Institut Supérieur d'Informatique

Université de Tunis el Manar

TP4 : Stockage de données

Programmation Mobile – 2ème Licence – Systèmes Embarqués

Année Universitaire : 2011/2012

MME. LILIA SFAXI

TP4 : Stockage de données

Programmation Mobile

Objectifs du TP

Ce TP a pour objectif de vous initier au stockage des données dans le téléphone Android, dans la mémoire interne du téléphone et dans une carte de stockage externe.

I. Stockage des données dans la mémoire interne

Une application peut stocker ses données dans la mémoire interne du téléphone. Par défaut, ces données sont *privées*, c'est à dire qu'elles ne sont pas accessibles à partir d'autres applications. A la suppression de l'application, ces données sont supprimées.

Pour chaque application, le système Android crée un répertoire qui s'appelle : « */data/data/package_de_l'application* ». Les fichiers internes sont stockés dans le répertoire *files* contenu dans ce répertoire.

I. 1. Lecture et écriture des données à partir d'un fichier

Les primitives pour la manipulation des fichiers en Java se déclinent en plusieurs types. On distingue principalement les primitives pour la lecture, et les primitives pour l'écriture.

I. 1. 1. Primitives de lecture

Il existe principalement deux types de primitives de lecture : les primitives de type *InputStream* et les primitives de type *Reader*. Ces deux primitives permettent de lire un flux de données à partir d'une source en entrée, mais la principale différence entre les deux, c'est que le *InputStream* lit un flux d'octets, tandis que le *Reader* lit un flux de caractères.

A partir d'un *InputStream*, il est possible de créer un *Reader* en utilisant la classe ***InputStreamReader***. Il est bien entendu plus facile de gérer un flux de caractères dans notre cas, puisque nous désirons lire des chaînes de caractères à partir d'un fichier. Nous utiliserons donc principalement les primitives de type *Reader*, et en particulier le ***FileReader***, qui est un *InputStreamReader* particulier, qui lit un flux de caractères à partir d'un fichier.

Enfin, pour faciliter la manipulation de ces flux, nous allons les envelopper dans un ***BufferedReader***. Les primitives de type *bufferisé* n'ajoutent pas de fonctionnalités supplémentaires, mais augmentent l'efficacité d'utilisation en fournissant des primitives facilitant l'utilisation des flux, comme par exemple *readLine()*.

I. 1. 2. Primitives d'écriture

Les primitives d'écriture duales à ***InputStreamReader***, ***FileReader*** et ***BufferedReader*** sont respectivement ***OutputStreamWriter***, ***FileWriter*** et ***BufferedWriter***.

I. 2. Création ou modification d'un fichier

Pour créer un fichier, ou modifier un fichier existant, la méthode prédéfinie *openFileOutput* est utilisée. En appelant cette méthode, il faut spécifier le nom du fichier, et ses modes d'ouverture, qui sont:

- *MODE_PRIVATE* : Le fichier n'est accessible que par l'application qui l'a créé.
- *MODE_WORLD_READABLE* : Le fichier est accessible **en lecture** par les autres applications.
- *MODE_WORLD_WRITEABLE* : Le fichier est accessible **en écriture** par les autres applications.
- *MODE_APPEND* : Si le fichier existe déjà, les données seront ajoutées à la fin.

Indication : **Il est possible de définir plusieurs modes à la fois en les séparant par un |.**

Une méthode simple pour stocker des données dans un fichier est d'utiliser le code suivant :

```
try {
        // ouverture (ou création) du fichier pour modification en mode privé
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
                                openFileOutput("nom_fichier",Context.MODE_PRIVATE)));
        // écriture de la chaîne de caractère dans le fichier
        bw.write("chaine_a_ecrire");
        // fermeture du fichier
        bw.close();
} catch (Exception e) {
        // Si une erreur existe, l'afficher dans un Toast
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
}
```

TAF-1 **:** Créer une application Android appelée *Stockage*.

L'activité principale contient un champ de saisie et un bouton appelé *Ecrire Interne*. En cliquant sur le bouton, le contenu du champs de saisie sera stocké dans un fichier appelé : « saisie.txt ».

Nous désirons que ce fichier soit accessible **en lecture** par les autres applications, et que la chaîne entrée soit **ajoutée à la fin du fichier**.

I. 3. Lecture du contenu d'un fichier

Le code suivant permet de lire le contenu d'un fichier, et de le stocker dans une chaîne de caractères.

```
try {
        // ouverture du fichier pour lecture
        BufferedReader br = new BufferedReader(new InputStreamReader(
                                openFileInput("nom_du_fichier")));
        // line est une variable qui stocke le contenu d'une ligne
        String line;
        // StringBuffer contient des caractères qui peuvent être modifiés
        StringBuffer buffer = new StringBuffer();
        // lecture des lignes du fichier et leur sauvegarde dans le StringBuffer
        while ((line = br.readLine()) != null) {
                buffer.append(line);
        }
        // fermeture du Reader
        br.close();
} catch (Exception e) {
        // Si une erreur existe, l'afficher dans un Toast
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
}
```

TAF-2 **:** Ajouter un autre bouton à votre activité, appelé *Lire Interne*. Le clic sur ce bouton affiche le contenu du fichie *saisie.txt* dans un Toast.

I. 4. Accès au fichier à partir d'une autre application

Il est possible d'accéder à un fichier créé dans une application à partir d'une autre application. Le fichier doit respecter les conditions suivantes :

- Il doit être créé avec le mode *MODE_WORLD_READABLE* pour être accessible en lecture
- Il doit être créé avec le mode *MODE_WORLD_WRITEABLE* pour être accessible en écriture

Pour cela, on ne doit pas appeler le fichier directement par son nom dans la deuxième application, car le système va le chercher dans le répertoire associé à cette application. Au lieu de cela, il faut ajouter une référence au *package* de l'application à laquelle appartient le fichier.

Par exemple, si le fichier *fich.txt* a été crée par l'application dont le package est *pack.app1*, alors remplacer : `openFileOutput("fich.txt")` par `createPackageContext("pack.app1", 0).openFileOutput("fich.txt")` et garder le reste du code identique, pour l'écriture ou la lecture du fichier.

TAF-3 **:**

- Créer une deuxième application, appelée *AccesStockage*.
- Modifier le fichier *saisie.txt* créé dans la première application en y ajoutant la ligne « ***Accessible d***

l'extérieur »

- Exécuter la première application, et afficher le contenu du fichier. Y trouvez-vous la chaîne ajoutée ? Si nor pourquoi ?

II. Stockage des données dans une mémoire externe

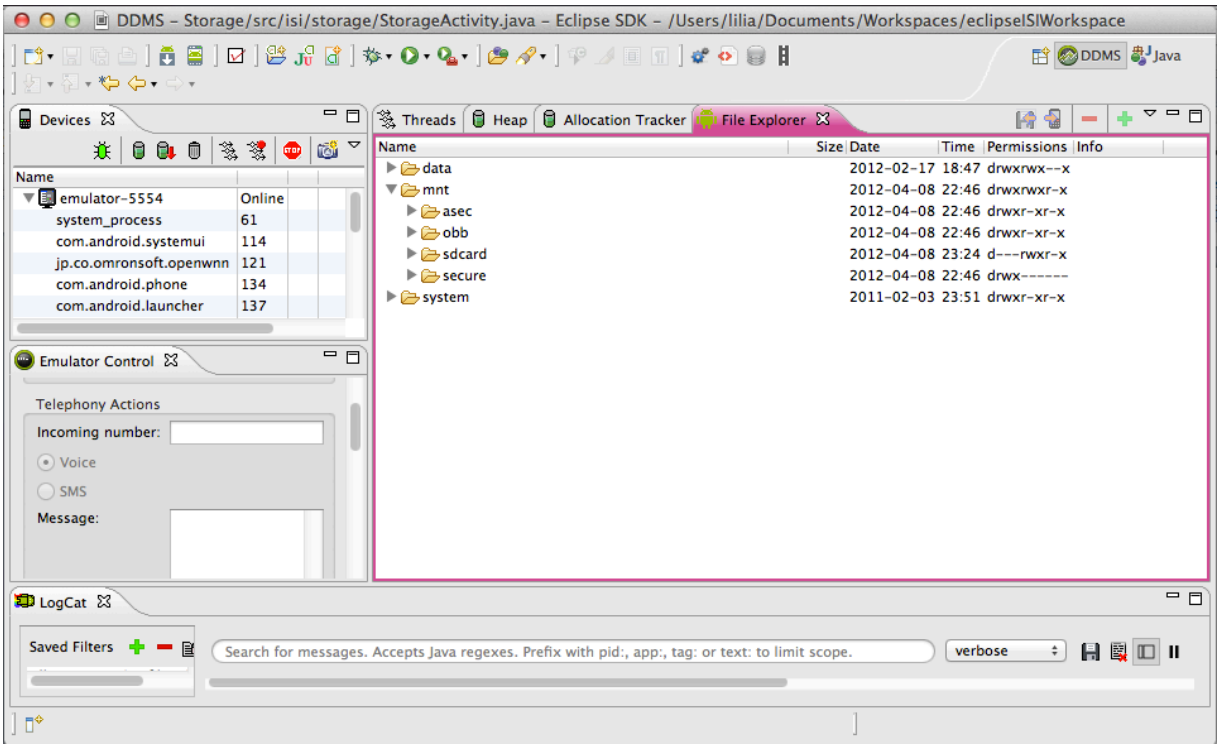
Android supporte l'accès à un système de stockage externe (une carte SD, par exemple). Tous les fichiers et répertoires de ce support de stockage sont accessibles en lecture pour toutes les applications.

II. 1. Accès et modification du support de stockage externe

Pour visualiser le contenu du support de stockage externe avec Eclipse, il est possible d'utiliser la perspective DDMS (*Dalvik Debug Monitor Server*). Pour cela :

- Aller à *Window -> Open Perspective -> Other…*
- Sélectionner *DDMS* et cliquer sur OK.

Une fois la perspective choisie, l'affichage devient comme dans la figure suivante :



Dans *FileExplorer*, vous pouvez voir le contenu de votre carte SD sous le répertoire *mnt/sdcard*.

- Pour ajouter un fichier à votre carte SD, sélectionner le répertoire *sdcard* et cliquer sur le bouton 
- Pour télécharger un fichier existant dans votre carte SD, cliquer sur le bouton 

Pour revenir à l'affichage du code, il faut revenir à la perspective *Java*. Pour cela, cliquer sur 

.

TAF-4 **:** Créer sur votre ordinateur un fichier qui s'appelle *chaine.txt*, dans lequel vous écrirez : « *Texte dans la carte SD* » Ensuite, ajouter ce fichier à la carte SD de votre émulateur.

II. 2. Tester le support de stockage externe

Pour tester si le support de stockage externe est bien monté et s'il est accessible en écriture, on utilise la variable suivante :

```
String etat = Environment.getExternalStorageState();
```

Cette variable *etat* est une chaîne de caractère qui contient l'état du support externe.

- Si *etat* est égale à `Environment.MEDIA_MOUNTED` alors le support externe existe, est monté et accessible en lecture et écriture.
- Si *etat* est égale à `Environment.MEDIA_MOUNTED_READ_ONLY",` alors le support externe existe, est monté mais est accessible en lecture seulement.

Ainsi, pour tester l'état du support externe, on peut exécuter le code suivant :

```
boolean stockageExiste = false;
boolean stockageEcriture= false;
String etat = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(etat)) {
    // Le support de stockage est accessible en lecture et écriture
    stockageExiste = stockageEcriture= true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(etat)) {
    // Le support de stockage est accessible en lecture seulement
    stockageExiste = true;
    stockageEcriture= false;
} else {
    // Le support de stockage n'est pas accessible
    stockageExiste = stockageEcriture= false;
}
```

TAF-5 : Dans l'application *Stockage*, créer une fonction qui s'appelle *mediaOK*, qui retourne *true* si la carte SD existe et es accessible en lecture et écriture, et *false* sinon.

II. 3. Lecture d'un fichier dans le support de stockage externe

Pour accéder à un fichier dans un support externe, on utilise le code suivant :

```
// accéder au répertoire par défaut du support externe
File directory = Environment.getExternalStorageDirectory();
// Accès au fichier
 File file = new File(directory + "/nom_du_fichier");
// Tester si le fichier désiré existe, sinon, un Toast affiche un message d'erreur
 if (!file.exists()) {
     Toast.makeText(this, "Fichier n'existe pas dans la carte SD", 1000).show();
 }else{        // Si le fichier existe
       try {
              // Créer un BufferedReader pour parcourir le fichier en lecture
              BufferedReader br = new BufferedReader(new FileReader(file));
              // Stocker le contenu du fichier dans un buffer
              String line;
              StringBuffer buffer = new StringBuffer();
              while ((line = br.readLine()) != null) {
                     buffer.append(line);
              }
              // Fermer le fichier
              br.close();
       } catch (Exception e) {
              Toast.makeText(this, e.getMessage(), Toast.LENGTH_LONG).show();
       }
}
```

TAF-6 : Ajouter un autre bouton dans l'application stockage appelé *Lire Externe*. Ce bouton permet de lire le contenu du fichier *chaine.txt* qui se trouve dans votre carte SD, et de l'afficher dans un Toast.

II. 4. Création ou modification d'un fichier dans un support externe

Pour modifier un fichier dans le support de stockage externe, il faut tout d'abord ajouter la ligne suivante dans le fichier AndroidManifest :

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Ensuite, utiliser les lignes suivantes dans le code de l'application, pour les mêmes variables *directory* et *file* que précédemment :

```
try {
        // ouvrir le fichier en écriture
        BufferedWriter bw = new BufferedWriter(new FileWriter(file));
        // écrire la chaine
        bw.write("chaine_a_ecrire");
        // fermer le fichier
        bw.close();
} catch (Exception e) {
        Toast.makeText(this, e.getMessage(), 2000).show();
}
```

Remarque : Si vous désirez ajouter le texte à la fin du fichier, il faut remplacer `new FileWriter(file)` par : `new FileWriter(file, true);` Le deuxième paramètre du constructeur détermine si la chaîne est concaténée (*true*) ou si le contenu est écrasé (*false*).

TAF-7 **:** Ajouter un autre bouton dans l'application stockage appelé *Ecrire Externe*. Ce bouton permet de stocker l contenu du champs de saisie dans le fichier *chaine.txt*.

III. Homework

Créer deux applications qui partagent un fichier *resultat.txt* qui se trouve dans la carte SD :

- La première application *SommeFichiers* contient un champs de saisie *txt* et deux boutons *ajouter* et *afficherListe*. Elle permet de :
  - Saisir un entier à partir du champs *txt*.
  - Le clic sur *ajouter* permet de :
    - Ajouter le contenu de *txt* à la fin d'un fichier interne appelé *entiers* (chaque nouvelle entrée sera séparée par une virgule de l'entrée précédente)
    - Lire le contenu du fichier *resultat.txt*
      - Si ce fichier est vide ou n'existe pas, le créer et y mettre le contenu de *txt*
      - Sinon, faire la somme de l'entier dans le fichier *resultat.txt* avec l'entier entré dans *txt*, puis stocker le résultat dans le fichier *resultat.txt* (l'ancienne valeur sera écrasée).
  - Le clic sur *afficherListe* permet d'afficher dans un Toast le contenu du fichier interne *entiers*.
- La deuxième application *VerifierFichiers* contient deux boutons :
  - Un bouton *afficherListe* qui affiche dans un Toast le contenu du fichier *entiers*
  - Un bouton *afficherResultat* qui affiche dans un Toast le contenu du fichier *resultat.txt*

Remarque :
- Pour convertir une chaîne en entier : int valeur = Integer.valueOf(chaine) ;
- Pour convertir un entier en chaîne : String chaine = String.valueOf(valeur) ;