Android
II – Activities and Intents

Stefan BORNHOFEN
EISTI

Principles

- Applications can launch other applications
- Application can be interrupted by the system (e.g. incoming call)
- On single application is visible to the user (activity stack)
- Save memory and battery

Activity Lifecycle

An activity can exist in essentially three states:

Resumed (Running)

- The activity is in the foreground and the user can interact with it.

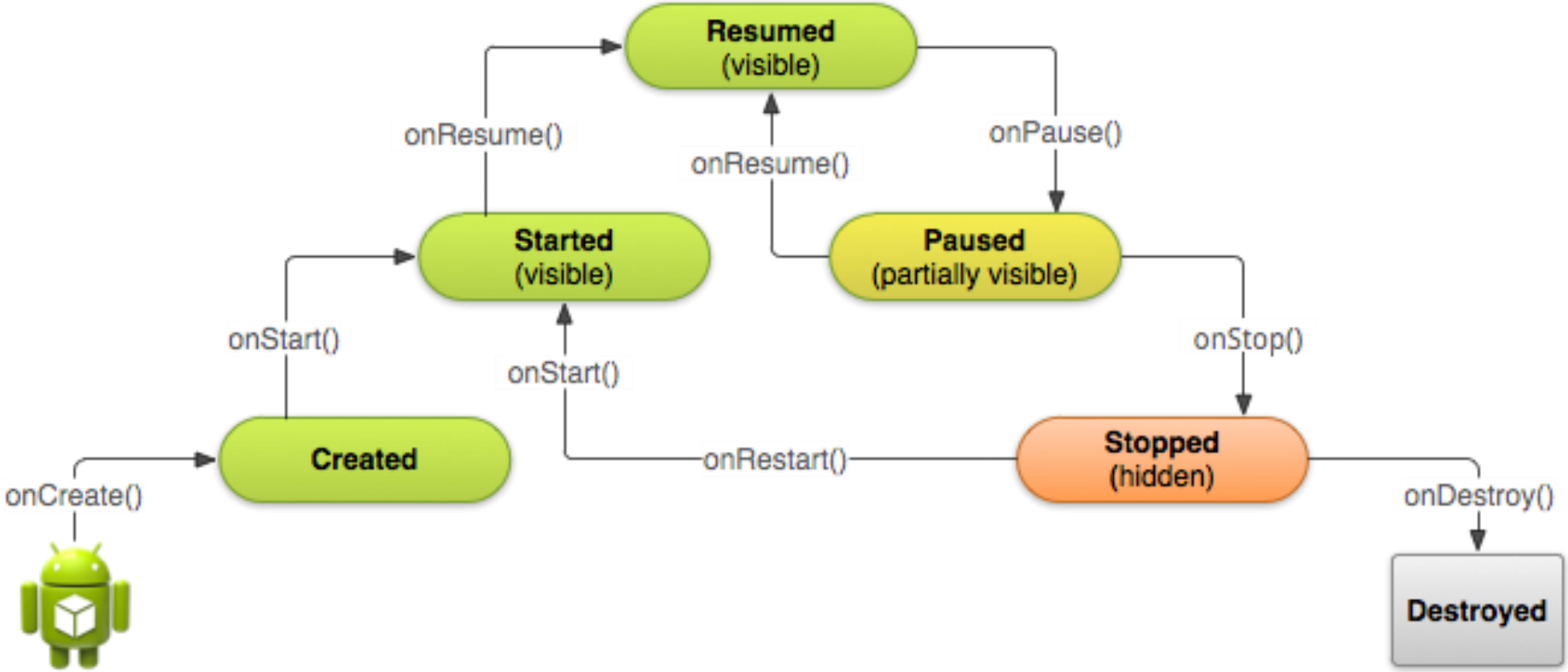
Paused

- The activity is partially obscured by another activity—the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.

Stopped

- The activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables is retained, but it cannot execute any code. It can be killed by the system when memory is needed elsewhere.

Activity Lifecycle



Callbacks

The two most important callback methods are:

onCreate()

- You must implement this method. This is where you call setContentView() to define the layout
- One parameter: Bundle. Null if first creation, previous state if the activity has been destroyed by the system (e.g. for memory)

onPause()

- The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed). This is usually where you should commit any changes that should persist beyond the current user session (because the user might not come back).

Instance state

- The system uses the Bundle instance state to save information about each View object in the activity layout (such as the text value entered into an EditText object). So, if your activity instance is destroyed and recreated, the state of the layout is automatically restored to its previous state.

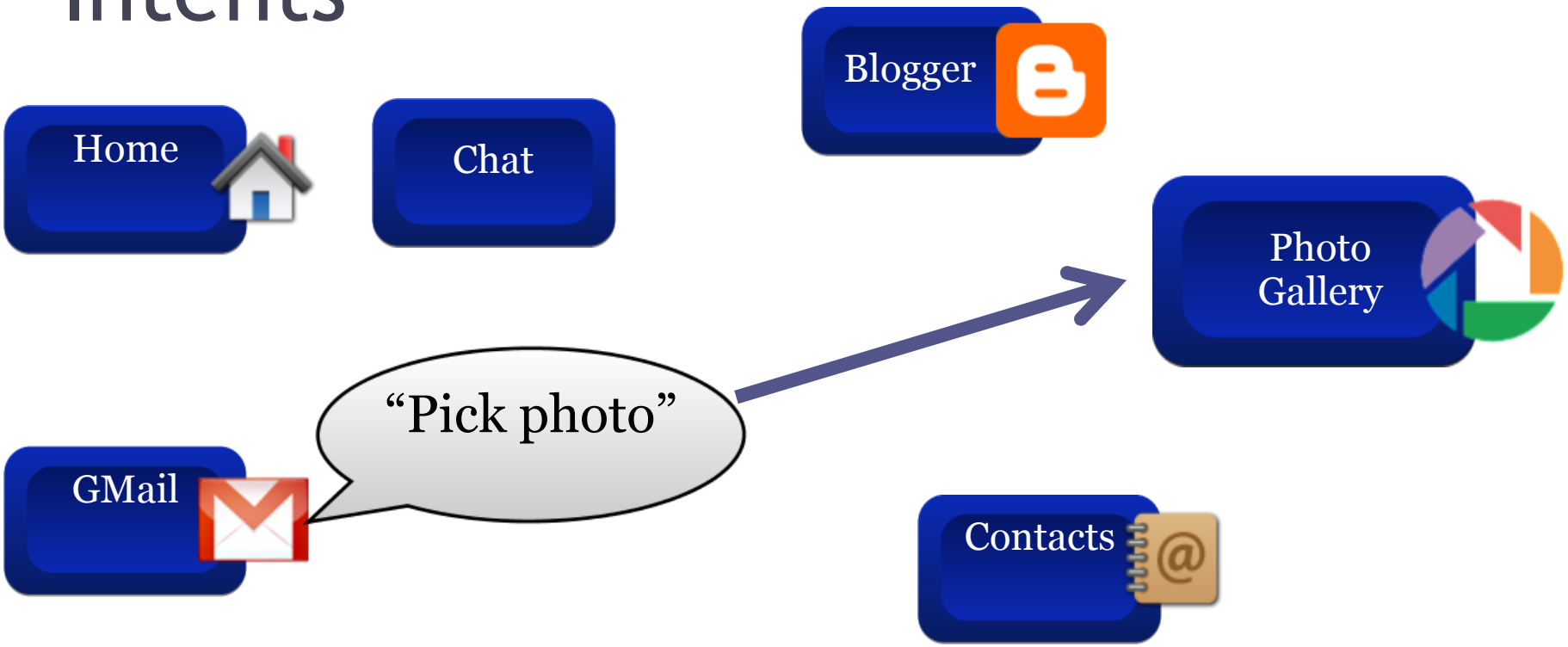
- In order for you to add additional data to the saved instance state for your activity, there are two additional callbacks:

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putInt("score", currentScore);
    super.onSaveInstanceState(savedInstanceState);
}
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    currentScore = savedInstanceState.getInt("score");
   // you can choose to do this in onCreate() instead of here
}
```

Intents

- Messages between Activities
- Activities are started by intents
- Explicit intent: you specify the exact Activity class to launch
- Implicit intent: the system matches Intent with the activity that can best provide the service
- Activities, Services and BroadcastReceivers describe what Intents they can service (*IntentFilters)*

Intents



Client component makes a request for a specific action

Think of Intents as a verb and object; a description of what you want done

e.g. VIEW, CALL, PLAY etc..

Android picks best component for that action

Advantage: New components can use existing functionality

Other Native Android Actions

- ACTION_ANSWER – handle incoming call
- ACTION_DIAL – bring up dialer with phone
- ACTION_PICK – pick item (e.g. from contacts)
- ACTION_INSERT – add item (e.g. to contacts)
- ACTION_SENDTO – send message to

More:
http://developer.android.com/guide/appendix/g-app-intents.html

Implicit Intents

```
// view contact list
Intent i = new Intent(Intent.ACTION_VIEW,
  android.provider.ContactsContract.Contacts.CONTENT_URI);

// make a call
Intent i = new Intent(Intent.ACTION_CALL_BUTTON,null);

// view picture gallery
Intent i = new
  Intent(Intent.ACTION_VIEW,android.provider.MediaStore.Im
  ages.Media.INTERNAL_CONTENT_URI);

startActivity(i);
```

Explicit Intents

```
Intent i = new Intent(this,
fr.eisti.android.SecondActivity.class);
```

Dont forget to register the second activity in the AndroidManifest.xml:

```
<activity android:name=".SecondActivity"
  android:label="@string/title_snd_activity">
</activity>
```

Implicit Intents of own activities

```
Intent i = new Intent("android.intent.action.MYACTION");
```

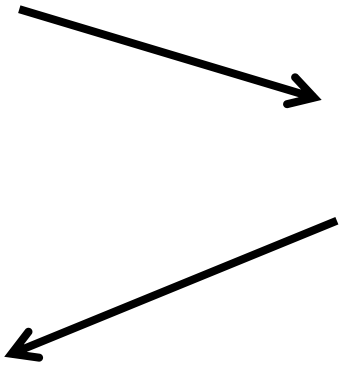
Dont forget to register the second activity in the AndroidManifest.xml:

```
<activity android:name=".SecondActivity"
  android:label="@string/title_snd_activity">
  <intent-filter>
    <action android:name="android.intent.action.MYACTION" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

Intents with extra values

Calling Activity	Called Activity

```
...
Intent i = new Intent(this, SecondActivity.class);
i.putExtra("key", value);
startActivityForResult (i, MYREQUEST);
```



```
// get calling intent
Intent i = getIntent();
String s = i.getStringExtra("key");
...
// create return intent
Intent returnIntent = new Intent();
returnIntent.putExtra("result", result);
setResult(RESULT_OK, returnIntent);
finish();
or
Intent returnIntent = new Intent();
setResult(RESULT_CANCELED,
returnIntent);
finish();
```

```
protected void onActivityResult (int
requestCode, int resultCode, Intent data) {

   if (requestCode == MYREQUEST) {
       if(resultCode == RESULT_OK){
        String result=data.getStringExtra("result");
   }
}
```

Service

- Application component representing an application's desire to perform a longer-running operation while not interacting with the user.
- Services run in the background.
- They do not have any user interface.
- *Unbound services* run in the background indefinitely, even if the activity which started this service ends.
- *Bound services* run until the end of the activity which started this service.

Example SoundService

```
public class BackgroundSoundService extends Service {

    MediaPlayer player;
    @Override
    public void onCreate() {
        super.onCreate();
        player = MediaPlayer.create(this, R.raw.mysong);
        player.setLooping(true);
        player.setVolume(100,100);

    }
    public int onStartCommand(Intent intent, int flags, int startId) {
        player.start();
        return 1;
    }
    public IBinder onBind(Intent intent) { return null; }
    public IBinder onUnBind(Intent intent) { return null; }
    public void onStart(Intent intent, int startId) { }
    public void onStop() { }
    public void onPause() { }
    @Override
    public void onDestroy() { player.stop(); player.release(); }
}
```

Service

Declare the service in the manifest

```
<service
  android:name=".MyService">
  <intent-filter>
    <action android:name="android.intent.action.MYSERVICE" />
   </intent-filter>
</service>
```

Service

Start and stop the service in your activity (unbound)

```
@Override
protected void onStart() {
    super.onStart();
    Intent serviceIntent = new Intent(this,
  fr.eisti.android.BackgroundSoundService.class);
    this.startService(serviceIntent);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Intent serviceIntent = new Intent(this,
  fr.eisti.android.BackgroundSoundService.class);
    this.stopService(serviceIntent);
}
```

Exercise

Write the Colorpicker with explicit and implicit intents (2p).

Add the following extensions (1p):
- Splash screen
- Background music
- Restore color data if restart

Implicit intent

Menu

Explicit intent