How to use the LCDUIUtil API

Overview

10 Oct 2010

LCDUIUtil is a utility class which provides two static methods, `getObjectTrait()` and `setObjectTrait()`, allowing the caller to respectively read and set value of "traits" associated with a given target object.

Each "trait" corresponds to an attribute or property, which is parsed and understood by the target object.

Those two static methods, a getter and a setter, currently allow you to access two different "traits" in Series 40 Touch and Type UI devices:

- Trait: `nokia.ui.s40.item.substate`, to control whether inline or sub-state screen editing style is used in editable form items (e.g. `DateField`, `TextField` and `Gauge`)
- Trait: `nokia.ui.s40.item.direct_touch`, to control whether a given `CustomItem` instance will receive touch events without having to be set as current (selected) first.

Methods

LCDUIUtil.getObjectTrait()

```
/**
 * @param Object target  - the target object to read the trait from
 * @param  String traitName  - the name of the trait to retrieve
 * @return Object   - valid Object if the request has succeeded or null otherwise
 *
 * @throw java.lang.IllegalArgumentException  - with String value "INVALID_ARGUMENT" if the target or traitName object is null
 * @throw java.lang.IllegalArgumentException  - with String value "ILLEGAL_ARGUMENT" if the target object is illegal for this trait
 * @throw java.lang.IllegalArgumentException  - with String value "NOT_SUPPORTED" if the specified trait is not supported by this pl
 * @throw java.lang.SecurityException   - if the application does not have the necessary privilege rights to access this content.
 */
public static java.lang.Object getObjectTrait(java.lang.Object target, java.lang.String traitName);
```

This method gets the trait of the specified target object as an object value. The returned object is a copy of the target's trait value and will not change if the corresponding trait changes.

Example:

```
Object traitValue = LCDUIUtil.getObjectTrait(myCustomItem,"nokia.ui.s40.item.direct_touch");
boolean directTouchUsed = ((Boolean) traitValue).booleanValue();
```

LCDUIUtil.*setObjectTrait()*

```
/**
 * @param Object target  - the target object to set the trait to
 * @param  String traitName  - the name of the trait to be set
 * @param  Object value   - the value of the trait to be set as an Object
 * @return boolean   - true if the request has succeeded or false otherwise (e.g. trait on this target is read only)
 *
 * @throw java.lang.IllegalArgumentException  - with String value "INVALID_ARGUMENT" if the target or traitName object is null
 * @throw java.lang.IllegalArgumentException  - with String value "ILLEGAL_ARGUMENT" if the target object or value object are illega
 * @throw java.lang.IllegalArgumentException  - with String value "NOT_SUPPORTED" if the specified trait is not supported by this pl
 * @throw java.lang.SecurityException   - if the application does not have the necessary privilege rights to access this content.
 */
public static boolean setObjectTrait(java.lang.Object target, java.lang.String traitName, java.lang.Object value);
```

This method sets the trait value of the target object as Object. Values in the given "value" Object are copied into the target's trait so subsequent changes to the "value" Object have no effect on the value of the target's trait.

Example:

```
LCDUIUtil.setObjectTrait(myTextField, "nokia.ui.s40.item.substate", Boolean.TRUE);
```

Traits

nokia.ui.s40.item.substate

Target(s): javax.microedition.lcdui.Item

On Series 40 touch devices, by default, when the user taps an editable Form Item, the MIDlet UI activates the item for in-line editing. However, on Series 40 6th Edition FP1 touch devices, the equivalent native UI components use a pop-up ("substate") window for editing the content. This trait allows application to control whether inline or sub-state screen editing style is used in editable Form items (e.g. DateField, TextField and Gauge) in Series 40 touch UI devices.

In practice, this means that the following Form Items have native UI counterparts on Series 40 6th Edition FP 1 devices that use a pop-up window for editing:

- DateField (of type DATE or TIME)

- Gauge (interactive)
- TextField

If you try to set the editing style for an editable Form Item that does not have a native UI counterpart, such as `DateField` of type `DATE_TIME`, the method call is ignored.

Example:

```
// Import the LCDUIUtil class:
import com.nokia.mid.ui.LCDUIUtil;
..

// Create the Form Item (in this case, a DateField of type DATE) whose editing style you want to align with the native UI,
// and set the editing style using the setObjectTrait method
DateField dateField = new DateField("Date", DateField.DATE);

// Set the editing style to pop-up
LCDUIUtil.setObjectTrait(dateField, "nokia.ui.s40.item.substate", Boolean.True);

// To set the editing style back to in-line, use the same method call, but set the third parameter to Boolean.False:
LCDUIUtil.setObjectTrait(dateField, "nokia.ui.s40.item.substate", Boolean.False);

...

// To query the editing style used by a Form Item, use the {{Icode|getObjectTrait}} method:

// Check whether or not a pop-up window is used for editing
Object traitValue = LCDUIUtil.getObjectTrait(dateField, "nokia.ui.s40.item.substate");

// Extract the boolean value from the returned Object:
// if true, a pop-up window is used
// if false, in-line editing is used
boolean popupUsed = ((Boolean)traitValue).booleanValue();
```

Warning: Error with the beta version
With the current **Series 40 6th Edition Feature Pack 1 SDK beta version**, the call of `LCDUIUtil.getObjectTrait()` with the `nokia.ui.s40.item.substate` trait on a normally compatible item (`DateField`, `Gauge`, `TextField`) will automatically raise a "**NOT_SUPPORTED**" exception. This is a **known issue** which will be fixed in the final release.

nokia.ui.s40.item.direct_touch

Target: javax.microedition.lcdui.CustomItem

On Series 40 devices, by default, a `CustomItem` must be focused before it can receive further touch events (pointer events or gesture events). In other words, the user must first tap the unfocused `CustomItem` or the MIDlet must first focus it by calling the `Display.setCurrentItem` method, before the content of the `CustomItem` can receive touch events. You can change this behavior with the `LCDUIUtil.setObjectTrait` method. If you want the first touch event on an unfocused `CustomItem` to both focus the component and register the touch event for the content, call `setObjectTrait()` as follows:

```
// Import the LCDUIUtil class:
import com.nokia.mid.ui.LCDUIUtil;
..

// Create the CustomItem
CustomItem customItem = new CustomItem("My customItem");

// Set the CustomItem to use single tap interaction so that it does not need to be focused first
LCDUIUtil.setObjectTrait(customItem, "nokia.ui.s40.item.direct_touch", Boolean.True);

// If you want to revert back to the default behavior, whereby the unfocused CustomItem must be focused first,
// call setObjectTrait as follows:
LCDUIUtil.setObjectTrait(customItem, "nokia.ui.s40.item.direct_touch", Boolean.False);
```

Known issue

As seen earlier, with the current Series 40 6th Edition Feature Pack 1 SDK beta version, a call of `LCDUIUtil.getObjectTrait()` with the `nokia.ui.s40.item.substat` trait on a normally compatible item (`DateField`, `Gauge`, `TextField`) will automatically raise a "NOT_SUPPORTED" exception. This is a known issue which will be fixed in the final release.

Device related KI: LCDUIUtil.getObjectTrait() with nokia.ui.s40.item.substate raises NOT SUPPORTED exception

Example application

This example uses a form on which 4 items are displayed, a `TextField`, an interactive `Gauge`, a `DateField` and a `CustomItem`. The three first option menus allow you to set the three first items in pop-up or inline editing mode and request the current editing mode. The next one allows you to reset the edited fields. The three last ones allow you to access the touch settings of the `CustomItem` that is to say to enable or disable the direct touch functionality and to request to the current touch mode as well. You will notice that if you click on the CustomItem, it will display a click information and its color will change. In direct touch mode the `CustomItem` reacts to every click you make while in "classic" touch you need to focus the item first. In both case, once the click has been recognized the item loses the focus, so that it is easy to notice the difference between the two modes.

Download the complete source code of the ready-to-use MIDlet described in this page: Media:LCDUIUtilMIDlet.zip