Activity

App di oggi

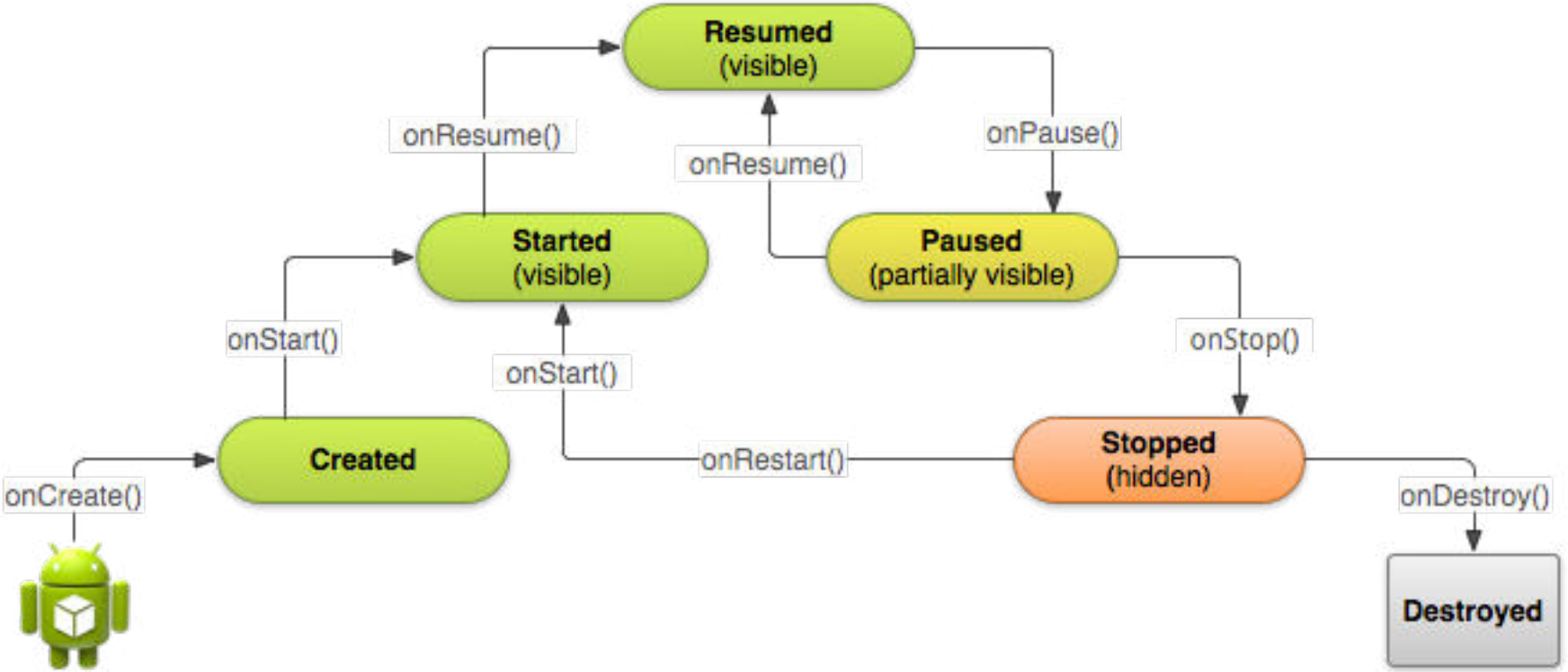

10:26
SingleNote
scrivi la nota....
COLOR
SHARE


10:28
SingleNote
scrivi la nota....
Color Picker
#FF3344
CANCEL
SAVE
COLOR
SHARE

Activity

- Una Activity
  - è realizzata con una classe java che estende android.app.Activity
  - rappresenta una singola schermata grafica della nostra applicazione
  - viene solitamente associata ad un'azione specifica che l'utente può fare
  - è si attiva se in primo piano
- Una applicazione contiene solitamente molte Activity
- Ogni Activity ha il compito di memorizzare e rispristinare lo stato delle sue variabili di istanza

Stati di una Activity



GESTIONE DELLO STATO

Gestione dello stato

Ci sono due metodi per salvare e rispristinare lo stato:

- onRestoreInstanceState
  - chiamata tra onResume e onCreate
- onSaveInstanceState
  - chiamata prima di onPause

Esempio

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("counter", counter);
}

@Override
protected void onRestoreInstanceState(Bundle
savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    counter = savedInstanceState.getInt("counter");
    showCounterState();
}
```

Esempio

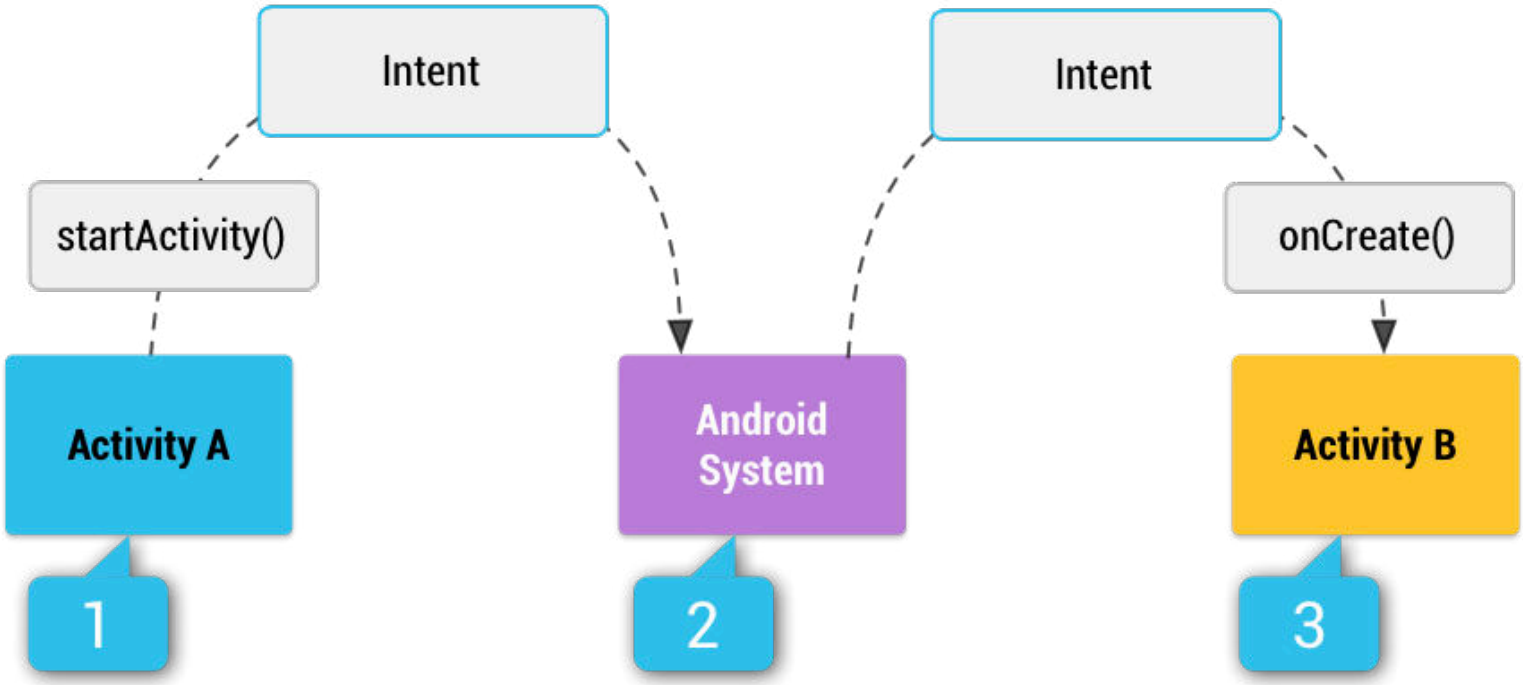
- ciclo di vita Activity

Intent

Intent

- un messaggio contenente informazioni inviate ad un altro oggetto
- chi lo riceve può effettuare operazioni in base al suo contenuto
- Tre casi base:
  - Fare lo start di una **activity**
  - Fare lo start di un **service**
  - Inviare un messaggio broadcast
- Tipi di Intent
  - Impliciti
  - Espliciti

Start di una activity



Struttura dell'Intent

- Component name
  - specifica in modo esplicito il componente che lo riceverà
- Action
  - è una stringa che identifica il tipo di operazione che sarà eseguita
- Data
  - l'URI e tipo di dato (MIME Type) della risorsa che vogliamo elaborare
- Category
  - è una stringa contenente informazione sul tipo di componente che lo può ricevere
- Extras
- Flags

INTENT ESPLICITI

Intent espliciti

Servono a lanciare Activity o Servizi della propria app

ES1

```
Intent intent = new Intent(EsplicitIntentTestActivity.this,SecondActivity.class);
startActivity(intent);
```

ES2

```
Intent intent = new Intent();
ComponentName component = new ComponentName(EsplicitIntentTestActivity.this,
                                            SecondActivity.class);
intent.setComponent(component);
startActivity(intent);
```

ES3

```
Intent intent = new Intent();
Class<?> destination = Class.forName("it.apogeo.android.cap04.lifecycleactivitytest.LifecycleActivityTestActivity");
ComponentName component = new ComponentName(EsplicitIntentTestActivity.this, destination);
intent.setComponent(component);
startActivity(intent);
```

Activity che restituiscono valori

- Lanciare una activity per far ottenre un valore (URI) di ritorno

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(android.provider.ContactsContract.Contacts.CONTENT_URI);
startActivityForResult(intent, CHOOSE_CONTACT_CODE);
```

- Quando l'activity child "esce" viene chiamato il metodo

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

Esempio

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode,resultCode,data);
        if(requestCode==CHOOSE_CONTACT_CODE){
                if(resultCode == Activity.RESULT_OK){
                }else if(resultCode == Activity.RESULT_CANCELED){
                }else{}
        }else{}
}
```

Passare dati in Bundle

```
Intent activityIntent = new Intent(this, NewActivity.class);

Bundle newActivityInfo = new Bundle();
newActivityInfo.putBlah(…); // putDouble, putString, etc.
activityIntent.putExtras(newActivityInfo);
startActivity(activityIntent);


Intent intent = getIntent();
Bundle info = intent.getExtras();
if (info != null) { /* Retrieve vals with info.getBlah(...) */ }
```

Generazione del risultato

- Nell'activity child
  - Gestisco l'esito
  - Posso impostare il dato nell'intent che ritorno

- Dati di ritorno
  - URI del dato
  - Extra
    - `public Bundle getExtras ()`
    - `public Intent putExtras (Bundle extras)`

Esito positivo

```
setResult(Activity.RESULT_OK, dataIntent);
finish();
```

Esito negativo

```
setResult(Activity.RESULT_CANCELED, dataIntent);
finish();
```

oppure solo finish()

INTENT IMPLICITI

Intent Impliciti

- Lanciare componenti di una qualsiasi app nel telefono
- Specifico una azione che voglio venga compiuta
  - ACTION SEND
  - ACTION EDIT
- I componenti dichiarano di poter compiere delle azioni
  - Intent filter

Se nessuno può ricevere l'intent l'app va in crash

Intent ed Action

- una action è una stringa che identifica il tipo di operazione che sarà eseguita
  - Intent.*ACTION_DIAL*
    - “android.intent.action.DIAL”
  - *it.apogeo.android.cap04.senderproject.intent.action.CUSTOM_ACTION*
- Sintassi della stringa
  - <package>.intent.action.<action name>
- Metodi
  - setAction()

Esempi azioni standard

- Costanti della classe Intent
  - ACTION_VIEW
    - content://contacts/people/1
  - ACTION_DIAL
    - content://contacts/people/1
  - ACTION_VIEW
    - tel:123
  - ACTION_DIAL
    - tel:123
  - ACTION_EDIT
    - content://contacts/people/1
  - ACTION_VIEW
    - content://contacts/people/

- Esempio di esecuzione della activity per inserire un numero

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_DIAL);
startActivity(intent);
```

Intent, action e dati

- In generale una coppia action/dato specifica l'activity da eseguire
- la forma del dato è una URI (RFC 2396)
  - scheme://host:port/path
- Esempi
  - ACTION_VIEW content://contacts/people/1
  - ACTION_DIAL content://contacts/people/1
  - ACTION_VIEW tel:123
  - ACTION_DIAL tel:123
  - ACTION_EDIT content://contacts/people/1
  - ACTION_VIEW content://contacts/people/

Action custom

Receiver

- Si usa un Intent filter per dichiarare la volontà di gestire l'azione nel receiver

```
<intent-filter>
        <action android:name="cap04.senderproject.intent.action.CUSTOM_ACTION">
        </action>
        <category android:name="android.intent.category.DEFAULT">
        </category>
</intent-filter>
```

Sender

- Si usa un Intent a cui si imposta l'action oppotrtuna

```
Intent intent = new Intent();
intent.setAction(CUSTOM_ACTION);
startActivity(intent);
```

Categorie

- Categoria dell'intent
- Categorie di default
  - **CATEGORY_DEFAULT**, CATEGORY_BROWSABLE, CATEGORY_TAB, CATEGORY_ALTERNATIVE, CATEGORY_SELECTED_ALTERNATIVE, CATEGORY_LAUNCHER, CATEGORY_INFO, CATEGORY_HOME, CATEGORY_PREFERENCE, CATEGORY_TEST ,...
- Si imposta con addCategory()

Data

- MIME Type
  - Identificatore del tipo di dato
  - RFC 2046 (http://en.wikipedia.org/wiki/Internet_media_type)
  - tipo/sottotipo
    - image/jpeg, image/png, image/*, text/plain, text/html, ...
- URI
  - <scheme>://<host>:<port>[<path>|<pathPrefix>|<pathPattern>]
- Metodi di Intent
  - setDataAndType (Uri data, String type)
  - setType(String)
  - setData(Uri)
  - setDataAndTypeAndNormalize(Uri, String)

Extra

- Coppie chiave-valore
  - Dati richiesti per compiere l'azione voluta
- Metodi
  - **putExtra**(String, <Tipo>)
  - get<Tipo>Extra
- Esempi di Extra standard
  - EXTRA_EMAIL, EXTRA_PHONE_NUMBER, EXTRA_REFERRER, EXTRA_SHORTCUT_ICON, EXTRA_SHORTCUT_NAME, EXTRA_SUBJECT, EXTRA_TEMPLATE, EXTRA_TEXT, EXTRA_TITLE, EXTRA_UID

Flags

- Servono a comunicare al sistema operativo come deve gerite il lancio del componente
- Si settano nell'intent
  - setFlags() o addFlag()

```
intent.addFlag(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
intent.setFlags(intent.getFlags() | FLAG_ACTIVITY_NEW_TASK);
intent.setFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT |
        Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
```

```
// example...
// value of flags: 1
intent.setFlags(2|4);
// now flags have this value: 110
intent.addFlags(8);
// now flags have this value: 1110
```

Costruttori di Intent

- Intent()
  - Create an empty intent.
- Intent(Intent o)
  - Copy constructor.
- Intent(String action)
  - Create an intent with a given action.
- Intent(String action, Uri uri)
  - Create an intent with a given action and for a given data url.
- Intent(Context packageContext, Class<?> cls)
  - Create an intent for a specific component.
- Intent(String action, Uri uri, Context packageContext, Class<?> cls)
  - Create an intent for a specific component with a specified action and data.

INTENT FILTER

Intent Filter

- Dichiarano quali intent impliciti un componente può ricevere
  - Ogni componente ne può dichiarare più di uno

- Un intent filter può essere composto da
  - action – nome action
  - category – nome category
  - data – MIME Type e/o schema URI

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Intent Resolution

- Action
  - Un Intent filter può dichiarare zero o più action
  - **L'action specificata nell'intent deve essre dichiarata nel filtro**
- Category
  - Un Intent filter può dichiarare zero o più category
  - **Tutte le categorie dell'intent devono essere presenti nel filter**
- Type e URI
  - se presenti devono essere dichiarate nel filtro
  - sono analizzate separatamente
  - possono essere dedotte dai dati nell'intent

Filtri sul contenuto

- Filtri su uri
  - <scheme>://<host>:<port>/<path>
    - http, mailto,
  - controllato in modo gerarchico
    - prima lo schema poi l'autority e poi il path
- Filtri mimeType
  - cioè il tipo di contenuto
    - immagini (image/jpeg)
    - video (audio/mpeg4-generic)
    - contatti (specifico di Android)
    - etc.

Esempi

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"></action>
    <data android:mimeType="vnd.android.cursor.dir/contact"></data>
    <category android:name="android.intent.category.DEFAULT"></category>
</intent-filter>
```

```
<intent-filter>
    <data android:mimeType="image/*" />
    ...
</intent-filter>
```

```
<intent-filter>
    <data android:scheme="http" android:type="video/*" />
    ...
</intent-filter>
```

Package manager

- La classe che permette gestire le app si chiama Package Manager
  - Permette di scoprire quali app soddisfano un intent

Esempio di Intent filter

```
<intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.HOME" />
        <category
android:name="android.intent.category.DEFAULT"></category>
</intent-filter>
```

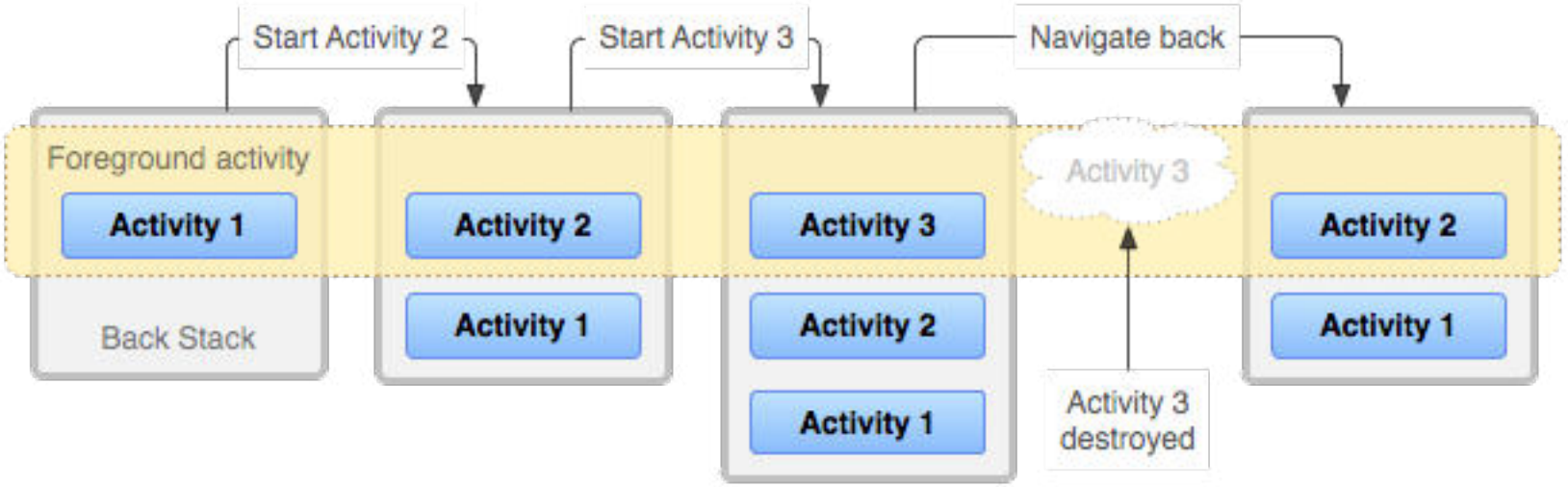
Esempio di ricerca delle activity che possono essere eseguite per lanciare una app

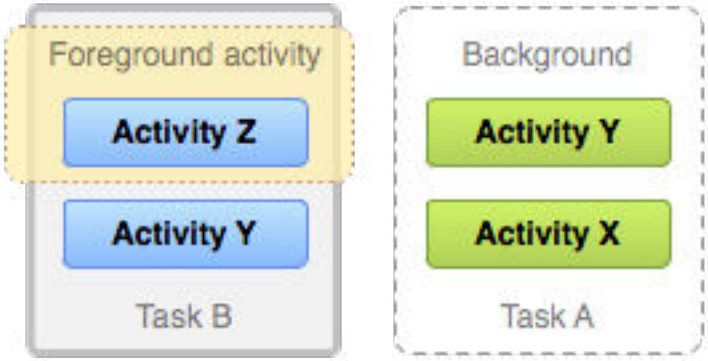
```
Intent intent = new Intent(Intent.ACTION_MAIN);
intent.addCategory(Intent.CATEGORY_LAUNCHER);
PackageManager packageManager = getPackageManager();
List<ResolveInfo> infoList =
        packageManager.queryIntentActivities(intent, 0);
```

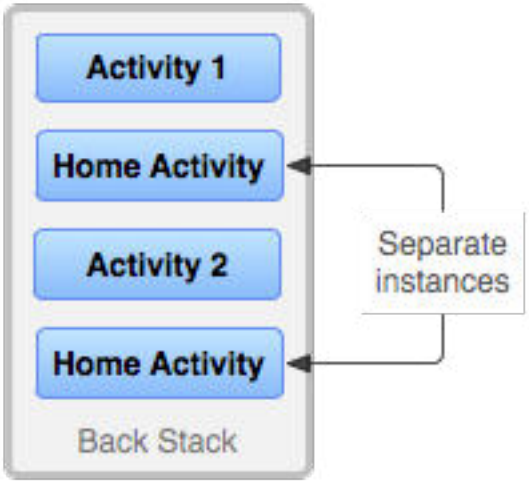
BACKSTACK E TASK

I Task

- Ad un applicazione viene associato almeno un task
  - numero che identifica una sequenza di activity
- Le activity si alternano sul display
  - c'è uno stack di Activity
- Se non specificato le activity condividono lo stesso task
- Per lanciare una Activity in un nuovo task posso fare due cose:
  - dichiarare un attributo taskaffinity
    - android:taskAffinity=*":Other"*
  - settare nell'Intent un flag specifico
    - Intent.*FLAG_ACTIVITY_NEW_TASK*







https://developer.android.com/guide/components/activities/tasks-and-back-stack