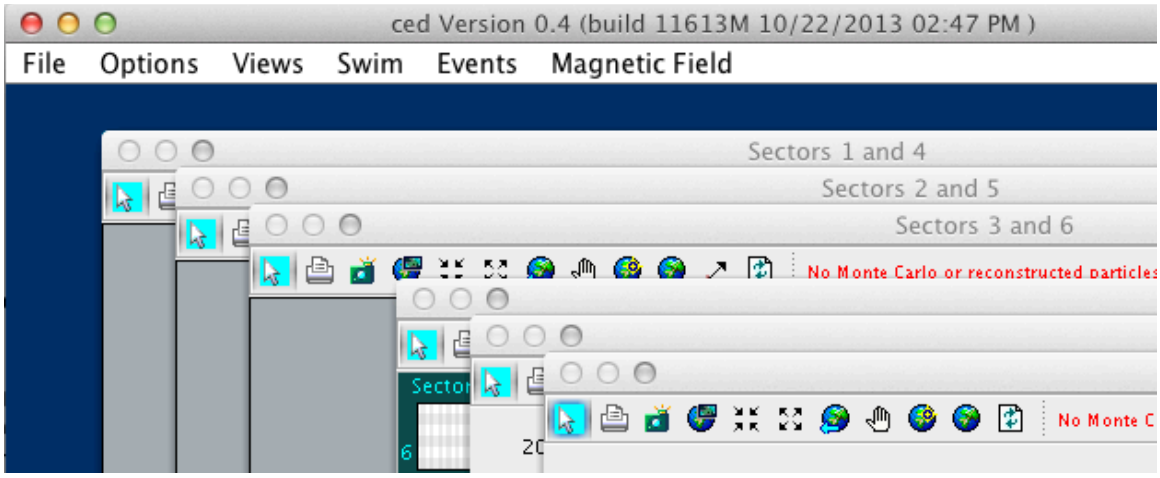
Adding build information to your application

D. Heddle

We discussed in a recent software meeting how to get build information into your Java application (typically via the jar file) in an automated manner. This is not as easy and it sounds. At least I don't know an easy way. This document describes how it is done in *ced*, so that the main *ced* title bar looks like this:



The "ced Version 0.4" is not a result of what is described below. That is put in by hand. We are interested only in what is inside the parentheses:

build 11613M 10/22/2013 0:2:47 PM

This gives us important information about the build, independent of what version we have assigned to our application, viz.,

- 11613 is the svn revision number from the clas12 repository.
- The **M** after 11613 means that this particular *ced* was built with **modified** source (M for modified) in my working directory. That is, I checked out revision 11613, made some mods, built and released. That is of course bad form: I should have checked in my changes, then built, and then the **M** would be absent. If the M is there we have no guarantee that the build could be reproduced. But we do have useful information.
- The rest of the line gives the time of the build.

Requirements:

This is done using *ant* and *svn*, so the requirements are:

You must have *ant*.

You must have an *ant* build script.

You must have command-line *svn* on your local machine.

How it works—the theory:

1) *ant* will create a resource (properties) file that will be included in your jar. That resource will contain the build string—which *ant* obtains via an *svn* call.
2) Your application reads the resource (properties) file and pulls out the string.

The build.xml: an example

Here is *ced*'s `build.xml`. The default target, *cedjar*, builds `ced.jar`. It calls the secondary target, *buildinfo*, which creates the properties file.

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="ced12gev" default="cedjar" basedir=".">

	<!-- ================================
	   target: cedjar
	   ================================ -->
	<target name="cedjar">
		<property name="version" value="v1.0"/>
		<property name="rdir" value="../../clasJlib/ced/${version}"/>
		<property name="vdir" value="${rdir}/lib"/>
		<mkdir dir="${vdir}" />

		<echo>Copying data dir</echo>
		<copy todir="${rdir}/data">
			<fileset dir="data"/>
		</copy>

		<echo>Building the ced jar</echo>
		<antcall target="buildinfo"></antcall>
		<jar destfile="${vdir}/ced.jar">
		 <fileset dir="bin"/>
		 <fileset dir="resources"/>
		</jar>
	</target>


	<!-- ================================
	     build a properties file that will be incuded in the ced jar
	   ================================ -->
	<target name="buildinfo">
	    <tstamp>
	        <format property="builtat" pattern="MM/dd/yyyy hh:mm aa"
timezone="America/New_York"/>
	    </tstamp>
	    <exec executable="svnversion" outputproperty="svnversion"/>
```

```
            <exec executable="whoami" outputproperty="whoami"/>
            <exec executable="uname" outputproperty="buildsystem"><arg value="-
a"/></exec>

            <propertyfile file="resources/ced.properties"
                comment="This file is automatically generated - DO NOT EDIT">
                <entry key="buildtime" value="${builtat}"/>
                <entry key="build" value="${svnversion}"/>
                <entry key="builder" value="${whoami}"/>
                <entry key="system" value="${buildsystem}"/>
            </propertyfile>
        </target>


</project>
```

Notice at the bottom that there is indeed a target called **buildinfo**. That does all the magic of using the *svn* command to get build information. It also puts some other information in the resource file—such as the builder and the system. Those could also be extracted but the example (see below) does not do so.

The only line that must be modified is the destination of the properties file. Here it is placed in a resources folder and given the name `ced.properties`.

The primary target is the *cedjar* target that builds the jar file. The lines with "version" "rdir" and "vdir" have to do with clasJLib bookkeeping and are not related to the build information discussed here. You see that the *cedjar* target calls the *buildinfo* task just after the "Building the ced jar" echo. Then the only mod that is necessary is to include the resources folder in the jar file as a fileset.

Modifying your code

If this all worked you have a resources folder and a properties file in your jar. You code can then, at runtime, extract the resource. Here is how ced does it.

```
    /**
     * Attempts to build a sensible build string. Assumes that a ced.properties
     * file exists either at the top level of the ced.jar or in a resources
     * folder parallel to "src" and "bin". Further assumes that the properties
     * "build" and "buildtime" are defined in that file. This is done by the ant
     * build script making calls to subversion and putting the outputs in the
     * properties file.
     *
     * @return a sensible build string containing svn versions and build time.
     * @throws IOException
     */
    private static String getBuildString() throws IOException {

        String propName = "/" + "ced.properties";
        Properties prop = new Properties();

        // try from jar
        InputStream is = Ced.class.getResourceAsStream(propName);
```

```
        if (is != null) {
                try {
                        prop.load(is);
                        is.close();
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }
        else { // try from file
                File file = new File("./resources" + propName);
                if (file.exists()) {
                        FileInputStream fis = new FileInputStream(file);
                        prop.load(fis);
                        fis.close();
                }
        }

        if (!prop.isEmpty()) {
                String build = prop.getProperty("build");
                String buildtime = prop.getProperty("buildtime");

                String s = "";
                if (build != null) {
                        s += build + " ";
                }
                if (buildtime != null) {
                        s += buildtime + " ";
                }

                if (s.length() > 0) {
                        return s;
                }
        }
        return null;
}
```

The only change you would have to make is to the line

```
String propName = "/" + “ced.properties";
```

Which you’d change in a hopfully obvious manner and

```
Ced.class.getResourceAsStream(propName);
```

Where you replace Ced by the name of the class in which you have placed this code. (Or any Class, most likely, since is a method of Object, I think.)

The code above will try to find the properties in a file if it fails to get them from a jar.

This is a pain but arguably a valuable one—since it puts build info from and center.

Feel free to ask me if you try this and are having difficulties.