

Using Alerts in Java ME

This snippet shows how you can use the Alert object in [Java ME](#).

Overview

Alert is an object that can display some useful information on the screen, such as dialogues. You can choose between different types of Alerts. The Alert can be info, error, or confirmation, either getting the attention of the user for a few seconds, or requesting a confirmation from the user. An Alert object includes a title, message text, type, and optionally a [Gauge](#) or a custom image.

Source file: AlertsMIDlet.java

```

/**
 * This Alert object represents a cofirmation alert.
 */
private Alert confirmationAlert;
/**
 * This Alert object shows to us
 * a info alert.
 */
private Alert infoAlert;
/**
 * This Alert object uses for showing errors
 */
private Alert errorAlert;
/**
 * Command for accepting action during coffee suggestion
 */
private static final Command OK_ALERT_COMMAND =
    new Command("Yes", Command.OK, 0);
/**
 * Command for renunciation of coffee suggestion
 */
private static final Command CANCEL_ALERT_COMMAND =
    new Command("No", Command.CANCEL, 0);
/**
 * Sets up the different kinds of alerts.
 */
private void setupAlerts() {
    //Sets up the confirmation alert.
    //We set name of the picture, title, text, and type for alert
    //appropriately
    confirmationAlert = createSetupAlert("/confirmation.png",
        "confirmation alert",
        "Confirmation text", AlertType.CONFIRMATION);
    //Adds comands to our Alert and sets up command listener
    confirmationAlert.addCommand(OK_ALERT_COMMAND);
    confirmationAlert.addCommand(CANCEL_ALERT_COMMAND);
    confirmationAlert.setCommandListener(this);

    //Sets up the info alert.
    //It's includes setting up title, text, and type for alert
    //appropriately. This Alert has no picture.
    infoAlert = createSetupAlert(null, "Info caption",
        "Info message", AlertType.INFO);

    //Initialization of error Alert. We set up only type.
    //Others parameters will be created later
    errorAlert = createSetupAlert(null, null,
        null, AlertType.ERROR);
}
/**
 * Sets up the Alert objects
 * @param imageName - name of image file, which sets as Alert image
 * @param title - title for created Alert
 * @param message - text body message for Alert
 * @param alertType - type of Alert object (can be INFO, WARNING, ERROR,
 * ALARM, CONFIRMATION)
 * @return created Alert object
 */
private Alert createSetupAlert(String imageName, String title,
    String message, AlertType alertType) {
    Image image = null;

    //Creating Image if imageName presented
    if(imageName != null) {
        try {
            // load the image for the image from file
            image = Image.createImage(imageName);
        } catch (IOException ex) {
            printString(ex.toString());
        }
    }

    //Creating Alert object with title, text message, image and type
    Alert alert = new Alert(title, message, image, alertType);

    //If alert type is CONFIRMATION
    //then timeout for this is forever, and we must select command for
    //switching to our mainDisplayableForm
    if(alertType == AlertType.CONFIRMATION) {
        alert.setTimeout(Alert.FOREVER);
    } else {
        //...else Alert will stay at the top for 4 seconds
        alert.setTimeout(4000);
    }
    return alert;
}
/**
 * From CommandListener.
 * Called by the system to indicate that a command has been invoked on a

```

```

* particular displayable.
* @param command the command that was invoked
* @param displayable the displayable where the command was invoked
*/
public void commandAction(Command command, Displayable displayable) {
    if(displayable == mainDisplayableForm) {
        if (command == EXIT_COMMAND) {
            // Exit the MIDlet
            exit();
        } else if (command == CONFIRMATION_COMMAND) {
            printString("Starting show Alert object...");
            //show a coffee suggestion
            Display.getDisplay(this).setCurrent(confirmationAlert,
                mainDisplayableForm);
            printString("Alert object Ok!");
        } else if (command == INFO_COMMAND) {
            printString("Starting show Alert object...");
            //shows an info alert
            Display.getDisplay(this).setCurrent(infoAlert,
                mainDisplayableForm);
            printString("Alert object Ok!");
        } else if (command == ERROR_COMMAND) {
            errorAlert.setTitle("Error title!");
            errorAlert.setString("Error text!");
            printString("Starting show Alert object...");
            //shows an error alert
            Display.getDisplay(this).setCurrent(errorAlert, mainDisplayableForm);
            printString("Alert object Ok!");
        }
    } else {
        if (command == OK_ALERT_COMMAND) {
            //if we confirmed this message...
            printString("Ok!");
        } else if (command == CANCEL_ALERT_COMMAND) {
            //...otherwise printing that message
            printString("Cancel.");
        }
        Display.getDisplay(this).setCurrent(mainDisplayableForm);
    }
}
}

```

Postconditions

After starting this snippet, we can invoke three different commands for different types of alerts.

- 'Confirmation alert' starts a dialogue with 'Confirmation text' message. The user can choose 'Yes' or 'No'.
- 'Info alert' is an informational alert which displays an info message for a duration of 4 seconds.
- 'Error alert' shows the error message for a duration of 4 seconds.

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:UsingAlerts.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:UsingAlerts.diff.zip](#).
- For general information on applying the patch, see [Using Diff](#)s.
- For unpatched stub applications, see [Example app stubs with logging framework](#).

