Macoratti.net Xamarin Android - Apresentando e usando Layouts - LinearLayout

Neste artigo vou apresentar uma alguns conceitos relacionados com **Views** e **Layouts** usados na plataforma Android usando o VS 2015 Community e a linguagem C#.

Tudo o que você vê em um aplicativo Android é uma **View** : botões, etiquetas, caixas de texto e botões de rádio são todos exemplos de **views**.

As Views e as ViewsGroups

As **Views** são organizadas em uma hierarquia utilizando diversos tipos de **ViewGroups**. Uma **ViewGroup** é um tipo especial de **View** que é usado para organizar (*layout*) as outras views na tela.

As **Views e as ViewGroups** podem ser criadas usando dois métodos diferentes: **programaticamente ou declarativamente.**

- Ao usar uma abordagem **programática**, o desenvolvedor faz chamadas de API para criar e posicionar cada **View** individual na interface.
- Ao usar uma abordagem **declarativa**, o desenvolvedor cria arquivos de layout XML que especificam como as **Views** devem organizadas.

A abordagem declarativa possui a seguintes vantagens:

• Proporciona uma melhor separação do design visual de um aplicativo da sua lógica de processamento;
• Permite que vários layouts sejam criados para apoiar vários dispositivos ou configurações de dispositivos com uma única base de código;
• Ferramentas de desenvolvimento, tais como Android Studio, Xamarin Studio e o plugin Android para Eclipse, permitem visualizar a interface do usuário, sem a necessidade de compilar e executar o aplicativo após cada alteração;

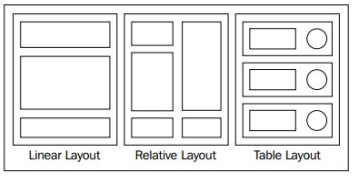
Embora a abordagem declarativa seja a mais indicada às vezes uma combinação das abordagens se faz necessária.

Interface do Usuário (Views, Widgets) e Layouts

O Android oferece um conjunto abrangente de ***widgets*** de interface que podem ser usados para construir uma rica experiência do usuário. Todos estes elementos são subtipos de views e podem ser organizados em layouts sofisticados que utilizam vários tipos de **ViewGroups**. Todos os **widgets** de interface podem ser encontrados no pacote **android.widget** no Application Framework.

O Application Framework possui um número ode subclasses de **ViewGroup**, cada uma fornece uma única forma de organizar o conteúdo da interface.

Os layouts mais comuns podem ser vistos na figura abaixo:



Cada layout pode ser usado para um propósito especifico.

Layout	Descrição	Cenário
Linear Layout	Organize seus '*filhos*' em uma única linha horizontal ou vertical e cria uma barra de rolagem quando necessário.	Indicado quando os widgets fluem horizontal ou verticalmente
Relative Layout	Organiza os objetos '*filhos*' relativamente uns aos outros ou e relação ao pai.	Indicado quando as posições dos widgets podem ser melhor descritas em relação a outro elemento (à esquerda) ou à área de fronteira do pai (lado direito, ou centrado)
Table Layout	Organiza os seus '*filhos*' em linhas e colunas	Indicado quando as posições dos widgets naturalmente se encaixam em linhas e colunas.

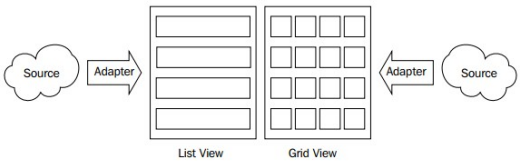
Para layouts que são orientados a uma fonte de dados dinâmica o Application Framework possui um conjunto de classes derivada de **AdapterView.**

Nota: Um AdapterView é uma view cujos filhos são determinados por um Adapter que atua como uma ponte entre a view e os dados relacionados.

Dentre esse layouts os mais comuns são :

- **List View -** Organiza o conteúdo da fonte de dados em uma lista de coluna única com rolagem.
- **Grid View** - Organiza o conteúdo da fonte de dados em uma grade de colunas e linhas

Conforme mostra a figura a seguir:



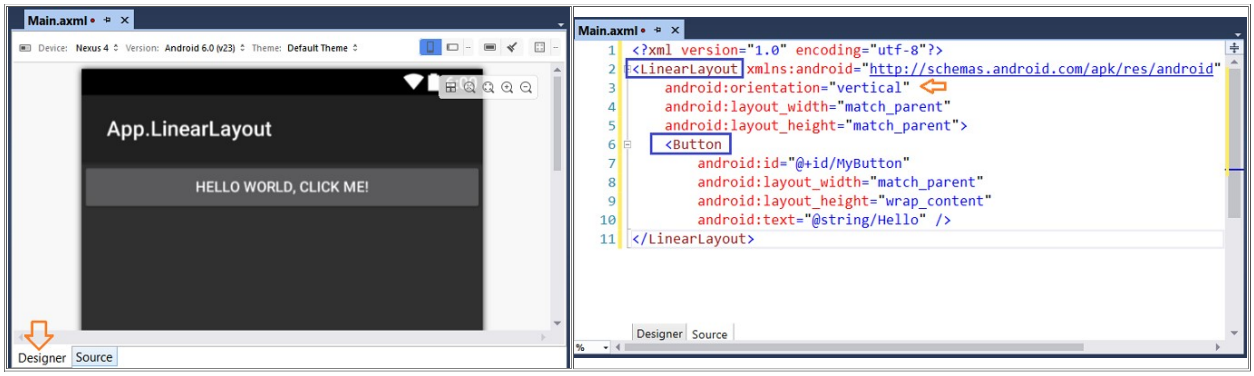
Dessa forma, na plataforma Android, um Layout é um tipo **ViewGroup** que atua como um container e pode conter outras views que passam a ser **filhas do Layout.**

Vamos começar abordando o Linear Layout.

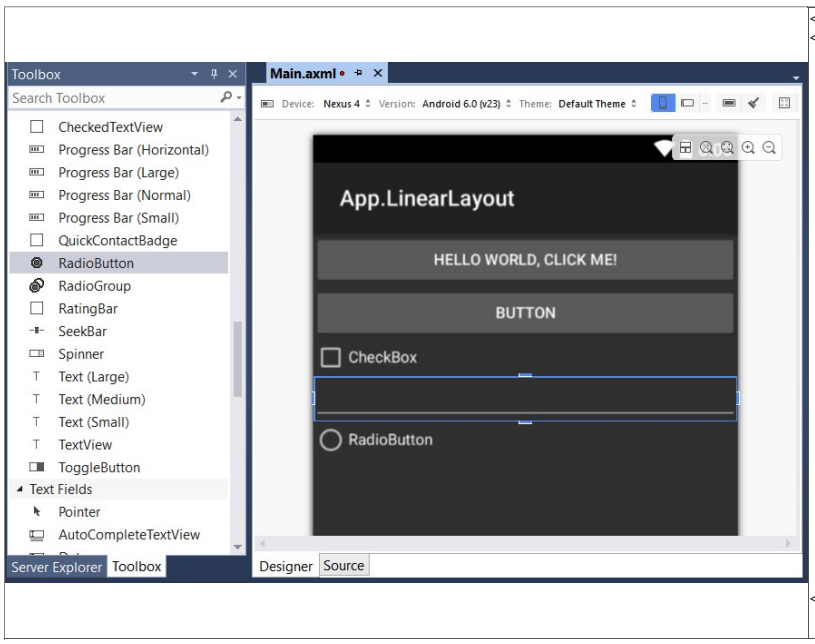
Linear Layout

Um Linear Layout é usado para organizar Views em uma interface de usuário (UI) de forma a serem exibidas na horizontal ou na vertical.

Quando você cria uma aplicação Android vazia no Visual Studio usando a opção **Visual C# -> Android -> Blank App (Android)**, ao abrir o arquivo **Main.axml** na pasta **Resources\layout**, você vai notar que é criado uma interface que utiliza o **Linear Layout** com orientação vertical contendo um **Button:**



Assim qualquer outro componente/view que for incluído no **LinearLayout** será uma view filha desse Layout e estará posicionado **verticalmente**:

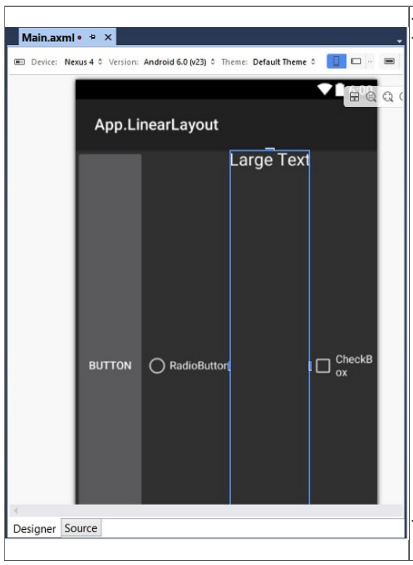



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    android:id="@+id/MyButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Hello" />
  <Button
    android:text="Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/button1" />
  <CheckBox
    android:text="CheckBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/checkBox1" />
  <EditText
    android:inputType="textEmailAddress"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText1" />
  <RadioButton
    android:text="RadioButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/radioButton1" />
</LinearLayout>
```

Observe que o código XML possui o nó XML <**LinearLayout**>, a **ViewGroup**, como o raiz, e 2 Buttons, 1 CheckBox, 1 EditText e 1 RadioButton, todas Views, colocadas no interior de **LinearLayout**.

A propriedade mais importante para o LinearLayout é **"android:orientation"** que no exemplo esta definida como '**vertical**'. Isso define o fluxo linear do layout de forma que os componentes/views são dispostos verticalmente na interface.

Se você pode alterar a orientação para horizontal definindo a propriedade **"android:orientation="horizontal",** e, após isso ao incluir views no Layout elas serão dispostas horizontalmente conforme mostra a figura abaixo:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    android:text="Button"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/button1" />
  <RadioButton
    android:text="RadioButton"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/radioButton1" />
  <TextView
    android:text="Large Text"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView1" />
  <CheckBox
    android:text="CheckBox"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/checkBox1" />
</LinearLayout>
```

Note que o Layout possui as propriedades **layout_width** e **layout_height** que são usadas para especificar a largura e altura do layout (ViewGroup).

Podemos especificar os valores para a **largura e altura** usando uma dos seguintes valores pré-definidos:

- **dp ou dpi** - density-independent pixels - baseado na densidade física da tela;

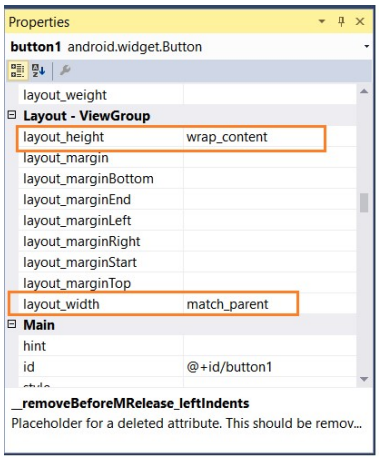
- **sp** - scale-independent pixels - recomendado para usar com fontes;
- **in** - Inches (polegadas) - baseado no tamanho físico da tela;
- **mm** - milímetros - baseado no tamanho físico da tela;
- **px** - pixels - corresponde aos pixels atuais na tela;
- **pt** - points - 1/72 da polegada baseado no tamanho físico da tela;

Existem também valores constantes que você pode usar para definir as propriedades **layout_width** e **layout_height** :

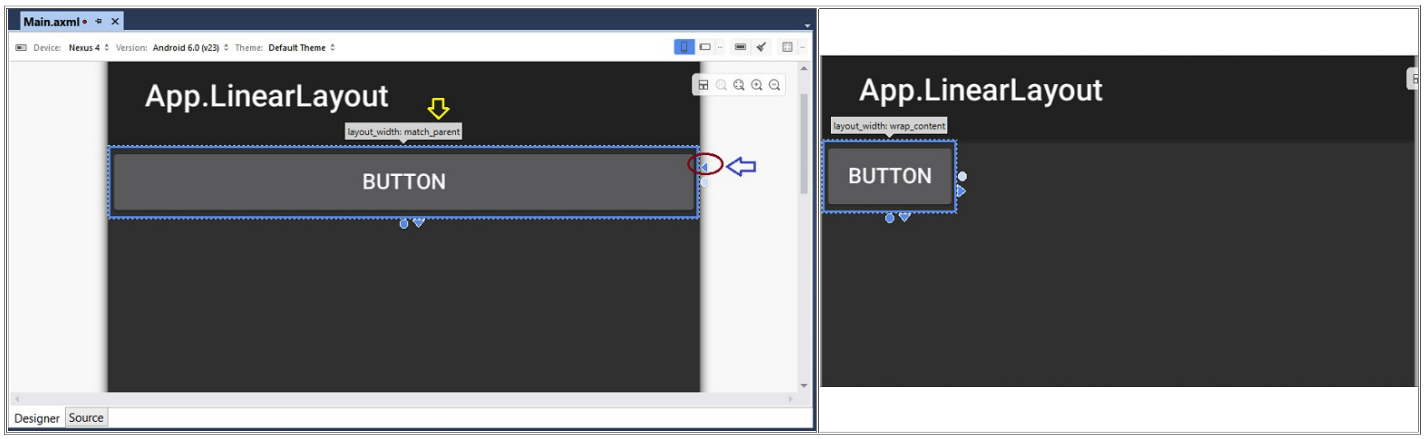
- **wrap_content** - informa ao componente/view para ocupar o espaço que ele vai necessitar *(altura e/ou largura)* para exibir suas informações no layout.
- **match_parent** - informa ao componente/view para ocupar o mesmo espaço da sua view pai, ou seja, ele vai preencher todo o conteúdo de seu layout pai.

Nota : *Você também pode encontrar o atributo **fill_parent** em versões anteriores á versão 2.3. Esse atributo atua da mesma forma que **match_parent**.*

Podemos definir os valores para essas propriedades na janela de propriedades do componente/view :



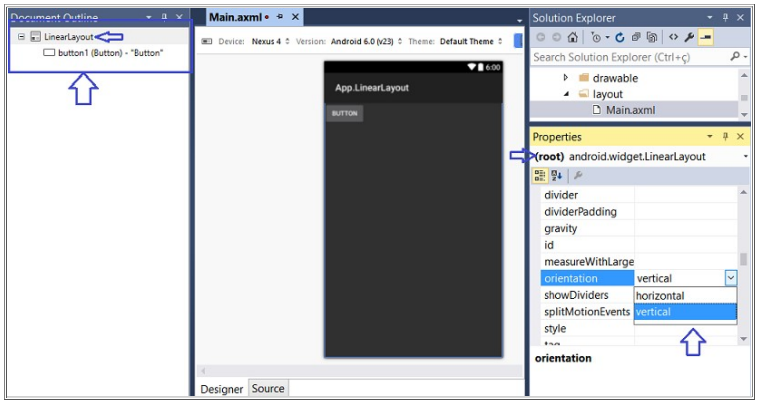
Outra forma de alterar o valor da largura é, estando no modo Designer, clicar no triângulo no lado direito do *widget* para alternar sua largura para **wrap_content**:



Para retornar para **match_parent** basta clicar novamente no triângulo lateral.

Para ter uma visão global do layout clique no menu **View -> Other Windows -> Document Outline**

Selecione a raiz **LinearLayout** e veja na janela de propriedades que podemos alterar o valor da propriedade **orientation :**



Outra propriedade muito importante é **layout_weight** .

Este atributo atribui um valor de "*importância*" o 'peso' para uma view em termos de quanto espaço ela deve ocupar na tela. Um valor maior permite que ela se expanda para preencher qualquer espaço restante na view principal. Views filhas podem especificar um valor para **weight** e, em seguida, todo o espaço restante no **ViewGroup** será atribuído ás filhas na proporção do seu peso declarado.

O valor padrão de **layout_weight** é zero.

Aguarde mais artigos sobre o desenvolvimento de aplicativos Android usando o Visual Studio e o Xamarin.

(*Disse Jesus aos fariseus*) Hipócritas, bem profetizou Isaías a vosso respeito, dizendo:
Este povo se aproxima de mim com a sua boca e me honra com os seus lábios, mas o seu coração está longe de mim.
Mas, em vão me adoram, ensinando doutrinas que são preceitos dos homens.
Mateus 15:7-9

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net

---

José Carlos Macoratti