Localising timestamp in Java ME

This code example demonstrates how to implement the localisation of timestamps in a Java ME application.

Overview

One way to implement localised timestamps is to add date formatting patterns for each locale into the application and then format the date based on these patterns. To do this, add the following lines in the manifest of the application:

1. date-jp: yyyy-mm-dd
2. date-ru: dd.mm.yyyy
3. date-us: mm/dd/yyyy
4. title-jp: Japan date format
5. title-ru: Russian Federation date format
6. title-us: United States date format

In this example, the first three lines contain the date patterns used in Japan (jp), Russian Federation (ru), and United States (us). During runtime, the application gets these lines using the `MIDlet.getAppProperty()` method and converts the current date to a string using the `DateFormat` class, with patterns for each locale. The `DateFormat` class is a very simple string parser and date formatter, and it can be used as a basis for developing faster and more complex date formatters. So, by getting the current locale using the `System.getProperty()` method, using the date format patterns as resources, and implementing the date formatter, timestamps can be localised.

This MIDlet consists of three source files:

1. LocalizingTimestampMidlet.java - contains the MIDlet class.
2. DateFormat.java - contains the class for converting the date to a string in the specified format.
3. manifest.mf - manifest of the application with specified formats.

Source file: manifest.mf

```
MIDlet-1: LocalizingTimestampMidlet, , LocalizingTimestampMidlet
MIDlet-Vendor: Vendor
MIDlet-Name: LocalizingTimestamp
MIDlet-Version: 1.0
title-jp: Japan date format
title-ru: Russian Federation date format
title-us: United States date format
date-jp: yyyy-mm-dd
date-us: mm/dd/yyyy
date-ru: dd.mm.yyyy
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Source file: LocalizingTimestampMidlet.java

```
    // Some locales
    private static final String[] locales = {"us", "ru", "jp"};

    /**
     * Adds localized timestamps to the specified form
     * @param form - form where fields with localized timestamps will be placed
     * @throws java.lang.Exception if error while formatting timestramp occurs
     */
    private void showLocalizedTimestamps(Form form) throws Exception {
        // Get current locale
        String currentLocale = System.getProperty("microedition.locale");
        TextField locField = new TextField("CurrentLocale", currentLocale,
                128, TextField.UNEDITABLE);
        form.append(locField);

        // Format dates
        Date currentDate = new Date();

        for(int n = 0; n < locales.length; n++) {
            addFormattedDateField(form, locales[n], currentDate);
        }
    }

    /**
     * Creates text field containing formatted date and appends it to the form
     * @param form - the form where field with localized timestamp
     * will be placed
     * @param locale - the id determining pattern string using
     * for formatting date
     * @param date - source date to be formatted
     * @throws java.lang.Exception
     */
    private void addFormattedDateField(Form form, String locale, Date date)
            throws Exception {
        String title = getLocaleTitle(locale);
        String dateFormat = getLocaleDateFormat(locale);
        if(title == null || dateFormat == null) {
            throw new Exception("Unable to get applicaton settings.");
        }

        StringBuffer buffer = new StringBuffer();
        buffer.append(title);
```


(C) Copyright Nokia 2013. All rights reserved.

```
        buffer.append(" (");
        buffer.append(dateFormat);
        buffer.append(")");

        title = buffer.toString();

        String formattedDate = DateFormat.toString(dateFormat, date);
        if(formattedDate == null) {
            throw new Exception("Unable to convert date to specified format.");
        }

        TextField txtField = new TextField(title, formattedDate,
                128, TextField.UNEDITABLE);

        form.append(txtField);
    }

    /**
     * Gets locale title from application settings
     * @param locale - the id determining returning title for locale
     * @return title for specified locale
     */
    private String getLocaleTitle(String locale) {
        return getAppProperty("title-" + locale);
    }

    /**
     * Gets locale date format from application settings
     * @param locale - the id determining returning date format for locale
     * @return date format for specified locale
     */
    private String getLocaleDateFormat(String locale) {
        return getAppProperty("date-" + locale);
    }

    /**
     * Executes the snippet.
     * Deletes all fields from form and
     * outputs localized timestamps on this form.
     */
    private void executeSnippet() {
        mainForm.deleteAll();

        // Output localized timestamp to the form
        try {
            showLocalizedTimestamps(mainForm);
        } catch(Exception exc) {
            printString(exc.getMessage());
        }

        mainForm.append(logField);
    }
```

Source file: DateFormat.java

```
import java.util.Date;
import java.util.Calendar;

/**
 * Converts date to string in specified format
 */
public class DateFormat {

    /**
     * Converts date to string in specified format
     * @param format format for date converting (consist of 'yyyy', 'mm', 'dd'
     * parts with dividers between them)
     * @param date date to be converted
     * @return string containing converted date
     */
    public static String toString(String format, Date date) {
        StringBuffer buf = new StringBuffer(format.toUpperCase());
        Calendar cal = Calendar.getInstance();
        cal.setTime(date);

        // Change "YYYY" to year
        String year = String.valueOf(cal.get(Calendar.YEAR));
        replace(buf, "YYYY", year);

        // Change "DD" to day of month
        String day = String.valueOf(cal.get(Calendar.DAY_OF_MONTH));
        if(day.length() == 1) {
            day = "0" + day;
        }
        replace(buf, "DD", day);

        // Change "MM" to month
        String month = String.valueOf(cal.get(Calendar.MONTH) + 1);
        if(month.length() == 1) {
            month = "0" + month;
        }
        replace(buf, "MM", month);

        return buf.toString();
    }

    /**
     * Replaces first found substring in string buffer to dest string
     * @param buf - string buffer
     * @param subStr - substring for replacing
     * @param dest - string to insert to string buffer
     * @return position of first symbol of replaced substring in buffer or -1
     */
    private static int replace(StringBuffer buf, String src, String dest) {
        int pos = find(buf.toString(), src, 0);
        if(pos == -1) {
            return -1;
        }

        buf.delete(pos, pos + src.length());
        buf.insert(pos, dest);
```


(C) Copyright Nokia 2013. All rights reserved.

```
        return -1;
    }

    /**
     * Returns position of substring in a string
     * @param str - string to search in
     * @param subStr - substring to find
     * @param offset - first position to search
     * @return position of first occurence of substring in string or
     * -1 if nothing was found
     */
    private static int find(String str, String subStr, int offset) {
        if(offset < 0 || offset >= str.length() ||
                str.length() < subStr.length()) {
            return -1;
        }

        int maxPos = str.length() - subStr.length();
        for(int pos = offset; pos <= maxPos; pos++) {
            if(str.startsWith(subStr, pos) == true) {
                return pos;
            }
        }

        return -1;
    }
}
```

Postconditions

After running the command 'Execute snippet', text fields with various formattings of the current date and a text field with the current locale are displayed.

Supplementary material

This code snippet is part of the stub concept, which means that it has been patched on top of a template application in order to be more useful for developers. The version of the Java ME stub application used as a template in this snippet is v1.1.

- The patched, executable application that can be used to test the features described in this snippet is available for download at Media:LocalizingTimestamp.zip.
- You can view all the changes that are required to implement the above-mentioned features. The changes are provided in unified diff and colour-coded diff (HTML) formats in Media:LocalizingTimestamp.diff.zip.
- For general information on applying the patch, see Using Diffs.
- For unpatched stub applications, see Example app stubs with logging framework.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.