



Also published [at J2ME effort - Embedded](#)

Contents

- [1 FileConnection](#)
- [2 All you have to know before writing files in mobile devices](#)
- [3 What does our sample Code do?](#)
- [4 Used classes and information](#)
- [5 Code](#)

FileConnection

FileConnection (JSR75) allows access to removable storage devices, such as external memory cards that many of today's devices support. It was specially developed based on mobile devices limitation so it is simple and lightweight. FileConnection gives the functionality to create, remove directories and files, list directory content, set permissions, get file information, and perform I/O operations on files.

All you have to know before writing files in mobile devices

- Internal files (file systems on mobile devices or external memory cards) cannot be accessed and the reason is simple: Not all MIDP devices have file systems, and the creators may not want to expose them to applications.
- Developers don't have permission to write anything directly on root directories because of security reasons.
- S60 devices have a private directory where files can be written. If your MIDlet will be used with S40 devices, be careful because you don't have this directory. However, don't worry there are other directories where you can write files (for example: file:///c:/My files/Images/, file:///c:/My files/Tones/, etc).

What does our sample Code do?

Our MIDlet is a simple example of how to write a text file. Running our MIDlet you will see a textbox where you can write something and save it. The file called 'thiago.txt' will be saved inside photo directory in a folder named 'bruno'. You can run the MIDlet with your favorite IDE or with your mobile device.

Used classes and information

To read and write files we used FileConnection class that was specially created based on mobile devices limitations. Our device have to implement JSR 75 to run this MIDlet properly.

As used in J2SE we need to use streams for connection. To open a connection we use `Connection.open()` and to open a stream we can use `openInputStream()`, `openOutputStream()`, `openDataInputStream()`, or `openDataOutputStream()`.

Pay special attention to always check file's or directory's existence after a connection is established to determine if the file or directory actually exists. You can face exceptions or errors if you forget this. The same holds true using `delete()` method, and developers should close the connection immediately after deletion to prevent exceptions from accessing a connection to a non-existent file or directory.

Inside the method `Connection.open()` we can use `file:///c:/My files/Images/` or `System.getProperty("fileconn.dir.photos")`, but it is better to use `System.getProperty` because from device to device this path can change so you will face some problems. If you don't know the list of properties you can check it [here](#)

For further information about FileConnection check [here](#)

Code

The full code is below to facilitate your understanding.

```
import java.io.InputStream;
import java.io.OutputStream;
import javax.microedition.io.Connector;
import javax.microedition.io.file.FileConnection;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

public class MIDletfinal extends MIDlet implements CommandListener {

    Command save, exit

    TextBox text

    Display; display

public MIDletfinal() {

    = new TextBox(" ", "", 400, TextField.ANY);
    = new Command("Save", Command.SCREEN, 1);
    = new Command("Exit", Command.EXIT, 1);
    addCommand(save);
    addCommand(exit);
```

FileConnection_Example_-_JSR_75

```
        setCommandListener(this);
    }

    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
        // TODO Auto-generated method stub
    }

    protected void pauseApp() {
        // TODO Auto-generated method stub
    }

    protected void startApp() throws MIDletStateChangeException {
        = Displayable display(this);
        setCurrentDisplay();
    }

    public void commandAction(Command arg0, Displayable arg1) {

        if (arg0 == save) {
            ();
            saveFile
            // Alert used for notify the user that the file had already been
            // saved
            = new Alert("File saved.");
            setTimeout(Alert.FOREVER);
            setCurrent(alertdisplaytext);
        }

        if (arg0 == exit)
        ;
        {
            try {
                (true);
                destroyApp
                this.notifyDestroyed();
            } catch (MIDletStateChangeException e) {
                // TODO Auto-generated catch block
                printStackTrace();
                e.
            }
        }

    }

    private void saveFile() {

        try {
            // Creating a connection.
            = (FileConnection) Connector.open(System
            getProperty("fileconn.dir.photos")
            + "bruno/", Connector.READ_WRITE);
            // Checking if the directory exists or not. If it doesn't exist we
            // create it.
            if (c.exists()) {
                System.out.println("existe");
            } else {
                System.out.println("nao existe");
                mkdir();
                c.
                = (FileConnection) Connector.open(System
```

FileConnection_Example_-_JSR_75

```
    getProperty("fileconn.dir.photos")
+ "bruno/thiago.txt", Connector.READ_WRITE);
// create the file
    create();
// create an OutputStream
OutputStream out = c.openOutputStream();
// Get the user text
String userText = text.getString();
// write out the user's text into the file
    write(userText.getBytes()); out.
    flush(); out.
    close(); out.
// Never forget to close a connection or you can face problems.
// Pay attention here! If you close the connection before and
// later try to
// write something it will throw an exception.
    close(); c.

// Opening the file again and
// show what was saved in console.
    = (FileConnection) Connector.open(System
    getProperty("fileconn.dir.photos")
+ "bruno/thiago.txt", Connector.READ_WRITE);
//
InputStream in = c.openInputStream();
byte[] b = new byte[50];
    read(b); in.
System.out.println(new String(b, 0, b.length));

}
} catch (Exception e) {

}

}

}
```