TabledAvalonDiscussions

Tabled Discussions

This document is to keep track of discusstion points related to Avalon 5 development. When things get to a point where they can be focused on in the Avalon-Dev site without being considered noise. Just follow the format already outlined below.

Should we incorporate Commons Logging?

This move would increase synergy with other projects and reduce the amount of code that we need to maintain. The issue was raised by Mauro Telivi, and we need further communication to figure out what is the best solution.

Update by LSD

Additional discussion as part of the framework 4.1.4 release. We didn't add a wrapper around commons-logging Log class, but it is easy to do and should be considered again for the next release.

Its turned out we cannot really migrate to Log at all in 4.x, as it doesn't provide getChildLogger(). And it makes no sense for commons either to add in that method into the Log interface (because of backwards compatibility).

Note discussions about this with commons were quite fun and productive, so good synergy vibes nonetheless 😃

PRO:

- (bloritsch) It has the appeal of synergy with other communities.

CON:

- (bloritsch) I think that synergy comes at too high a price for Avalon's architecture.
- (pdonald) Nah. For the sharing of one interface it does not seem useful to couple the projects. Commons also passes down objects which is not a great idea as it means that you are binding to specific subsystems if you use that "feature".

Topic: Should we use Maven for the build?

Our current build architecture is chaotic to say the least. We have no less than three methods of generating documentation, and serveral fairly complex build scripts. Maven helps smooth out the inconsistencies in the approach, and still allows for plugging in a documentation building tool like Fortress. Most importantly, it will get rid of many JAR files in our CVS directory, and simplify the build process for our users.

Update by LSD=

We've started restructuring the build structure to get rid of jars in CVS by using standard ant tasks to get stuff from the maven repo. We've come a large way for generating docs (now using forrest in most places, still anakia in some, but no "custom cocoon" anymore).

The build structure is still quite complex and inconsistent. Its still a good idea to move away from "just ant". Jason suggested we wait for beta9 if we want to move.

PRO:

- (bloritsch) We are in desparate need of a unified and simple build architecture. IMO, Maven is the best choice we have available. Let's simplify. Speed is only part of the issue here. We need to be able to manage project dependencies properly, and Maven's repository is a good solution. We can treat each individual project separately for the time being, so there is no heirarchy to worry about. When Maven does support it, and I think they will, we can consolidate things again.
- (pdonald) I am using maven to build the site for all the stuff I am moving out of Avalon. It is absolutely fantastic. There is a few hickups and it is definetly still alpha quality but it is better than what we have now. The great thing about it is that a lot of people have already done all the integration work and it is easy enough to get coverage, junit etc all integrated in.

CON:

- (pdonald) The negatives:

```
- speed (not a great problem in CVS version because of
```

```
  "console" plugin)
```

```
- project inheritance is not fully done so we have to
```

```
  copy a 3 line property file around when we are using;
```

```
  however, this is supposed to be fixed before next release
```

```
- inheritance of maven.xmls is not possible as far as I
```

```
  know so you can end up copying some around.  Not sure
```

```
  if this will be fixed .. but will become irrelevent if
```

```
  next point picked up ...
```

```
- painful to manage your own plugins if not in maven CVS.
```

```
  I believe this will hopefully be fixed sometime in the
```

```
  future ... maybe :)
```

I am using their last release and it seems to work well. However it may be best to wait till next release before converting Avalon (Just to avoid a few bugs that the last Maven has). However I suggest we keep our current ANT build system around until Maven gets the ability to easily install plugins on per project basis at which point we can dump our exisitng build system.

Topic: Should we unify the CVS structures?

We currently have ~9 CVS structures. The purpose was to separate the different projects and make it very clear what code belongs to which project. That project also helps us identify areas of tight coupling and what projects depend on which other projects. That can still be done in one CVS structure, using directories to distinguish the projects.

Update by LSD

Well, testlet allready nuked, and we want to keep avalon, avalon-site and avalon-sandbox even if we do a unification, so definately not one repo. Doesn't apply to a SVN setup though.

PRO:

- (bloritsch) We have some projects that are no longer supported or needed (like Testlet). It would be a good way to clean out the cruft, and help manage the focus and scope of Avalon.
- (pdonald) Yes, BUT Maybe we should wait for subversion (subversion.tigris.org). It looks like a much better tool and if we go to that we can do the big re-arrange then.

CON:

- (lsimons) its lots of work, and we've been restructuring

our asses off, and it has led to headaches where phoenix depended via-via on sandbox code and stuff like that. We've done enough restructuring.

- (lsimons) a very large CVS module takes a long time to

update!

Topic: Should we remove unused sets of code?

Excalibur is rife with obsolete, competing, and confusing stuff. Also mixed in there are some *really* useful gold nuggets. By getting rid of the failed experiments, we can focus our energies and web site on what is good.

Update by LSD

General consensus seems to be: yes, we should. Excalibur is looking a lot cleaner already!

PRO:

- (bloritsch) For competing technologies, they should be merged back into the parent project they came out of. Nine times out of ten, the only maintainers and developers for those projects support the parent project. We should also archive and remove all the CVS stuff for our deprecated packages. We aren't maintaining them any more--and they are still draining energies (of users, etc.). It will help clean the slate.

CON:

- (bloritsch) What projects are we removing? It is a potential sticking point, and we still need a copy of the old code.