# CodeStyle

## Code Style (DRAFT)

This page is the place where we collect and evaluate code style rules for Commons. The goal is to use this document as a base for a Commons wide Checkstyle configuration, and perhaps templates for IDE's as well.

Feel free to add new rules to the table below. The rules will be tagged, now or later, on its severity. The proposed tags are:

- *info* - Follow this if you like. If you don't it's no big deal
- *warning* - You should consider following this rule
- *error* - This must must be fixed before the next release

| **Rules** | **Severity** |
|---|---|
| No tabs allowed in source files | error |
| Each statement should be on a separate line (simplifies debugging) | error |
| Imports: No wildcards | error |
| Imports: Order by groups: java, javax, org, com | warning |
| Imports: Order in alphabetical order with a group | warning |
| Indentation: (Java) use 4 spaces | warning |
| SVN keywords: $Date$ should not be used | warning |
| Indentation: (POM) prefer 4 spaces, allow 2, but be consistent within a file (1) | info |
| JavaDoc: @author Tag should not be used | info |
| JavaDoc: @deprecated Tag must include version where first deprecated, and a link to the replacement (if any) | warning |
| JavaDoc: @since Tag must be used to document new classes and methods where these form part of the public API | warning |
| Object visibility should be the minimum required (See notes) | warning |
| Prefer immutable classes as these are automatically thread-safe | info |
| Mutable fields must be private (See notes) | error |
| Mutable fields (except primitives) should not be exposed via getters/setters (See notes) | warning |
| Add your rule here | warning |

Notes:

1. POMs tend to have quite deeply nested elements, and many elements can be long and awkward to wrap, so using 2 spaces is sometimes easier to read.
2. The SVN $Date$ keyword should not be used, because it relies on the clients locale. Use the $Id$ keyword instead (see http://markmail.org/message/zx4ii6pq4iin2tol).
3. Document authors in POM, not in source files (see http://markmail.org/message/k34w6gsx5iic45z2).
4. Object visibility: once code is released, it can be impossible to reduce the visibility of fields, classes, methods without breaking compatibility, so initial releases should use the minimum visibility possible.
5. Mutable data: this increases the difficulty of ensuring thread-safety (including safe publication of changes). Data should be confined to a class; mutation should only be allowed via a setter which can ensure thread-safety. Be careful that the getter does not expose array contents (which are always mutable)
6. Even constants can cause problems: the Java compiler can inline constant values from another class. So if the constant should ever change, classes that are not recompiled could contain the old value.
7. Array entries are always mutable. Only empty arrays are immutable; they can be safely shared.
8. If a reference to a mutable object is exposed via a getter, then it can be modified in a way that breaks thread-safety.
9. If a mutable object is saved by a setter and the caller keeps a reference to the object then it can be changed without using the setter, bypassing any restrictions the setter would have enforced.