Introduction aux styles sous Android

par Nazim Benbourahla

Date de publication : 05/12/2011

Dernière mise à jour :

Dans ce tutoriel, vous allez apprendre comment **personnaliser votre application Android** à l'aide de **styles**.
Les **styles** vous permettent de changer l'aspect général de votre application, d'un élément (par exemple les boutons) ou d'une partie de votre application (l'apparence de tous les textes de votre application).

1 - Qu'est-ce qu'un style ?

Un **style** est une collection de propriétés qui spécifient le design d'une vue, d'un élément ou d'une application. Il peut spécifier différentes propriétés :

- Padding,
- Margin,
- Hauteur,
- Largeur,
- Couleur du texte,
- Taille du texte,
- etc…

Un **style** se définit dans un fichier ressource XML séparé du fichier XML de votre vue. Ce fichier se nomme en général **styles.xml** et se situe dans le dossier **values**.

2 - Personnaliser un bouton

Nous allons créer un projet Android qui ne se compose que d'un bouton. Voici le code correspondant à la vue :

```
<?xmlversion="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">

<Button  android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"/>

</LinearLayout>
```

Nous allons créer notre premier **style**, afin de personnaliser ce bouton. Voici le code correspondant à notre personnalisation :

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
<stylename="MyBtnStyle">
    <itemname="android:layout_width">wrap_content</item>
    <itemname="android:layout_height">wrap_content</item>
    <itemname="android:textColor">@color/textColor</item>
    <itemname="android:textSize">@dimen/textSize</item>
    <itemname="android:background">@color/btnBackground</item>
    <itemname="android:layout_marginTop">@dimen/layoutMarginTop</item>
    <itemname="android:layout_gravity">center_horizontal</item>
</style>
</resources>
```

Le code est assez simple, on créé un **style** en lui spécifiant un nom, puis on liste les différents items que l'on souhaite personnaliser dans ce **style** (hauteur, largeur, textColor, textSize, margin etc…)

Vous remarquez que les différentes couleurs et les dimensions ne sont pas mentionnées explicitement mais dans des fichiers à part. Ceci est une *best practice* dans la programmation Android.

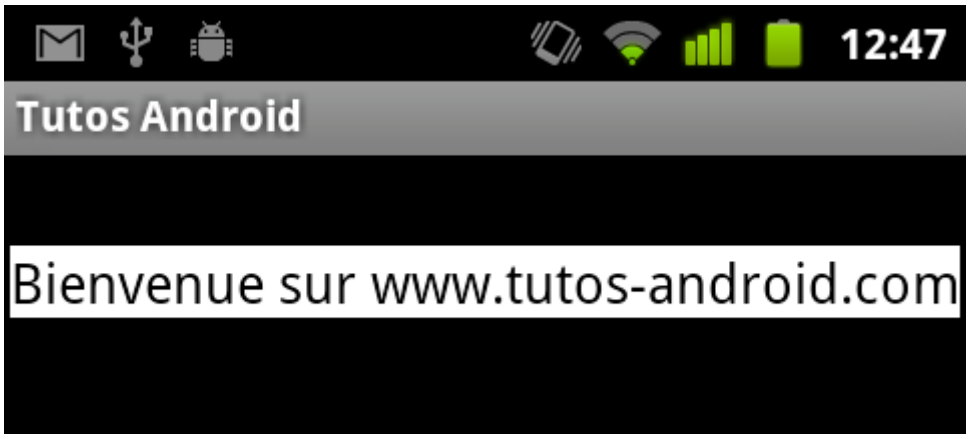
Nous allons créer deux fichiers dans le dossier **values** :

colors.xml
```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
```

colors.xml
```
    <colorname="textColor">#000</color>
    <colorname="btnBackground">#fff</color>
</resources>
```

dims.xml
```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
    <dimenname="textSize">18dp</dimen>
    <dimenname="layoutMarginTop">30dp</dimen>
</resources>
```

Voici le résultat obtenu :



De la même manière vous pouvez personnaliser un TextView, EditText ou un Button.

3 - Personnaliser le style de tous les textes

Les styles vous permettent de personnaliser par exemple tous les textes se trouvant dans votre application.

Prenons l'exemple d'une vue contenant un **TextView**, un **EditText** et **Button**. Nous allons personnaliser tous les textes qui composent cette vue.

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextViewandroid:layout_width="wrap_content"
        android:layout_height="wrap_content"android:text="@string/hello"/>

    <EditTextandroid:layout_width="wrap_content"
        android:layout_height="wrap_content"android:text="@string/hello"/>

    <Buttonandroid:layout_width="wrap_content"
        android:layout_height="wrap_content"android:text="@string/hello"/>

</LinearLayout>
```

Voici comment définir ce nouveau style pour les textes :

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
    <stylename="GreenText"parent="@android:style/TextAppearance">
        <itemname="android:textColor">@color/textColor</item>
        <itemname="android:textSize">@dimen/textSize</item>
        <itemname="android:textStyle">italic</item>
```

```
        <itemname="android:typeface">serif</item>
    </style>
</resources>
```

Puis il suffit de rajouter un attribut sur les textes dont vous souhaitez changer l'apparence.

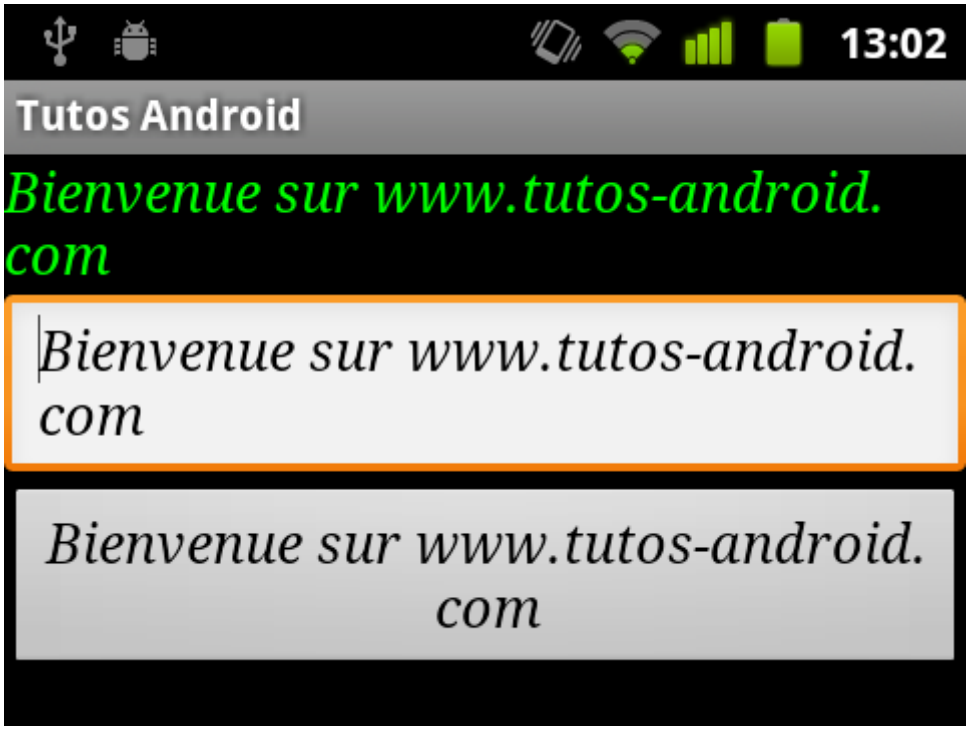
```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextViewandroid:layout_width="wrap_content"android:textAppearance="@style/GreenText"
        android:layout_height="wrap_content"android:text="@string/hello"/>

    <EditTextandroid:layout_width="wrap_content"android:textAppearance="@style/GreenText"
        android:layout_height="wrap_content"android:text="@string/hello"/>

    <Buttonandroid:layout_width="wrap_content"android:textAppearance="@style/GreenText"
        android:layout_height="wrap_content"android:text="@string/hello"/>

</LinearLayout>
```



Vous pouvez aussi surcharger une apparence déja existante :

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
    <stylename="GreenText"parent="@android:style/TextAppearance.Medium">
        <itemname="android:textColor">@color/textColor</item>
        <itemname="android:textSize">@dimen/textSize</item>
        <itemname="android:textStyle">italic</item>
        <itemname="android:typeface">serif</item>
    </style>
</resources>
```

4 - Définir un style général pour votre application

Vous pouvez surcharger le style par défaut et ainsi appliquer tout un nouveau style à votre application.

Voici un petit exemple :

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
    <stylename="MyCustomTheme"parent="android:Theme.Light.NoTitleBar">
        <itemname="android:textColor">@color/textColor</item>
        <itemname="android:textSize">@dimen/textSize</item>
        <itemname="android:textStyle">italic</item>
        <itemname="android:typeface">serif</item>
        <itemname="android:background">@color/background</item>
    </style>
</resources>
```

Dans ce cas, on peut surcharger ou non un thème déja existant sur Android.
Vous pouvez rajouter le mot NoTitleBar à la fin du thème surchargé pour supprimer la TitleBar de votre application.

Une fois votre thème défini, il suffit d'ouvrir votre **AndroidManifest.xml** et modifier votre balise comme ci-dessous :

```
<applicationandroid:icon="@drawable/icon"android:label="@string/app_name"
    android:theme="@style/MyCustomTheme">
```

Voici le résultat obtenu :

5 - Conclusion

Merci d'avoir lu ce tutoriel, en espérant qu'il vous ait facilité la compréhension du fonctionnement des styles et la personnalisation d'une application.

6 - Remerciements

Je tiens à remercier tout particulièrement **djibril** qui a mis ce tutoriel au format Developpez.com.
Merci également à **Kaera** d'avoir pris le temps de le relire et de le corriger.

7 - Liens

Tutoriel origine sur Tutos-Android