Java: z/OS® Topics - Batch Integration, Tomcat and Eclipse

Steve Goetze
Kirk Wolf
Dovetailed Technologies, LLC.

http://www.dovetail.com

SHARE Session #8368 - March 8, 2006

Agenda

- Integrating Java with z/OS batch
- Accessing system services
- Apache Tomcat on z/OS
- SDK 5.0 shared classes
- Batch XML processing
- Using Eclipse to develop and deploy Java apps to z/OS

Java™ on z/OS®

- The IBM Java Developer Kits provide a Java VM and runtime environment for all z/OS Java-enabled subsystems, including WebSphere Application Server, CICS, DB2, and IMS.
- Currently five versions that run on z/OS:
  - IBM Developer Kit for OS/390, Java 2 Technology Edition, Version 1.3.1
  - IBM SDK for z/OS, Java 2 Technology Edition, Version 1.4
  - IBM 64-Bit SDK for z/OS, Java 2 Technology Edition, Version 1.4
  - IBM SDK 5.0 for z/OS, 31-bit and 64-bit versions
- The IBM JDKs also provide the standard *java* command line launcher for Unix System Services.
- The BPXBATCH utility can be used to run the *java* command line in a batch job

zAAP your mainframe Java costs

- Why is zAAP important? (a personal account)
- (Session 2825) z990 and z890 zAAP - What it Can Do for You

Running Java using BPXBATCH

```
//BPXBATCH JOB (999,XXX),'JAVA BPXBATCH',CLASS=A,MSGLEVEL=(1,1)
//   MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
```

```
//*****************************************************************
//* Run Java under a Unix System Service shell
//*****************************************************************
//STEP1 EXEC PGM=BPXBATCH,
// PARM='SH java com.foo.MyClass arg1 arg2'
//STDIN  DD DUMMY
//STDOUT DD PATH='/tmp/&SYSUID..bpxbatch.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/&SYSUID..bpxbatch.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
CLASSPATH=/u/myuid/classes:/u/myuid/lib/foo.jar:/u/myuid/lib/bar.jar
//
```

Running BPXBATCH (continued)

```
//*****************************************************************
//* Copy HFS output files to SYSOUT, since BPXBATCH can only write
//* STDOUT and STDERR to HFS files.
//*****************************************************************
//STEP2 EXEC PGM=IKJEFT01,DYNAMNBR=300,COND=EVEN
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/tmp/&SYSUID..bpxbatch.out'
//HFSERR DD PATH='/tmp/&SYSUID..bpxbatch.err'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
//
```

Problems with BPXBATCH

- Output cannot be routed directly to MVS datasets, another step is required to copy output from HFS files to SYSOUT
- The *java* command and JVM are run in a separate address space
  - Don't have access to DDs in the JCL from Java program
  - Accounting data doesn't attribute to the batch job address space
  - Separate Workload Manager (WLM) policy
- Output will be encoded using default **file.encoding** (more later)
- Can't pass condition code from Java program to subsequent steps
- Configuration of Java VM and environment variables is inflexible

BPXBATSL

A special entry-point into BPXBATCH, which can be used to run a USS command in the original batch

address space

```
//BPXBATSL JOB (999,XXX),'JAVA BPXBATSL',CLASS=A,MSGLEVEL=(1,1)
//   MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//*****************************************************************
//* Run Java under a Unix System Service shell
//*****************************************************************
//JAVA EXEC PGM=BPXBATSL,
// PARM='PGM /bin/sh -c java com.foo.MyClass arg1 arg2'
//STDIN  DD DUMMY
//STDOUT DD PATH='/tmp/&SYSUID..bpxbatch.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/&SYSUID..bpxbatch.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
... need to set ALL environment variables here, since no login shell
//
```

You say codepage, I say charset

- Internally, JVM Strings and chars are (always) Unicode
- The **file.encoding** system property:
  - Specifies the platform's file system character set encoding
  - Determines the *default* charset set when converting unicode characters to/from bytes
- It's common for Java code to (incorrectly) assume that file.encoding is ASCII
  - It's probably best when this happens just to run the JVM with file.encoding="ISO-8859-1" (ASCII/Latin)
  - But we (usually) don't want ASCII output!
- www.unicode.org/reports/tr17/ - "A character map may also be known as a charset, a character set, a code page"

z/OS Batch Java: WIBNIs

- System.out and System.err output routed directly to any dataset or JES spool file
- Control of output/platform encoding separate from default encoding
- Single address space
  - Simplifies job accounting and WLM profiles
  - Can use job step DD names
- Access wide variety of MVS datasets
- Pass condition code from Java `System.exit(--code--)`
- Flexible/scriptable configuration of environment variables and settings
- Communicate with MVS console (WTO, START, STOP, MODIFY)

BPXBATCH/SL New Features

PTFs for z/OS 1.7 were just released to address some of these problems.

- To obtain, install appropriate PTF for APAR OA11699
- Allows STDOUT/STDERR to be redirected to SYSOUT
- DD "STDPARM" can now be used to allow long "PARM=" input

Batch Integration - How?

- Implement a custom Java VM launcher as a program executable in a batch initiator or as a started task, using the Java Native Interface "launcher" API
- Have the launcher redirect output to DDnames, with platform encoding
- Allow scripting of environment variables and JVM settings
- Build JNI wrappers for "C" file I/O and system services
- JNI interfaces to WTO and console START, STOP and MODIFY commands
- Java `System.exit(--code--)` is used as step condition code

IBM JZOS Press Release

ARMONK, N.Y. –February 14, 2006 – IBM announced today its purchase of the JZOS batch technology developed by Dovetailed Technologies, LLC. The JZOS toolkit, which provides a simple, easy-to-use facility for running Java batch applications on the IBM mainframe flagship operating system, z/OS, is now available as a complimentary download on IBM alphaWorks.

... IBM intends to integrate JZOS technology into its SDK for z/OS, Java 2 Technology Edition products in 2006. The existing technology is available now at no charge at
http://www.ibm.com/alphaworks/tech/zosjavabatchtk

JZOS Batch Launcher and Toolkit

```
//JZOSBAT JOB (999,XXX),'JAVA JZOS',CLASS=A,MSGLEVEL=(1,1)
//   MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//JAVAJVM  EXEC PGM=JZOSVM14,
//   PARM='com.foo.MyClass arg1 arg2'
//STEPLIB  DD DSN=JZOS.LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD SYSOUT=*          < System stderr
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.err
//STDENV   DD *
. /etc/profile
. ~/.profile
export CLASSPATH=~/myapp
for i in ~/myapp/lib/*.jar; do
    export CLASSPATH=$i:$CLASSPATH
    done
```

File I/O

- Standard java.io package can only access HFS (Unix) files
- IBM Record and IO (JRIO) classes not complete, difficult...
- JZOS "ZFile" class provides thin JNI wrapper for "C" library I/O routines, which are well tested and documented

```
ZFile zFile = new ZFile("//DD:INPUT", "rb,type=record,noseek");
try {
    byte[] recBuf = new byte[zFile.getLrecl()];
    int nRead;
    while((nRead = zFile.read(recBuf)) > 0) {
       String line = new String(recBuf,0,nRead,
                                ZUtil.getDefaultPlatformEncoding());
       System.out.println(line);
    };
} finally {
   zFile.close();
}
```

Portable File I/O

"C" I/O routines allow MVS datasets to be processed in stream mode
Can we write portable code to process files (on PC) or datasets (on z/OS)?

```
import com.dovetail.jzos.FileFactory;
    ...
BufferedReader brdr = FileFactory.newBufferedReader(inFileName);
BufferedWriter bwtr = FileFactory.newBufferedWriter(outFileName);
try {
    String line;
    while ((line = brdr.readLine()) != null) {
        bwtr.write(line);
        bwtr.newLine();
    }
} finally {
    brdr.close();
    bwtr.close();
}
```

File names, at runtime could be: "c:\mypath.txt", "//MVS.DATASET.NAME", or "//DD:DDNAME".
The FileFactory will select either a ZFile or a java.io.File as appropriate

MVS Console interface

- WTO

```
ZUtil.wto("FOO1233E processing terminated",
      0x0020,   // routecde
      0x4000);  // descriptor code
```

- Operator commands
  - START - Java started task can access options
  - STOP(P) - Causes the JZOS batch launcher to issue a Java System.exit()
  - MODIFY(F) - Can send commands into your Java apps via a callback interface

Apache Tomcat

- A popular, open source, pure-Java Servlet / JSP container
- Fine for many web applications
- Can be bundled/embedded in a turnkey Java web application
- Can be configured for DB2 JDBC connectivity and SAF authentication and authorization
- Not a full J2EE container -- no EJBs :-)
- No Sysplex clustering, etc
- Limited J2EE connector support
- An "on-ramp" to WebSphere, since....

Mainframe Tomcat in an hour or less!

1. Download and install JZOS
2. Download and install Tomcat
3. Tailor and submit the JZOS Tomcat JCL (just as easily a started task)
4. Open for business!
5. Graceful shutdown via MVS STOP command (SDSF "Y")

- Tomcat runs best under the ISO-8859-1 file.encoding
  - Java Server Page (JSP) compilation works "out of the box"
- Step-by-step instructions at www.jzos.com/docs
- Try it yourself!... SHARE Session #8369: Java Lab: z/OS Tomcat Installation in an Hour

JZOS Launcher Tomcat JCL

```
//XXXXXXXX JOB ()
//PROCLIB JCLLIB ORDER=JZOS.SAMPJCL
//JAVA EXEC PROC=EXJZOSVM,VERSION='14',
// JAVACLS='org.apache.catalina.startup.Bootstrap',
// ARGS='start'
//STDENV DD *
# This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
. /etc/profile
export JZOS_HOME=/usr/local/jzos
```

```
export JAVA_HOME=/usr/lpp/java/J1.4
TOMCAT_HOME=/usr/local/tomcat

export PATH="$PATH":"${JAVA_HOME}"/bin:

LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic
LIBPATH="$LIBPATH":"${JZOS_HOME}"
export LIBPATH="$LIBPATH":
```

JZOS Launcher Tomcat JCL (cont'd)

```
CLASSPATH="${JAVA_HOME}/lib/tools.jar"
CLASSPATH="$CLASSPATH":"${TOMCAT_HOME}/bin/bootstrap.jar"
CLASSPATH="$CLASSPATH":"${JZOS_HOME}/jzos.jar"
export CLASSPATH="$CLASSPATH":

# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional arguments to main
#export JZOS_MAIN_ARGS=""
# Configure JVM options
# Note that Tomcat requires default ASCII file.encoding
IJO="-Xms64m -Xmx128m"
IJO="$IJO -Dfile.encoding=ISO8859-1"
IJO="$IJO -Djzos.home=${JZOS_HOME}"
IJO="$IJO -Dcatalina.base=${TOMCAT_HOME}"
IJO="$IJO -Dcatalina.home=${TOMCAT_HOME}"
IJO="$IJO -Djava.io.tmpdir=${TOMCAT_HOME}/temp"
IJO="$IJO -Djava.util.logging.config.file="
IJO="${IJO}${JZOS_HOME}/tomcat/logging.properties"
```

JZOS Launcher Tomcat JCL (cont'd)

```
IJO="$IJO -Djava.endorsed.dirs="
IJO="${IJO}${TOMCAT_HOME}/common/endorsed"
# Uncomment the following if you want to run without JIT
#IJO="$IJO -Djava.compiler=NONE"
export IBM_JAVA_OPTIONS="$IJO "
export JAVA_DUMP_HEAP=false
export JAVA_PROPAGATE=NO
export IBM_JAVA_ZOS_TDUMP=NO
//
```

Accessing DB2 under Tomcat

- First test/verify standalone Java SDK/JDBC DB2 Connectivity

- Define JNDI datasources for use under Tomcat
  - Define a DB2DataSource as a GlobalNamingResource
  - Individual webapps define ResourceLink(s)

```
<Resource type="com.ibm.db2.jcc.DB2DataSource" name="jdbc/mydb2"/>
    <ResourceParams name="jdbc/mydb2">
        <parameter><name>factory</name>
            <value>com.ibm.db2.jcc.DB2DataSourceFactory</value>
        </parameter>
        <parameter><name>database</name>
            <value>DSN1</value>     <!-- Your DB2 database name -->
        </parameter>
        <parameter><name>type</name>
            <value>2</value>
       </parameter>
    </ResourceParams>
<Resource>
```

System Authorization Facility (SAF)

- There are lots of Java security interfaces (JAAS, LDAP, JDBC, custom, etc)
- It's often desirable to leverage existing z/OS SAF interfaces.
- JZOS toolkit provides SAF interface routines:
  - Authentication - userid/password validation
  - Authorization - security roles mapped to SAF classes/entities (compatible with WebSphere usage)

Using SAF Security under Tomcat

- Program Control JZOS and supporting libraries (extattr +P, RALTER)
- Use JzosRealm for Tomcat authentication and authorization
- Map roles to SAF facilities, classes and entities:

```
<jzos-roles>
  <role rolename="admin"
            safclass="EJBROLE"
            safentity="MYAPP.PROD.ADMIN"
            saflevel="READ"/>
</jzos-roles>
```

- Use Tomcat's administration webapp to configure users and roles
- Use JZOS SAFTest utilities to troubleshoot

SDK 5.0 Shared Classes

- Facility with allows multiple JVMs (jobs) to share Java classes

- Cached in named shared memory, and dynamically updated
- Can reduce overall memory footprint
- Can reduce JVM startup time
- Caches can be private or shared across groups of users/jobs

For example, when using shared classes, Tomcat 5.5 can be started in *half* the time.

Eclipse

- A *great* open-source Java IDE.
  www.eclipse.org
- The basis for the WebSphere Studio product line (and many other IDEs)
- Code, compile, and test on workstation...
- ...one-click deploy your application to z/OS

Eclipse setup and coding

- Download, install, and configure Eclipse (see how-to at dovetail.com)
- Download the batch `XMLSample` Eclipse project from dovetail.com
- Customize zos.properties and move to the workspace directory
- Write code...
- Tailor JCL
- Run deploy.xml to deploy Java application to z/OS
- Submit JCL
- Can easily be extended to handle web applications, see `TomcatSample` project

Batch example: XMLSample

```
//JOBNAME JOB (),'ME',MSGCLASS=H,NOTIFY=&SYSUID
//PROCLIB JCLLIB ORDER=
//*
//PARSEXML EXEC PROC=EXJZOSVM,
// VERSION='50',
// LOGLVL='+I',
// JAVACLS='com.dovetail.jzos.sample.ParseXmlToFile',
// ARGS='//DD:XMLIN //DD:OUTFILE'
//XMLIN   DD DSN=MY.XML.DATA,DISP=SHR
//OUTFILE DD DSN=MY.RECORDS,DISP=(NEW,CATLG),
//           DCB=(RECFM=VB,LRECL=200,BLKSIZE=0),
//           SPACE=(CYL,(1,1),RLSE)
//STDENV  DD *
...
```

In Summary

- Effectively run Java in batch jobs

- Simplify WLM profiles, accounting/zAAP usage in a single batch address space
- Access z/OS datasets and services through Java JNI wrappers
- Install and run Tomcat on the mainframe in an hour or less
- Shared classes and SDK 5.0 performance improvements make short Java job step practical
- Use Eclipse to develop and one-click deploy applications to z/OS

References

- JZOS batch launcher and toolkit - http://www.ibm.com/alphaworks/tech/zosjavabatchtk
- Dovetailed Technologies - http://dovetail.com
- IBM developerWorks Article: Java Batch Jobs on z/OS and OS/390 http://www.ibm.com/developerworks/eserver/library/es-java-batchz.html
- IBM's home page for Java on z/OS - http://www.ibm.com/servers/eserver/zseries/software/java
- The zSeries Application Assist Processor (zAAP) - http://www.ibm.com/servers/eserver/zseries/zaap
- IBM's WebSphere Application Server for z/OS home page - http://www.ibm.com/software/webservers/appserv/zos_os390

References - continued

- z/OS Unix System Services home page - http://www.ibm.com/servers/eserver/zseries/zos/unix/
- z/OS Unix System Services users guide: Using BPXBATCH - http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/bpxza450/11.2
- The Java Record IO package, which is part of the IBM SDK - http://www.ibm.com/servers/eserver/zseries/software/java/jrio/overview.html
- The z/OS C/C++ Programming Guide - http://publibz.boulder.ibm.com/epubs/pdf/cbcpg130.pdf
- Sun's documentation on the Java JNI Launcher interface - http://java.sun.com/j2se/1.4.2/docs/guide/jni/
- IBM's article on Writing a simple JNI program on OS/390 - http://www.ibm.com/servers/eserver/zseries/software/java/os390jni.html