UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

disim

Mobile Programming Laboratory

ANDROID
Menu and Dialog

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

- Menu
- Dialog
  - AlertDialog
  - Custom Dialog

Menu

In Android the Menu is a list of action buttons showed on Application Toolbar.

The developers can design the menu using:

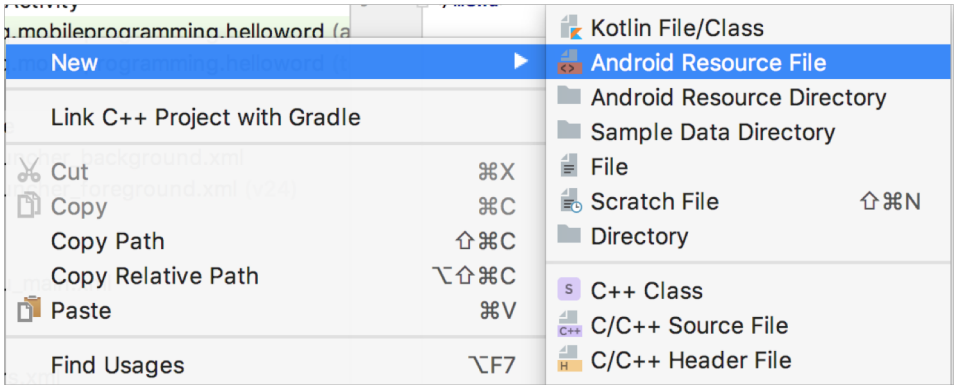
- XML file (recommended)
- JAVA

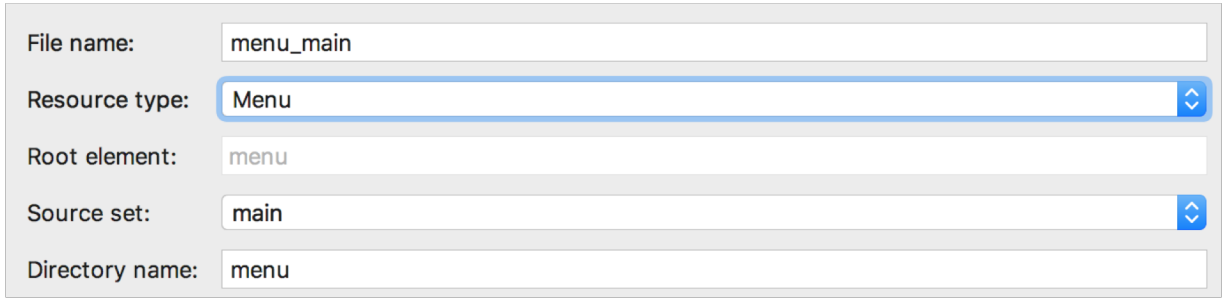
Menu

The developer must create a new file inside the folder **res/menu/**

Using the Android Studio resource tool the developer can create it easily.

the tool appears if you perform a right click on res folder and select Android resource file



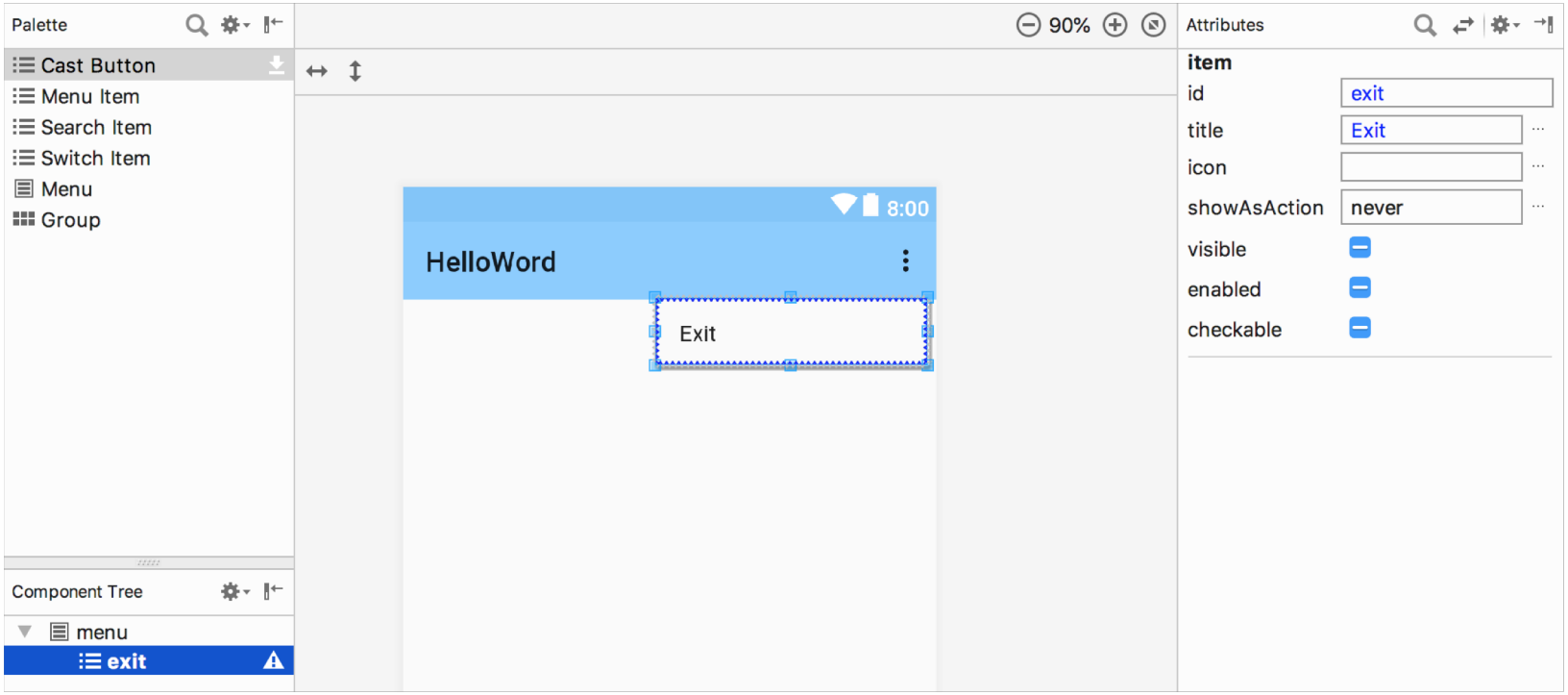


Menu

The Menu is composed in a list of items. This items can be:

menu (simple), **switch**, **search**

There is the possibility to have more groups with own items.




Menu

The corresponding XML code is:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <item android:id="@+id/exit"
     app:showAsAction="never"
     android:title="Exit"/>

</menu>
```

id: is required to retrieve the item in Java code

title: is the only required attribute to show the item

showAsAction: when and how the item should appear as an action item.

never: Never place this item in the app bar
always: Always place this item in the app bar
withText: Also include the title text
ifRoom: Only place this item in the app bar if there is room for it
collapseActionView: The action view is collapsible

Menu

The menu is linked to the Activity or Fragment where it will show.

These classes have two method to manage the menu layout and its items:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  return super.onOptionsItemSelected(item);
}
```

onCreateOptionsMenu connects the "**menu**" object (in input) to the XML layout using a specific Inflater of Menu: **MenuInflater**

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.menu_main, menu);
  return super.onCreateOptionsMenu(menu);
}
```

Menu

If the developer want to implement the action when a menu item is clicked, he has to use the method onOptionItemSelected.

The MenuItem in input is the item clicked.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  int id = item.getItemId();
  switch (id){
     case R.id.exit:
        // Your implementation
        return true;
  }
  return super.onOptionsItemSelected(item);
}
```

Menu

If the developer want create a menu in Java without XML file, he have to define the items of the menu in the onCreateOptionsMenu method.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

  MenuItem item = menu.add("Exit");
  item.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
     @Override
     public boolean onMenuItemClick(MenuItem item) {
        // Your implementation
        return true;
     }
  });

  return super.onCreateOptionsMenu(menu);
}
```

Also the click action is defined inside this method and the developer have to set the right listener: OnMenuItemClickListener

Dialog

A Dialog is a small window that prompts the user to make a decision or enter additional information.

A Dialog does not fill the screen and is normally used for modal events.

Two main dialogs exists natively in Android:

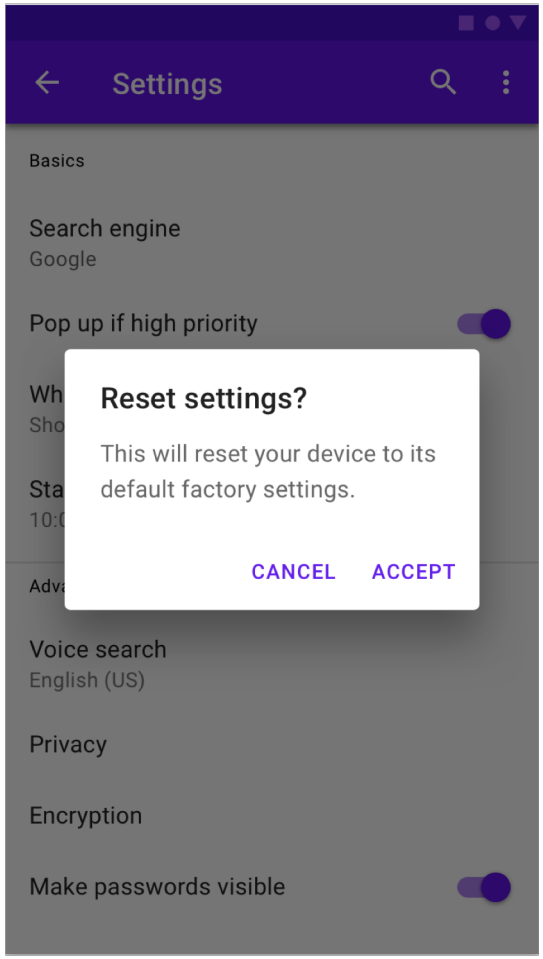
AlertDialog: using a specific Builder

DialogFragment: totally customizable

Dialog - AlertDialog

An AlertDialog is a dialog that can show a title, up to three buttons, a message, or a list of selectable items, or a custom layout.

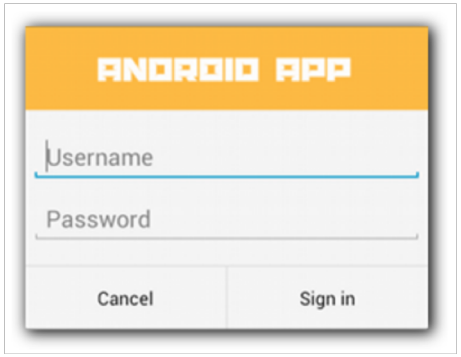
```
AlertDialog.Builder builder = new AlertDialog.Builder(YourActivity.this);
builder.setTitle("Title");
builder.setMessage("Message");
builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {

   }
});
builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {

   }
});
AlertDialog dialog = builder.create();
dialog.show();
```



Dialog - DialogFragment

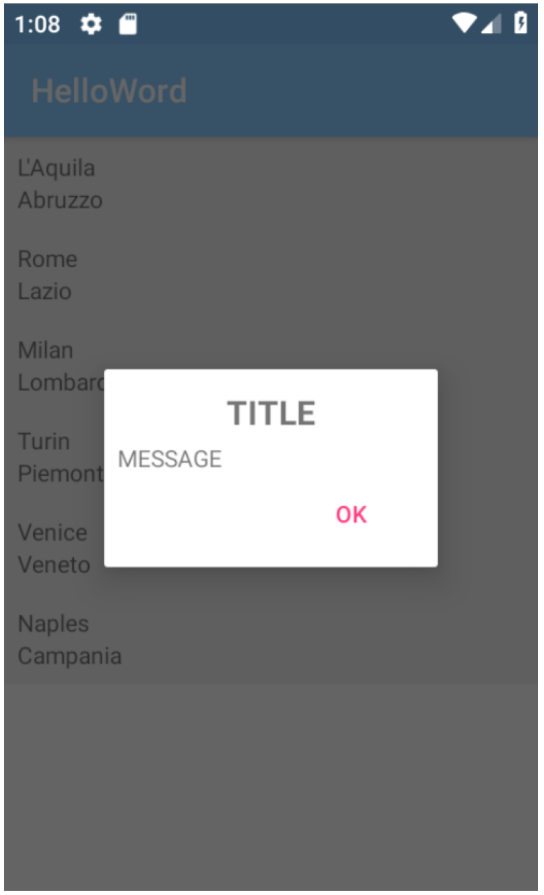
The DialogFragment class provides all the controls the developer needs to create his dialog and manage its appearance.

Using DialogFragment to manage the dialog ensures that it correctly handles lifecycle events such as when the user presses the Back button or rotate the screen.



Dialog - DialogFragment

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:minWidth="200dp"
  android:orientation="vertical"
  android:padding="8dp">

  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:gravity="center"
     android:padding="4dp"
     android:text="TITLE"
     android:textSize="20sp"
     android:textStyle="bold" />

  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="MESSAGE" />

  <Button
     style="@style/Base.Widget.AppCompat.Button.Borderless.Colored"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_gravity="right"
     android:text="OK" />

</LinearLayout>
```



Dialog - DialogFragment

The developer has to create a Java class, extending DialogFragment and overriding the method onCreateView and onCreateDialog

```
public class MyDialog extends DialogFragment {

  @Nullable
  @Override
  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
      return inflater.inflate(R.layout.dialog, container, false);
  }

  @NonNull
  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {

      Dialog dialog = super.onCreateDialog(savedInstanceState);
      return dialog;
  }
}
```

Dialog - DialogFragment

To show the dialog in the Activity or in the Fragment, the developer has to create an instance of his dialog class and has to call the method show().

```
MyDialog dialog = new MyDialog();
dialog.show(getSupportFragmentManager(), "MyDialog");
```

In show method one of the inputs is the FragmentManager because we use the DialogFragment, in detail the support version of it.

The developer has to use always the support version, not only for the FragmentManager but for the DialogFragment, too.

The second input is a String that reveals the name or tag of the Dialog.

Dialog - DialogFragment

If the developer want create a Progress dialog, he has to use the DialogFragment and create a custom layout with Progress view.

The ProgressDialog class is deprecated.