

# Android - Les Widgets

Jérémy S. Cochoy

INRIA Paris-Saclay | [jeremy.cochoy@u-psud.fr](mailto:jeremy.cochoy@u-psud.fr)

Novembre 2015

- 1 Le design
- 2 Créer un widget d'informations
  - Le layout
  - Le XML descriptif
  - Le provider
  - Le manifest
  - Réagir a un clic
- 3 Widgets de collection
  - Coming soon...
- 4 Conclusion

# La documentation



Votre nouveau livre de chevet.

*<https://developer.android.com/guide/index.html>*

# Le design



# Les types de widgets



Les différentes catégories de widgets sont :

Informations

Collection

Contrôle

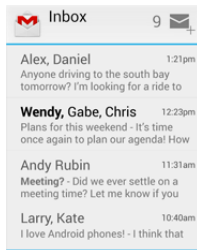
Hybride

# Widgets d'information



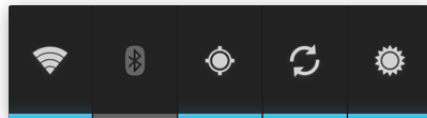
Ces widgets servent à afficher des informations utiles à l'utilisateur, et suivent leur évolution au cours du temps. De bons exemples sont les widgets météo, les horloges, les traqueurs de résultats sportifs...

# Widgets de collections



Ils sont spécialisés dans l'affichage de collections d'éléments d'un même type, comme des collections d'e-mail, de messages, d'images ou encore d'articles. Ces widgets se concentrent sur deux opérations : parcourir la collection, et ouvrir un élément de la collection pour visualiser l'information complète (contenu d'un e-mail).

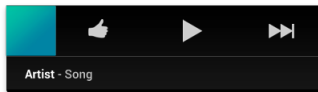
# Widgets de contrôle



Le but des widgets de contrôle est de permettre l'utilisation rapide d'une fonction très utilisée depuis l'écran d'accueil, sans avoir besoin de lancer une application. Un exemple typique proviens des lecteurs de musique, qui proposent de mettre à pause la lecture, ou de passer à la musique suivante, sans devoir ouvrir l'application de lecture.



# Widgets hybrides

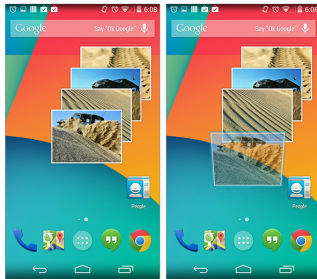


La plupart des widgets rentrent dans une des catégories précédentes, mais il peut arriver que l'emploi de fonctions provenant des différentes catégories soit nécessaire. Dans ce cas, il est recommandé de se concentrer sur l'interface sur l'un des types précédent.

## Un lecteur de musique

est avant tout un widget de contrôle, mais il permet aussi de suivre le nom du morceau joué, et utilise donc quelques composants d'un widget d'information.

# Mouvements



Du à leur contexte d'utilisation, les widgets ont des limitations technique sur la façon dont un utilisateur inter-réagira avec. Par exemple, un widget présent sur l'écran d'accueil ne peut accéder qu'aux pressions de l'utilisateur, et au slide vertical. Le slide horizontale est déjà utilisé pour naviguer entre les différents écrans d'accueil.

# Créer un widget



# Les RemoteViews

Les widgets vivent dans une application *hôte*, et il faut un mécanisme pour conserver les droits de l'application dont ils proviennent. Pour cela, leur vue est construite via des `RemoteViews`. Les `RemoteViews` conservent les droits de l'application *invitée*.

Les interfaces sont construites par un `BroadcastReceiver`, qui produit les objets `RemoteViews`. Cette objet est maintenue en vie par le système Android.

# Créer son widget

Pour créer un widget, il faut :

- Définir son layout

- Créer un fichier XML (`AppWidgetProviderInfo`) qui décrit ses propriétés

- Construire un `BroadcastReceiver` qui est utilisé pour construire l'interface du Widget.

- Déclarer le widget dans le Manifest.

- (Optionel) Ajouter une activité qui configure le widget.

# Les éléments du layout

On ne peut utiliser que certains composants graphiques.

Ne sont disponibles que les layouts :

FrameLayout

LinearLayout

RelativeLayout

# Les éléments du layout

On ne peut utiliser que certains composants graphiques.

Seulement les composants suivants :

AnalogClock

ImageView

Button

ProgressBar

Chromometer

TextView

ImageButton

# Mon Layout

Par la suite, on supposera avoir décrit l'apparence de notre widget dans un fichier `mywidget_layout.xml`.



# Les actions disponibles

## OnClickListener

La seule interaction possible avec la vue s'effectue via un `OnClickListener`. Ce listener peut être lié à un *composant graphique*, et déclenché par l'utilisateur au moment du clic sur ce *composant*.

# Description du Widget

On créer un fichier `mywidget_info.xml` dans `/res/xml/` contenant :

```
<appwidget-provider xmlns:android="http://schemas.android.com  
    /apk/res/android"  
    android:minWidth="40dp"  
    android:minHeight="40dp"  
    android:updatePeriodMillis="86400000"  
    android:previewImage="@drawable/preview"  
    android:initialLayout="@layout/mywidget_layout"  
    android:resizeMode="horizontal|vertical"  
    android:widgetCategory="home_screen">  
</appwidget-provider>
```

# Description du Widget

Si l'on veut une activité de configuration :

```
<appwidget-provider xmlns:android="http://schemas.android.com  
    /apk/res/android"  
    ...  
    android:configure="com.example.android.MyWidgetConfigure"  
    >  
</appwidget-provider>
```

Cette ligne indique la classe correspondant à l'activité qui doit être exécutée.

# Description du Widget

## Description des attributs.

`minWidth` and `minHeight` contrôlent la taille minimale,  
`updatePeriodMillis` définit le temps de rafraichissement,  
`initialLayout` indique le layout de votre widget,  
`reviewImage` indique l'image qui sera affiché dans la liste des widgets,  
`resizeMode` indique dans quels directions l'on peut redimensionner le widget,  
`widgetCategory` indique dans quels zone le widget peut être placé (`home_screen` et `keyguard`).

# La classe provider

On ajoute une nouvelle classe `MyWidgetProvider`, chargée de construire les vues.

Notre classe provider :

```
public class MyWidgetProvider extends AppWidgetProvider {  
    public void onUpdate(Context context, AppWidgetManager  
        appWidgetManager, int[] appWidgetIds) {  
        //Effectue la mise a jour de la vue  
    }  
}
```

# La fonction `onUpdate`

Les arguments de cette fonction sont :

`context` Informations sur votre application,  
`appWidgetManager` Le gestionnaire de widgets,  
`appWidgetIds` La liste de toute les instances de notre widget.

# La fonction onUpdate

## Implémentation de onUpdate

```
// On parcourt tous les widgets.
final int N = appWidgetIds.length;
for (int i = 0; i < N; i++) {
    int appWidgetId = appWidgetIds[i];

    // Construit une nouvelle vue.
    RemoteViews views = new RemoteViews(context.
        getPackageName(), R.layout.mywidget_layout);

    // Change le contenu du champ text textField.
    views.setTextViewText(R.id.textField, "Blabla");

    // Met a jour la vue avec notre nouvelle vue.
    appWidgetManager.updateAppWidget(appWidgetId, views);
}
```

On déclare le provider dans le block `<application></application>`.

```
<receiver android:name="MyWidgetProvider" >
  <intent-filter>
    <action android:name="android.appwidget.action.
      APPWIDGET_UPDATE" />
  </intent-filter>

  <meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/widget_info" />
</receiver>
```



Pour me contacter : [jeremy.cochoy@u-psud.fr](mailto:jeremy.cochoy@u-psud.fr), merci et à bientôt.

