Should I keep my project files under version control? [closed]

Asked 16 years, 3 months ago Modified 9 years, 2 months ago

Viewed 20k times

92

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 13 years ago.

Should I keep project filesm like Eclipse's .project, .classpath, .settings, under version control (e.g. Subversion, GitHub, CVS, Mercurial, etc)?

java eclipse version-control ide

Share

Improve this question

Follow

edited Oct 22, 2015 at 9:02

jamesmortensen

34k ● 11 ● 101 ● 123

asked Sep 22, 2008 at 17:05

cretzel
20.1k ● 19 ● 60 ● 73

See also stackoverflow.com/questions/1429125/… – VonC Sep 16, 2009 at 6:04

12 So very constructive – QED Jun 20, 2013 at 20:43

12 Answers

Sorted by: Highest score (default)

99

You do want to keep in version control **any portable setting files**,
meaning:
Any file which has no absolute path in it.
That includes:

- .project,
- .classpath (*if no absolute path used*, which is possible with the use of IDE variables, or user environment variables)
- **IDE settings** (which is where i disagree strongly with the 'accepted' answer). Those settings often includes **static code analysis rules** which are vitally important to enforce consistently for any user loading this project into his/her workspace.
- IDE specific settings recommandations must be written in a big README file (and versionned as well of course).

Rule of thumb for me:
You must be able to load a project into a workspace and have in it everything you need to properly set it up in your IDE and get going in minutes.
No additional documentation, wiki pages to read or what not.
Load it up, set it up, go.

Share Improve this answer

Follow

answered Sep 23, 2008 at 6:25

VonC
1.3m ● 558 ● 4.7k ● 5.6k

@Rich: thank you. For the other readers, see Rich's answer to the same question one year later: stackoverflow.com/questions/1429125/… – VonC Sep 16, 2009 at 6:04

3 key phase "...get going in minutes..." – Eric Labashosky Oct 24, 2009 at 11:00

3 So, the VC repository should have the config files for every IDE? NetBeans, Eclipse, Emacs, vi, whatever else? I specifically disagree with the idea that these files because the developer should be responsible for setting up their own IDE. – RHSeeger Oct 28, 2009 at 20:53

3 @RH - "... should be responsible for setting up their own IDE". That's fine, until some clown forgets to set their Eclipse code style properties .... and checks in source files with TABs in them. Grrrr!! – Stephen C Oct 29, 2009 at 7:39

2 I disagree as well - if you're using CI, multiple versions, platforms, IDEs, etc. this breaks DRY pretty badly. Esp. since different plugins/settings have a big influence on what ends up here. – jayshao Apr 11, 2010 at 1:58

31

.project and .classpath files yes. We do not however keep our IDE settings in version control. There are some plugins that do not do a good job of persisting settings and we found that some settings were not very portable from one dev machine to the next. So, we instead have a Wiki page that highlights the steps required for a developer to setup their IDE.

Share Improve this answer

Follow

answered Sep 22, 2008 at 17:12

Kevin

1,247 ● 1 ● 13 ● 17

2 -1 I'd suggest to file bug reports for those plugins instead of allowing them to waste the time of thousands of people. And then, I'd put all the stable settings files under version control and trim down that Wiki page to the bare bones.
– Aaron Digulla May 22, 2013 at 10:17

18

These are what I consider to be generated files, and as such I never place them under version control. They can be different from machine to machine and developer to developer, for instance when people have different Eclipse plugins installed.

Instead, I use a build tool (Maven) that can generate initial versions of these files when you make a new checkout.

Share Improve this answer

Follow

answered Sep 22, 2008 at 17:13

Chris Vest
8,672 ● 3 ● 38 ● 43

To be precise: mvn eclipse:eclipse will generate an appropriate .classpath and .project. You can pass it -DdownloadSources=true and -DdownloadJavadocs=true. – Aleksandar Dimitrov Sep 23, 2008 at 8:18

you can also configure the eclipse plugin in your pom to always download sources and javadoc. – Chris Vest Jun 9, 2009 at 22:03

1 -1 This works as long as you don't ever change the project settings in your IDE. And the danger is: If you change them, you won't notice because the files aren't versioned. So I'm very much tempted to vote this down. Only ever do this for really simple projects like those you don't intend to share with anyone. In a team, the defaults usually don't work and asking each developer to change their setup is a sure-fire recipe for chaos. – Aaron Digulla May 22, 2013 at 10:21

Aaron, this works even if you do change the project files in your IDE. What happens next, is that you don't push your changes out to your unsuspecting team. IDE projects are prone to contain machine and developer specific information, that won't work for everybody. I find that the more plug-ins you use, and the more complex your use of the IDE becomes, the worse this gets. People should know how their tools work, and have control over their configuration. I *do*, however, agree that this approach is not without its own set of problems. – Chris Vest May 22, 2013 at 15:44

Also, fixing merge conflicts caused by plug-ins that frivolously edit these files gets old pretty quick. – Chris Vest May 22, 2013 at 15:50

I am torn between two options here.

7

On one hand, I think that everyone should be free to use the set of developemnt tools they are most productive with, as long as all source artifacts are stored in version control, and the build script (say ANT or Maven) ensures standards compliance by specifying exactly which JDK to use, which versions of which third party libraries to depend upon, running style checks (e.g. checkstyle) and running unit tests etc.

On the other hand, I think so many people use the same tools (e.g. Eclipse) and often it is much better to have some things standardised at design time instead of build time - for example Checkstyle is far more useful as an Eclipse plugin than as an ANT or Maven task - that it is better to standardise on the set of development tools and a common set of plugins.

I worked on a project where everyone used exactly the same JDK, same version of Maven, the same version of Eclipse, the same set of Eclipse plugins and the same configuration files (e.g. Checkstyle profiles, code formatter rules etc.). All of these were kept in source control - .project, .classpath and everything in the .settings folder. It made life really easy during the initial phases of the project when people were continually tweaking the dependencies or the build process. It also helped immensely when adding new starters to the project.

On balance, I think that if there are not too many chances of a religious war, you should standardise on the basic

set of develop tools and plugins and ensure version compliance in your build scripts (for example by explicitly specifying the Java version).I don't think that there is much benefit to storing the JDK and the Eclipse installation in source control. Everything else that is not a derived artifact - including your project files, configuration and plugin preferences (particularly code formatter and style rules) - should go into source control.

P.S. If you use Maven, there is an argument for saying that the .project and .classpath files are derived artifacts. This is only true if you generate them every time you do a build, and if you have never had to tweak them by hand (or inadvertently changed them by changing some preferences) after generating them from the POM

Share Improve this answer

Follow

answered Sep 23, 2008 at 9:56

Vihung

13.4k ● 18 ● 67 ● 95

6

No, because I only version control files that are needed to build the software. Besides, individual developers may have their own project-specific settings.

Share Improve this answer

Follow

answered Sep 22, 2008 at 18:27

John Topley

115k ● 47 ● 199 ● 240

5

No, I'm a heavy Maven user and use the Q for Eclipse plugin that creates and keeps .project and .classpath updated. For other things such as settings for plugins I usually mantain a README or Wiki-page about that.

Also those I've worked with that prefer other IDEs just use the Maven-plugins to generate the files needed to keep their IDE (and themselves) happy.

Share Improve this answer

Follow

answered Sep 22, 2008 at 18:24

Andreas Holstenson

1,428 ● 9 ● 7

5

This is all opinion, I suppose - but best practices over the years indicate that files specific to a given IDE shouldn't be stored in source control, unless your entire organization is standardized on one IDE and you never have any intent on switching.

Either way, you definitely don't want user settings stored - and .project can contain settings that are really developer specific.

I recommend using something like Maven or Ant as a standardized build system. Any developer can get a classpath configured in their IDE in a few seconds.

Share Improve this answer

Follow

answered Sep 23, 2008 at 4:50

Kevin Day

16.4k ● 8 ● 48 ● 71

1

Yes, except for the .settings folder. Committing the other files works well for us. There is a similar question here.

Share Improve this answer

Follow

edited May 23, 2017 at 12:09

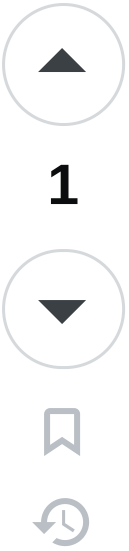
Community Bot

1 ● 1

answered Sep 22, 2008 at 17:06

Jay R.

32.2k ● 17 ● 55 ● 61



Although I generally agree on the "do not version generated files" approach, we have problems with it and have to switch back.

> Note: I am also interested in VonC's answer, particularly about the "get Eclipse up within minutes" point. But it is not decisive to us.

Our context is Eclipse+Maven, using m2eclipse plug-in. We have a common development environment, with common directories as much as possible. But it happens sometimes that someone would try a plug-in, or change little things in the configuration, or import a second workspace for a different branch...

Our problem is that the generation of .project is done when importing a project in Eclipse, but is **not updated in all cases later on**. It's sad, and probably not permanent as the m2eclipse plug-in will improve, but it's true right now. So we end up having different configurations. What we had today was that: *several natures were added to many projects on some machine, which then behaved much differently* :-(

The only solution we see is to version the .project file (to avoid risks, we'll do the same for .classpath and .settings). That way, when one developer changes her pom, the local files get updated using m2eclipse, all of them get committed together, and other developers will see all changes.

> Note : in our case, we use relative file names, so we have no problem to share those files.

So, to answer your question, I say yes, commit those files.

I also liked:

- Rich Seller's answer

Share Improve this answer

Follow

edited May 23, 2017 at 12:09

Community Bot

1 ● 1

answered Oct 28, 2009 at 17:05

KLE

24.1k ● 4 ● 57 ● 62

"...changes her pom using m2eclipse..."? What does m2eclipse have to do with the pom file? Certainly it uses it, but changes to the pom should be independent of the plugin. – RHSeeger Oct 28, 2009 at 20:55

@RHSeeger Thanks, I will improve clarity on this part of my answer. – KLE Oct 29, 2009 at 7:23

0

It seems like these project files can change over time as you work on a project so yes, I place them under version control.

Share Improve this answer

Follow

answered Sep 22, 2008 at 17:06

neuroguy123

1,365 ● 10 ● 14

0

Yes. Everything but build output.

Share Improve this answer

Follow

answered Oct 15, 2008 at 6:33

nitind

20k ● 4 ● 36 ● 44

0

We use IntelliJ IDEA, and keep '.sample' versions of the project (.ipr) and module (.iml) files under version control.

Somewhat bigger thing here is *sharing and re-use* than versioning, IMHO. But if you are going to share these configurations, what better place to put them than the repository, right next to everything else.

Some advantages of shared & versioned project files:

- You can check out any tag/branch and start working on it quickly
- Makes it easier for a new developer to first set up the development environment and get up to speed

- This better adheres to DRY, which is always deeply satisfying. Before this, *all* developers had to set these things up every now and then, essentially doing repeated work. Of course everyone had their own little ways to avoid repeating themselves, but looking at the team as a whole, there was a lot of duplicated effort.

Note that in IDEA these files contain configurations such as: what are the "source" and "test source" dirs; everything about external dependencies (where are library jars located, as well as related sources or javadocs); build options, etc. This is stuff that does *not* vary from developer to developer (I disagree with this quite strongly). IDEA stores more personal IDE settings elsewhere, as well as any plugin configurations. (I don't know Eclipse that well; this may or may not be quite different.)

I agree with this answer that says:

> You must be able to load a project into a workspace and have in it everything you need to properly set it up in your IDE and get going in minutes. [...] Load it up, set it up, go.

And we have it like this, thanks to versioned project files.

Share Improve this answer

Follow

edited May 23, 2017 at 12:09

Community Bot

1 ● 1

answered Jun 9, 2009 at 13:10

Jonik

81.7k ● 76 ● 269 ● 379