# Developer Guidelines


1 Ergo Development
2 Eclipse Workspace Settings
    2.1 Java Compiler
    2.2 Java Errors/Warnings
    2.3 Additional requirements for Java source
    2.4 Ergo formatting and cleanup profiles for Java
3 Code Policy
4 Adding a New Plug-in
    4.1 Externalizing Strings
5 Contributions from ergo-dev members


## Ergo Development

Ergo is built upon the Eclipse Rich Client Platform (RCP), a Java based plug-in architecture for building client applications on top of the Eclipse platform. If you are not familiar with Java or want a deeper understanding of some Java concepts, we recommend starting with the Java tutorials found at the link in the Java section below. For those unfamiliar with the Eclipse Rich Client Platform (RCP) development and Git source control, it is recommended that you go through the tutorials in those sections as well. There are two RCP tutorials, the first is for 3.x and the other is for 4.x. Although 3.x is considered deprecated, it is the platform that Ergo is built upon and is still widely used until 4.x is considered feature complete and applications can be transitioned. The second tutorial is primarily on 4.x; however, most of the concepts are applicable to both (e.g. extension points, extensions, etc) so we recommend going through both RCP tutorials.

- Java
  - https://docs.oracle.com/javase/tutorial/
- RCP
  - 3.x - http://www.vogella.com/tutorials/Eclipse3RCP/article.html
  - 4.x - http://www.vogella.com/tutorials/EclipseRCP/article.html
- Git
  - http://www.vogella.com/tutorials/Git/article.html
- EGit
  - http://www.vogella.com/tutorials/EclipseGit/article.html

## Eclipse Workspace Settings

These are some additional configuration settings for Eclipse and Java source requirements when developing Ergo. You can find these under Window > Preferences

### Java Compiler

Developers should set the preference Java > Compiler > JDK Compliance to 1.7 and do a full re-build when prompted to do so.

### Java Errors/Warnings

Developers should override default compiler error/warning and use project specific errors/warnings. These errors should be enabled:

- Method with a constructor name - Error
- Non-externalized strings (missing/unused $NON-NLS$ tag) - Warning
- Assignment has no effect - Error
- Possible accidental boolean assignment - Error
- finally does not complete normally - Error
- Using a char array in string concatenation - Error
- Null pointer access - Error
- Potential null pointer access - Warning
- Unused Import - Error

**Patches with errors listed above including API errors will not be accepted without corrections.**

### Additional requirements for Java source

- Use interfaces as much as possible
- All interfaces and methods must include a Javadoc comment
- Interface implementations must include a non-Javadoc comment (using Generate Element Comment)
- All classes, methods and fields should have an access modifier (public, protected, private)

- Use the following class, method and field modifier order:
    1. Access modifier
    2. abstract
    3. static
    4. final
    5. transient
    6. volatile
- All source should be formatted using the "Ergo" code formatter profile (see below)
- All members should be cleaned up using the "Ergo" profile (Source->Clean Up)
- Statement blocks should always be used, even for single line 'if', 'for', etc.
- Column width should be set to 132

All source files must also start with an approved license and copyright declaration:

```
/*******************************************************************************
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/.
 *
 * Contributors:
 *          {INITIAL AUTHOR}- initial API and implementation and/or initial documentation
 *******************************************************************************/
```

## Ergo formatting and cleanup profiles for Java

The Ergo formatter and clean-up profiles are available here To use the profiles,

- Unzip on your machine to find two files: ergo_format.xml and ergo_cleanup.xml
- **Formatter:** in Window > Preferences > Java > Code Style > Clean Up and select "Import" and import the ergo-cleanup.xml file. The active profile should say "ERGO".
    - To use: Select a source file in Package Explorer (or container/folder/etc.) and select Source menu > Format
- **Cleanup** in Window > Preferences > Java > Code Style > Formatter and select "Import" and import the ergo-formatter.xml file. The active profile should say "ERGO".
    - To use: Select a source file in Package Explorer (or container/folder/etc.) and select Source menu > Cleanup ...
- Optionally **Set the editor to apply the formatting changes automatically on save** for convenience and consistency in use in Window > Preferences > Java > Editor > Save Actions. Check "Perform the selected actions on save", check "Format source code", and check "Additional actions"

# Code Policy

If you haven't already done so, please read the section on code policy here: Contributing Code to Ergo before writing software for the Ergo project.

# Adding a New Plug-in

If you are contributing new plug-ins to Ergo, it is recommended that you include unit tests that verify everything is working properly. In addition the the above code guidelines, please do the following:

- Add an about.html file to the plug-in **(about.html)**
    - Open up your about.html and make sure the date correspond with the release date of the version of ERGO. For example, if the version of ERGO is on July 10, 2014 then the date on the about.html should correspond to that date.
- Add the plug-in to a feature so it gets built.
- Announce the new plug-in on ergo-dev so developers know to import the new plug-in

## Externalizing Strings

# Contributions from ergo-dev members

Contributions from members of ergo-dev should be handled following these guidelines: Handling Git Contributions