How to access the RecordStore from different MIDlet Suites for read and write operations

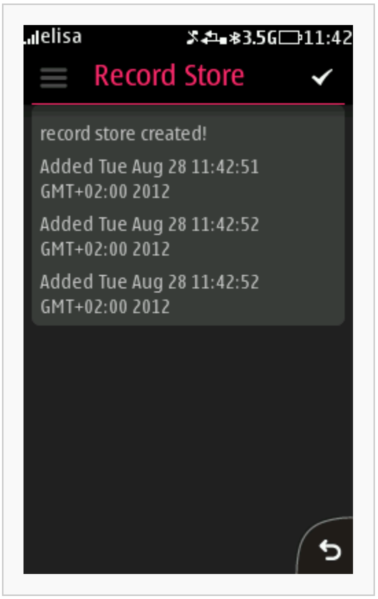
Overview

In the same way as in S60 devices, a 'Publish and Subscribe' type of mechanism can be simulated in Java ME using the Record Store API. The Record Store API is meant for creating a Record Store whose status can be made public or private for sharing the record store accross MIDlet suites or keep it private for one MIDlet suite, respectively.

When a record store is shared, a MIDlet can register for changes on the record store. Registration enables the MIDlet to get the following notifications about the changes in the store: record added, record modified, and record deleted.

The MIDlet which is the creator of a Record Store make it shareable or non-shareable. A MIDlet suite cannot decide on the mode (shareable/non-shareable) of a record store for which it does not own.

Description



elisa 3.5G 11:42

Record Store

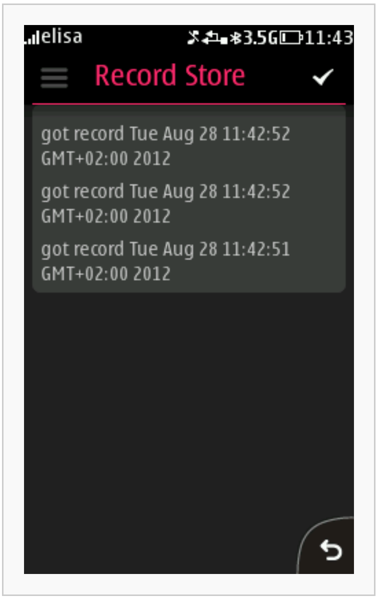
record store created!

Added Tue Aug 28 11:42:51 GMT+02:00 2012

Added Tue Aug 28 11:42:52 GMT+02:00 2012

Added Tue Aug 28 11:42:52 GMT+02:00 2012

The RMS Producer stores the current date in the Record Store



elisa 3.5G 11:43

Record Store

got record Tue Aug 28 11:42:52 GMT+02:00 2012

got record Tue Aug 28 11:42:52 GMT+02:00 2012

got record Tue Aug 28 11:42:51 GMT+02:00 2012

The RMS Consumer reads any records added by the Producer

In this example, a MIDlet named 'RMSProducerMIDlet' from a vendor called 'NokiaDeveloper' decides to create a Record store named 'MyStore' and make it shareable by setting its mode to `RecordStore.AUTHMODE_ANY`. The information stored in the RecordStore is the current date.

A MIDlet called 'RMSConsumerMIDlet' is interested in getting updates about the 'MyStore'. It can open the record store by specifying the Record Store name, the Vendor name, and the MIDlet suite name. After opening, it can register for any updates on that record store. If it does not need any updates at a certain point of time, it can unregister itself from the updates, too.

Solution

On the producer side, the code can be as follows:

```
//open a record store for updates.
RecordStore myRecordStore = RecordStore.openRecordStore("ShareableRecordStore", true);

//set the mode of the record store.
//RecordStore.AUTHMODE_ANY enables the record store to be available outside
//the suite for other MIDlet. The second parameter denotes that
//the record store is shared in read-only mode so that other
//MIDlets would not be corrupting it.
myRecordStore.setMode(RecordStore.AUTHMODE_ANY, false);

//after this the producer MIDlet can do any updates on the record store
//such as add record, update record, or delete a record using the record
```

```
 //store API.
 myRecordStore.addRecord(dataBytes,0,dataBytes.length);</b>
 myRecordStore.setRecord(recordId, myUpdatedData.getBytes(), 0,
myUpdatedData.getBytes().length);
```

On the consumer side, the code can be as follows:

```
 //On the consumer side there needs to be an implementation
 //of the RecordListener interface which demands to have 3
 //functions to be coded:
 //<b>1.public void recordAdded(RecordStore aRecordStore, int aRecId)</b>
 //<b>2.public void recordChanged(RecordStore aRecordStore, int aRecId)</b>
 //<b>3.public void recordDeleted(RecordStore aRecordStore, int aRecId)</b>
 //where aRecordStore is handle to the store and aRecId is the Id of the
 //affected record.

 //open the record store by providing the details
 RecordStore myRecordStore =
RecordStore.openRecordStore("ShareableRecordStore","ProducerMidletSuiteVendor","ProducerMidletSuite");

 //register for the updates
 myRecordStore.addRecordListener(this);

 //here "this" denotes the object that implements RecordListener
 //whenever the producer updates the record store the appropriate
 //function (recordAdded/recordChanged/recordDeleted) shall be called
 //when there is no need for updates, unregistration can be done
 myRecordStore.removeRecordListener(this);
```