**Refactoring commands:**

<http://stackoverflow.com/questions/1804839/eclipse-most-useful-refactorings>

**Rename** - because giving things meaningful names is the best way to write self-documenting code. Shift+Alt+R

**Extract method** - whenever a method gets too long. Shift+Alt+M

**Extract constant** - because magic numbers are bad. Shift+Alt+T (refactoring menu, there's no direct shortcut).

**Inline/introduce variable** - to remove clutter from methods. Shift+Alt+I (inline), Shift+Alt+L(introduce)

My favorites (in order of using):

1. Rename (Alt-Shift-R, or Ctrl-1 for faster in-file renaming)  
   Good renaming variables,methods,etc. without side effects.
2. Extract Variable (Ctrl-1, Alt-Shift-L)  
   Good for splitting a quick-made-100-character-line to separate steps.
3. Extract Method (Alt-Shift-M)  
   Create a method out of some code without any side effects.
4. Split Variable Declaration (Ctrl-1)  
   Good when you initialize a variable at the declaration and now find out, that the initialisation needs to be in a try- or if-block.
5. Change Method Signature (Alt-Shift-C)  
   The handy Swiss Army Knife of method signature manipulation, including default values for new parameters.
6. Pull Up/Push Down Pull methods and variables to a generic interface or superclass or push it down to a subclass
7. Extract Interface/Superclass  
   Extract an interface or a superclass out of the current class. Very handy.

**How do I externalize my Eclipse plugins?**

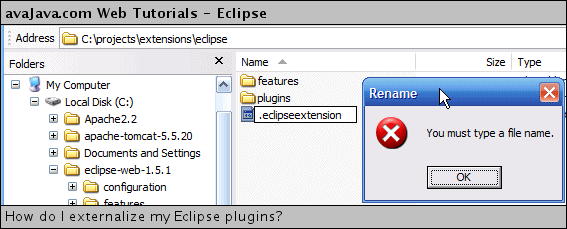
**http://www.avajava.com/tutorials/lessons/how-do-i-externalize-my-eclipse-plugins.html**

It can be useful to 'externalize' the plug-ins that you have added to Eclipse[S](http://www.eclipse.org/)[W](http://en.wikipedia.org/wiki/Eclipse_%28software%29). One way of adding plug-ins to Eclipse is to simply download a plug-in and drop it in your Eclipse plug-ins directory (ie, 'C:\eclipse-web-1.5.1\plugins').

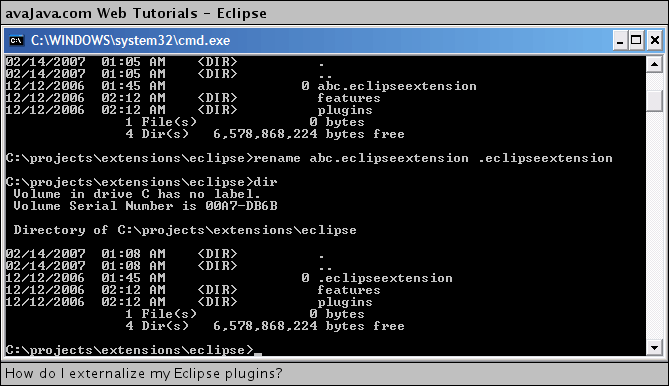
However, there are problems with adding plug-ins in this fashion. To begin with, it can be difficult to tell which plug-ins you've added to Eclipse, and which plug-ins came with Eclipse. For example, with the Eclipse WTP installation mentioned above, there are 375 folders and jar[W](http://en.wikipedia.org/wiki/JAR_%28file_format%29) files within the 'C:\eclipse-web-1.5.1\plugins' directory! Also, if you ever install another version of Eclipse (such as eclipse-web-1.5.2), it can be a painful process to go through your plugins directory to figure out which plugins you want to copy/paste over into the new plugins directory.

One solution to these problems is to 'externalize' your plug-ins to a separate directory outside of your Eclipse installation directory. In your 'C:\projects' directory, create an 'extensions' directory. Inside this directory, create an 'eclipse' directory. Within this directory, create a 'features' directory and a 'plugins' directory.

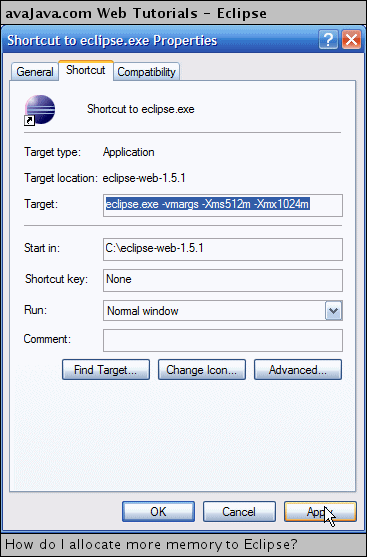
We also need a '.eclipseextension' blank file within the 'C:\projects\extensions\eclipse' directory that we created. If you try creating a file with no name and just an extension using Windows Explorer, you'll probably see a message like the following:



So how do we create a file on Windows that is just an extension? There are different ways, but a quick way is to give the file a name and extension, and then open up a command prompt and 'rename' the file to have just an extension:



**How do I allocate more memory to Eclipse?**



# [Eclipse + Maven: force Execution Environment “JavaSE-1.6” instead of fixed JDK](http://stackoverflow.com/questions/4759242/eclipse-maven-force-execution-environment-javase-1-6-instead-of-fixed-jdk)

|  |  |  |  |
| --- | --- | --- | --- |
| I'm using Eclipse as IDE and Maven outside of it to keep all dependencies up to date and update the Eclipse-project as such. In Eclipse I usually specify the JRE system library as an execution environment for any Java-project, usually "JavaSE-1.6".  In the pom.xml I use the following lines:  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>2.3.2</version>  <configuration>  <source>1.6</source>  <target>1.6</target>  </configuration>  </plugin>  <plugin>  <artifactId>maven-eclipse-plugin</artifactId>  <version>2.8</version>  <configuration>  <projectNameTemplate>[groupId].[artifactId]</projectNameTemplate>  <downloadJavadocs>true</downloadJavadocs>  <wtpversion>2.0</wtpversion>  </configuration>  </plugin>  This seems to let Maven pick a JDK matching 1.6. It assigns its system libraries to the classpath of the project. However, it uses jdk1.6.20 directly rather than JavaSE-1.6.  Is there a way to make Maven use JavaSE-1.6 instead?  EDIT: Here's the batch file that I use to run Maven:  SET JAVA\_HOME=C:\Programme\java\jdk1.6.0\_20  SET MAVEN\_HOME=C:\Programme\apache-maven-2.2.1  SET MAVEN\_OPTS=-Xmx512m -XX:MaxPermSize=128m  SET PATH=%JAVA\_HOME%\bin;%MAVEN\_HOME%\bin;%PATH%  SET EXTRA\_MVN\_OPTS=-Dmaven.test.skip=true  CD /D "%~dp0"  cmd /c mvn %EXTRA\_MVN\_OPTS% eclipse:clean eclipse:eclipse  cmd /c mvn %EXTRA\_MVN\_OPTS% clean install  [java](http://stackoverflow.com/questions/tagged/java) [eclipse](http://stackoverflow.com/questions/tagged/eclipse) [maven-2](http://stackoverflow.com/questions/tagged/maven-2) [maven](http://stackoverflow.com/questions/tagged/maven)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/q/4759242)|[improve this question](http://stackoverflow.com/posts/4759242/edit) | [edited Jan 21 '11 at 14:31](http://stackoverflow.com/posts/4759242/revisions)  [[https://www.gravatar.com/avatar/65a1e72ed7a72ed7b0831fbcd1fee8b0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/342852/sean-patrick-floyd)](http://stackoverflow.com/users/342852/sean-patrick-floyd)  [Sean Patrick Floyd](http://stackoverflow.com/users/342852/sean-patrick-floyd) **109k**19173305 | asked Jan 21 '11 at 13:22  [[https://www.gravatar.com/avatar/5e6ce1def86e084cf06ac3a5a7e2d74f?s=32&d=identicon&r=PG](http://stackoverflow.com/users/483113/sjngm)](http://stackoverflow.com/users/483113/sjngm)  [sjngm](http://stackoverflow.com/users/483113/sjngm) **3,970**62964 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | I added the batch file for Maven and the pom.xml-lines for the maven-eclipse-plugin –  [sjngm](http://stackoverflow.com/users/483113/sjngm" \o "3970 reputation) [Jan 21 '11 at 13:34](http://stackoverflow.com/questions/4759242/eclipse-maven-force-execution-environment-javase-1-6-instead-of-fixed-jdk#comment5264724_4759242) |   add a comment |

## 3 Answers

[active](http://stackoverflow.com/questions/4759242/eclipse-maven-force-execution-environment-javase-1-6-instead-of-fixed-jdk?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/4759242/eclipse-maven-force-execution-environment-javase-1-6-instead-of-fixed-jdk?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/4759242/eclipse-maven-force-execution-environment-javase-1-6-instead-of-fixed-jdk?answertab=votes#tab-top)

|  |  |
| --- | --- |
| up vote10down voteaccepted | You can do it by setting a classpath container.  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-eclipse-plugin</artifactId>  <version>2.8</version>  <configuration>  <classpathContainers>  <classpathContainer>  org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.6  </classpathContainer>  </classpathContainers>  </configuration>  </plugin>  **Reference:**   * [eclipse:eclipse Mojo > classpathContainers](http://maven.apache.org/plugins/maven-eclipse-plugin/eclipse-mojo.html#classpathContainers) |

Java debug

<http://www.vogella.com/tutorials/EclipseDebugging/article.html>

java