Travaux Dirigés et Pratiques de Programmation Android n° 7

Widgets

Les objectifs de ce TD sont :

- La mise en place d'un widget ;
- La mise à jour de l'affichage du widget lors d'événements ;
- La communication via des intentions asynchrones ;
- La persistance d'un petit jeu de données.

Pour cela, nous allons créer un widget affichant sur le bureau nos score et niveau actuels à notre jeu « Click the Button ». Nous allons également pouvoir les sauvegarder d'une session à l'autre.

Mise en place

Reprenez votre code des TP5 et TP6.

Notre premier widget

Nous allons associer un widget à cette application, et décrire son interface.

- Dans l'arborescence de l'application sous Android Studio, faire un clic droit sur `app` et aller dans `New/Widget/App Widget`. Choisissez le nom que vous voulez (par exemple, `TP7AppWidget`) et regarder les différentes options possibles, sans les modifier.
- Installer l'application sur la tablette ou l'émulateur et attacher le widget au bureau (pour cela, appuyer longtemps sur une partie vide du bureau, et choisir `Widgets` puis le widget du TP7 dans la liste).

Observer les modifications opérées et les fichiers créés par Android Studio :

- Le fichier `manifests/AndroidManifest.xml` déclare un nouveau receveur correspondant à notre widget, avec comme action possible `APPWIDGET_UPDATE`, car le widget peut être régulièrement mis à jour par le système.
- Le fichier `res/xml/tp7_app_widget_info.xml` contient les méta-données du widget : ce sont notamment les options que l'on a choisies lors de la création du widget, ainsi que d'autres données, comme par exemple l'attribut `android:updatePeriodMillis` qui donne en millisecondes la période de rafraîchissement du widget (qui est au minimum de 1800000ms, à savoir 30min).
- Le fichier `res/layout/tp7_app_widget.xml` décrit comme toujours l'interface du widget.
- Le fichier `java/TP7AppWidget` est la partie Java du widget, héritant de la classe `AppWidgetProvider`. Par défaut, Android Studio surcharge trois méthodes :
  - La méthode `onUpdate` est appelée à l'installation du widget sur le bureau et à chaque rafraîchissement du widget. Elle sert donc à mettre en place des évènements lorsque l'utilisateur installe le widget, et à effectuer des actions lors des rafraîchissements ; malheureusement, nous n'allons pas pouvoir tester ce deuxième point pendant le TP, car cela se recharge au maximum toutes les 30min (mais je vous invite à tester chez vous). On peut commenter cette méthode (et celle qu'elle appelle) pour l'instant.
  - Les méthodes `onEnabled` et `onDisabled` sont appelées respectivement lorsque le premier widget de ce type est ajouté au bureau et lorsque le dernier widget de ce type est supprimé du bureau. On ne s'en servira pas dans ce TP.

Modifier l'interface du widget, de manière à avoir quatre zones de texte affichant comme d'habitude :
Score : 0
Niveau : 1

Mise à jour du widget

Pour l'instant, lorsqu'on joue à « Click the Button », le widget n'est pas modifié, ce qui ne permet pas de connaître notre score.

Diffusion d'une intention

- Dans la méthode appelée lorsqu'on clique sur le bouton, créer une intention destinée à la classe `TP7AppWidget` et lui associer le score et le niveau (souvenez-vous du TP3).
- Associer également une action à cette intention : c'est une chaîne de caractères non mutable (`public final static String`) identifiant l'action à effectuer.
- Diffuser cette intention.

Réception d'une intention

Dans la classe `TP7AppWidget`, surcharger la méthode `onReceive` afin de :

- Faire appel à la méthode de la super-classe.
- Tester si l'action de l'intention est celle qui nous intéresse.

Lorsque c'est le cas :

- Récupérer le score et le niveau de l'intention (toujours comme au TP3).
- Effectuer la mise à jour de la façon suivante :

```
// Récupération des vues du widget : on n'a accès qu'à quelque chose de restreint
// (RemoteViews) car ce n'est pas nous qui gérons directement l'interface du widget,
// mais le bureau sur lequel il est
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.tp7_app_widget);

// On modifie les zones du texte en fonction du score et du niveau (remplacer les ids
// et les variables par ceux que vous aurez choisis) : on ne peut pas accéder
// directement au vues avec RemoteViews, mais en contrepartie, on a des méthodes
// effectuant les modifications sur les vues
views.setTextViewText(R.id.textViewScore, "" + score);
views.setTextViewText(R.id.textViewLevel, "" + level);

// On lance la mise à jour du widget
AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
ComponentName watchWidget = new ComponentName(context, TP7AppWidget.class);
appWidgetManager.updateAppWidget(watchWidget, views);
```

en remplaçant les noms des variables `score`, `level`, `R.id.textViewScore` et `R.id.textViewLevel` par ceux que vous avez choisis.

Tester.

Remarque : plus généralement, la diffusion d'intentions est le moyen pour les applications de communiquer entre elles : il suffit pour cela que la classe concernée de l'application réceptrice hérite de `BroadcastReceiver` et surcharge la méthode `onReceive`.

À vous de jouer

1. Arrêter et relancer l'émulateur (ou éteindre et allumer la tablette). Que se passe-t-il au niveau du widget ? Faites en sorte qu'il retrouve les bonnes valeurs au lancement de l'application.
2. Surcharger la méthode `onUpdate` pour que l'application se lance lorsqu'on clique sur le widget.

Pour aller plus loin

Les widgets offrent d'autres possibilités, vous pouvez par exemple :

- tester la mise à jour périodique du widget ;
- fournir un menu de préférences au widget.