Adding a new calendar event in Java ME

This code example demonstrates how to use the Java ME PIM API a add calendar event to the list of events

Overview

To add a new event, the user should fill in all form items and press the "Add new event" menu command. This will cause the following to happen in the MIDlet:

1. `PIMList.openPIMList` is called to open the list of events.
2. To create a new event, the `EventList.createEvent` method is called.
3. The event fields are filled with values from appropriate items on the form.
4. After calling the `Event.commit` method on the event object, this event will be saved in the list of events.
5. If the list of events is not needed any more, it can be closed by calling the `PIMList.close` method.
6. A reminder value can also be added in order to generate a sound at intervals before (and when) the event is triggered.

Source file: AddCalendarEvent.java

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.TextField;
import javax.microedition.lcdui.DateField;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Alert;

import javax.microedition.pim.PIM;
import javax.microedition.pim.PIMItem;
import javax.microedition.pim.EventList;
import javax.microedition.pim.Event;
import javax.microedition.pim.PIMException;

import java.util.Date;


public class AddCalendarEvent extends MIDlet implements CommandListener {

    private Display display;
    // Form where user can enter data of new event.
    private Form addEventForm;

    // Command for adding event to list of events. Placed on addEventForm.
    private Command cmdAddEvent;
    // Command for exiting from application. Placed on addEventForm.
    private Command cmdExit;

    // Text field for summary of event.
    private TextField summaryField;
    // Date field for start data of event.
    private DateField startDateField;
    // Date field for end data of event.
    private DateField endDateField;
    // Text field for note of event.
    private TextField noteField;
    // Text field for location of event.
    private TextField locationField;

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

        initializeComponents();
    }

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

        // Create form for adding event.
        addEventForm = new Form("Add event");

        // Add commands to form and set listener for it.
        cmdAddEvent = new Command("Add event", Command.SCREEN, 0);
        addEventForm.addCommand(cmdAddEvent);

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

        addEventForm.setCommandListener(this);

        try {
            // Get list of events.
            EventList eventList = (EventList)PIM.getInstance().openPIMList(
                    PIM.EVENT_LIST, PIM.READ_WRITE);

            // Create controls based on supported fields for event.
            if(eventList.isSupportedField(Event.SUMMARY) == true) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
            summaryField = new TextField("Summary", null, 20,
                                         TextField.ANY);
            addEventForm.append(summaryField);
        } else {
            // At least "summary" field must be supported.
            // If not, throw exception.
            eventList.close();
            throw new Exception("Summary field is not supported");
        }

        if(eventList.isSupportedField(Event.START) == true) {
            startDateField = new DateField("Start date",
                                           DateField.DATE_TIME);
            startDateField.setDate(new Date());
            addEventForm.append(startDateField);
        }

        if(eventList.isSupportedField(Event.END) == true) {
            endDateField = new DateField("End date", DateField.DATE_TIME);
            endDateField.setDate(new Date());
            addEventForm.append(endDateField);
        }

        if(eventList.isSupportedField(Event.NOTE) == true) {
            noteField = new TextField("Note", null, 20, TextField.ANY);
            addEventForm.append(noteField);
        }

        if(eventList.isSupportedField(Event.LOCATION) == true) {
            locationField = new TextField("Location", null, 20,
                                          TextField.ANY);
            addEventForm.append(locationField);
        }

        // Close list of events.
        eventList.close();

    } catch(PIMException pimExc) {
        // TODO: Handle error on working with PIM.
    }
    catch(SecurityException secExc) {
        // TODO: Handle error on access to PIM.
    }
    catch(Exception exc) {
        // If unknown error was catched, exit from application.
        exitMIDlet();
    }
}

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

/**
 * Adds event to list of events.
 * Gets data for event from addEventForm controls.
 */
private void addEvent() {
    try {
        // Get list of events.
        EventList eventList = (EventList)PIM.getInstance().openPIMList(
                PIM.EVENT_LIST, PIM.READ_WRITE);

        // Create new event.
        Event event = eventList.createEvent();

        // Get data from controls
        if(eventList.isSupportedField(Event.SUMMARY) == true) {
            String summary = summaryField.getString();
            event.addString(Event.SUMMARY, PIMItem.ATTR_NONE, summary);
        } else {
            // At least summary must be supported.
            eventList.close();
            throw new Exception("Summary field for event is not supported");
        }

        if(eventList.isSupportedField(Event.START) == true) {
            long startDate = startDateField.getDate().getTime();
            event.addDate(Event.START, PIMItem.ATTR_NONE, startDate);
        }

        if(eventList.isSupportedField(Event.END) == true) {
            long endDate = endDateField.getDate().getTime();
            event.addDate(Event.END, PIMItem.ATTR_NONE, endDate);
        }

        if(eventList.isSupportedField(Event.NOTE) == true) {
            String note = noteField.getString();
            event.addString(Event.NOTE, PIMItem.ATTR_NONE, note);
        }

        if(eventList.isSupportedField(Event.LOCATION) == true) {
            String location = locationField.getString();
            event.addString(Event.LOCATION, PIMItem.ATTR_NONE, location);
        }

        // Commit event.
        event.commit();
        // Close list of events.
        eventList.close();

        // Notify user that event was added
        showAlert("Info", "Event was successfully added.");

    } catch(PIMException pimExc) {
        // TODO: Handle error on working with PIM.
        showAlert("PIMException", pimExc.getMessage());
    }
    catch(SecurityException secExc) {
        // TODO: Handle error on access to PIM.
        showAlert("SecurityException", secExc.getMessage());
```


(C) Copyright Nokia 2013. All rights reserved.

```
        }
        catch(Exception exc) {
            // TODO: Handle all other errors.
            showAlert("Exception", exc.getMessage());
        }
    }

    /**
     * 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);
        display.setCurrent(alert);
    }

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

    /**
     * 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 == cmdAddEvent) {
            addEvent();
        }

        if(command == cmdExit) {
            exitMIDlet();
        }
    }
}
```

Postconditions

This code snippet represents a MIDlet with a form for adding values to supported fields of calendar events.

The menu command "Add new event" adds a new event to the list of calendar events and fills the fields with values from the form.

If the addition of the new event is successful, an appropriate notification is shown to the user. If any errors occur, the user will see a notification describing the error.

Defining a reminder value (optional)

Please note that by default, when defining a calendar event, its reminder value is set to none. The reminder value can be used to generate a sound at several predefined time intervals before the event is triggered. You can use the following code snippet to import a calendar event to the native Calendar application, with its reminder value set to 1 (at start time).

```
pim = PIM.getInstance();
try {
        EventList eventList = (EventList) pim.openPIMList(PIM.EVENT_LIST, PIM.READ_WRITE);
        Event event = eventList.createEvent();
        // get the current date
        Date date = new Date();
        // create a reminder starting 1 minute from now
        event.addDate(Event.START, Event.ATTR_NONE, date.getTime() + 60000);
        // and ending in 2 minutes from now
        event.addDate(Event.END, Event.ATTR_NONE, date.getTime() + 120000);
        /*
         * set the reminder value.  1   = at start time
         *                          300 = 5 minutes before
         *                          600 = 10 minutes before
         *                          900 = 15 minutes before
         */
        event.addInt(Event.ALARM, 0, 1);
        // import the event to calendar
```


(C) Copyright Nokia 2013. All rights reserved.

```
            Event imported = eventList.importEvent(event);
            imported.commit();
            eventList.close();
    }
     catch (PIMException ex) {
           System.out.println(ex.getMessage());
    }
```

Supplementary material

Executables and source files is available at Media:AddingCalendarEvent.zip.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.