Modifying an existing calendar event in Java ME

This code snippet demonstrates how to modify calendar events using Java ME.

Overview

To modify an existing event in the calendar events list:

1. Open the list of calendar events by calling the `PIM.openPIMList` method and get the event you want to modify from it.
2. Change the field values of the event by executing `Event.setXxx` methods where "Xxx" is either "Int", "String", or another data type. For a list of these methods, see Java ME documentation.
3. Call the `Event.commit` method to save changes of event data in the list of events.
4. Close the list of events if it is not needed anymore.

This MIDlet consist of 2 source files:

1. **ModifyCalendarEvents.java** contains the MIDlet class.
2. **ModifyEventForm.java** contains a form that allows the user to modify the field values of selected event.

Source file: ModifyCalendarEvents.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.PIMItem;
import javax.microedition.pim.EventList;
import javax.microedition.pim.Event;
import javax.microedition.pim.PIMException;

import java.util.Enumeration;
import java.util.Vector;

public class ModifyCalendarEvents extends MIDlet implements CommandListener {

    private Display display;

    // List where events will be placed.
    private List eventListCtrl;

    // Command for showing details of chosen event. Placed in eventListCtrl.
    private Command cmdModifyEvent;
    // Command for returning back from detailsForm to eventListCtrl.
    private Command cmdBack;
    // Command for exiting from application.
    private Command cmdExit;

    // Contains events retrivied from thePIM event list.
    private Vector eventsArray;

    // Form for modifying event.
    private ModifyEventForm modifyForm;

    // Event list.
    private EventList eventList;

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

        display = Display.getDisplay(this);

        openEventList();
        initializeComponents();
    }

    /**
     * Initialises components of midlet.
     */
    private void initializeComponents() {
        // Create list of events
        eventListCtrl = new List("Events", List.IMPLICIT);
        cmdModifyEvent = new Command("Modify", Command.ITEM, 0);
        eventListCtrl.addCommand(cmdModifyEvent);
        cmdExit = new Command("Exit", Command.EXIT, 0);
        eventListCtrl.addCommand(cmdExit);
        eventListCtrl.setCommandListener(this);

        addEventsToListCtrl();
    }

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


(C) Copyright Nokia 2013. All rights reserved.

```
            return true;
        } else {
            return false;
        }
    }

    /**
     * Opens list of event and stores link to it in eventList.
     * List of events remains opened to the exiting from application.
     */
    private void openEventList() {
        try {
            // Get list of events.
            eventList = (EventList)PIM.getInstance().openPIMList(
                    PIM.EVENT_LIST, PIM.READ_WRITE);
        } catch(PIMException pimExc) {
            // TODO: handle pim-specific error on accessing to PIM event list.
        } catch (Exception exc) {
            // TODO: handle other errors.
        }
    }

    /**
     * Closes list of events.
     */
    private void closeEventList() {
        try {
            eventList.close();
        } catch(PIMException pimExc) {
            // TODO: handle pim-specific error on accessing to PIM event list.
        } catch (Exception exc) {
            // TODO: handle other errors.
        }
    }

    /**
     * Adds events to list.
     */
    private void addEventsToListCtrl() {
        try {
            // Create array of events
            eventsArray = new Vector();

            eventListCtrl.deleteAll();

            int index = 0;
            Enumeration events = eventList.items();
            while(events.hasMoreElements() == true) {
                Event event = (Event)events.nextElement();
                // Add event to array of events
                eventsArray.addElement(event);
                // Add event's summary to list control
                String eventSummary = event.getString(Event.SUMMARY,
                        PIMItem.ATTR_NONE);
                eventListCtrl.append(eventSummary, null);
            }

        } catch(PIMException pimExc) {
            // TODO: handle PIM error.
        } catch(Exception exc) {
            // TODO: handle error.
        }
    }

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

    /**
     * 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) {
        closeEventList();
    }

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

    /**
     * Gets selected event from list and passes it to modifyEventForm.
     */
    private void modifyEvent() {
        int eventIndex = eventListCtrl.getSelectedIndex();
        Event event = (Event)eventsArray.elementAt(eventIndex);
        if(event != null) {
            modifyForm = new ModifyEventForm(event, eventList);
            modifyForm.setCommandListener(this);
            display.setCurrent(modifyForm);
        }
    }

    /**
     * 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) {
        // Handles "show details" command on selected event.
        if(command == cmdModifyEvent) {
            modifyEvent();
        }
        // Handles "exit" command.
```


(C) Copyright Nokia 2013. All rights reserved.

```
        if(command == cmdExit) {
            exitMIDlet();
        }
        // Handles "commit" command of modifyForm.
        if(command == modifyForm.getCommitCommand()) {
            modifyForm.commitEvent();
            addEventsToListCtrl();
            display.setCurrent(eventListCtrl);
        }
        // Handles "back" command from modifyForm.
        if(command == modifyForm.getBackCommand()) {
            display.setCurrent(eventListCtrl);
        }
    }
}
```

Source file: ModifyEventForm.java

```
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextField;
import javax.microedition.lcdui.DateField;
import javax.microedition.lcdui.Command;

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

import java.util.Date;

/**
 * Form gets event, provides user controls to modify its fields and
 * saves changes to event list.
 */
public class ModifyEventForm extends Form {

    // 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;

    // Event to modify.
    private Event event;
    // List of events.
    private EventList eventList;

    // Command for commiting chenges of event.
    private Command cmdCommit;
    // Command for returning back from form.
    private Command cmdBack;

    /**
     * Constructor.
     */
    public ModifyEventForm(Event event, EventList eventList) {
        super("Modify event");
        this.event = event;
        this.eventList = eventList;
        initializeComponent();
    }

    /**
     * Initializes components on the form.
     */
    private void initializeComponent() {
        try {
            // Create controls based on supported fields for event and fill it
            // with fields of event.
            if(eventList.isSupportedField(Event.SUMMARY) == true) {
                String summary = event.getString(Event.SUMMARY,
                        PIMItem.ATTR_NONE);
                summaryField = new TextField("Summary", summary, 30,
                        TextField.ANY);
                append(summaryField);
            }

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

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

            if(eventList.isSupportedField(Event.NOTE) == true) {
                String note = event.getString(Event.NOTE, PIMItem.ATTR_NONE);
                noteField = new TextField("Note", note, 30, TextField.ANY);
                append(noteField);
            }

            if(eventList.isSupportedField(Event.LOCATION) == true) {
                String location = event.getString(Event.LOCATION,
                        PIMItem.ATTR_NONE);
                locationField = new TextField("Location", location, 30,
                        TextField.ANY);
                append(locationField);
            }

        } catch(SecurityException secExc) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
            // TODO: Handle error on access to PIM.
        } catch(Exception exc) {
            // TODO: handle unknown error.
        }

        // Create commands.
        cmdCommit = new Command("Commit changes", Command.SCREEN, 0);
        addCommand(cmdCommit);

        cmdBack = new Command("Back", Command.BACK, 0);
        addCommand(cmdBack);
    }

    /**
     * Fills fields of event with values from controls and
     * saves changes to event list.
     */
    public void commitEvent() {
        try {
            // Get data from controls
            if(eventList.isSupportedField(Event.SUMMARY) == true) {
                String summary = summaryField.getString();
                event.setString(Event.SUMMARY, 0, PIMItem.ATTR_NONE, summary);
            }

            if(eventList.isSupportedField(Event.START) == true) {
                long startDate = startDateField.getDate().getTime();
                event.setDate(Event.START, 0, PIMItem.ATTR_NONE, startDate);
            }

            if(eventList.isSupportedField(Event.END) == true) {
                long endDate = endDateField.getDate().getTime();
                event.setDate(Event.END, 0, PIMItem.ATTR_NONE, endDate);
            }

            if(eventList.isSupportedField(Event.NOTE) == true) {
                String note = noteField.getString();
                event.setString(Event.NOTE, 0, PIMItem.ATTR_NONE, note);
            }

            if(eventList.isSupportedField(Event.LOCATION) == true) {
                String location = locationField.getString();
                event.setString(Event.LOCATION, 0, PIMItem.ATTR_NONE, location);
            }

            // Commit event
            event.commit();

        } catch(PIMException pimExc) {
            // TODO: Handle error on working with PIM.
        }
        catch(SecurityException secExc) {
            // TODO: Handle error on access to PIM.
        }
        catch(Exception exc) {
            // TODO: Handle all other errors.
        }
    }

    /**
     * @return "commit" command object.
     */
    public Command getCommitCommand() {
        return cmdCommit;
    }

    /**
     * @return "back" command object.
     */
    public Command getBackCommand() {
        return cmdBack;
    }
}
```

Postconditions

A list of calendar events is displayed.

By choosing an event from the list and pressing the "Modify", user can edit the values of fields for a selected event.

Supplementary material

Executables and source files are available at Media:ModifyingCalendarEvent.zip.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.