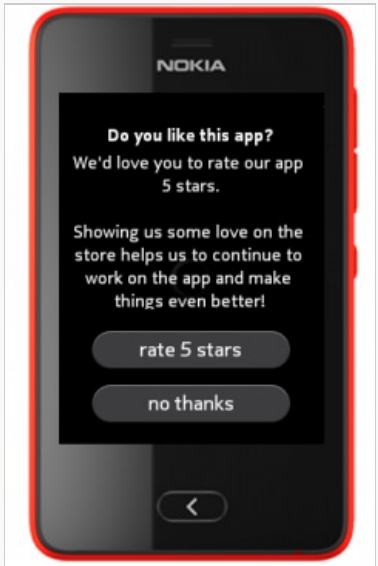
Porting the Rate My App component to the Asha software platform

This article illustrates the main steps performed to port the Rate My App component to the Asha software platform.

16 Feb 2014

Introduction

Rate My App is a component, originally developed for Windows Phone, prompting the user to rate an app and provide feedback to the developer.



Rate My App for Asha devices

This article illustrates the main steps performed in porting the Rate My App component to the Asha software platform.

Full source code and sample Java apps illustrating the component's features are available on GitHub: https://github.com/jappit/rate-my-asha-app

How does it work?

Rate My App displays a first prompt, after a set number of launches, asking the user to rate the app. If the user accepts, he/she is redirected to the app page on Nokia Store to review the app. If he/she declines, a second prompt asking the user to send a feedback about the app is displayed. If the user accepts, a preformatted email is opened within the device email client, and the user can add his/her own message before sending. If the user declines, the prompt is dismissed.

If the user doesn't rate the app on the first prompt, a second prompt is displayed after another set number of launches. Differently from the first rating prompt, in this case, if the user declines to rate the app, the feedback prompt is not displayed.

Porting to Java

The following sections show how the various features of Rate My App can be ported to Java, using the functionality offered by the Asha software platform.

Prompting the user

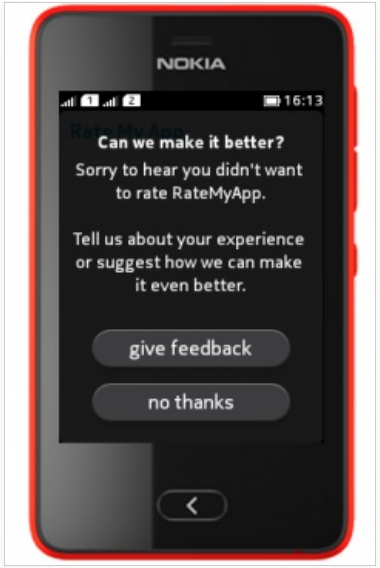
Rate My App prompts are composed by the following elements:

- a title
- a message
- a confirm button
- a cancel button

Following the design guidelines for Asha apps, the LCDUI Alert screen can be used in this case, since it provides a modal dialog that fits this scenario.

The picture below shows the prompt, localized in English, asking the user to send a feedback about the app to the developer.

(C) Copyright Nokia 2014. All rights reserved.



Feedback Alert dialog

Localization

The Windows Phone version of the Rate My App component uses XML files containing the localized resources of the various component's strings. In order to speed up the porting process, and allow developers to easily add support for new languages once they're available in the Windows Phone version, the same resource files are used.

Parsing the resource files

The parsing of the XML resource files can be performed using kXML.

Within the XML documents, the parser must search for <data> elements: each of those represents a localized strings. Once the parser finds one, it retrieves its `name` attribute to extract the localized resource key, and then gets the text content of its `<value>` child elements to extract the localized string. The following code shows this process in action.

```
KXmlParser parser = new KXmlParser();

parser.setInput(xmlInputStream);

strings = new Hashtable();

parser.nextTag();

parser.require(XmlPullParser.START_TAG, null, "root");

parser.nextTag();

while(parser.getEventType() != XmlPullParser.END_TAG)
{
 String nodeName = parser.getName();

 if(nodeName.compareTo("data") == 0)
 {
  String resourceName = parser.getAttributeValue("", "name");

  parser.nextTag();

  while(parser.getEventType() != XmlPullParser.END_TAG)
  {
   if(parser.getName().compareTo("value") == 0)
   {
    String resourceValue = parser.nextText();
```


(C) Copyright Nokia 2014. All rights reserved.

```
    strings.put(resourceName, resourceValue);
   }
   else
   {
    parser.skipSubTree();
   }
   parser.nextTag();
  }
 }
 else
 {
  parser.skipSubTree();
 }
 parser.nextTag();
}
```

Using the correct localization

The resource files are named with the following convention:

```
AppResources[.<LANGUAGE_CODE>[-<COUNTRY_CODE>]].resx
```

In order to display the correct localized strings, the component must access the locale currently set on the device: this can be performed by using the Locale class from the Nokia UI API.

Specifically, the following code shows how to retrieve the current locale:

```
Locale locale = LocaleManager.getInstance().getDefaultLocale();
```

In order to format the correct resource filename, the following code retrieves the locale language and country codes:

```
String languageCode = locale.getLanguage();

String countryCode = locale.getCountry();
```

Managing the component state

In order to properly work, the component must maintain history of the following data:

- the total number of app launches
- the date of the last app launch
- a flag that indicates if the user has already rated the app

This can be performed by using a RecordStore, where individual settings can be stored and retrieved among multiple app launches, and persist over app updates.

Component initialization

As seen from the previous sections, the component initialization must perform several steps, including:

- reading the component's state from `RMS`
- reading and parsing the localized resources

Since these are blocking operations, the initialization process is performed in an asynchronous process. In order to be notified when the initialization completes, an interface can be implemented as shown below.

```
public interface RateMyAppListener
```


(C) Copyright Nokia 2014. All rights reserved.

```
{
 public void rmaComponentReady();
}
```

Rating the app

If the user accepts to rate the app, he/she must be redirected to the Nokia Store page of the app itself, whose URL follows the syntax below:

```
http://store.ovi.mobi/content/<APP_ID>
```

On the current versions of the Asha software platform, the above URL correctly opens the Nokia Store native client, but fails to open the app's page within the client. As a workaround, the above URL is modified to redirect the user to the device Web browser, by using the following URL syntax:

```
http://www.store.ovi.mobi/content/<APP_ID>
```

Rating an Asha app on Nokia Store

Since the above URL opens the app's page, the user would still need to perform further actions to review the app. In order to remove these steps, that would prevent some users to actually rate the app, the URL is modified in order to let the user directly reach the review page, as follows:

```
http://www.store.ovi.mobi/content/<APP_ID>/comments/add
```

Sending the feedback email

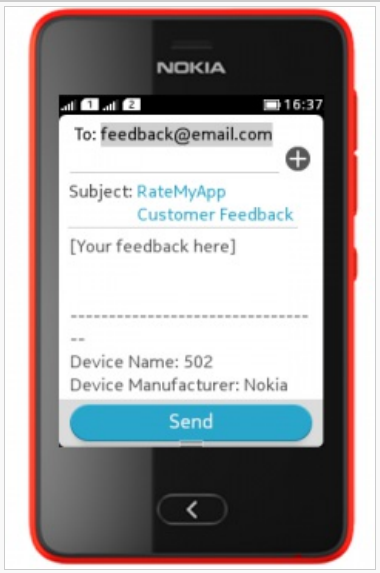
If the user accepts to send a feedback message to the developer, the native device client must be opened. In order to provide the maximum benefit to the developer, the email is preformatted with a subject and a message that already contain meaningful and useful information about the app and the device it's running on:

- the MIDlet version
- the device model
- the device manufacturer
- the firmware version
- the hardware version

The MIDlet version can be programmatically retrieved via the MIDlet getAppProperty, as shown below.

(C) Copyright Nokia 2014. All rights reserved.

```
String appVersion = midlet.getAppProperty("MIDlet-Version");
```



The preformatted feedback email message

The other four values can be extracted from the `microedition.platform` system property, whose syntax is explained in the dedicated page on the Java developer's library.

Handling the component

A Java app could need to know when the Rate My App prompts are displayed, in order to take appropriate actions (for instance, suspending timers or stopping animations).

For this reason, the `RateMyAppListener` interface is extended to include a method that gets called when the component's visibility changes.

```
public interface RateMyAppListener
{
 public void rmaVisibilityChanged(boolean visible);

 public void rmaComponentReady();
}
```

Usage example

A Java app can initialize the Rate My App component with the following line of code:

```
RateMyApp.init(midlet, appID, new MyRateMyAppListener());
```

Where `MyRateMyAppListener` is a custom class implementing the `RateMyAppListener` with the following code:

```
class MyRateMyAppListener implements RateMyAppListener
{
 public void rmaVisibilityChanged(boolean visible)
 {
  // manage the component's changed visibility
 }
 public void rmaComponentReady()
 {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    // the component has loaded all the necessary resources

    RateMyApp rma = RateMyApp.getInstance();

    rma.setFeedbackEmailAddress("feedback@email.com");

    rma.setStringResource(RateMyApp.STRING_RATING_TITLE, "Do you like this app?");

    rma.setFirstCount(1);
    rma.setSecondCount(2);
    rma.setCountDays(false);

    rma.launch();
  }
}
```

Specifically, once the component is initialized, the listener performs these operations:

- sets the email address for sending the feedback to
- overrides a single resource string
- sets the number of launches needed to display the first and second prompts
- launches the component

Summary

By using the Rate My App component on the Asha platform, developers can easily add the rate and feedback feature to Java apps, getting several benefits:

- incrementing the app's rating number on the Nokia Store
- getting direct feedback from app users about bugs and feature requests
- following the Asha guidelines for app design and user experience

Complete source code, together with example Java apps, is available on GitHub: https://github.com/jappit/rate-my-asha-app

(C) Copyright Nokia 2014. All rights reserved.