Temari

1) Introducció
2) Framework Android
3) Projectes Android i Android SDK
4) Activity
5) Fragments, Views i ListViews
6) Intents
7) Layouts i Custom Views
8) Resources i Themes
9) Dialogs, Menus i WebView
10) Persistència de dades
11) Tasques en Background i internet
12) SQLite i content providers
13) Notificacions

9 – Dialogs, menus i WebView

- Dialogs
- AlertDialog
- ProgressDialog
- Toasts
- Menus
- Context Menus
- Pop-up menus
- WebView

Interacció amb l'usuari i missatges

Per donar missatges de feedback o preguntar a l'usuari, és essencial.

Hi ha vàries alternatives:

- Fer servir les classes que android ofereix per fer **Dialogs**
- Crear una activity i donar-li aspecte de **dialog** canvinat-li el **theme**
- Fer un **Toast** (aquests no necessiten una activity, es poden fer servir des de processos en background)

Dialogs

Petites finestres que apareixen davant l'activity, i ocupen una porció de la pantalla → són part de l'activity (no cal declarar-los al manifest)

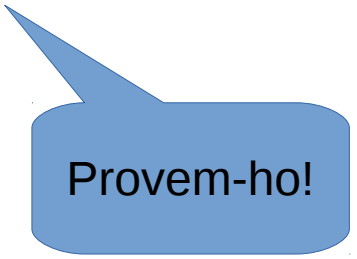
Android ofereix algunes sub-classes de Dialog per a l'ús més comú

- AlertDialog
- ProgressDialog
- DatepickerDialog
- TimePickerDialog

Dialogs amb un layout propi

A través d'un layout en XML podem *inflar* un **dialog**

- Crear un layout en un xml
- Crear una variable de tipus **Dialog**
- Cridar al mètode **setContentView()** del Dialog
- Amb el mètode **findViewById()** del Dialog, obtenir les Views i configurar-les
- Cridar al mètode **show()** del dialog
- Quan el vulguem tancar, cridar al mètode **cancel()**



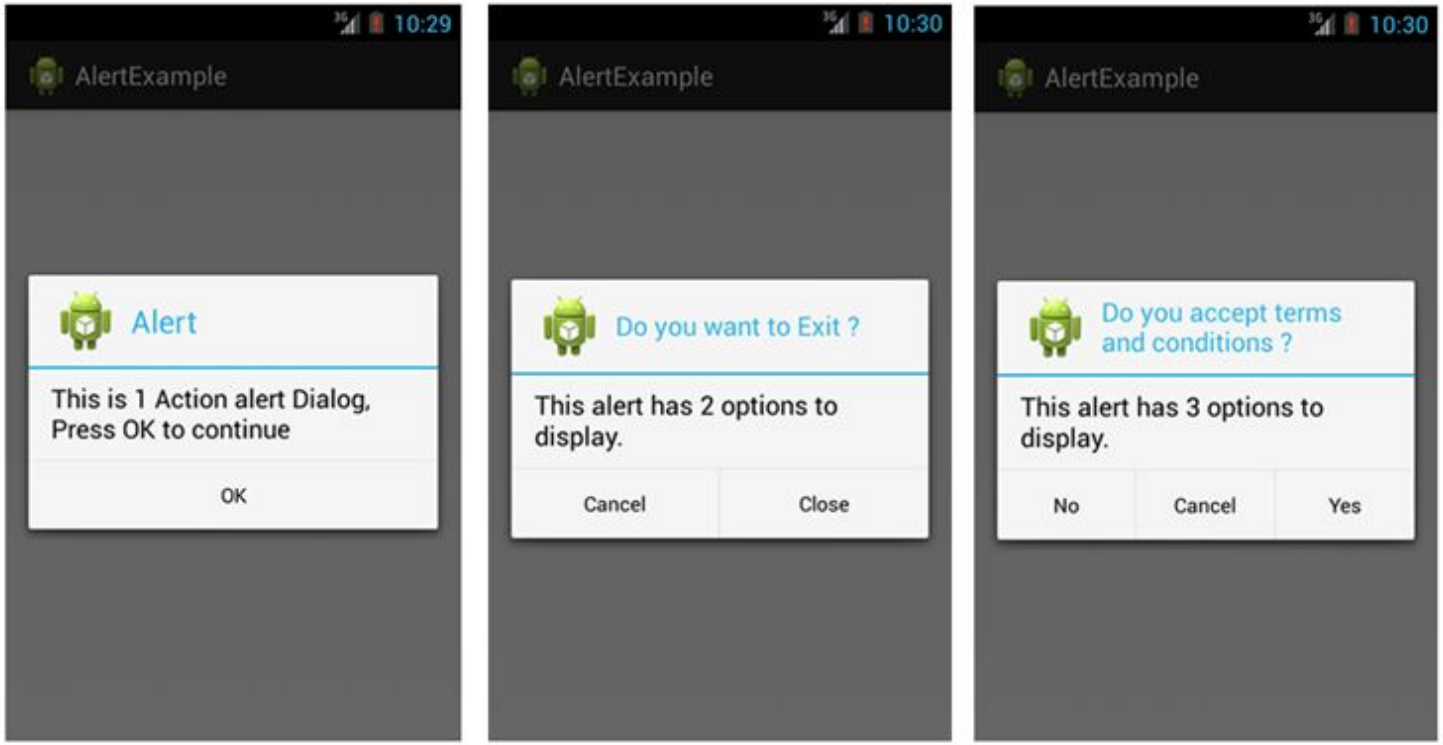
Dialogs: Modificar tota la finestra

Cal obtenir i modificar l'objecte **Window:**

```
Window window = dialog.getWindow();
...
Window.setFlags(WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
                WindowManager.LayoutParams.FLAG_BLUR_BEHIND);
Window.setLayout(LayoutParams.MATCH_PARENT,
                LayoutParams.WRAP_CONTENT);
```


AlertDialogs

Subclasse de Dialog, que ofereix directament l'opció de posar fins a 3 botons, un títol i un missatge



Provem-ho!

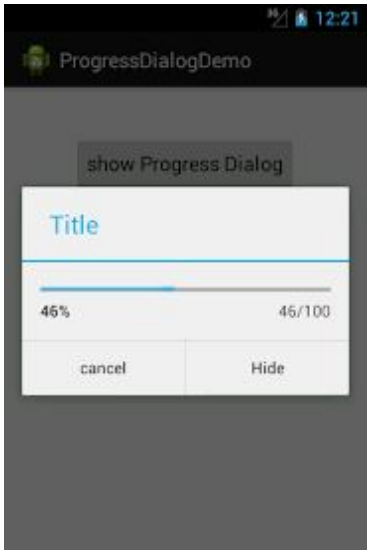
ProgressDialog

Subclasse de AlertDialog, A més del que ofereix l'AlertDialog, mostra un element que pot ser una barra de progrés o un *Spinner*

```
progress=new ProgressDialog(this);
progress.setMessage("Downloading Music");
progress.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progress.setIndeterminate(true);
progress.setProgress(0);
progress.show();

final int totalProgressTime = 100;
final Thread t = new Thread() {
   @Override
   public void run() {
      int jumpTime = 0;

      while(jumpTime < totalProgressTime) {
         try {
            sleep(200);
            jumpTime += 5;
            progress.setProgress(jumpTime);
         }
         catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   }
};
t.start();
```



Activities que semblen Dialogs

Tot i que la majoria de vegades amb la classe Dialog o una de les seves subclasses es pot resoldre, també es pot fer que una activity sembli un Dialog

→ posar-li al AndroidManifest.xml el tema dels Dialogs:

```
<activity android:theme="@android:style/Theme.Dialog" />
```


Toasts

Permeten donar un missatge sense interrompre l'activitat de l'usuari → per missatges que **no** són crucials

Desapareixen al cap d'un temps: configurable amb dos possibles valors:

- Toast.LENGHT_LONG
- Toast.LENGHT_SHORT (per defecte)

Se'ls hi pot posar un layout i Views a dins per configurar-los, però no es poden **inflar amb un xml**.

Provem-ho amb una imatge

Alternativa: **SnackBar**

Menus

En dispositus/versions anteriors a 3.0, en pressionar la tecla de menú

En dispositius posteriors, en un botó a la ActionBar

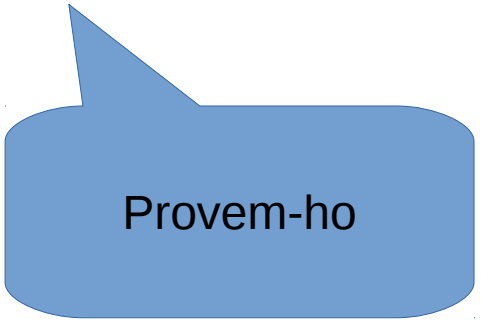
Dues opcions per definir quins elements hi ha al menú:

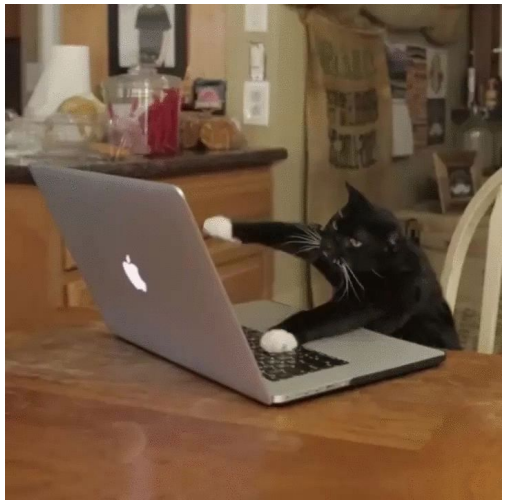
- Des de codi Java (Menu API)
- Des de XML (inflant el menú)

Sobreescriure el mètode
OnCreateOptionsMenu()
De l'Activity

Menus: Codi Java

- Fer servir el mètode **Menu.add()** per obtenir l'ítem de menú
- Configurar l'ítem de menú (setShortcut(), setIcon(), **setCheckable()** ...)
- Opcionalment, definir l'Intent que s'enviarà en seleccionar-lo amb **setIntent()**





Menus: XML

- Crear un XML dins **res/menu**
- En el **onCreateOptionsMenu()** inflar-lo

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```


Menus: gestionant la selecció

- Sobreescriure el mètode **onOptionsItemSelected()**
- Segons el id de l'Item sabem quina opció s'ha triat → fer l'acció adequada

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```


Menus: Submenus

- En Java:

  SubMenu sub = **Menu.addSubMenu();**

  MenuItem subItem = sub.add();

- En XML:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```


ContextMenus

Menus que es mostren en fer click sobre una View durant 3 segons

- Obtenir una View
- Cridar al mètode **registerForContextMenu(view)**

- Sobreescriure el mètode **onCreateContextMenu()** i crear el menú (o inflar-lo) com en el cas del menú principal

- Sobreescriure el mètode **onContextItemSelected()** per a gestionar els clicks

Popup Menus

Menus configurables en fer click sobre una View

1) Crear un menú al **res/menu**
2) Obtenir una view i afegir-li l'event **onClick()** o configurar-lo directament al layout (com els buttons)
3) Al onClick, crear un nou **PopupMenu** i inflar-lo

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.actions, popup.getMenu());
    popup.show();
}
```


WebView

Un tipus de View que mostra contingut web

- Afegir una WebView al layout
- Obtenir-la al codi java amb **findViewById()**
- Opcionalment, configurar el «navegador» (per ex. habilitar javascript)
  webView.getSettings().setJavascriptEnabled();
- Opcionalment, un **WebClient** per a que gestioni els redirects

```
webview.setWebViewClient(new WebViewClient() {
    public boolean shouldOverrideUrlLoading(WebView view, String url){
        // do your handling codes here, which url is the requested url
        // probably you need to open that url rather than redirect:
        view.loadUrl(url);
        return false; // then it is not handled by default action
   }
});
```

- Fer que carregui una URL o un HTML: **loadUrl(url)** o **loadData(html)**
- Cal que la app tingui permisos per **Internet**