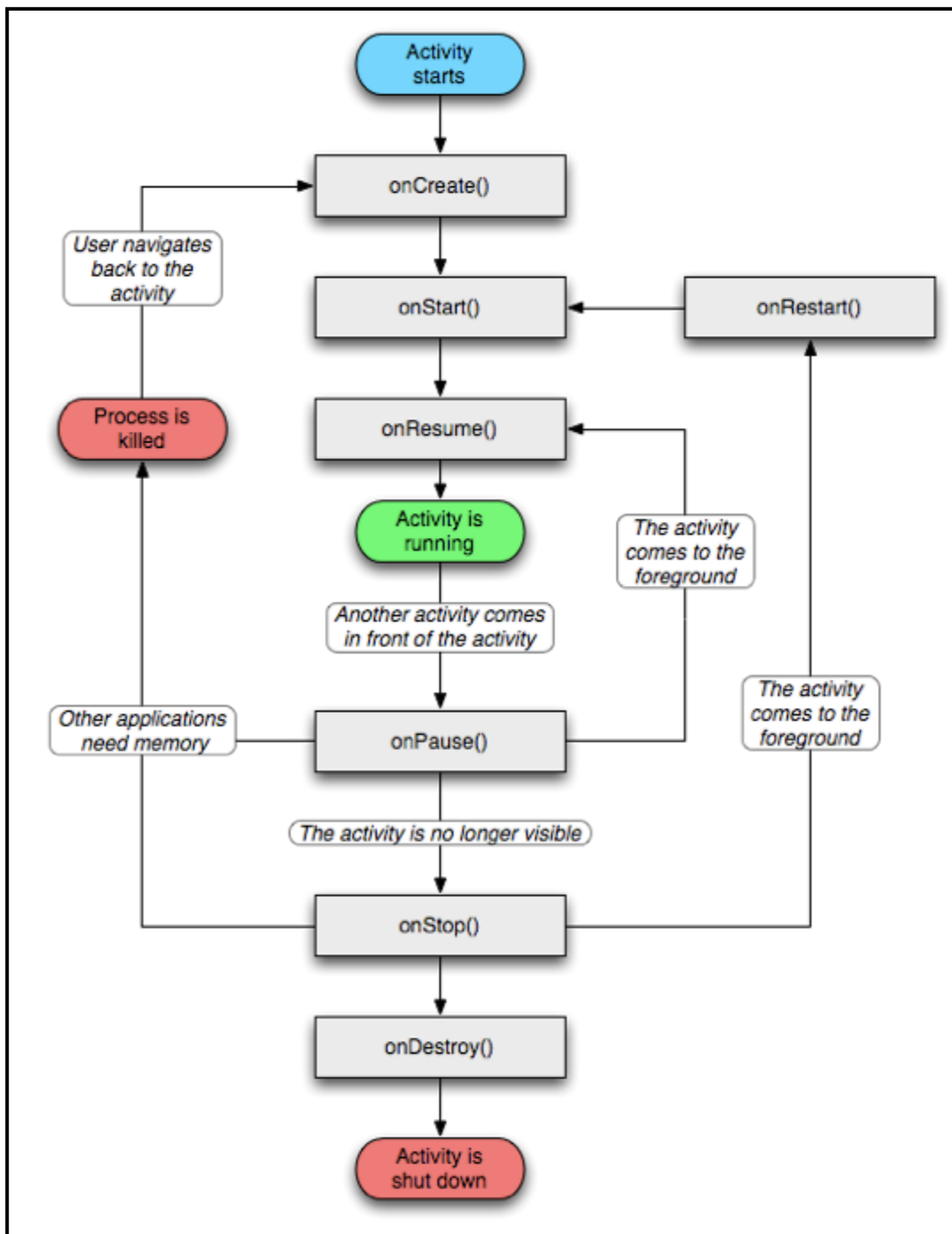
In extreme memory situations may be killed

Stopped

Not visible

Retains all state information

Often will be killed



Saving State

When low on memory system will kill activities

- In activity stack

- Not visible

When user goes back to killed activity

- Activity must appear as it did before it was killed

Must save state of activity

- System will save state of views

Types of State to Save

Dynamic instance state

- State of instance variables of activity
- Needed so activity object can operate

Persistent state

- Information that should be available next time application is run
- Contact information in Address book

Overlap

- Persistent state is usually subset of dynamic state

Saving Persistent State

Do it in the onPause() method

It will always be called

One method that will always be called before activity is killed

onStop() and onDestroy() are not always called

onStop()

Called when activity is no longer visible

Not always called

onDestroy()

Used to free resources like threads

There are situations when

"system will simply kill the activity's hosting process
without calling this method"

finnish()

Sending "finnish()" to an activity will kill the activity

Normally don't call this method

Saving/Restoring Dynamic Instance State

protected void onSaveInstanceState(Bundle outState)

Called after onPause

Save data in bundle

Restore state in

onCreate or

onRestoreInstanceState

Activity State Change Example

Showing State Changes

Count number of times called
onPaused(), onStoped(), onDestroy()

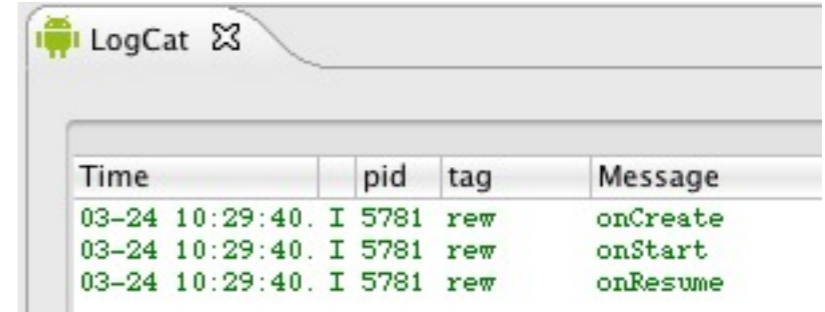
Send to screen message from each onXXX()

Send to log a message from each onXXX()

Touch lower textfield - start web browser

Touch upper textfield - open dialog

Start up



LogCat window showing log messages:

Time	pid	tag	Message
03-24 10:29:40. I	5781	rew	onCreate
03-24 10:29:40. I	5781	rew	onStart
03-24 10:29:40. I	5781	rew	onResume



Web Browser for second activity



touch →



back button →

LogCat

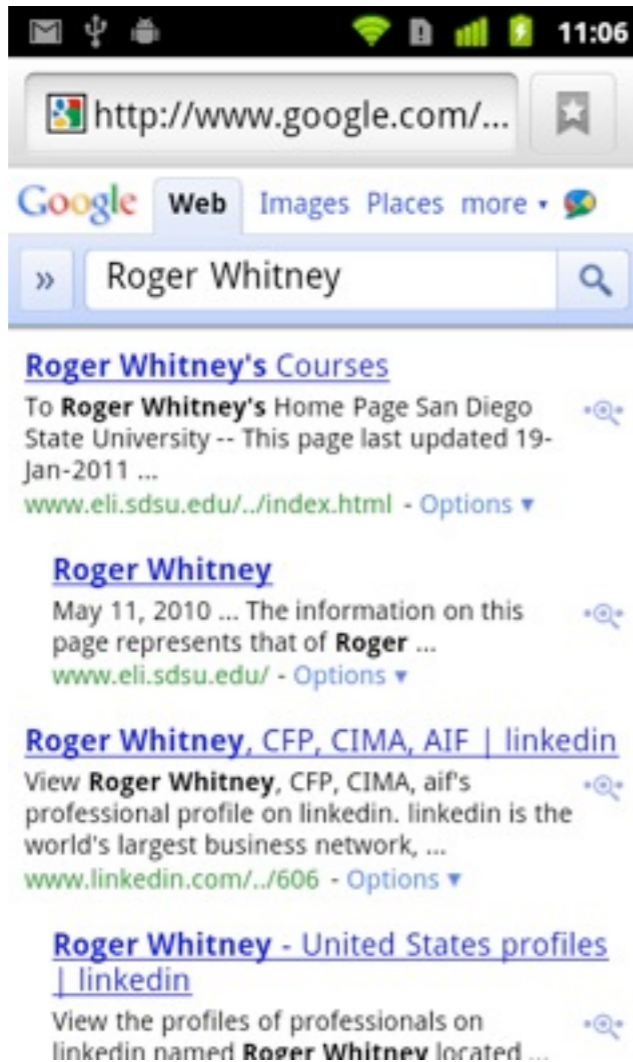
Time	pid	tag	Message
03-24 11:00:25. I	5938	rew	onCreate
03-24 11:00:25. I	5938	rew	onStart
03-24 11:00:25. I	5938	rew	onResume

LogCat

Time	pid	tag	Message
03-24 11:00:25. I	5938	rew	onCreate
03-24 11:00:25. I	5938	rew	onStart
03-24 11:00:25. I	5938	rew	onResume
03-24 11:01:24. I	5938	rew	onSaveInstanceState
03-24 11:01:24. I	5938	rew	onPause
03-24 11:01:25. I	5938	rew	onStop

Web Browser for second activity

back
button



back
button



LogCat

Time	pid	tag	Message
03-24 11:00:25	I 5938	rew	onCreate
03-24 11:00:25	I 5938	rew	onStart
03-24 11:00:25	I 5938	rew	onResume
03-24 11:01:24	I 5938	rew	onSaveInstanceState
03-24 11:01:24	I 5938	rew	onPause
03-24 11:01:25	I 5938	rew	onStop
03-24 11:04:26	I 5938	rew	onRestart
03-24 11:04:26	I 5938	rew	onStart
03-24 11:04:26	I 5938	rew	onStop

LogCat

Time	pid	tag	Message
03-24 11:00:25	I 5938	rew	onCreate
03-24 11:00:25	I 5938	rew	onStart
03-24 11:00:25	I 5938	rew	onResume
03-24 11:01:24	I 5938	rew	onSaveInstanceState
03-24 11:01:24	I 5938	rew	onPause
03-24 11:01:25	I 5938	rew	onStop
03-24 11:04:26	I 5938	rew	onRestart
03-24 11:04:26	I 5938	rew	onStart
03-24 11:04:26	I 5938	rew	onStop
03-24 11:08:07	I 5938	rew	onRestart
03-24 11:08:07	I 5938	rew	onStart
03-24 11:08:07	I 5938	rew	onResume

Dialog



touch



back



LogCat

Time	pid	tag	Message
03-24 11:26:59. I	5938	rew	onCreate
03-24 11:26:59. I	5938	rew	onStart
03-24 11:26:59. I	5938	rew	onResume

LogCat

Time	pid	tag	Message
03-24 11:26:59. I	5938	rew	onCreate
03-24 11:26:59. I	5938	rew	onStart
03-24 11:26:59. I	5938	rew	onResume

Rotation



LogCat

Time	pid	tag	Message
03-24 11:26:59. I	5938	rew	onCreate
03-24 11:26:59. I	5938	rew	onStart
03-24 11:26:59. I	5938	rew	onResume

LogCat

Time	pid	tag	Message
03-24 11:26:59. I	5938	rew	onCreate
03-24 11:26:59. I	5938	rew	onStart
03-24 11:26:59. I	5938	rew	onResume
03-24 11:30:18. I	5938	rew	onSaveInstanceState
03-24 11:30:18. I	5938	rew	onPause
03-24 11:30:18. I	5938	rew	onStop
03-24 11:30:18. I	5938	rew	onDestroy
03-24 11:30:18. I	5938	rew	onCreate
03-24 11:30:18. I	5938	rew	onStart
03-24 11:30:18. I	5938	rew	onRestoreInstanceState
03-24 11:30:18. I	5938	rew	onResume

Source Code

onCreate

```
public class CountStates extends Activity implements View.OnTouchListener {
    int paused = 0;
    int killed = 0;
    int stopped = 0;
    TextView text;
    TextView logger;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (savedInstanceState != null) {
            paused = savedInstanceState.getInt("paused");
            killed = savedInstanceState.getInt("killed");
            stopped = savedInstanceState.getInt("stopped");
        }
        setContentView(R.layout.main);
        text = (EditText) this.findViewById(R.id.text);
        text.setOnTouchListener(this);
        logger = (EditText) this.findViewById(R.id.log);
        logger.setText("");
        logger.setOnTouchListener(this);
        updateText("onCreate");
    }
}
```

Touch and updateText

```
public boolean onTouch(View v, MotionEvent event) {
    if (v == logger) {
        Intent web = new Intent(Intent.ACTION_WEB_SEARCH);
        web.putExtra(SearchManager.QUERY, "Roger Whitney");
        startActivity(web);
    }
    if (v == text) {
        showDialog(0);
    }
    return true;
}

private void updateText(String eventType) {
    Log.i("rew", eventType);
    text.setText("Paused: " + paused + " stopped: " + stopped + " killed "
        + killed);
    logger.append(eventType + "\n");
}
```


Dialog

```
protected Dialog onCreateDialog(int id) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Hello").setPositiveButton("Ok",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                Toast.makeText(getApplicationContext(), "Good Bye",
                    Toast.LENGTH_SHORT).show();
            }
        });
    return builder.create();
}
```


Saving Instance State

```
protected void onRestoreInstanceState(Bundle savedInstanceState) {  
    super.onRestoreInstanceState(savedInstanceState);  
    updateText("onRestoreInstanceState");  
    if (savedInstanceState != null) {  
        paused = savedInstanceState.getInt("paused");  
        killed = savedInstanceState.getInt("killed");  
        stopped = savedInstanceState.getInt("stopped");  
    }  
}
```

```
protected void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
    updateText("onSaveInstanceState");  
    outState.putInt("paused", paused);  
    outState.putInt("killed", killed);  
    outState.putInt("stopped", stopped);  
}
```

onXXX()

```
protected void onResume() {  
    super.onResume();  
    updateText("onResume");  
}
```

```
protected void onPause() {  
    paused++;  
    updateText("onPause");  
    super.onPause();  
}
```

```
protected void onStart() {  
    super.onStart();  
    updateText("onStart");  
}
```

```
protected void onStop() {  
    stopped++;  
    updateText("onStop");  
    super.onStop();  
}
```

```
protected void onRestart() {  
    super.onRestart();  
    updateText("onRestart");  
}
```

```
protected void onDestroy() {  
    killed++;  
    updateText("onDestroy");  
    super.onDestroy();  
}
```