

# Android - Les Widgets

Jérémy S. Cochoy

INRIA Paris-Saclay | [jeremy.cochoy@gmail.com](mailto:jeremy.cochoy@gmail.com)

Novembre 2015

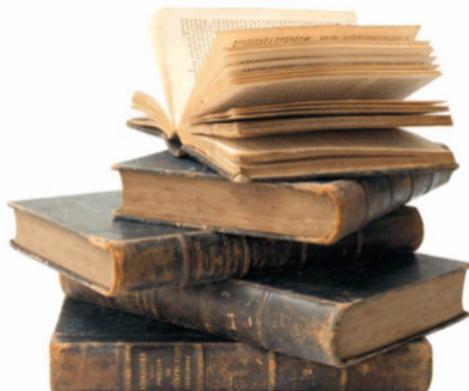
## 1 Le design

- Widget d'informations
- Widgets de collection
- Widget de contrôle
- Widget de hybride

## 2 Créer un widget d'informations

- Le layout
- Le XML descriptif
- Le provider
- Le manifest
- Réagir a un clic
- Configurer un widget

## 3 Conclusion



Votre nouveau livre de chevet.

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

## Le design



# Les types de widgets



Les différentes catégories de widgets sont :

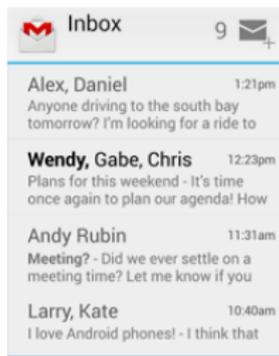
- Informations
- Collection
- Contrôle
- Hybride

# Widgets d'information



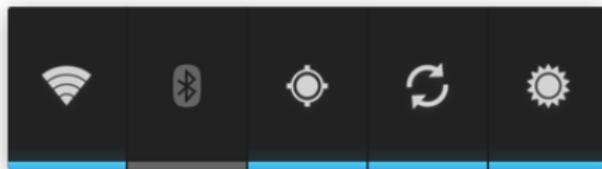
Ces widgets servent à afficher des informations utiles à l'utilisateur, et suivent leur évolution au cours du temps. De bons exemples sont les widgets météo, les horloges, les traqueurs de résultats sportifs...

# Widgets de collection



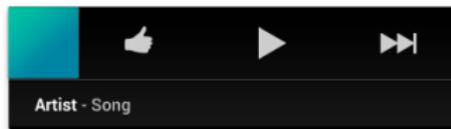
Ils sont spécialisés dans l'affichage de collections d'éléments d'un même type, comme des collections d'e-mails, de messages, d'images ou encore d'articles. Ces widgets se concentrent sur deux opérations : parcourir la collection, et ouvrir un élément de la collection pour visualiser l'information complète (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 provient des lecteurs de musique, qui proposent de mettre en 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 de différentes catégories soit nécessaire. Dans ce cas, il est recommandé de se concentrer sur l'interface d'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 lu, et utilise donc quelques composants d'un widget d'information.

# Mouvements



Du à leur contexte d'utilisation, les widgets sont techniquement limités pour ce qui est de leurs interactions avec l'utilisateur. Par exemple, un widget présent sur l'écran d'accueil ne peut réagir qu'aux pressions de l'utilisateur et au slide vertical. Le slide horizontal est déjà utilisé pour naviguer entre les différents écrans d'accueil.

# Créer un widget



# Les RemoteViews



## 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*.

## BroadcastReceiver

Les interfaces sont construites par un `BroadcastReceiver` qui produit les objets `RemoteViews`. Cet objet est maintenu 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 containers.

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
- Button
- Chromometer
- ImageButton
- ImageView
- ProgressBar
- TextView

# 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 crée 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 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` et `minHeight` contrôlent la taille minimale,
- `updatePeriodMillis` définit le temps de rafraichissement,
- `initialLayout` indique le layout de votre widget,
- `reviewImage` indique l'image qui sera affichée dans la liste des widgets,
- `resizeMode` indique dans quelles directions on peut redimensionner le widget,
- `widgetCategory` indique dans quelle 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 toutes 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/mywidget_info" />
</receiver>
```

# L'intent ACTION\_APPWIDGET\_UPDATE

## Une pending intent

C'est intention qui est conservé un certain laps de temps avant d'être envoyé au système. Les données qu'elle contient sont conservées aussi longtemps que l'intent existe.

## ACTION\_APPWIDGET\_UPDATE

L'intention ACTION\_APPWIDGET\_UPDATE permet de mettre à jour la vue d'un widget en provoquant l'appel à `onUpdate`.

## Lier un clic à une mise à jours (dans onUpdate)

```
// Creation de notre intention
Intent intent = new Intent();

//Intent de type mise a jour de widget
intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
//On dit quel widget on veut mettre a jour
intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
    appWidgetIds);

//Si l'intent existe deja, on la met juste a jour.
PendingIntent pendingIntent = PendingIntent.getBroadcast(
    context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
remoteViews.setOnClickPendingIntent(R.id.button,
    pendingIntent);
```

# Activité de configuration



# L'activité de configuration

L'activité sera lancée par une intent

```
<activity android:name=".ExampleAppWidgetConfigure">
  <intent-filter>
    <action android:name="android.appwidget.action.
      APPWIDGET_CONFIGURE"/>
  </intent-filter>
</activity>
```

Et elle doit aussi être spécifiée au provider

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

# Bon à savoir...

- L'application de configuration doit toujours renvoyer un résultat. Ce résultat doit inclure l'ID du widget obtenu par l'application de configuration à travers le champ `EXTRA_APPWIDGET_ID`.
- La fonction `onUpdate` n'est pas appelée lors de la création du widget. Ce sera à l'application de configuration de provoquer son appel.

# Step by step...

## 1 - Récupérer l'ID de l'App Widget

```
Intent intent = getIntent();
Bundle extras = intent.getExtras();
if (extras != null) {
    mAppWidgetId = extras.getInt(
        AppWidgetManager.EXTRA_APPWIDGET_ID,
        AppWidgetManager.INVALID_APPWIDGET_ID);
}
```

## 2 - Configuration

Faites ce que vous avez à faire dans votre activité.

# Step by step...

## 3 - Récupérer le manager

```
AppWidgetManager appWidgetManager = AppWidgetManager.  
    getInstance(context);
```

## 4 - Mettre à jour le widget

```
RemoteViews views = new RemoteViews(context.getPackageName(),  
R.layout.example_appwidget);  
appWidgetManager.updateAppWidget(mAppWidgetId, views);
```

# Step by step...

## 5 - Ajouter le résultat et terminer l'activité

```
Intent resultValue = new Intent();  
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,  
    mAppWidgetId);  
setResult(RESULT_OK, resultValue);  
finish();
```

### Conseil :

Créez l'intent de résultat dès le début de l'activité, avec la valeur `RESULT_CANCELED`, et liez-la avec `setResult()`. De cette façon, si l'utilisateur quitte l'activité de configuration avec le bouton back, le widget ne sera pas créé.



## Widgets

Avec un widget, vous pouvez enrichir vos applications et proposer à vos utilisateurs le suivi d'un monitoring sportif, l'affichage de données agrégées, les derniers messages reçus sur une application sociale, etc.

Pour me contacter : [jeremy.cochoy@gmail.com](mailto:jeremy.cochoy@gmail.com), merci et à bientôt.

