Android中的命名空间

2016年1月23日 谭瀚涛 Android 1 Comment

基本概念介绍

- 命名空间(namespace)

> *XML 命名空间提供避免元素命名冲突的方法。*
> *—w3school.com*

打个比方,**A**学校有名学生叫做**林小明**,**B**学校也有名学生叫**林小明**,那我们如何识别这两名拥有相同名字的同学呢?这时候**命名空间**就派上用场了。**A**和**B**此时就可以被当成是命名空间了。也就是说，**命名空间里面存放的是特定属性的集合**,

Android中常见的命名空间

android

1. **xmlns:android=”http://schemas.android.com/apk/res/android”**
   在Android布局文件中我们都必须在**根**元素上定义这样一个命名空间,接下来对这行代码进行逐一讲解：
   xmlns:即**xml namespace**，声明我们要开始定义一个命名空间了
   android处于这个位置的字符串,我们称作**namespace-prefix**,实际上它代表的含义就是赋予命名空间一个名字
   http://schemas.android.com/apk/res/android这看起来是一个

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

URL，但是这个地址是不可访问的。实际上这是一个URI(统一资源标识符),所以它的值是**固定不变的**,相当于一个**常量**)。
那么这行代码有什么作用呢？在我们写在**根**元素的代码的可以引用到命名空间中的属性,如:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/re
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="New Text"
        android:id="@+id/textView" />
</LinearLayout>

```

在这个布局中,只要以*android:*开头的属性便是引用了命名空间中的属性,上面已经提到过,**android**便是赋予命名空间一个名字,就跟我们平时在定义变量一样，所以这个名字也可以取成是我们喜欢的,比如我把它取成**myns**,那么上面的代码我们也可以写成:

```
<LinearLayout xmlns:myns="http://schemas.android.com/apk/res/a
    myns:layout_width="match_parent"
    myns:layout_height="match_parent" >
    <TextView
        myns:layout_width="wrap_content"
        myns:layout_height="wrap_content"
        myns:layout_gravity="center"
        myns:text="New Text"
        myns:id="@+id/textView" />
</LinearLayout>

```

这样的代码看起来不符合规范,但是它确是可以正常编译成功的！

> *注意:在Android中,是必须给命名空间起一个别名的,*
> *因为在Android中不止有一种命名空间存在！*
> *接下来介绍另外两种常用的命名空间：*

tools

xmlns:tools="http://schemas.android.com/tools"
我们可以把他理解为一个工具(tools)的命名空间,是帮助开发人员的工具.它的作用只于开发阶段发挥,当app被打包时,所有关于**tools**属性将都会被摒弃掉！
例如,基本上在**android**命名空间内的属性,我们想在编写代码阶段测试某个组件在屏幕上的效果,而当app安装到手机上时，摒弃掉这条代码,那么我们就可以用**tools**命名空间来代替掉**android**:

谭瀚涛

点击这里可联系到我

技术之旅

目录

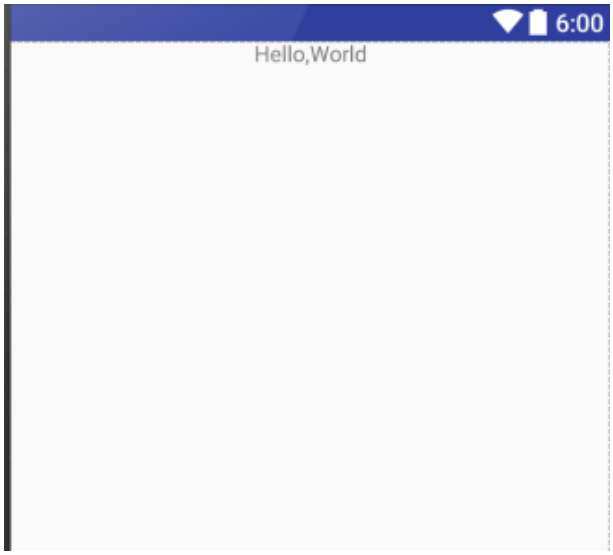
推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

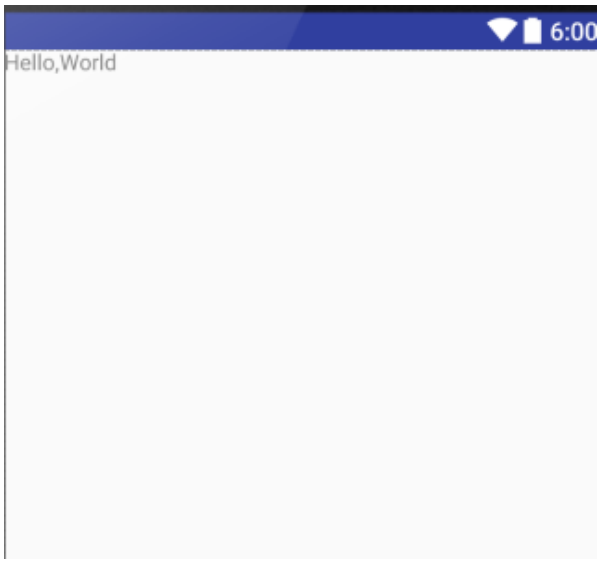
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        tools:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello,World"/>
</LinearLayout>

```

以上是在layout中的布局,当我们切换到视图窗口(Design)中查看时,看到的是标签顶部居中显示:



然后,当我们运行到手机上时,确是这样的:



如上所示,**tools:layout*gravity="center"***确实在运行后背抛弃掉了!

这是tools的用法之一,tools自己的命名空间内也有一些独特的属性,例如: ***context****,这个属性后面跟的是一个Activtiy的完整包名,它有什么作用呢?*

*当我们设置一个Activity主题时,是在AndroidManifest.xml中设置中,而主题的效果又只能在运行后在Activtiy中显示,借助****context****属性,我们便可以在开发阶段中看到设置在Activity中的主题在布局中的效果,*

tools:context="com.littlehan.myapplication.MainActivity"

在布局中加入这行代码,便design视图中看到与MainActivity绑定主题的效果

如你对文章有问题，可联系下方『文
疑人QQ』。为避免他人反感，维护
习社区，联系前请先阅读『提问的智

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

当我们在Activity上加载一个fragment时，是需要在运行后才可以看到加载后的效果,有没有方法在测试阶段就在布局预览窗口上显示呢?答案是有的,借助layout属性,例如,
在布局中加入这样一行代码:
tools:layout=@layout/yourfragment/layoutname
这样你的编写的fragment布局就会预览在指定主布局上了

自定义命名空间

app

xmlns:app="http://schemas.android.com/apk/res-auto"
前面讲的都是Android自带的命名空间,那么我们如何定义属于自己的命名空间。 通常自定义命名空间往往是和自定义View分不开的，当Android自带的控件不能满足我们的需求时，我们会自己去绘制一些**View**，而要为这些自定义View加上**自定义的属性**时,这时候就需要我们去创建属于自己的命名空间。开头已经说了，**命名空间里面存放的是特定属性的集合**，这样一来，思路就很清晰,也就是说自定义命名空间的实际过程就是去自定义属性。我们通过一个简单的**自定义TextView**来学习下自定义命名空间是怎么一回事，自定义View的过程可以分成以下几个步骤：

继承View类

创建一个类名为CustomTextView继承View
（View是所有视图的父类）并实现它三个构造方法

```
1  public class CustomTextView extends View {
2      private Paint mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);//
3      public CustomTextView(Context context) {
4          super(context);
5      }
6      public CustomTextView(Context context, AttributeSet attrs)
7          this(context, attrs, 0);//注意不是super(context,attrs,0
8      }
9      public CustomTextView(Context context, AttributeSet attrs,
10         super(context,attrs,defStyleAttr);
11     }
12 }
13
```

绘制视图

重载**onDraw()**方法绘制一个一段文本

```
1 @Override
2     protected void onDraw(Canvas canvas) {
3         super.onDraw(canvas);
```


谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

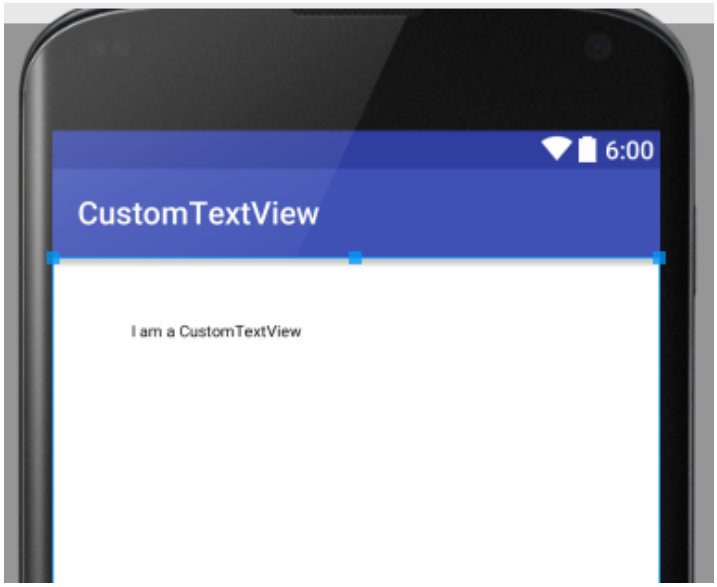
```
4           canvas.drawText("I am a CustomTextView",100, 100, mPair
5       }
6
```

引入布局

将自定义的控件引入布局

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:background="#ffffff"
6      >
7  <com.littlehan.customtextview.CustomTextView
8      android:layout_width="wrap_content"
9      android:layout_height="wrap_content" />
10 </RelativeLayout>
11
```

到了这里，一个自定义的控件就被引入布局使用了，我们可以切换到视图窗口看看效果



但是，这个控件是固定死的，它并不能**自定义文本**和**颜色**。接下来我们来定义这两个功能。这个功能的实现实际上就是**XML创建自定义属性**和在**自定义View中解析属性**的过程

自定义属性

创建一个values根目录下新建一个名为attrs的xml文件来自定义我们的属性（在这里定义的属性便是我们自定义命名空间里面的属性）

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <declare-styleable name="CustomTextView">
4      <attr name="customColor" format="color"/>
5      <attr name="customText" format="string"/>
6      </declare-styleable>
```


如你对文章有问题，可联系下方『文疑人QQ』。为避免他人反感，维护习社区，联系前请先阅读『提问的智

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

```
7 </resources>
8
```

name定义的是属性的名字
format定义的是属性的类型

解析属性

在CustomeTextView中解析这些属性

```
public class CustomTextView extends View {
    private int mColor = Color.RED;//默认为红色
    private String mText="I am a Custom TextView";//默认显示该文本
    private Paint mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);//画笔
    public CustomTextView(Context context) {
        super(context);
//        init();
    }
    public CustomTextView(Context context, AttributeSet attrs){
        this(context, attrs, 0);//注意不是super(context,attrs,0);
        init();
    }
    public CustomTextView(Context context, AttributeSet attrs, in
        super(context,attrs,defStyleAttr);
        TypedArray typedArray = context.obtainStyledAttributes(at
        mColor = typedArray.getColor(R.styleable.CustomTextView_c
//          如果没有判断，当没有指定该属性而去加载该属性app便会崩溃掉
        if(typedArray.getText(R.styleable.CustomTextView_customTe
            mText = typedArray.getText(R.styleable.CustomTextView
        }
        typedArray.recycle();//释放资源
        init();
    }
    private void init(){
        mPaint.setColor(mColor);// 为画笔添加颜色
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawText(mText, 100, 100, mPaint);
    }
}
```

这样解析属性便完成了

使用自定义属性

声明自定义命名空间并写入属性
想要使用自己定义的属性，我们还需要最后一步，那就是引入这些属性的命名空间的位置，在布局文件的根元素下插入这样一行代码：
xmlns:app=”http://schemas.android.com/apk/res-auto”
这行代码看不起并不像我们自己去自定义了一个命名空间，实际上也可以这么写：
xmlns:app=”http://schemas.android.com/apk/res/com.littlehan.customtextview”
在**res/**后面填写我们的包名即可。但是，在我自己Android Studio2.0

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

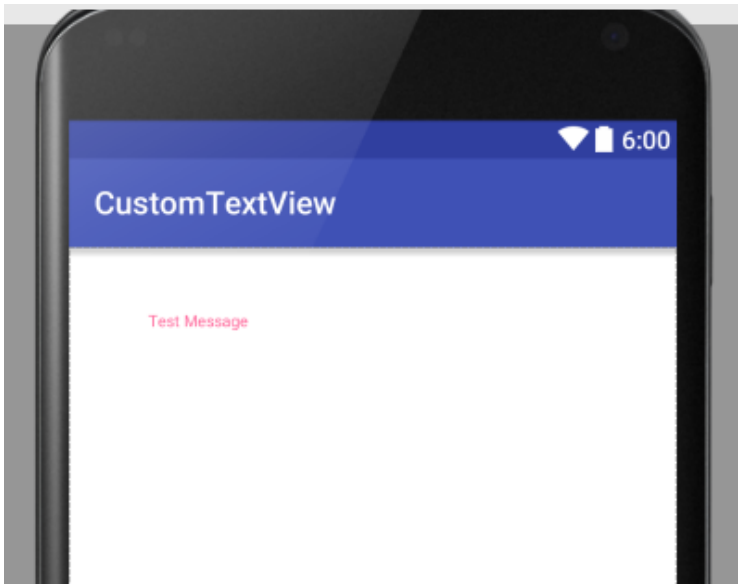
1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

上，是不推荐这么写的，所以更加建议大家还是用第一种的命名方法。
这样我们就可以通过属性的前缀来使用我们的自定义属性了:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    >
<com.littlehan.customtextview.CustomTextView
    app:customColor="@color/colorAccent"
    app:customText="Test Message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</RelativeLayout>

```

切换到视图预览窗口，可以看到自定义的属性已经生效了：



总结

在Android中，命名空间可分为3种:
1.**xmlns:android="http://schemas.android.com/apk/res/android"**
2.**xmlns:tools="http://schemas.android.com/tools"**
3.**xmlns:app="http://schemas.android.com/apk/res-auto"**
其中，1和2命名空间里的属性是系统封装好的，第3种命名空间里的属性是用户自定义的

Tagged: Android xmlns 命名空间

Comments

Pingback: 《第一行代码第二版》读书笔记（二）— 百分比布局、布局文件命名空间（xmlns）、Nine-Patch 图片 – 项目经验积累与分享

如你对文章有问题，可联系下方『文
疑人QQ』。为避免他人反感，维护
习社区，联系前请先阅读『提问的智

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发

发表评论

电子邮件地址不会被公开。 必填项已用*标注

姓名 *

电子邮件 *

站点

发表评论

谭瀚涛

点击这里可联系到我

技术之旅

目录

推荐书籍

1. [Head First Java] 深入浅出 J
2. [Thinking In Java] Java 编程
3. Android编程权威指南
4. 深入浅出 Android 开发