Como armazenar dados usando RMS, em Java ME

Ao executarmos determinadas aplicações em nossos dispositivos móveis, muitas das vezes se faz necessário o armazenamento de dados para um possível reúso. Em Java ME isso pode ser feito atráves da API RMS (Record Management System).

O RMS é o gestor da base de dados e se constitui basicamente de um conjunto de *record stores* (ou armazéns de registros). *Record store*, por sua vez, nada mais é do que um conjunto de registros composto de dois campos. Um identificador(número) e um campo de dados(um array de bytes). Os *record stores* têm ligação direta com a MIDlet suite e não com os MIDlets, portanto, se desejar que cada MIDlet da MIDlet suite tenha o seu *record store*, faz-se necessário o uso de nomes diferentes, já que estes são *case sensitive*. Quando a MIDlet suite é removida, o mesmo acontece com os *record stores*.

A seguir temos um simples exemplo de código que nos mostra como utilizar a API RMS:

```
import javax.microedition.rms.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
public class ReadWriteRMS
        extends MIDlet implements CommandListener {
    private Display display;
    private Alert alert;
    private Form form;
    private Command exit;
    private Command start;
    private Command delete;


    private RecordStore recordstore,recordstore1,recordstore2,recordstore3,recordstore4,recordstore5 = null;
    RecordEnumeration recEnum;

    public ReadWriteRMS() {
        display = Display.getDisplay(this);
        exit = new Command("Exit", Command.EXIT, 1);
        start = new Command("Start", Command.SCREEN, 1);
        delete = new Command("Delete",Command.SCREEN,2);
        form = new Form("Mixed Record");
        form.addCommand(exit);
        form.addCommand(start);
        form.addCommand(delete);

        form.setCommandListener(this);
    }
    public void startApp() {
        display.setCurrent(form);
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
    public void commandAction(Command command, Displayable displayable) {
        if (command == exit) {
            destroyApp(true);
            notifyDestroyed();
        } else if (command == start) {
            try {
                //Abre um novo armazém de registros chamado "myRecordStore".
                //A variável true informa que este armazém deve ser criado se este ainda não existe
                recordstore = RecordStore.openRecordStore("myRecordStore", true );


            } catch (Exception error) {
                error.printStackTrace();
                alert = new Alert("Error Creating", error.toString(), null, AlertType.WARNING);
                alert.setTimeout(Alert.FOREVER);
                display.setCurrent(alert);
            }
            try {
                byte[] outputRecord;

                // Dados a serem adicionados no armazém de registros
                String outputString[] = {"Rajesh","Chandra","k","Raj","Chand","k"};
                int outputInteger[] = {25,21,20,23,34,34};
                boolean outputBoolean[] = {true,false,true,true,false,true};

                ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                DataOutputStream outputDataStream = new DataOutputStream(outputStream);

                for(int i=0; i<outputString.length;i++) {
                    outputDataStream.writeUTF(outputString[i]); //Armazena objeto do tipo string
                    outputDataStream.writeBoolean(outputBoolean[i]); //Armazena primitivo do tipo boolean
                    outputDataStream.writeInt(outputInteger[i]); //Armazena primitivo do tipo int
                    outputDataStream.flush(); //descarrega os dados
                    outputRecord = outputStream.toByteArray(); //cria vetor de bytes contendo os dados
                    recordstore.addRecord(outputRecord, 0, outputRecord.length); //adiciona os dados no armazém de registros
                }

                outputStream.reset();
                outputStream.close();
                outputDataStream.close();
            } catch ( Exception error) {
                alert = new Alert("Error Writing",
                        error.toString(), null, AlertType.WARNING);
                alert.setTimeout(Alert.FOREVER);
                display.setCurrent(alert);
            }

// Read records from RMS.........
// É importante mencionar que para cada tipo de dado escrito, a leitura deve ser realizada na mesma sequência da escrita
// Ou seja: lê String, boolean, int (nesta sequência)
            try {
/* String inputString = null;
int inputInteger = 0;
boolean inputBoolean = false; */

                //dados lidos serão armazenados neste buffer
```


(C) Copyright Nokia 2013. All rights reserved.

```
                StringBuffer buffer = new StringBuffer();

                byte[] byteInputData = new byte[100];
                ByteArrayInputStream inputStream = new ByteArrayInputStream(byteInputData);
                DataInputStream inputDataStream = new DataInputStream(inputStream);

                //obtém o número de registros neste (recordstore) armazém de registros
                int numrec = recordstore.getNumRecords();
                System.out.println(" Num rec "+numrec);
                for(int i=1;i<=numrec;i++) {
// recordstore.getRecord(i);
                    recordstore.getRecord(i,byteInputData,0); //obtém o registro de identificador "i"
                    buffer.append(inputDataStream.readUTF()); //lê objeto string
                    buffer.append("\n");
                    buffer.append(inputDataStream.readBoolean()); //lê primitivo boolean
                    buffer.append("\n");
                    buffer.append(inputDataStream.readInt()); //lê primitivo int
                    buffer.append("\n");
                    alert = new Alert("Reading", buffer.toString(),
                            null, AlertType.WARNING);
                    alert.setTimeout(Alert.FOREVER);
                    display.setCurrent(alert);
                }
/* recEnum = recordstore.enumerateRecords(null,null,false);
while (recEnum.hasNextElement())
{
recordstore.getRecord(recEnum.nextRecordId(),byteInputData,0);
buffer.append(inputDataStream.readUTF());
buffer.append("\n");
buffer.append(inputDataStream.readBoolean());
buffer.append("\n");
buffer.append(inputDataStream.readInt());
buffer.append("\n");
alert = new Alert("Reading", buffer.toString(),
null, AlertType.WARNING);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert);
}
 */
                inputStream.close();
                inputDataStream.close();
            } catch (Exception error) {
                alert = new Alert("Error Reading",
                        error.toString(), null, AlertType.WARNING);
                alert.setTimeout(Alert.FOREVER);
                display.setCurrent(alert);
            }
            try {
                recordstore.closeRecordStore();
            } catch (Exception error) {
                alert = new Alert("Error Closing",
                        error.toString(), null, AlertType.WARNING);
                alert.setTimeout(Alert.FOREVER);
                display.setCurrent(alert);
            }
        }

        else if( command == delete) {

            if (RecordStore.listRecordStores() != null) {
                try {
                    //Deleta o armazém de registros "myRecordStore"
                    RecordStore.deleteRecordStore("myRecordStore");
                } catch (Exception error) {
                    alert = new Alert("Error Removing",
                            error.toString(), null, AlertType.WARNING);
                    alert.setTimeout(Alert.FOREVER);
                    display.setCurrent(alert);
                }
            }
        }
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.