Widgets

Chantal Keller

6 novembre 2015

Plan

Quelques précisions

Remarques :

- une préférence a plusieurs états : agir lorsque l'utilisateur coche la case **et** lorsqu'il la décoche
- penser à factoriser le code (ex : entre `onStart` et `onSharedPreferenceChanged`)
- la méthode `PreferenceManager.setDefaultValues` lit les valeurs par défaut du fichier XML, ni plus ni moins :
  - n'efface pas la sauvegarde
  - n'applique pas les préférences

La question 3

Remettre les préférences à leurs valeurs par défaut :

- `PreferenceManager.setDefaultValues` pour lire les valeurs par défaut du fichier XML
- mais il faut aussi réinitialiser le fichier de préférences :

```
PreferenceManager.getDefaultSharedPreferences(this).edit().clear().commit();
```

- et appliquer ces préférences par défaut

↪ voir une correction possible en ligne
↪ voir TD8 pour `edit()`

Plan

Widget

"Morceau d'application" sur le bureau :

- donne un aperçu, des informations
- permet de gérer les aspects les plus utiles de l'application
- permet de la contrôler en partie

But :

- accès direct
- visibilité

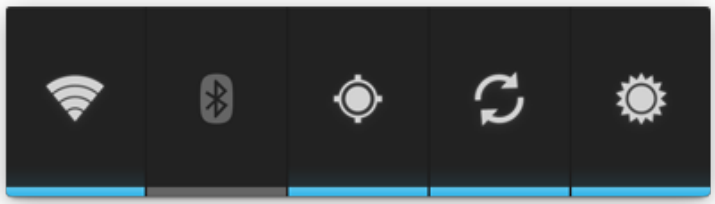
↪ façade de l'application

Widgets informatifs

©Android

- informations importantes et variables au cours du temps
- ex : météo, heure, chronomètre. . .

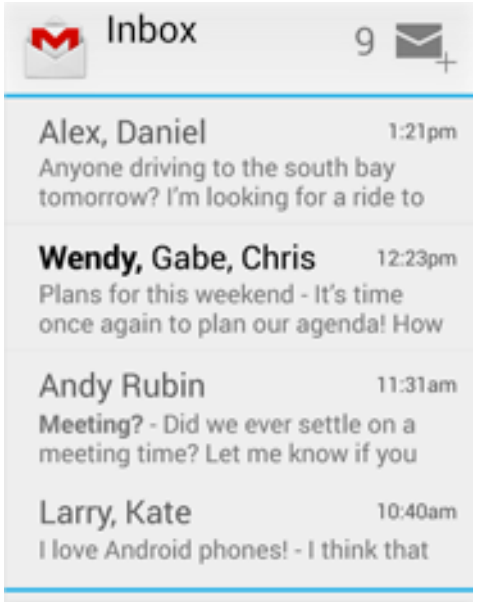
Widgets de contrôle



©Android

- accès direct aux contrôles d'une application ("télécommande")
- ex : lancement de musique ou de vidéo, du wifi...

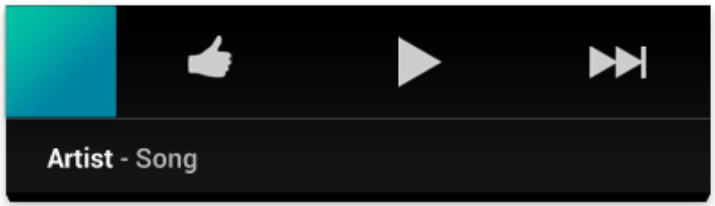
Widgets de navigation



©Android

- navigation dans une collection d'éléments
- ex : mails, photos, liste d'applications...

Widgets hybrides



©Android

- affichage, contrôle et navigation
- ex : lecteur de musique avec affichage du titre courant et choix de la liste de lecture

Ça ne remplace pas une application

Tout est réduit :

- taille (souvent plusieurs widgets sur le bureau)
- interaction avec l'utilisateur : moins de gestes
- possibilités : c'est l'application du bureau qui gère l'affichage

Choix du contenu

"Vitrine" de l'application :

- attractif
- simple mais utile
- accès rapide aux parties les plus utiles de l'application

Choix du design

Plus ou moins de place :

- orientation du widget
- faire une interface qui s'adapte aux différents écrans
- possibilité d'avoir plusieurs formats pour un même widget (ex de la météo : nombre de jours) : l'utilisateur règle selon sa place et ses besoins

↪ faire des widgets configurables

Plan

Une interface programmable

Interface :

- fichier XML habituel
- attention au fait que ça va être sur un petit rectangle

Programmation :

- code Java pour donner vie à l'interface
- cycle de vie, . . . etc

Compléments

Un fichier XML de méta-données :

- taille minimale, possibilités d'agrandissement, ...
- période de rafraîchissement (30min minimum)
- lien vers le fichier XML d'interface

Déclaration dans le manifeste :

- balise `receiver`
- capable de se mettre à jour (`android.appwidget.action.APPWIDGET_UPDATE`)

Interface

Fichier XML habituel, mais :

- moins de vues possibles, avec moins de paramétrisation
- on n'accède pas aux vues individuellement mais via un **ensemble de vues** (`RemoteViews`)
- gestion des évènements (ex : clic sur un bouton) via la **diffusion d'intentions**

↪ cause : interface destinée à être gérée par une autre application

Programmation : la classe `AppWidgetProvider`

Le système appelle la méthode `onUpdate` :

- lorsque l'utilisateur **installe** le widget :
  - mettre un écouteur sur un bouton
  - charger des préférences
- à chaque **rafraîchissement** :
  - lire des données (fichier, réseau, ...)
  - actualiser l'affichage en conséquence
  - remarque : rafraîchissement au maximum toutes les 30min, coûteux en énergie

↪ surcharger cette méthode pour faire les actions dont on a besoin à l'installation et lors du rafraîchissement

Mise à jour de l'interface d'un widget

```
// On récupère l'ensemble de vues
RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget);

// On effectue les modifications que l'on souhaite
views.setTextViewText(R.id.textView, "bonjour");

// On valide les modifications
AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
ComponentName watchWidget = new ComponentName(context, Widget.class);
appWidgetManager.updateAppWidget(watchWidget, views);
```

Remarques :

- pas un accès direct aux vues
- les objets `appWidgetManager` et (une variante de) `watchWidget` font partie des arguments de `onUpdate` : inutile de les redéfinir dans ce cas

Interaction entre widget et application

Ils ne tournent pas dans la même application :

- communication via des intentions
- par *diffusion* :
  - l'émetteur diffuse l'intention en précisant la classe destinataire
  - celle-ci implante un écouteur pour pouvoir réagir
- remarque : s'applique à la communication entre applications en toute généralité

Du côté de l'émetteur

```
// Création de l'intention: contexte courant, classe visée
Intent intent = new Intent(this, Widget.class);

// On peut lui associer des données
intent.putExtra(EXTRA_DATA, donnee);
...

// On précise l'action
intent.setAction(EFFECTUER_TELLE_ACTION);

// On la diffuse
sendBroadcast(intent);
```

Remarque :

- `EXTRA_DATA` et `EFFECTUER_TELLE_ACTION` sont des variables `final static String`

Du côté du récepteur

Doit hériter de `BroadcastReceiver` :

```
@Override
public void onReceive(Context context, Intent intent) {
    // Appel à la méthode de la super-classe
    super.onReceive(context, intent);

    // On regarde l'action demandée
    if (intent.getAction().equals(EFFECTUER_TELLE_ACTION)) {
        // On récupère les données
        String donnee = intent.getStringExtra(EXTRA_DATA, "");
        ...

        // On effectue l'action
        ...
    } else ...
}
```

Remarques :

- la classe `AppWidgetProvider` hérite de `BroadcastReceiver`
- utiliser les alternatives de `getStringExtra` en fonction du type de `donnee`

Conclusion

Design :

- vitrine d'une application
- accès direct aux informations et aux contrôles les plus utiles
- ne remplace pas une application

Technique :

- avoir en tête que ça ne tourne pas dans l'application qui fournit le widget
- échange d'informations par la diffusion d'intentions
- interface un peu plus restreinte

Plan

TD8

Objectifs :

- widget communiquant avec son application
- persistance des données via les préférences partagées

↪ vous êtes encouragés à en faire le maximum possible et vous pouvez me le rendre si vous souhaitez un retour