Chapitre 7
La persistance des données

1. Introduction

Ce chapitre a pour but de présenter la persistance des données sous Android.

Les données persistantes d'une application sont les données sauvegardées avant la fermeture de l'application de telle sorte qu'elles puissent être restaurées ultérieurement.

Android propose plusieurs mécanismes permettant de gérer la persistance de données, selon la nature de ces données. Nous découvrirons les fichiers de préférences, les fichiers standards et les bases de données.

Nous terminerons par les fournisseurs de contenus qui, au-delà de la persistance des données, proposent un mécanisme de partage de données entre les applications.

2. Fichiers de préférences

Android fournit un framework simple pour sauvegarder et restaurer des données de types primitifs. Ces données sont sauvegardées dans des fichiers au format XML sous la forme d'associations clés-valeurs. Ces fichiers sont appelés fichiers de préférences.

Nous allons étudier dans un premier temps comment cibler un fichier de préférences, puis comment le lire et y écrire des données. Nous terminerons par décrire comment supprimer toutes ou partie des données de ces fichiers.

Remarque

Le système Android permet d'afficher et de sauvegarder les préférences générales de l'utilisateur. Toute application peut adopter la même fonctionnalité et le même affichage. La hiérarchie des préférences proposées peut être réalisée directement dans un fichier XML. L'implémentation d'un tel écran de préférences est réalisée en dérivant de la classe `PreferenceActivity`*. Depuis Android 3.0 (API 11), cette classe fonctionne de concert avec les fragments de type* `PreferenceFragment` *afin de pouvoir, entre autres, afficher côte à côte les titres des sections et les préférences qu'elles proposent.*

2.1 Cibler le fichier

Par défaut, un fichier de préférences est associé à l'activité qui le crée. Ce fichier porte automatiquement le nom qualifié entier de l'activité concernée, par exemple `fr.mondomaine.android.monappli.prefsFic1.xml`.

La création et la gestion du fichier de préférences sont réalisées au travers d'un objet de type `SharedPreferences` retourné par la méthode `getPreferences` de la classe `Activity`.

Syntaxe

```
public SharedPreferences getPreferences (int mode)
```

Cette méthode prend en paramètre le mode d'accès à assigner au fichier lors de sa création. Les valeurs possibles pour ce paramètre sont :

- `Context.MODE_PRIVATE` : mode privé. C'est le mode par défaut. Le fichier ne peut être lu et écrit que par l'application courante, ou une application partageant le même identifiant utilisateur.
- `Context.MODE_WORLD_READABLE` : les autres applications peuvent lire le fichier.
- `Context.MODE_WORLD_WRITEABLE` : les autres applications peuvent modifier le fichier.

Exemple

```
SharedPreferences prefs = getPreferences(Context.MODE_PRIVATE);
```

Il est également possible de spécifier explicitement un autre nom de fichier. Cela permet de créer plusieurs fichiers de préférences. Pour cela, il faut utiliser la méthode `getSharedPreferences` en spécifiant le nom du fichier en premier paramètre.

Syntaxe

```
public abstract SharedPreferences getSharedPreferences (String
name, int mode)
```

Exemple

```
SharedPreferences prefs =
   getSharedPreferences("nomFichierPrefs1.xml",
     Context.MODE_PRIVATE);
```

2.2 Lecture

Les données contenues dans un fichier de préférences sont enregistrées sous forme d'associations clés-valeurs. Une telle association est composée :

– D'une clé qui est une chaîne de caractères de type `String`.
– D'une valeur de type primitif : `boolean` (booléen), `float` (nombre à virgule flottante), `int` ou `long` (entiers) ou `String` (chaîne de caractères).

Pour lire les données contenues dans un fichier de préférences, on utilise l'objet de type `SharedPreferences` récupéré précédemment. On invoque ensuite certains de ses accesseurs permettant de lire individuellement une donnée selon son type.

Syntaxe

```
public abstract boolean getBoolean (String key, boolean defValue)
public abstract float getFloat (String key, float defValue)
public abstract int getInt (String key, int defValue)
public abstract long getLong (String key, long defValue)
public abstract String getString (String key, String defValue)
```

Le premier paramètre est le nom de la clé. Le second paramètre est la valeur par défaut à retourner si la clé n'existe pas.

Exemple

```
boolean modeWifi = prefs.getBoolean("modeWifi", false);
int compteur = prefs.getInt("compteur", 0);
String commentaire = prefs.getString("commentaire", "");
```

On peut aussi récupérer toutes les données d'un seul coup en utilisant la méthode `getAll`.

Syntaxe

```
public abstract Map<String, ?> getAll ()
```

Exemple

```
Map<String, ?> valeurs = prefs.getAll();
Boolean modeWifi = (Boolean)valeurs.get("modeWifi");
```

La méthode `contains` de l'objet `SharedPreferences` permet de vérifier la présence d'une clé donnée qu'on lui spécifie en paramètre.

Syntaxe

```
public abstract boolean contains (String key)
```

Exemple

```
if (prefs.contains("modeWifi")) {
   traitement();
}
```

2.3 Écriture

L'écriture de données dans un fichier de préférences se fait via un objet de type `SharedPreferences.Editor`. Cet objet est retourné par la méthode `edit` appelée sur l'objet de type `SharedPreferences` récupéré précédemment.

Syntaxe

```
public abstract SharedPreferences.Editor edit ()
```

Exemple

```
SharedPreferences.Editor editeur = prefs.edit();
```

L'objet `Editor` ci-dessus permet de spécifier les nouvelles données ou de modifier les données existantes en les écrasant avec les nouvelles. On invoque ses méthodes permettant d'écrire individuellement une association clé-valeur. À l'instar des méthodes de lecture, il existe une méthode d'écriture par type primitif. Ces méthodes prennent en paramètres le nom de la clé ainsi que la valeur de la donnée.

Syntaxe

```
public abstract SharedPreferences.Editor putBoolean (String key,
  boolean value)
public abstract SharedPreferences.Editor putFloat (String key,
  float value)
public abstract SharedPreferences.Editor putInt (String key,
  int value)
public abstract SharedPreferences.Editor putLong (String key,
  long value)
public abstract SharedPreferences.Editor putString (String key,
  String value)
```

Exemple

```
editeur.putBoolean("modeWifi", true);
editeur.putInt("compteur", 42);
editeur.putString("commentaire", "Ceci est un commentaire");
```

L'écriture des données ne sera effectivement réalisée dans le fichier qu'une fois la méthode `commit` de l'objet `Editor` appelée.

Syntaxe

```
public abstract boolean commit ()
```

Exemple

```
editeur.commit();
```

Remarque

Attention à ne pas oublier d'appeler la méthode `commit`. Car sans cet appel, l'objet `Editor` ne sert à rien ; les modifications qu'il contient n'étant pas enregistrées.

2.4 Suppression

La suppression des données contenues dans un fichier de préférences se fait en utilisant l'objet `Editor` de type `SharedPreferences.Editor` comme pour l'écriture des données.

La méthode `remove` de l'objet `Editor` permet de supprimer une association clé-valeur. On spécifie le nom de la clé en paramètre.

Syntaxe

```
public abstract SharedPreferences.Editor remove (String key)
```

Exemple

```
editeur.remove("modeWifi");
```

La méthode `clear` permet de supprimer toutes les données, c'est-à-dire toutes les associations clés-valeurs.

Syntaxe

```
public abstract SharedPreferences.Editor clear ()
```

Exemple

```
editeur.clear();
```

Comme pour l'écriture, il faut appeler la méthode `commit` pour enregistrer les modifications.

Il est également possible d'enchaîner les modifications puisque les méthodes de l'objet `Editor` retournent cet objet.

Exemple

```
editeur.clear().putBoolean("modeWifi", modeWifi).commit();
```

Remarque

Lors de l'appel à la méthode `commit`, la méthode `clear` est exécutée en premier quelle que soit la position de son appel.

On peut donc par exemple réécrire la ligne précédente sans en modifier le résultat.