Checking memory card in Java ME

This code snippet demonstrates how to check memory card availability.

Overview

The application uses the `System.getProperty` method to check the API version and memory card drive path.

After the path has been retrieved, the application establishes a file connection using the `Connector.open` method and gets the amount of space available on the memory card by calling `FileConnection.availableSize`.

Source file: CheckingMemoryCard.java

```
import javax.microedition.io.Connector;
import javax.microedition.io.file.FileConnection;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
/**
 * MIDP is an example checking memory card availability.
 */
public class CheckingMemoryCard extends MIDlet implements CommandListener {
    private Display display;
    /**
     * For displaying results.
     */
    private Form frmMain;
    /**
     * Calls exitMIDlet method.
     */
    private Command cmdExit;
    /**
     * Starts MC-checking procedure.
     */
    private Command cmdCheck;
    /**
     * Constructor. Constructs the object and initializes displayables.
     */
    public CheckingMemoryCard() {
        InitializeComponents();
    }
    /**
     * Initializes a ListBox object and adds softkeys.
     */
    protected void InitializeComponents() {
        display = Display.getDisplay( this );
        //initializing device list
        frmMain = new Form( "MC Info" );

        cmdExit = new Command( "Exit", Command.EXIT, 1 );
        cmdCheck = new Command( "Check", Command.SCREEN, 1);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        frmMain.addCommand( cmdExit );
        frmMain.addCommand( cmdCheck );
        frmMain.setCommandListener( this );
    }

    /**
     * From MIDlet.
     * Called when MIDlet is started.
     * @throws javax.microedition.midlet.MIDletStateChangeException
     */
    public void startApp() throws MIDletStateChangeException {
        display.setCurrent( frmMain );
    }
    /**
     * From MIDlet.
     * Called to signal the MIDlet to enter the Paused state.
     */
    public void pauseApp() {
        //No implementation required
    }
    /**
     * From MIDlet.
     * Called to signal the MIDlet to terminate.
     * @param unconditional whether the MIDlet has to be unconditionally
     * terminated
     * @throws javax.microedition.midlet.MIDletStateChangeException
     */
    public void destroyApp(boolean unconditional)
        throws MIDletStateChangeException {
        exitMIDlet();
    }
    /**
     * 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( command == cmdExit ) {
            exitMIDlet();
        }
        if( command == cmdCheck ) {
            //checking API version
            if( !checkAPIVersion() )
                return;
            //getting Localized UI name for the memory card
            checkMCDirName();
            //checking Root directory of a memory card
            if( !checkMCRoot() )
                return;
            //if Root directory retrieved in the previous step
            //then we do a file connection test
            checkMCConnection();
        }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
    /**
     * Method calls stopDiscover method and notifyDestroyed after that.
     */
    protected void exitMIDlet() {
        notifyDestroyed();
    }
    /**
     * Adds a StringItem to the frmMain.
     * @param strPrint string to add to frmMain.
     * @see Form#append(java.lang.String)
     * @see Form
     */
    protected void printToFrm( String strPrint ) {
        frmMain.append( strPrint );
    }
    /**
     * Gets JSR-75 API version using MIDP system property and
     * check whether it is null or not.
     * @return true if version detected and false otherwise.
     */
    protected boolean checkAPIVersion() {
        String fileconVer =
                System.getProperty(
                 "microedition.io.file.FileConnection.version" );
        printToFrm( "Getting JSR-75 API version..." );
        if( fileconVer != null ) {
            printToFrm( "Version : " + fileconVer );
        } else {
            printToFrm( "Version NOT found!");
            return false;
        }
        return true;
    }
    /**
     * Gets Localized MC directory name using MIDP system property and
     * check whether it is null or not.
     * @return true if name detected and false otherwise.
     */
    private boolean checkMCDirName() {
        String nameMC = System.getProperty( "fileconn.dir.memorycard.name" );
        printToFrm( "Getting MC directory name ..." );
        if( nameMC != null ) {
          printToFrm( "MC directory name : " + nameMC );
        } else {
            printToFrm( "MC directory name NOT found!");
            return false;
        }
        return true;
    }
    /**
     * Gets MC directory connection URL using MIDP system property and
     * check whether it is null or not.
     * @return true if URL detected and false otherwise.
     */
    private boolean checkMCRoot() {
        String rootMC = System.getProperty( "fileconn.dir.memorycard" );
```


(C) Copyright Nokia 2014. All rights reserved.

```
        printToFrm( "Getting MC root ..." );
        if( rootMC != null ) {
            printToFrm( "MC root : " + rootMC );
            } else {
                printToFrm( "MC root NOT found!");
                return false;
            }
        return true;
    }
    /**
     * Gets MC directory connection URL using MIDP system property and
     * check whether it is null or not.
     * @return true if URL detected and false otherwise.
     */
    private boolean checkMCConnection() {
        String rootMC = System.getProperty( "fileconn.dir.memorycard" );
        FileConnection fconn = null;
        try {
            fconn = (FileConnection)Connector.open( rootMC );
            printToFrm( "Getting available size ...");
            printToFrm( "Size: " + ( fconn.availableSize() / 1024 ) + "KB" );
            printToFrm( "Closing file connection...");
            fconn.close();
        } catch ( Exception e ) {
            printToFrm( e.toString() );
            return false;
        }
        return true;
    }
}
```

Postconditions

When the Check command is pressed, the following steps are performed:

- API availability check. If available, it is printed to the form.
- Getting Localized UI name for the memory card. If found, it is printed to the form.
- Checking Root directory of a memory card. If found, it is printed to the form.
- File connection test. The available size is printed to the form if the connection

was successful.

Supplementary material

The source file and executable application are available for download at Media:Checking memory card in Java ME.zip.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.