Using a Java-method in a Package in a JAR-file from COBOL

Java programmers use packages to give their classes worldwide unique names. Additionally, these packages are usually provided in JAR files, which can contain several class files. This article describes how to use a Java-class, located in a package in a JAR file, based on Net Express 4.0, and Java SDK 1.4.2_08. The java and the COBOL source file, **20060401_001.zip**, are provided along with this article.

This article does **not** describe how to use a JAR file as an EJB in a J2EE Java Application Server. If you are interested in this, please refer to the online manuals.

Look at the programs:

In the java source file **demo.java** the package is defined as:

```
package com.microfocus.hco;
```

where a class named demo is defined:

```
public class demo …
```

This must correspond with the declaration in COBOL in the repository paragraph of **Cbltrig.cbl**:

```
repository.
   class demo as "$JAVA$com.microfocus.hco.demo".
```

The syntax above is ISO2002 COBOL – for the older MFOO syntax, please view the comments in the attached demo.

To uses Java classes and methods from COBOL, use the INVOKE statement from OO COBOL:

```
invoke demo "new" returning aJavaObj
invoke aJavaObj "displayHW" returning ret
```

Prepare a Net Express command prompt:

This article uses the command prompt, where **all** steps can be done. In a non-demo real application, the Java part and the COBOL part may be done with separate development environments for each language.

Please open a Net Express command prompt to follow the instructions.

Make sure, that the following environment variables are set correctly:

- **PATH** - must contain the directory of the Java compiler, javac.exe, where the jar.exe for building the jar-file should also be located and the directory of the Java Virtual Machine, jvm.dll, necessary for executing the java part.
- Entering: **Set PATH** in the command prompt should report a path like: **C:\j2sdk1.4.2_08\bin** for the Java compiler directory and a path like **C:\j2sdk1.4.2_08\jre\bin\client** for the JVM.

Using a Net Express command prompt the following should set for you:

- **PATH** to the <Net Express>\base\bin directory
- **CLASSPATH** to the file **mfcobol.jar**, which is located in the <Net Express>\base\bin directory

Build the jar-file for the demo:

To compile the java source file, type in the command prompt:

```
javac -d . demo.java
```

This instruction will place the compiled file into:

```
com\microfocus\hco\demo.class
```

Using a Java-method in a Package in a JAR-file from COBOL

To build the JAR file from the command prompt, type:

```
jar -cf mydemo.jar com\*
```

The JAR file can by checked with WinZip:

- start WinZip
- open the JAR file
- the file demo.class must have a path: com\microfocus\hco

Compile and link the COBOL part:

To compile and link the COBOL source file from a Net Express command prompt, type:

```
cbllink –RM cbltrig.cbl
```

It is necessary to link using the multi-threaded runtime system, which is done here by the M in –RM.

The –R means to link with the dynamic runtime system, which is located either via environment variable PATH or via registry settings from the installation of Net Express or the Application Server.

When you use the IDE to compile/link your program, pay attention to select the multi-threaded runtime system, i.e. right-click on the EXE, select “Build settings…”, choose tab “Link” and select “Multi-threaded”.

Run the demo:

To extend the environment variable **CLASSPATH** by the name of the JAR file, type in the Net Express command prompt:

```
set CLASSPATH=mydemo.jar;%CLASSPATH%
```

In a non-demo real application the jar-file may be specified with its full path.

Now the program is ready to run. Check, that the files **Cbltrig.exe** and **mydemo.jar** are in the local directory. Then type in the command prompt:

```
Cbltrig.exe
```

and you will receive the output, programmed in Java:

```
in: hcodemo
Hello world from JAVA
```

Please remember, that everything in Java is case sensitive, which is uncommon for COBOL programmers and Windows users.