XML for GUIs

# Introduction:

In the android framework there are two ways of defining the layout for Graphical User Interfaces, we can either programmatically manipulate the properties of objects or use XML in order to format out GUI according to our preferences. In this section, the focus will be on XML and how GUI builder is used in order to generate it.

# Using XML to set the layout:

The main advantage of choosing to use XML over formatting the GUI programmatically is that we still have the ability to modify the layout and state of objects at run time, yet we separate the main code from the UI layout.

Declaring the UI in XML also means that the code can be reused or with slight changes multiple views can be created in order to suit different devices or screen resolutions in order to achieve the best possible optimization. XML driven UI’s are also much easier to edit and format as they allow the user to visualize the structure and make necessary changes using a graphical interface which generates the appropriate cod. It also provides the ability to replicate properties of a single object so that other objects so that consistency can be easily achieved amongst them when designing the UI.

The XML code is generated through the use of the inbuilt GUI builder which allows the user to drag and drop objects onto the View in order to create a GUI. Objects can easily be positioned and moved around whilst setting up various preferences is only click away and can be set up very easily. The XML code is automatically generated for the user so that no coding is required in order to set up the GUI. Therefore it’s very easy to create, modify and reuse the View specification which has been constructed using the UI builder to generate XML code.

# View Group Layouts:

The android framework comes with a series of layouts and views which are prewritten, they come as a subclass of ViewGroup, and some of the main ones include:

* **Linear Layout:**

This layout organises the children elements in vertical rows

* **Relative Layout:**

This layout organises the children elements in relative position to other elements or in relation to the container that it resides in

* **Absolute Layout:**

This layout organises the children in exact positions on the view, the positions are defined by X and Y coordinates of the object

* **List View:**

This view groups the children and allows them to be viewed in the scrollable list

* **Grid View:**

This view groups the children and allows them to be displayed in a scrollable grid

# ID:

Each object can be uniquely identified with by an ID which can be assigned to it. IDs are strings, but when the program gets compiled they get changed into integers.   
The ID can be used in order to refer to individual objects in order to manipulate them programmatically after we create an instance of them. We refer to object through the use of findViewById() method.

In order to declare a new ID we have to use a “+” sign, which basically means it will be added to the resources directory, “R.java” by default. After that they can just be referenced without the “+” sign.

**Declaring and adding a new object in XML:**  
 *android:id="@+id/my\_button"*

**Creating an instance:**  
*Button myButton = (Button) findViewById(R.id.my\_button);*

# Attributes:

Attributes are used to define properties of different objects; they are different for various objects. For instance, we can create a button object, in order to define its properties we use attributes such as “*text*”, ”*layout\_width*” or “*layout\_height*”.  
“*id*” is an attribute that is available to all kinds of objects.

Before we refer to an attribute we use the “*android:*” tag to refer to the library.  
An example of an object reference with attributes:  
*<Button  
 android:id="@+id/button\_id"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="wrap\_content"  
 android:text="@string/self\_destruct"   
/>*