Creating the App and First View

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

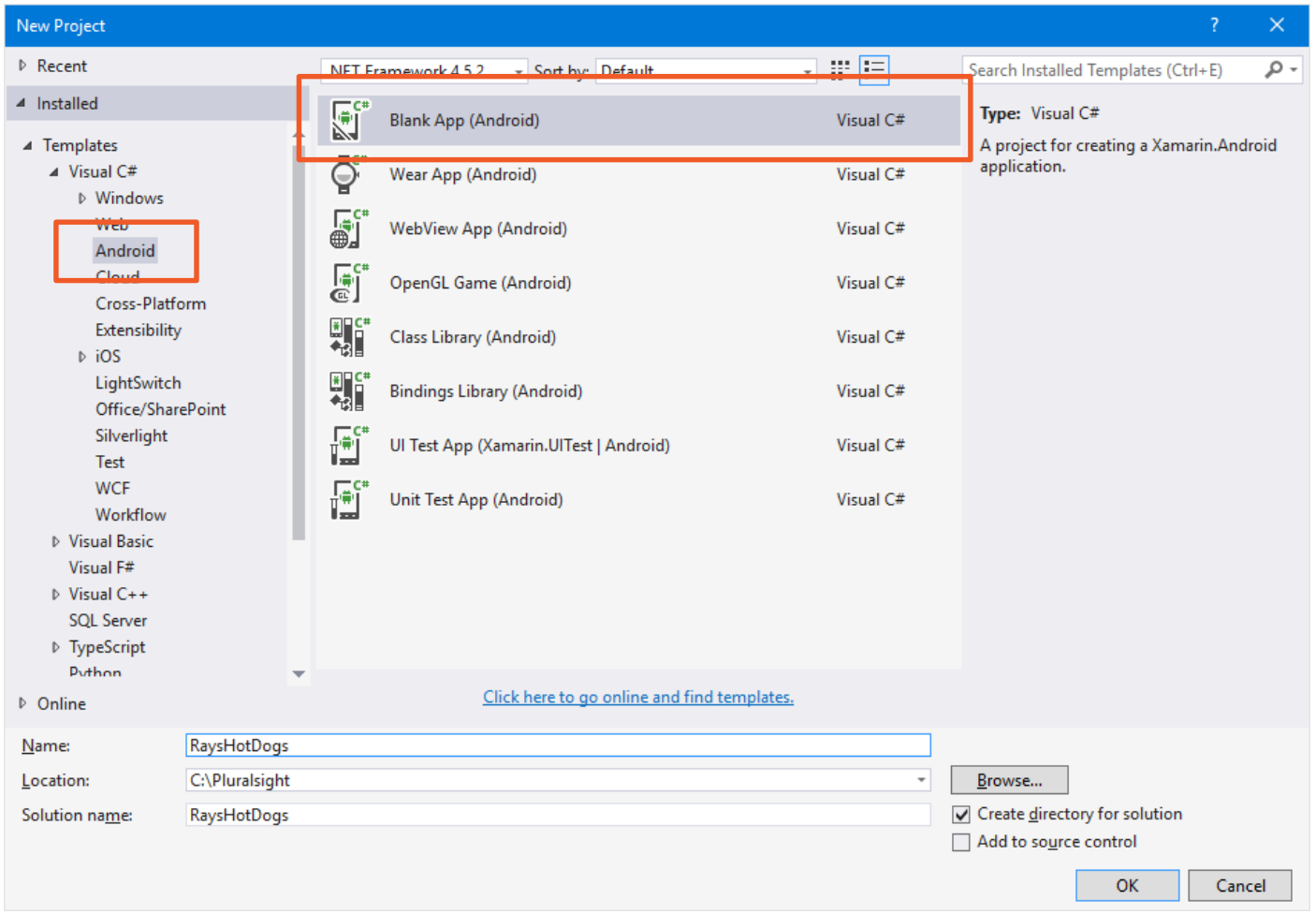
Agenda

Xamarin.Android fundamentals

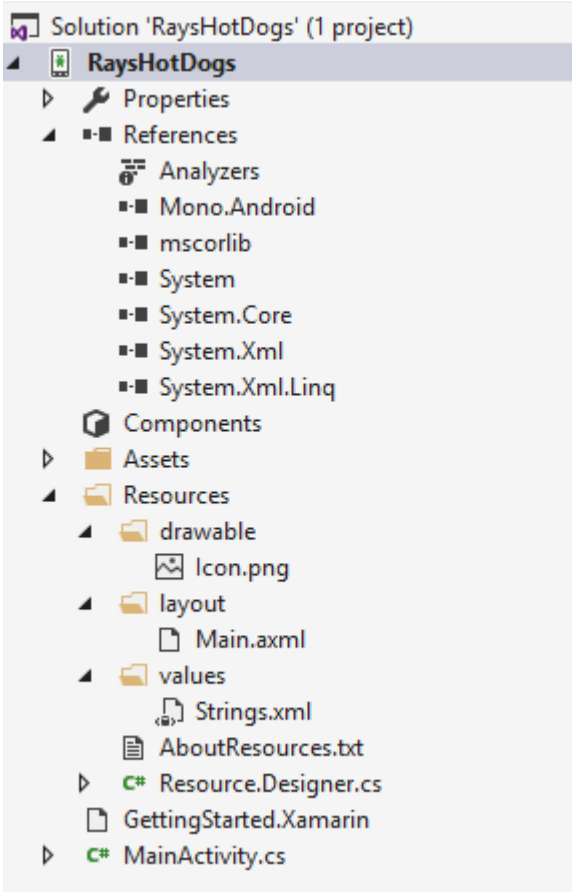
Creating our first view

Xamarin.Android Fundamentals

Available Project Templates

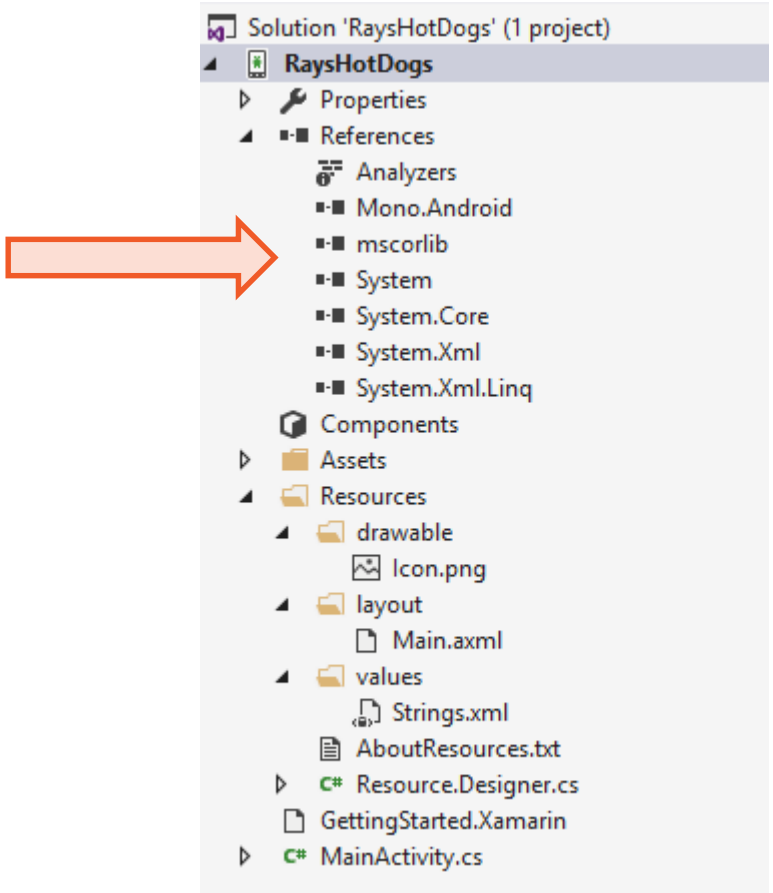


File → New Project



References

References are created to some System assemblies
as well as to Mono.Android

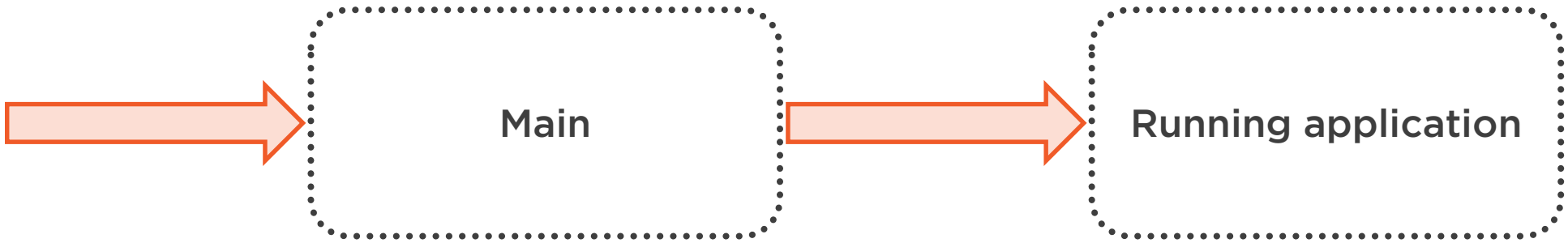


Base Concepts in Android

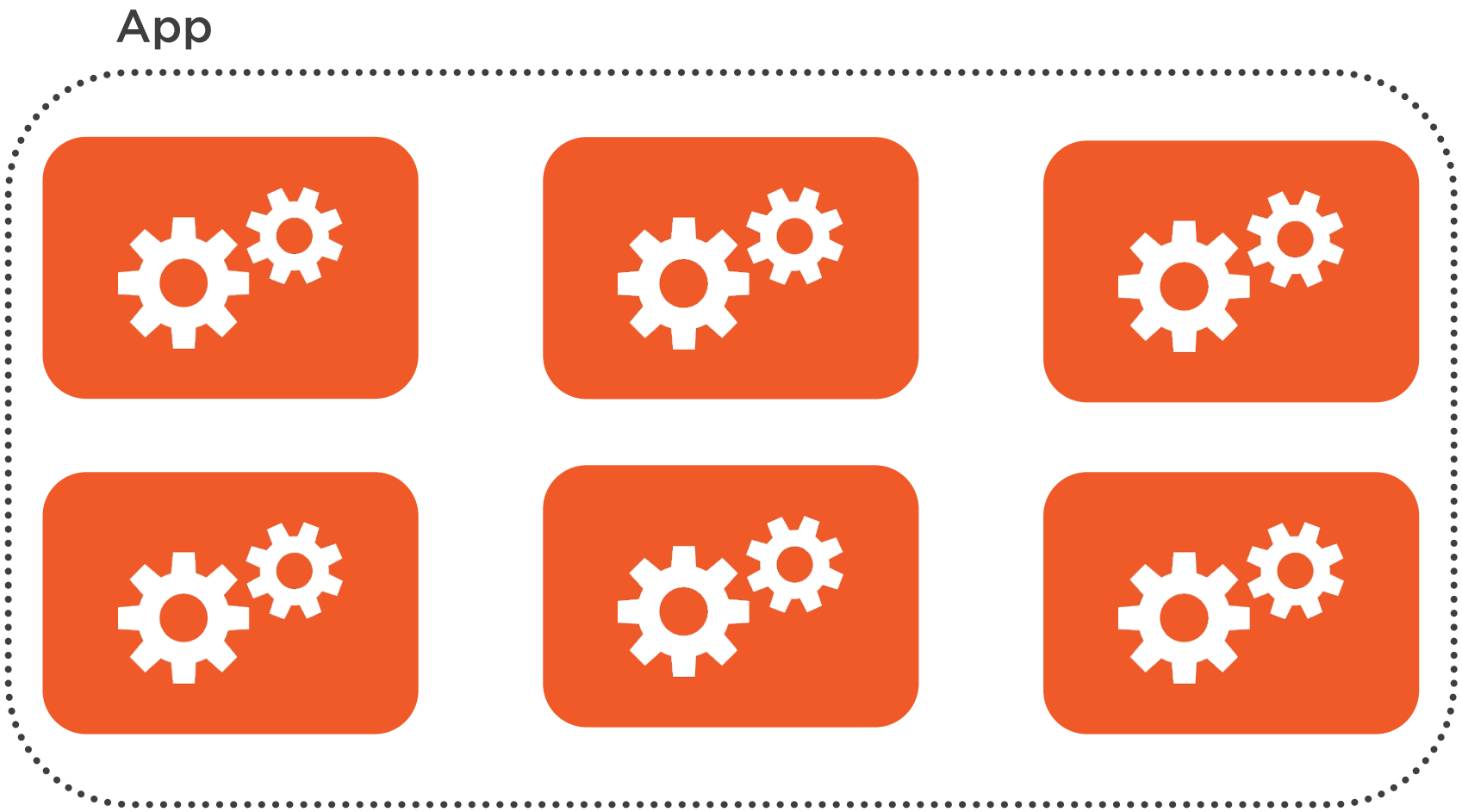
Activities

Views

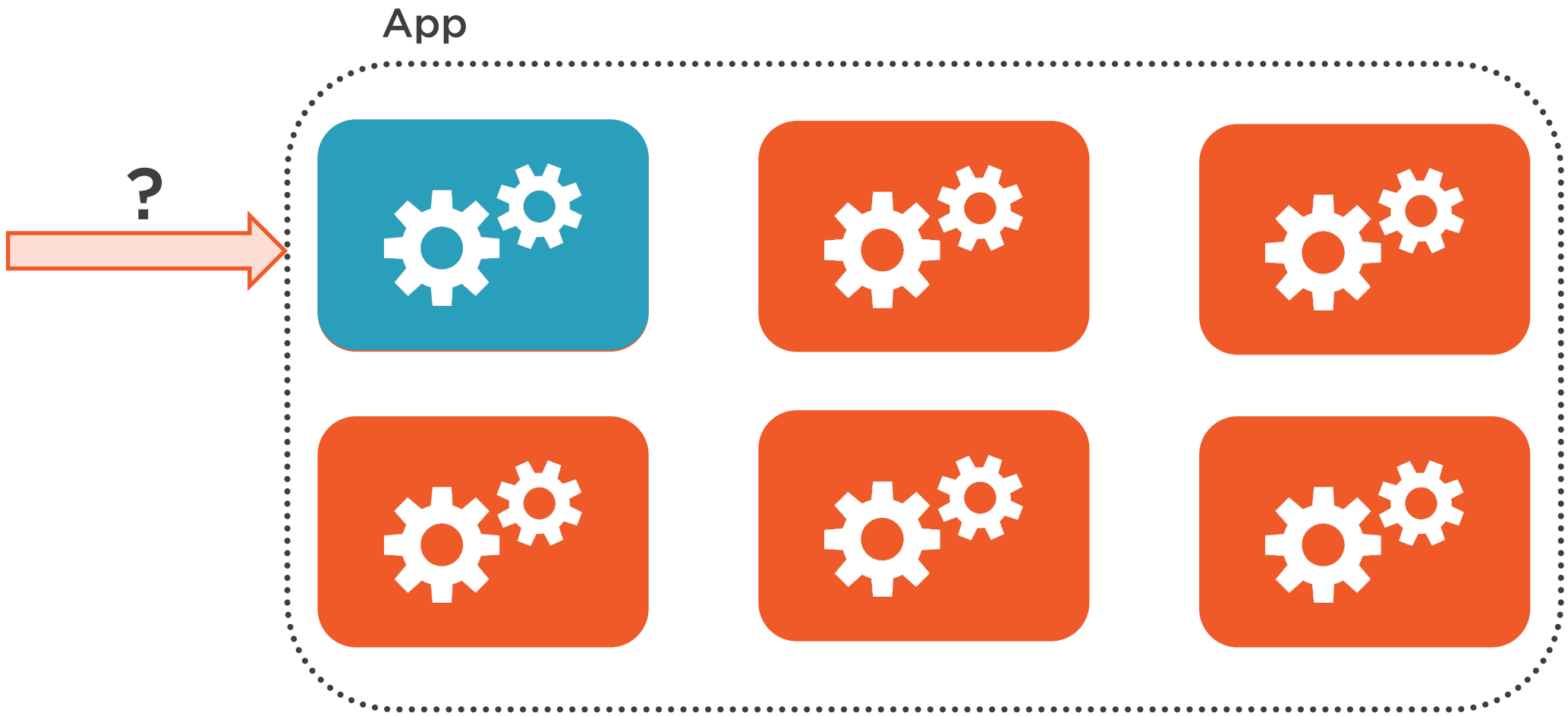
The Typical “Main” Method



Activities



Starting an App



Activities

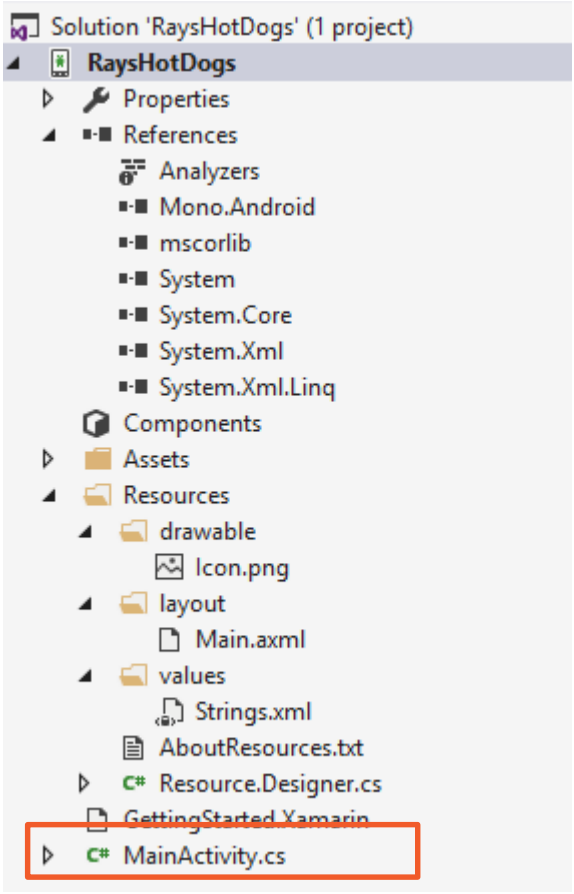
All activities inherit from base Activity

Functionality for a single task

Corresponds to one screen

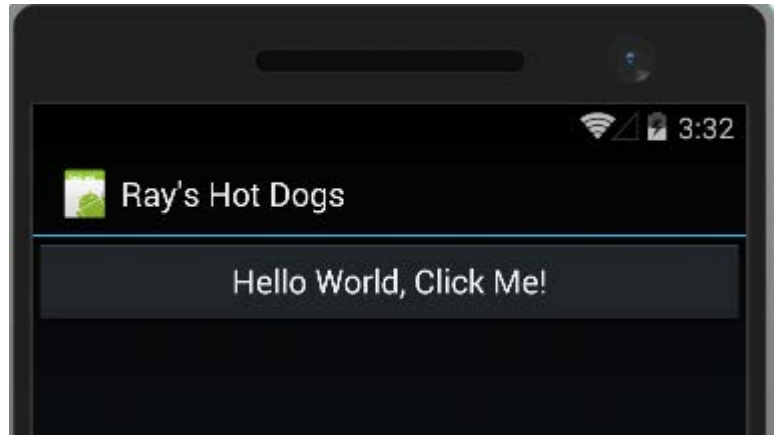
Specific lifecycle

The Default Activity

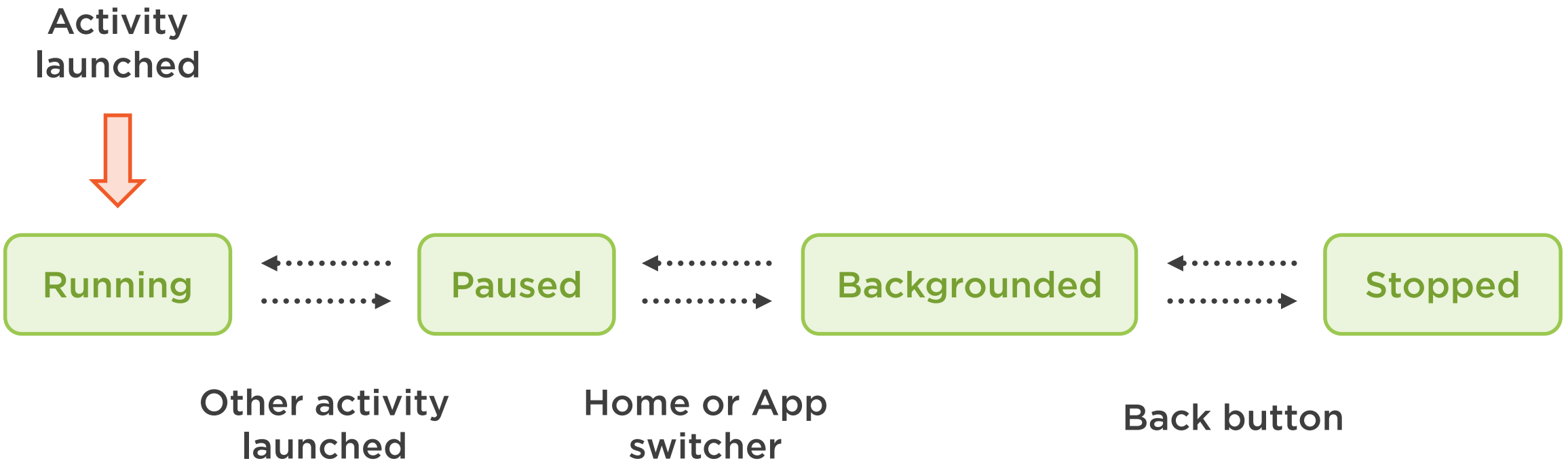


```
[Activity(Label = "Ray's Hot Dogs",
          MainLauncher = true,
          Icon = "@drawable/icon")]
public class MainActivity : Activity
{
    ...
}
```

A Sample Activity



Activity States




Activity Starts
onCreate()
onStart()
onRestart()
onResume()
Activity Running
Process Killed
NEW ACTIVITY STARTED
onFreeze()
onPause()
USER NAVIGATES BACK
TO YOUR ACTIVITY
OTHER APPLICATIONS
NEED MEMORY
ACTIVITY COMES
TO FOREGROUND
ACTIVITY COMES
TO FOREGROUND

Important Lifecycle Methods

OnCreate

OnStart

OnResume

OnPause

OnStop

OnCreate

```
[Activity(Label = "RaysHotDogs", MainLauncher = true, Icon = "@drawable/icon")]
public class MainActivity : Activity
{
    protected override void OnCreate(Bundle bundle)
    {

        base.OnCreate(bundle);

        // Set our view from the "main" layout resource
        SetContentView(Resource.Layout.Main);

    }

}
```

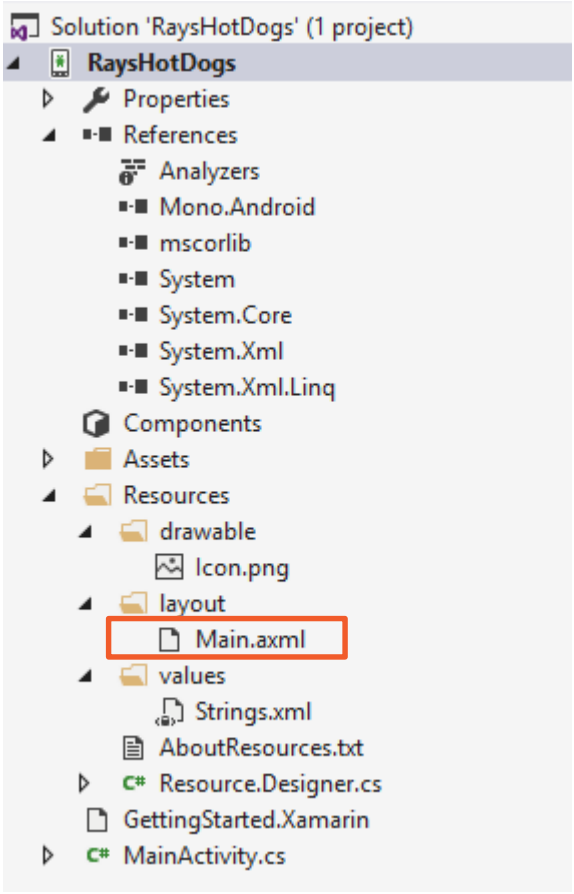
Views

Visual part

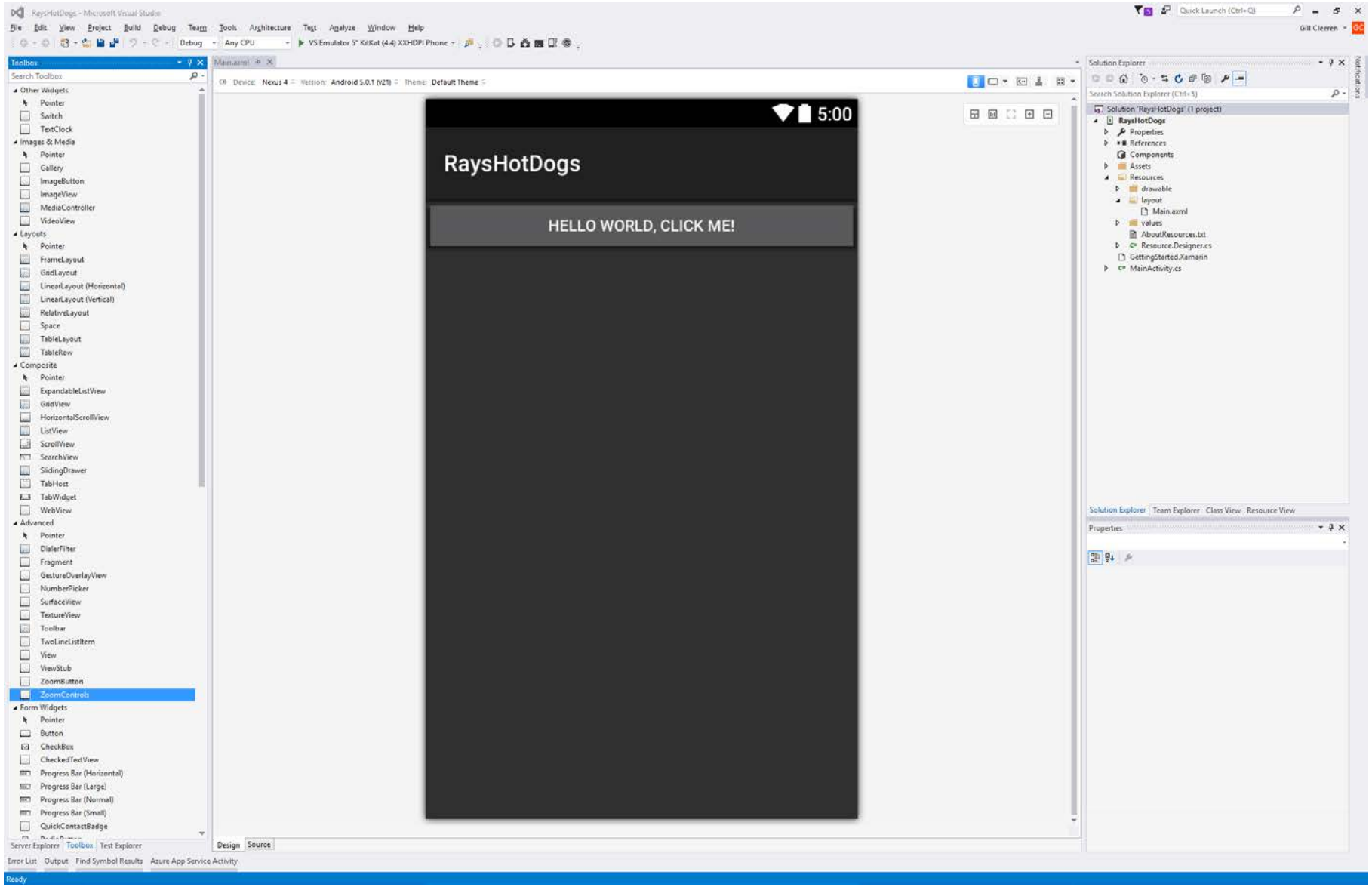
*.axml

Live in Resources\Layout

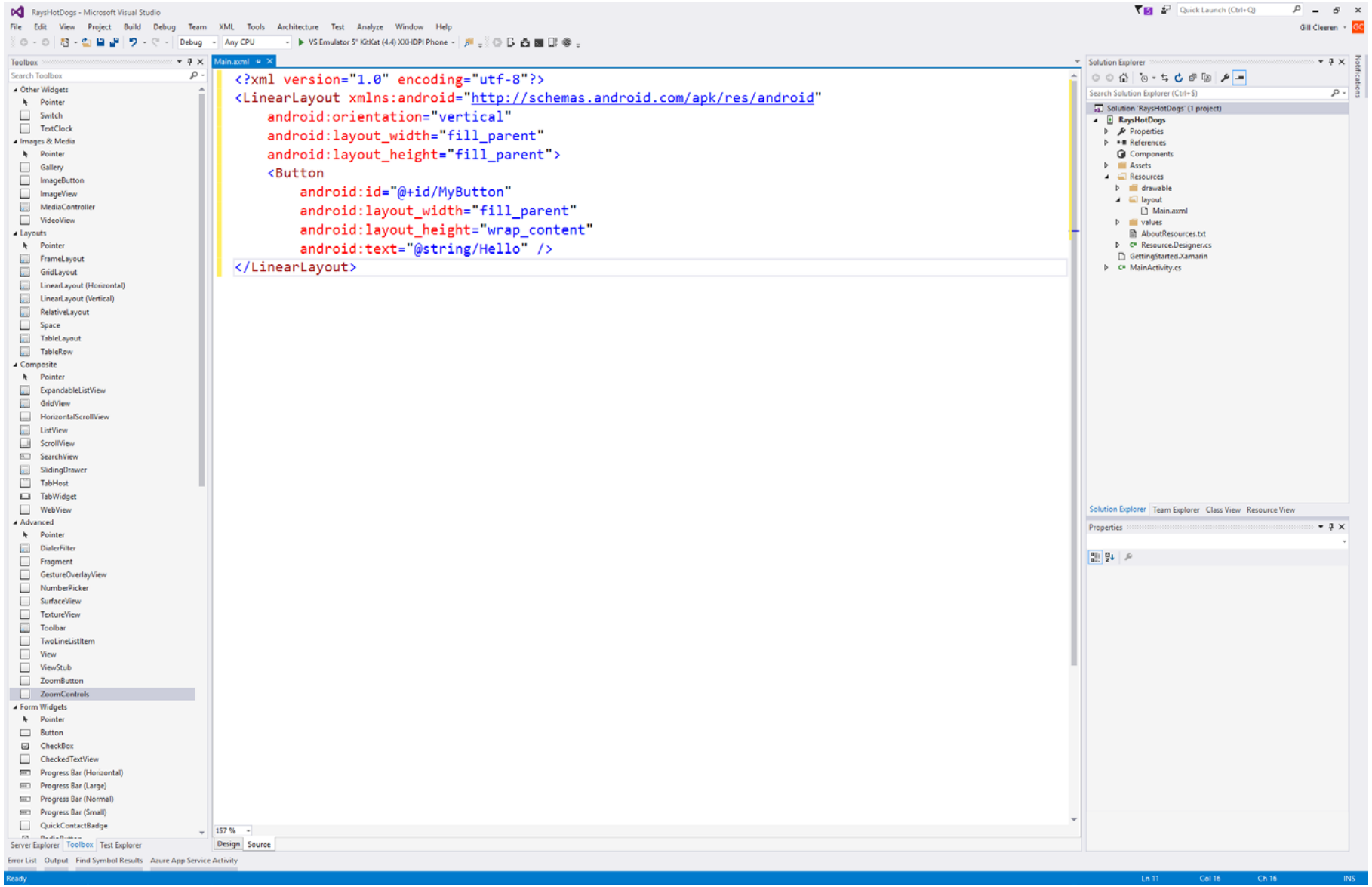
The Default View



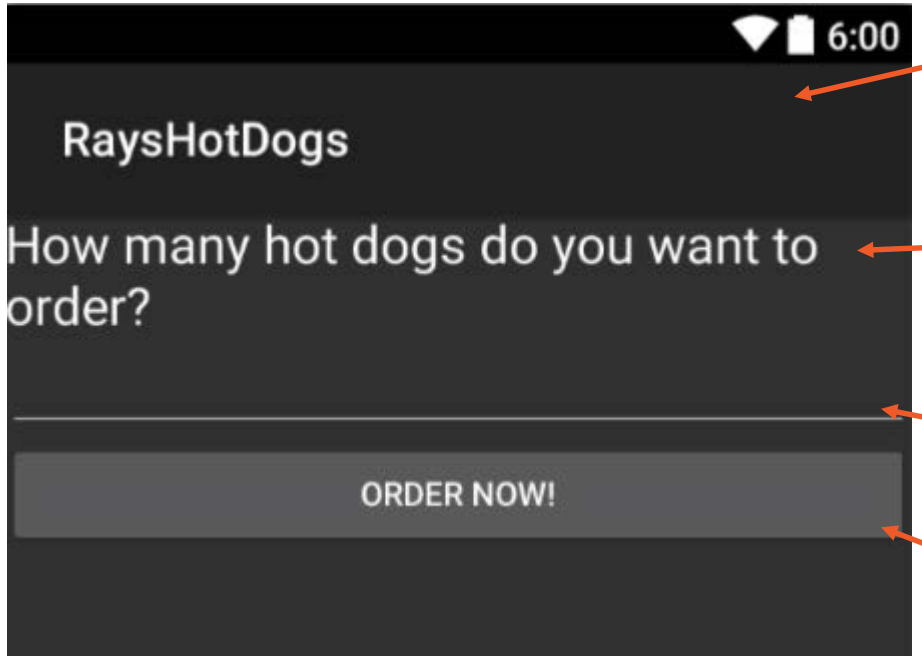
The Designer in Visual Studio



The Source Code Editor



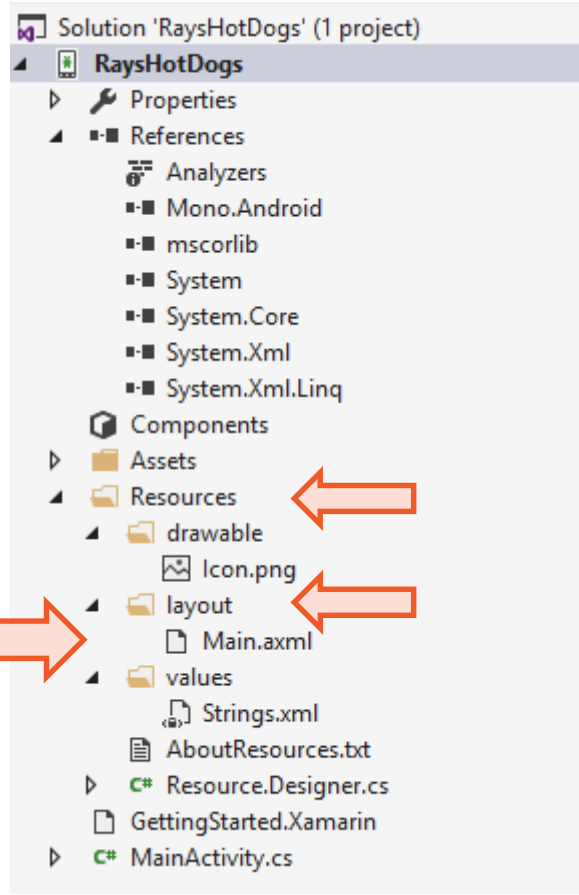
Sample View Code




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:text="How many hot dogs do you want to order?"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/AmountTextView" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/amountEditText" />
    <Button
        android:id="@+id/OrderButton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Order now!" />
</LinearLayout>
```

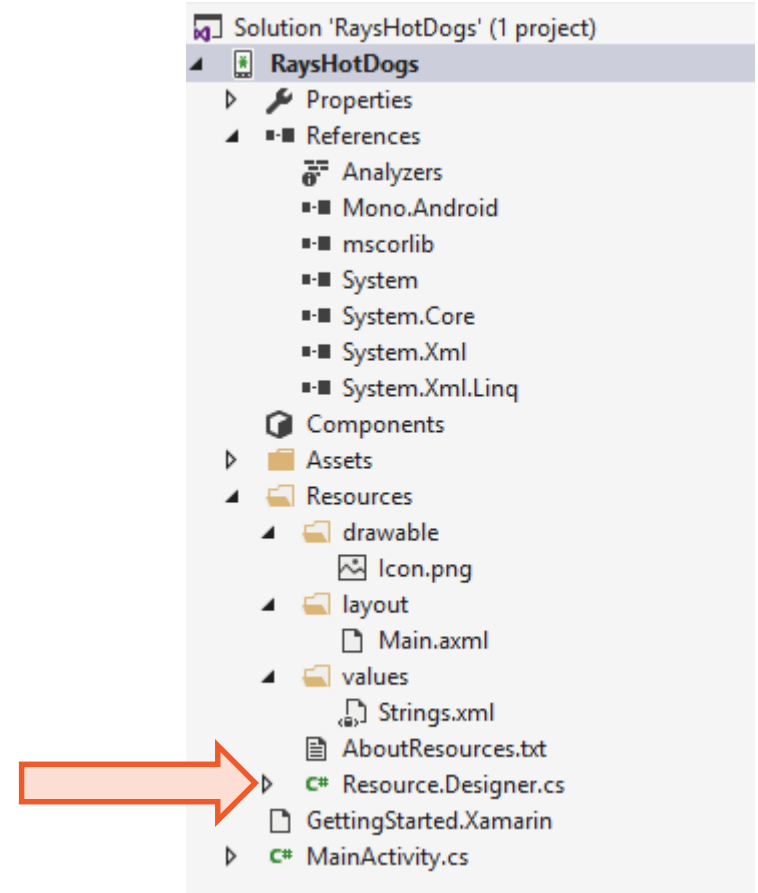
Linking the View and the Activity

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    SetContentView(Resource.Layout.Main);
}
```



The Magic: Resource.Designer.cs

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("Xamarin.Android.Build.Tasks", "1.0.0.0")]
2 references
public partial class Resource
{

    0 references
    static Resource()...

    0 references
    public static void UpdateIdValues()...

    2 references
    public partial class Attribute...

    2 references
    public partial class Drawable...

    2 references
    public partial class Id...

    3 references
    public partial class Layout
    {

        // aapt resource value: 0x7f030000
        public const int Main = 2130903040;

        0 references
        static Layout()...

        0 references
        private Layout()
        {
        }
    }

    2 references
    public partial class String...
}
```



```
public partial class Layout
{
  // aapt resource value: 0x7f030000
  public const int Main = 2130903040;

  static Layout()
  {
    global::Android.Runtime.ResourceIdManager.UpdateIdValues();
  }

}
```

The “Magic”: Resource.Designer.cs

Accessing Controls from Code

```
<Button
    android:id="@+id/MyButton"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/Hello"
    />
```

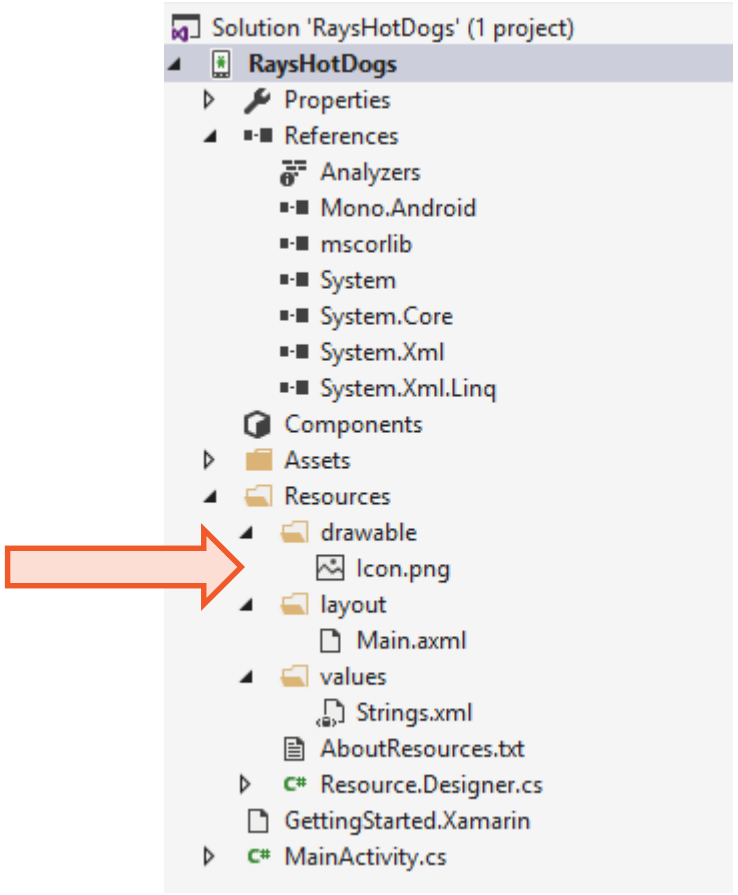
```
public partial class Resource
{
  public partial class Id
  {
    public const int MyButton = 2131034112;

    static Id()
    {
      global::Android.Runtime.ResourceIdManager.UpdateIdValues();
    }
  }
}
```

Accessing Controls from Code

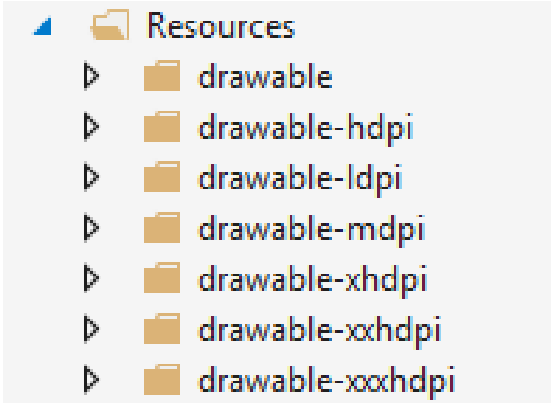
```
protected override void OnCreate(Bundle bundle)
{

    base.OnCreate(bundle);

    // Set our view from the "main" layout resource
    SetContentView(Resource.Layout.Main);

    // Get our button from the layout resource,
    // and attach an event to it

    Button button = FindViewById<Button>(Resource.Id.MyButton);
    button.Click +=
      delegate { button.Text = string.Format("{0} clicks!", count++); };

}
```

Drawables

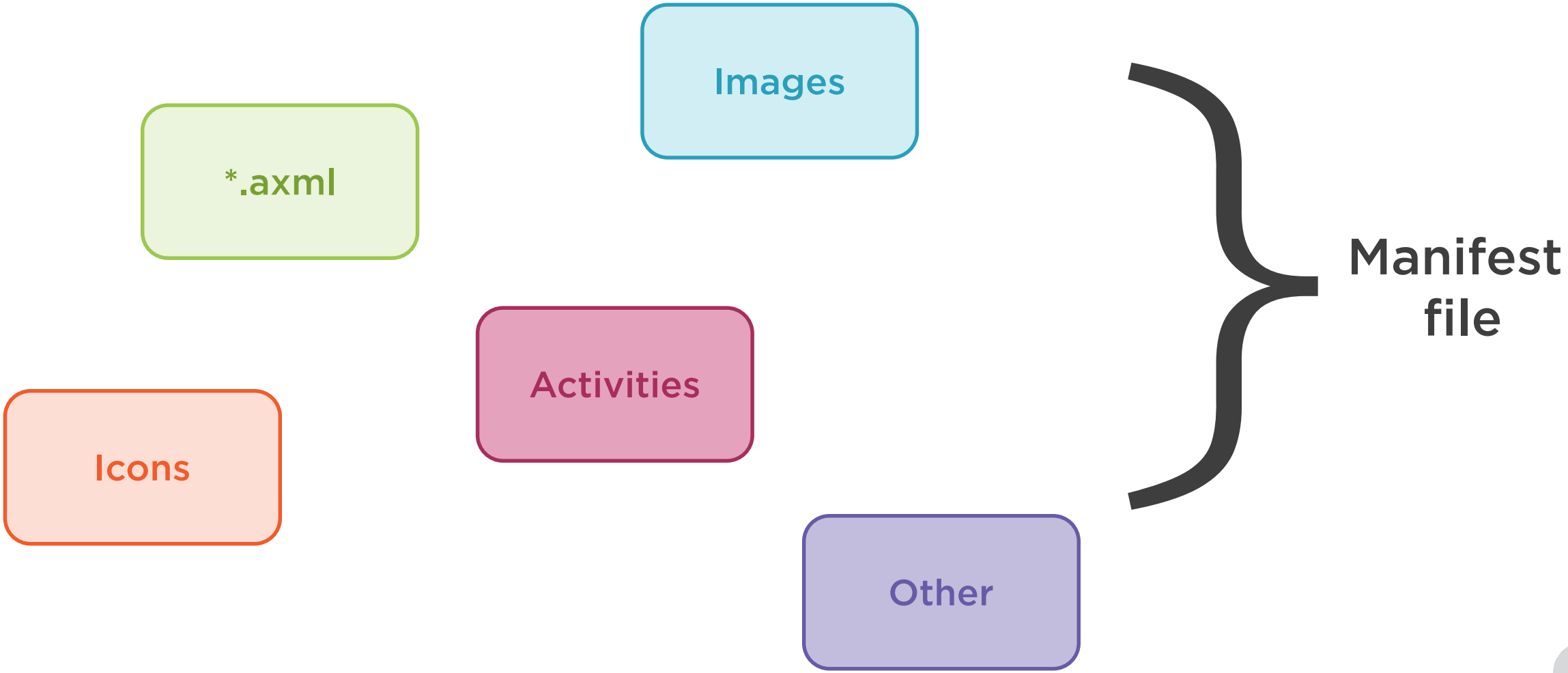
```
[Activity(Label = "RaysHotDogs",
          MainLauncher = true,
          Icon = "@drawable/icon")]
public class MainActivity : Activity
{
    ...
}
```



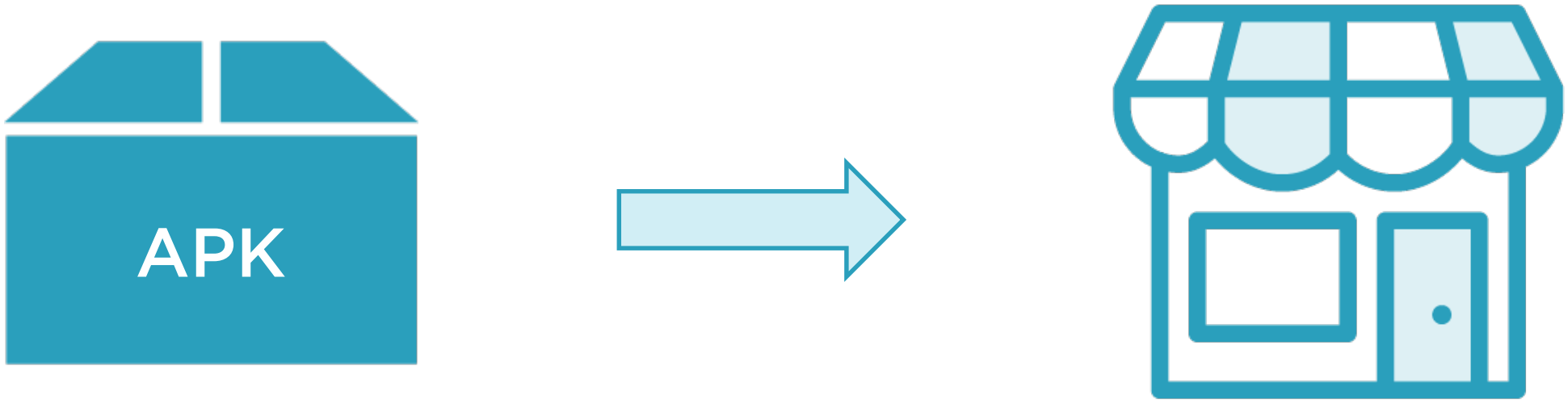
Supporting Many Resolutions



Application Manifest



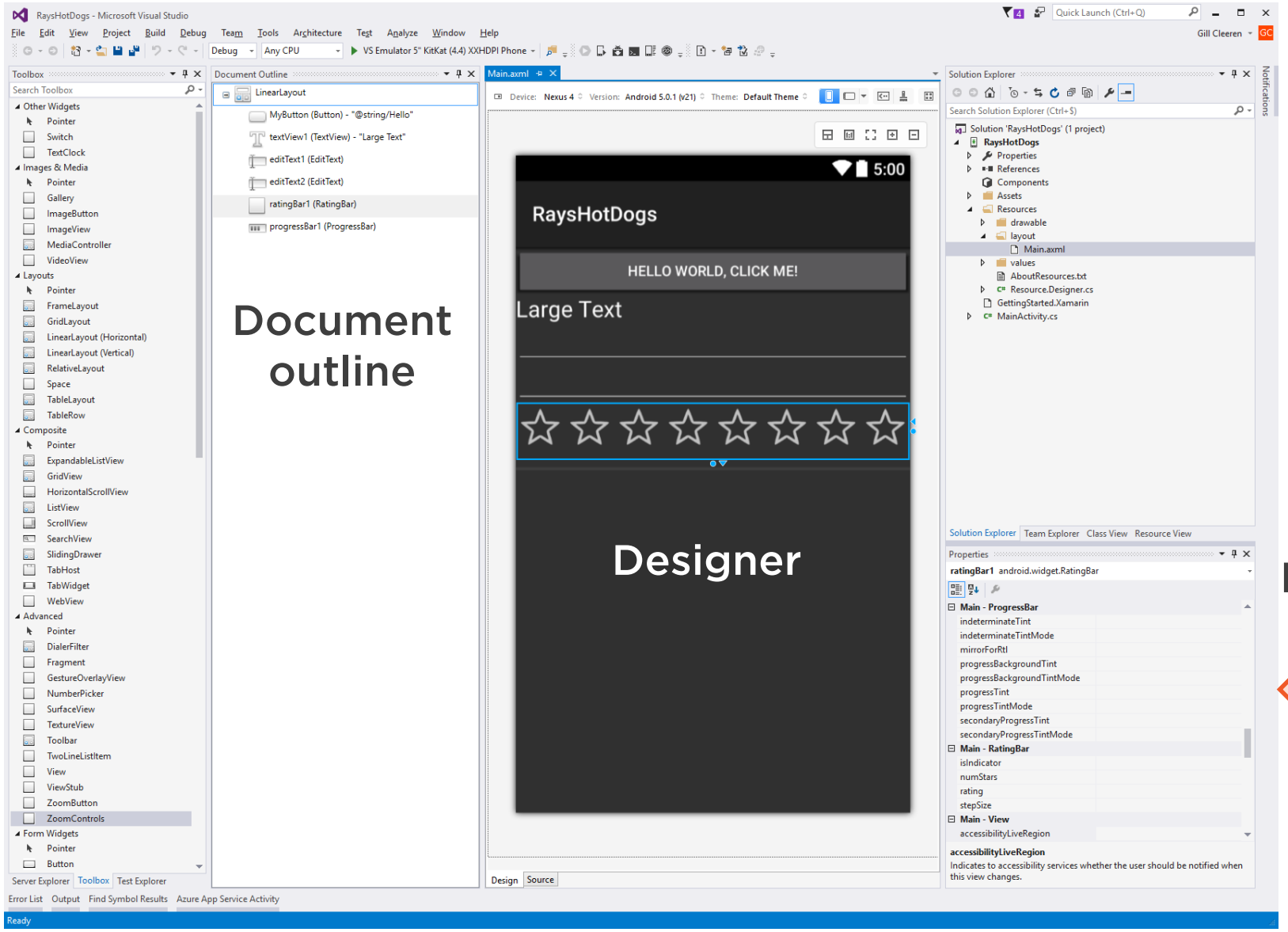
Compilation & Deployment



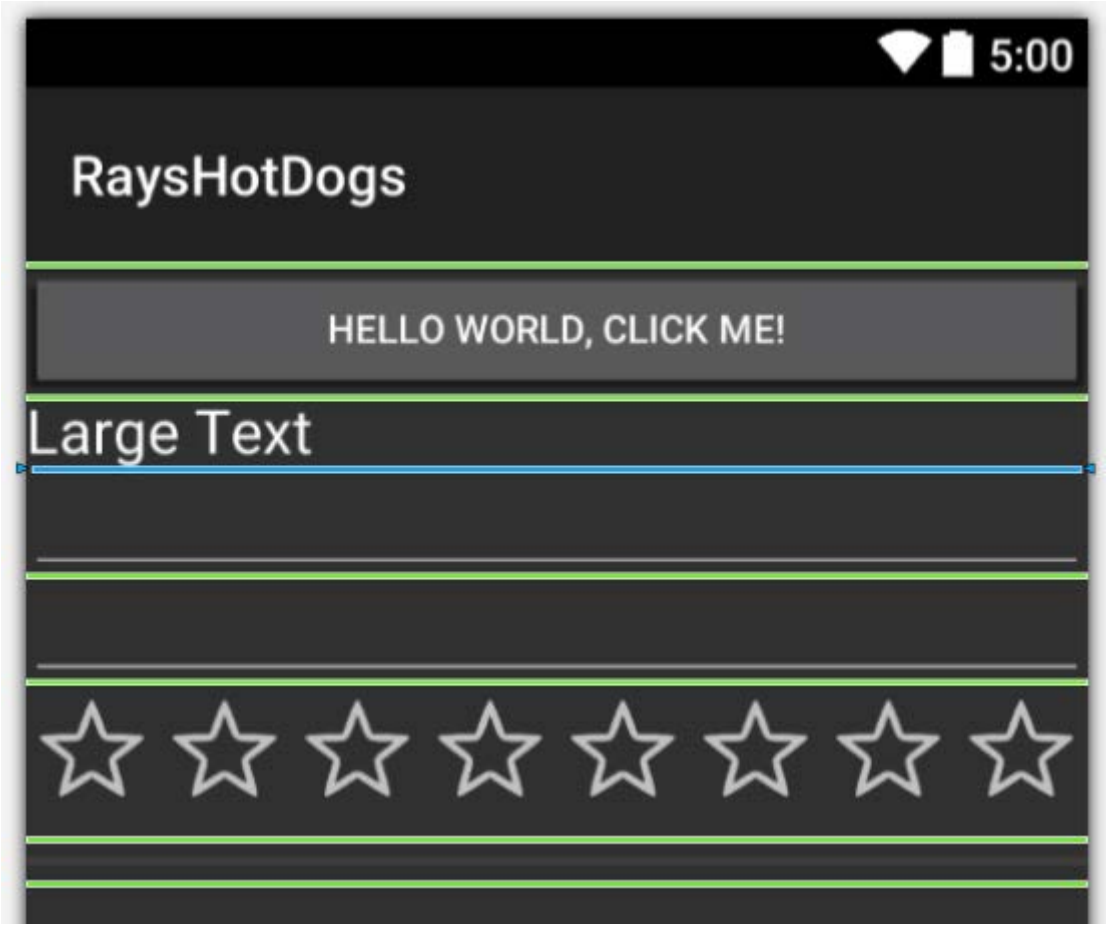
Demo Adding the Xamarin.Android project

Creating Our First View

Designer Features



Drag and Drop



Demo Taking a Look at the Designer

Layout Views

LinearLayout

RelativeLayout

TableLayout

GridView

And quite a few more!

LinearLayout


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/mainLinearLayout">
    <TextView
        android:text="Hot Dog Name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/hotDogNameTextView" />
    <TextView
        android:text="Short Description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/shortDescriptionTextView" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/descriptionTextView"
        android:text="Lorem ipsum dolor sit amet"/>
    <TextView
        android:text="Price: 1111"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/priceTextView" />
</LinearLayout>
```

RelativeLayout


```
<RelativeLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button
        android:text="Cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/cancelButton"
        android:layout_alignParentLeft="true"
        android:layout_centerInParent="true"
        android:layout_marginTop="20dp" />
    <EditText
        android:inputType="number"
        android:text="1"
        android:layout_width="30dp"
        android:layout_height="40dp"
        android:id="@+id/amountEditText"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />
    <Button
        android:text="Order now!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/orderButton"
        android:layout_alignParentRight="true"
        android:layout_centerInParent="true" />
</RelativeLayout>
```

TableLayout


```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1">
    <TableRow>
        <Button
            android:layout_column="0"
            android:text="Cancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/cancelButton" />
        <EditText
            android:layout_column="2"
            android:inputType="number"
            android:text="1"
            android:layout_width="30dp"
            android:layout_height="40dp"
            android:id="@+id/amountEditText" />
        <Button
            android:layout_column="3"
            android:text="Order now!"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/orderButton" />
    </TableRow>
</TableLayout>
```

Base Views

Button

TextView

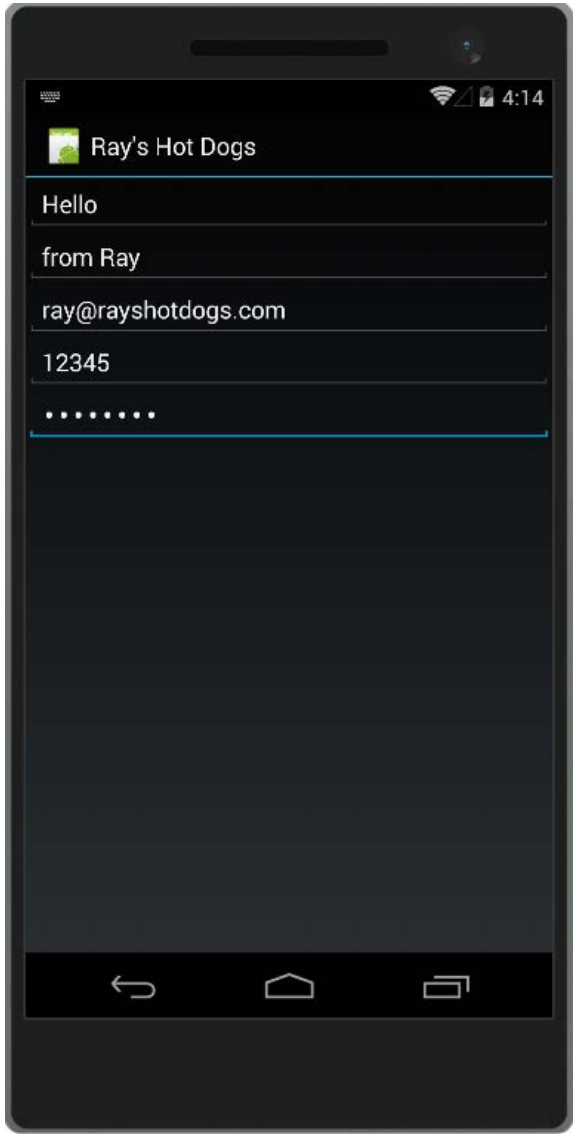
EditText

CheckBox

RadioButton

ImageView

EditText

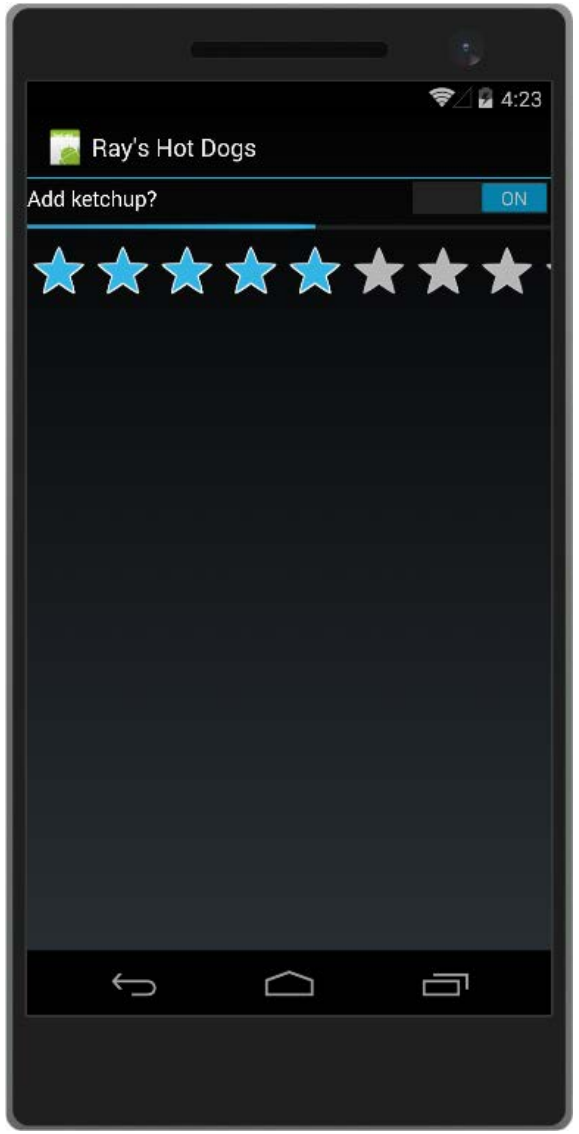



```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText1" />
<EditText
    android:inputType="textMultiLine"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText2" />
<EditText
    android:inputType="textEmailAddress"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText3" />
<EditText
    android:inputType="number"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText5" />
<EditText
    android:inputType="textPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText6" />
```

DatePicker and TimePicker


```
<DatePicker
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/datePicker1" />
<TimePicker
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/timePicker1" />
```

Other Controls




```
<Switch
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/switch1"
    android:text="Add ketchup?  " />
<ProgressBar
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/progressBar1"
    android:progress="55" />
<RatingBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/ratingBar1"
    android:rating="3" />
```

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    SetContentView(Resource.Layout.Main);

    Button button = FindViewById<Button>(Resource.Id.MyButton);

    button.Click += delegate
    {
        //Event handling code goes here
    };

}
```

Handing UI Events

Demo Creating the Detail View

Summary

Activities and Views are the heart of any Xamarin.Android app

Visual Studio offers a good editing experience to create Views