Persistance des données
RMS

RecordStore Management System

Présentation

- Un RecordStore (entrepôt de données) constitue une collection d'enregistrements de tailles et de structures différentes.
- Chaque enregistrement est identifié par un id

- L'AMS est responsable de l'intégrité du recordstore

Gestion d'un RecordStore

- Le nom d'un recordstore (1 à 32 caractères unicode) doit être unique dans une même suite de midlets
- Ouverture et création d'un recordstore
  - RecordStore rs=RecordStore.openRecordStore(String nom, boolean Acreer); si le paramètre Acreer vaut true alors, le recordstore sera créé s'il n'existe pas.
- Suppression d'un RecordStore
  - deleteRecordStore(String nom)
- Méthodes d'un recordStore
  - String getName()
  - int getNumRecords()
  - int getSize(): retourne la taille du recordstore en octets.
- Remarque
  - Toutes les méthodes d'un recordStore déclenchent des exceptions

Gestion des enregistrement d'un RecordStore

- Ajout

  int addRecord(byte [] d, int offset, int nombre): ajoute un enregistrement et retourne son id.

- Lecture
  - int getRecord(int id, byte [] b, int offset): retourne le nombre d'octets copiés dans le tableau b à partir de la position offset.
  - byte [] getRecord(int id)
- Modification
  - void setRecord(int id, byte[] ndonnees, int offset, int nombre)
- Suppression
  - deleteRecord(int id)

Parcours d'un RecordStore

- La méthode enumarateRecords retourne la liste des énumération d'un recordStore

  public RecordEnumeration **enumerateRecords**(RecordFilter filtre, RecordComparator comparateur, boolean MiseAjour) throws RecordStoreNotOpenException

  - filtre: permet de filtrer les enregistrement
  - comparateur: définit un critère de tri des enregistrement
  - MiseAjour: pour tenir la liste retournée en synchronisation avec les éventuelles modifications du RecordStore.

- Exemple

```
RecordEnumeration liste=rs.enumerateRecords(null, null, false);
        byte[] rec;
        while (liste.hasNextElement()){
           rec=liste.nextRecord();
        }
```