Serge SIMON. 12/06/2012. Updated 31/07/2012.

Présentation propulsée sous Deck.js. Navigation par ←, →, m (menu), t (toc), g (goto), F11 (fullscreen).

Eclipse

« Astuces, utilisation avancée & optimisations »

Introduction

Cette présentation ...

... détaille quelques **généralités**, **modes de fonctionnement**, **raccourcis-clavier** (disséminés tout au long de la présentation), **astuces**, etc.

Cette présentation ...

... détaille quelques **généralités**, **modes de fonctionnement**, **raccourcis-clavier** (disséminés tout au long de la présentation), **astuces**, etc.

Chaque item a une note :

Cette présentation ...

... détaille quelques **généralités**, **modes de fonctionnement**, **raccourcis-clavier** (disséminés tout au long de la présentation), **astuces**, etc.

Chaque item a une note :

- A appliquer dans tous les cas - gains intéressants et/ou coût d'utilisation ou de mise en place faible
- Souvent intéressant à appliquer mais peut dépendre du contexte, etc.
- Peut dépendre des habitudes ou modes de fonctionnement de chacun

Cette présentation ...

... détaille quelques **généralités**, **modes de fonctionnement**, **raccourcis-clavier** (disséminés tout au long de la présentation), **astuces**, etc.

Chaque item a une note :

- A appliquer dans tous les cas - gains intéressants et/ou coût d'utilisation ou de mise en place faible
- Souvent intéressant à appliquer mais peut dépendre du contexte, etc.
- Peut dépendre des habitudes ou modes de fonctionnement de chacun

La plupart des développeurs passent beaucoup de temps chaque jour sur Eclipse ... à ce titre, il (me) sembe important de bien connaître l'outil, sa paramétrie, les raccourcis, pour en tirer le meilleur parti (et surtout, pour s'éviter un certain nombre de manipulations fastidieuses).

Versions d'Eclipse disponibles

Versions d'Eclipse disponibles

- **Eclipse CLASSIC** : pour le développement quel que soit le langage
- **Eclipse Java** : version de base pour développement Java
- **Eclipse Java/J2EE** : version avancée pour développement Java

Versions d'Eclipse disponibles

- **Eclipse CLASSIC** : pour le développement quel que soit le langage
- **Eclipse Java** : version de base pour développement Java
- **Eclipse Java/J2EE** : version avancée pour développement Java

Quelle version choisir ?

Versions d'Eclipse disponibles

- **Eclipse CLASSIC** : pour le développement quel que soit le langage
- **Eclipse Java** : version de base pour développement Java
- **Eclipse Java/J2EE** : version avancée pour développement Java

Quelle version choisir ?

Dans la majorité des cas, **la version "Java" est amplement suffisante**. Remarque : elle ne contient cependant pas WTP (Web Tools Platform, pour le lancement des serveurs web), mais avantageusement remplacé par un usage local de Jetty.

Versions d'Eclipse disponibles (tableau)

Release	Date	Version	Notes
4.x	Kepler	June 2013 (prévisionnel)	e4 Project (http://www.eclipse.org/e4/)
3.8	Juno	June 2012	
3.7	Indigo	June 2011	Version utilisee dans l'equipe.
3.6	Helios	23 June 2010	
3.5	Galileo	24 June 2009	
3.4	Ganymede	25 June 2008	
3.3	Europa	29 June 2007	
3.2	Callisto	30 June 2006	
3.1	Eclipse 3.1	28 June 2005	
3.0	Eclipse 3.0	28 June 2004	

Versions d'Eclipse disponibles (tableau)

Release	Date	Version	Notes
4.x	Kepler	June 2013 (prévisionnel)	e4 Project (http://www.eclipse.org/e4/)
3.8	Juno	June 2012	
3.7	Indigo	June 2011	Version utilisee dans l'equipe.
3.6	Helios	23 June 2010	
3.5	Galileo	24 June 2009	
3.4	Ganymede	25 June 2008	
3.3	Europa	29 June 2007	
3.2	Callisto	30 June 2006	
3.1	Eclipse 3.1	28 June 2005	
3.0	Eclipse 3.0	28 June 2004	

Remarque : Eclipse 4 (Kepler) est une réecriture complète "from scratch" du projet.

Versions d'Eclipse disponibles (tableau)

Release	Date	Version	Notes
4.x	Kepler	June 2013 (prévisionnel)	e4 Project (http://www.eclipse.org/e4/)
3.8	Juno	June 2012	
3.7	Indigo	June 2011	Version utilisee dans l'equipe.
3.6	Helios	23 June 2010	
3.5	Galileo	24 June 2009	
3.4	Ganymede	25 June 2008	
3.3	Europa	29 June 2007	
3.2	Callisto	30 June 2006	
3.1	Eclipse 3.1	28 June 2005	
3.0	Eclipse 3.0	28 June 2004	

Remarque : Eclipse 4 (Kepler) est une réecriture complète "from scratch" du projet.

INFO

A savoir : Eclipse est bâti sur OSGi (pour les plugins notamment), via Equinox.
Propose la plateform RCP (**Rich Client Platform**) permettant de construire des clients lours avec la même architecture qu'Eclipse (IHM en SWT, notions de perspectives/vues/plugins, etc.).

Concurrents

Concurrents

Premier concurrent de poids, **Netbeans** (http://fr.netbeans.org/) qui est, pour rappel, l'outillage de développement **officiel** dans le monde Java puisque développé par Oracle (anciemment Sun). Contrairement à Eclipse, Netbeans **respecte la philosophie Java** avec une IHM construite en SWING sans aucune dépendance système (alors qu'Eclipse, via SWT, a des dépendances systèmes).

Concurrents

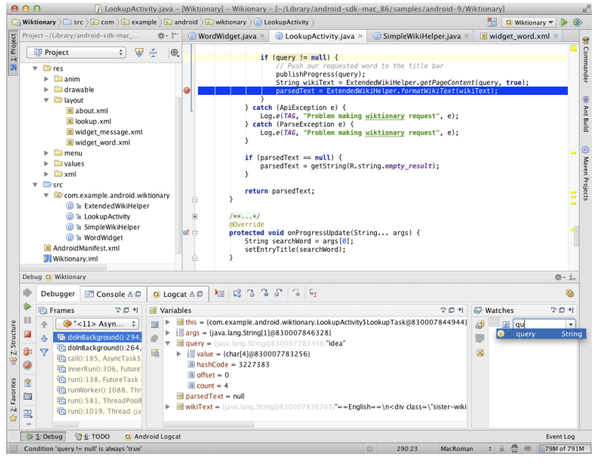
Premier concurrent de poids, **Netbeans** (http://fr.netbeans.org/) qui est, pour rappel, l'outillage de développement **officiel** dans le monde Java puisque développé par Oracle (anciemment Sun). Contrairement à Eclipse, Netbeans **respecte la philosophie Java** avec une IHM construite en SWING sans aucune dépendance système (alors qu'Eclipse, via SWT, a des dépendances systèmes).

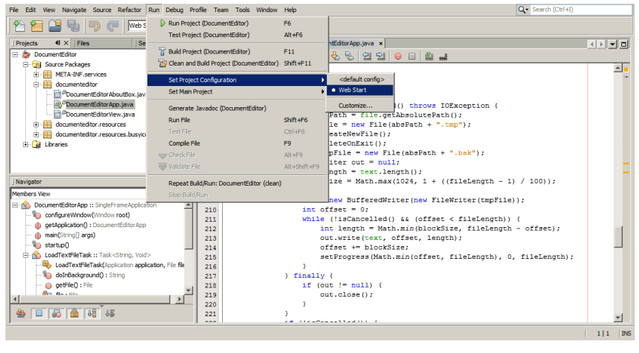
L'outsider qui a le vent en poupe est **IntelliJ IDEA** (http://www.jetbrains.com/idea/), "The Most Intelligent Java IDE" (sic). Relativement similaire à Eclipse, il est payant à la base même si une Community Edition existe désormais.

Concurrents

Premier concurrent de poids, **Netbeans** (http://fr.netbeans.org/) qui est, pour rappel, l'outillage de développement **officiel** dans le monde Java puisque développé par Oracle (anciemment Sun). Contrairement à Eclipse, Netbeans **respecte la philosophie Java** avec une IHM construite en SWING sans aucune dépendance système (alors qu'Eclipse, via SWT, a des dépendances systèmes).

L'outsider qui a le vent en poupe est **IntelliJ IDEA** (http://www.jetbrains.com/idea/), "The Most Intelligent Java IDE" (sic). Relativement similaire à Eclipse, il est payant à la base même si une Community Edition existe désormais.





Sommaire

Chapitre 1.
Optimisations

Taille mémoire

HOW

Essayez d'avoir un maximum de mémoire.
3 GO = grand minimum pour les gros projets.
4 GO sous Seven 64bits = recommandés
Prévoir **8 GO** si nécessité d'intervenir sur plusieurs projets) (voir plus loin sur la répartition des projets en workspaces)

INFO

Il peut être intéressant de **démarrer certains éléments hors Eclipse** (ex. HSQLDB) pour les cas vraiment limites (évite de consommer de la mémoire au sein d'Eclipse et de devoir être redémarré en cas de plantage d'Eclipse par ex.)

Optimisation mémoire système

HOW Activer le "new garbage collector" (disponible à partir du JDK 1.6.0.11) : dans eclipse.ini

```
-XX:+UseConcMarkSweepGC
-XX:+UnlockExperimentalVMOptions
-XX:+UseG1GC

```

HOW Une paramétrie d'Eclipse standard devrait proposer niveau RAM (paramétrie à faire au sein du eclipse.ini ou en ligne de commande (raccourci de lancement))

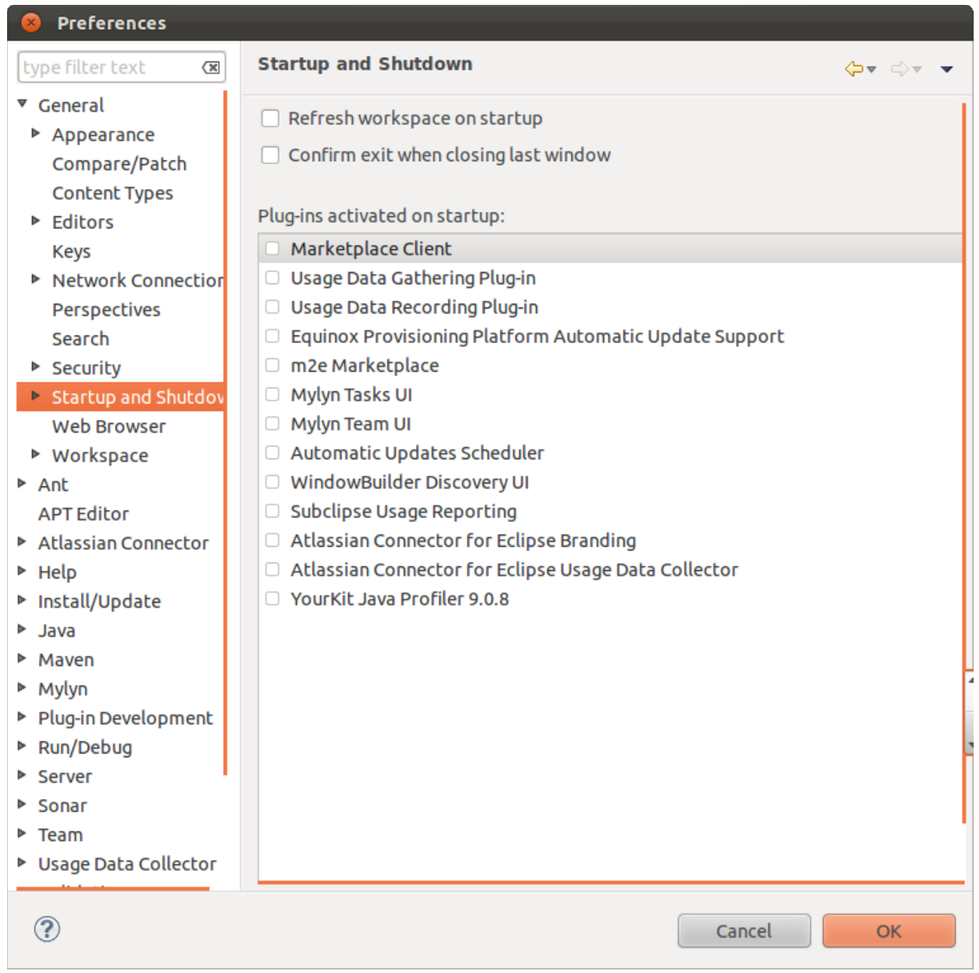
```
-Dosgi.requiredJavaVersion=1.6
-XX:MaxPermSize=384m
-Xms512m
-Xmx768m

```

INFO Attention, ces paramétries sont à valider / tester et peuvent dépendre de vos contextes (taille des projets, utilisation d'Eclipse, etc. : il peut être intéressant de tailler la mémoire différemment, etc.)

Optimisation temps de démarrage

HOW General > Startup and Shutdown > tout décocher

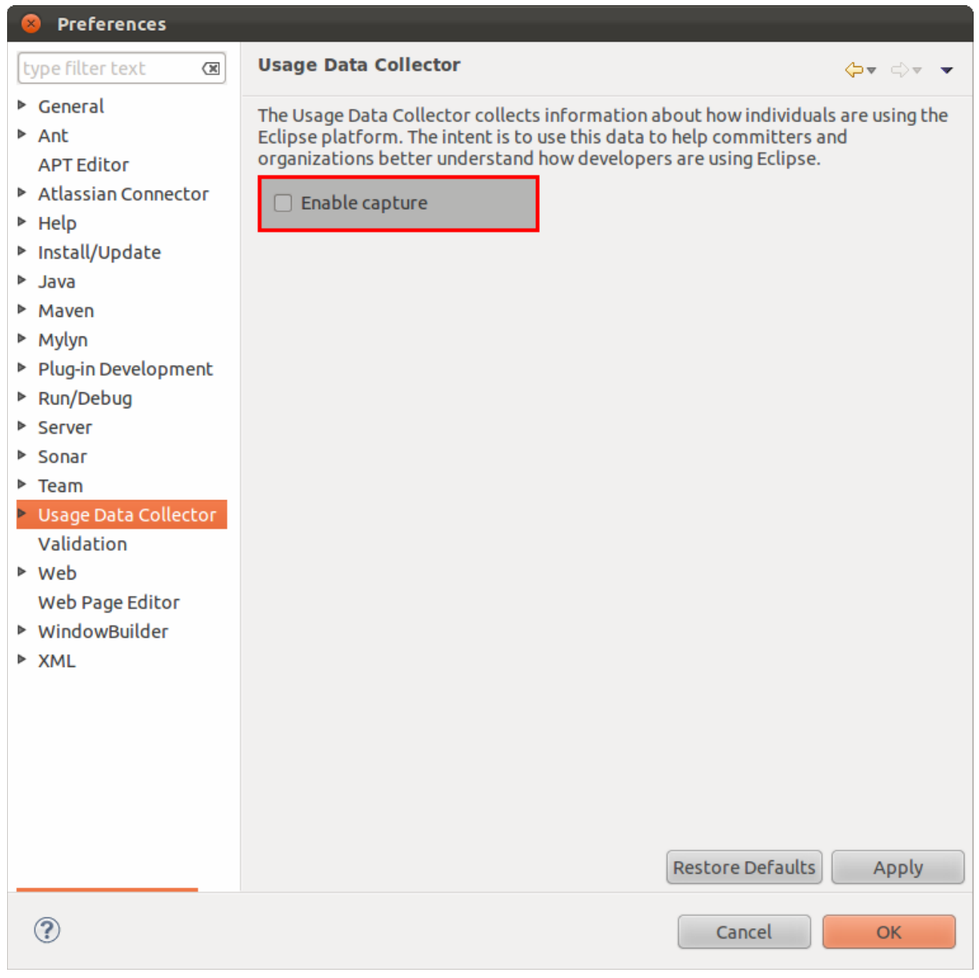
WHY Réduction du temps d'affichage. Certaines fonctions sont inutiles sur les projets usuels (tous les mécanismes de collectes d'usages, tout ce qui tourne autour de JAX-WS, etc.), d'autres peuvent être n'activées qu'au besoin (plugin de profiling, etc.).



Optimisation ressources

HOW Usage Data Collector > désactiver

WHY Monitore l'usage d'Eclipse et remonte périodiquement les statistiques correspondantes sur le site des développeurs d'Eclipse. Un mécanisme similaire existe sur le connecteur JIRA pour Mylyn.



Optimisation visuelle

HOW	Window > Preferences > General > Appearance > [OFF] Enable animations
WHY	Réduction du temps d'affichage, notamment lors des aggrandissements / réductions de vues.

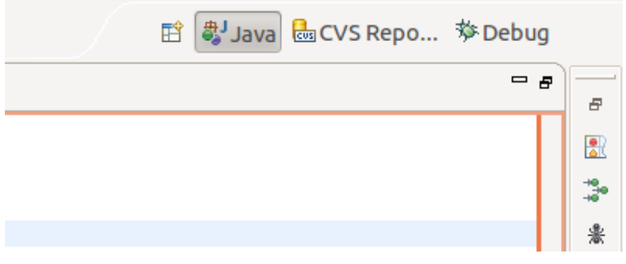
[raccourci clavier] Gérer la fenêtre d'édition courante

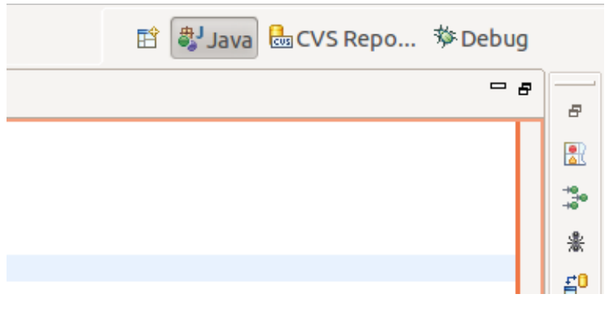
HOW `CTRL - M`

WHY Aggrandir / réduire la vue Eclipse en cours (ex. zone d'édition Java > passage en plein écran / retour à la normale).

Optimisation visuelle

HOW	Window > Preferences > General > Appearance > [ON] Show traditionnal style tabs
WHY	Gagner (à peine) en surface d'affichage à certains endroits du workspace (pour les petits écrans type PC portable)
INFO	Remarque : le look global change à partir d'Eclipse 3.8 (Juno)

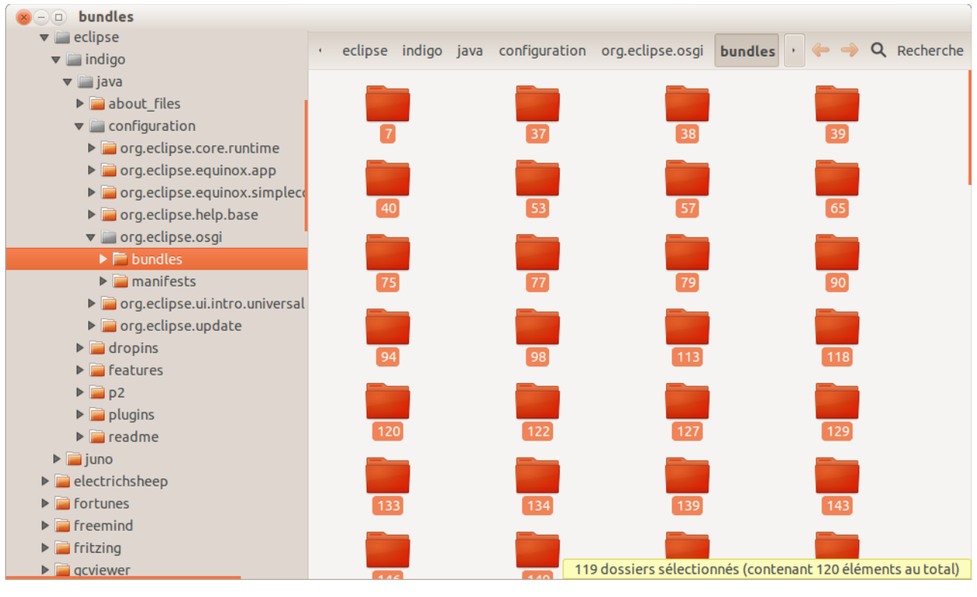




Optimisation du temps de démarrage

HOW Purger le sous-répertoire **configuration/org.eclipse.osgi/bundles** (dans le répertoire d'installation d'Eclipse)

WHY Tous les plugins disponibles créent un cache local qui à terme pénalise le démarrage d'Eclipse.



Travailler impérativement avec un JDK récent (dernières version 1.6, voire 1.7) ★★★

HOW

Installer manuellement le JDK à jour (depuis le site d'Oracle.
Paramétrer Eclipse pour démarrer avec ce JDK, éventuellement paramétrer Eclipse pour pointer dessus (Preferences > Java > Installed JRE)

WHY

Les nouvelles versions du JRE/JDK apportent régulièrement des gains de performances, notamment entre les versions majeures (1.5 > 1.6 > 1.7).
Celà n'empêche absolement pas de garder un JDK d'une autre version pour assurer la compilation (cf. dans les Préférences "Installed JRE") + niveau de compatibilité du projet dans ses préférences (1.3 / 1.4 / 1.5 / 1.6, indépendamment du JDK utilisé pour la compilation).

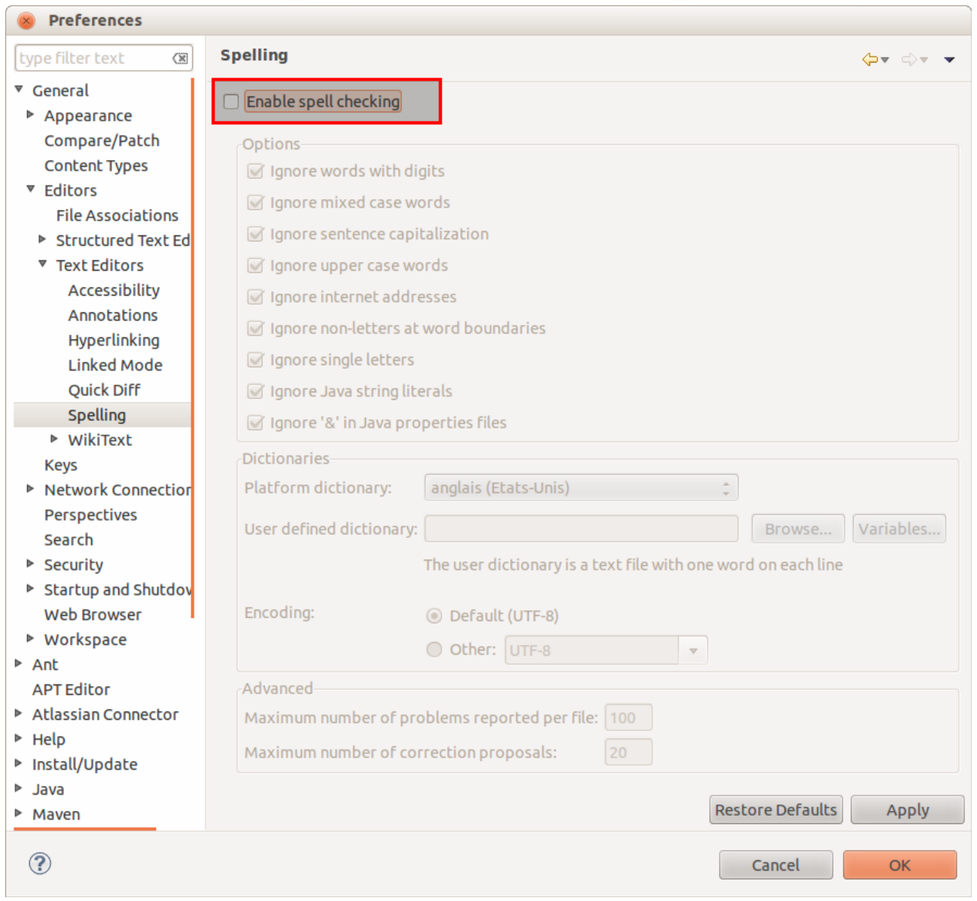
INFO

Rappel : lors de la mise à jour d'un projet Maven, le niveau de JDK utilisé sous Eclipse est défini par le plugin "**maven-compiler-plugin**"

Désactiver le correcteur orthographique

HOW Window > Preferences > General > Editors > Text Editors > Spelling > [OFF] Enable spell checker

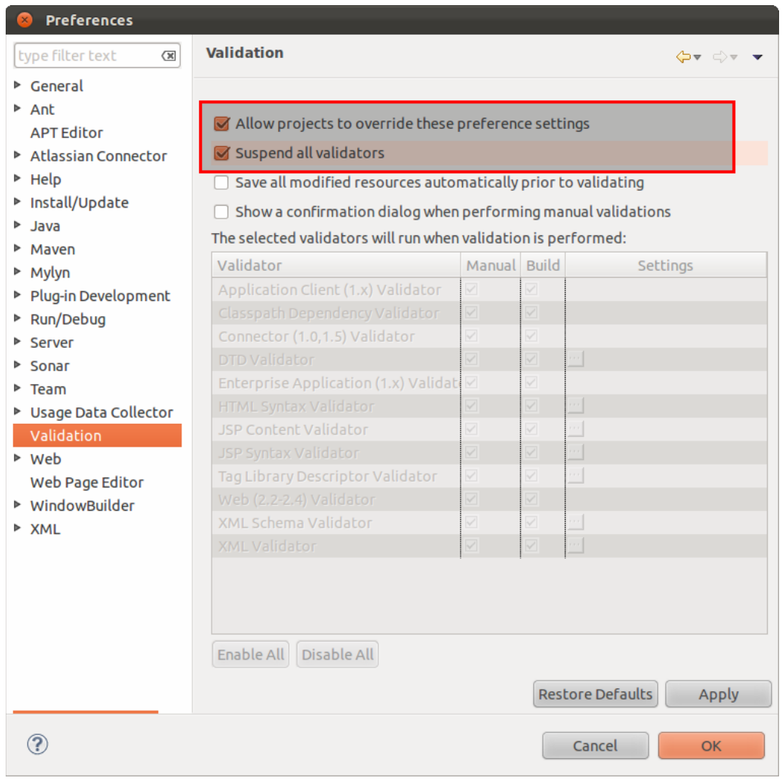
WHY Améliore les performances d'Eclipse.



Désactiver les validators

HOW De manière globale sous Préférences > Validators, ou dans la configuration de chaque projet.

WHY Améliore un peu la réactivité d'Eclipse (moins de fichiers à analyser) et évite certaines erreurs récurrentes (sur les POM ou certains HTML, etc.). A réactiver ponctuellement au besoin.



Désactiver les plugins type Checkstyle ou FindBugs sous Eclipse

HOW	Désinstallation du plugin (ou ne pas l'installer) ou le désactiver dans les Préférences
WHY	Ces plugins sont souvent très pénalisants en terme de performances (analyses continues à la volée). A n'utiliser que ponctuellement en les réactivant sous Eclipse. La bonne pratique consiste à analyser ces informations via les règles idoines Sonar.

Chapitre 2. Configuration

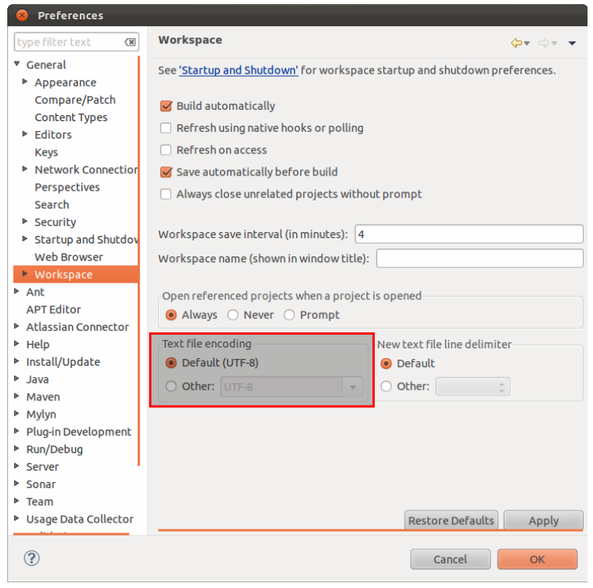
Configurer l'encoding du workspace en UTF-8

HOW General > Workspace > UTF-8

WHY Important, pour éviter d'avoir des problèmes d'encodage par ex. dans les commentaires sous CVS !

INFO Si vous voyez où que ce soit des caractères gerne @~é, c'est que vous avez un problème d'encodage sur votre poste (ou éventuellement que qq'un ayant un problème d'encodage a commité). A corriger / faire corriger rapidement.

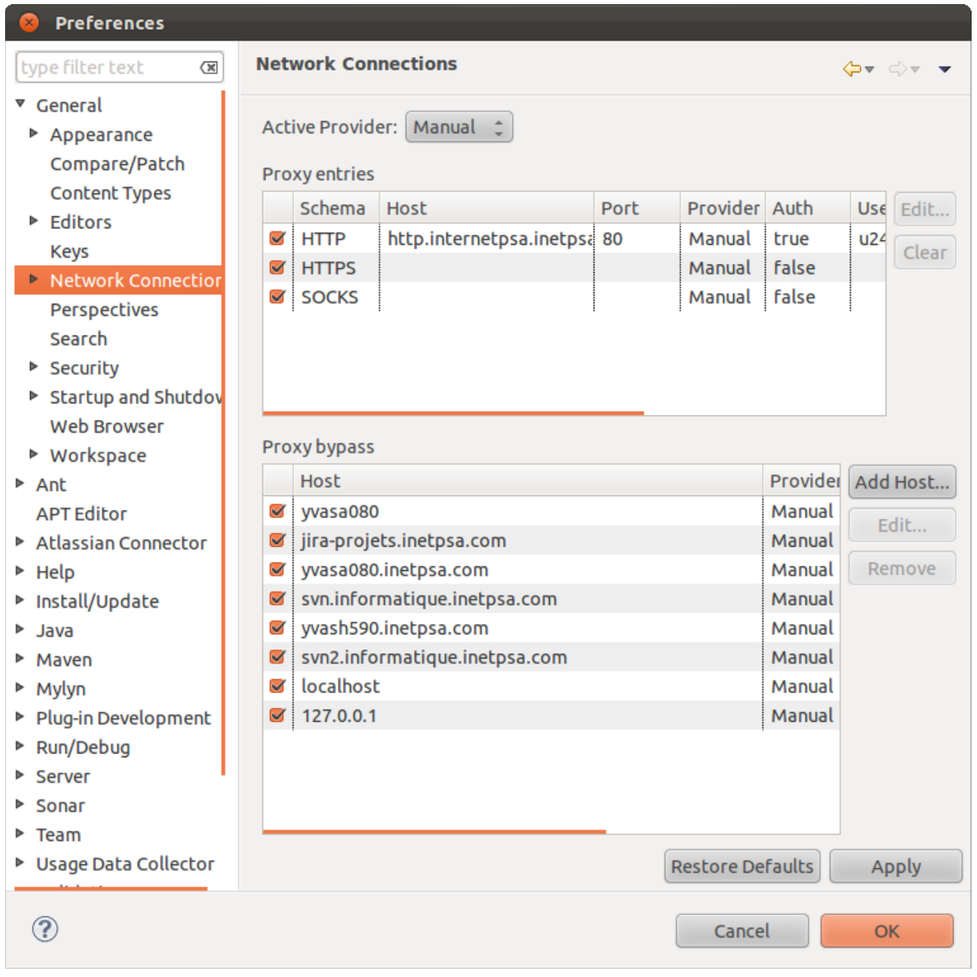
Sous Linux, l'encoding est par défaut en UTF-8, mais à positionner correctement sous Windows.



Configurer correctement le réseau

HOW General > Network Connections > configurer le proxy + proxy bypass (jira-projets.domaine.tld, server.domaine.tld, svn.domaine.tld, localhost, 127.0.0.1, etc.) (attention, case sensitive !). Au final le plus simple est de mettre "*.domaine.tld".

WHY Permet d'accéder tant aux ressources internes (SVN, CVS, Sonar, etc.) qu'externes (mises à jour, marketplace, ...)

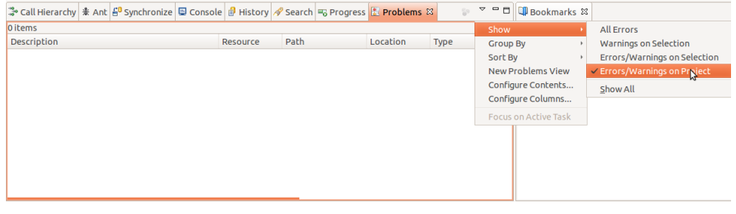


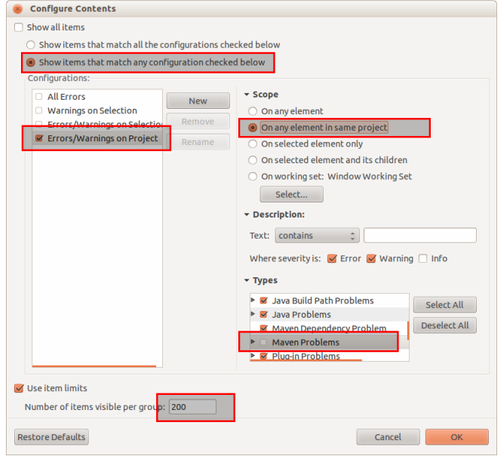
Configurer correctement la vue "Problems"

HOW Bien spécifier une option permettant de ne voir les erreurs que sur le projet courant

WHY Pour ne pas être noyé par les erreurs

INFO Il est également possible de retirer certains types d'erreurs (les erreurs sur DTD ou au sein des pom.xml Maven par ex. - certains sont difficilement corrigeables aujourd'hui)



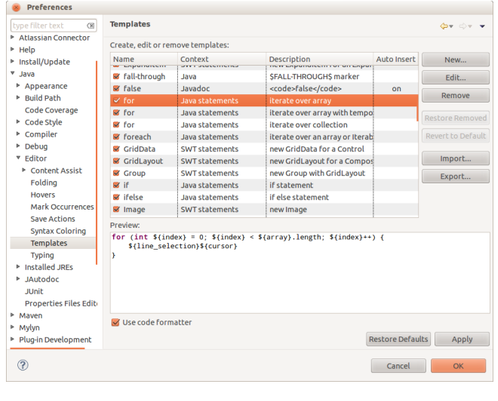


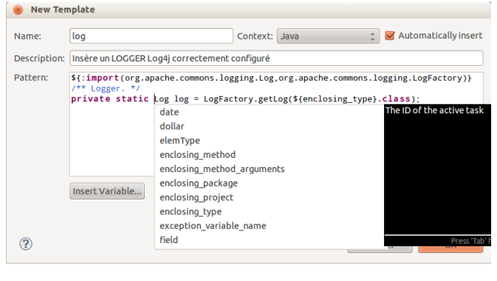
Automatiser l'insertion d'un bloc de texte répétitif

HOW En créant ses propres templates dans Preferences > Java > Editor > Templates, [New] ou [Import] pour la partie Java, ou, par ex., Preferences > XML > XML Files > Editor> Templates

WHY Permet par ex. d'insérer en Java très facilement un LOGGER, y compris avec les imports côté code ; un pattern HTML dans un template ; ou un pattern XML dans une fiche de tests, etc.

INFO De nombreux templates existants sont déjà disponibles, souvent méconnus (et tous accessibles via `CTRL` - `ESPACE`), par ex. côté Java pour la création des boucles **for** ou la mise en place d'un **try ... catch**





Automatiser l'insertion de code répétitif (templates Java)

INFO

Ex. de template pour un LOGGER SLF4J (template **log**) :

```
${:import(org.slf4j.Logger,org.slf4j.LoggerFactory)}
/** Logger. */
private static final Logger LOGGER = LoggerFactory.getLogger(${enclosing_type}.class);
```

INFO

Ex. de template pour un LOGGER LOG4J (template **log**) :

```
${:import(org.apache.commons.logging.Log,org.apache.commons.logging.LogFactory)}
/** Logger. */
private static Log log = LogFactory.getLog(${enclosing_type}.class);
```

INFO

Ex. de template pour créer une constante (le type n'est à renseigner qu'une seule fois) (template **const**) :

```
private static final ${type} ${name} = new ${type}() ${cursor};
```

INFO

Ex. de template pour créer une liste (template **list**) :

```
${:import(java.util.Collection, java.util.ArrayList)}
Collection<${argType}> ${newName} = new ArrayList<${argType}>();
```

INFO

Ex. de template pour créer une map (template **map**)

```
${:import(java.util.Map,java.util.HashMap)}
Map<${argType},${argType2}> ${newName} = new HashMap<${argType},${argType2}>();
```

Automatiser l'insertion de code répétitif (template XML)

INFO

Ex. de base de template pour une fiche de test XML (au format LTP/Tester) (template **test**):

```
<test id="${cursor};">
  <description></description>
  <message_pour_appli id="" description="" emetteur="" requete="" contenu="">
    <assertions>
    </assertions>
  </message_pour_appli>
</test>
```

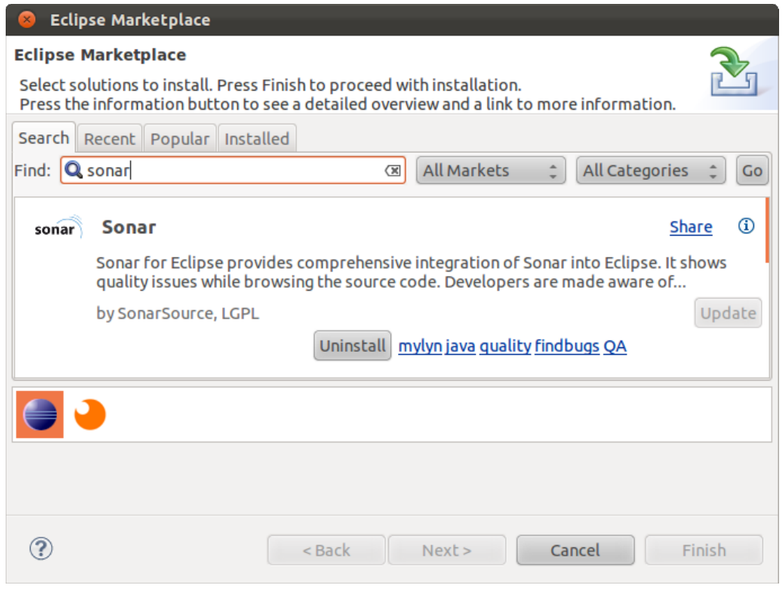
Chapitre 3. Plugins

Installation des plugins

HOW Toujours passer par le Marketplace. Eviter les installations manuelles par dézippage (de plus en plus rares / inutiles).

Il faut bien sûr avoir son proxy correctement configuré (cf. point plus haut).

WHY N/A



Plugins pertinents

Plugins présentés :

- **APTEditor**
- **JAutoDoc**
- **Grep Console**
- **SONAR**
- **EclEmma**
- **Mylyn**
- **JInto**
- **QuickREx**

Autres pistes pour les plugins

Plugins NON présentés ici mais pouvant avoir un intérêt :

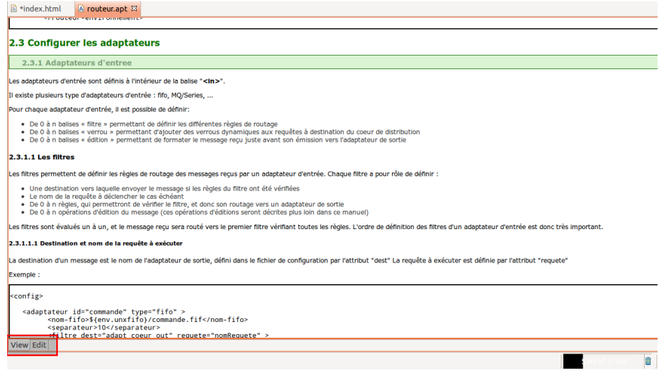
- JADClipse (obsolète) > **JD-Eclipse** (http://java.decompiler.free.fr/?q=jdeclipse)
- **Lockness** (analyse de heapdump / threads) (http://marketplace.eclipse.org/content/lockness-thread-dump-analyser)
- **Eclipse Color Theme** (http://marketplace.eclipse.org/content/eclipse-color-theme), permet d'avoir par ex. une couleur de fond différente selon les workspaces
- Plugin accès base de données (personnellement j'utilise toujours des outils externes, SQLWorkench/J pour sa rapidité, ou SQLDeveloper pour les fonctions Oracle)
- AnyEdit
- nTail http://www.certiv.net/
- classpathChecker http://classpathchecker.free.fr/
- simpleSearch http://www.emilmont.net/eclipse/simplesearch
- sqlexplorer.org http://eclipsesql.sourceforge.net/
- regexUtil http://regex-util.sourceforge.net/
- ehep http://ehep.sourceforge.net/update
- EditBox http://editbox.sourceforge.net/updates
- eclEmma http://update.eclemma.org
- jOra http://toaddownload.quest.com/toadextensions/
- Jsch http://eclipse.jcraft.com
- atlassianPlugin http://update.atlassian.com/atlassian-eclipse-plugin/e3.7
- frills http://eclipsefrills.sourceforge.net/
- m2e http://download.eclipse.org/technology/m2e/

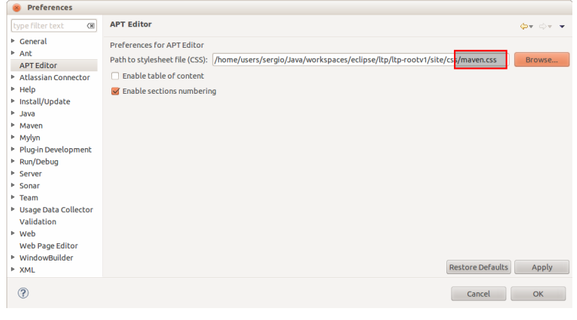
APTEditor

HOW Récupération du plugin depuis le web. Installation du .css Maven dans la config du produit.

HOW Une alternative est le plugin YEP Apt Editor http://marketplace.eclipse.org/content/yep-apt-editor, qui pose à priori moins de problèmes sous Windows (pas de ralentissements en prévisu).

WHY Permet de prévisualiser le contenu des fichiers Maven apt

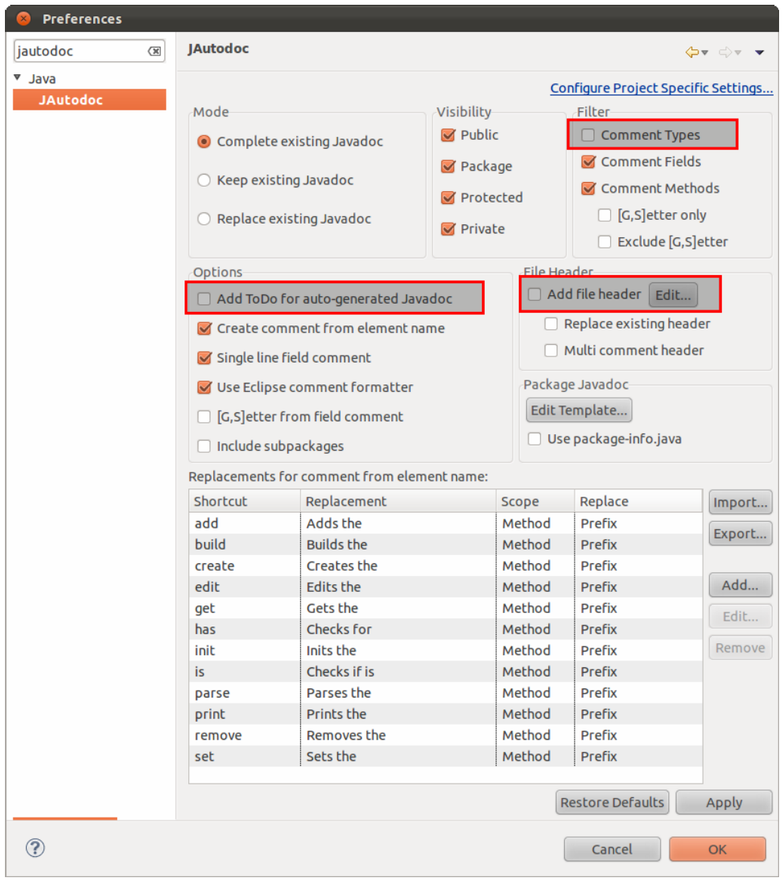




JAutoDoc

HOW Installation depuis le Marketplace.

WHY Permet de compléter automatiquement la Javadoc, soit à la souris, soit, sur une méthode ou une classe, avec le raccourci `CTRL - ALT - J`. Permet de renseigner les constructeurs, l'entête de classe, les méthodes, les propriétés, les getters/setters (en mode création ou mise à jour). Il reste ensuite juste en général à raffiner les commentaires des méthodes importantes / sensibles / complexes.



[raccourci clavier] Ajout de la javadoc

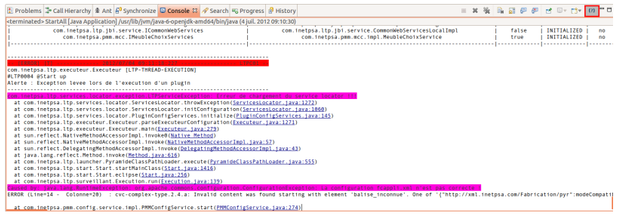
HOW `CTRL - ALT - J`

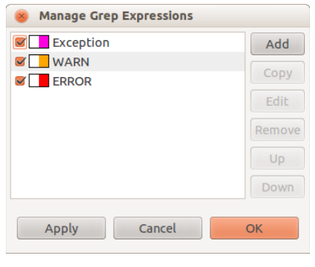
WHY Ajouter la Javadoc sur l'élément courant (méthode, classe, ...) (voir la partie plugin JAutoDoc).

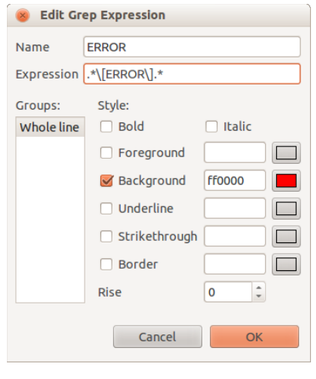
Grep Console

HOW Installation depuis le Marketplace.

WHY Permet à partir d'expressions régulières de colorer les logs. Indispensable pour voir passer les erreurs ou les warnings. Pas d'impact notable sur les performances de l'affichage en console. Ex. pour les erreurs, paramétrer le fond en rouge et l'expression ".*\[ERROR\].*".



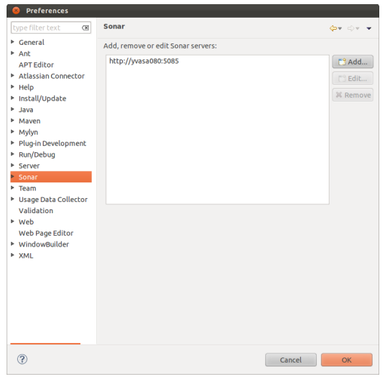


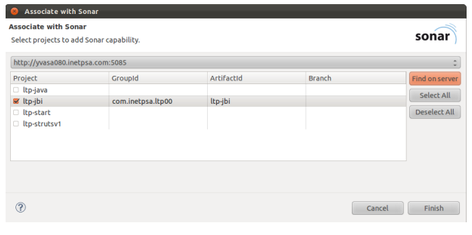


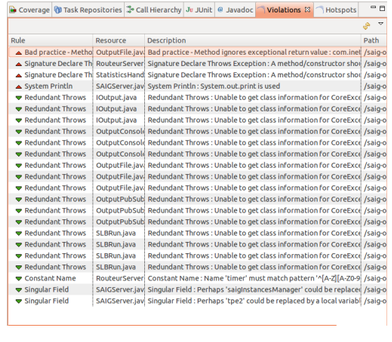
SONAR

HOW Installation depuis le Marketplace.

WHY Cf. la présentation SONAR, permet de consulter une partie de l'analyse SONAR directement depuis Eclipse (violations notamment).



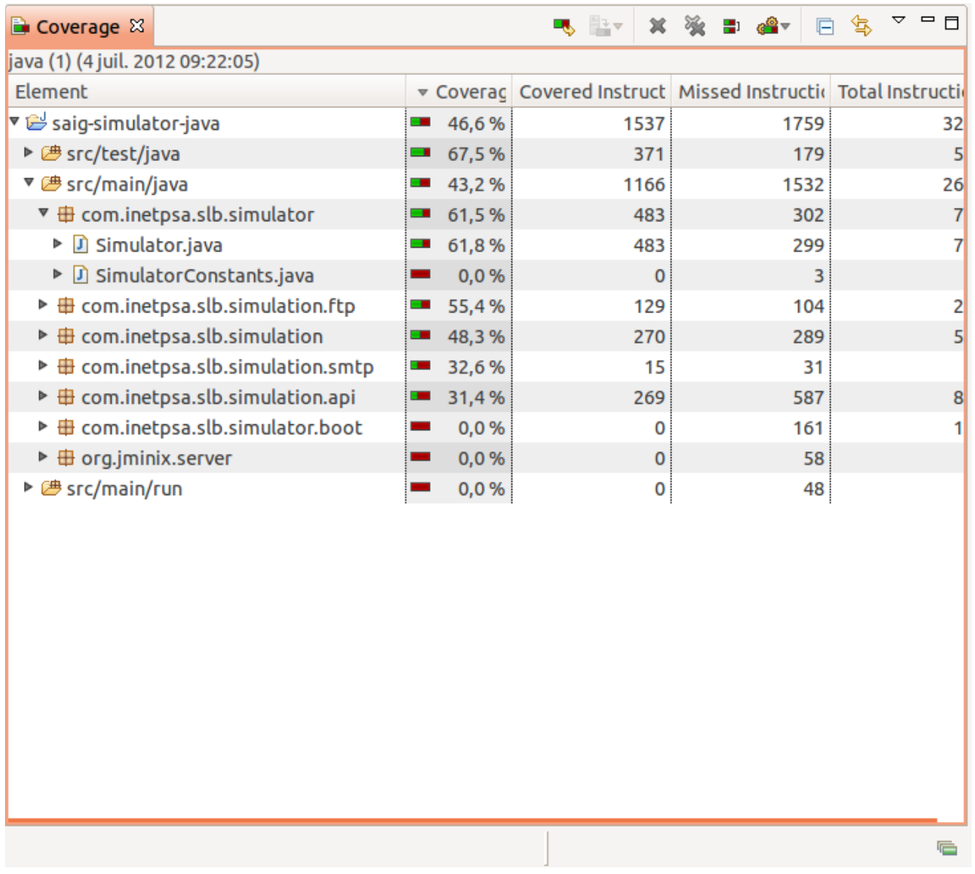




EclEmma

HOW Installation depuis le Marketplace. Lancement par bouton droit sur le répertoire junit/ et "Coverage As" au lieu de "Run As"

WHY Permet de visualiser la couverture de test des tests junit directement sous Eclipse. Basé sur Jacoco pour le moteur d'analyse.

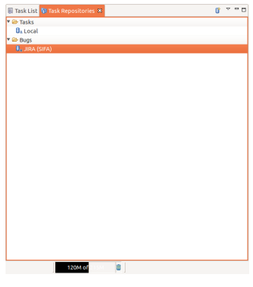
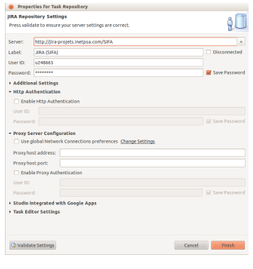
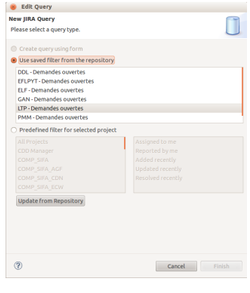
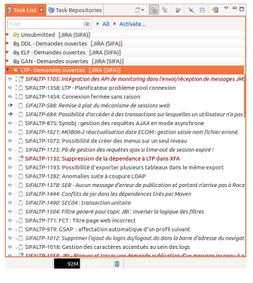
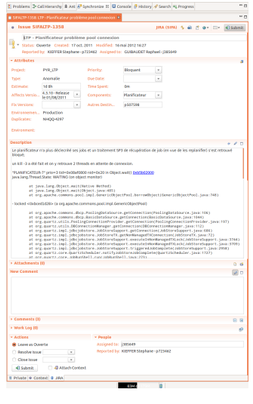


Element	Coverag	Covered Instruct	Missed Instructi	Total Instructi
saig-simulator-java	46,6 %	1537	1759	32
src/test/java	67,5 %	371	179	5
src/main/java	43,2 %	1166	1532	26
com.inetpsa.slb.simulator	61,5 %	483	302	7
Simulator.java	61,8 %	483	299	7
SimulatorConstants.java	0,0 %	0	3	
com.inetpsa.slb.simulation.ftp	55,4 %	129	104	2
com.inetpsa.slb.simulation	48,3 %	270	289	5
com.inetpsa.slb.simulation.smtp	32,6 %	15	31	
com.inetpsa.slb.simulation.api	31,4 %	269	587	8
com.inetpsa.slb.simulator.boot	0,0 %	0	161	1
org.jminix.server	0,0 %	0	58	
src/main/run	0,0 %	0	48	

Mylyn

HOW Normalement installé par défaut. Conseil : toujours utiliser des filtres sauvegardés sous JIRA plutôt que des filtres locaux (pour éviter de les paramétrer sur chaque poste)

WHY Permet de : naviguer dans les JIRA, créer de nouveaux JIRA, indiquer qu'on travaille sur un JIRA donné pour sauvegarde du contexte (historique des modifications apportées dans le cadre de ce JIRA), etc.

JInto

HOW Depuis le Marketplace. Utilisation par click droit sur un .properties et ouverture avec l'éditeur JInto

WHY Permet d'éditer en colonne un fichier properties dans chaque langue existante

SearchMessages

SearchMessages ResourceBundle

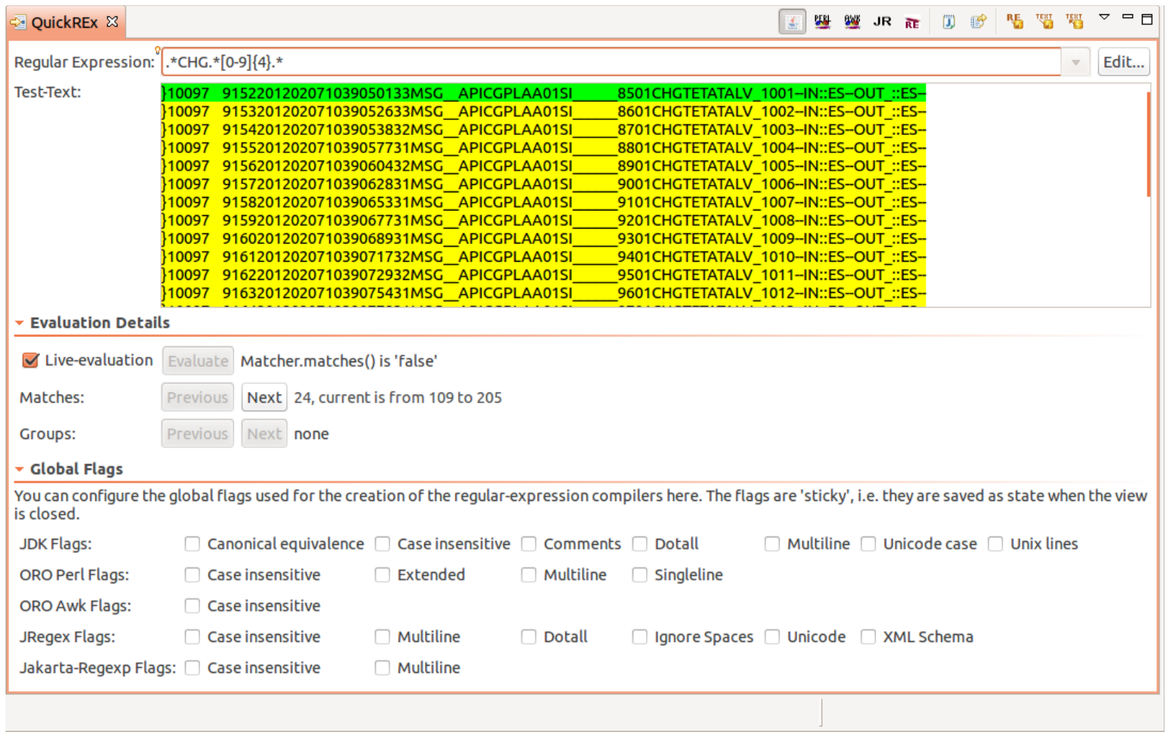
Keys	-	German
CopyToClipboardAction_error_message	There was a problem when accessing the s...	Beim Zugreifen auf die Zwischenablage des ...
CopyToClipboardAction_error_title	Problem Copying to Clipboard	Fehler beim Kopieren in die Zwischenablage
CopyToClipboardAction_label	Copy	Kopieren
CopyToClipboardAction_tooltip	Copy to Clipboard	In die Zwischenablage kopieren
ExceptionDialog_seeErrorLogMessage	See error log for details	Details finden Sie im Fehlerprotokoll.
FileLabelProvider_count_format	({0} matches)	({0} Übereinstimmungen)
FileLabelProvider_dashSeparated	{0} - {1}	{0} - {1}
FileLabelProvider_line_number	{0}:	{0}:
FileLabelProvider_removed_resource_label	<removed resource>	<entfernte Ressource>
FileSearchPage_error_marker	Could not create marker	Die Markierung konnte nicht erstellt werden.
FileSearchPage_limited_format_files	{0} (showing {1} of {2} files)	{0} ({1} von {2} Dateien angezeigt)
FileSearchPage_limited_format_matches	{0} (showing {1} of {2} matches)	{0} (zeigt {1} von {2} Übereinstimmungen)
FileSearchPage_open_file_dialog_title	Open File	Datei öffnen
FileSearchPage_open_file_failed	Opening the file failed.	Das Öffnen der Datei ist fehlgeschlagen.
FileSearchPage_sort_by_label	Sort By	Sortieren nach
FileSearchPage_sort_name_label	Name	Name
FileSearchPage_sort_path_label	Path	Pfad
FileSearchQuery_label	File Search	Dateisuche
FileSearchQuery_pluralPattern	"{0}" - {1} matches in {2}	"{0}" - {1} Übereinstimmungen in {2}
FileSearchQuery_pluralPattern_fileNameSea...	{1} file names matching "{0}" in {2}	{1} Dateinamen stimmen mit "{0}" in {2} üb...
FileSearchQuery_pluralPatternWithFileExt	"{0}" - {1} matches in {2} ({3})	"{0}" - {1} Übereinstimmungen in {2} ({3})
FileSearchQuery_singularLabel	"{0}" - 1 match in {1}	"{0}" - 1 Übereinstimmung in {1}
FileSearchQuery_singularLabel_fileNameSea...	1 file name matching "{0}" in {1}	1 Dateiname stimmt mit "{0}" in {1} überein
FileSearchQuery_singularPatternWithFileExt	"{0}" - 1 match in {1} ({2})	"{0}" - 1 Übereinstimmung in {1} ({2})
FileTextSearchScope_scope_double	"{0}", "{1}"	"{0}", "{1}"

Quick REx

HOW Installation manuelle depuis http://sourceforge.net/projects/quickrex/. Usage par la vue Quick RegExp dédiée.

WHY Permet de tester des expressions régulières Java sans devoir faire un main. Utile seulement si beaucoup de regexp à écrire.

INFO Disponible aussi en standalone. Homepage : http://www.bastian-bergerhoff.com/eclipse/features/web/QuickREx/toc.html.

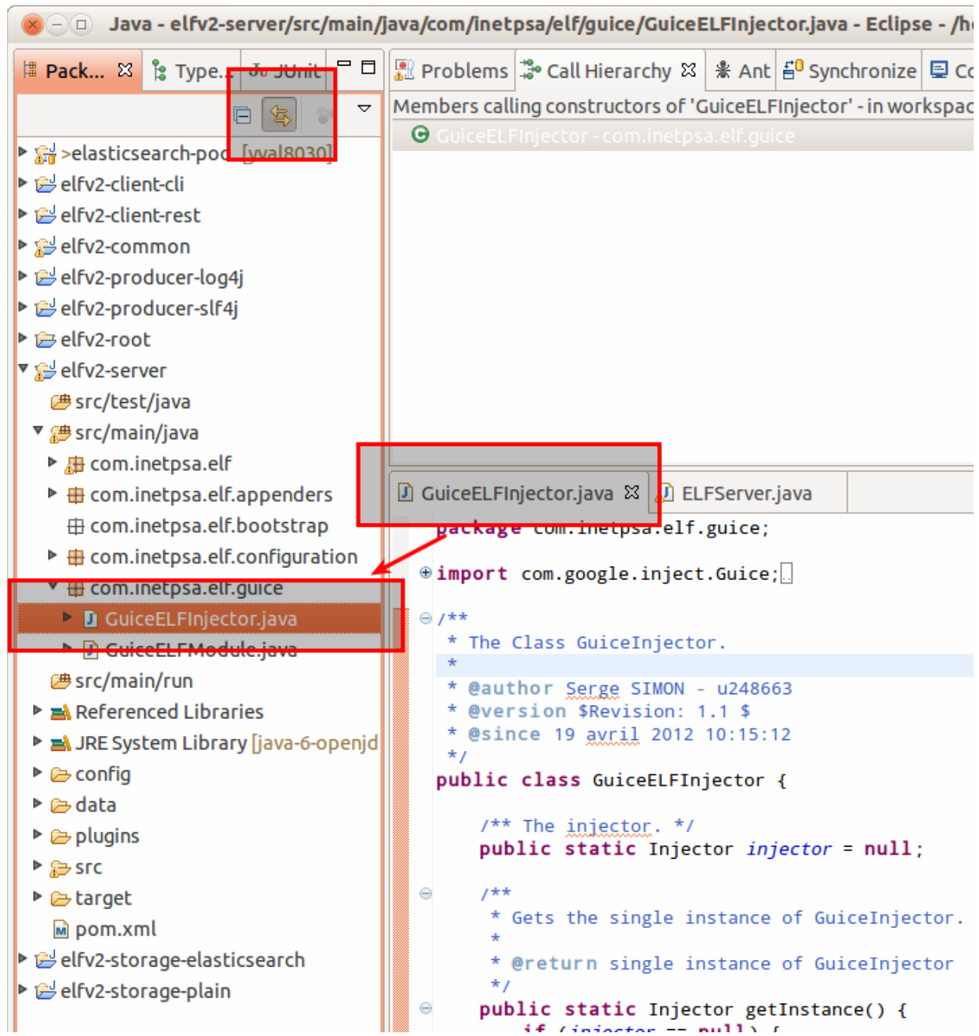


Chapitre 4. Utilisation d'Eclipse

Lier l'explorateur de fichiers avec le fichier courant

HOW Utiliser le bouton prévu à cet effet.

WHY Permet de retrouver rapidement un fichier pour y faire une action contextuelle. Peut être pénible parfois si toujours activé, donc à activer/désactiver au besoin.



[raccourci clavier] Ouvrir l'objet courant dans une vue

HOW `ALT - SHIFT - W`

WHY Sur un élement quelconque (une classe par ex.), ouvre un popup contextuel permettant de choisir dans quelle vue on veut ouvrir cet élément (par ex., dans la vue "History" ou dans le "Package Explorer")

[raccourci clavier] Quick-edit

HOW `CTRL - Q`

WHY Ramène au dernier point d'édition sous Eclipse !

[raccourci clavier] Navigation au clavier

HOW

`CTRL` - `D`

`ALT` - `FLECHE_HAUT` ou `ALT` - `FLECHE_BAS`

`CTRL` - `SHIFT` - `ENTER`

WHY

`CTRL` - `D` < permet d'effacer la ligne courante, sans toucher au presse-papier.

`ALT` - `FLECHE` (HAUT ou BAS) permet de déplacer la ligne ou le bloc sélectionné vers le haut ou le bas (sans nécessiter là aussi de passer par le presse-papier).

`CTRL` - `SHIFT` - `ENTER` ajoute une ligne vide avant la ligne courante et y positionne le curseur.

Vue CVS/SVN en console

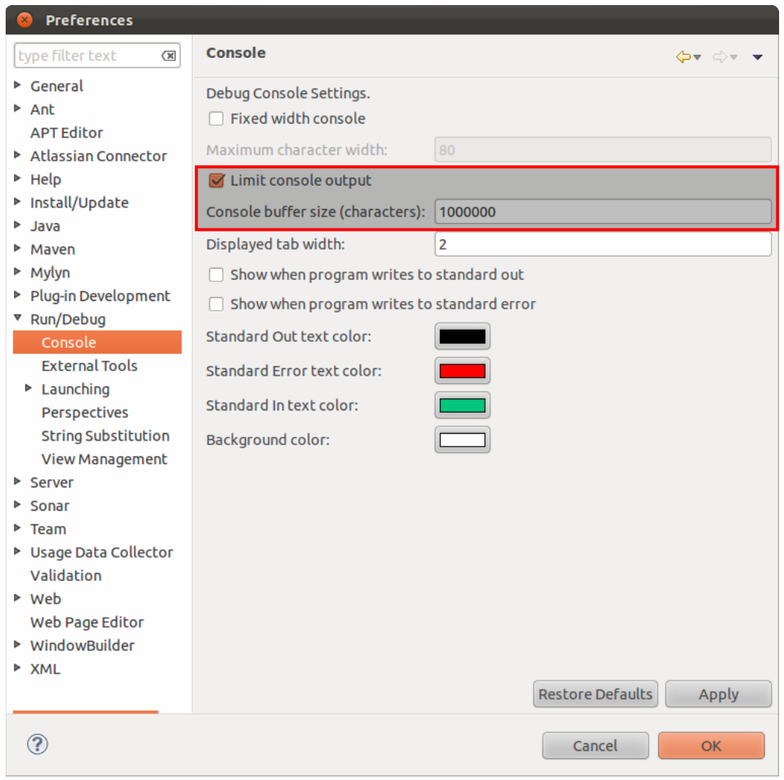
HOW Dans l'onglet **Console**, sélectionner en haut à droite dans la liste déroulante "CVS Console" ou "SVN Console" pour voir toutes les opérations SCM réalisées par Eclipse.

WHY Permet dans certains cas de voir des erreurs qui peuvent survenir, ou de voir sur quel fichier le checkout restait bloqué.

Taille de la console en debug

HOW Sous Windows > Preferences > Run/Debug > Console, changer la taille du buffer de console.

WHY Permet d'avoir + de lignes affichés. Attention, consomme de la mémoire !.



Un exemple de workspace optimisé

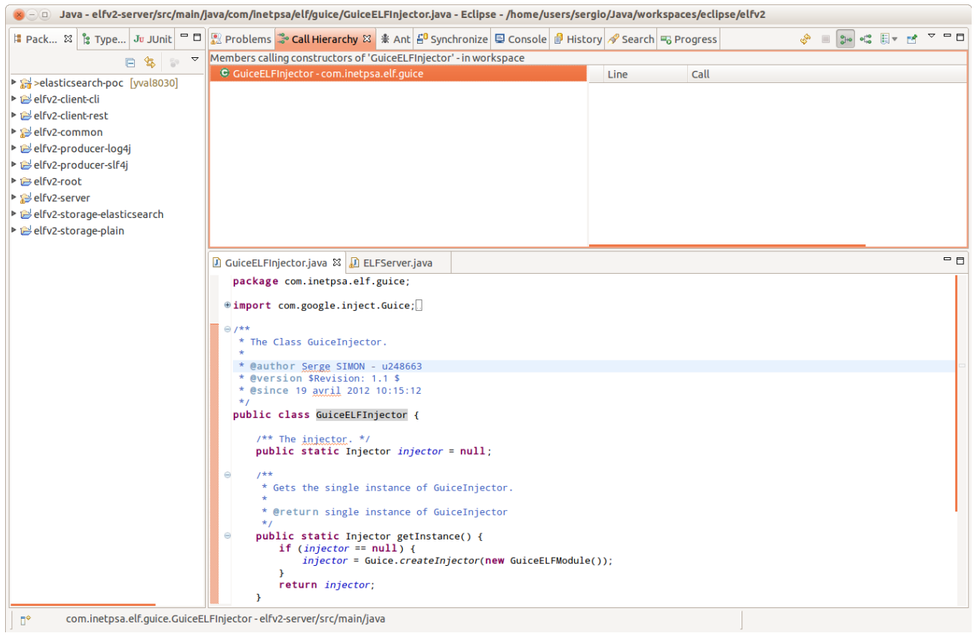
HOW

C'est évidemment propre à chacun, mais voilà comment je configure le mien.
- **Pas de barre d'outils**.
- Seulement **trois zones** : en vertical, le package explorer + JUnit. En horizontal, la zone d'édition. Toutes les vues rassemblées au même endroit : Problems, Call Hierarchy, Ant, Synchronize, Console, History, Search, Coverage, Progress.
- **Pas de line number** sur les éditeurs.

WHY

Maximiser la zone d'édition du code.

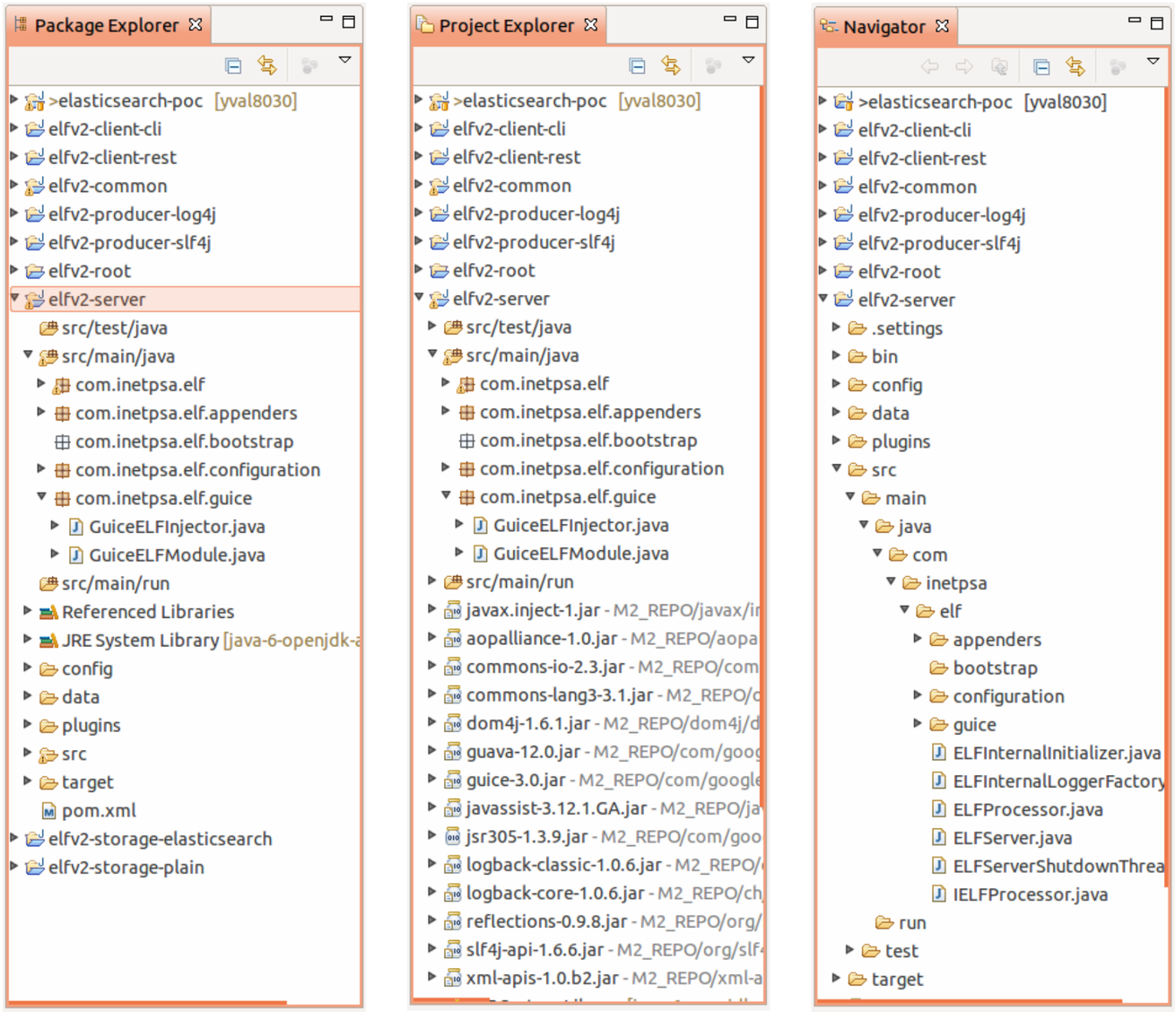
Avoir ni trop ni trop peu de vues à disposition (notamment la vue Synchronize, évite de switcher sans arrêt sur la Perspective CVS/SVN).



Différentes vues disponibles

HOW Privilégiez la vue **Package Explorer** et au besoin le raccourci clavier " `CTRL - SHIFT - R` " pour ouvrir les fichiers cachés genre .classpath

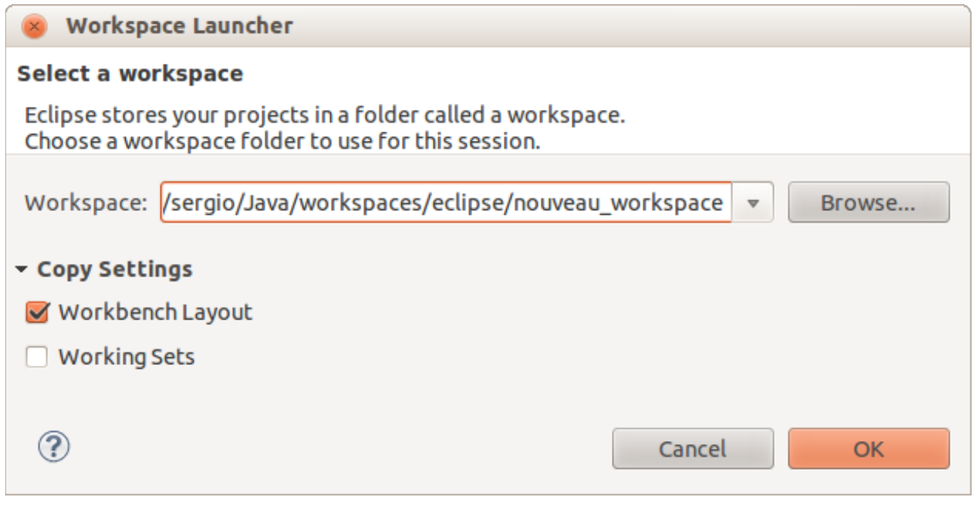
WHY **Package Explorer** montre une vision Java du projet (mais ne montre pas les fichiers cachés genre .classpath). Les librairies externes sont notamment masquées sous un item "Referenced Libraries" (ce qui est plus clair).



Initialiser un nouveau workspace à partir des settings d'un précédent

HOW Démarrer un Eclipse bien configuré. Faire File > Switch workspace > Other. Saisir le nom du nouveau workspace et cocher la case "**Copy Settings > Workbench Layout**".

WHY Permettra de retrouver la même disposition sur le nouveau workspace que sur l'ancien.

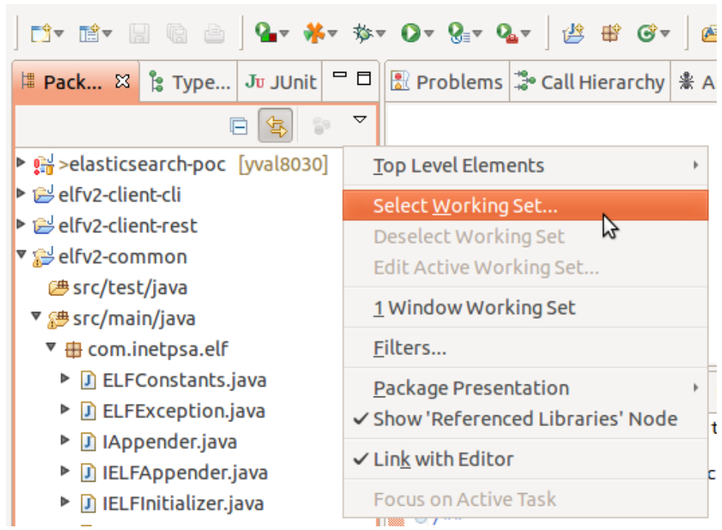


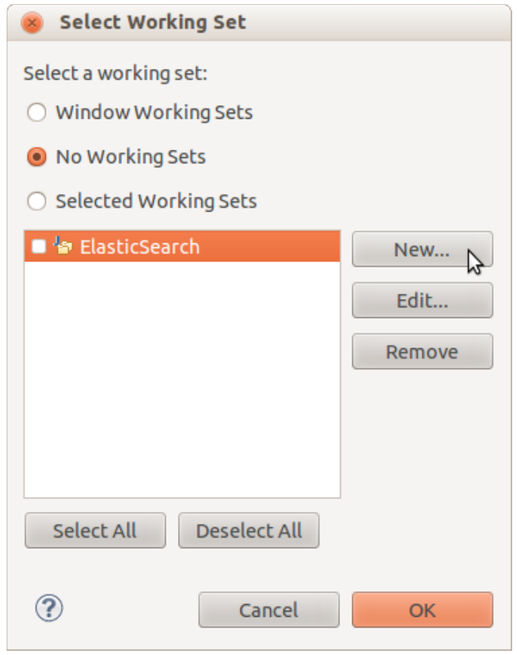
Regrouper ses projets en groupes de projets ("working sets")

HOW Accès à l'IHM de création/sélection des workings set. Affectation d'un ou plusieurs projets.

WHY Les projets ne figurant pas dans le working set "disparaissent" du workspace courant. Attention, les projets ne sont PAS fermés (un `CTRL` - `SHIFT` - `T` prendra par ex. toujours les projets hors-working set dans son contexte)

INFO Il est également conseillé de **fermer les projets inutilisés**. Ainsi, ils n'apparaissent plus lors des recherches (que ce soit recherches fichiers ou recherches Java type héritage, etc.)



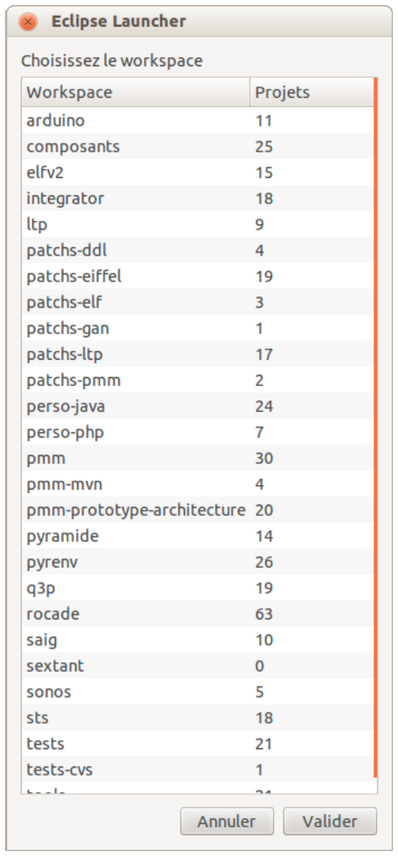


Répartitions des workspaces

HOW Conseil : faire un workspace par thématique ou par niveau de patchs (et éviter de tout mettre dans le même workspace).

WHY Améliore le démarrage d'Eclipse. Améliore toutes les recherches (appel d'une méthode, héritage de classes, etc.). Evite les erreurs (notamment quand on commence à avoir plusieurs branches à descendre). Evite les renommages de projets (un workspace par patch / branche = les noms de projets restent toujours les mêmes).

INFO **Astuce :** ajouter le paramètre "**-showlocation**" (sur le raccourci Eclipse ou dans le fichier "eclipse.ini") pour afficher dans la barre de titres le nom du workspace (afin de mieux s'y retrouver sur `ALT` - `TAB`)



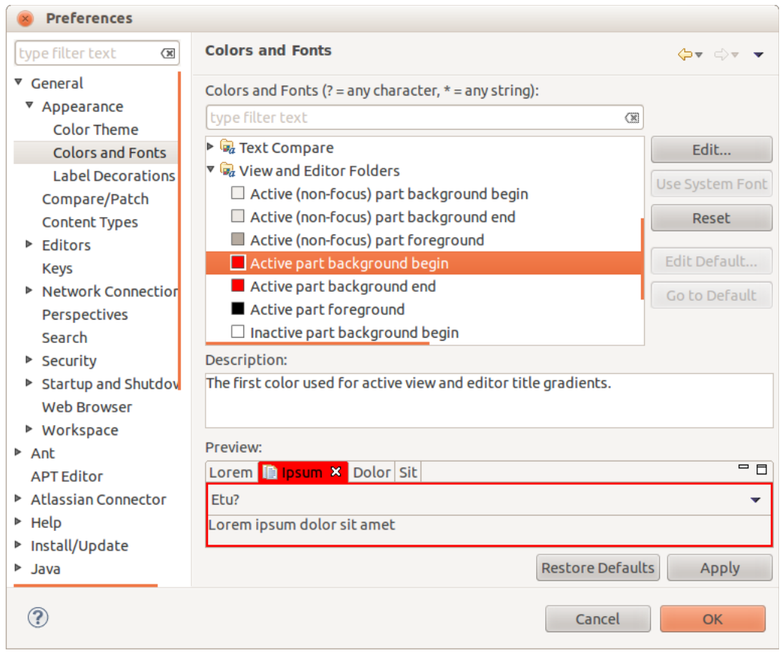
Colorer en rouge le fond d'un workspace de prod

HOW

Sur le workspace considéré, sous General > Appearance > Colors and Fonts, sélectionnez "View and Editor Folders" puis "Active Part background begin / end" et passez ces champs en couleur rouge

WHY

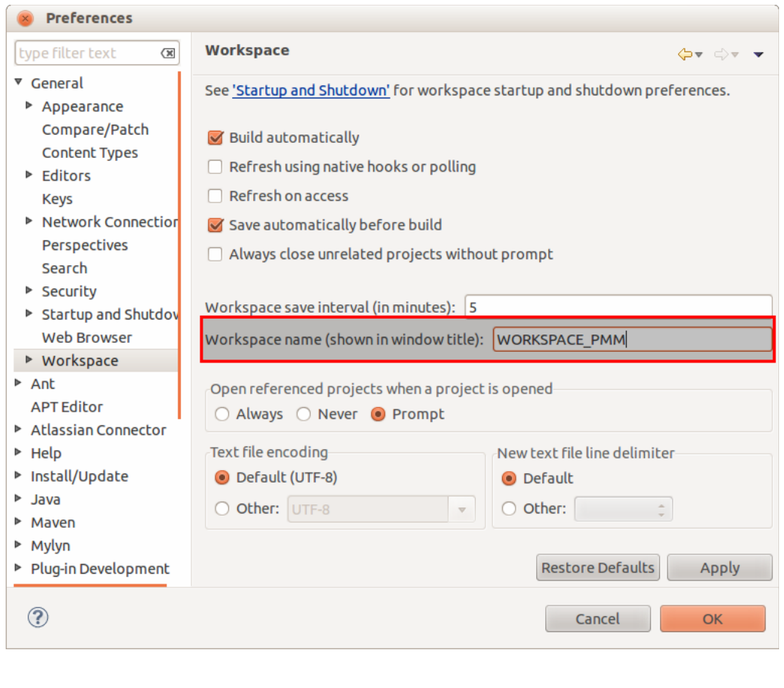
Permet, si on a un workspace par ex. réservé à un ou plusieurs projets de configuration à destination de la prod (ex. que des projects occurrences Pyramide v5 ne contenant que des .xml/.sql), de démarquer visualement ce workspace par rapport aux autres.



Nommer les workspaces pour mieux les retrouver sur alt-tab

HOW Sous Preferences > General > Workspace, modifiez le champ "Workspace Name"

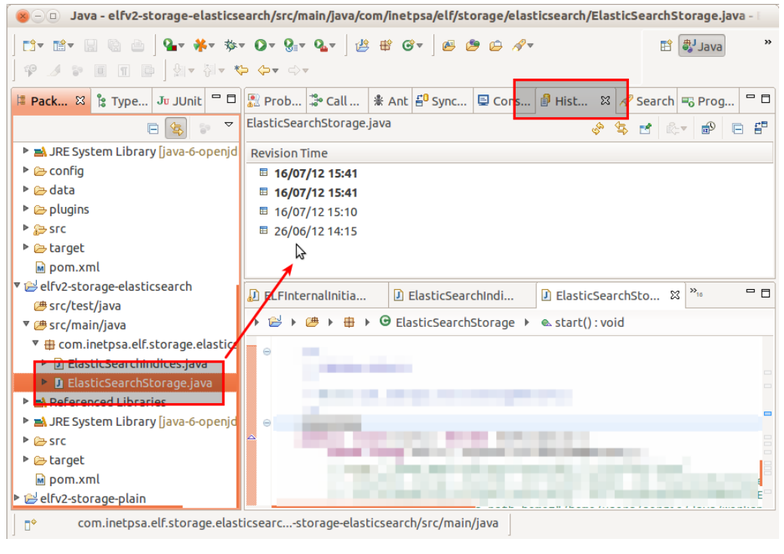
WHY Permet sur `ALT` - `TAB` de mieux voir le nom du workspace. Remplace l'astuce du **-showlocation** évoquée au chapitre Répartitions des workspaces.



Accès rapide à l'historique d'une classe

HOW Par drag'n'drop depuis le Package Explorer vers la vue History

WHY N/A



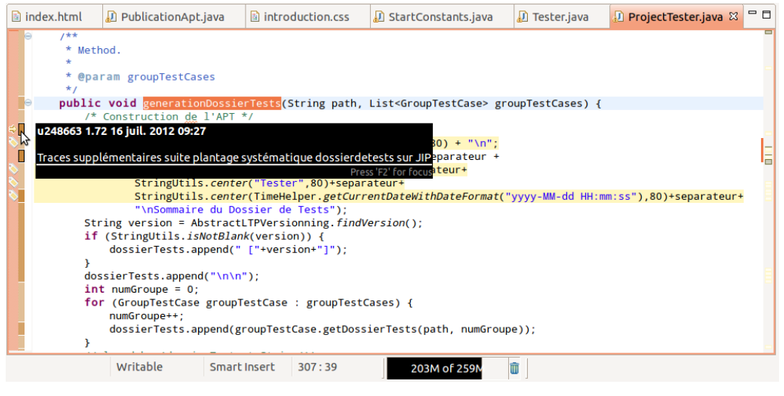
Voir rapidement qui a modifié quel morceau de code et quand

HOW

Sur une classe ayant un historique CVS/SVN : dans la gouttière (là où se posent les breakpoints), faire click droit > Show Annotations

WHY

Plus rapide que de passer par l'historique / permet une navigation directement dans le code indépendamment de la version de la classe (vision transversale)



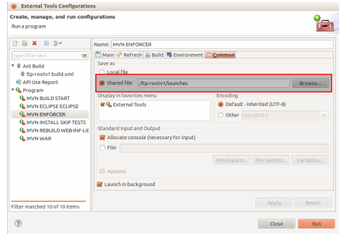
Stocker les .launch (external tools) sous CVS/SVN

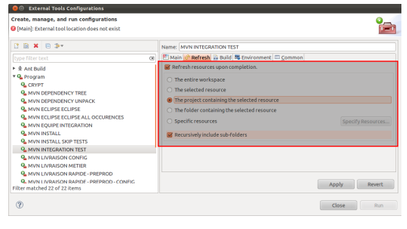
HOW

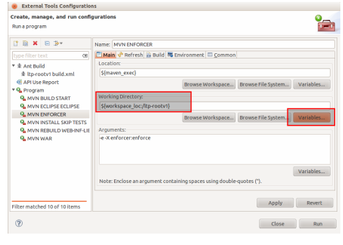
Remonter les configuration de "run" d'Eclipse sous CVS/SVN (mais attention aux infos contenant des chemins en dur genre sur des classpath ou des paths vers des commandes système ...)
Pour les .launch qui modifient des éléments au sein du projet (fichier .classpath sur un "mvn eclipse:eclipse" par ex., activer le Refresh automatique en fin d'opération) (voir capture)

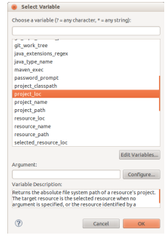
WHY

Partage des .launch entre plusieurs développeurs (les .launch stockés dans un projet apparaissent automatiquement dans la liste des External Tools sous Eclipse)









Entourer une chaîne avec CTRL-ESPACE

HOW Sélectionner la chaîne ou la méthode. Activer `CTRL` - `ESPACE` . La liste déroulante propose par exemple "sysout". Le résultat du sysout englobera la chaîne qui était sélectionnée auparavant.

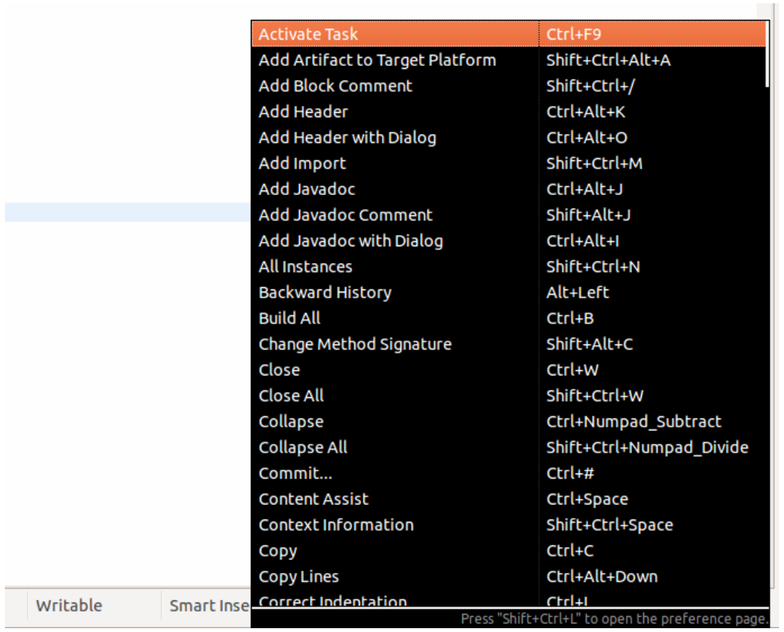
WHY Evite de faire des manips (copier/coller, etc.) après l'apparition du template.

[raccourci clavier] Lister les raccourcis clavier

HOW `CTRL - SHIFT - L`

HOW Lister tous les raccourcis claviers d'Eclipse les plus intéressants / importants !

INFO Un 2e enchaînement de cette combinaison de touches permet d'accéder directement à la fenêtre d'édition des raccourcis.

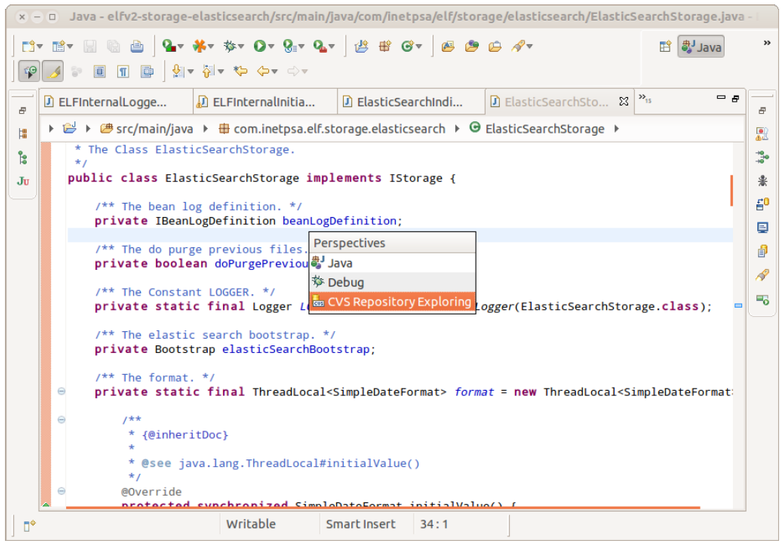


[raccourci clavier] Switcher entre les perspectives Eclipse

HOW CTRL - F8

WHY Permet de navigurer entre les perspectives Eclipse actuellement ouvertes (par ex., Java / Debug / CVS).

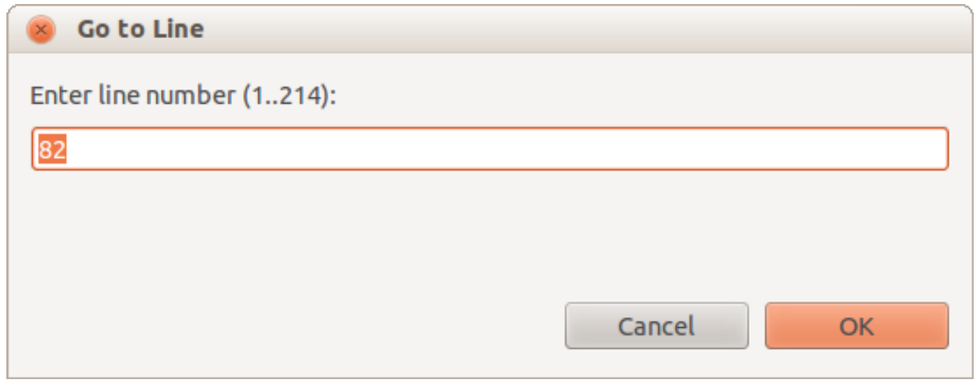
INFO Pensez à bien fermer les perspectives inutilisées pour ne pas gaspiller de mémoire.



[raccourci clavier] Sauter à un numéro de ligne quelconque

HOW `CTRL - L`

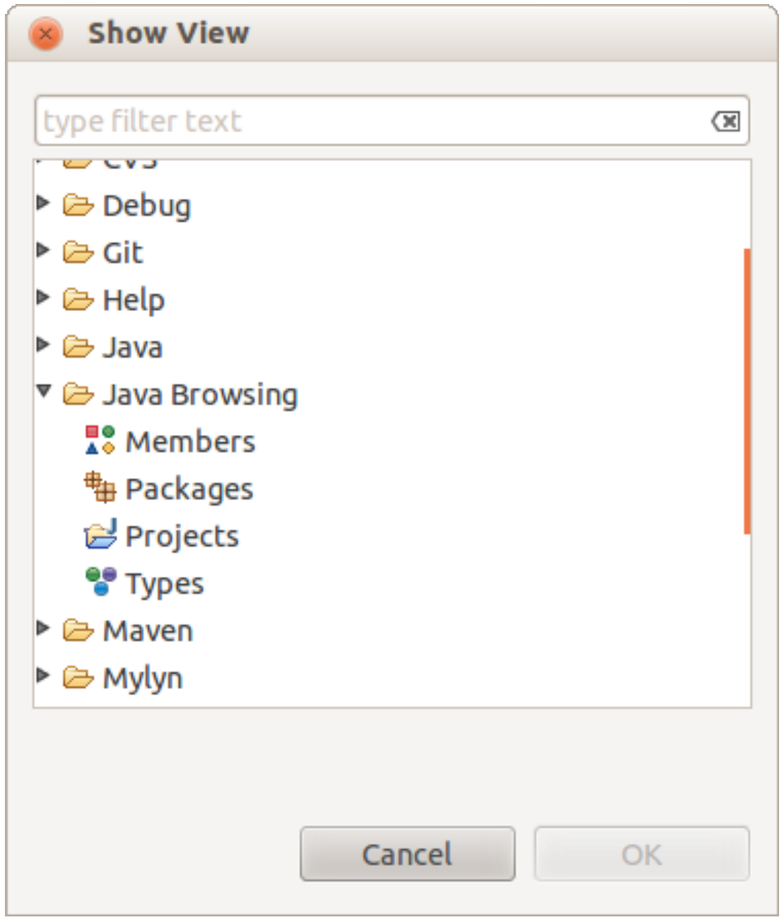
WHY Aller à un numéro de ligne (indispensable pour retrouver un numéro de ligne suite à une stacktrace).



[raccourci clavier] Liste des vues disponibles

HOW `CTRL - ALT - Q , Q`

WHY Afficher la fenêtre de sélections des vues éclipse.



[raccourci clavier] Renommer un élément

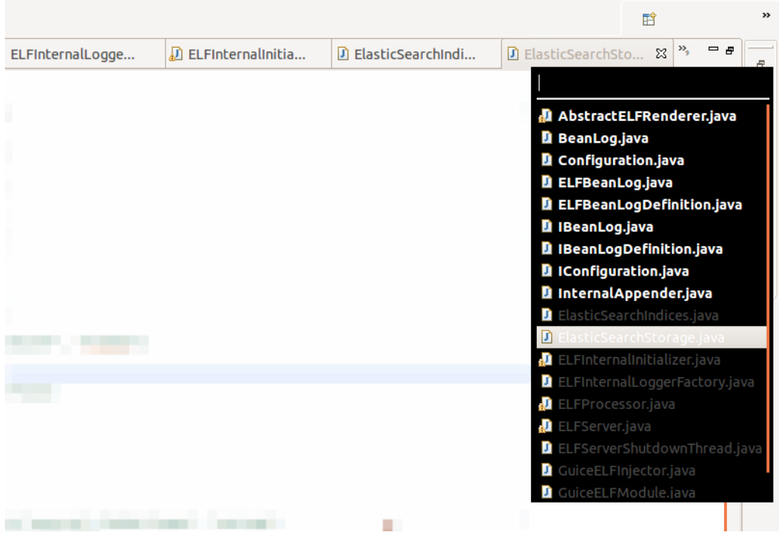
HOW `SHIFT - ALT - R`

WHY Renommer l'élément courant.

[raccourci clavier] Sélection rapide d'un éditeur

HOW `CTRL - E`

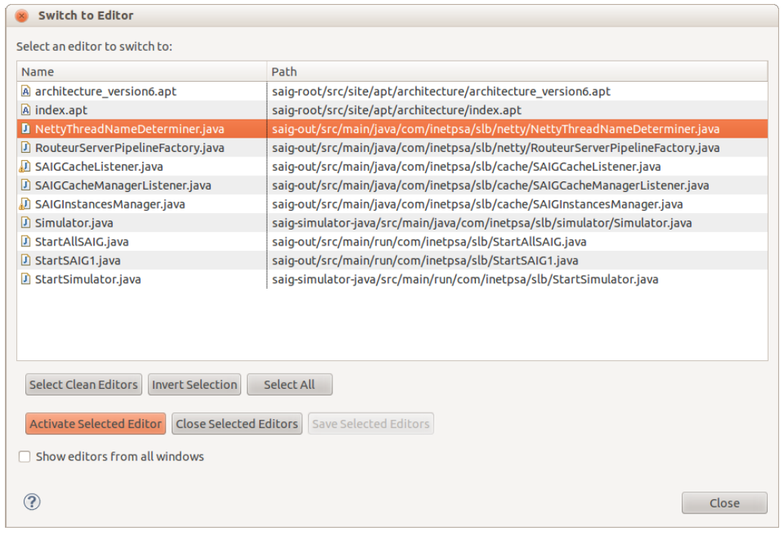
WHY Affiche dans une popup (pour sélection rapide au clavier) la liste des onglets d'édition ouverts dans Eclipse. Les onglets déjà visibles sont affichés en "actifs", ceux invisibles en "inactif". Il est possible de taper au clavier les premières lettres de l'éditeur que l'on recherche.



[raccourci clavier] Sélection d'un éditeur via boîte de saisie

HOW `CTRL - SHIFT - E`

WHY Affiche (pour sélection rapide au clavier) la liste des onglets d'édition ouverts dans Eclipse



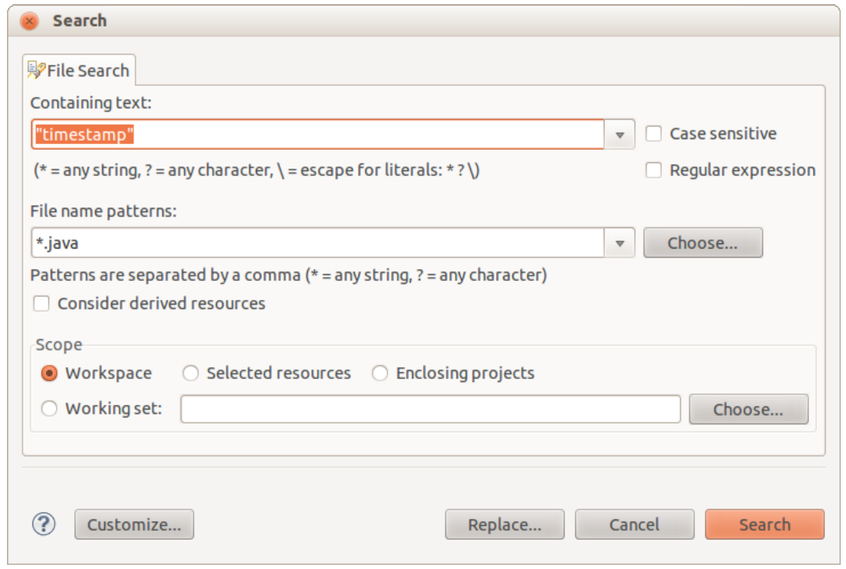
[raccourci clavier] Recherche

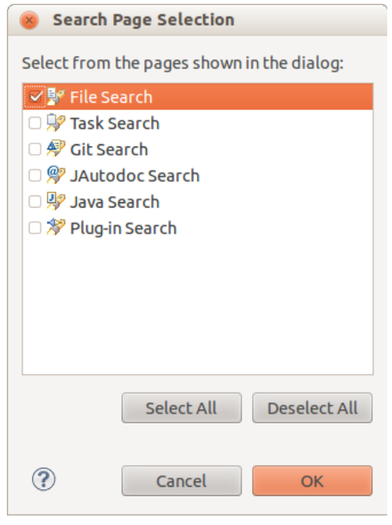
HOW `CTRL - H`

WHY Recherche au sein de tous les fichiers.

INFO **Remarque :** il est conseillé via le bouton "Customize" en bas à gauche de retirer les onglets qu'on n'utilise jamais.

Ne pas oublier qu'il est possible d'utiliser des **expressions régulières** depuis cette fonction, et de faire un **replace global** au sein de tout le projet.





[raccourci clavier] Recherche incrémentale

HOW `CTRL` - `J`

WHY Recherche incrémintale à la volée. Appuyer sur `CTRL` - `J` pour passer en mode "incremintal find" et commencez à taper. Une fois le texte à chercher satisfaisant, chaque `CTRL` - `J` suivant amène à l'occurrence suivante. Au final, beaucoup + rapide que de faire un `CTRL` - `F` .

[raccourci clavier] Commenter / décommenter

HOW

`CTRL` - `SHIFT` - `C` ou `CTRL` - `SHIFT` - `/`

WHY

Ajoute des commentaires sur le bloc courant.

`CTRL` - `C` ajoute des // sur une classe Java (sur une ou plusieurs lignes)

`CTRL` - `SHIFT` - `C` ajoute des // sur une classe Java (sur une ou plusieurs lignes) FIXME

Fonctionne sur les .java, .html, .xml, etc.

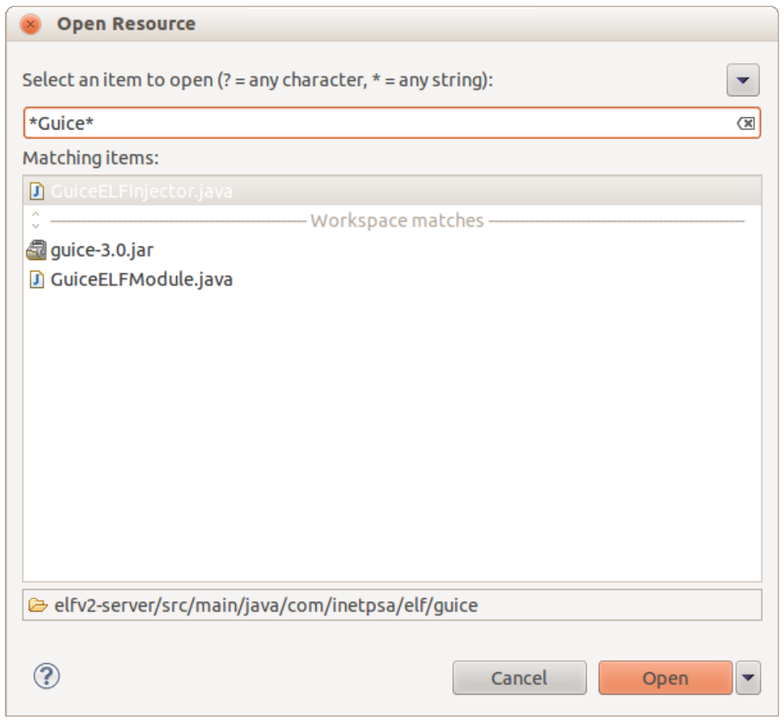
INFO

Bien veiller à désactiver le "block comment formatting" sous Windows > Preferences > Java > Code Style > Formatter > Edit > Comments pour ne pas voir les commentaires multilignes re-formatés

[raccourci clavier] Accès rapide à une ressource

HOW `CTRL - SHIFT - R`

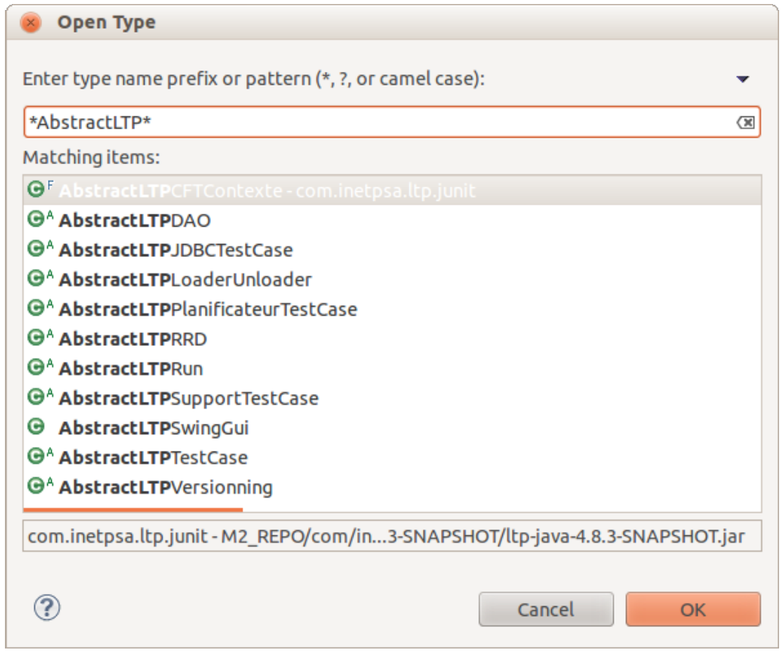
WHY Ouvrir rapidement un élément quelconque (classe, fichier HTML, etc.) en saisissant quelques lettres (ou une expression régulière). Veillez à bien mettre en "derived" les répertoires **target** pour qu'ils n'apparaissent pas dans ce type de boîtes de dialogue.



[raccourci clavier] Accès rapide à un type

HOW `CTRL - SHIFT - T`

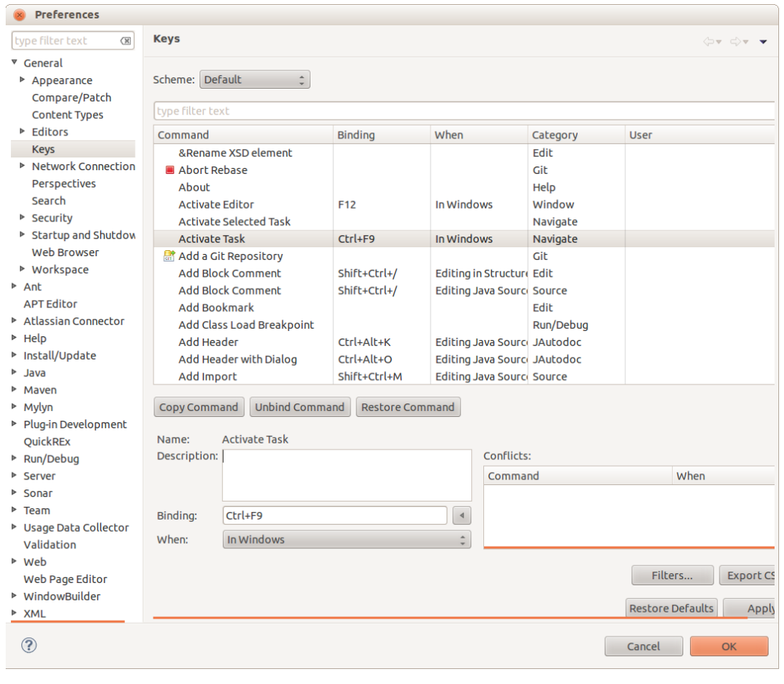
WHY Ouvrir rapidement une classe quelconque en saisissant quelques lettres.



Personnaliser ses raccourcis claviers

HOW Sous General > Keys

WHY Affectation d'une action quelconque d'Eclipse à un nouveau raccourci clavier. Selon les usages de chacun ...



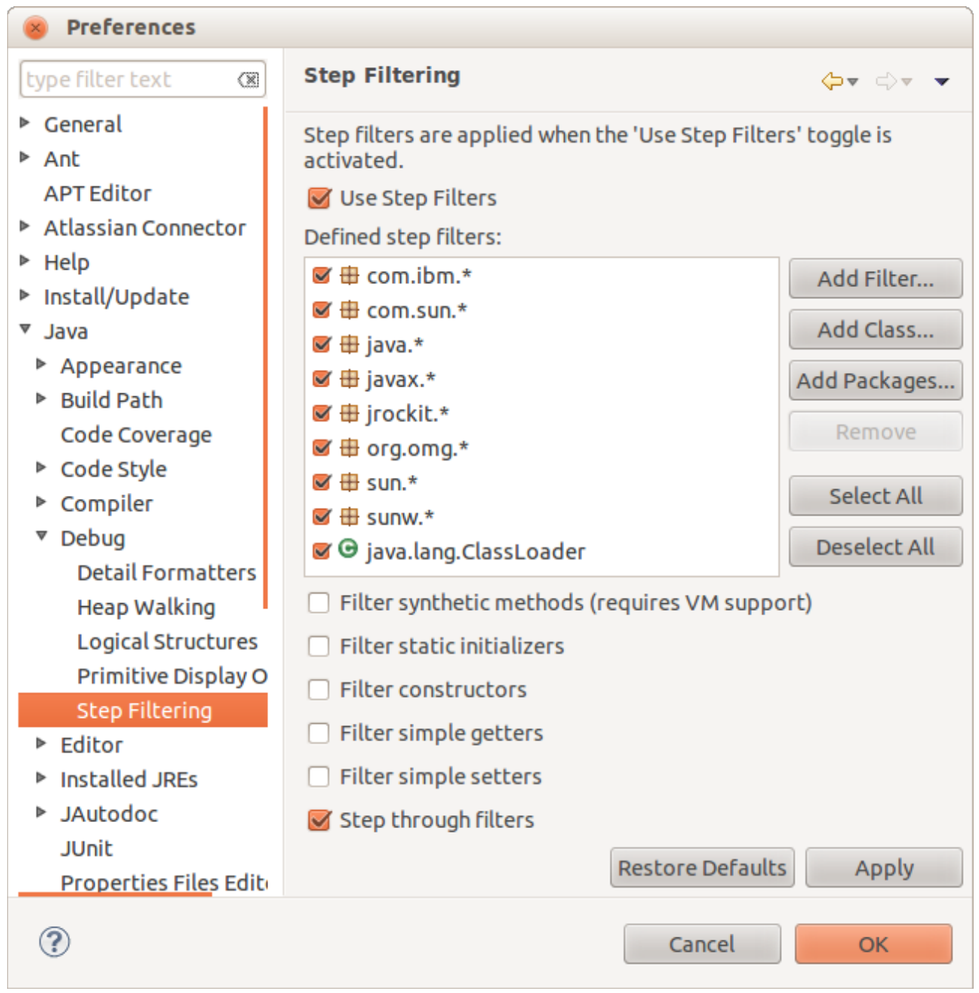
Chapitre 5. Java

Optimisations du debuggage

HOW Dans Preferences > Java > Debug > Step Filtering, tout cocher et activer.

WHY Indispensable : permet de ne PAS rentrer dans les classes Java ou les classes des librairies communes lorsqu'on débug en pas à pas ! (l'exclusion ne se fait que sur les masques spécifiés ; il est possible d'ajouter de nouveaux masques au besoin, par ex. "org.apache.*" ou, dans un environnement métier, un masque pour une librairie type Q3P ou LTP).

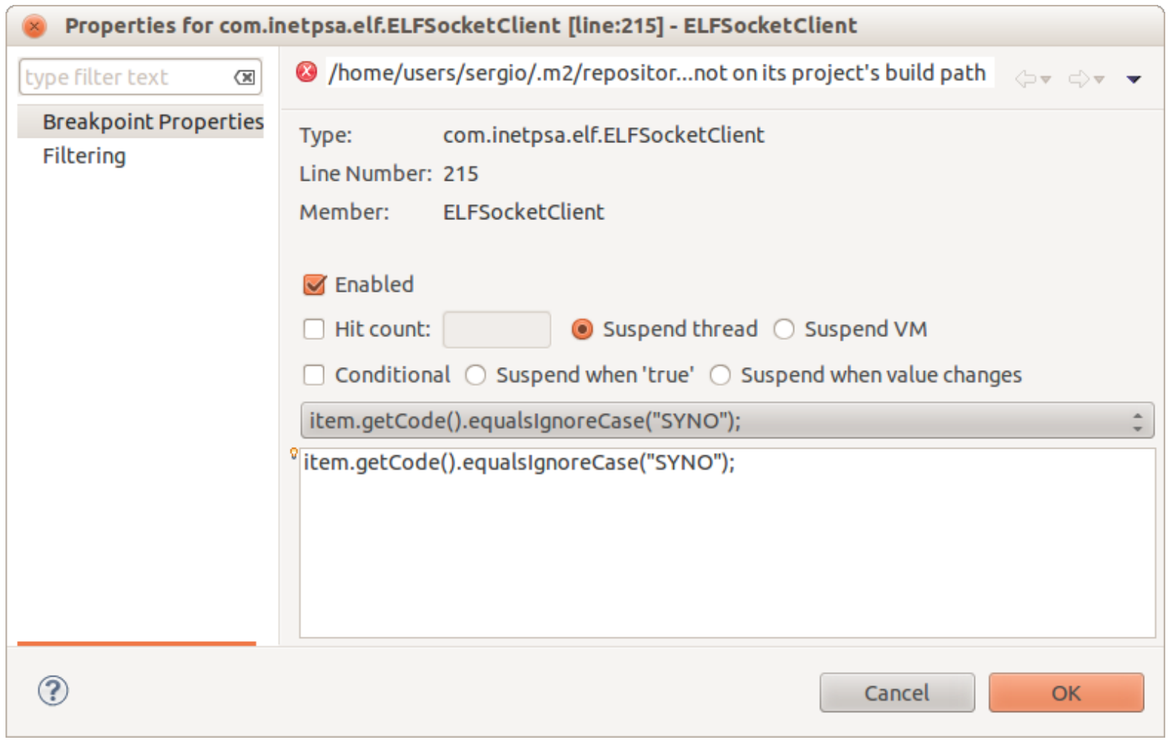
INFO Si vous utilisez des getters/setters "simples" (qui n'ont pas de code supplémentaires), il peut être intéressant également de cocher les options pour ne pas débugger les getter/settter (sur la même IHM).



Breakpoint conditionnel

HOW Sur les propriétés d'un point de debuggage, possibilité de mettre n'importe quelle condition (à écrire en java avec les objets accessibles sur ce breakpoint). Accès par "Breakpoint properties" depuis la vue idoine ou par click droit directement dans l'éditeur sur le breakpoint.

WHY Indispensable : permet par ex. sur une boucle d'activer le breakpoint seulement quand telle ou telle valeur est atteinte.



Optimiser le "content-assist"

HOW

Dans Preferences > Java > Editor > Content Assist > Advanced, quasiment tout décocher (voir capture) (mais conservez impérativement "Java" et "Template") .

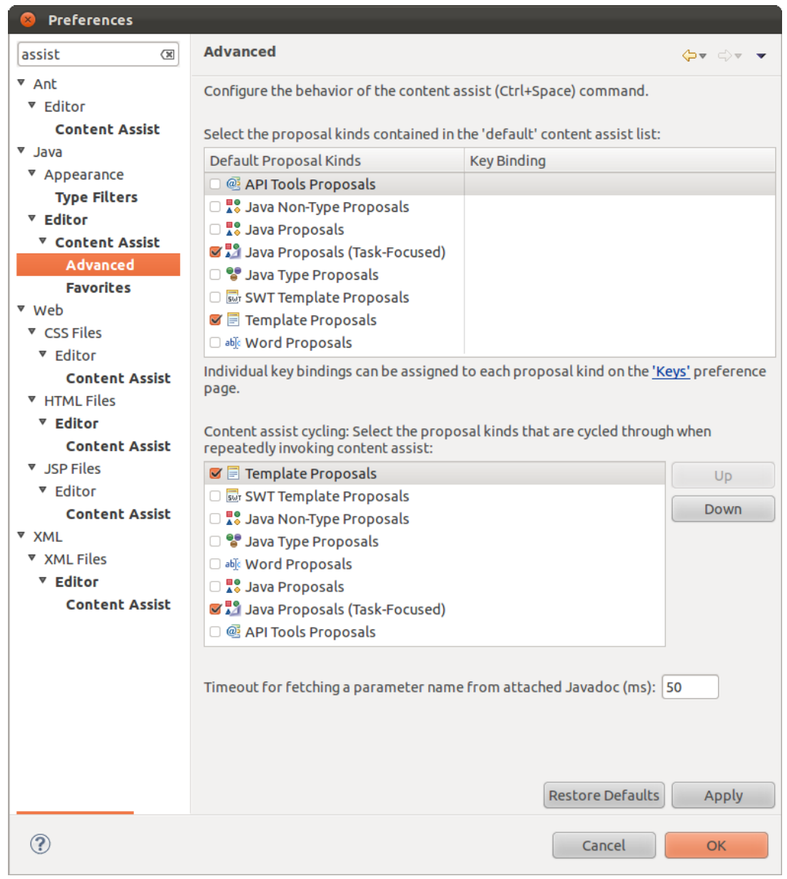
WHY

Le "content-assist" est **l'aide à la complétion par** `CTRL` - `ESPACE` .

Par défaut il propose beaucoup de choses (élements SWT, JPA, JAX-WS, PDE pour la création de plugins Eclipse, etc. - sauf bien sûr si vous utilisez certaines de ces fonctionnalités).

Retirer les blocks de complétions possible accélère ensuite l'usage du `CTRL` - `ESPACE` pour l'écriture de Java standard.

Encore + important si utilisation d'Eclipse J2EE (car beaucoup de de protocoles proposables)).

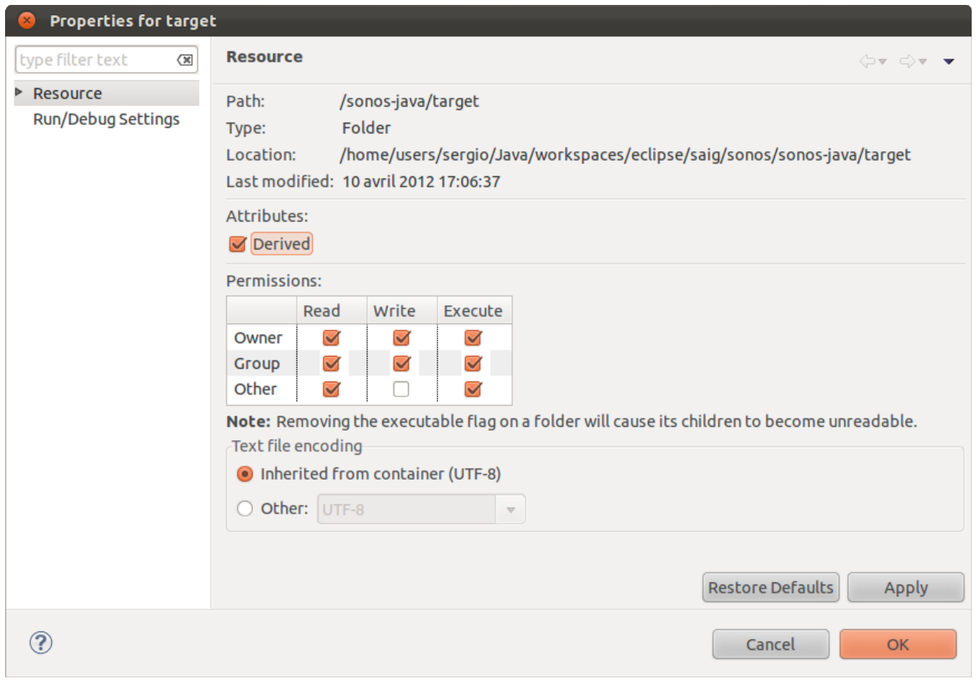


Répertoires "derived"

HOW Bouton droit sur un répertoire > Propriétés > Case à cocher "Derived"

WHY Quand vous faites une recherche au sein d'un projet, Eclipse montre souvent des résultats dans des répertoires contenant du code généré (ex. "bin/" ou "dist/" ou "target/"), ce qui ne sert à rien.
Pour masquer ces répertoires sous Eclipse il faut les indiquer comme "DERIVED" ce qui l'exclut de toutes les rercherches (qui deviennent du coup beaucoup plus rapides une fois les répertoires "target" (Maven) ou "dist", "build" (Ant) ou "work" (tomcat) marqués comme tel).

INFO Attention, ne fonctionne qu'avec le **pyr-pom-root** et la bonne configuration Maven (sinon Maven efface le répertoire à chaque "mvn clean", ce qui entraîne alors la perte du flag sous Eclipse !)



Création rapide d'une classe

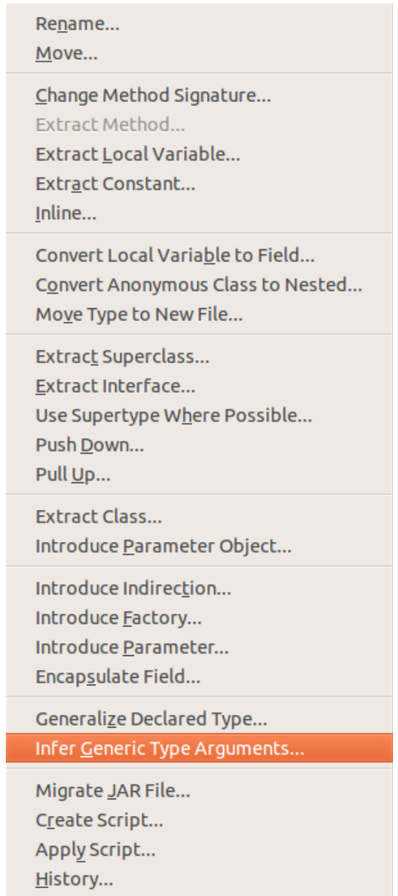
HOW Créer la classe à la volée dans le fichier courant, puis utiliser la fonction Refactor > Move Type To New File

WHY Evite pas mal de manipulations (File > Create New Class).

Trouver automatiquement depuis Eclipse les Generics

HOW Sur une ou plusieurs classes sélectionnées, utiliser Refactor > Infer Generic Types Arguments

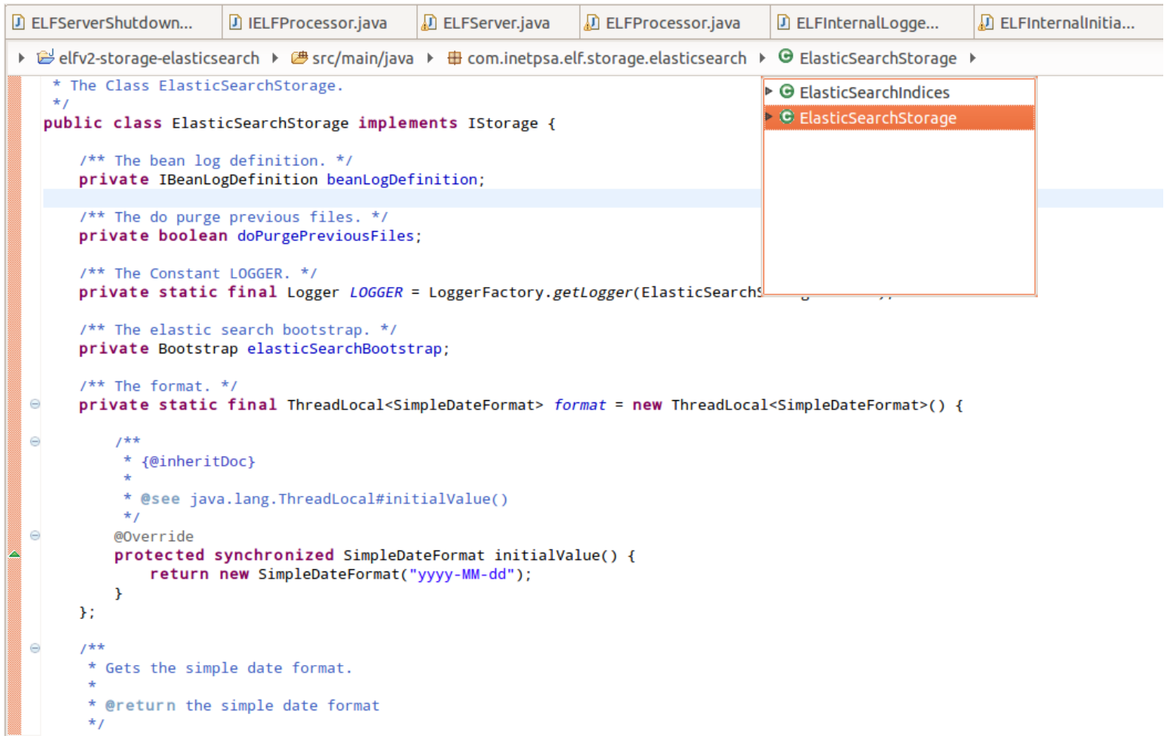
WHY Permet de transformer une classe Java 1.4 ou Generics Java 1.5. Fonctionne à 90% (les cas qu'Eclipse n'arrive pas à identifier sont laissés en l'état).



Barre de navigation rapide

HOW Depuis Navigate > Show In Breadcrumb. Se désactive par bouton droit sur la ""Breadcrumb bar" elle même.

WHY = Navigation rapide au sein des packages / méthodes.



Utilisation des fonctions d'Eclipse pour la création de beans

HOW

Utiliser tout ce qui est dans le menu "Source" :
- création des **getters / setters** (Source > Generate Getters/Setters)
- **surcharge** de méthodes (Quick Fix `CTRL - SHIFT - 1` > Add unimplemened methods)
- création de **constructeurs** (héritage ou à partir des champs) (Source > Generate constructors from ...)
- mise en place de la **Javadoc** (plugin JAutoDoc, `ALT - SHIFT - J`)
- **imports automatiques** (`CTRL - SHIFT - O`)
- **formattage automatique** (`CTRL - A` puis `CTRL - SHIFT - F`)

WHY

Pour en pas s'embêter avec des manipulations répétitives

Création rapide des champs depuis le constructeur

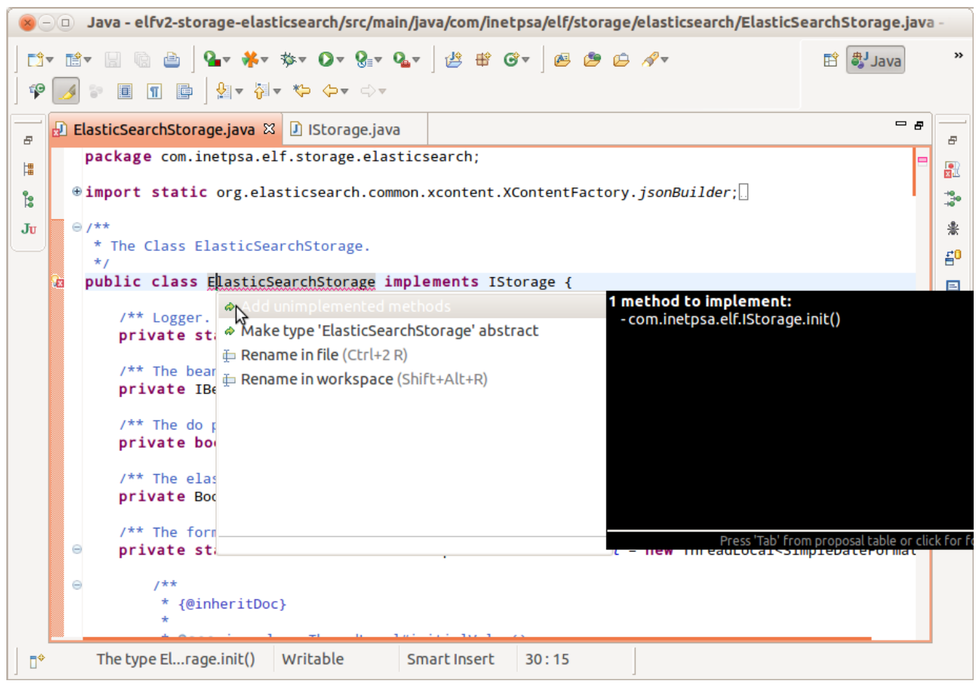
HOW Créer un constructeur. Placer les champs dans la signature. Faire `CTRL` - `SHIFT` - `1` (quick fix) et choisir l'option d'affectation vers un nouveau champ "**Assign parameter to new field**"

WHY Pour en pas s'embêter avec des manipulations répétitives

[raccourci clavier] Quick Fix

HOW `CTRL - SHIFT - 1`

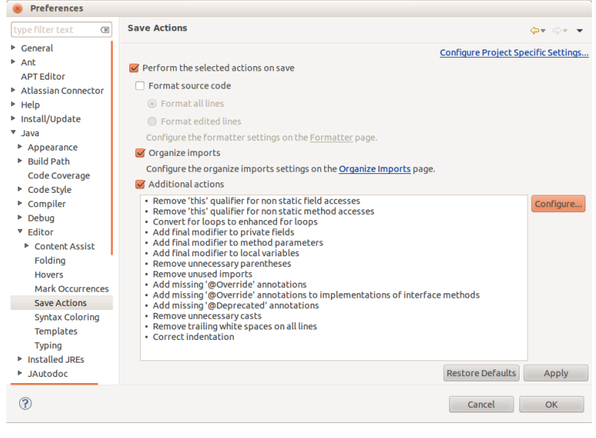
WHY Affiche le menu de correction rapide, contenant des entrées contextuelles par rapport au problème courant (correction d'import, création de méthodes, ajout de paramètre dans la signature, etc.).

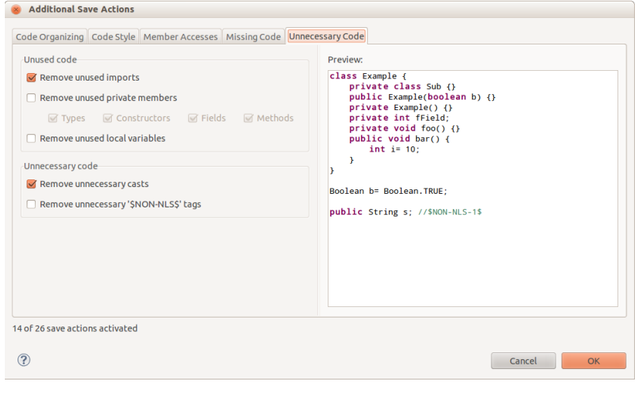


Actions de sauvegarde automatique

HOW Windows > Preferences > Java > Editor > Save Actions

WHY Permet d'automatiser toute une floppée d'opérations lors de chaque sauvegarde, par ex. :
- **suppression automatique des imports inutiles**
- **ajout automatique** des annotations type @Override
- ajout des mots-clés manquants au sein du code, ex. mot-clé "**final**" quand il est nécessaire (sur les paramètres et/ou à l'intérieur des méthodes et/ou en variable de classe)
- conversion des boucles for en version étendue
- etc.





Formattage spécifique Java

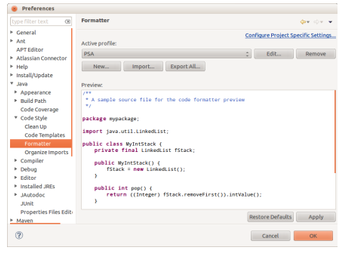
HOW Windows > Preferences > Java > Formatter

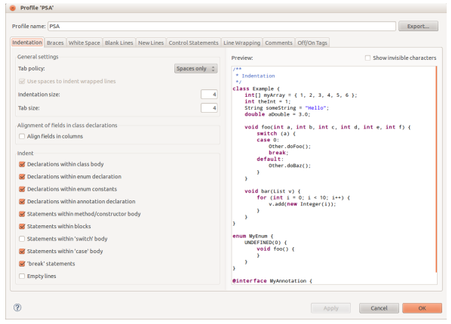
WHY Permet de définir tout un ensemble de propriétés intéressantes quand au formattage du code : - **taille maximale des lignes** ou des commentaires (indispensable lors du formattage automatique)
- formattage des commentaires
- gestion des tabulations / espaces afin d'être homogène (espaces partout recommandés)
- etc.

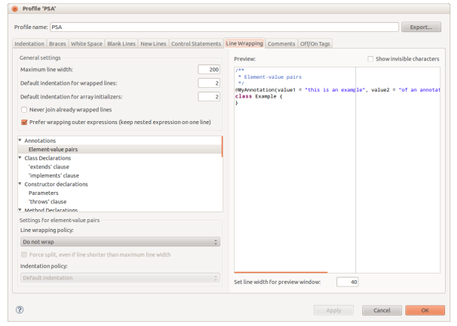
INFO **Astuce :** le formattage du code peut également être appliqué automatiquement en "Save Action"

INFO **Info :** cette paramétrie est stocké soit de manière globale au workspace, soit de manière spécifique au projet : dans les .settings, et peut donc dans ce cas de figure être mutualisée facilement sous Eclipse (ce qui est conseillé afin que sur un même projet tout le monde ait les mêmes règles de formattage)

INFO **Astuce :** cette paramétrie peut être stocké dans Maven et réappliqué facilement à plusieurs projets à l'identique







[raccourci clavier] Formattage automatique du code

HOW `CTRL` - `SHIFT` - `F`

WHY Applique le formattage défini côté Eclipse / projet sur la classe ou l'élément sélectionné (faire `CTRL` - `A` puis `CTRL` - `SHIFT` - `F` par ex.)

INFO **Astuce :** fonctionne aussi sans problèmes sur les fichiers XML ou HTML

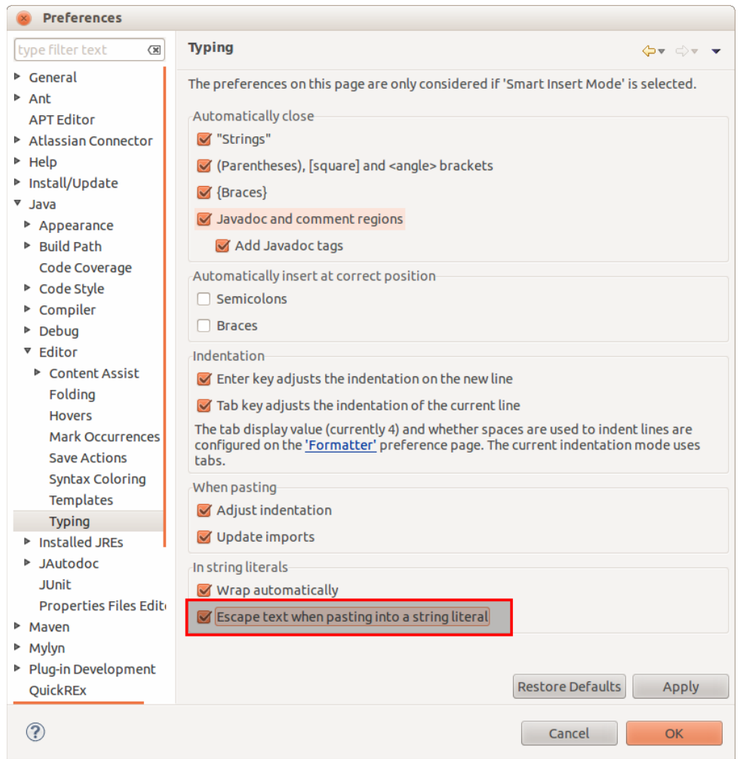
Gestion par Eclipse du multiligne pour les chaînes

HOW Paramétrer sous Preferences > Java > Editor > Typing l'option "Escape text when pasting into a String literral", puis réaliser le coller de texte par ex. dans `String s = "${cursor}";`

WHY Evite d'écriture un bloc fastidieux. Permet de copier/coller un XML, etc. Eclipse gère les encodages de guillemets et les retours à la ligne.

INFO Possibilité d'utiliser la balise Eclipse `// @formatter:off` (et son inverse) pour bloquer le formattage automatique du code sur cette partie

INFO Utiliser chaque fois que possible StringBuilder plutôt que StringBuffer (ce dernier est synchronisé par multithread, donc plus lent). Pas de gains notables pour les sorties standards (pertinent seulement si boucle avec beaucoup d'itérations, ou par commodité). A l'écriture, la version avec "+" sur une String est la plus efficace (car évalué par le compilateur et non à l'exécution).



[raccourci clavier] Lancement rapide d'une classe Java AVEC debug

HOW

F11

WHY

Relance en mode debug le dernier élément exécuté ou l'élément courant (indispensable pour relancer un test plusieurs fois d'affilée par ex.) ;

[raccourci clavier] Lancement rapide d'une classe Java SANS debug

HOW `CTRL` - `F11`

WHY Idem `F11`, mais sans activer le debug

[raccourci clavier] Lancement d'une classe

HOW

`ALT` - `SHIFT` - `X` , `J`

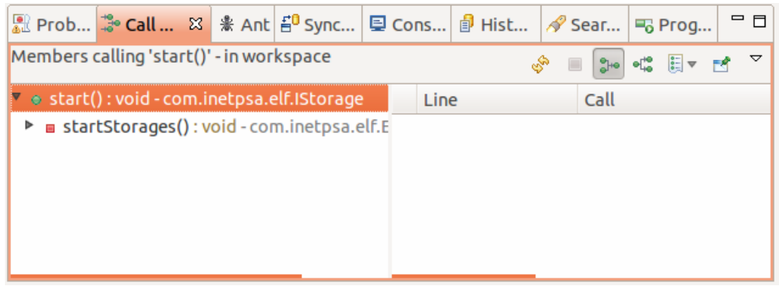
WHY

Exécuter en mode Java le fichier courant. Idem en debug (`ALT` - `SHIFT` - `D` , `J`). Idem pour les tests avec ou sans debug (`ALT` - `SHIFT` - `X` , `T` et `ALT` - `SHIFT` - `D` , `T`).

[raccourci clavier] Hiérarchie d'appel

HOW `CTRL - ALT - H`

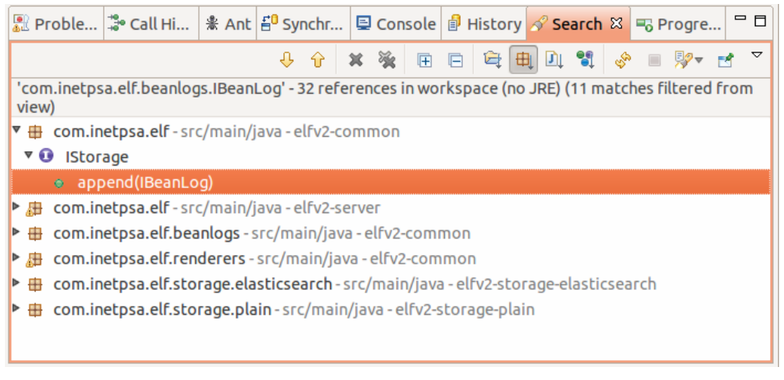
WHY Accéder à la hiérarchie d'appel de la méthode en cours de sélection.



[raccourci clavier] Trouver toutes les références à un élément

HOW `CTRL - SHIFT - G`

WHY Permet de retrouver toutes les références à un élément, par ex. un objet.



[raccourci clavier] Classes filles

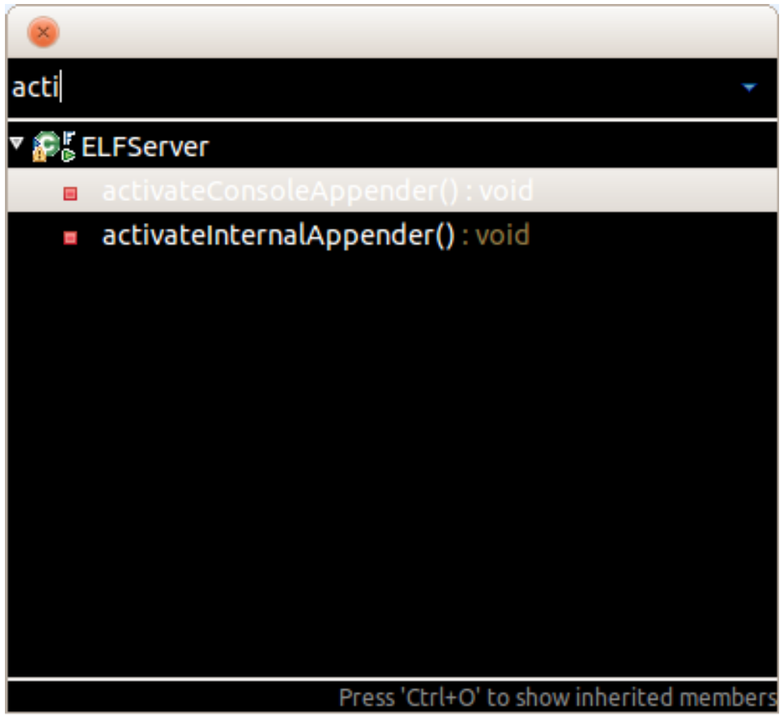
HOW `CTRL - T`

WHY Montre les classes qui implémentent l'objet sélectionné. Fonctionne sur une **Interface** ou sur une **classe parente**. Peut s'appliquer sur le nom de la classe ou sur une méthode.

[raccourci clavier] Ouvrir ressource

HOW `CTRL - O`

WHY Trouver rapidement une méthod (popup avec filtre possible pour sélection rapide au clavier)



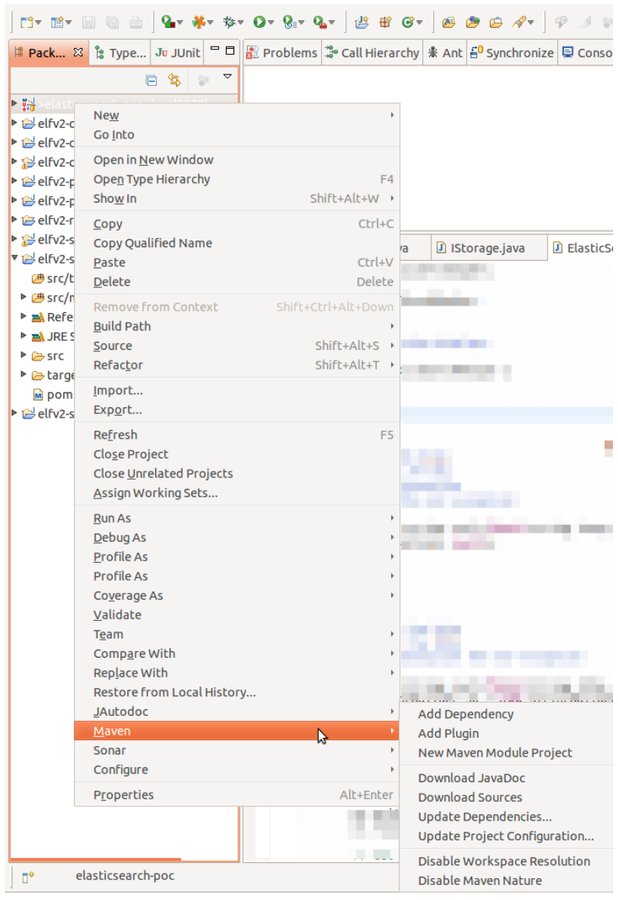
Utiliser les fonctions natives d'Eclipse pour Maven

HOW Activer la configuration via menu contextuel sur le projet > Configure > Convert to Maven Projet. Ensuite un menu contextuel Maven est disponible avec les opérations courantes. Permet également l'édition du POM (attention, très lourd, personnellement je désactive et édite les pom.xml en tant que simple fichier XML).

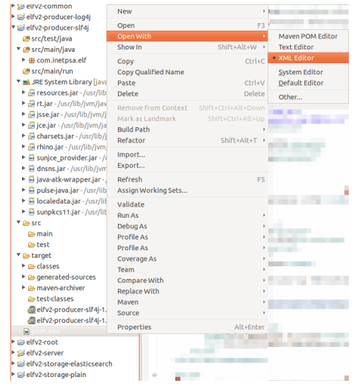
WHY Pas besoin de lancer / configurer d'external tools.

INFO **Bonne pratique :**En interne, préconistation de rester sur des .launch stockés sous CVS : le plugin m4e reste toujours un peu buggé (sur la résolution des dépendances par ex. ou la gestion des plugins en pluginManagement), et les .launch permettent des opérations Maven plus fines (enchaînement de plusieurs opérations par ex.).

INFO **Remarque :** m4e intégré dans Eclipse 3.7 utilise nativement le moteur Maven 3



Fonctions m4e



Ouverture du pom.xml en tant que simple XML

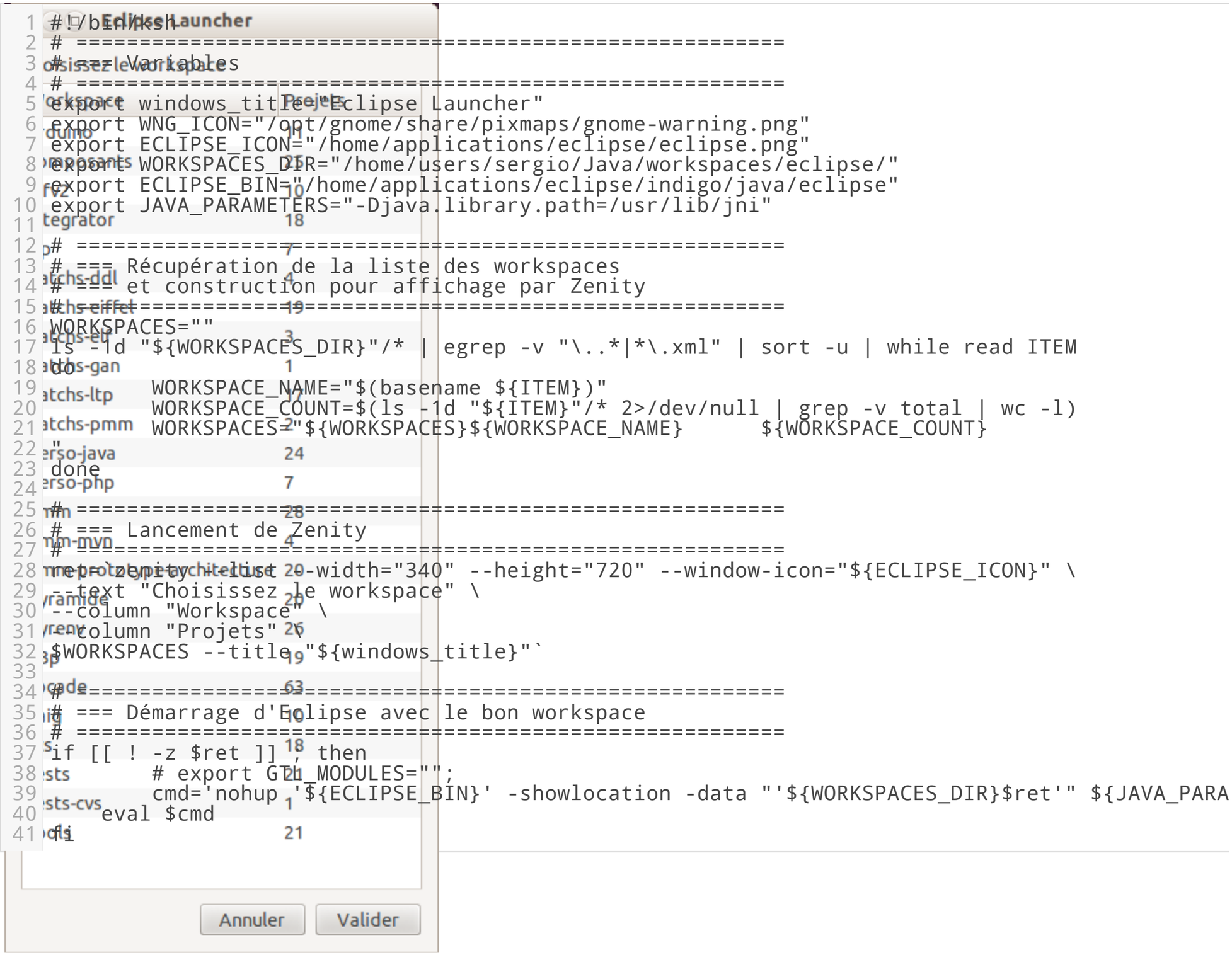
Annexes

Table des raccourcis claviers

Lanceur sous Linux

Permet de choisir le workspace au démarrage parmis tous les workspaces disponibles

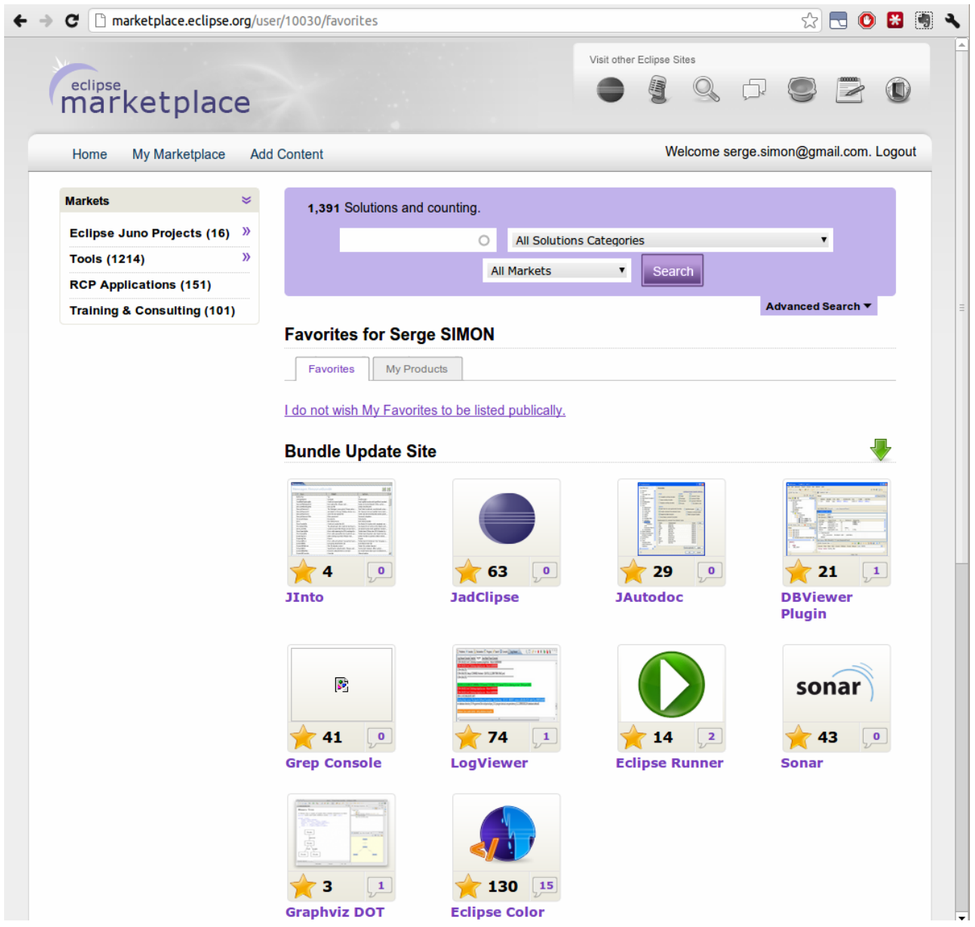
A savoir (pour usage en raccourci Windows par ex.) : démarrage automatique d'un workspace avec le paramètre ligne de commande "**-data**"




```
#!/bin/ksh
# ============================================================
# === Variables
# ============================================================
export windows_title="Eclipse Launcher"
export WNG_ICON="/opt/gnome/share/pixmaps/gnome-warning.png"
export ECLIPSE_ICON="/home/applications/eclipse/eclipse.png"
export WORKSPACES_DIR="/home/users/sergio/Java/workspaces/eclipse/"
export ECLIPSE_BIN="/home/applications/eclipse/indigo/java/eclipse"
export JAVA_PARAMETERS="-Djava.library.path=/usr/lib/jni"

# ============================================================
# === Récupération de la liste des workspaces
# === et construction pour affichage par Zenity
# ============================================================
WORKSPACES=""
ls -1d "${WORKSPACES_DIR}"/* | egrep -v "\..*|*\.xml" | sort -u | while read ITEM
do
    WORKSPACE_NAME="$(basename ${ITEM})"
    WORKSPACE_COUNT=$(ls -1d "${ITEM}"/* 2>/dev/null | grep -v total | wc -l)
    WORKSPACES="${WORKSPACES}${WORKSPACE_NAME}      ${WORKSPACE_COUNT}
"
done

# ============================================================
# === Lancement de Zenity
# ============================================================
ret=`zenity --list --width="340" --height="720" --window-icon="${ECLIPSE_ICON}" \
--text "Choisissez le workspace" \
--column "Workspace" \
--column "Projets" \
$WORKSPACES --title "${windows_title}"`

# ============================================================
# === Démarrage d'Eclipse avec le bon workspace
# ============================================================
if [[ ! -z $ret ]] ; then
        # export GTK_MODULES="";
        cmd='nohup '${ECLIPSE_BIN}' -showlocation -data "'${WORKSPACES_DIR}$ret'" ${JAVA_PARA
        eval $cmd
fi
```

Liens (web)

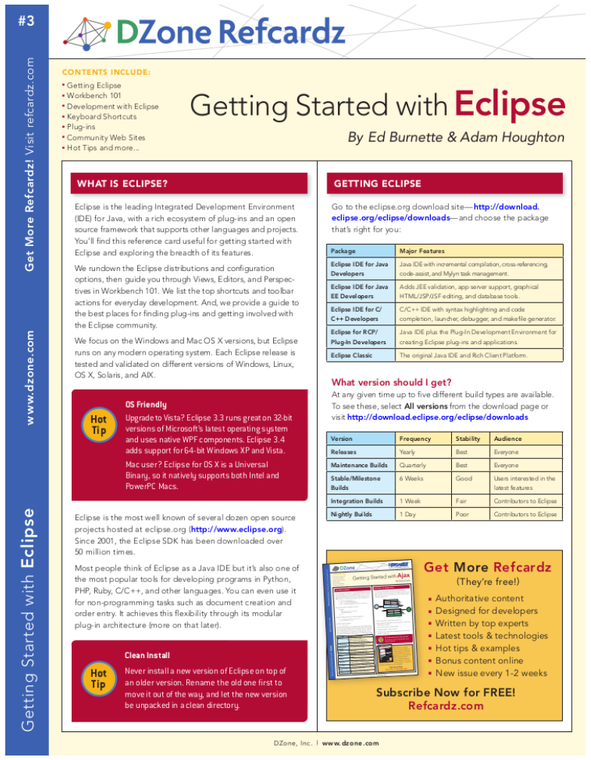
- **Site officiel :** http://www.eclipse.org/
- **Plugins (avec commentaires, notes) :** http://marketplace.eclipse.org/
- Q&A StackOverflow - **"What are the best JVM settings for Eclipse?"** : http://stackoverflow.com/questions/142357/what-are-the-best-jvm-settings-for-eclipse
- Q&A StackOverflow - **"Useful Eclipse Java Code Templates"** : http://stackoverflow.com/questions/1028858/useful-eclipse-java-code-templates
- Q&A StackOverflow - **"Hidden features of Eclipse"** : http://stackoverflow.com/questions/54886/hidden-features-of-eclipse
- Q&A StackOverflow - **"Do you have any recommended plugins for Eclipse?"** : http://stackoverflow.com/questions/2826/do-you-have-any-recommended-plugins-for-eclipse
- FAQ Developpez.com : http://eclipse.developpez.com/faq/



Liens (reference card)

INFO **RefCardz DZone** propose de nombreuses fiches en .pdf pour découvrir en 5 pages une techno (serveurs d'applications, base de données, caches partagés, etc.), en survolant chaque aspect (installation, administration, prise en main, etc.)

Référence Card Eclipse : http://refcardz.dzone.com/refcardz/getting-started-eclipse

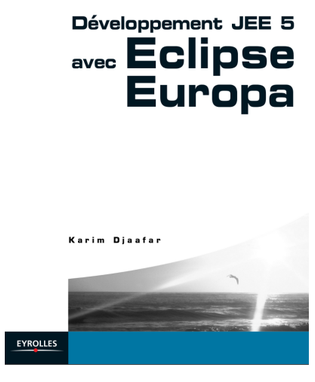


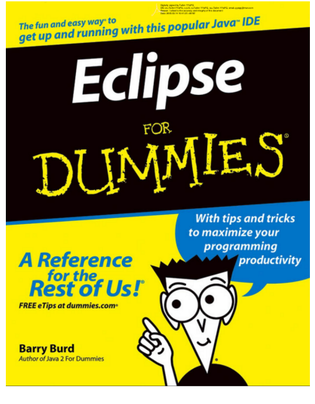
DOWNLOAD

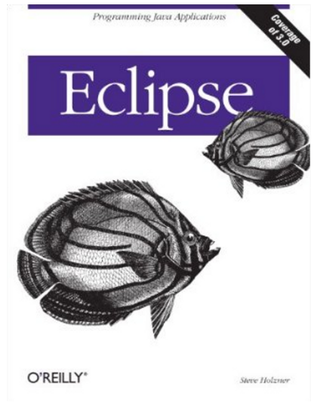
Liens (books)

Peu de livres récents sur le sujet, ni en français, ni en anglais.

- Livre "**Développement JEE5 avec Eclipse Europa**" (2008)
- Livre "**Refactoring des applications Java/J2EE**" (2005) (parce plus de refactoring applicatif que de fonctions Eclipse)
- Livre "**Eclipse for Dummies**" (2005)
- Livre "**Java et Eclipse - Développez une application avec Java et Eclipse**" (réédition 2012)
- Livre "**Eclipse: A Java Developer's Guide**" (réédition 2009)



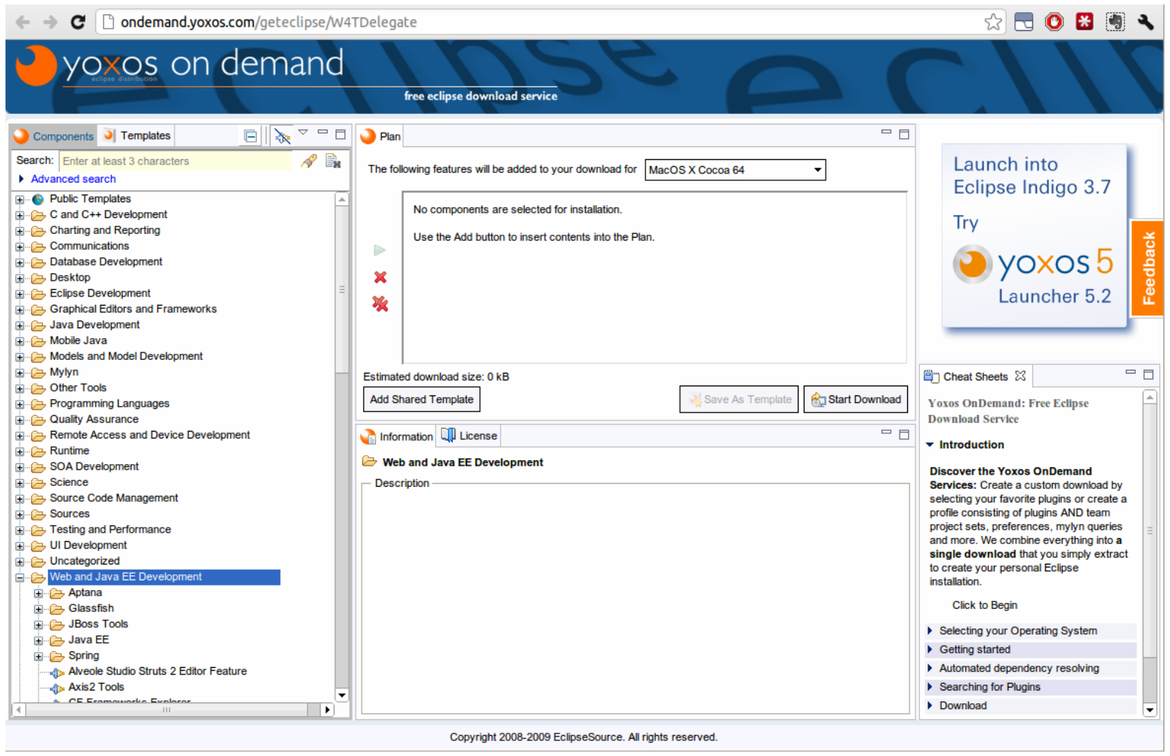




Liens (Eclipse On Demand)

Eclipse On Demand (via Yoxos) : http://ondemand.yoxos.com/geteclipse/W4TDelegate. Permet :

- De gérer les préférences Eclipse (appliquées au démarrage d'Eclipse via le lanceur Yoxos)
- De construire une installation personnalisée en préselectionnant les packages que l'on souhaite. L'outil garantit que les dépendances seront respectées !



Merci de votre attention.

A disposition pour répondre à vos questions.