Repository layout for large Maven projects

Asked 16 years, 4 months ago Modified 14 years, 8 months ago
Viewed 4k times

15

I have a large application (~50 modules) using a structure similar to the following:

- Application
  - Communication modules
    - Color communication module
    - SSN communication module
    - etc. communication module
  - Router module
  - Service modules
    - Voting service module
      - Web interface submodule for voting
      - Vote collector submodule for voting
      - etc. for voting
    - Quiz service module
    - etc. module

I would like to import the application to Maven and Subversion. After some research I found that two practical approaches exists for this.

One is using a tree structure just as the previous one. The drawback of this structure is that you need a ton of tweaking/hacks to get the multi-module reporting work well with Maven. Another downside is that in Subversion the standard trunk/tags/branches approach add even more complexity to the repository.

The other approach uses a flat structure, where there are only one parent project and all the modules, submodules and parts-of-the-submodules are a direct child of the parent project. This approach works well for reporting and is easier in Subversion, however I feel I lose a bit of the structure this way.

Which way would you choose in the long term and why?

java svn maven-2

edited Aug 23, 2008 at 20:08

James A. Rosen

65.2k ● 62 ● 184 ● 263

asked Aug 21, 2008 at 14:06

Zizzencs

5,128 ● 7 ● 32 ● 31

2 Answers

Sorted by: Highest score (default)

16

We have a largish application (160+ OSGi bundles where each bundle is a Maven module) and the lesson we learned, and continue to learn, is that flat is better. The problem with encoding semantics in your hierarchy is that you lose flexibility. A module that is 100% say "communication" today may be partly "service" tomorrow and then you'll need to be moving things around in your repository and that will break all sorts of scripts, documentation, references, etc.

So I would recommend a flat structure and to encode the semantics in another place (say for example an IDE workspace or documentation).

I've answered a question about version control layout in some detail with examples at another question, it may be relevant to your situation.

Share Improve this answer

Follow

edited May 23, 2017 at 12:33

Community Bot

1 ● 1

answered Aug 21, 2008 at 18:07

Boris Terzic

10.9k ● 8 ● 46 ● 60

I think you're better off flattening your directory structure. Perhaps you want to come up with a naming convention for the directories such that they sort nicely when viewing

3

all of the projects, but ultimately I don't think all of that extra hierarchy is necessary.

Assuming you're using Eclipse as your IDE all of the projects are going to end up in a flat list once you import them anyway so you don't really gain anything from the additional sub directories. That in addition to the fact that the configuration is so much simpler without all the extra hierarchy makes the choice pretty clear in my mind.

You might also want to consider combining some of the modules. I know nothing about your app or domain, but it seems like a lot of those leaf level modules might be better suited as just packages or sets of packages inside another top level module. I'm all for keeping jars cohesive, but it can be taken too far sometimes.

Share Improve this answer

Follow

answered Aug 21, 2008 at 18:06

Mike Deck

18.4k ● 16 ● 70 ● 94