

Templated POM Sections

Templating Sections of the POM

There have been a few requests to make management of dependency and plugin sets easier. I wanted to start this page to capture these use cases, and see if we can include some of these concepts in the next major release.

Dependency Templates

It's a fairly common practice to have multiple different "kinds" of subprojects in a multimodule build. For instance, multiple presentation-tier webapp modules, several supporting data-layer projects, and in some cases even a set of service-layer projects that are architecturally in between the other two.

In most cases, the overall application will be using the same technologies across the board for persistence, web presentation, remoting, etc. with a few special-case "extra" dependencies in any given subproject. This high level of overlap suggests that these projects should take an extra effort to centralize the specification of shared dependencies. Simple inheritance may not be a good option in some cases, because subprojects may be of a few distinct "flavors", each of which requiring a different set of dependencies.

My real question here is: Is there ever a case where properly layered levels of inheritance cannot make this type of application easy to manage?

jdcasey

My personal feeling is that proper inheritance can handle these issues fairly elegantly. Sure, it will amount to multiple levels in the project hierarchy, with each collection of projects that contains overlap having at least one parent POM. These parent POMs may declare dependencies explicitly; declare managed dependencies for use at the lower levels; declare usage of dependencies which are managed by an ancestor of this parent-POM; or all three.

The only case I can imagine where this wouldn't take care of it is when you have a project that straddles two project-flavor definitions. In this case, inheriting from one parent POM means having to respecify dependencies it would have gotten from the other parent POM. However, the maintenance burden represented by this project would be mitigated by using managed dependencies at an even higher level, and simply declaring usage of those managed deps that weren't inherited.

Plugin Templates

The other templating issue is for sets of plugins and plugin-configurations. The argument here is that the project is relying on its parent-POM to supply dependencies, or to describe the project itself in other ways. Injecting plugin configuration-sets into this mix can get confusing, because while all subprojects may need to inherit the dependencies, etc. NOT all of the subprojects will need the same suite of plugins. In other words, the inheritance of the **how** of the build should be orthogonal to the inheritance of **what** information.

One suggested solution is to have templates of plugins and plugin-configurations which could be referenced by a project in order for it to "adopt" a certain build behavior. Plugin templates could be specified in ancestor POMs, but the suggestion was to keep these templates separate, and have them imported in a similar manner to build extensions. When approached this way, such a template would look quite a bit like a custom lifecycle mapping.

My question here: Is there ever a place where the plugin inheritance and custom lifecycle mapping features cannot elegantly handle the scenario? Perhaps we should look at making the lifecycle mapping mechanism a little more

accessible, and promote it as a way of accomplishing this sort of "templating" need?