Exporting phonebook contact in vCard format in Java ME

This code example demonstrates how to export contacts from phonebook to a file in vCard format.

Overview

To export a phonebook contact to a file, perform the following steps:

1. Get a list of available formats for exporting contact records by calling `PIM.supportedSerialFormats`. In this example user selects the format.
2. Open a file and create an output stream to it.
3. Get a contact from the list of contacts. In this example, the first contact from the list of contacts will be used.
4. Set output file stream, the contact that should be exported, encoding, and format name as parameters in `PIM.toSerialFormat` method. This method exports an event to a file using the specified format and encoding.
5. Close the file.

Source file: ExportContact.java

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.List;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Alert;

import javax.microedition.pim.PIM;
import javax.microedition.pim.ContactList;
import javax.microedition.pim.Contact;
import javax.microedition.pim.PIMException;

import javax.microedition.io.Connector;
import javax.microedition.io.file.FileConnection;

import java.io.IOException;
import java.io.OutputStream;

import java.util.Enumeration;


public class ExportContact extends MIDlet implements CommandListener  {

    private Display display;

    // List for displaying vCard formats.
    private List formatsList;

    // Command for exporting first contact in list contacts.
    private Command cmdExportContact;
    // Command for exiting from application.
    private Command cmdExit;

    private final String EXPORT_FILENAME = "exportedContact.vcf";

    /**
     * Constructor
     */
    public ExportContact() {
        if(checkPIMSupport() == false) {
            exitMIDlet();
        }

        initializeComponents();
    }

    /**
     * Initializes components of midlet.
     */
    private void initializeComponents() {
        // Get display
        display = Display.getDisplay(this);

        // Create list control
        formatsList = new List("Choose vCard version", List.IMPLICIT);

        cmdExportContact = new Command("Export first contact", Command.OK, 0);
        formatsList.addCommand(cmdExportContact);

        cmdExit = new Command("Exit", Command.EXIT, 0);
        formatsList.addCommand(cmdExit);

        formatsList.setCommandListener(this);

        showVCardVersions();
    }

    /**
     * Checks PIM support.
     * @return - true if PIM is supported, false otherwise.
     */
    private boolean checkPIMSupport() {
        String propValue = System.getProperty("microedition.pim.version");
        return (propValue != null);
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    /**
     * Gets supported vCard formats and adds it to formatList.
     */
    private void showVCardVersions() {
        // Get supported formats for exporting contacts.
        PIM pimInst = PIM.getInstance();
        String[] formats = pimInst.supportedSerialFormats(PIM.CONTACT_LIST);

        int formatsCount = formats.length;
        for(int index = 0; index < formatsCount; ++index) {
            String format = formats[index];
            if(format.startsWith("VCARD") == true) {
                formatsList.append(formats[index], null);
            }
        }
    }

    /**
     * Exports selected event to file in private folder.
     */
    private void exportContact(String exportFormat) {
        FileConnection file = null;
        OutputStream outStream = null;

        try {
            // Get first contact
            Contact firstContact = getFirstContact();
            if(firstContact == null) {
                throw new Exception("Contact list is empty.");
            }

            // Create filename string
            String path = System.getProperty("fileconn.dir.memorycard");
            if(path == null) {
                throw new Exception("Unable to get path to private directory.");
            }
            String fileName = path + EXPORT_FILENAME;
            // Open file for writing
            file = (FileConnection)Connector.open(fileName,
                    Connector.READ_WRITE);
            // If there is no file then create it
            if(file.exists() == false) {
                file.create();
            }

            // Export contact.
            outStream = file.openOutputStream();
            PIM.getInstance().toSerialFormat(firstContact, outStream, "UTF-8",
                    exportFormat);

            showAlert("Info", "First contact was exported in " + fileName);

        } catch(IOException ioExc) {
            showAlert("IO error", ioExc.getMessage());
        } catch(PIMException pimExc) {
            showAlert("PIM error", pimExc.getMessage());
        } catch(Exception exc) {
            showAlert("Error", exc.getMessage());
        } finally {
            // Try to close file.
            try {
                if(outStream != null) {
                    outStream.close();
                }
                if(file != null) {
                    file.close();
                }
            } catch(Exception exc) {
                // Do nothing.
            }
        }
    }

    /**
     * Returns first contact from phonebook
     * @return first contact from phonebook or null if no contacts were found
     * or error occurs.
     */
    private Contact getFirstContact() {
        ContactList contactList = null;
        Contact contact = null;

        try {
            PIM pimInst = PIM.getInstance();
            contactList = (ContactList)pimInst.openPIMList(
                    PIM.CONTACT_LIST, PIM.READ_WRITE);

            Enumeration contactEnum = contactList.items();
            if(contactEnum.hasMoreElements() == false) {
                contactList.close();
                return null;
            }

            contact = (Contact)contactEnum.nextElement();
        } catch(PIMException pimExc) {
            contact = null;
        } finally {
            // Try to close pim list
            if(contactList != null) {
                try {
                    contactList.close();
                } catch(Exception exc) {
                    // Do nothing.
                }
            }
        }

        return contact;
    }

    /**
     * Shows alert with specified title and text.
     * @param title - Title of alert.
     * @param message - text of alert.
     */
    private void showAlert(String title, String message) {
        Alert alert = new Alert(title);
        alert.setString(message);
        alert.setTimeout(Alert.FOREVER);
```


(C) Copyright Nokia 2013. All rights reserved.

```
        display.setCurrent(alert);
    }

    /**
     * From MIDlet.
     * Signals the MIDlet that it has entered the Active state.
     */
    public void startApp() {
        display.setCurrent(formatsList);
    }

    /**
     * From MIDlet.
     * Signals the MIDlet to enter the Paused state.
     */
    public void pauseApp() {
        // No implementation required.
    }

    /**
     * From MIDlet.
     * Signals the MIDlet to terminate and enter the Destroyed state.
     */
    public void destroyApp(boolean unconditional) {
        // No implementation required.
    }

    /**
     * Performs exit from midlet.
     */
    private void exitMIDlet() {
        notifyDestroyed();
    }

    /**
     * From CommandListener.
     * Indicates that a command event has occurred on Displayable displayable.
     * @param command - a Command object identifying the command.
     * @param displayable - the Displayable on which this event has occurred.
     */
    public void commandAction(Command command, Displayable displayable) {
        if(command == cmdExportContact) {
            int selectIndex = formatsList.getSelectedIndex();
            exportContact(formatsList.getString(selectIndex));
        }
        if(command == cmdExit) {
            exitMIDlet();
        }
    }
}
```

Postconditions

List of supported versions of the vCard format is shown on display.

The user can export the first contact from the phonebook to file by choosing a version of vCard format from the list and pressing "Export first contact". The file will be created in the root directory of the memory card and will have the name of EXPORT_FILENAME constant.

Supplementary material

Executables and source files can be found in Media:ExportingContact.zip.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.