Custom Button in Android

Posted: 1 year ago

For most of the applications, the default button available in Android API is good enough, but however, there will be instances when you will have to customize or add more properties/functionality to your button. The easiest way to do so is to extend from default Button and create your own Custom Button. In this Step by Step tutorial I will show you how to create a custom button, and how to make it configurable from XML.

For the purpose of this tutorial, imagine we are building a Calculator App. In an Calculator App, we have several buttons (10 buttons for Numbers, and few for operators). Instead of writing onClick listeners for each Number button separately, we can write a single onClick listener and we will make each button store its corresponding value within its own object.

Step 1: Create MyButton Class

```
1  package com.jtechniques.calc;
2  public class MyButton extends Button {
3      private int myNumber ;
4
5      public MyButton(Context ctx) {
6          super(ctx);
7      }
8      public String getMyNumber() {
9          return myNumber;
10     }
11     public void setMyNumber(int myNumber) {
12         this.myNumber = myNumber;
13     }
14 }
```

Now we have our own Button class, and as you can see this is a simple POJO class, and only important thing is that the Constructor which takes a Context parameter is required, and we just call the super(..) constrcutor. Now, we can initialize our MyButton from java, and set myNumber value by using setter method. But, with above implementation we will not be able to set button properties from Layout XML.

To make our button widget attributes to be set able from XML, we need to add

Follow us on

Most Popular

Canvas Demo - Build a Chessboard using HTML5 Canvas

Show Pretty Date using Javascript

Integrating Java with Perl

Activity Lifecycle and Intents in Android

Custom Button in Android

Markers, InfoWindow and Event Listeners in Google Maps

Disadvantages of AJAX

another constructor which takes in 2 parameters, Context and AttributeSet. Here is the updated MyButton class

```
package com.jtechniques.calc;
public class MyButton extends Button {
    private int myNumber ;

    public MyButton(Context ctx) {
        super(ctx);
    }
    public MyButton(Context ctx, AttributeSet
        super(ctx,attrs);
    }

    public String getMyNumber() {
        return myNumber;
    }
    public void setMyNumber(int myNumber) {
        this.myNumber = myNumber;
    }
}
```

Now, we will be able to set button attributes from layout XML file, but, still we cannot set our own property "myNumber" from xml, since this attribute is not defined and not available in schema.

Step 2: Define attribute in Schema

Open styles.xml file under /res/values/ folder. and add the below code snippet

```
<declare-styleable name="style_myButton">
    <attr name="my_number" format="string">
</attr></declare-styleable>
```

Step 3: Modify MyButton to retrieve my_number attribute from XML

```
public class MyButton extends Button {
    private int myNumber ;

    public MyButton(Context ctx) {
        super(ctx);
    }
    public MyButton(Context ctx, AttributeSet
        super(ctx,attrs);
        initMyButton(attrs);
    }

    public String getMyNumber() {
        return myNumber;
    }
    public void setMyNumber(int myNumber) {
        this.myNumber = myNumber;
    }

    private void initMyButton(AttributeSet at
        TypedArray a = getContext().obtainSty
        String my_number = a.getString(R.styl

        setText(my_number);
        this.my_number = new Integer(my_numbe
        a.recycle();
    }
}
```

We call initMyButton method to retrieve our "my_number" attribute from attribute set, and set the text of the button and also save the number in our object's state.

Step 4: Using our Button from Layout XML

In your activity's layout xml, define the namespace to use custom widget. Below is the code snippet:

```
<relativelayout xmlns:android="http://schemas?
</relativelayout>
```

Define our button in Layout XML

```
ick="calculateAction" android:paddingright="50dp" my
```

The above code will layout the button and set its value to 3.

> To build an actual Calculator App, you would layout the buttons programmatically or from the XML layout, and set the onClick event to point to same function and retrieve the button's value from its object.

Example: onClick Listener

```
public void calculateAction(View view) {
    MyButton btn = (MyButton)view;
    System.out.println("Number pressed : " + b
}
```

Author - Harish B N