XamarinでAndroidアプリを作ろう

Androidアプリ作成の基本のキ

自己紹介

名前：重本　尚志（しげもと　ひさし）

年齢：31歳

趣味：トレーディングカード収集

好きな食べ物：味噌ラーメン、奈良漬

目次

- 目的
- Xamarinとは？
- 今回利用した開発環境
- HelloWorld
- レイアウトの基本
- じゃんけんアプリ

目的

- VisualStudioを利用したAndroidアプリのプロジェクト作成～起動の方法を知る
- C#でAndroidアプリが簡単に作れることを知る

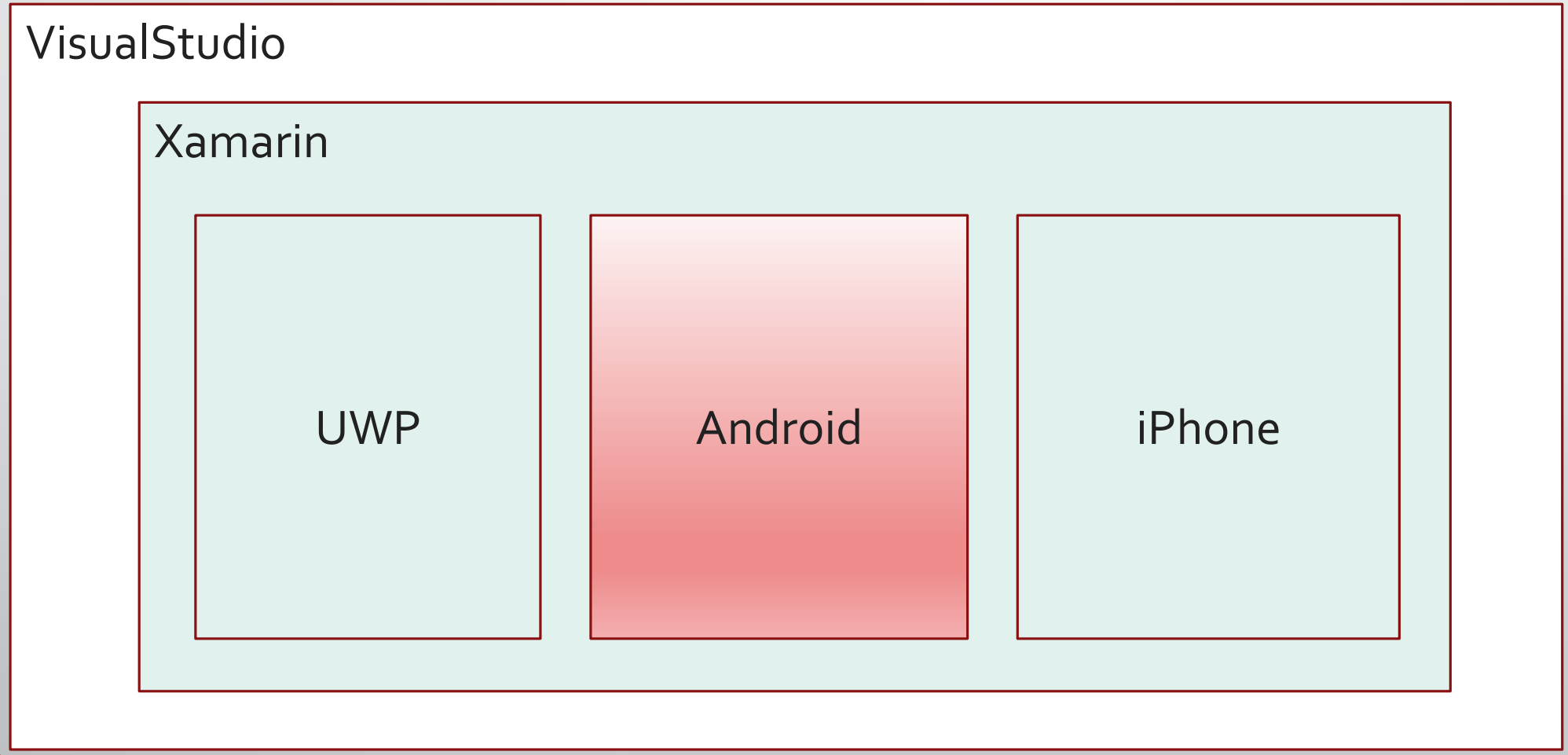
対象受講者

- Androidアプリを作ってみたいけど、Javaはちょっと・・・・という人。
- C#を勉強したけど、何を作ってみていいかわからない人。
- C#でAndroidアプリを作成する際の基本のキを知りたい人。

Xamarinとは？

- アメリカの企業名
  - 2016年にMicrosoftに買収されている。
  - 同社が開発したSDKを今回利用する。
- SDKはVisualStudioに統合されており、AndroidやiOS等のアプリが作成可能となっている。

Xamarinとは？



今回利用した開発環境

- VisualStudio2015 Community
  - https://www.microsoft.com/ja-jp/dev/products/community.aspx
- AndroidSDK
  - VisualStudioと同時にインストールできます（今回は別でインストールしました）。
  - https://developer.android.com/studio/index.html
- 実機（Softbank 403SH）
  - 今回の動作確認は全て実機で行っています。

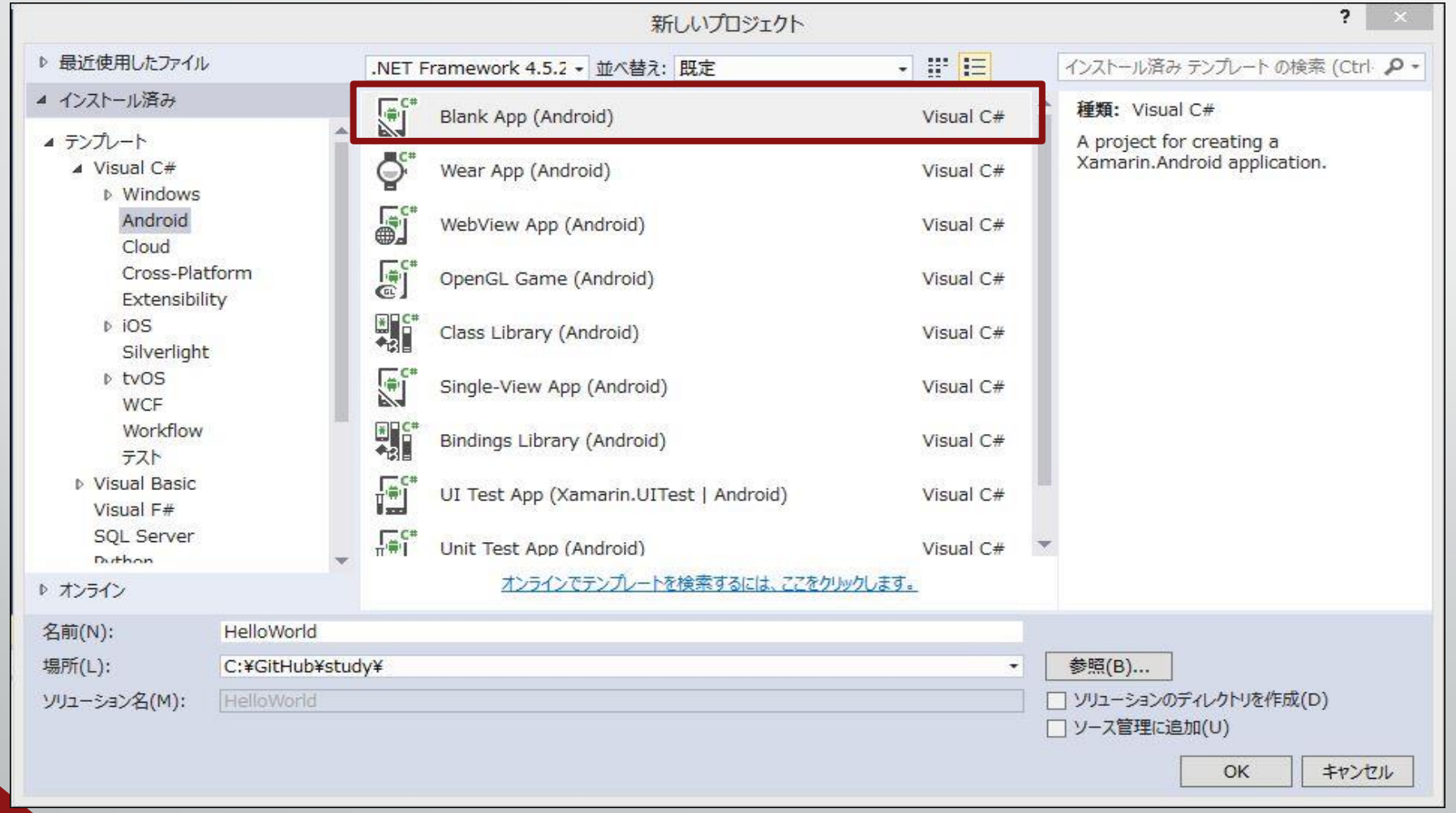
HelloWorld

- 目的
  - AndroidのBlankプロジェクトを利用して、HelloWorldのアプリを起動する
  - VisualStudioでのAndroidプロジェクトの作り方を学ぶ
- ソース
  - https://github.com/PUreatio/study/tree/master/HelloWorld

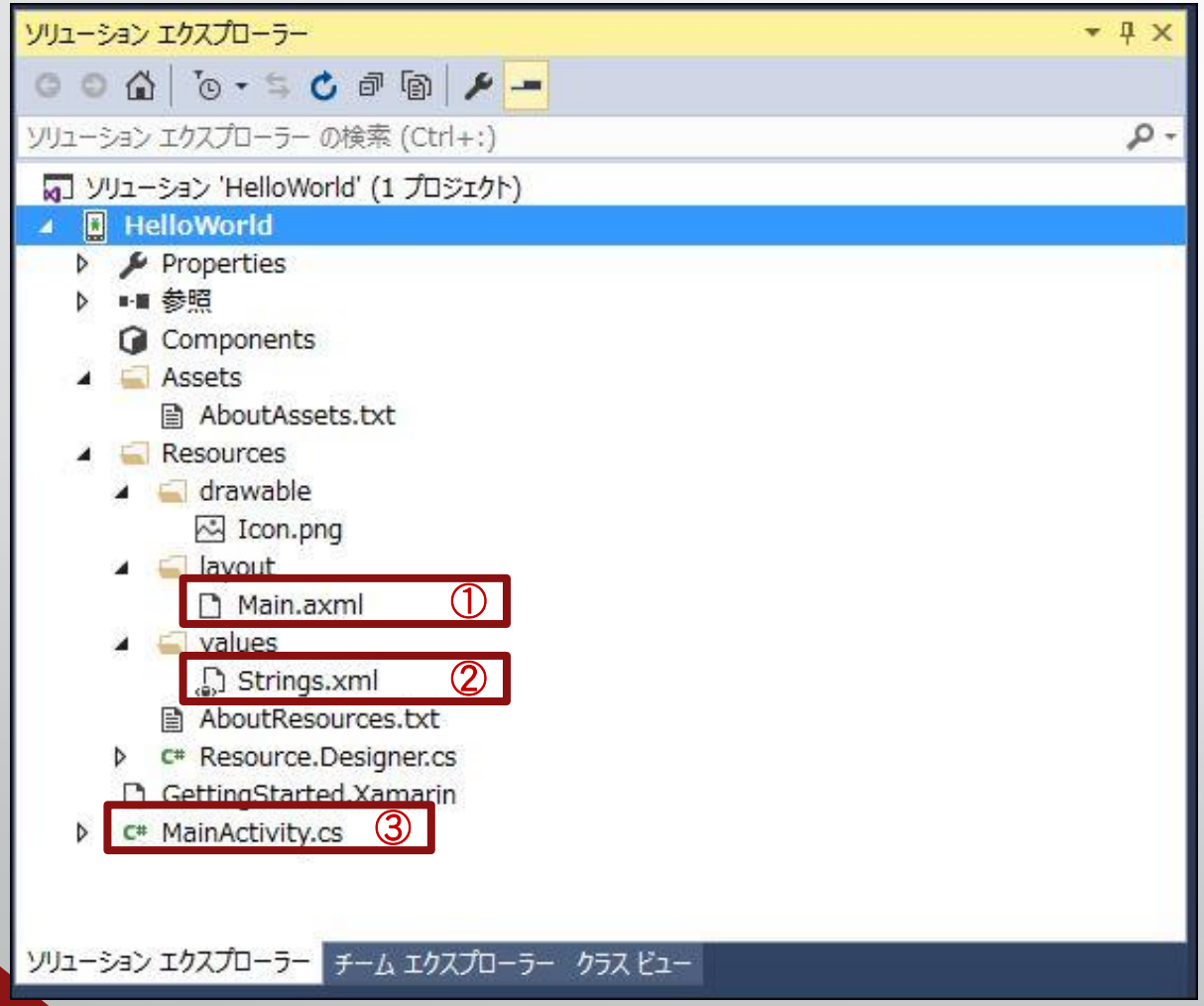
HelloWorld

1. プロジェクトの作成
2. 構成
3. レイアウト
4. Activity
5. 起動

HelloWorld①（プロジェクト作成）



HelloWorld②（構成）



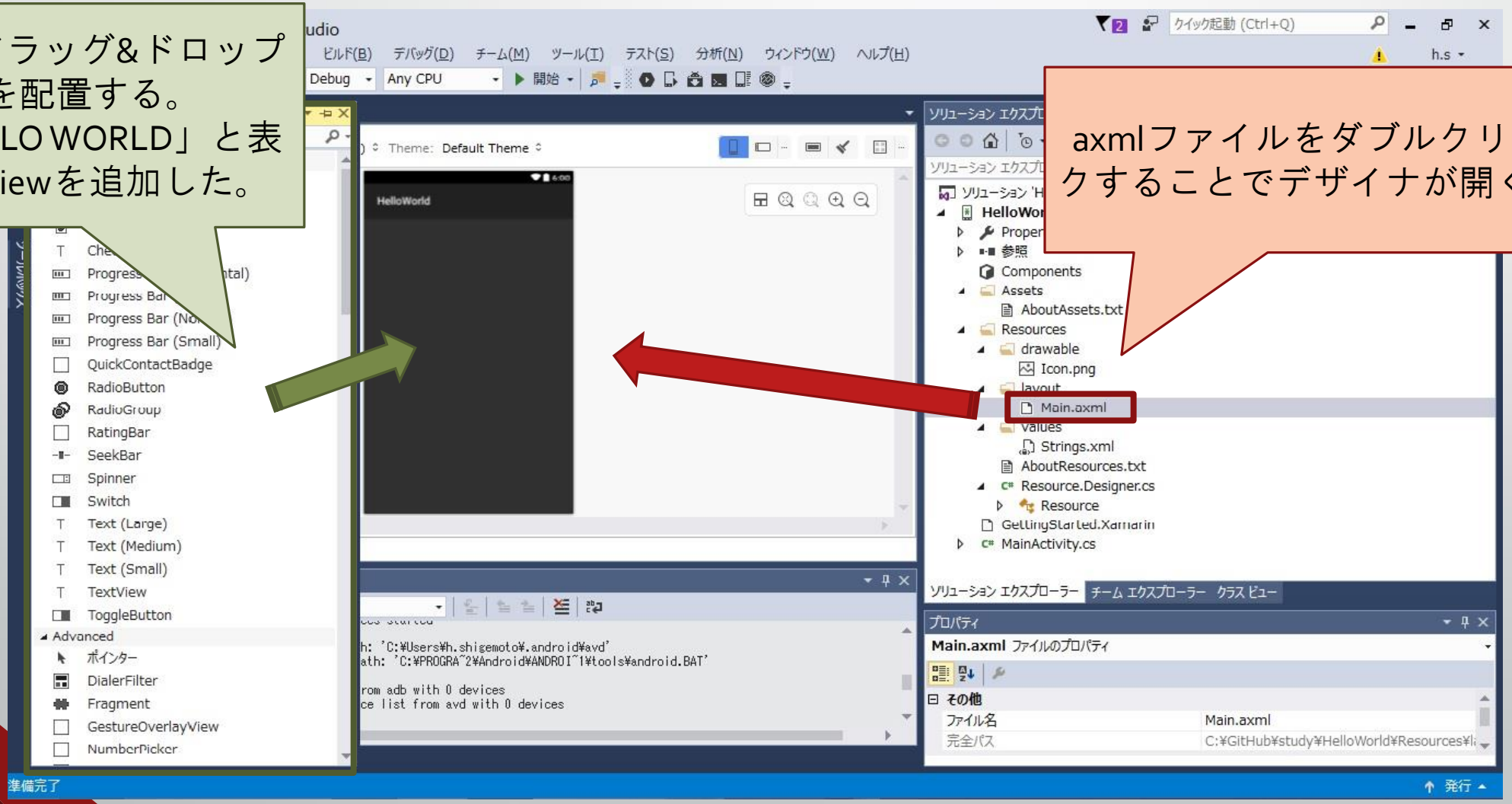
①Main.axml
画面のレイアウトファイル。

②Strings.xml
リソースファイル。
文字列をリソースとして登録することができる。

③MainActivity.cs
画面のソースファイル。
基本的に実装はこのファイルを中心に行っていく。

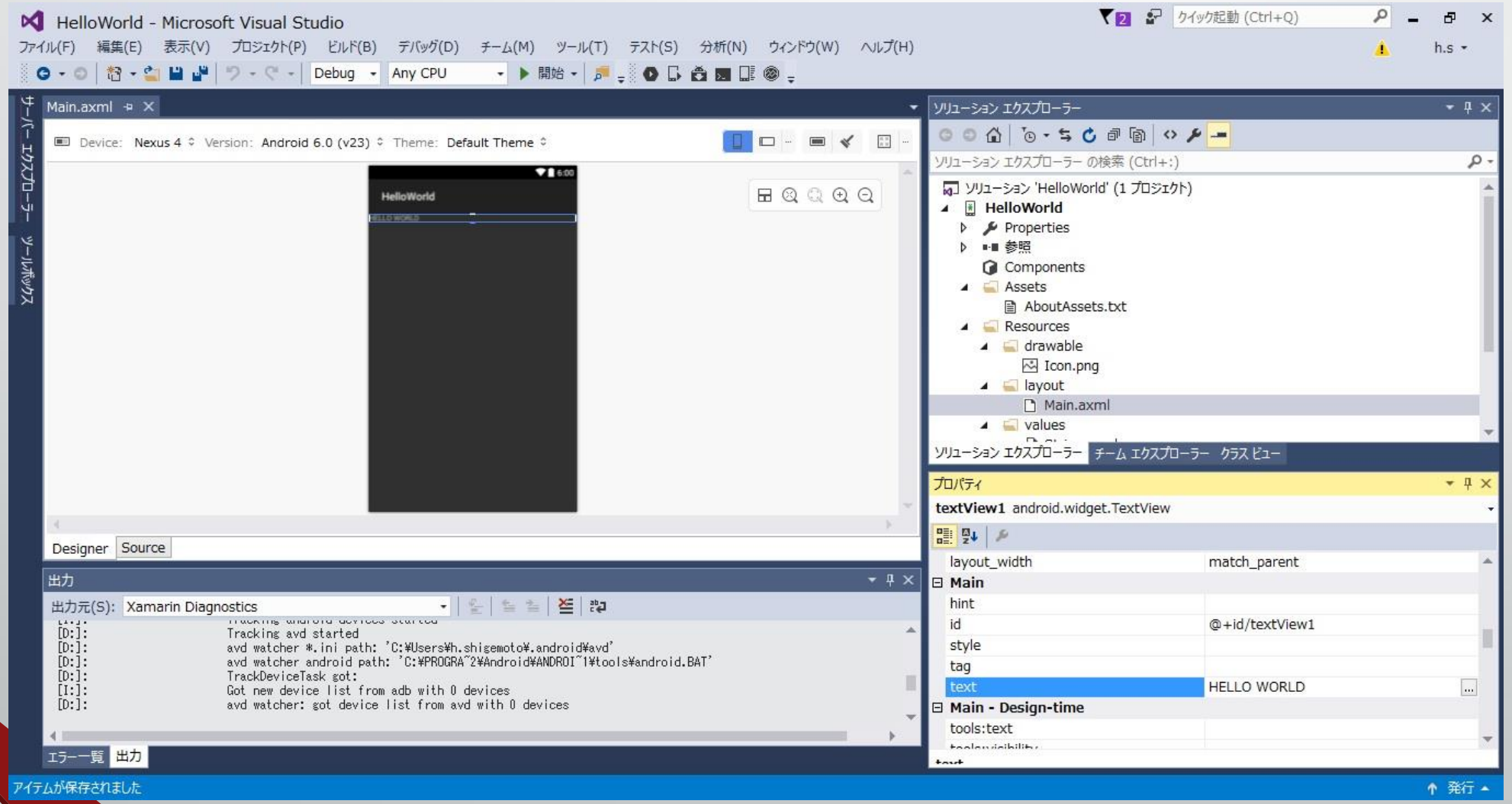
HelloWorld③（レイアウト）

デザインにドラッグ&ドロップでViewを配置する。
今回は「HELLO WORLD」と表示するTextViewを追加した。



axmlファイルをダブルクリックすることでデザイナが開く。

HelloWorld③（レイアウト）



HelloWorld④（Activity）

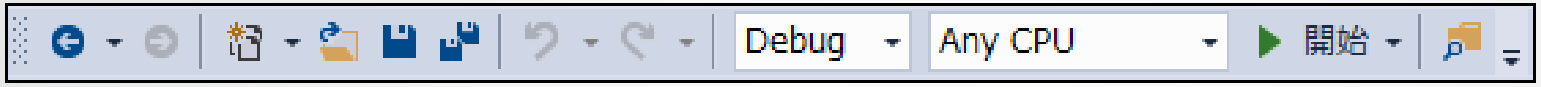
```
namespace HelloWorld
{
    [Activity(Label = "HelloWorld", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

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

赤枠部分は最初コメントアウトされているが、外しておく。

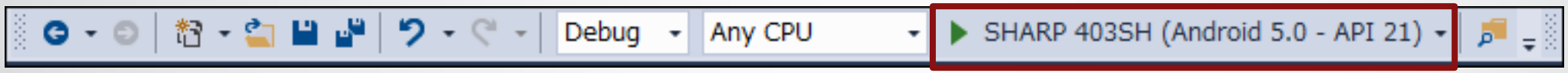
コメントアウトされたままだと、いくらレイアウトを変更しても反映されず、真っ暗な画面が表示され続ける。

HelloWorld⑤（起動）



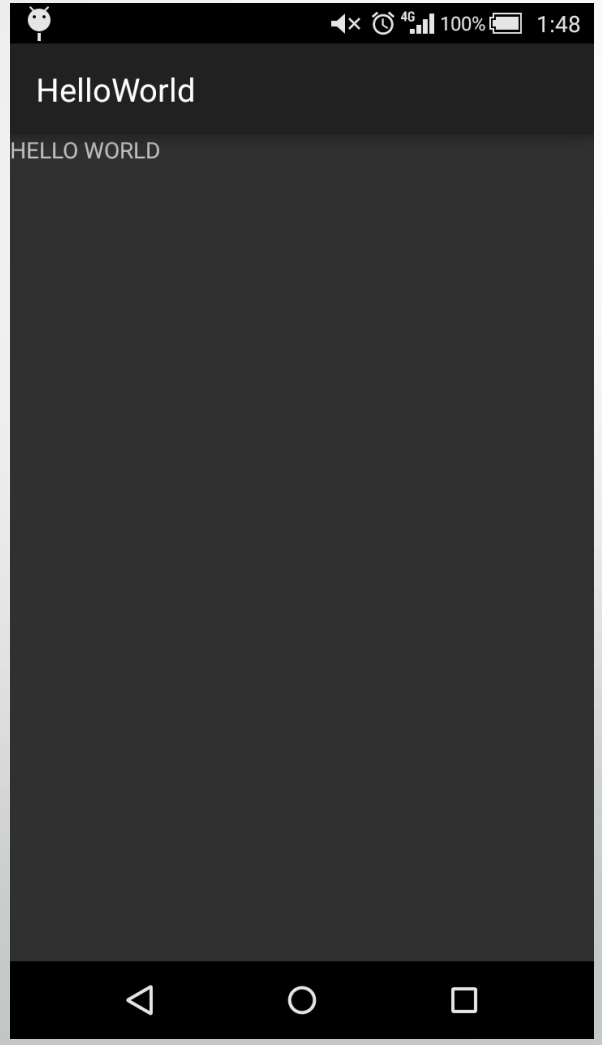
どのボタンでアプリケーションが起動すると思いますか？

HelloWorld⑤（起動）



先ほどの  ボタンの部分に、実機名（エミュレータで実行する場合はエミュレータ名）が表示されていないと、アプリケーションが起動できない。

HelloWorld⑤（起動）



レイアウトの基本

- 目的
  - Androidには様々なレイアウトが存在していることを知る
  - 今回の勉強会で作成するじゃんけんアプリで利用するレイアウトの基本を学ぶ

レイアウトの基本

1. レイアウトの種類
2. LinearLayout
3. 高さ・幅の指定
4. IDの指定

レイアウトの基本①（種類）

Androidには様々なレイアウトが存在している

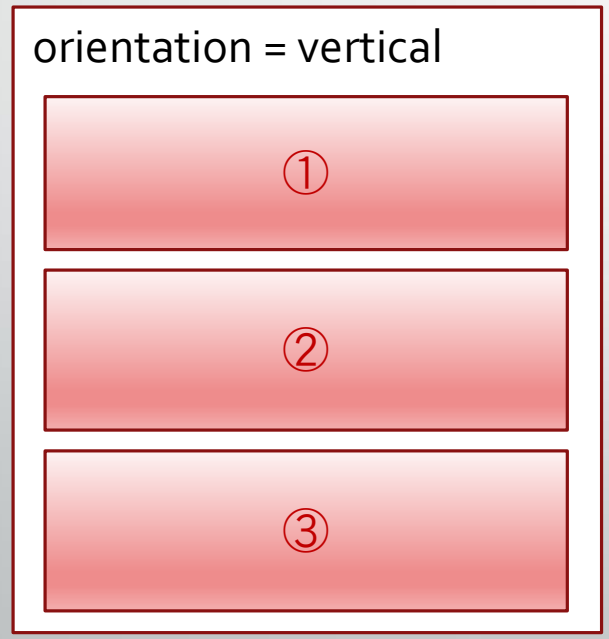
- RelativeLayout（各View同士の相対位置を指定）
- AbsoluteLayout（各Viewの絶対位置を指定）
- LinearLayout（各Viewを線形に配置）
- TableLayout（各Viewを格子に配置）

etc....

レイアウトの基本②（LinearLayout）

LinearLayoutは線形（縦方向）にViewを配置できる。

- android:orientation=“vertical” ⇒ 縦方向にViewを配置

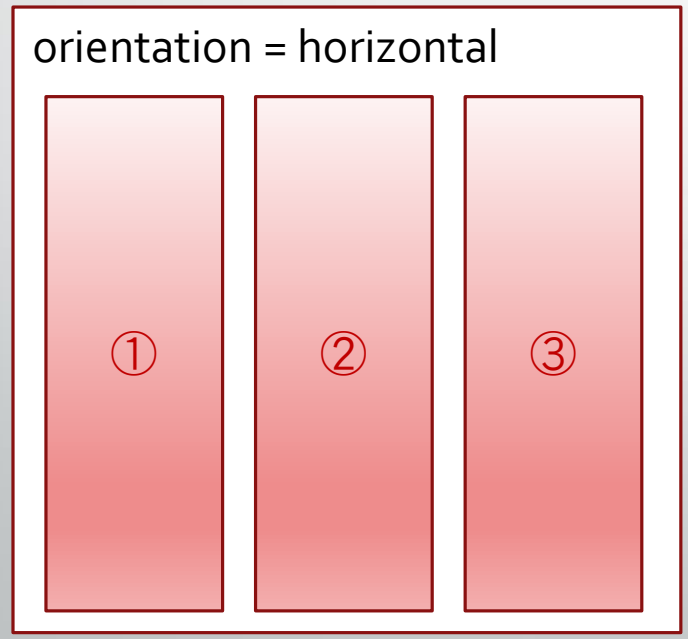



```
<LinearLayout
   android:orientation=“vertical">
   <!-- Viewを必要なだけ配置 -->
   <TextView android:text=“①” />
   <TextView android:text=“②” />
   <TextView android:text=“③” />
</LinearLayout>
```

レイアウトの基本②（LinearLayout）

LinearLayoutは線形（横方向）にもViewを配置できる。

- android:orientation="horizontal" ⇒ 横方向にViewを配置




```
<LinearLayout
   android:orientation="horizontal">
   <!-- Viewを必要なだけ配置 -->
   <TextView android:text="①" />
   <TextView android:text="②" />
   <TextView android:text="③" />
</LinearLayout>
```

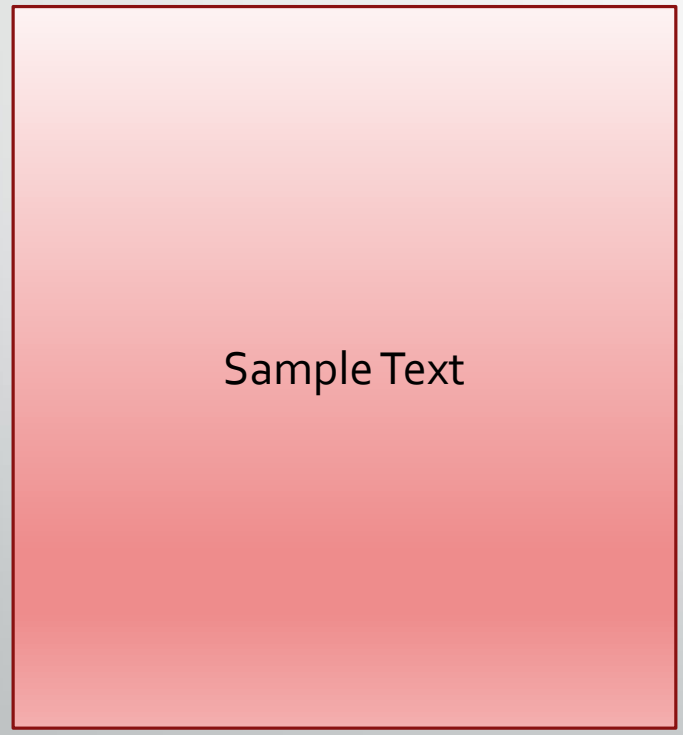
レイアウトの基本③（高さ・幅の指定）

各Viewの高さ（height）、横幅（width）の指定方法も複数存在する。

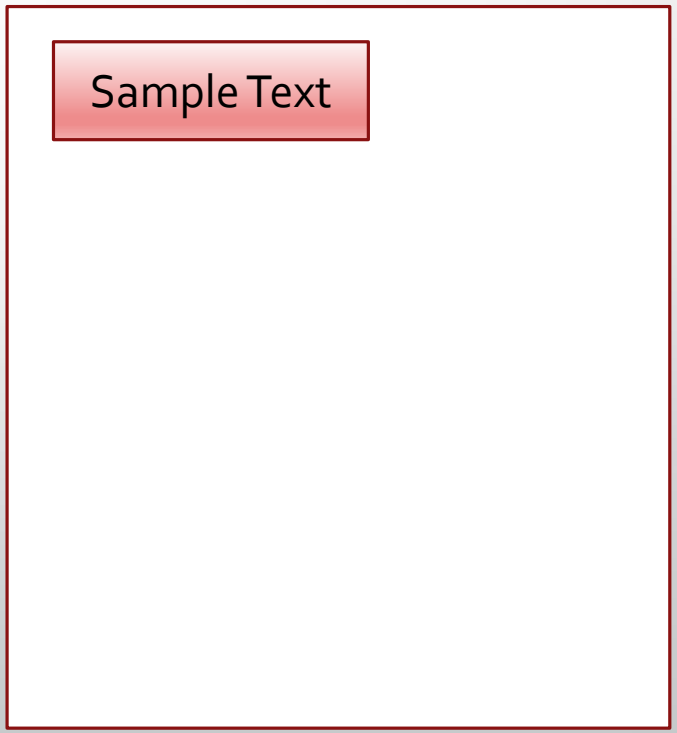
- match_parent（fill_parent）
  - 親Viewの幅に合わせた高さ、横幅になる。
- wrap_content
  - 自Viewの保持する値（文字列等）に合わせた高さ、横幅になる。
- dp（dip）
  - 密度非依存のPixel指定。マルチレイアウトに対応する際に利用する。
- px
  - 一般的なPixel指定。各画面の解像度に依存する。

レイアウトの基本③（高さ・幅の指定）

android:layout_height="match_parent"
android:layout_width="match_parent"

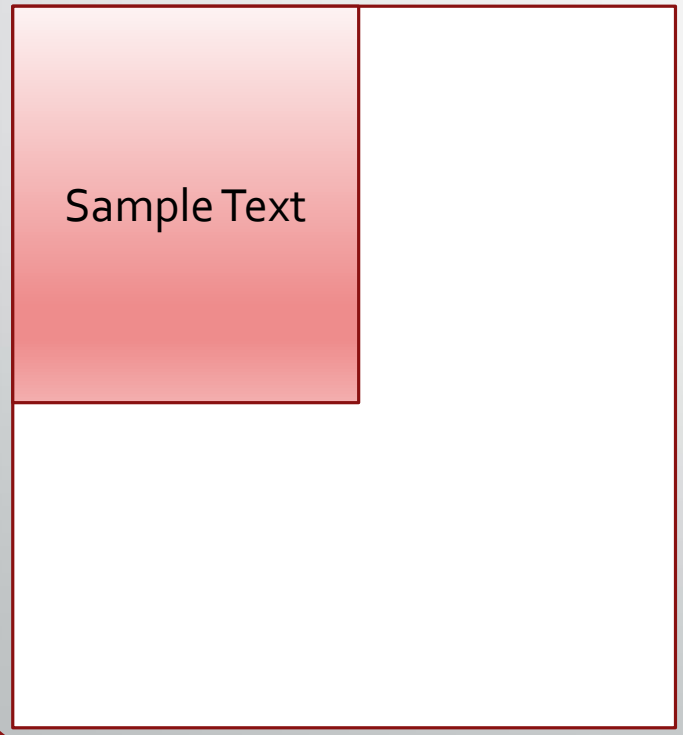


android:layout_height="wrap_content"
android:layout_width="wrap_content"

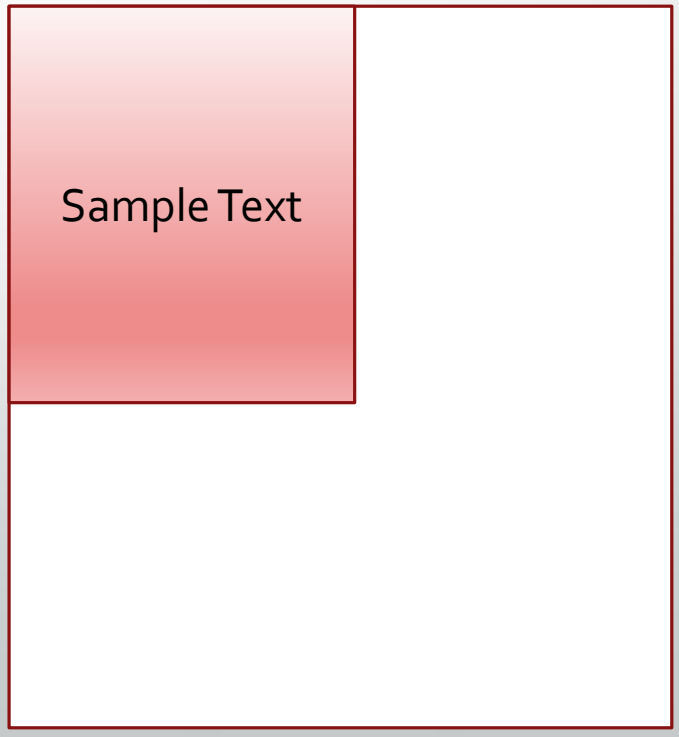


レイアウトの基本③（高さ・幅の指定）

android:layout_height="100dp"
android:layout_width="100dp"
画面のサイズ = 100px * 100px

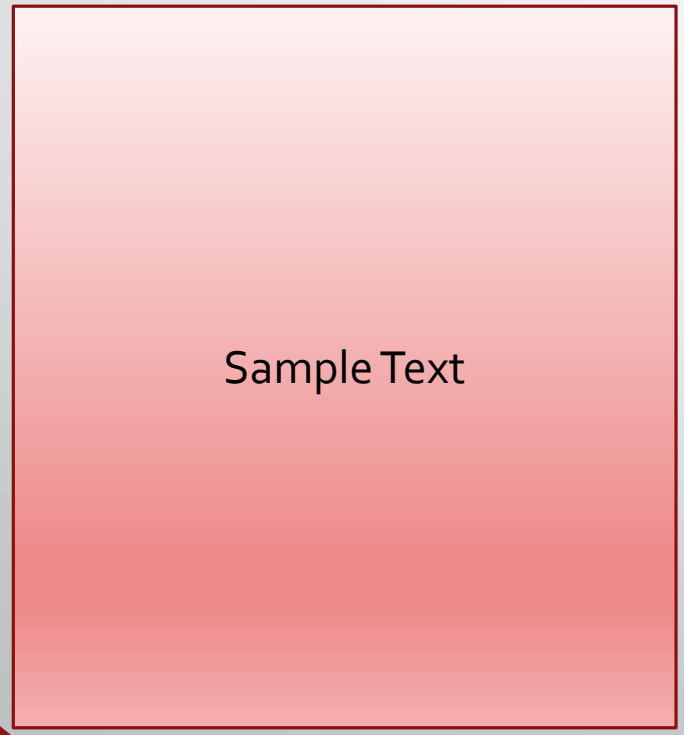


android:layout_height="100dp"
android:layout_width="100dp"
画面のサイズ = 200px * 200px

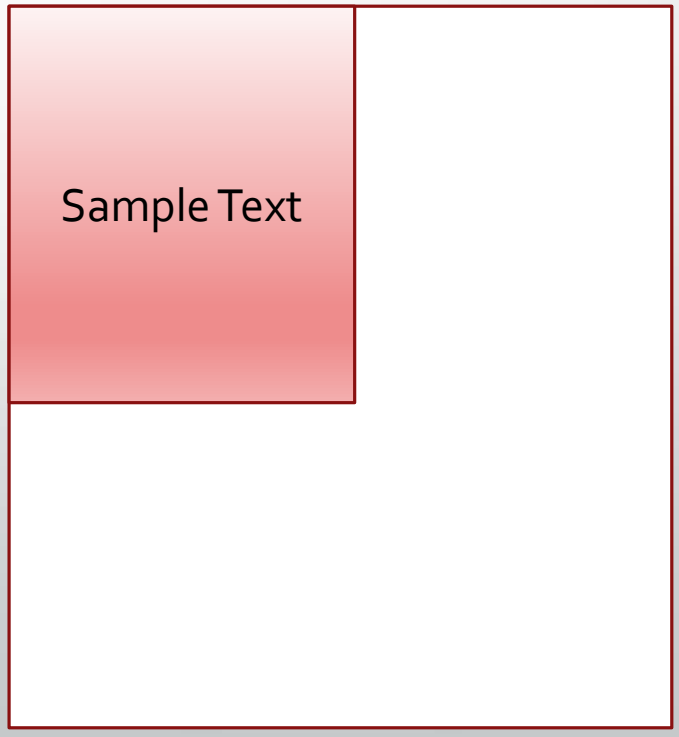


レイアウトの基本③（高さ・幅の指定）

android:layout_height="100px"
android:layout_width="100px"
画面のサイズ = 100px * 100px



android:layout_height="100px"
android:layout_width="100px"
画面のサイズ = 200px * 200px



レイアウトの基本④（IDの指定）

レイアウト上に配置した各Viewに対して、android:idプロパティに以下の形式でIDを指定する。

- @+id/任意のID

「textView1」というIDを指定したい場合は・・・・

- android:id="@+id/textView1"

レイアウトの基本④（IDの指定）

対象プロジェクト¥Resources¥Resource.Designer.cs

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

    static Resource()...

    public static void UpdateIdValues()...

    public partial class Attribute...

    public partial class Drawable...

    public partial class Id
    {

        // aapt resource value: 0x7f050000
        public const int textView1 = 2131034112;

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

「@+id/任意のID」の形式で定義するとC#に反映される。Resourceに反映されると、Activity側から参照できるようになる。

じゃんけんアプリ

- 目的
  - HelloWorldから一歩進んだ簡単なアプリケーションを作ってみる
- ソース
  - https://github.com/PUreatio/study/tree/master/RockPaperScissors

じゃんけんアプリ

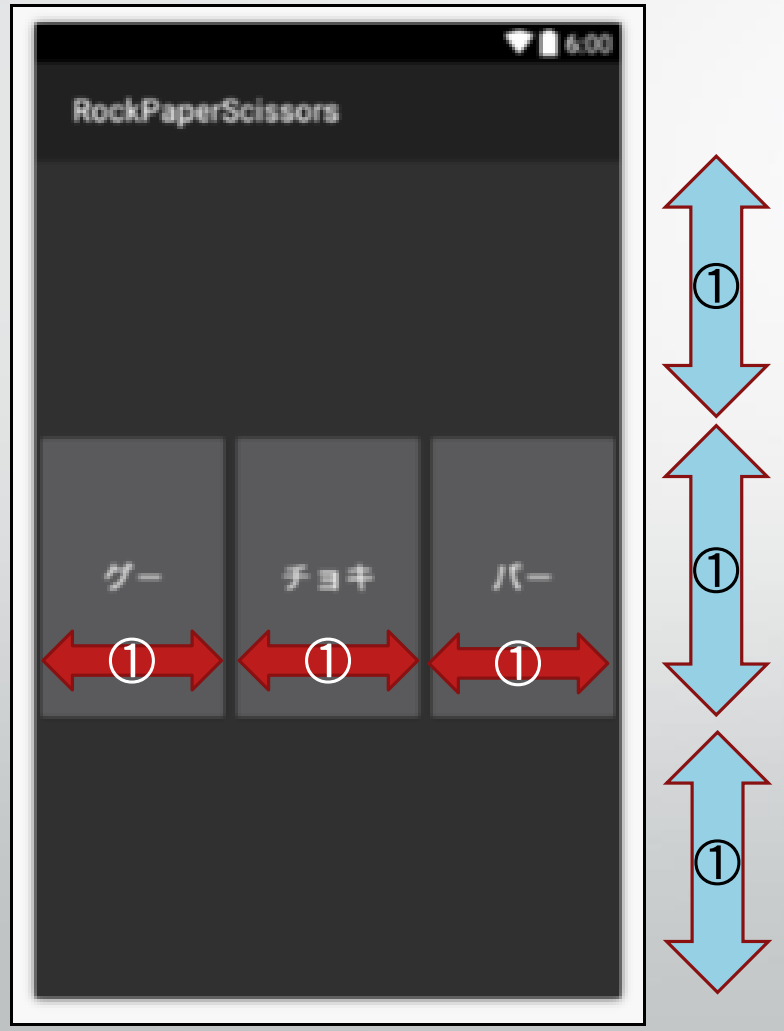
ソースについては、Android特有の実装をしている部分を主に説明していきます。

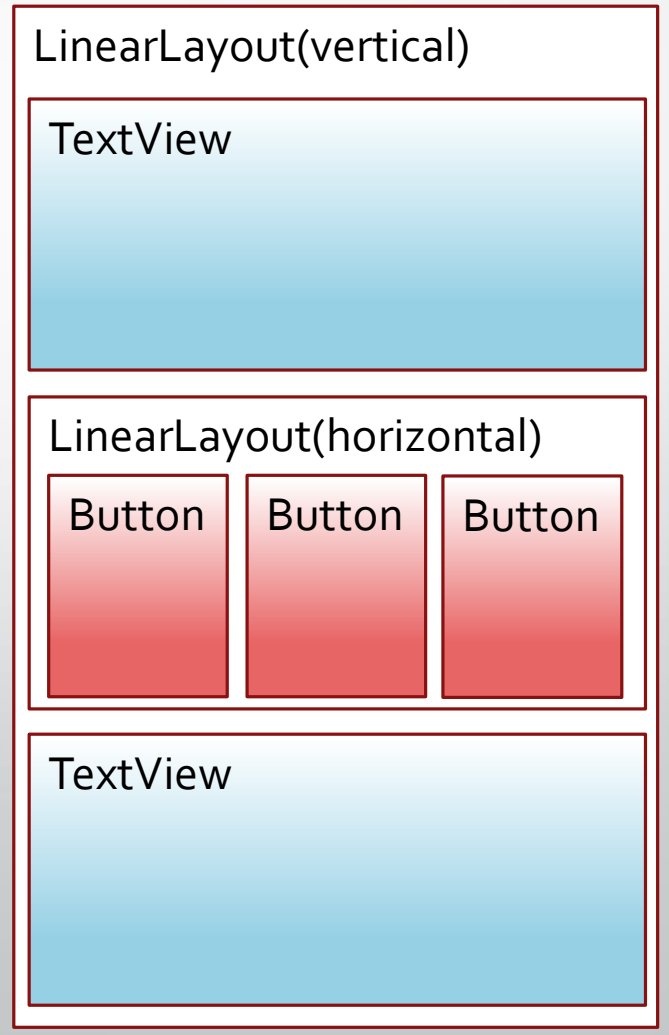
- 仕様
  - じゃんけんを行い、結果を画面に表示する（1画面で処理は完結させる）
  - 自分が出す手のボタンを選択し、相手の出す手をロジックで判定し、勝敗を決める
  - 相手の手については、0～2の乱数を発生させ、数値の小さいほうからグー、チョキ、パーとする

じゃんけんアプリ

1. レイアウト
2. Activity
3. 起動

じゃんけんアプリ①（レイアウト）



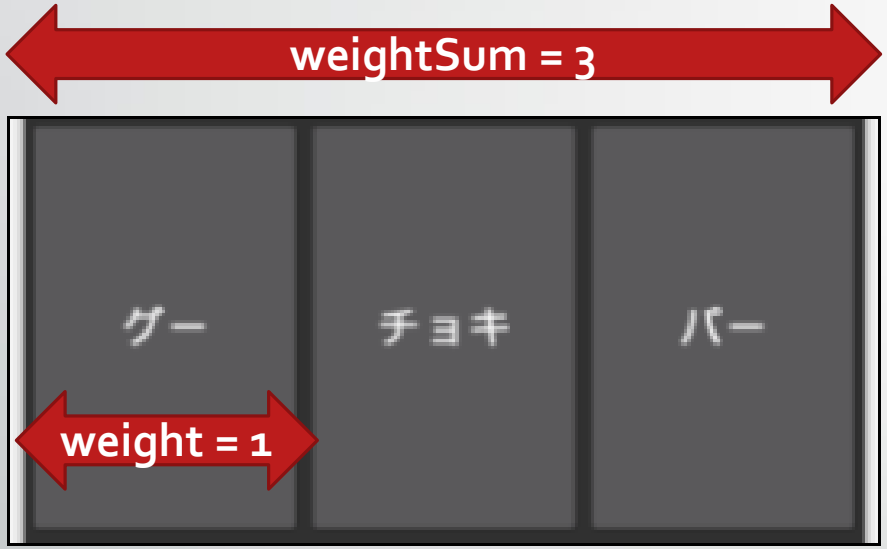


じゃんけんアプリ①（レイアウト）

レイアウトの重み付けをすることによって、Viewの高さ、幅を割合定義することもできる。

- android:weightSum ⇒ 各Viewの重みの合計。親Viewに定義する。
- android:layout_weight ⇒ 各Viewの重み

じゃんけんアプリ①（レイアウト）




```
<LinearLayout   android:orientation="horizontal"
   android:layout_weight="1"
   android:weightSum="3">
   <Button      android:text="@string/Rock"
      android:layout_width="wrap_content"
      android:layout_height="match_parent"
      android:layout_weight="1"/>
   <Button      android:text="@string/Scissors"
      android:layout_width="wrap_content"
      android:layout_height="match_parent"
      android:layout_weight="1" />
   <Button      android:text="@string/Paper"
      android:layout_width="wrap_content"
      android:layout_height="match_parent"
      android:layout_weight="1" />
</LinearLayout>
```

じゃんけんアプリ①（レイアウト）

Strings.xmlに定義した値を、レイアウト上の文字列として利用できる。

- 「@string/任意の名称」で指定する。

android:text="@string/Rock"

対象プロジェクト¥Resources¥values¥Strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="Rock">グー</string>
 <string name="Scissors">チョキ</string>
 <string name="Paper">パー</string>
 <string name="Win">あなたの勝ち！</string>
 <string name="Loose">あなたの負け！</string>
 <string name="Tie">引き分け！</string>
 <string name="ApplicationName">RockPaperScissors</string>
</resources>
```

じゃんけんアプリ①（レイアウト）

対象プロジェクト¥Resources¥Resource.Designer.cs

Resource.Designer.cs

RockPaperScissors　　RockPaperScissors.Resource.Id

```
90
91          public partial class Layout...
106
107         public partial class String
108         {
109
110             // aapt resource value: 0x7f040006
111             public const int ApplicationName = 213096858
112
113             // aapt resource value: 0x7f040004
114             public const int Loose = 2130968580;
115
116             // aapt resource value: 0x7f040002
117             public const int Paper = 2130968578;
118
119             // aapt resource value: 0x7f040000
120             public const int Rock = 2130968576;
121
122             // aapt resource value: 0x7f040001
123             public const int Scissors = 2130968577;
124
```

Strings.xmlにリソースを定義すると、
Resource.Designer.csに反映される。

じゃんけんアプリ②（Activity）

```
/// <summary>
/// 初期化処理を行う。
/// </summary>
private void Initialize()
{
    // メッセージをTextViewに設定する。
    TextView readyView = this.FindViewById<TextView>(Resource.Id.textViewReady);
    readyView.Text = string.Format(Messages.READY_MESSAGE, SysEnvironment.NewLine);
    // 結果にも初期表示メッセージを設定する。
    TextView resultView = this.FindViewById<TextView>(Resource.Id.textViewResult);
    resultView.Text = string.Format(Messages.RESULT_INIT_MESSAGE, SysEnvironment.NewLine);

    // 各ボタンにイベントを設定する。
    Button rockButton = this.FindViewById<Button>(Resource.Id.buttonRock);
    rockButton.Click += RockPaperScissorsButton_Click;
    Button scissorsButton = this.FindViewById<Button>(Resource.Id.buttonScissors);
    scissorsButton.Click += RockPaperScissorsButton_Click;
    Button paperButton = this.FindViewById<Button>(Resource.Id.buttonPaper);
    paperButton.Click += RockPaperScissorsButton_Click;
}
```

じゃんけんアプリ②（Activity）

ViewをActivityで取得するためには、FindViewByIdメソッドを利用する。

```
TextView readyView = this.FindViewById<TextView>(Resource.Id.textViewReady);
```

レイアウト側で定義したID

じゃんけんアプリ②（Activity）

```
// 相手が勝ちな場合
if (opponentHand.Item1 == winHand)
{
    gameResultMessage = this.GetString(Resource.String.Win);
}
// 相手が負けな場合
else if (opponentHand.Item1 == looseHand)
{
    gameResultMessage = this.GetString(Resource.String.Loose);
}
```

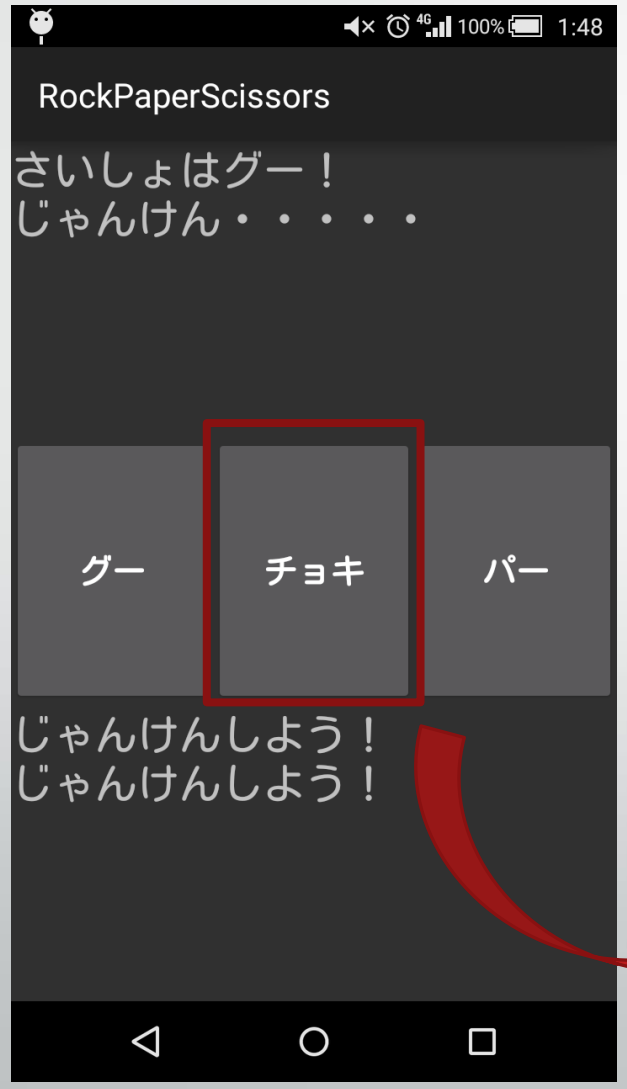
じゃんけんアプリ②（Activity）

リソース（文字列）を取得するためには、GetStringメソッドを利用する。

```
gameResultMessage = this.GetString(Resource.String.Win);
```

Strings.xmlに定義したID

じゃんけんアプリ③（起動）



ご清聴ありがとうございました。

勉強会をより良くしていくため、アンケートにご協力お願いします。
http://pureatio.com/script/mailform/study_questionnaire/