

LU01b - Linting & Formatting

Formatting

Formatter sorgen dafür, dass der gemeinsame Code einheitlich formatiert wird. Oftmals besitzen IDE bereits eine Standard-Formatierung oder die Möglichkeit, Formatierungsregeln mit Entwicklern mit derselben IDE auszutauschen.

Beispiele von IDE-spezifischen Formatierungen:

- **IntelliJ:** Code Style XML unter `.idea/codeStyles/...`
- **Visual Studio Code:** Settings unter `.vscode/settings.json`
- **Eclipse: Code Style Formatter:** Eclipse Code Formatter XML
- **Diverse IDE:** `.editorconfig` (aber nicht alle Regeln funktionieren aber bei allen IDE's)

IntelliJ Project.xml	VS Code settings.json	Eclipse java-formatter.xml	.editorconfig
<pre>... </JavaCodeStyleSettings> <codeStyleSettings language="JAVA"> <!-- Tab size --> <indentOptions> <option name="INDENT_SIZE" value="4"/> <option name="TAB_SIZE" value="4"/> <option name="USE_TAB_CHARACTER" value="false"/> </indentOptions> <!-- New line at end of file --> <option name="INSERT_FINAL_NEWLINE" value="true"/> </codeStyleSettings> ...</pre>	<pre>{ // Tab size "editor.tabSize": 4, "editor.insertSpaces": true, // New line at end of file "files.insertFinalNewline": true, // Java-specific "editor.formatOnSave": true, // Remove unused imports "editor.codeActionsOnSave": { "source.organizeImports": true }, }</pre>	<pre>... <profile kind="CodeFormatterProfile" name="Project Formatter" version="12"> <!-- Tab size --> <setting id="org.eclipse.jdt.core.formatter.tabulation.size" value="4"/> <setting id="org.eclipse.jdt.core.formatter.indentation.size" value="4"/> <setting id="org.eclipse.jdt.core.formatter.tabulation.char" value="space"/> <!-- New line at end of file --> <setting id="org.eclipse.jdt.core.formatter.insert_new_line_at_end_of_file_if_missing" value="true"/> </profile> ...</pre>	<pre>... [*] indent_style = space indent_size = 4 tab_width = 4 end_of_line = lf insert_final_newline = true trim_trailing_whitespace = true ...</pre>

Das Problem von solchen Konfigurationen ist ihre Abhängigkeit zu der IDE. Sobald ein Teammitglied eine andere IDE nutzt, kommt es früher oder später zu Formatierungsunterschieden, da viele (besonders komplexere) Regeln nicht für alle IDE's verfügbar ist. Ein weiteres Nachteil ist, dass solche Formatierungen in der Regel nicht in die Build-Pipeline eingebaut werden können.

Daher würde ich IDE-unabhängige Formatter bei komplexeren Formatierungsregeln bevorzugen. Einfache Regeln (Einrückungen, Trailing New Line etc.) sind oft in den gängigsten IDE's vorhanden.

Beispiele von IDE-unabhängigen Formatter-Bibliotheken:

- Javascript, Typescript, HTML, CSS, JSON → Prettier
- Python → black, pep8, ...
- Java → Google Java Format

Linting

Linting (dt. „fusseln“) bezeichnet das automatische finden und teilweise sogar korrigieren von möglichen Fehlern o.Ä. anhand von einem definierten Regelset. Entsprechende Bibliotheken sind für diverse Sprachen erhältlich.

Gängige Linter-Bibliotheken

- Javascript, Typescript → ESLint(<https://eslint.org/>) ,
TSLint wurde eingestellt und ist daher nicht mehr empfohlen
- Python → Pylint (<https://www.pylint.org/>)
- Java → PMD (<https://pmd.github.io/>)
- Diverse Sprachen → SonarQube for IDE (SonarLint)

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/de/modul/ffit/3-jahr/cicd/learningunits/lu01/b>

Last update: **2026/01/20 15:28**

