

Decoupling of Maven Artifact

Context

The artifact mechanism was intended to be a general API for the retrieval of binary artifacts and their accompanying metadata. As such it is intended to serve as the single body of code, and a single API to manipulate artifacts. Currently it is not serving this purpose because two copies of the code exist in the trunk and the 2.0.x branch of Maven, and a proliferation of complicated uses in various plugins (the site plugin, assembly plugin, staging plugin) and Maven related tools like Archetype. Any who needs artifact resolution ends up duplicating much of the same code which means maintaining a single point of use is becoming increasingly difficult. The API currently used for artifact resolution is overly complicated and the only way to reign in its use and create something simpler is to place all the code in one place and start improving it.

Solution

Separate the artifact handling code into a separate trunks/tags/branches structure and maintain it from one location. This has the benefit of

- Decoupling the artifact handling code from Maven itself. This makes developing plugins easier that require it's use because we are no longer bound to a specific version of Maven.
- We can invite other groups to work on the code. We can eventually promote it to an official sub-project.
- We can maintain the code from a single location actually giving us a fighting chance to improve it.

Once the code is separated I think the following should be done:

- Create a facade that can be used by all external clients. In almost all cases people want to retrieve a set of artifacts for a particular scope, most predominantly the runtime scope. The best facade that I've seen is the `RuntimeDependencyResolver` created by Jan Bartel for use in the Jetty plugin. I think this should be the basis of the new API we create. At the very least to prevent people from using the artifact components directly as that's just painful. Jan's facade is far simpler.
- Deprecate all the methods in the current resolver in favor of the facade for external use. Some users may actually be grabbing metadata, and they can use the deprecated methods until we expand the officially supported API to deal with artifact metadata handling. But for the vast majority of users the facade Jan has created will do almost everything that is necessary.
- Make sure the decoupled code works with the 2.0.x branch (it should be, but how knows because the code from trunk should be used). We can ensure capatibility for as long as necessary but these will be internal methods which are known by hard core Maven developers but the one we should not subject the general population to. All you have to do is watch someone open an IDE and look at the `ArtifactResolver` and watch their facial expression to know that we should not be showing these methods to anyone.

We can easily try the decoupled code with the trunk of Maven and then attempt it in the 2.0.x branch, then attempt to use them in the plugins and release them. I think this is the only way to move toward improving the code while making sure it works with everything that has been using this code in the past.

Votes

+1: jvanzyl, snicoll, brianfox, jdcasey, kenney

+0:

-1: