Logging with Microlog in Java ME

This article explains how to use Microlog, a Log4j-based logging tool for Java ME

Introduction

Logging is important way of getting information about an application status at runtime. The simplest form of logging is really only effective when testing using an emulator (print variable values to the IDE's console by adding `System.out.println()` lines to the code) - for many features, like LBS, Bluetooth, sensors, etc, you really need to test on a device. In real devices the MIDlet output needs to be directed to `RecordStore`, a file, or to PC by (for example) using a Bluetooth connection.

To make this redirection process as simple and efficient as possible, separate tools are needed. Microlog is a great logging tool for MIDlets and it is based on well-known log4j API. The wiki already has one excellent article written about Microlog here.

This article tries to bring even more detailed step-by-step instructions for logging with Microlog using files and Bluetooth connection.

Note: This article has been written mainly by using Nokia Asha SDK 1.0 and Nokia Asha 501 device, but it should be useful for older devices and SDKs too.

Adding Microlog libraries to your project

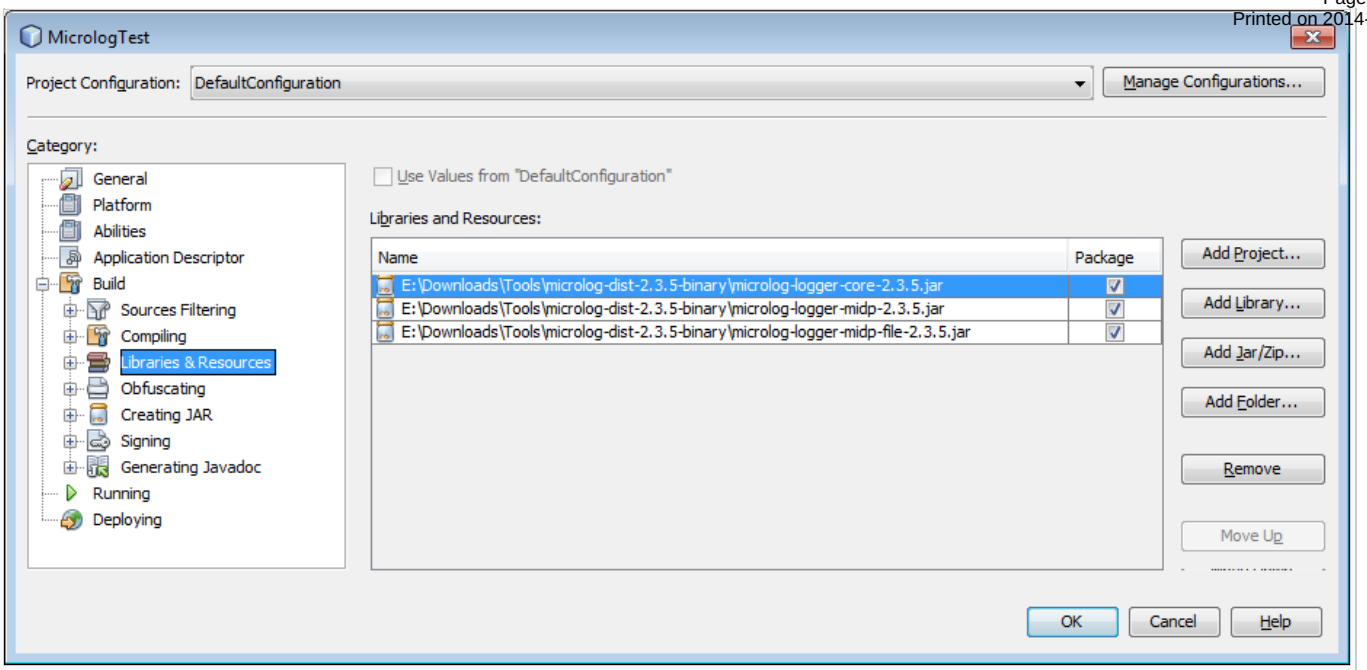
The latest Microlog logging library can be downloaded from here. When writing the article, the latest binary version was microlog-dist-2.3.5-binary.zip. The zip file contains the following library files:

- microlog-logger-core-2.3.5.jar
- microlog-logger-midp-2.3.5.jar
- microlog-logger-midp-bluetooth-2.3.5.jar
- microlog-logger-midp-file-2.3.5.jar
- microlog-logger-midp-wma-2.3.5.jar
- microlog-server-bluetooth-2.3.5.jar
- microlog-server-socket-2.3.5.jar

Some of the library files need to be included in your Java ME project, based on the `Appender` you use in your MIDlet. For example, if you use `FileAppender` to log data to a file, you need to include the following jar files in your project:

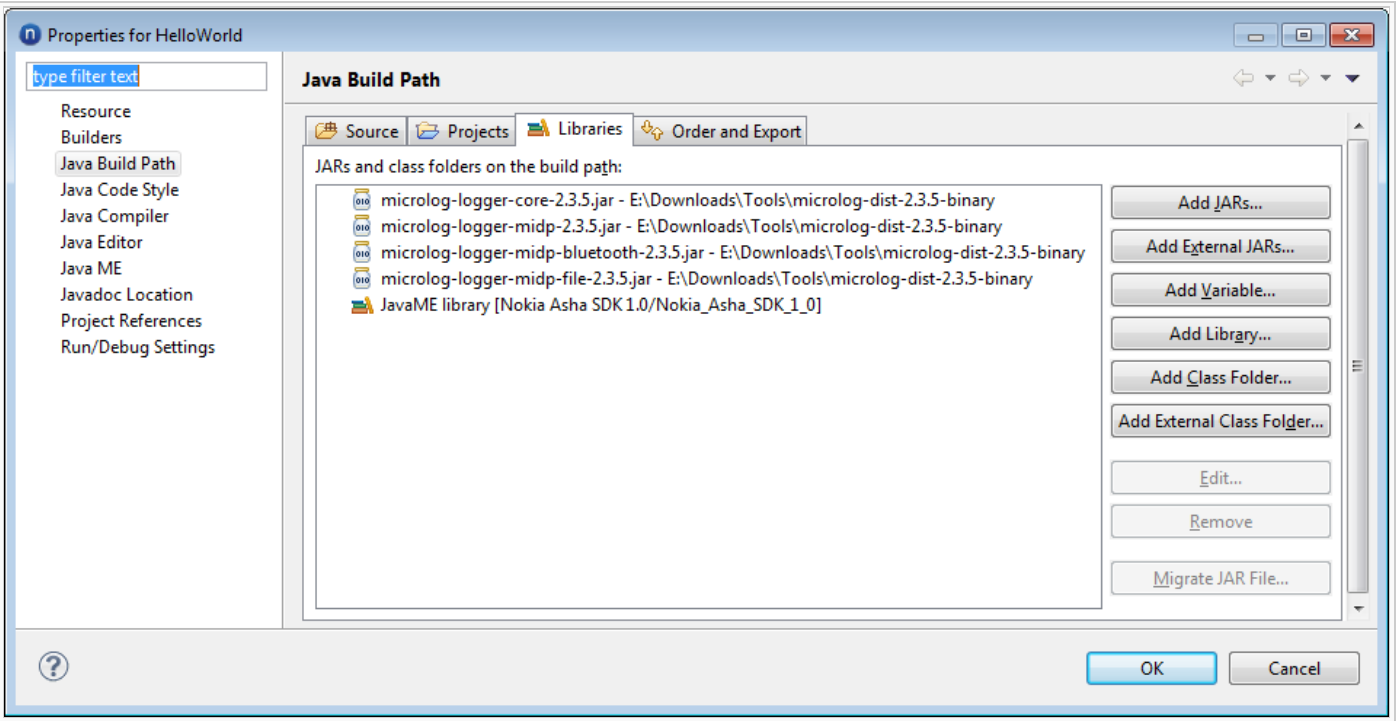
- microlog-logger-core-2.3.5.jar
- microlog-logger-midp-2.3.5.jar
- microlog-logger-midp-file-2.3.5.jar

In case of NetBeans IDE this can be done in **project properties -> Build -> Libraries & Resources** by pressing **Add Jar/Zip...** and adding the needed .jar files:



Microlog libraries added in NetBeans project

In case of Nokia IDE for Java ME this can be done in **project properties -> Java Build Path -> Libraries** by pressing **Add External JARs...** and adding the needed jar files:



Microlog libraries added in NetBeans project

Simple logging using System.out.println

For comparison only, below is a **HelloWorldMIDlet** which demonstrates simple logging using the `System.out.println()` method:

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

public class HelloWorldMIDlet extends MIDlet implements CommandListener {
 private Form form;
 private Command exitCommand;
 private Command logCommand;

 protected void startApp()  {
```

```
  System.out.println("Starting app...");
  form = new Form("Hello, world!");
  logCommand = new Command("Log", Command.SCREEN, 1);
  exitCommand = new Command("Exit", Command.EXIT, 1);
  form.setCommandListener(this);
  form.addCommand(logCommand);
  form.addCommand(exitCommand);
  Display.getDisplay(this).setCurrent(form);
 }

 protected void destroyApp(boolean arg0)  {
  System.out.println("Closing app...");
  LoggerFactory.shutdown();
 }

 protected void pauseApp() {
 }

 public void commandAction(Command c, Displayable d) {
  if (c == logCommand) {
   System.out.println("Logging...");
  }
  if (c == exitCommand) this.notifyDestroyed();
 }
}
```

Adding Microlog to your MIDlet

You can add Microlog to your MIDlet in two ways: by adding the needed code lines to your code or using properties file.

Adding Microlog in your code

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

import net.sf.microlog.core.Logger;
import net.sf.microlog.core.LoggerFactory;
import net.sf.microlog.core.format.PatternFormatter;
import net.sf.microlog.midp.file.FileAppender;

public class HelloWorldMIDlet extends MIDlet implements CommandListener {
 private Form form;
 private Command exitCommand;
 private Command logCommand;

    // A logger instance for this class
    private static final Logger log = LoggerFactory.getLogger(HelloWorldMIDlet.class);
    private FileAppender appender = new FileAppender();

 protected void startApp()  {
         appender.setFileName("micrologtestlog.txt");
         PatternFormatter formatter = new PatternFormatter();
         formatter.setPattern("[%P] %c %d (%r): %m %T");
         appender.setFormatter(formatter);
         log.addAppender(appender);
         log.info("FileAppender set!");

  log.info("Starting app...");
  form = new Form("Hello, world!");
```

```
  logCommand = new Command("Log", Command.SCREEN, 1);
  exitCommand = new Command("Exit", Command.EXIT, 1);
  form.setCommandListener(this);
  form.addCommand(logCommand);
  form.addCommand(exitCommand);
  Display.getDisplay(this).setCurrent(form);
 }

 protected void destroyApp(boolean arg0)  {
  log.info("Closing app...");
  LoggerFactory.shutdown();
 }

 protected void pauseApp() {}

 public void commandAction(Command c, Displayable d) {
  if (c == logCommand) {
   // logging into a file must be done in a separate Thread in Nokia Asha 501.
   new Thread() {
    public void run() {
     log.info("Logging...");
    }
   }.start();
  }
  if (c == exitCommand) this.notifyDestroyed();
 }
}
```

Adding Microlog using a configuration file

The alternative way is to use a separate configuration file and import it into your code using the `PropertyConfigurator` as shown below.

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

import net.sf.microlog.core.Logger;
import net.sf.microlog.core.LoggerFactory;
import net.sf.microlog.core.PropertyConfigurator;

public class HelloWorldMIDlet extends MIDlet implements CommandListener {
 private Form form;
 private Command exitCommand;
 private Command logCommand;

     // A logger instance for this class
     private static final Logger log = LoggerFactory.getLogger(HelloWorldMIDlet.class);

 protected void startApp()  {
         // Configure Microlog for using FileAppender
         PropertyConfigurator.configure("/microlog_file.properties");

         log.info("FileAppender set!");

  log.info("Starting app...");
  form = new Form("Hello, world!");
  logCommand = new Command("Log", Command.SCREEN, 1);
  exitCommand = new Command("Exit", Command.EXIT, 1);
  form.setCommandListener(this);
```

```
   form.addCommand(logCommand);
   form.addCommand(exitCommand);
   Display.getDisplay(this).setCurrent(form);
  }

  protected void destroyApp(boolean arg0)  {
   log.info("Closing app...");
   LoggerFactory.shutdown();
  }

  protected void pauseApp() {}

  public void commandAction(Command c, Displayable d) {
   if (c == logCommand) {
    // logging into a file must be done in a separate Thread in Nokia Asha 501.
    new Thread() {
     public void run() {
      log.info("Logging...");
     }
    }.start();
   }
   if (c == exitCommand) this.notifyDestroyed();
  }
 }
```

The configuration file, in this case named **microlog_file.properties** contains these lines:

```
# This is a simple Microlog configuration file
microlog.level=DEBUG
microlog.appender=FileAppender
microlog.appender.FileAppender.filename=MemoryCard/micrologtestlog.txt
microlog.formatter=net.sf.microlog.core.format.PatternFormatter
microlog.formatter.PatternFormatter.pattern=[%P] %c %d (%r): %m %T
```

The file extension is **.txt**, so that the file browser in Nokia Asha 501 recognises it to be text file. Now this can be opened by using the device's own file browser:

Logging file opened in Nokia Asha 501's file browser

Logging using Bluetooth

Microlog also allows you to use a Bluetooth connection and show log data directly on the PC (either on console window or in Microlog Bluetooth Server GUI screen). In both cases the needed btspp server URL is shown when the server is launched. This URL needs to be added to your code as URL, where the Bluetooth connection is opened.

Depending on the hardware configuration in your computer, you may need to download additional BlueCove library (BlueCove

is the Bluetooth library used in the MicroLog server library). For logging over Bluetooth the following files needs to be downloaded (note, that the version numbers might be different - check bluecove.org for the latest versions):

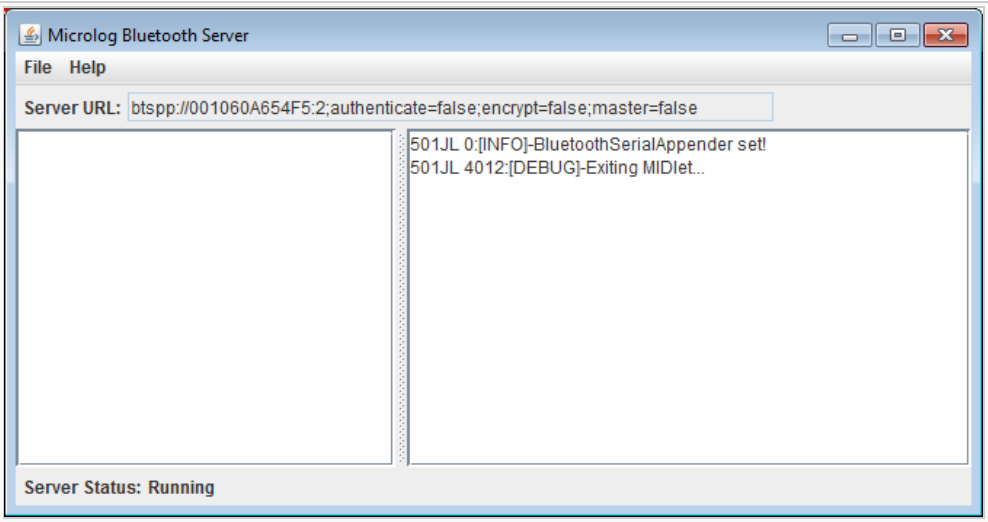
- bluecove-2.1.0.jar
- bluecove-gpl-2.1.0.jar

Now when the needed libraries have been downloaded, we need to make the long command with parameters for launching the Bluetooth server. This is done by entering "java -cp" with the jar file names and finally the main class name in the command prompt. For example, launching the server with GUI:

```
java -cp bluecove-2.1.1-SNAPSHOT.jar;bluecove-gpl-2.1.1-SNAPSHOT.jar;microlog-server-
bluetooth-2.3.5.jar net.sf.microlog.server.btspp.gui.MicrologBluetoothServerUI
```

For making this work you need to have Oracle JRE or JDK properly installed and configured. Also the library jar files must be in the same folder, where you enter the command. For making it easier in practice, it is a good idea to write this long command to a .bat file and use it for launching the server GUI.

When the command is run, it should launch the Microlog Bluetooth Server, as shown in the picture below. Note, that the needed Bluetooth server URL is shown on the window:



Microlog bluetooth server window showing the server URL

What makes this server GUI especially handy, is the possibility to save the log data to a file on your PC.

Logging over Bluetooth is very simple to implement in your code, just create an instance of `BluetoothSerialAppender` with the correct server URL as parameter:

```
import net.sf.microlog.midp.bluetooth.BluetoothSerialAppender;
...

BluetoothSerialAppender appender = new
BluetoothSerialAppender("btspp://001060A654F5:1;authenticate=false;encrypt=false;master=false");
```

Note, that for this you need to include the **microlog-logger-midp-bluetooth-2.3.5.jar** library file in your project, if `BluetoothSerialAppender` is used.

Summary: the needed steps

Here are the needed steps for starting to use Microlog:

1. Download the Microlog library from here
2. Add the needed library files to your project (based on logging method) and add the import lines to your code
3. Create `Logger` and `Appender` class instances
4. Add logging code to your source code (and create properties file if it is used)

Useful links

1. Microlog web site
2. Microlog - a Log4j-based tool for Java ME
3. Jarle Hansen's Blog: Using MicroLog over Bluetooth with Ubuntu and Eclipse
4. Article: Java ME Logging over Bluetooth using MicroLog
5. Article: Powerful Logging in Java ME
6. Article: Formatting Your Log in Microlog
7. BlueCove web site