Wicket Source Code Style

Bookmarkable link

Table of contents

Overview

At present, the definitive Wicket Code Style may be considered to be that defined by the `/trunk/jdk-1.4/wicket/.settings/org.eclipse.jdt.core.prefs` file, which is all well & good as long as you're using Eclipse! At present, as all the main committers appear to use that IDE, this hasn't a big problem, but this page attempts to document the main points to help those using alternative formatters.

Note - the intention here is to act as an aid for any developers interested in replicating the Wicket Code Style in something other than Eclipse, in order to help with supplying patches, etc. Even with IDEA (and as anyone with IDEA will see, I've only mentioned about 1/2 the settings available), you're almost certainly not going to be able to 100% replicate the Eclipse formatter, as there are too many 'edge cases', even when JavaDoc formatting is disabled. (That's a whole can of worms in itself...)

- Wicket codestyle for IDEA 7 - Wicket.xml
  - Place in your local version of `$HOME/.IntelliJIdea70/config/codestyles/`

Java Style

General - Tabs and Indents

Option	Value	Description
Use Tabs or Spaces	Tabs	
Smart Tabs	✓	Inserts tabs for indentation and reformatting but the fine alignment to a necessary column is done using spaces. Otherwise, only tabs are used.
Tab Size	4	Number of spaces included within a tab
Indent	4	Number of spaces to insert for each indent level
Continuation indent	8	Number of spaces to insert in case of a construct break to be inserted at the next line
Right Margin (columns)	120	Column to wrap at

Alignment & Braces

Class declaration	On next line	
Method declaration	On next line	
Other	On next line	
Special 'else if' treatment	✓	i.e., keep on same line

Indent 'case' from 'switch'	✓	

Place 'else' on new line	✓	

Blank Lines

Keep Blank Lines

In declarations	0	Maximum number of blank lines that will be kept before class or method declarations after reformatting
In code	0	Maximum number of blank lines that will be kept among items within classes or methods after reformatting
Before '}'	0	Maximum number of blank lines that will be kept before the closing code block brace after reformatting

Blank Lines

Before package statement	0
After package statement	1
Before imports	1
After imports	1
Around class	1
Around field	0
Around method	1
After class header	0

Spaces

Before Parentheses

Method call parentheses	✗
Method declaration parentheses	✗
'if' parentheses	✓
'while' parentheses	✓
'for' parentheses	✓
'catch' parentheses	✓
'switch' parentheses	✗
'synchronized' parentheses	✗
Annotation parentheses	✗

Around operators

All operators	✓

Before left brace ({)

All left braces	✓

Within Parentheses

All parentheses	✗

Within Ternary Operator

i.e. `a ? b : c;`

Before '?'	✓
After '?'	✓

Before ':'	✓
After ':'	✓

Other

After comma (e.g. in argument lists)	✓
After semicolon (e.g. in 'for' loops)	✓
After type cast	✗

Imports

Use single class import	✓	Import a particular class from a package is imported rather than '.*'
Use fully qualified class names in Javadoc	✓	Use fully qualified class name rather than importing
Class count to use import with '*'	99	Specify number of classes to import from a package until switching to '.*'
Names count to use static import with '*'	99	Specify number of classes to import from a package until switching to '.*'

Layout

Imported classes will be grouped as below and sorted alphabetically within a grouping

- java.*...
- <blank line>
- javax.*...
- <blank line>
- <all other imports>

JavaDoc

Alignment

Use these options to define how Javadoc comments are aligned.

Align parameter description	✓	If checked, parameter descriptions are aligned against the longest parameter name. Otherwise, the description is separated from the parameter name by a single space.
Align thrown exception description	✓	If checked, thrown exception descriptions are aligned against the longest exception name. Otherwise, the description is separated from the exception name by a single space.

Blank lines

Use these options to define where blank lines are inserted in the Javadoc comments.

After description	✓	If checked, a blank line is automatically inserted after the description section of Javadoc comment.
After parameter descriptions	✗	If checked, a blank line is inserted after the group of @param tags.
After return tag	✗	If checked, a blank line is inserted after the @return tag.

Invalid tags

Keep invalid tags	✓	If checked, the @invalidTag is preserved.
Keep empty @param tags	✓	If checked, the @param tags without description are preserved.
Keep empty @return tags	✓	If checked, the @return tags without description are preserved.
Keep empty @throws tags	✓	If checked, the @throws tags without description are preserved.

Other

Enable leading asterisks	✓	If checked, each line of a Javadoc comment starts with an asterisk.
Use @throws rather than @exception	✓	If checked, @throws tag is used.
Wrap at right margin	✗	If checked, the text that exceeds the right margin, is wrapped to the next line.
Generate </p> on empty lines	✗	If checked, </p> tag is automatically inserted to an empty line

Keep empty lines		If checked, the empty lines, inserted by the user, are preserved
Do not wrap one-line comment		If checked, short comments will be kept in one line with the opening and closing tags