Modify Local Repository Layout to Support finalName

Background

The Maven POM has an element, ${project.build.finalName}, that specifies the name of the artifact file. finalName overrides the default name: "${artifactId}-${version}".

Today, finalName is typically used when it's important for technical reasons that the artifact not include a version number. For example, Java WAR files typically load into a virtual directory named after the WAR, but we don't want the URL of the website to change when we change the version number of the WAR file.

Unfortunately, today, artifacts in the local repository always use the default name and never honor the finalName specified in the POM. e.g. the war artifact "foo" version "1.0" is stored as "foo-1.0.war" in the local repository, even if its finalName is just "foo".

Under this proposal, if a ${project.build.finalName} is specified, the file in the local repository would have that name, e.g. just "foo.war".

Justification: Windows Libraries

The finalName is especially and uniquely important for Windows libraries.

Windows has no symlink feature and no automated LD-style re-linker. Therefore, Windows libraries (DLL files) must have the same name at build-time and at run-time; DLLs will not work if they are renamed after they are built (unless they are re-renamed back to their build-time names).

Suppose you have two libraries: "myBusinessLogic" and "myUI", where myUI depends on myBusinessLogic. If you build myBusinessLogic under the name "myBusinessLogic-1.0.dll", then it's impossible to replace it with "myBusinessLogic-1.1.dll".

In other words, if you need to upgrade your DLLs at runtime, you have to build them without version numbers in their names.

Today, even if you specify an unversioned <finalName> for your DLL, the DLL in the local repository will have the wrong (versioned) name.

Objections to Windows libraries

For some odd reason, arguments from Windows libraries have been very controversial. Here are some common arguments and rebuttals.

- **Windows sucks. Maven shouldn't support DLLs.** Windows does suck, but it's too important for us to stick our head in the sand and ignore it.
    - *OK, OK, but we don't have to go out of our way to make it easy to use DLLs with Maven, do we?* You don't, but we do. You don't have to do work on this proposal, but this isn't a reason to vote against volunteers working on this proposal.
- **Unversioned file names are a bad practice. Maven should only support good practices.** Windows requires unversioned filenames; this is one of the ways in which Windows sucks (but we can't ignore it). Sometimes best practices have to yield to getting the job done; this is one of those places. Note also that Windows does burn metadata into DLLs recording the version number; you can examine the metadata by right-clicking on the file and reading the properties. It's not like the DLLs will be totally anonymous if they don't contain version numbers.
- **This is only a problem for .NET libraries, right?** No, it affects all Windows libraries.
- **Can't Windows DLL plugins take care of this, rather than affecting the core?** Many plugins deal with *any* kind of file; without Core support, we'd have to update all of those plugins to support DLLs. Letting the plugins handle it duplicates work that should be done just once in Core.

Implementation

This change is pretty simple, except for one important detail. Today, all copies of 1.0-SNAPSHOT are stored in the same directory, as foo-20090905.194035-1.jar. If all of those files were just called "foo.jar", they would conflict with each other.

So, instead, the 1.0-SNAPSHOT directory in the local repository should change to contain a "20090905.194035-1" subdirectory. Within that, you can put "foo.jar". A sibling directory "20090905.194036-2" could contain its own copy of "foo.jar" and so on.