Conversion Tools - Support #7257

Should deploy.prepare delete all contents of deploy/lib/ before copying?

04/06/2023 06:45 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 04/06/2023 06:49 AM - Greg Shah

- Subject changed from Should deploy.prepare delete all contents of lib/ before copying? to Should deploy.prepare delete all contents of deploy/lib/ before copying?

When we change jar files to move to new versions, we get into a scenario where a deploy.prepare leaves multiple versions of the same project jars. Depending on the ordering of these jars in the classpath (usually the lowest version number first), this gives bad results. This is happening now due to recent H2 jar changes.

It seems like we can solve this by running a clean/delete all of the deploy/lib/ contents before copying in deploy.prepare. Is there any reason we should not do that?

#3 - 04/06/2023 06:54 AM - Constantin Asofiei

Cleaning up the entire deploy/lib at deploy.prepare will remove any customer-specific jars (like for web apps running from within the FWD server).

p2j.jar already has the correct jars in the classpath set at its manifest (assuming this was a clean build).

Usually the server.sh script for a customer application has something like:

```
cpath=""
for f in ../lib/*.jar
do
   cpath=$cpath:$f
done
```

which gets everything from deploy/lib and sets it as the classpath for the FWD server JVM.

We may want to clean deploy/lib on ant deploy.prepare, but with some exclude clauses for customer-specific jars.

The alternative would be to change server.sh so that it explicitly adds:

- the FWD jars (../lib/p2j.jar)
- application jars, like (../lib/hotel.jar)
- any other customer-specific jars

#4 - 04/06/2023 07:04 AM - Greg Shah

I think we need to handle customer libs more explicitly.

- Isolate them into their own static directory. OR
- Add a copy step to place them in deploy/lib/ during deploy.prepare.

I prefer the first option. Doing that would allow us to cut down the classpath because all the extra gorp would be in the manifest file for FWD. The current classpath is unmanageable.

#5 - 04/06/2023 07:05 AM - Constantin Asofiei

Greg Shah wrote:

> I prefer the first option.

Yes, that is the cleaner way.

#6 - 04/06/2023 10:22 AM - Roger Borrello

- File getspi.sh.ori added

- File getcp.sh.ori added

I have been using some varieties of the below description in some applications' server.sh and/or client.sh such that the FWD jars could be kept separate from the application jars:

```
prog="java"
fwdlib="libp2j.so"
appjar="hotel"

# getcp.sh and getspi.sh are helpers for setting up the classpath and spi for the given application.
# This allows more flexibility in where the FWD configuration is placed. It can be one directory up,
# and then in lib, or in /opt/fwd/lib, or specified by FWD_LIB exported or up to the build directory,
# then down to lib. If the helper isn't found, we can only fall back to the original method.
export PATH=..:$PATH
if [ ".$(which getcp.sh)." == ".." ]; then
   cpath=""
   for f in ../lib/*.jar
   do
      cpath=$cpath:$f
   done
   spi="-Djava.locale.providers=SPI,JRE -Djava.ext.dirs=$(${prog} ${cpath} com.goldencode.util.PrintSystemProp
 java.ext.dirs):../lib/spi"
else
   cpath=$(getcp.sh "$appjar" $cfgdir )
   spi=$(getspi.sh $cpath)
fi
```

I attached the helpers, which were written because there wasn't a distinct direction for how the to find the jar files, so it utilizes a multitude of methods:

```
# Setup classpath. First get the FWD jars and libraries, then the application's.
# For FWD, use libp2j.so as the determining factor. Rules:
```

```
# 1) Look for FWD_LIB
# 2) One directories above the cfgdir and in the lib directory from there (../lib)
# 3) Two directories above our current directory, and in the p2j/lib directory from there (../../p2j/lib)
# 4) Two directories above our current directory, and in the p2j/build/lib directory from there (../../p2j/bui
ld/lib)
# 5) A well-known location
```

It would be cleaner to standardize one (like the FWD_LIB environment variable) with a single fallback.

This has to be matched by the build.xml's deploy.prepare task. I have done this by breaking out the task into deploy.prepareapp and deploy.prepare:

```
   <!-- deploy the application jars and database -->
   <target name="deploy.prepare" depends="deployapp.prepare"
           description="Deploys *only* the generated jars by copying them to the location expected by the runt
ime scripts." >

      <!-- create and populate the extension lib folder -->
      <mkdir dir="${deploy.home}/lib/spi"/>
      <copy file="${fwd.lib.home}/fwdspi.jar" todir="${deploy.home}/lib/spi"/>

      <!-- Copy external jars -->
      <copy todir="${deploy.home}/lib">
        <!--  exclude the AspectJ tools -->
        <fileset dir="${fwd.lib.home}" excludes="aspectjtools.jar, fwdspi.jar"/>
      </copy>
   </target>

   <target name="deployapp.prepare"
           description="Deploys the FWD *and* generated jars by copying them to the location expected by the r
untime scripts." >

      <!-- create the deploy lib folder -->
      <mkdir dir="${deploy.home}/lib"/>

      <!-- copy the required jars to the distribution directory -->
      <copy file="${build.home}/lib/${appname}.jar" todir="${deploy.home}/lib"/>
   </target>
```

This allows you to keep them app jar(s) separate. Then the scripts to start the server/client match up.

Files

getcp.sh.ori	3.2 KB	04/06/2023	Roger Borrello
getspi.sh.ori	1.54 KB	04/06/2023	Roger Borrello