

# Développons en Java avec Eclipse

**Développons en Java avec Eclipse**

Jean Michel DOUDOUX

# Table des matières

<u>Développons en Java avec Eclipse</u> .....	1
<b>Préambule</b> .....	2
<u>A propos de ce document</u> .....	2
<u>Note de licence</u> .....	3
<u>Marques déposées</u> .....	3
<u>Historique des versions</u> .....	3
<b>Partie 1 : les bases pour l'utilisation d'Eclipse</b> .....	5
<b>1. Introduction</b> .....	6
1.1. Les points forts d'Eclipse.....	6
1.2. Les différentes versions d'Eclipse.....	7
1.3. Les différents sous projets d'Eclipse.....	8
1.4. Callisto.....	9
<b>2. Installation et exécution</b> .....	11
2.1. Installation d'Eclipse sous Windows.....	11
2.1.1. Installation d'Eclipse 1.0.....	11
2.1.2. Installation d'Eclipse 2.0.....	11
2.1.3. Installation des traductions de la version 2.x.....	12
2.1.4. Installation d'Eclipse 3.0.1.....	14
2.1.5. Installation des traductions de la version 3.0.x.....	16
2.1.6. Installation d'Eclipse 3.2.x.....	16
2.1.7. Installation des traductions de la version 3.2.x.....	17
2.2. Installation d'Eclipse sous Linux.....	18
2.2.1. Installation d'Eclipse 1.0 sous Mandrake 8.1.....	18
2.2.2. Installation Eclipse 2.1 sous Mandrake 8.0.....	19
2.2.3. Installation Eclipse 3.0.x sous Mandrake 10.1.....	20
2.2.3.1. Installation par et pour un seul utilisateur.....	21
2.2.3.2. Installation par root pour plusieurs utilisateurs.....	23
2.2.3.3. Installation des traductions.....	24
2.3. Les paramètres.....	25
2.4. La résolution de problèmes.....	26
2.4.1. Un plug-in installé n'est pas visible.....	26
2.4.2. L'espace de travail est déjà utilisé.....	26
<b>3. Le plan de travail (Workbench)</b> .....	27
3.1. Les perspectives.....	27
3.2. Les vues et les éditeurs.....	30
3.2.1. Les éditeurs.....	30
3.2.2. Les vues.....	33
3.3. Les assistants.....	35
3.4. Organiser les composants de la perspective.....	36
3.5. Fermer le plan de travail.....	37
3.6. Exécution de traitements en arrière plan.....	37
<b>4. L'espace de travail (Workspace)</b> .....	39
4.1. La perspective "Ressource".....	39
4.1.1. La vue "Navigateur".....	39
4.2. La création de nouvelles entités.....	41
4.2.1. La création d'un projet.....	41
4.2.2. La création d'un répertoire.....	42
4.2.3. La création d'un fichier.....	43
4.3. La duplication d'un élément.....	43

# Table des matières

<b>4. L'espace de travail (Workspace)</b>	
4.4. Le déplacement d'un élément.....	44
4.5. Renommer un élément.....	44
4.6. La suppression d'un élément.....	45
4.7. L'importation.....	45
4.8. L'exportation.....	47
4.9. Lier des ressources.....	50
4.10. L'option « Fermer les projets non associés ».....	52
4.11. Importer une copie d'un projet.....	52
<b>5. Les fonctions pratiques du plan de travail.....</b>	<b>54</b>
5.1. La fonction de recherche.....	54
5.1.1. La recherche dans les fichiers.....	54
5.1.2. L'exploitation des résultats de recherche.....	56
5.2. La liste des tâches.....	58
5.2.1. La création d'une tâche.....	58
5.2.2. La création d'une tâche associée à un élément.....	59
5.2.3. La suppression d'une tâche associée à un élément.....	60
5.3. La liste des signets.....	60
5.3.1. La création d'un signet.....	60
5.3.2. La suppression d'un signet.....	61
5.4. La comparaison d'éléments.....	61
5.5. La vue « Explorateur de projets ».....	62
<b>6. L'aide dans Eclipse.....</b>	<b>64</b>
6.1. L'aide en ligne.....	64
6.2. L'aide Javadoc.....	71
6.3. Le plug-in Java docs de Crionics.....	72
<b>Partie 2 : le développement en Java.....</b>	<b>74</b>
<b>7. Le Java Development Tooling (JDT).....</b>	<b>75</b>
7.1. Les projets de type Java.....	75
7.1.1. La création d'un nouveau projet Java.....	75
7.1.2. Les paramètres d'un projet Java.....	82
7.2. La création d'entité.....	85
7.2.1. La création de packages.....	85
7.2.2. La création de classes.....	85
7.2.3. La création d'interfaces.....	87
7.2.4. La création de classe et de package par copier/coller du code source dans l'explorateur de package.....	87
7.3. Le support de Java 5.0.....	88
7.3.1. La création d'une énumération.....	89
7.3.2. La création d'une annotation.....	91
7.4. Le support de Java 6.0 par le compilateur.....	91
7.5. Les vues du JDT.....	92
7.5.1. La vue "Packages".....	92
7.5.2. La vue "Hiérarchie".....	94
7.5.3. La vue "Javadoc".....	96
7.5.4. La vue "Déclaration".....	96
7.5.5. La vue Erreurs.....	96
7.5.6. La vue Historique.....	100
7.6. L'éditeur de code.....	100
7.6.1. Utilisation de l'éditeur de code.....	101
7.6.2. Compléction de code.....	102

# Table des matières

## **7. Le Java Development Tooling (JDT)**

<u>    7.6.3. Affichage des paramètres sous la forme d'une bulle d'aide</u> .....	106
<u>    7.6.4. Hiérarchie de type dans une bulle d'aide</u> .....	107
<u>    7.6.5. Affichage des membres dans une bulle d'aide</u> .....	107
<u>    7.6.6. L'éditeur et la vue Structure</u> .....	107
<u>    7.6.7. La coloration syntaxique</u> .....	108
<u>    7.6.8. Utilisation des modèles</u> .....	109
<u>    7.6.9. La gestion des importations</u> .....	110
<u>    7.6.10. Le tri des membres</u> .....	112
<u>    7.6.11. La génération de getter et setter</u> .....	113
<u>    7.6.12. La génération des méthodes hashCode() et equals()</u> .....	114
<u>    7.6.13. Formater le code</u> .....	116
<u>    7.6.14. Mise en commentaire d'une portion de code</u> .....	118
<u>    7.6.15. Protéger une portion de code avec un bloc try/catch</u> .....	119
<u>    7.6.16. La fonctionnalité « Entourer avec »</u> .....	120
<u>    7.6.17. Les avertissements</u> .....	122
<u>        7.6.17.1. Détection de l'assignation d'une valeur à un paramètre</u> .....	122
<u>        7.6.17.2. Détection d'un oubli potentiel de l'instruction break</u> .....	123
<u>    7.6.18. Les erreurs</u> .....	123
<u>    7.6.19. Masquer certaines portions de code</u> .....	125
<u>    7.6.20. Le mode « Insertion Avancée »</u> .....	127
<u>    7.6.21. Marquer les occurrences trouvées</u> .....	127
<u>    7.6.22. Marquer les points de sortie d'une méthode</u> .....	128
<u>    7.6.23. Marquer les emplacements où une exception est levée</u> .....	128
<u>    7.6.24. L'assistant de correction rapide</u> .....	129
<u>    7.6.25. La génération de constructeur</u> .....	130
<u>    7.6.26. Raccourci clavier pour avoir accès aux fonctions de modification du code source</u> .....	131
<u>    7.6.27. Les raccourcis clavier des éditeurs</u> .....	131
<u>    7.6.28. La gestion des modèles de commentaires</u> .....	132
<u>    7.6.29. Le support du tag @category</u> .....	134
<u>7.7. Exécution d'une application</u> .....	136
<u>7.8. Génération de la documentation Javadoc</u> .....	140
<u>7.9. Définition du JRE à utiliser</u> .....	143
<u>7.10. Les environnements d'exécution</u> .....	143
<u>7.11. Utilisation de l'historique local</u> .....	146
<u>7.12. Externaliser les chaînes</u> .....	147
<u>        7.12.1. Rechercher les chaînes externalisées erronées</u> .....	150
<u>        7.12.2. Recherche de l'utilisation d'une clé</u> .....	151
<u>7.13. Ouverture d'un type</u> .....	152
<u>7.14. Utilisation du scrapbook</u> .....	153
<u>7.15. Le développement d'applets</u> .....	158
<u>7.16. Le nettoyage du code</u> .....	160

## **8. Déboguer du code Java**.....**163**

<u>    8.1. La perspective "Débogage"</u> .....	163
<u>    8.2. Les vues spécifiques au débogage</u> .....	163
<u>        8.2.1. La vue "Débogage"</u> .....	164
<u>        8.2.2. La vue "Variables"</u> .....	164
<u>        8.2.3. La vue "Points d'arrêts"</u> .....	165
<u>        8.2.4. La vue "Expressions"</u> .....	168
<u>        8.2.5. La vue "Affichage"</u> .....	169
<u>    8.3. Mise en oeuvre du débogueur</u> .....	169
<u>        8.3.1. Mettre en place un point d'arrêt</u> .....	169
<u>        8.3.2. Exécution dans le débogueur</u> .....	170
<u>        8.3.3. Exportation / Importation des points d'arrêt</u> .....	171

# Table des matières

<b>9. Le refactoring.....</b>	<b>173</b>
9.1. Extraction d'une méthode.....	175
9.2. Intégrer.....	177
9.3. Renommer.....	178
9.4. Déplacer.....	184
9.5. Changer la signature de la méthode.....	186
9.6. Convertir une classe anonyme en classe imbriquée.....	188
9.7. Convertir un type imbriqué au niveau supérieur.....	189
9.8. Extraire.....	190
9.9. Transferer.....	191
9.10. Extraire une interface.....	192
9.11. Utiliser le supertype si possible.....	193
9.12. Convertir la variable locale en zone.....	194
9.13. Encapsuler la zone.....	196
9.14. Extraire la variable locale.....	197
9.15. Extraire une constante.....	198
9.16. Généraliser le type.....	199
9.17. Introduire une fabrique.....	200
9.18. Introduire un paramètre.....	201
9.19. Introduction de l'adressage indirect.....	201
9.20. Extraire une super classe.....	203
9.21. Annuler ou refaire une opération de refactoring.....	206
9.22. L'historique de restructuration.....	207
<b>10. Ant et Eclipse.....</b>	<b>208</b>
10.1. Structure du projet.....	208
10.2. Création du fichier build.xml.....	210
10.3. Exécuter Ant.....	212
10.4. Les paramètres.....	213
10.5. Résolution des problèmes.....	213
10.5.1. Utilisation de caractères accentués.....	214
10.5.2. Impossible de lancer la tâche javadoc.....	214
10.5.3. Impossible d'utiliser la tâche JUnit.....	214
10.6. Un exemple complet.....	215
10.7. L'éditeur de fichiers Ant.....	216
10.8. Le débogage de fichiers Ant.....	219
<b>11. JUnit et Eclipse.....</b>	<b>220</b>
11.1. Paramétrage de l'environnement.....	220
11.2. Ecriture des cas de tests.....	221
11.3. Exécution des cas de tests.....	223
11.4. Le support de JUnit 4.....	224
<b>Partie 3 : les fonctions avancées d'Eclipse.....</b>	<b>229</b>
<b>12. La gestion de la plate-forme.....</b>	<b>230</b>
12.1. Informations sur les plug-ins installés.....	230
12.2. Installation du plug-in JadClipse sous Eclipse 1.0.....	231
12.3. La mise à jour des plug-ins avec Eclipse 2.0.....	232
12.3.1. La perspective « Installation / Mise à jour ».....	232
12.3.2. Recherche et installation des mises à jour.....	233
12.3.3. Installation d'un nouveau plug-in.....	235
12.3.4. Sauvegarde et restauration d'une configuration.....	238
12.3.5. Résolution des problèmes de dépendances.....	240
12.4. La mise à jour des plug-ins avec Eclipse 3.0.....	242

# Table des matières

<b>12. La gestion de la plate-forme</b>	
12.4.1. Recherche et installation de plug-ins.....	242
12.4.2. La configuration du produit.....	246
12.4.3. Mises à jour automatiques.....	249
12.4.4. Mise à jour de la plate-forme via des miroirs.....	250
<b>13. CVS 2.0 et Eclipse 2.1.....</b>	<b>254</b>
13.1. Installation de cvstn.....	254
13.2. La perspective CVS.....	260
13.2.1. La création d'un emplacement vers un référentiel.....	260
13.2.2. Partager un projet.....	261
13.2.3. Voir le projet dans la perspective CVS.....	263
13.3. L'utilisation des révisions.....	264
13.3.1. Créer une révision.....	264
13.3.2. Gestion des révisions.....	265
13.4. La gestion des versions d'un projet.....	265
13.4.1. La création d'une version d'un projet.....	265
13.5. Obtenir une version dans le workspace.....	266
<b>14. CVS 2.5 et Eclipse 3.0.....</b>	<b>267</b>
14.1. Installation et configuration de CVS 2.5.....	267
14.2. La perspective « Exploration du référentiel CVS ».....	270
14.3. Ajouter un projet au référentiel.....	274
14.4. Reporter des modifications dans le référentiel.....	277
14.5. Déconnecter un projet du référentiel.....	279
14.6. La perspective Synchronisation de l'équipe.....	279
14.7. Importation d'un projet à partir de CVS.....	281
14.8. Versionner un projet.....	284
<b>15. Subversion et Eclipse.....</b>	<b>285</b>
15.1. Installation de subversion sous Windows.....	285
15.2. Le plug-in Subclipse.....	292
15.2.1. Installation de Subclipse.....	293
15.2.2. Paramétrage du plug-in.....	295
15.2.3. Utilisation du plug-in.....	295
15.2.3.1. La connexion à un repository.....	296
15.2.3.2. Ajouter un projet au repository.....	298
15.2.3.3. Synchroniser l'espace de travail et le référentiel.....	303
15.2.3.4. Checkout d'un projet.....	304
15.3. Le plug in Subversive.....	305
<b>Partie 4 : le développement avancé avec Java.....</b>	<b>306</b>
<b>16. Des plug-ins pour le développement avec Java.....</b>	<b>307</b>
16.1. Le plug-in Jalopy.....	307
16.2. Log4E.....	309
16.2.1. Installation.....	309
16.2.2. Les paramètres.....	309
16.2.3. Utilisation.....	310
<b>17. Le développement d'interfaces graphiques.....</b>	<b>313</b>
17.1. Eclipse et SWT.....	313
17.1.1. Configurer Eclipse pour développer des applications SWT.....	313
17.1.2. Un exemple très simple.....	315
17.2. Le plug-in Eclipse V4all.....	316

# Table des matières

<b>17. Le développement d'interfaces graphiques</b>	
17.2.1. Installation	316
17.2.2. Utilisation	316
17.2.3. Un exemple simple	318
17.3. Eclipse VE	319
17.3.1. Installation	320
17.3.2. Mise en oeuvre et présentation rapide	320
17.3.3. L'ajout d'éléments	324
17.3.4. La gestion des événements	329
17.3.5. Exécution	330
<b>18. Le plug-in TPTP (Test &amp; Performance Tools Platform)</b>	332
18.1. Installation	332
18.2. Profiler une application	333
18.3. Profiler une application Web	342
18.4. Les outils de tests	342
18.4.1. Les tests avec JUnit	342
<b>19. Hibernate et Eclipse</b>	346
19.1. Le plug-in Hibernate Synchronizer	346
19.1.1. Installation	346
19.1.2. La base de données utilisée	347
19.1.3. Création des fichiers de mapping	348
19.1.4. Génération des classes Java	353
19.1.5. Création du fichier de configuration	353
19.1.6. La mise en oeuvre des classes générées	355
<b>Partie 5 : le développement d'applications d'entreprise</b>	359
<b>20. Le développement avec J2EE</b>	360
20.1. Le plug-in Tomcat de Sysdeo	360
20.1.1. Installation et paramétrage de la version 2.2.1	360
20.1.2. Installation et paramétrage de la version 3.0	363
20.1.3. Lancement et arrêt de Tomcat	365
20.1.4. La création d'un projet Tomcat	366
20.2. Le plug in WTP (Eclipse Web Tools Platform)	367
20.2.1. Installation de la version 1.0M3	368
20.2.2. Configuration de XDoclet	371
20.2.3. Configuration d'un serveur	371
20.3. Lomboz	377
20.3.1. Installation et configuration	377
20.3.2. Création d'un nouveau projet	379
<b>21. XML et Eclipse</b>	385
21.1. JAXB et Eclipse	385
21.1.1. Créer et configurer une tâche d'exécution pour JAXB	386
21.1.2. Exécuter Ant en tant qu'outil externe	391
21.2. Le plug-in WTP pour utiliser XML	393
21.2.1. Créer un nouveau document XML	394
21.2.2. La création d'une DTD	396
21.2.3. La création d'un schéma	397
21.2.4. Les préférences	399
21.2.5. Création d'un document XML à partir d'une DTD	401
21.2.6. La validation des documents	404

# Table des matières

<b>22. Le développement d'applications web.....</b>	<b>406</b>
<b>22.1. Le développement d'applications web avec WTP 1.0.....</b>	<b>406</b>
<b>22.1.1. La création d'un nouveau projet.....</b>	<b>406</b>
<b>22.1.2. La création d'une servlet.....</b>	<b>408</b>
<b>22.1.3. La création d'une JSP.....</b>	<b>413</b>
<b>22.1.4. L'exécution de l'application.....</b>	<b>415</b>
<b>22.2. Le développement d'applications web avec le plug-in Lomboz 2.1.....</b>	<b>420</b>
<b>22.2.1. Création d'une webapp.....</b>	<b>420</b>
<b>22.2.2. Ajouter un fichier HTML à une webapp.....</b>	<b>420</b>
<b>22.2.3. Ajouter une JSP à une webapp.....</b>	<b>422</b>
<b>22.2.4. Ajouter une servlet à une webapp.....</b>	<b>424</b>
<b>22.2.5. Tester une webapp.....</b>	<b>427</b>
<b>23. Struts et Eclipse.....</b>	<b>431</b>
<b>23.1. Le plug-in Easy Struts.....</b>	<b>431</b>
<b>23.1.1. Installation et configuration d'Easy Struts.....</b>	<b>432</b>
<b>23.1.2. L'exécution de l'application.....</b>	<b>444</b>
<b>23.1.3. La modification du fichier struts-config.xml.....</b>	<b>445</b>
<b>24. Java Server Faces et Eclipse.....</b>	<b>447</b>
<b>24.1. Utilisation de JSF sans plug-in dédié.....</b>	<b>447</b>
<b>24.1.1. Crédit du projet.....</b>	<b>447</b>
<b>24.1.2. Crédit des éléments qui composent l'application.....</b>	<b>448</b>
<b>24.1.3. Exécution de l'application.....</b>	<b>451</b>
<b>25. EJB et Eclipse.....</b>	<b>453</b>
<b>25.1. Le développement d'EJB avec le plug-in WTP 1.5.....</b>	<b>453</b>
<b>25.1.1. La création d'un projet de type EJB.....</b>	<b>453</b>
<b>25.1.2. La création d'un EJB de type SessionBean stateless.....</b>	<b>458</b>
<b>25.1.3. La création et l'exécution d'un client de test.....</b>	<b>464</b>
<b>25.2. Le développement d'EJB avec le plug-in Lomboz 2.1.....</b>	<b>467</b>
<b>25.2.1. La création d'un EJB de type Session.....</b>	<b>467</b>
<b>25.2.2. Ajouter une méthode à un EJB.....</b>	<b>468</b>
<b>25.2.3. La génération des fichiers des EJB.....</b>	<b>470</b>
<b>25.2.4. Déploiement du module EJB.....</b>	<b>471</b>
<b>26. Les services web et Eclipse.....</b>	<b>472</b>
<b>26.1. La mise en oeuvre manuelle d'Axis.....</b>	<b>472</b>
<b>26.1.1. La configuration de l'environnement.....</b>	<b>472</b>
<b>26.1.2. La création d'un projet de type web.....</b>	<b>476</b>
<b>26.1.3. La configuration du projet.....</b>	<b>479</b>
<b>26.1.4. La création d'une nouvelle classe.....</b>	<b>483</b>
<b>26.1.5. Le lancement de l'application web.....</b>	<b>484</b>
<b>26.1.6. Le déploiement du service dans Axis.....</b>	<b>485</b>
<b>26.1.7. La vérification du déploiement du service web.....</b>	<b>486</b>
<b>26.1.8. La modification du type du service web.....</b>	<b>488</b>
<b>26.2. La consommation du service Web en .Net.....</b>	<b>489</b>
<b>26.3. Le développement de services web avec WTP 1.0.....</b>	<b>492</b>
<b>26.3.1. Convertir une classe en service web.....</b>	<b>492</b>
<b>26.3.2. Les différentes fonctionnalités sur les services web.....</b>	<b>502</b>
<b>26.3.3. L'explorateur de services web.....</b>	<b>507</b>
<b>26.3.4. L'éditeur de fichier .wsdl.....</b>	<b>513</b>
<b>26.4. La mise en oeuvre d'Axis avec le WTP 1.5.....</b>	<b>514</b>
<b>26.4.1. La création du service web.....</b>	<b>514</b>
<b>26.4.2. Le test du service web avec l'explorateur de services web.....</b>	<b>517</b>

# Table des matières

<b>26. Les services web et Eclipse</b>	
26.4.3. La création d'un client de test.....	518
<b>27. JPA et Eclipse.....</b>	<b>522</b>
27.1. Dali.....	522
27.1.1. La création et la configuration d'un nouveau projet.....	522
27.1.2. Ajouter une entité persistante.....	528
27.1.3. Exemple d'utilisation de l'API.....	532
27.1.4. Connexion à la base de données.....	533
27.1.5. La génération des entités à partir de la base de données.....	535
27.1.6. L'éditeur du fichier persistence.xml.....	536
27.1.7. La synchronisation des classes.....	537
27.1.8. Transformer une classe existante en entité.....	537
27.1.9. La création d'une association 1-1.....	539
<b>Partie 6 : le développement d'applications mobiles.....</b>	<b>545</b>
<b>28. Le développement avec J2ME.....</b>	<b>546</b>
28.1. EclipseME.....	546
28.1.1. Installation.....	546
28.1.2. Les préférences du plug-in.....	547
28.1.3. Création d'un premier exemple.....	551
28.1.4. Exécution de l'application.....	554
28.1.5. Déboguer l'application.....	556
28.1.6. Modification des propriétés du descripteur d'application.....	557
28.1.7. Packager l'application.....	558
<b>Partie 7 : d'autres plug-ins.....</b>	<b>562</b>
<b>29. Le plug-in CDT pour le développement en C / C++.....</b>	<b>563</b>
29.1. Installation du CDT.....	563
29.2. Création d'un premier projet.....	567
29.3. Installation de MinGW.....	570
29.4. Première configuration et exécution.....	572
29.5. Utilisation du CDT.....	573
<b>30. Le plug-in EclipseUML.....</b>	<b>574</b>
30.1. Installation.....	574
30.2. Les préférences.....	575
30.3. Mise en oeuvre du plug-in.....	575
30.3.1. Création d'un diagramme de cas d'utilisation.....	576
30.3.2. Création d'un diagramme de classe.....	579
<b>31. Les bases de données et Eclipse.....</b>	<b>584</b>
31.1. Quantum.....	584
31.1.1. Installation et configuration.....	584
31.1.2. Afficher le contenu d'une table.....	586
31.1.3. Exécution d'une requête.....	588
31.2. JFaceDbc.....	590
31.2.1. Installation et configuration.....	590
31.2.2. La mise à jour des données d'une table.....	595
31.2.3. L'éditeur SQL.....	596
31.2.4. Vue graphique d'une base de données.....	597
31.3. DBEdit.....	600
31.3.1. Installation et configuration.....	600

# Table des matières

<b><u>31. Les bases de données et Eclipse</u></b>	
<u>31.3.2. La vue « Tables »</u>	602
<u>31.3.3. L'éditeur « Table »</u>	605
<u>31.3.4. La vue Log</u>	606
<u>31.3.5. L'éditeur Instant SQL</u>	606
<u>31.4. Clay Database Modelling</u>	608
<u>31.4.1. Installation et configuration</u>	608
<u>31.4.2. Mise en oeuvre</u>	609
<u>31.4.3. Rétro-conception d'un modèle</u>	615
<u>31.4.4. Génération du DDL</u>	617
<b><u>Partie 8 : Annexes</u></b>	<b>619</b>
<b><u>28. Annexes</u></b>	<b>620</b>
<u>Annexe A : GNU Free Documentation License</u>	620
<u>Annexe B : Webographie</u>	624

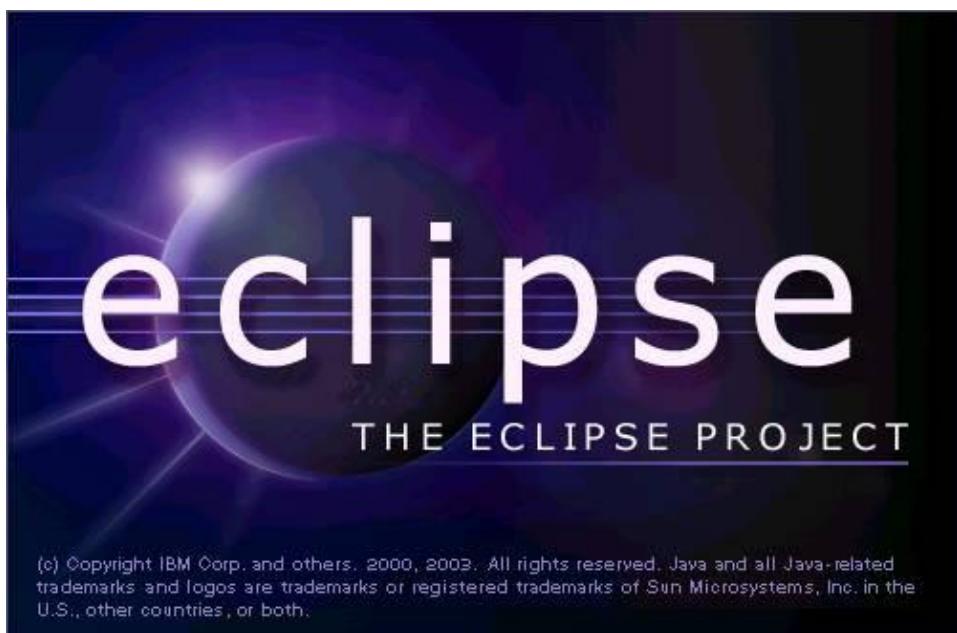
Développons en Java avec Eclipse

# Développons en Java avec Eclipse

Version 0.80.1

du 15/12/2008

par Jean Michel DOUDOUX



# Préambule

## A propos de ce document

Ce document fait suite à mon premier didacticiel "Développons en Java". C'est un didacticiel qui se propose de fournir des informations pratiques sur la mise en oeuvre et l'utilisation d'Eclipse et de quelques un de ces nombreux plug-ins.

Celui-ci est composé de sept grandes parties :

1. les bases pour l'utilisation d'Eclipse
2. le développement en Java
3. les fonctions avancées d'Eclipse
4. le développement avancé avec Java
5. le développement d'applications d'entreprise
6. le développement d'applications mobiles
7. d'autres plug-ins

Chacune de ces parties est composée de plusieurs chapitres dont voici la liste complète :

- ◆ Préambule
- ◆ Introduction
- ◆ Installation et exécution d'Eclipse
- ◆ Le plan de travail (Workbench)
- ◆ L'espace de travail (Workspace)
- ◆ Les fonctions pratiques du workbench d'Eclipse
- ◆ L'aide dans Eclipse
- ◆ Le Java development tooling (JDT) d'Eclipse
- ◆ Déboguer du code java
- ◆ Le refactoring
- ◆ Ant et Eclipse
- ◆ JUnit et Eclipse
- ◆ La gestion de la plate-forme
- ◆ CVS 2.0 et Eclipse 2.1
- ◆ CVS 2.5 et Eclipse 3.0
- ◆ Subversion et Eclipse
- ◆ Des plug-ins pour le développement avec Java
- ◆ Le développement d'interfaces graphiques
- ◆ Le plug-in TPTP (Test & Performance Tools Platform)
- ◆ Hibernate et Eclipse
- ◆ Le développement avec J2EE
- ◆ XML et Eclipse
- ◆ Le développement d'applications web
- ◆ Struts et Eclipse
- ◆ Java Server Faces et Eclipse
- ◆ EJB et Eclipse
- ◆ Les services web et Eclipse
- ◆ JPA et Eclipse
- ◆ Le développement avec J2ME
- ◆ Le plug-in CDT pour le développement en C / C++
- ◆ Le plug-in Eclipse et UML
- ◆ Les bases de données et Eclipse

Les sections qui concernent des plug-ins n'ont pas pour vocation d'être une documentation complète sur l'utilisation de ces plug-ins mais simplement de fournir les bases pour les installer et mettre en oeuvre un minimum de fonctionnalités qu'ils proposent.

Je suis ouvert à toutes réactions ou suggestions concernant ce document notamment le signalement des erreurs, les points à éclaircir, les sujets à ajouter, etc. ... N'hésitez pas à me contacter : [jean-michel.doudoux@wanadoo.fr](mailto:jean-michel.doudoux@wanadoo.fr)

Ce document est disponible aux formats HTML et PDF à l'adresse suivante :  
<http://perso.wanadoo.fr/jm.doudoux/java/>

Ce manuel est fourni en l'état, sans aucune garantie. L'auteur ne peut être tenu pour responsable des éventuels dommages causés par l'utilisation des informations fournies dans ce document.

La version pdf de ce document est réalisée grâce à l'outil HTMLDOC 1.8.23 de la société Easy Software Products. Cet excellent outil freeware peut être téléchargé à l'adresse : <http://www.easysw.com>

## Note de licence

Copyright (C) 2003-2008 DOUDOUX Jean-Michel

Vous pouvez copier, redistribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU, Version 1.1 ou toute autre version ultérieure publiée par la Free Software Foundation; les Sections Invariantes étant constituées du chapitre Préambule, aucun Texte de Première de Couverture, et aucun Texte de Quatrième de Couverture. Une copie de la licence est incluse dans la section [GNU FreeDocumentation Licence](#).

La version la plus récente de cette licence est disponible à l'adresse : [GNU Free Documentation Licence](#).

## Marques déposées

Sun, Sun Microsystems, le logo Sun et Java sont des marques déposées de Sun Microsystems Inc.

Les autres marques et les noms de produits cités dans ce document sont la propriété de leur éditeur respectif.

## Historique des versions

Version	Date	Evolutions
0.10	08/04/2003	1ere version diffusée sur le web.
0.20	13/07/2003	Ajout des chapitres Junit, Ant, Aide, Déboguer du code Java  Ajout des sections : importation, exportation, génération javadoc, informations sur les plug-ins, le plug-in Jalopy  Compléments ajoutés au chapitre JDT  Application d'une feuille de styles CSS pour la version HTML

		Corrections et ajouts divers
0.30	04/01/2004	<p>Ajouts dans les chapitres "JDT", "déboguer du code Java" et "La gestion de la plate-forme"</p> <p>Ajout du chapitre "le refactoring" , "le développement sans java" et "Le développement d'interfaces graphiques"</p> <p>Réduction de la taille des images : réduction de la taille et passage en niveau de gris pour la version PDF</p> <p>Corrections et ajouts divers</p>
0.40	24/05/2004	<p>Ajout des chapitres "Eclipse et les bases de données", "le développement J2EE" et "JAXB et Eclipse"</p> <p>Ajouts dans le chapitre "Refactoring" et dans la section "Eclipse VE"</p> <p>Corrections et ajouts divers</p>
0.50 0.50.1	07/12/2004 11/12/2004	<p>Prise en compte d'Eclipse 3.0 dans différents chapitres</p> <p>Ajout des chapitres : "Eclipse et UML", "Java Server Faces et Eclipse" et "Struts et Eclipse"</p> <p>Ajout d'une section consacrée au plug-in "Log4E"</p> <p>Remise en page complète pour éviter autant que possible les blancs en bas de pages. Découpage en 5 parties</p> <p>Corrections et ajouts divers</p>
0.60	27/06/2005	<p>Ajout des chapitres "Eclipse Webtools Platform", "Eclipse et Hibernate", "Eclipse et J2ME" et "CVS 2.5 et Eclipse 3.0"</p> <p>Corrections et ajouts divers</p>
0.70	12/04/2006	<p>Ajout des chapitres : le plug-in TPTP, le développement de services web, Eclipse WTP (captures en français)</p> <p>Modification des chapitres : Visual Editor v1.0,</p> <p>Corrections et ajouts divers</p>
0.80 0.80.1	26/01/2007 15/12/2008	<p>Migration vers le site <a href="http://www.jmdoudoux.fr/java/">http://www.jmdoudoux.fr/java/</a></p> <p>Ajout des chapitres : Subversion et Eclipse, JPA et Eclipse (Dali)</p> <p>Ajout de la section : développement d'EJB avec WTP 1.5</p> <p>Modification de nombreux chapitres : Eclipse 3.2, réorganisation des parties, chapitres et section</p> <p>Corrections et ajouts divers</p>

## **Partie 1 : les bases pour l'utilisation d'Eclipse**

# **Partie 1 : les bases pour l'utilisation d'Eclipse**

Cette première partie est chargée de présenter les bases de l'utilisation d'Eclipse.

Elle comporte les chapitres suivants :

- ◆ Introduction : présentation générale d'Eclipse
- ◆ Installation et exécution : détaille l'installation et l'exécution des trois versions majeurs d'Eclipse sous Windows et Linux
- ◆ Le plan de travail (Workbench) : présente le plan de travail qui fournit les éléments de l'ergonomie notamment au travers des perspectives, des vues et des éditeurs
- ◆ L'espace de travail (Workspace) : présente l'espace de travail qui stocke les projets et leur contenu et détaille des opérations de base sur les élément de l'espace de travail
- ◆ Les fonctions pratiques du plan de travail : détaille certaines fonctions avancées du plan de travail : la recherche, la gestion des tâches et des signets et la comparaison d'éléments
- ◆ L'aide dans Eclipse : présente comment obtenir de l'aide lors de l'utilisation d'Eclipse.

# 1. Introduction

# Chapitre 1

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

I.B.M. est à l'origine du développement d'Eclipse qui est d'ailleurs toujours le cœur de son outil Websphere Studio Workbench (WSW), lui même à la base de la famille des derniers outils de développement en Java d'I.B.M. Tout le code d'Eclipse a été donné à la communauté par I.B.M afin de poursuivre son développement.

Eclipse utilise énormément le concept de modules nommés "plug-ins" dans son architecture. D'ailleurs, hormis le noyau de la plate-forme nommé "Runtime", tout le reste de la plate-forme est développé sous la forme de plug-ins. Ce concept permet de fournir un mécanisme pour l'extension de la plate-forme et ainsi fournir la possibilité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse.

Les principaux modules fournis en standard avec Eclipse concernent Java mais des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception avec UML, ... ). Ils sont tous développés en Java soit par le projet Eclipse soit par des tiers commerciaux ou en open source.

Les modules agissent sur des fichiers qui sont inclus dans l'espace de travail (Workspace). L'espace de travail regroupe les projets qui contiennent une arborescence de fichiers.

Bien que développé en Java, les performances à l'exécution d'Eclipse sont très bonnes car il n'utilise pas Swing pour l'interface homme-machine mais un toolkit particulier nommé SWT associé à la bibliothèque JFace. SWT (Standard Widget Toolkit) est développé en Java par IBM en utilisant au maximum les composants natifs fournis par le système d'exploitation sous jacent. JFace utilise SWT et propose une API pour faciliter le développement d'interfaces graphiques.

Eclipse ne peut donc fonctionner que sur les plate-formes pour lesquelles SWT a été porté. Ainsi, Eclipse 1.0 fonctionne uniquement sur les plate-formes Windows 98/NT/2000/XP et Linux.

SWT et JFace sont utilisés par Eclipse pour développer le plan de travail (Workbench) qui organise la structure de la plate-forme et les interactions entre les outils et l'utilisateur. Cette structure repose sur trois concepts : la perspective, la vue et l'éditeur. La perspective regroupe des vues et des éditeurs pour offrir une vision particulière des développements. En standard, Eclipse propose huit perspectives.

Les vues permettent de visualiser et de sélectionner des éléments. Les éditeurs permettent de visualiser et de modifier le contenu d'un élément de l'espace de travail.

## 1.1. Les points forts d'Eclipse

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins
- Plusieurs versions d'un même plug-in peuvent cohabiter sur une même plate-forme.
- Un support multi langage grâce à des plug-ins dédiés : Cobol, C, PHP, C#, ...
- Support de plusieurs plate-formes d'exécution : Windows, Linux, Mac OS X, ...
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT
- Les nombreuses fonctionnalités de développement proposées par le JDT (refactoring très puissant, complétion de code, nombreux assistants, ...)
- Une ergonomie entièrement configurable qui propose selon les activités à réaliser différentes « perspectives »
- Un historique local des dernières modifications
- La construction incrémentale des projets Java grâce à son propre compilateur qui permet en plus de compiler le code même avec des erreurs, de générer des messages d'erreurs personnalisés, de sélectionner la cible (java 1.3 ou 1.4) et de mettre en œuvre le scrapbook (permet des tests de code à la volée)
- Une exécution des applications dans une JVM dédiée sélectionnable avec possibilité d'utiliser un débogueur complet (points d'arrêts conditionnels, visualiser et modifier des variables, évaluation d'expression dans le contexte d'exécution, changement du code à chaud avec l'utilisation d'une JVM 1.4, ...)
- Propose le nécessaire pour développer de nouveaux plug-ins
- Possibilité d'utiliser des outils open source : CVS, Ant, Junit
- La plate-forme est entièrement internationalisée dans une dizaine de langues sous la forme d'un plug-in téléchargeable séparément
- Le gestionnaire de mise à jour permet de télécharger de nouveaux plug-ins ou nouvelles versions d'un plug-in déjà installées à partir de sites web dédiés (Eclipse 2.0).
- ...

## 1.2. Les différentes versions d'Eclipse

Pour chaque version d'Eclipse possède un nombre plus ou moins important de types nommés "build" :

- "Nightly Builds" : version en cours de développement créée de façon journalière contenant les modifications de la journée.
- "Integration Builds" : assemblage des sous-projets pour la réalisation de tests
- "Stable Builds" : version testée
- "Releases" : version diffusée et "fiable"

Il existe plusieurs versions de type "Release" d'Eclipse :

Date	Version
Septembre 2006	3.2.1
Juin 2006 <a href="#">La liste des nouvelles fonctionnalités de la version 3.2 sur le site d'Eclipse</a>	3.2
Janvier 2006	3.1.2
Septembre 2005	3.1.1
Juin 2005 <a href="#">La liste des nouvelles fonctionnalités de la version 3.1 sur le site d'Eclipse</a>	3.1
Mars 2005	3.0.2
Septembre 2004	3.0.1

Juin 2004 <a href="#">La liste des nouvelles fonctionnalités de la version 3.0 sur le site d'Eclipse</a>	3.0
Mars 2004	2.1.3
Novembre 2003	2.1.2
Juillet 2003	2.1.1
Avril 2003 <a href="#">La liste des nouvelles fonctionnalités de la version 2.1 sur le site d'Eclipse</a>	2.1
Novembre 2002	2.0.2
Septembre 2002	2.0.1
Juillet 2002	2.0
Novembre 2001	1.0

Ce document couvre essentiellement les versions 2.x et 3.x d'Eclipse. Différents pictogrammes sont utilisés pour signaler des fonctionnalités apportées par une version particulière d'Eclipse :



2.0 : pour la version 2.0.x d'Eclipse



2.1 : pour la version 2.1.x d'Eclipse



3.0 : pour la version 3.0.x d'Eclipse



3.1 : pour la version 3.1.x d'Eclipse



3.2 : pour la version 3.2.x d'Eclipse

Par défaut, les fonctionnalités décrites le sont pour la version 2.x d'Eclipse.

### 1.3. Les différents sous projets d'Eclipse

Le projet Eclipse est divisé en plusieurs sous projets.

Projet	Description
Eclipse	ce projet développe l'architecture et la structure de la plate-forme Eclipse.
Eclipse Tools	ce projet développe ou intègre des outils à la plate-forme pour permettre à des tiers d'enrichir la plate-forme. Il possède plusieurs sous projets tel que CDT (plug-in pour le développement en C/C++), AspectJ (AOP), GEF (Graphical Editing Framework ), PHP (plug-in pour le développement en PHP), Cobol (plug-in pour le développement en Cobol), VE (Visual Editor) pour la création d'IHM.
Eclipse Technology	ce projet, divisé en trois catégories, propose d'effectuer

	des recherches sur des évolutions de la plate-forme et des technologies qu'elle met en oeuvre.
Web Tools Platform (WTP)	ce projet a pour but d'enrichir la plate-forme enfin de proposer un framework et des services pour la création d'outils de développement d'applications web. Il est composé de plusieurs sous projets : WST (Web Standard Tools) , JST (J2EE Standard Tools) , ATF (Ajax Toolkit Framework), Dali (mapping avec JPA) et JSF (Java Server Faces)
Test and Performance Tools Platform (TPTP)	ce projet a pour but de développer une plate-forme servant de support à la création d'outils de tests et d'analyses
Business Intelligence and Reporting Tools (BIRT)	ce projet a pour but de développer une plate-forme facilitant l'intégration de générateur d'états. Il est composé de 4 sous projets : ERD (Eclipse Report Designer), WRD (Web based Report Designer), ERE (Eclipse Report Engine) et ECE (Eclipse Charting Engine).
Eclipse Modeling	ce projet contient plusieurs sous projet dont EMF (Eclipse Modeling Framework) et UML2 pour une implémentation d'UML reposant sur EMF
Data Tools Platform (DTP)	ce projet a pour but de manipuler des sources de données (bases de données relationnelles)
Device Software Development Platform	ce projet a pour but de créer des plug-ins pour faciliter le développement d'applications sur appareils mobiles
Eclipse SOA Tools Platform	ce projet a pour pour but de développer des outils pour faciliter la mise en d'architecture de type SOA

## 1.4. Callisto

Le but du projet Callisto est de proposer la mise à disposition simultanée d'une nouvelle version de 10 des principaux sous projets du projet Eclipse :

- [Eclipse Platform v3.2](#)
- [Eclipse Modeling Framework \(EMF\) v2.2](#)
- [Graphical Editor Framework \(GEF\) v3.2](#)
- [Graphical Modeling Framework \(GMF\) v1.0](#)
- [Visual Editor \(VE\) v1.2](#)
- [Web Tools Platform \(WTP\) v1.5](#)
- [Data Tools Platform \(DTP\) v1.0](#)
- [Test and Performance Tools Platform \(TPTP\) v4.2](#)
- [Business Intelligence and Reporting Tools \(BIRT\) v2.1](#)
- [C/C++ IDE \(CDT\)](#)

Callisto ne propose pas de rassembler les différents sous projets mais simplement de coordonner la diffusion d'une nouvelle version de chacun des sous projets qui possède par ailleurs leur propre cycle de vie.

Callisto facilite la vie de tous les intervenants de la communauté Java pour les développeurs de chaque sous projet, pour tiers qui propose des outils basés sur Eclipse mais aussi pour les utilisateurs d'Eclipse puisque Callisto assure une compatibilité des version des sous projets qui le compose.

Pour pouvoir utiliser calliston, il faut au préalable télécharger et installer la version 3.2 d'Eclipse sur laquelle repose Callisto. Via le gestionnaire de mise à jour, Callisto propose un site de mise à jour dédié qui permet un accès au site de mise à jour des différents sous projet qui le compose : chaque sous projet peut être téléchargé séparément. Le site "Site des recherches Callisto" est proposé par défaut par le gestionnaire de mise à jour : il permet pour chaque sous projet d'accéder aux sites miroir.

Callisto a été diffusé à partir du 30 juin 2006.

Le prochain projet de synchronisation de la sortie de plusieurs projets simultanée porte le nom de code Europa et sera publié fin juin 2007.

## **2. Installation et exécution**

# **Chapitre 2**

Eclipse 1.0 peut être installé sur les plate-formes Windows ( 98ME et SE / NT / 2000 / XP) et Linux.

Eclipse 2.0 peut être installé sur les plate-formes Windows ( 98ME et SE / NT / 2000 / XP), Linux (Motif et GTK), Solaris 8, QNX, AIX 5.1, HP-UX 11.

Eclipse 2.1 peut être installé sur toutes les plate-formes citées précédemment mais aussi sous MAC OS X.

Quel que soit la plate-forme, il faut obligatoirement qu'un JDK 1.3 minimum y soit installé. La version 1.4 est fortement recommandée pour améliorer les performances et pouvoir utiliser le remplacement de code lors du débogage (technologie JPDA).

Lors de son premier lancement, Eclipse crée par défaut un répertoire nommé Workspace qui va contenir les projets et les éléments qui les composent.

Le principe pour l'installation de toutes les versions d'Eclipse restent le même et d'une grande simplicité puisqu'il suffit de décompresser le contenu de l'archive d'Eclipse dans un répertoire du système.

### **2.1. Installation d'Eclipse sous Windows**

Eclipse est téléchargeable sur le site d'Eclipse :

- la version courante à l'url : <http://www.eclipse.org/downloads/>
- les versions précédentes à l'url : <http://archive.eclipse.org/eclipse/downloads/index.php>

#### **2.1.1. Installation d'Eclipse 1.0**

Il faut télécharger le fichier eclipse-SDK-R1.0-win32.zip (36,5 Mo) et le dézipper.

Pour lancer Eclipse sous Windows, il suffit de double cliquer sur le fichier eclipse.exe.

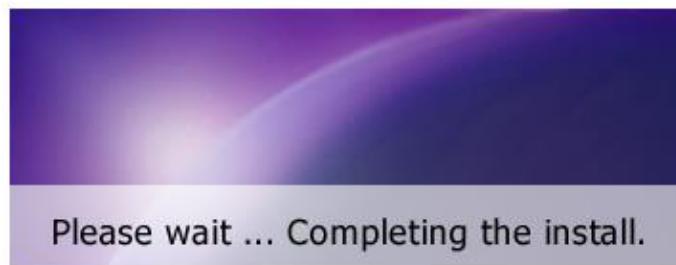
Si la splash screen reste affichée et que l'application ne se lance pas, c'est qu'elle n'arrive pas à trouver le JDK requis.

#### **2.1.2. Installation d'Eclipse 2.0**

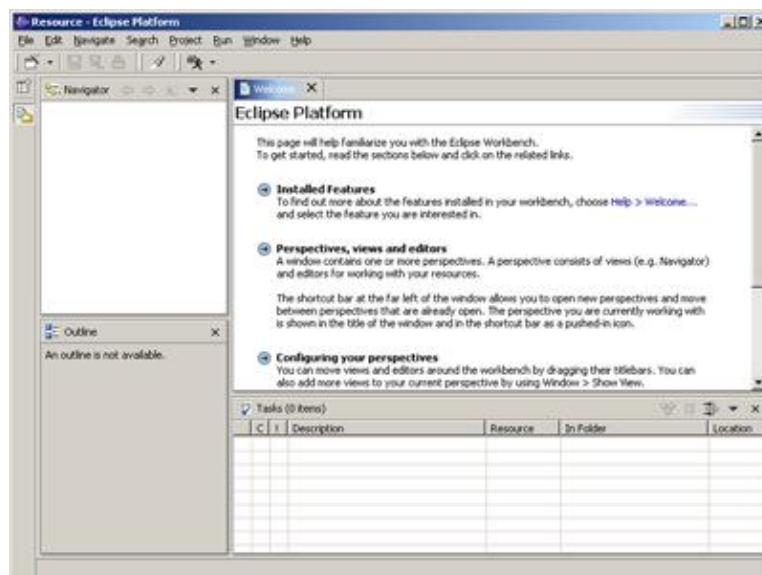
Il faut télécharger le fichier eclipse-SDK-2.0.2-win32.zip sur le site :

<http://archive.eclipse.org/eclipse/downloads/drops/R-2.0.2-200211071448/index.php>

Il suffit ensuite d' extraire l'archive dans un répertoire par exemple c:\java et d'exécuter le programme eclipse.exe situé dans le répertoire eclipse créé lors de la décompression.



L'application termine l'installation, puis s'exécute.



### 2.1.3. Installation des traductions de la version 2.x

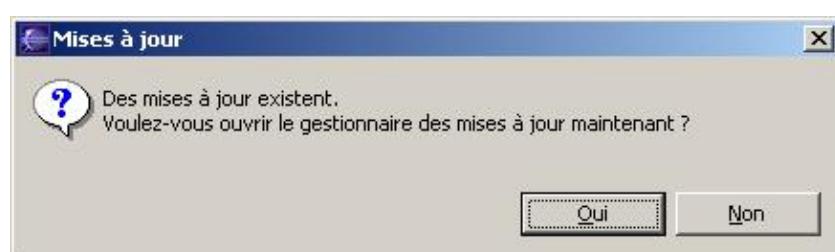
Par défaut, la langue utilisée dans Eclipse est l'anglais. I.B.M. propose des traductions pour la version 2.0.2 d'Eclipse dans différentes langues.

Il faut télécharger le fichier eclipse-nls-SDK-2.0.x.zip sur la page

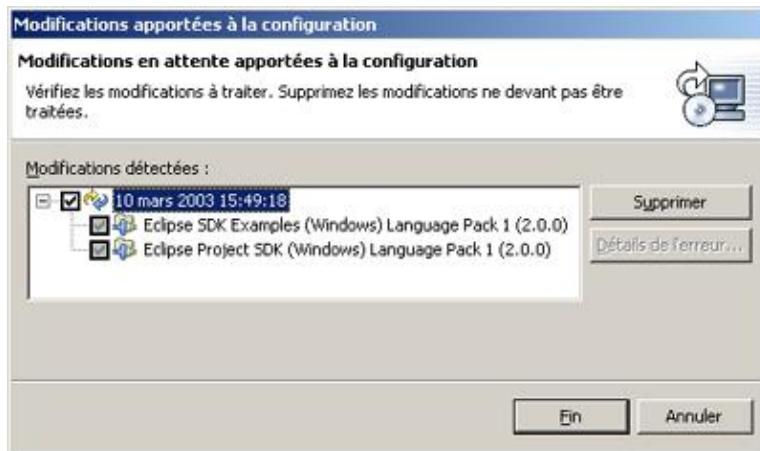
[http://archive.eclipse.org/eclipse/downloads/drops/L-2.0.x\\_Translations-200301071000/index.php](http://archive.eclipse.org/eclipse/downloads/drops/L-2.0.x_Translations-200301071000/index.php)

Il suffit ensuite le décompresser dans le répertoire qui contient le répertoire Eclipse (par exemple : c:\java).

Avec une connexion internet, lors de l'exécution suivante, l'application vérifie si une mise à jour des traductions n'est pas disponible.



Un clic sur le bouton "Oui" ouvre une boîte de dialogue qui permet de sélectionner la mise à jour à installer.

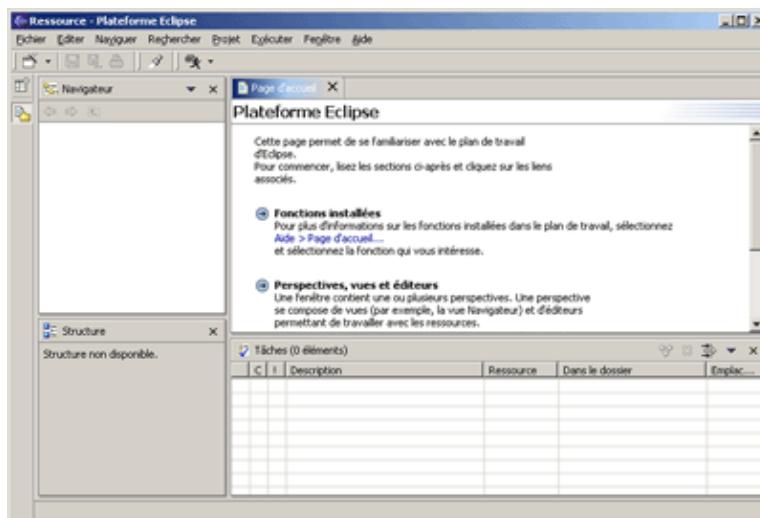


Il suffit de cocher la mise à jour souhaitée et de cliquer sur le bouton "Fin".

Les mises à jour sont téléchargées et installées. L'application doit être redémarrée.



L'application redémarre automatiquement après un clic sur le bouton "Oui", localisé dans la langue du poste.



Pour la version 2.1., il faut télécharger le fichier correspondant au système d'exploitation utilisé sur la page :

[http://archive.eclipse.org/eclipse/downloads/drops/L-2.1.1\\_Translations-200307021300/index.php](http://archive.eclipse.org/eclipse/downloads/drops/L-2.1.1_Translations-200307021300/index.php)

La procédure d'installation est identique à celle de la version 2.0.

Pour la version 2.1.1., il existe aussi un patch pour les traductions téléchargeable à la même url nommé `eclipse2.1.1.1-SDK-LanguagePackFeature-patch.zip`

Ce patch doit être installé après avoir installé les traductions initiales de la version 2.1.1 en dézippant le contenu du fichier dans le répertoire qui contient le répertoire Eclipse.

## 2.1.4. Installation d'Eclipse 3.0.1

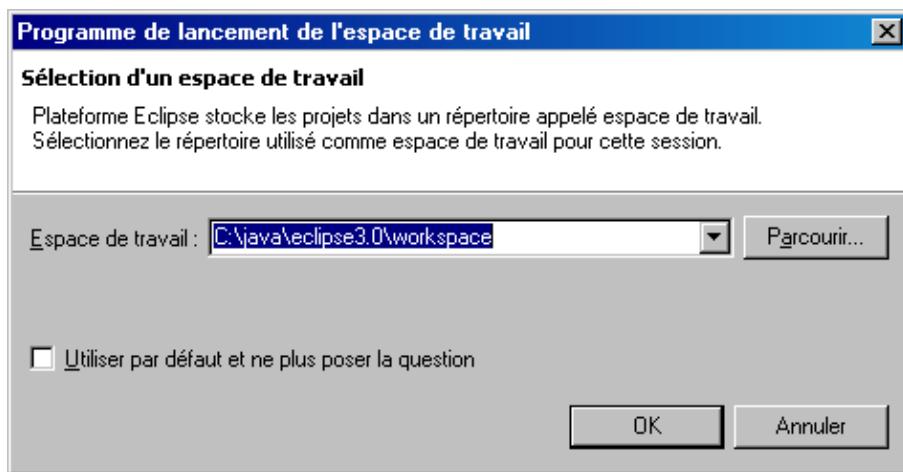
Il faut télécharger le fichier eclipse-SDK-3.0.1-win32.zip sur le site :

<http://archive.eclipse.org/eclipse/downloads/drops/R-3.0.1-200409161125/index.php>

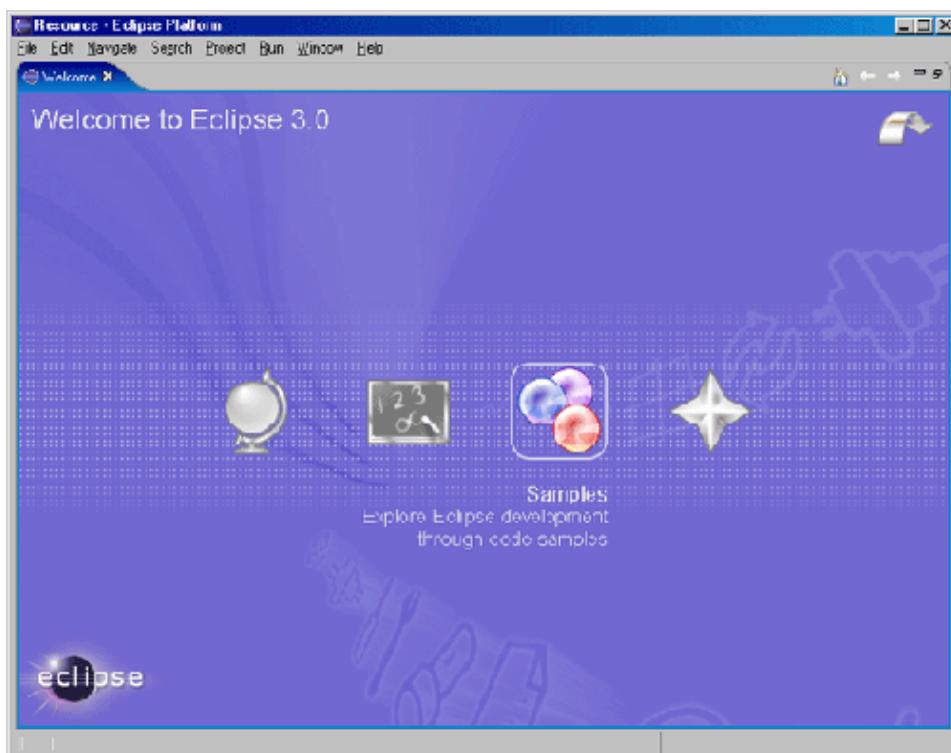
L'installation d'Eclipse 3.0 se fait de la même façon que pour les versions antérieures : il suffit d' extraire l'archive dans un répertoire par exemple c:\java et d'exécuter le programme eclipse.exe situé dans le répertoire eclipse décompressé.

Pour lancer Eclipse, il suffit de lancer le fichier eclipse.exe.

Durant la phase de lancement, Eclipse 3 propose de sélectionner l'espace de travail à utiliser. Par défaut, c'est celui présent dans le répertoire workspace du répertoire d'Eclipse qui est proposé.

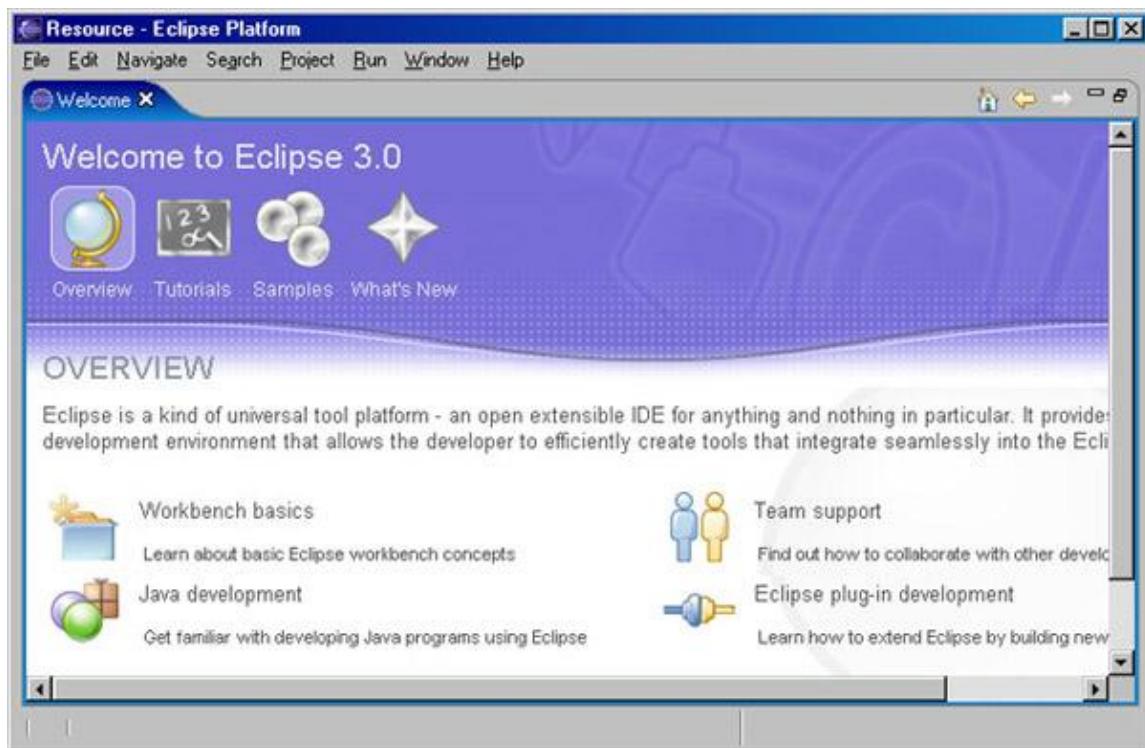


Cette demande est particulièrement utile lors de l'utilisation de plusieurs espaces de travail. Si un seul et unique workspace est utilisé, le plus simple est de cocher la case « Utiliser par défaut et ne plus poser la question » avant de cliquer sur le bouton « OK ». L'espace précisé deviendra alors celui par défaut qui sera toujours utilisé par Eclipse.

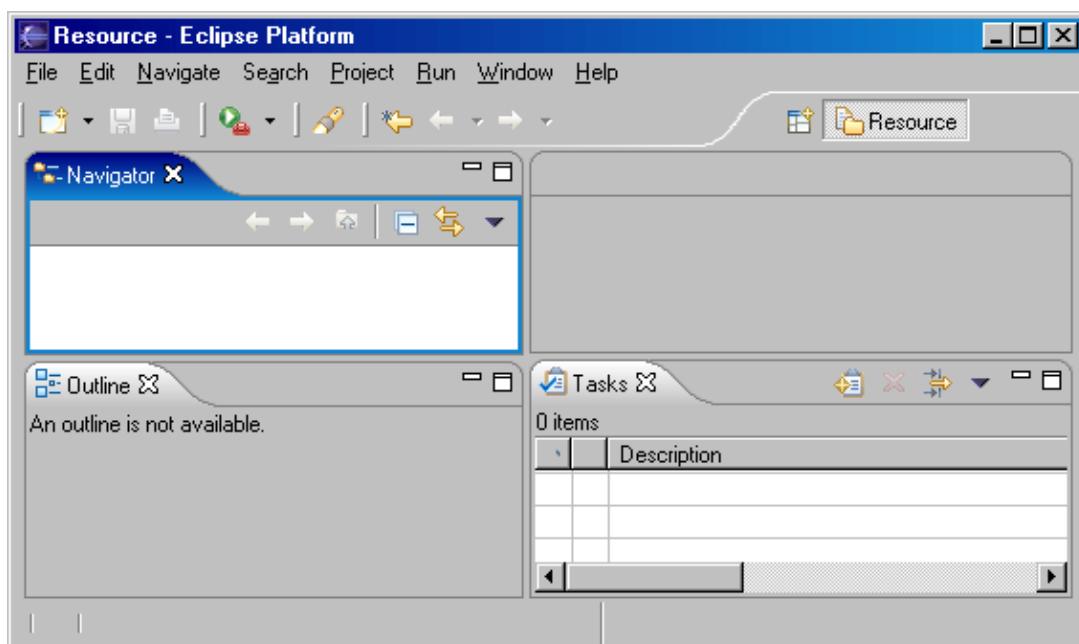


Eclipse 3.0 possède une page d'accueil permettant d'obtenir des informations sur l'outil réparties en quatre thèmes :

- overview : permet d'accéder rapidement à la partie de l'aide en ligne correspondant au thème sélectionné
- tutorials : permet d'accéder à des assistants qui permettent sous la forme de didacticiel de réaliser de simples applications ou plug-ins
- samples : permet de lancer des exemples d'applications avec SWT et JFace qu'il faut télécharger sur internet
- what's new : permet d'accéder rapidement à la partie de l'aide en ligne concernant les nouveautés d'Eclipse 3.0



Eclipse 3.0 possède une nouvelle interface liée à une nouvelle version de SWT.



## 2.1.5. Installation des traductions de la version 3.0.x

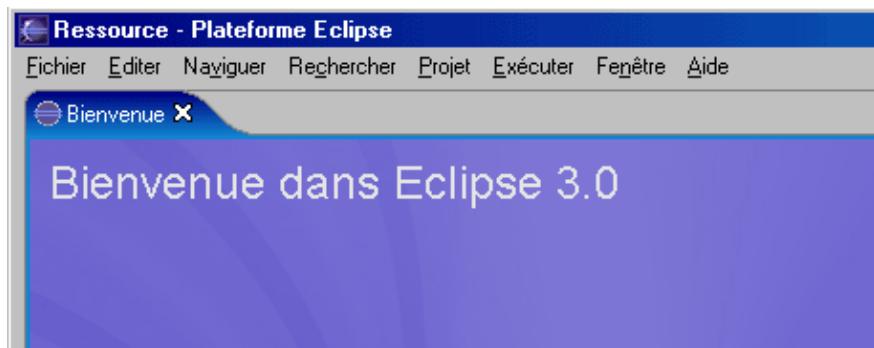
Par défaut, la langue utilisée dans Eclipse est l'anglais. I.B.M. propose des traductions pour les versions 3.0.x d'Eclipse dans différentes langues.

Il faut télécharger le fichier NLpack-eclipse-SDK-3.0.x-win32.zip sur la page

[http://archive.eclipse.org/eclipse/downloads/drops/L-3.0.1\\_Translations-200409161125/index.php](http://archive.eclipse.org/eclipse/downloads/drops/L-3.0.1_Translations-200409161125/index.php)

Il suffit ensuite de le décompresser dans le répertoire qui contient le répertoire Eclipse (par exemple : c:\java).

Exécuter la commande eclipse -clean pour l'exécution suivante d'Eclipse.

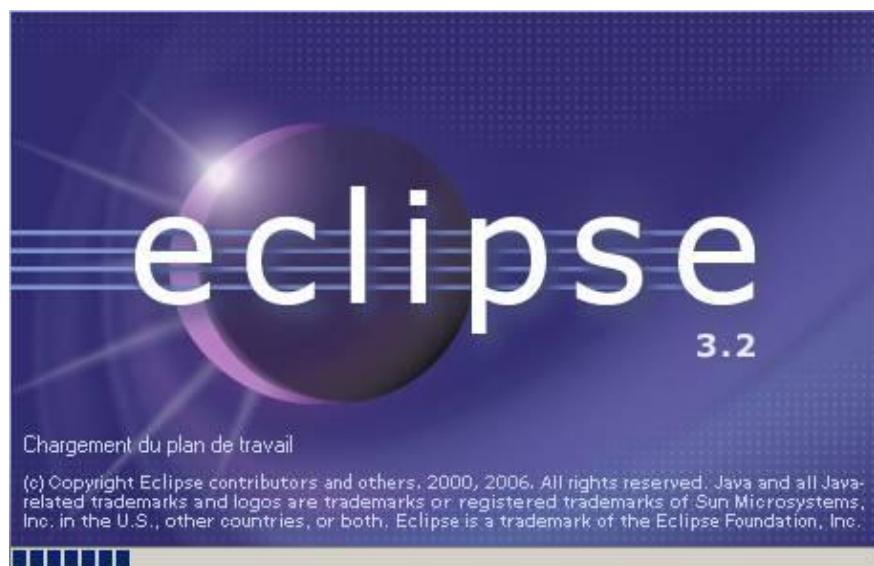


## 2.1.6. Installation d'Eclipse 3.2.x

L'installation de cette version se fait comme pour les précédentes versions : téléchargez le fichier eclipse-SDK-3.2-win32.zip et décompressez le dans un répertoire du système.

Pour lancer l'application, exécutez le programme eclipse.exe dans le répertoire créé lors de la décompression.

La splashscreen intègre la barre de progression des étapes de l'initialisation de l'application.



La vue d'accueil a été modifiée.

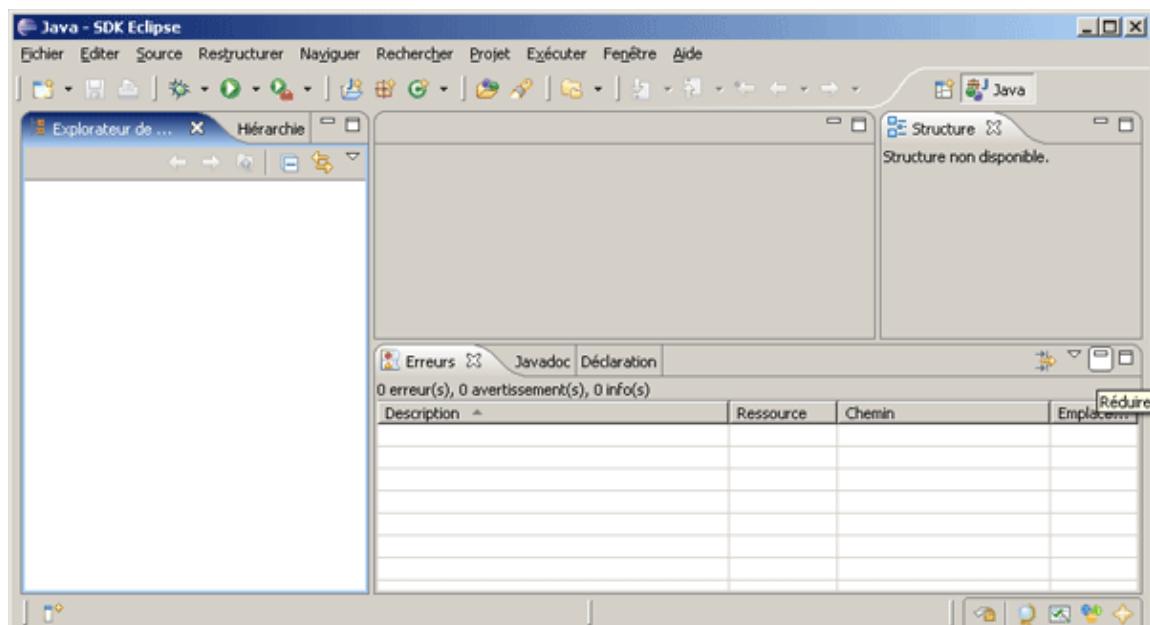


### 2.1.7. Installation des traductions de la version 3.2.x

Les traductions de la version 3.2.1 sont téléchargeables à l'url [http://download.eclipse.org/eclipse/downloads/drops/L-3.2\\_Language\\_Packs-200607121700/index.php](http://download.eclipse.org/eclipse/downloads/drops/L-3.2_Language_Packs-200607121700/index.php) depuis le 12 juillet 2006.

Téléchargez les fichiers NLpack1-eclipse-JDT-3.2.zip, NLpack1-eclipse-PDE-3.2.zip, NLpack1-eclipse-platform-3.2-win32.zip, NLpack1-eclipse-RCP-3.2-win32.zip et NLpack1-eclipse-SDK-3.2-win32.zip et décompresser leur contenu dans un répertoire du système.

Déplacez les répertoires plugins et features créés lors de la décompression dans le répertoire d'Eclipse.



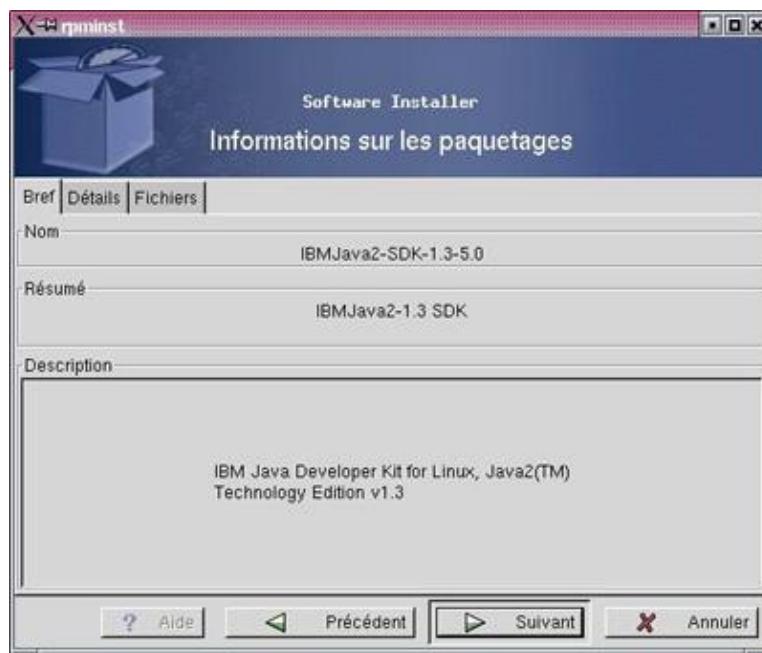
Les traductions de la version 3.2.1 sont téléchargeables à l'url  
[http://download.eclipse.org/eclipse/downloads/drops/L-3.2.1\\_Language\\_Packs-200609210945/index.php](http://download.eclipse.org/eclipse/downloads/drops/L-3.2.1_Language_Packs-200609210945/index.php)  
depuis le 21 septembre 2006.

## 2.2. Installation d'Eclipse sous Linux

Toutes les versions d'Eclipse peuvent être installées sous Linux.

### 2.2.1. Installation d'Eclipse 1.0 sous Mandrake 8.1

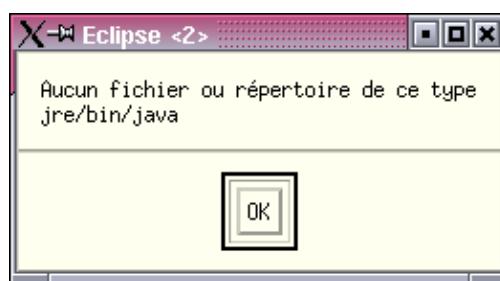
Il faut installer un JDK 1.3 minimum, par exemple celui fourni par IBM qu'il est possible d'installer grâce au gestionnaire de paquet.



Il faut ajouter le répertoire bin du JDK à la variable système PATH pour permettre au système de trouver les exécutables nécessaires.

PATH=\$PATH:/opt/IBMJava2-13/bin

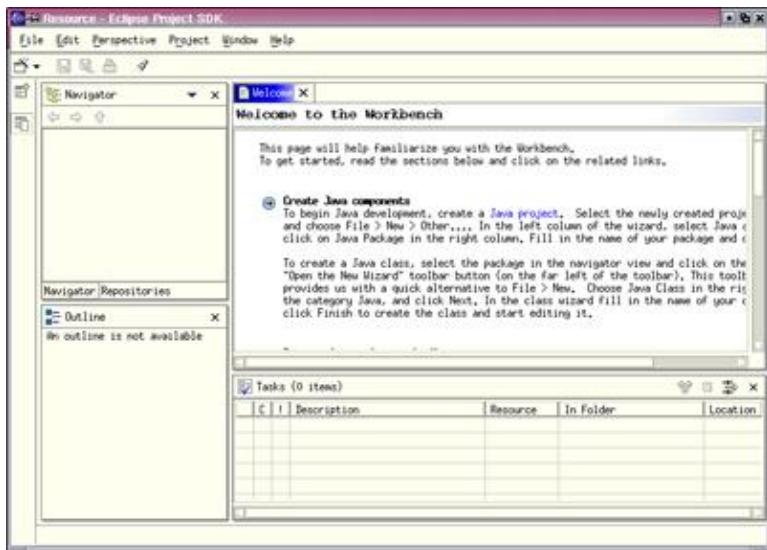
Si les exécutables ne sont pas trouvés, une boîte de dialogue affiche le message suivant :



Pour exécuter Eclipse, il suffit de lancer eclipse dans un shell.

Exemple :

```
[jumbo@charlemagne eclipse]$ ./eclipse -data workspace
```



## 2.2.2. Installation Eclipse 2.1 sous Mandrake 8.0

Il faut installer un JDK 1.3 minimum, par exemple celui fourni par IBM qu'il est possible d'installer grâce au gestionnaire de paquet. Il faut aussi que la variable JAVA\_HOME contienne le chemin vers le JDK.

Exemple :

```
[java@localhost eclipse]$ echo $JAVA_HOME
/usr/java/jdk1.3.1
[java@localhost eclipse]$ java -version
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1-b24)
Java HotSpot(TM) Client VM (build 1.3.1-b24, mixed mode)
[java@localhost eclipse]$ which java
/usr/java/jdk1.3.1/bin/java
```

Il existe deux versions pour Linux en fonction de la bibliothèque graphique utilisée : une utilisant Motif et une autre utilisation GTK 2.

Pour la version Motif, il faut télécharger le fichier eclipse-SDK-2.1.1-linux-motif.zip.

Il faut décompresser le fichier dans un répertoire, par exemple /opt avec l'utilisateur root.

Exemple :

```
[root@localhost opt]# unzip eclipse-SDK-2.1-linux-motif.zip
Archive:  eclipse-SDK-2.1-linux-motif.zip
  creating: eclipse/
  inflating: eclipse/libXm.so.2.1
  linking:  eclipse/libXm.so          -> libXm.so.2.1
  linking:  eclipse/libXm.so.2        -> libXm.so.2.1
  creating: eclipse/plugins/
...
[java@localhost eclipse]$ ll
total 2004
drwxrwxr-x    5 root      root          4096 Mar 27 22:11 .
drwxr-xr-x   11 root      root          4096 Jul 10 00:12 ..
-rw-rw-r--    1 root      root           59 Mar 27 22:11 .eclipseproduct
-rw-rw-r--    1 root      root         15048 Mar 27 22:11 cpl-v10.html
-rwxr-xr-x    1 root      root         41003 Mar 27 22:11 eclipse*
drwxrwxr-x   10 root      root          4096 Mar 27 22:11 features/
-rw-rw-r--    1 root      root         10489 Mar 27 22:11 icon.xpm
```

```

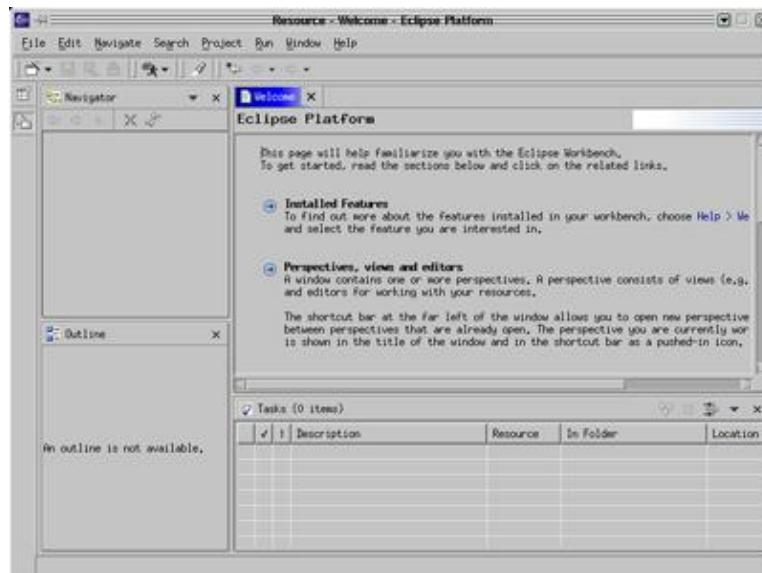
-rw-rw-r--    1 root      root          619 Mar 27 22:11 install.ini
lrwxrwxrwx    1 root      root          12 Jul 10 00:11 libXm.so -> libXm.so.2.1*
lrwxrwxrwx    1 root      root          12 Jul 10 00:11 libXm.so.2 -> libXm.so.2.1*
-rwxr-xr-x    1 root      root        1915756 Mar 27 22:11 libXm.so.2.1*
-rw-rw-r--    1 root      root          4743 Mar 27 22:11 notice.html
drwxrwxr-x   64 root     root         4096 Mar 27 22:11 plugins/
drwxrwxr-x   2 root     root         4096 Mar 27 22:11 readme/
-rw-rw-r--    1 root      root        16549 Mar 27 22:11 startup.jar

```

Pour utiliser Eclipse avec un utilisateur sans privilège particulier, il faut définir la variable LD\_LIBRARY\_PATH et exécuter Eclipse

Exemple :

```
[java@localhost java]$ cd /opt/eclipse
[java@localhost eclipse]$ LD_LIBRARY_PATH=/opt/eclipse:$LD_LIBRARY_PATH
[java@localhost eclipse]$ /opt/eclipse/eclipse -data $HOME/workspace
```



Si la variable LD\_LIBRARY\_PATH n'est pas correctement valorisée, le message d'erreur suivant est affiché et Eclipse ne peut pas se lancer.

Exemple :

```
[java@localhost java]$ /opt/eclipse/eclipse -data $HOME/workspace
/opt/eclipse/eclipse: error while loading shared libraries: libXm.so.2: cannot load shared object file: No such file or directory
```

### 2.2.3. Installation Eclipse 3.0.x sous Mandrake 10.1

Il faut obligatoirement qu'un JDK 1.3 minimum soit installé sur le système. Dans cette section le JDK utilisé est le 1.4.2 de Sun.

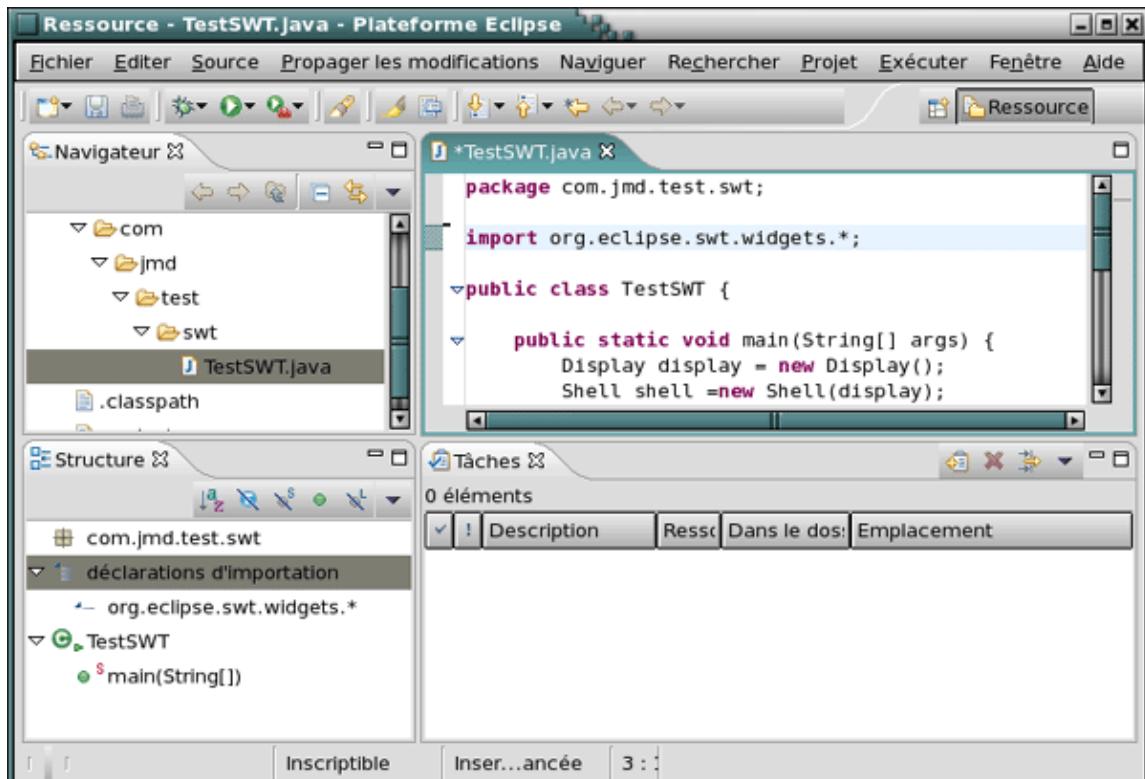
Exemple :

```
[java@localhost eclipse]$ java -version
java version "1.4.2_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_06-b03)
```

Deux versions existent pour Linux selon la bibliothèque graphique utilisée :

- une utilisant Motif
- une utilisant Gtk : eclipse-SDK-3.0-linux-gtk.zip

La version Gtk sera utilisée dans cette section.



### 2.2.3.1. Installation par et pour un seul utilisateur

Le plus simple est de décompresser le fichier eclipse-SDK-3.0-linux-gtk.zip dans le répertoire home de l'utilisateur.

L'inconvénient de cette méthode est que par défaut seul cet utilisateur pourra utiliser Eclipse.

#### Exemple :

```
[java@localhost eclipse]$ cp eclipse-SDK-3.0-linux-gtk.zip ~
[java@localhost eclipse]$ cd ~
[java@localhost java]$ unzip eclipse-SDK-3.0-linux-gtk.zip
  creating: eclipse/plugins/org.eclipse.pde.build_3.0.0/
  inflating: eclipse/plugins/org.eclipse.pde.build_3.0.0/pdebuild.jar
  creating: eclipse/plugins/org.eclipse.pde.build_3.0.0/lib/
  inflating: eclipse/plugins/org.eclipse.pde.build_3.0.0/lib/pdebuild-ant.jar
  inflating: eclipse/plugins/org.eclipse.pde.build_3.0.0/.options
  creating: eclipse/plugins/org.eclipse.pde.build_3.0.0/feature/
...
  inflating: eclipse/startup.jar
  inflating: eclipse/icon.xpm
  inflating: eclipse/eclipse
[java@localhost java]$ ll
total 87140
```

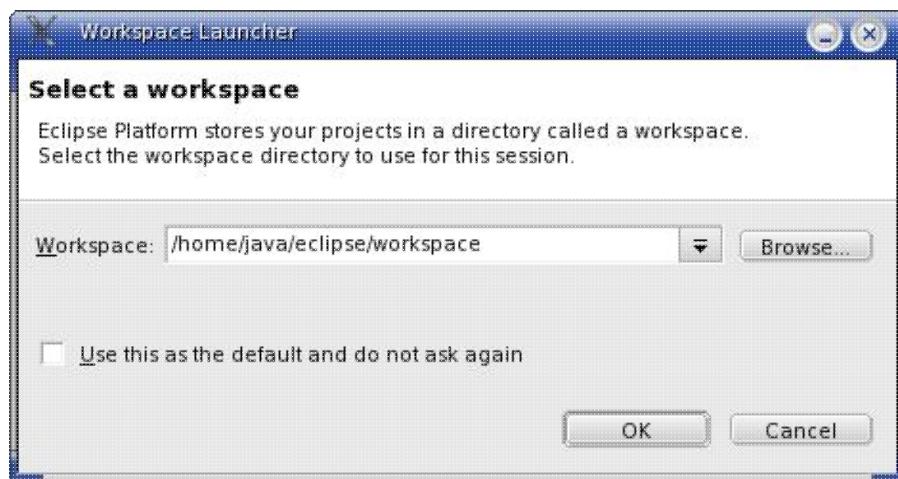
```

drwxr-xr-x  3 java java      4096 oct 16 21:24 Desktop/
drwxr-xr-x  2 java java      4096 oct 16 22:34 Documents/
drwxr-xr-x  6 java java      4096 oct 18 23:23 eclipse/
-rw-r--r--  1 java java  89113015 oct 18 23:23 eclipse-SDK-3.0-linux-gtk.zip*
-rw-rw-r--  1 java java          2 oct 16 22:23 java.sh~
drwx----- 3 java java      4096 oct 18 23:17 tmp/
[java@localhost java]$ cd eclipse
[java@localhost eclipse]$ ll
total 100
drwxrwxr-x  2 java java  4096 jun 25 18:43 configuration/
-rw-rw-r--  1 java java 15049 jun 25 18:43 cpl-v10.html
-rwxr-xr-x  1 java java 27119 jun 25 18:43 eclipse*
drwxr-xr-x  9 java java  4096 oct 18 23:23 features/
-rw-rw-r--  1 java java 10489 jun 25 18:43 icon.xpm
-rw-rw-r--  1 java java  5810 jun 25 18:43 notice.html
drwxr-xr-x 85 java java  4096 oct 18 23:23 plugins/
drwxrwxr-x  2 java java  4096 jun 25 18:43 readme/
-rw-rw-r--  1 java java 17663 jun 25 18:43 startup.jar
[java@localhost eclipse]$ ./eclipse&
[1] 3093
[java@localhost eclipse]$

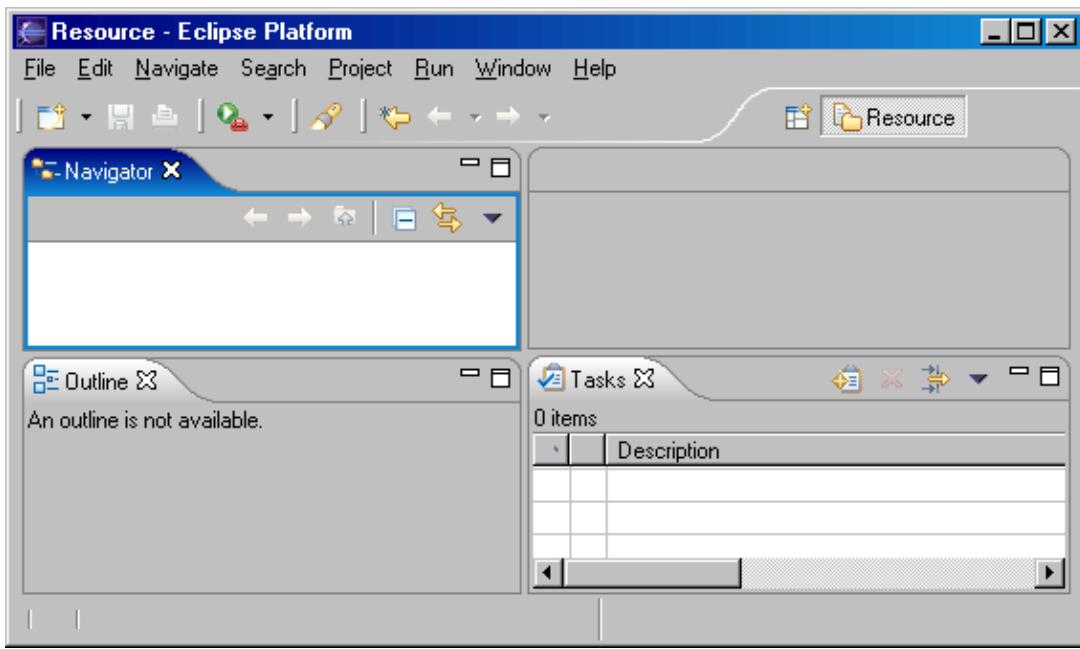
```

Le fichier `eclipse-SDK-3.0-linux-gtk.zip` peut être supprimé.

Le workspace à utiliser peut être celui proposé par défaut (celui dans le répertoire d'installation d'Eclipse)



L'apparence d'Eclipse dépend du thème et de l'environnement graphique utilisé, ici sous KDE :



### 2.2.3.2. Installation par root pour plusieurs utilisateurs

Eclipse peut être décompresser dans un répertoire accessible aux utilisateurs, par exemple /usr/local.

Exemple :

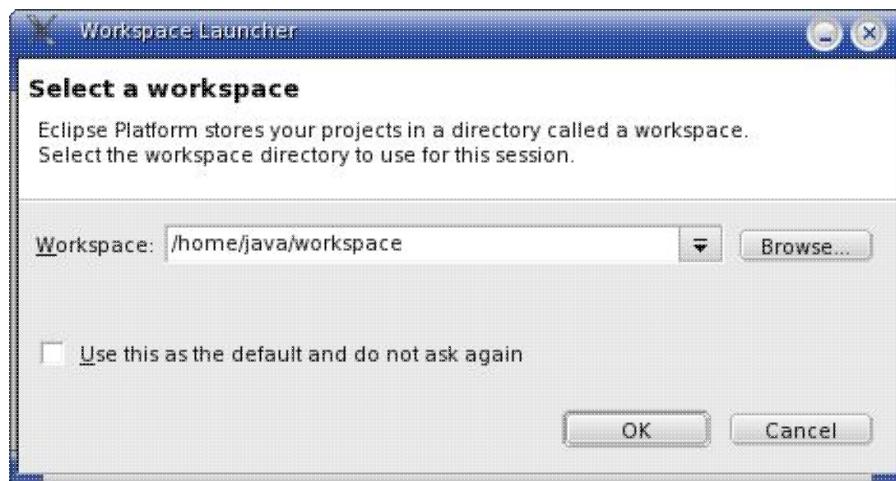
```
[root@localhost eclipse3.0.x]# cp eclipse-SDK-3.0-linux-gtk.zip /usr/local
[root@localhost eclipse3.0.x]# cd /usr/local
[root@localhost local]# ll
total 87164
drwxr-xr-x  2 root root    4096 jan  5  2004 bin/
drwxr-xr-x  2 root root    4096 jan  5  2004 doc/
-rw-r--r--  1 root root  89113015 oct 18 23:55 eclipse-SDK-3.0-linux-gtk.zip*
drwxr-xr-x  2 root root    4096 jan  5  2004 etc/
drwxr-xr-x  2 root root    4096 jan  5  2004 games/
drwxr-xr-x  2 root root    4096 jan  5  2004 include/
drwxr-xr-x  2 root root    4096 jan  5  2004 lib/
drwxr-xr-x  2 root root    4096 jan  5  2004 libexec/
drwxr-xr-x  3 root root    4096 oct 16 22:19 man/
drwxr-xr-x  2 root root    4096 jan  5  2004 sbin/
drwxr-xr-x  5 root root    4096 oct 14 01:57 share/
drwxr-xr-x  2 root root    4096 jan  5  2004 src/
[root@localhost local]# unzip -q eclipse-SDK-3.0-linux-gtk.zip
[root@localhost local]# ll
total 87168
drwxr-xr-x  2 root root    4096 jan  5  2004 bin/
drwxr-xr-x  2 root root    4096 jan  5  2004 doc/
drwxr-xr-x  6 root root    4096 oct 18 23:56 eclipse/
-rw-r--r--  1 root root  89113015 oct 18 23:55 eclipse-SDK-3.0-linux-gtk.zip*
drwxr-xr-x  2 root root    4096 jan  5  2004 etc/
drwxr-xr-x  2 root root    4096 jan  5  2004 games/
drwxr-xr-x  2 root root    4096 jan  5  2004 include/
drwxr-xr-x  2 root root    4096 jan  5  2004 lib/
drwxr-xr-x  2 root root    4096 jan  5  2004 libexec/
drwxr-xr-x  3 root root    4096 oct 16 22:19 man/
drwxr-xr-x  2 root root    4096 jan  5  2004 sbin/
drwxr-xr-x  5 root root    4096 oct 14 01:57 share/
drwxr-xr-x  2 root root    4096 jan  5  2004 src/
[root@localhost local]# rm eclipse-SDK-3.0-linux-gtk.zip
rm: détruire fichier régulier `eclipse-SDK-3.0-linux-gtk.zip'? o
[root@localhost local]#
```

Il suffit alors à un utilisateur d'exécuter Eclipse

**Exemple :**

```
[java@localhost java]$ /usr/local/eclipse/eclipse&
[1] 3676
[java@localhost java]$
```

Attention dans ce cas, il sera nécessaire de préciser le chemin d'un workspace utilisable en écriture par l'utilisateur.



### 2.2.3.3. Installation des traductions

L'installation des traductions se fait en décompressant le fichier NLpack-eclipse-SDK-3.0.x-linux-gtk.zip, téléchargé sur le site d'Eclipse, dans le répertoire contenant le répertoire d'Eclipse.

**Exemple :**

```
[java@localhost java]$ ll
total 23664
drwxr-xr-x  3 java  java    4096 nov  6  00:42 Desktop/
drwxr-xr-x  2 java  java    4096 oct 19  00:00 Documents/
drwxr-xr-x  7 java  java    4096 nov  6 11:03 eclipse/
-rw-rw-r--  1 java  java     2 oct 16 22:23 java.sh~
-rwxr--r--  1 root  root 24175833 nov  6 11:02 NLpack-eclipse-SDK-3.0.x-linux-gtk.zip*
drwx----- 4 java  java    4096 nov  6 11:00 tmp/
drwxr-xr-x  3 java  java    4096 oct 18 23:59 workspace/
[java@localhost java]$ unzip NLpack-eclipse-SDK-3.0.x-linux-gtk.zip
[java@localhost eclipse]$ ./eclipse -clean&
```

Comme sous Windows, l'exécution suivante d'Eclipse doit se faire avec l'option -clean.



## 2.3. Les paramètres

Eclipse 2.x accepte plusieurs paramètres sur la ligne de commande qui lance l'application.

Le paramètre `-vm` permet de préciser la machine virtuelle qui va exécuter Eclipse. Ce paramètre est utile lorsque la machine possède plusieurs JRE installés.

Exemple sous Windows :

```
Eclipse.exe -vm C:\java\jre1.4.2\bin\javaw.exe
```

Le paramètre `-data` permet de préciser l'emplacement du workspace. Ce paramètre est utile pour pouvoir utiliser plusieurs workspaces.

Exemple sous Windows :

```
Eclipse -data C:\Test\workspace
```

Le paramètre `-ws` permet sous les environnements de type Unix de préciser la bibliothèque graphique utilisée. Les valeurs possibles sont "motif" et "gtk".

La paramètre `-arch` permet de préciser l'architecture d'exécution.

La paramètre `-vmargs` permet de préciser des arguments à la machine virtuelle qui exécute Eclipse.

## 2.4. La résolution de problèmes

Cette section présente quelques résolutions à des problèmes qui peuvent survenir lors de l'utilisation d'Eclipse.

### 2.4.1. Un plug-in installé n'est pas visible

La première chose à faire est de relancer Eclipse dans une ligne de commande avec l'option -clean.

Si cela ne fonctionne toujours pas, il faut vérifier que toutes les dépendances nécessaires au plug-in soient installées dans Eclipse.

Remarque : le pré requis est que le plug-in installé soit installé avec la version d'Eclipse utilisée.

### 2.4.2. L'espace de travail est déjà utilisé



Ce message apparaît si Eclipse est déjà ouvert sur le Workspace à utiliser.

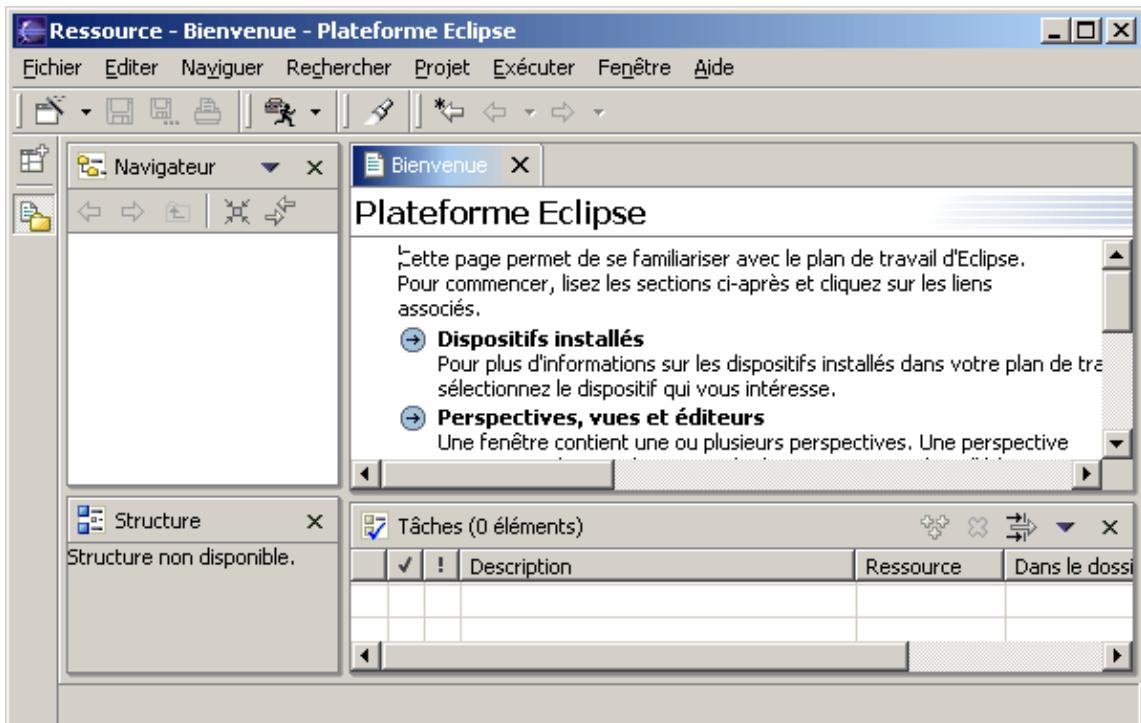
Cependant, en cas de plantage d'Eclipse, il est possible que ce message s'affiche même si Eclipse n'est plus en cours d'exécution.

Dans ce cas, il faut supprimer le processus javax.exe toujours en cours d'exécution dans la liste des tâches.

### 3. Le plan de travail (Workbench)

# Chapitre 3

Au lancement d'Eclipse, une seule fenêtre s'ouvre contenant le plan de travail (Workbench). Le plan de travail est composé de perspectives dont plusieurs peuvent être ouvertes mais une seule est affichée en même temps. A l'ouverture, c'est la perspective "Ressource" qui est affichée par défaut.



Une perspective contient des sous fenêtres qui peuvent contenir des vues (views) et des éditeurs (editors)

La partie gauche du plan de travail est composée d'une barre qui contient une icône pour chaque perspective ouverte et une icône pour ouvrir une nouvelle perspective. L'icône enfoncee est celle de la perspective actuellement affichée. Le titre de la fenêtre du plan de travail contient le nom de la perspective courante.

Eclipse possède dans le plan de travail une barre de menu et une barre de tâches. Elles sont toutes les deux dynamiques en fonction du type de la sous fenêtre active de la perspective courante.

Eclipse propose de nombreux assistants pour faciliter la réalisation de certaines tâches comme la création d'entités.

#### 3.1. Les perspectives

Une perspective présente une partie du projet de développement selon un certain angle de vue.

Chaque perspective possède une icône qui permet de l'identifier plus rapidement. La version 1.00 d'Eclipse possède les perspectives suivantes :

Perspective	Icône	Rôle
Debug		Débogueur
Help		Aide en ligne
Java		Ecriture de code Java
Java Type Hierarchy		Navigation dans la hiérarchie et les éléments des classes
Plug-in Development		Création de plug-in
Resource		Gestion du contenu de l'espace de travail
Scripts		
Team		Gestion du travail collaboratif



Perspective	Icône	Rôle
Débogage		Débogueur
Java		Ecriture de code Java
Navigation Java		Navigation dans la hiérarchie et les éléments des classes
Hiérarchies des types Java		
Développement de plug-in		Création de plug-in
Ressource		Gestion du contenu de l'espace de travail
Installation/Mise à jour		Installation et mise à jour de plug-ins via le web
Exploration du référentiel CVS		Gestion du travail collaboratif avec CVS



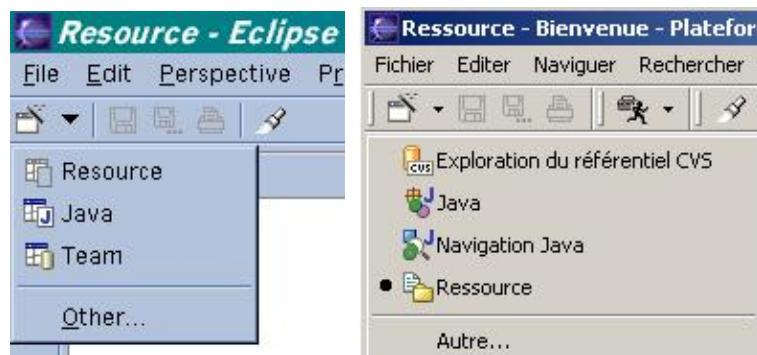
Perspective	Icône	Rôle
Débogage		Débogueur
Java		Ecriture de code Java
Navigation Java		Navigation dans la hiérarchie et les éléments des classes
Hierarchie de type Java		
Développement de plug-in		Création de plug-ins
Ressource		Gestion du contenu de l'espace de travail
Synchronisation de l'équipe		
Exploration du référentiel CVS		Gestion du travail collaboratif avec CVS

Pour ouvrir une nouvelle perspective, il y a deux manières de procéder :

- Cliquer sur l'icône dans la barre des perspectives

- Utiliser l'option "Ouvrir" du menu "Perspective" (Eclipse 1.0) ou l'option "Ouvrir la perspective" du menu "Fenêtre" (Eclipse 2.0)

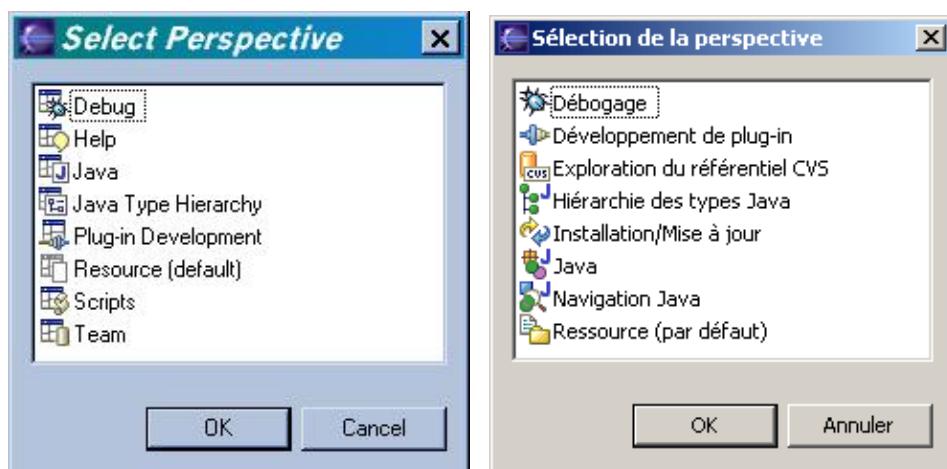
Lors d'un clic sur l'icône , un menu flottant s'ouvre pour permettre la sélection de la perspective à ouvrir. Si elle n'appartient pas à la liste, elle est accessible en cliquant sur l'option « Autre ».



Eclipse 1.0

Eclipse 2.0

Un clic sur l'option « Autre... » ouvre une boîte de dialogue dans laquelle il est possible de sélectionner la nouvelle perspective à afficher.



Eclipse 1.0

Eclipse 2.0

La perspective par défaut (celle qui est affichée à l'ouverture de l'application) est indiquée.

Il est possible d'ouvrir plusieurs perspectives d'un même type en même temps. Cependant une seule perspective, quelque soit son type est affichée à un moment donné.

Toutes les perspectives ouvertes possèdent une icône dans la barre des perspectives. Pour en afficher une, il suffit de cliquer sur son icône. La perspective courante est celle dont l'icône est enfoncee.



Dans Eclipse 3, la position de la barre des perspectives peut être modifiée. Par défaut, elle se situe en haut à droite. Le menu contextuel « Verrouiller » permet de modifier son positionnement.



L'option « Afficher le texte », cochée par défaut, permet d'avoir à côté de l'icône un nom court facilitant l'identification de chaque perspective.

## 3.2. Les vues et les éditeurs

Une perspective est composée de sous fenêtres qui peuvent être de deux types :

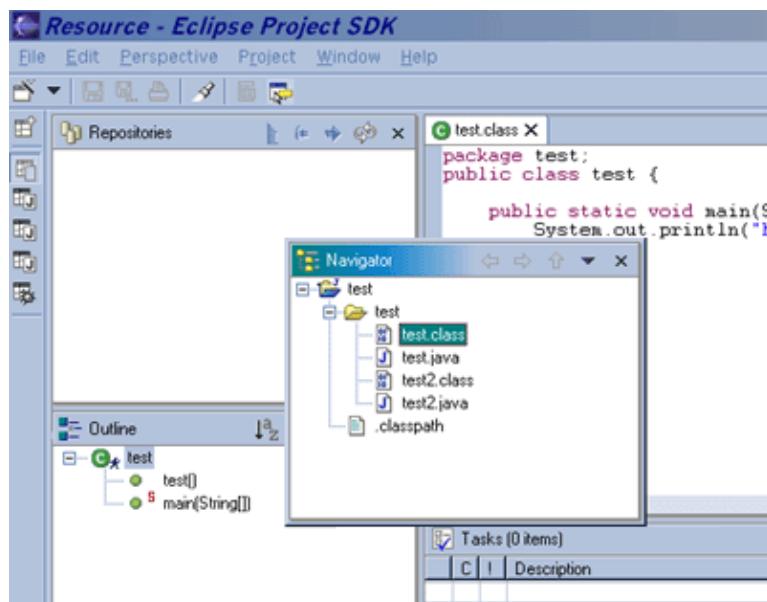
- les vues
- les éditeurs

Une vue permet de visualiser et de sélectionner des éléments. Il ne peut y avoir qu'une seule vue particulière dans une même perspective (il n'est pas possible d'afficher plusieurs fois la même vue dans une même perspective). Plusieurs vues différentes peuvent être rassemblées dans une même sous fenêtre en effectuant un cliquer/glisser de la barre de titre d'une vue sur une autre. L'accès à chaque vue se fait alors grâce à un onglet.

Un éditeur permet de visualiser mais aussi de modifier le contenu d'un élément. Un éditeur peut contenir plusieurs éléments, chacun étant identifié par un onglet.

Dans une perspective, il ne peut y avoir qu'une seule sous fenêtre active contenant soit un éditeur soit une vue. La barre de titre de cette sous fenêtre est colorée. Pour activer une sous fenêtre, il suffit de cliquer dessus.

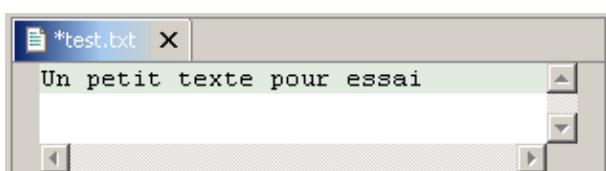
Avec Eclipse 1.0 sous Windows, les vues peuvent être extraites du workbench pour devenir des fenêtres indépendantes. Pour cela, il faut cliquer sur la barre de titre de la vue, en maintenant le bouton de la souris enfoncé, effectuer un glissement avec la souris vers une zone non empilable (sur un éditeur, les bords de l'écran, ...) et de relâcher le bouton de la souris (le curseur de la souris prend la forme d'une petite fenêtre aux endroits adéquats).



Pour réaliser l'opération inverse, il faut faire glisser la fenêtre au-dessus d'une vue existante : elles seront alors empilées.

### 3.2.1. Les éditeurs

Il existe plusieurs éditeurs en fonction du type de l'élément qui est édité.



L'onglet de l'éditeur contient le libellé de l'élément traité. Une petite étoile apparaît à droite de ce libellé si l'élément a été modifié sans être sauvegardé.

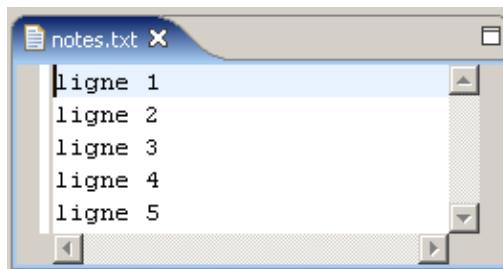
Pour fermer l'éditeur contenant l'élément en cours, il suffit de cliquer sur l'icône en forme de croix de l'onglet.

Une confirmation sera demandée en cas de fermeture alors que l'élément a été modifié sans sauvegarde. Il est aussi possible d'utiliser l'option "Fermer" et "Fermer tout" du menu "Fichier" du plan de travail pour fermer respectivement le fichier en cours ou tous les fichiers.

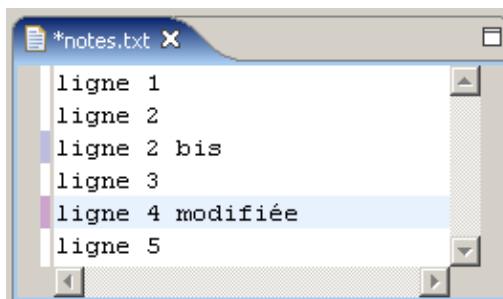
Si l'élément édité ne possède pas d'éditeur dédié dans Eclipse, il tente d'ouvrir un outil associé au type de la ressource dans le système d'exploitation.



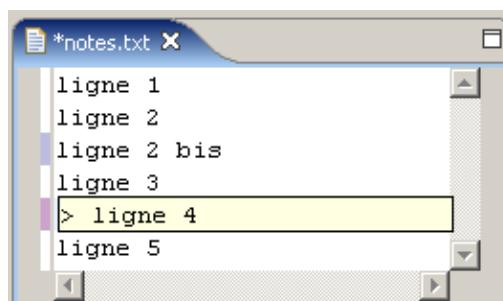
Quick Diff est une fonctionnalité qui permet de visualiser rapidement les modifications apportées dans un éditeur par rapport à une source (la version sur disque dans l'espace de travail par défaut).



Ainsi les lignes ajoutées et modifiées apparaissent avec une couleur différente dans la barre à gauche de la zone d'édition.

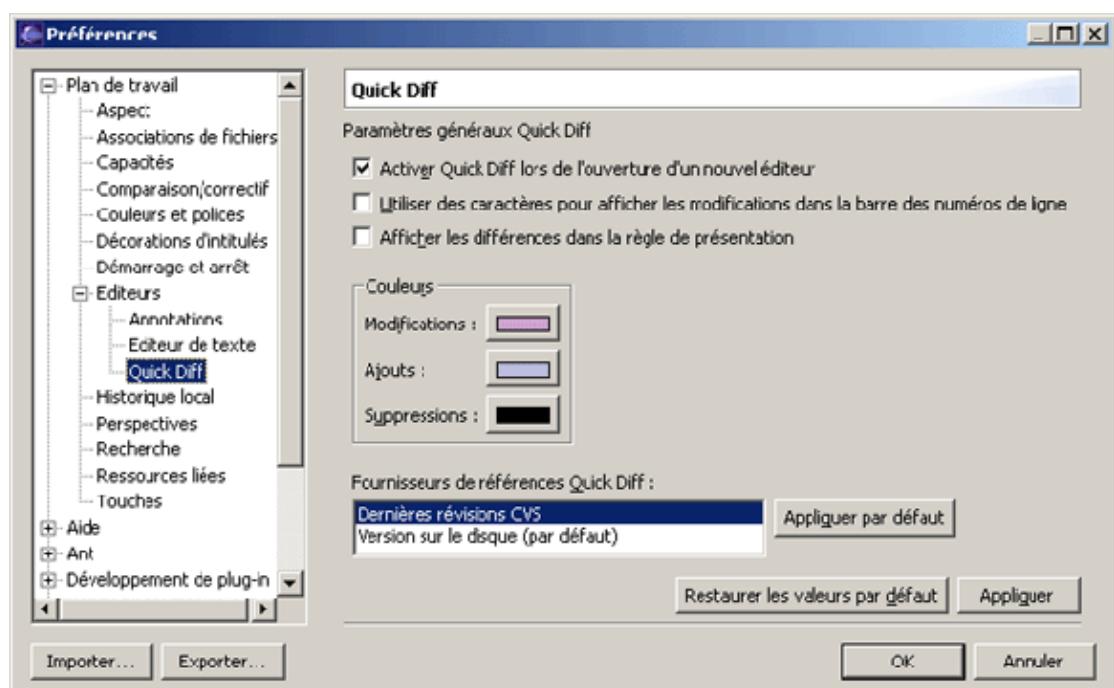


En laissant le curseur de la souris sur la zone colorée, un bulle d'aide affiche le contenu de la zone originale.

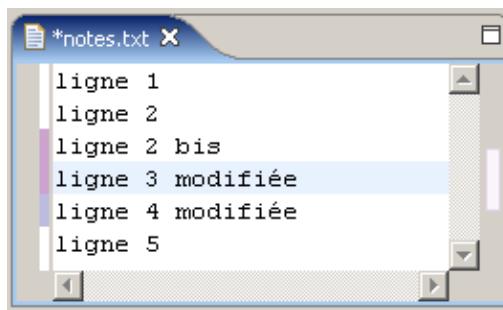


Pour activer ou désactiver Quick Diff, il est possible d'utiliser la combinaison de touche Ctrl+Maj+Q ou d'utiliser l'option « Activer/Désactiver Quick Diff » du menu contextuel de la barre de gauche.

Les paramètres de Quick Diff peuvent être configurés dans les préférences sous l'arborescence « Plan de travail / Editeurs/ Quick Diff ».



L'option "Afficher les différences dans la règle de présentation" permet de marquer les lignes modifiées par une zone blanche dans la barre à droite de l'éditeur.



Plusieurs raccourcis ont été ajoutés dans les éditeurs :

Raccourci clavier	Rôle
Alt+flèche vers le haut / bas	Déplacement d'un ensemble de lignes sélectionnées
Ctrl +Alt+flèche vers le haut	Copie d'un ensemble de lignes sélectionnées
Ctrl+Maj+Entrée	Insérer une ligne au dessus de la ligne courante
Maj+Entrée	Insérer une ligne en dessous de la ligne courante
Ctrl+Maj+Y	Conversion du texte sélectionné en minuscule
Ctrl+Maj+X	Conversion du texte sélectionné en majuscule

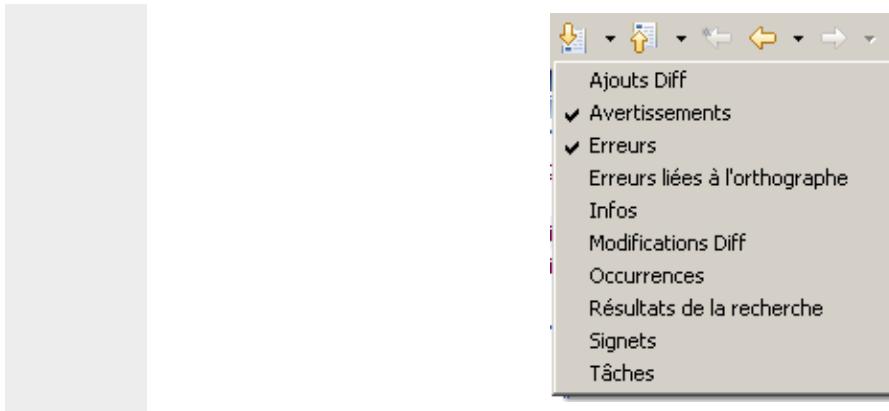


Deux icônes dans la barre d'outils permettent de naviguer dans les annotations du contenu de l'éditeur :

: permet de passer à l'annotation suivante

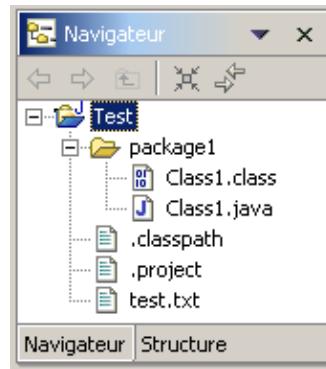
: permet de passer à l'annotation précédente

Un clic sur la petite flèche ouvre un menu déroulant qui permet de sélectionner le type d'annotation concernée par la navigation dans chaque sens.



### 3.2.2. Les vues

Les vues permettent de présenter des informations et de naviguer dans les ressources. Plusieurs vues peuvent être réunies dans une même sous fenêtre : dans ce cas, le passage d'une vue à l'autre se fait via un clic sur l'onglet concerné.

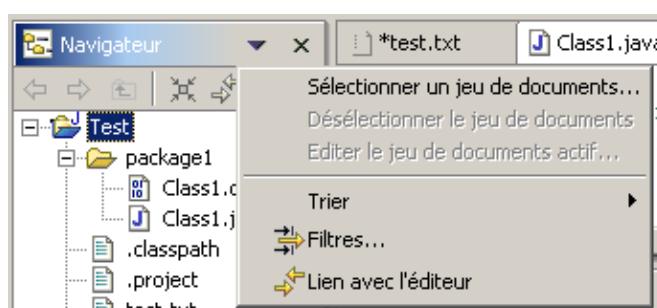


Les vues possèdent deux menus :

- un menu associé à la sous fenêtre activable en cliquant sur la petite icône en haut à gauche. Les options de ce menu concerne la sous fenêtre elle-même.

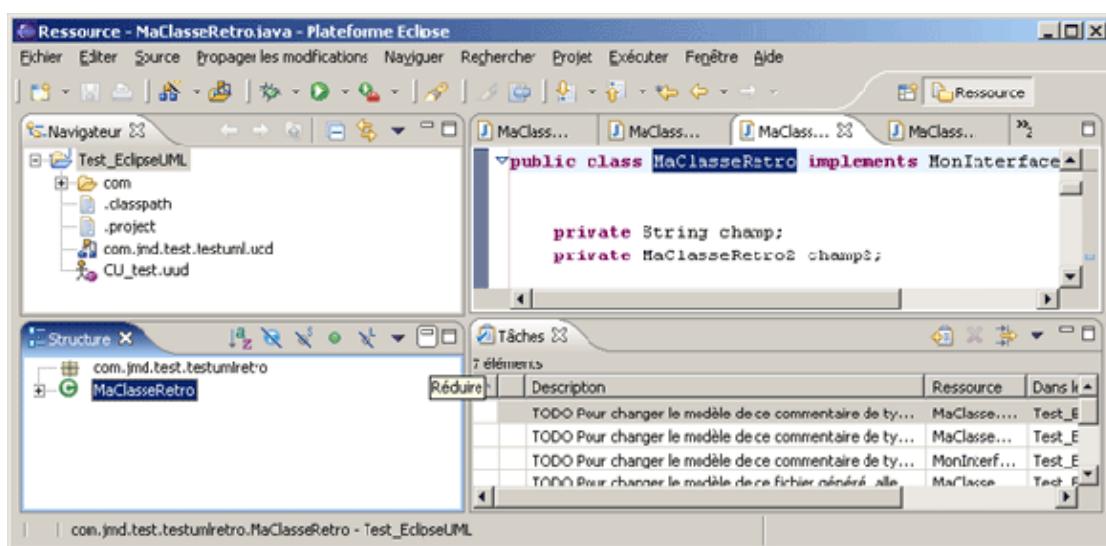


- le second menu est activable en cliquant sur l'icône en forme de triangle dont la base est en haut . Les options de ce menu concerne le contenu de la vue notamment le tri ou le filtre.

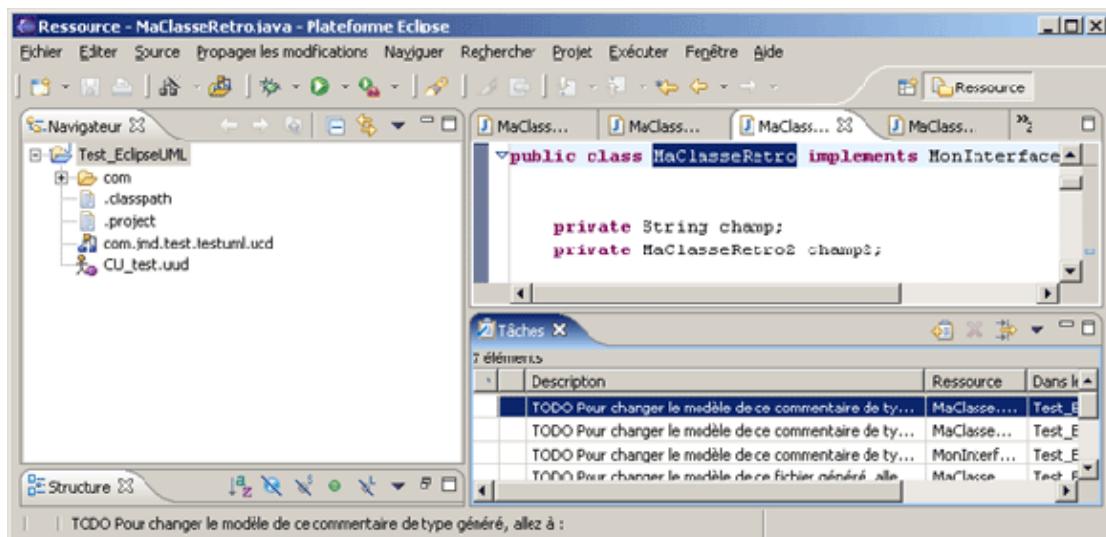




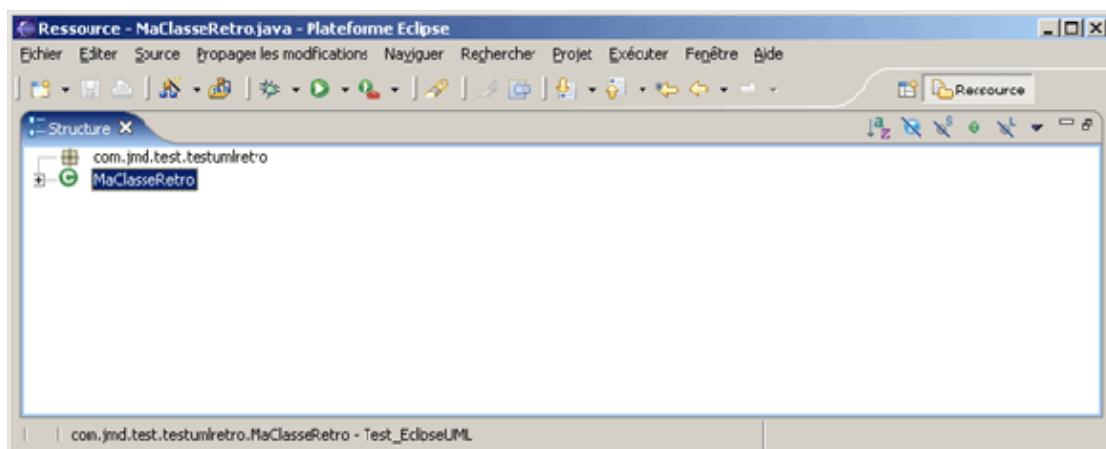
Eclipse 3.0 permet de réduire une vue ou d'agrandir une vue ou un éditeur.



Les vues peuvent être réduites en cliquant sur le bouton .



Inversément un clic sur le bouton  permet de restaurer la taille de la vue ou de l'éditeur. Un clic sur le bouton  permet d'agrandir la vue ou l'éditeur.



Cette option est particulièrement intéressante car elle évite d'avoir à modifier l'agencement des vues et des éditeurs pour maximiser la taille de l'un d'entre eux.

L'agrandissement et la restauration peuvent aussi être obtenues en double cliquant dans la barre de titre de la vue ou de l'éditeur.

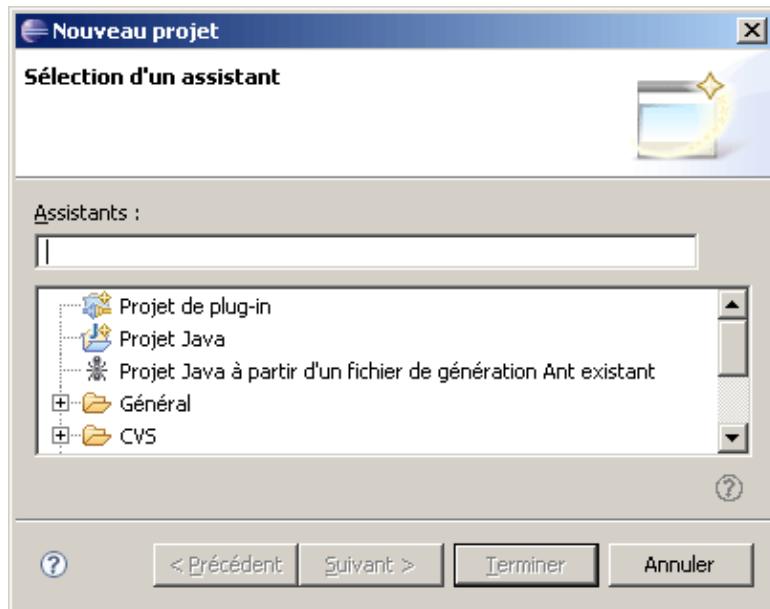
### 3.3. Les assistants

Eclipse propose de nombreux assistants pour permettre de faciliter la saisie des informations requises pour certaines tâches par l'utilisateur.



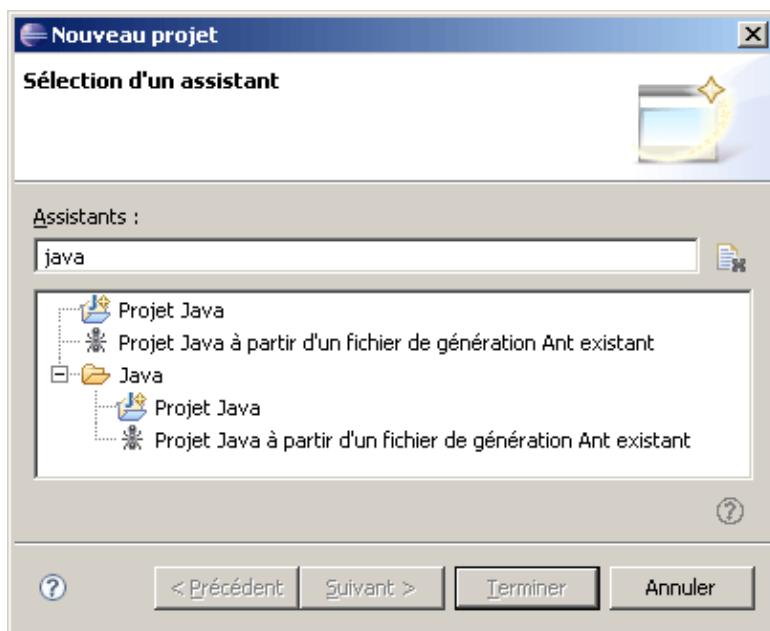
De nombreux assistants proposent un filtre qui permet de faciliter la sélection d'un élément.

Exemple lors de la création d'un nouveau projet



La zone de saisie permet de n'afficher que les projets correspondants au filtre saisi

Exemple :



Les messages d'erreur et d'avertissement sont affichés dans un panneau animé qui s'affiche en glissant du bas vers le haut dans l'entête de l'assistant.

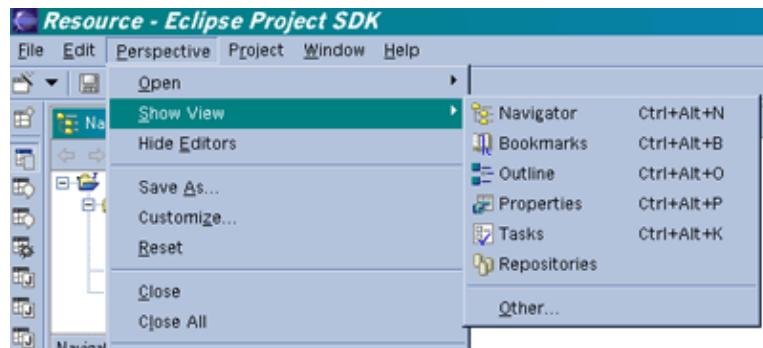
Exemple :



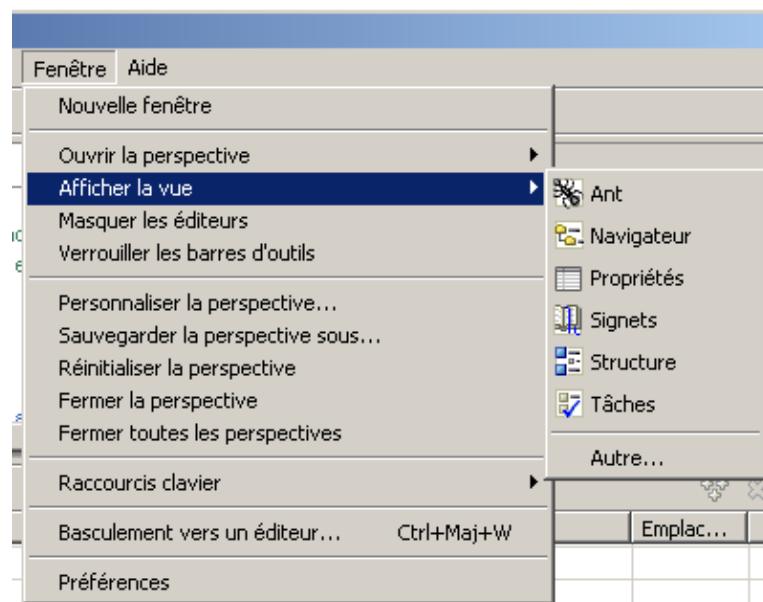
### 3.4. Organiser les composants de la perspective

Chaque perspective possède une organisation par défaut de ses sous fenêtres. Pour revenir à cette organisation par défaut, il faut utiliser l'option "Reset" du menu "Perspective" avec Eclipse 1.0 ou l'option "Réinitialiser la perspective" du menu "Fenêtre" dans Eclipse 2.0.

Avec Eclipse 1.0, l'option "Show View" du menu "Perspective" permet de visualiser une vue particulière qu'il suffit de sélectionner dans le sous menu.



Avec Eclipse 2.0, l'opération équivalente est effectuée en sélectionnant l'option "Afficher la vue" du menu "Fenêtre".



### 3.5. Fermer le plan de travail

Pour fermer le plan de travail et donc quitter Eclipse, il y a deux possibilités :

- Fermer la fenêtre du plan de travail
- Sélectionner l'option "Quitter" du menu "Fichier"

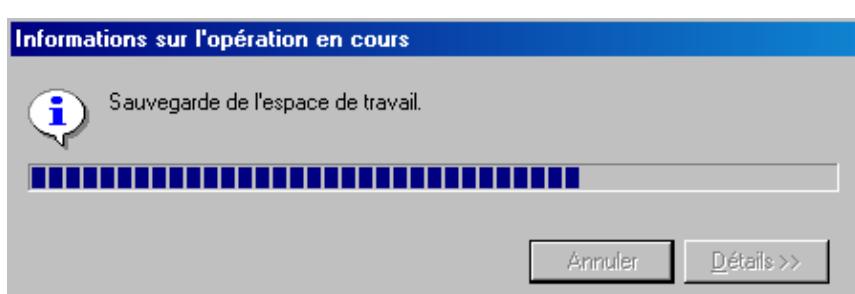
Une boîte de dialogue permet de confirmation la fermeture de l'application.



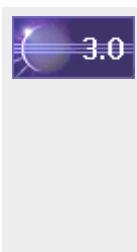
Si des ressources doivent être sauvegardées, une boîte de dialogue apparaît contenant ces ressources. Il faut indiquer celles qui doivent être enregistrées et cliquer sur le bouton "OK".



A sa fermeture, Eclipse enregistre certaines données dans l'espace de travail.



### 3.6. Exécution de traitements en arrière plan

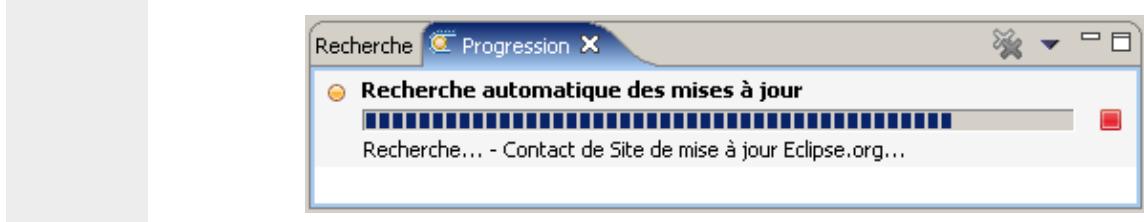


Certains fonctionnalités longues en temps de traitement sont réalisées en tâche de fond.

L'exécution d'un traitement en tâche de fond est signalée par un message dans la barre de statut



La vue « Progression » affiche des informations sur l'état d'avancement des traitements



## 4. L'espace de travail (Workspace)

# Chapitre 4

L'espace de travail est l'entité qui permet de conserver les projets et leur contenu. Physiquement c'est un répertoire du système d'exploitation qui contient une hiérarchie de fichiers et de répertoires. Il y a d'ailleurs un répertoire pour chaque projet à la racine de l'espace de travail.

Il est possible de parcourir cette arborescence et d'en modifier les fichiers avec des outils externes à Eclipse.

L'espace de travail contient tous les éléments développés pour le projet : il est possible de créer, de dupliquer, de renommer ou de supprimer des éléments. Ces opérations de gestion sont réalisées dans la vue "Navigateur" de la perspective "Ressource".

### 4.1. La perspective "Ressource"

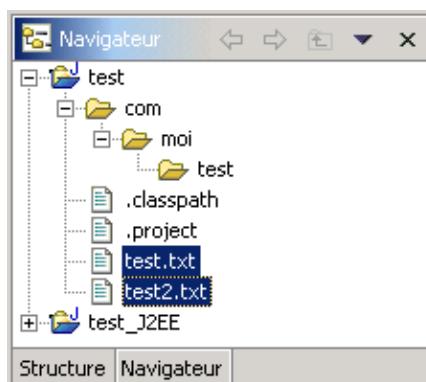
La perspective "Ressource" est la perspective qui s'ouvre par défaut au lancement d'Eclipse. Elle a pour but de gérer les différents éléments qui composent l'espace de travail : projets, dossiers et fichiers.

Par défaut, cette perspective contient les fenêtres suivantes :

- la vue "Navigateur" qui affiche les ressources (arborescence des fichiers) de l'espace de travail
- un éditeur qui permet d'éditionner une ressource sélectionnée dans la vue "Navigateur"
- la vue "Structure" qui permet d'obtenir une arborescence présentant les grandes lignes de certaines ressources en cours de traitement
- la vue "Tâches" qui affiche une liste de tâche à effectuer

#### 4.1.1. La vue "Navigateur"

Dans l'espace de travail, chaque projet contient une hiérarchie composée de dossiers et de fichiers. La vue "Navigateur" permet de présenter, de naviguer dans l'arborescence et de sélectionner une ressource.

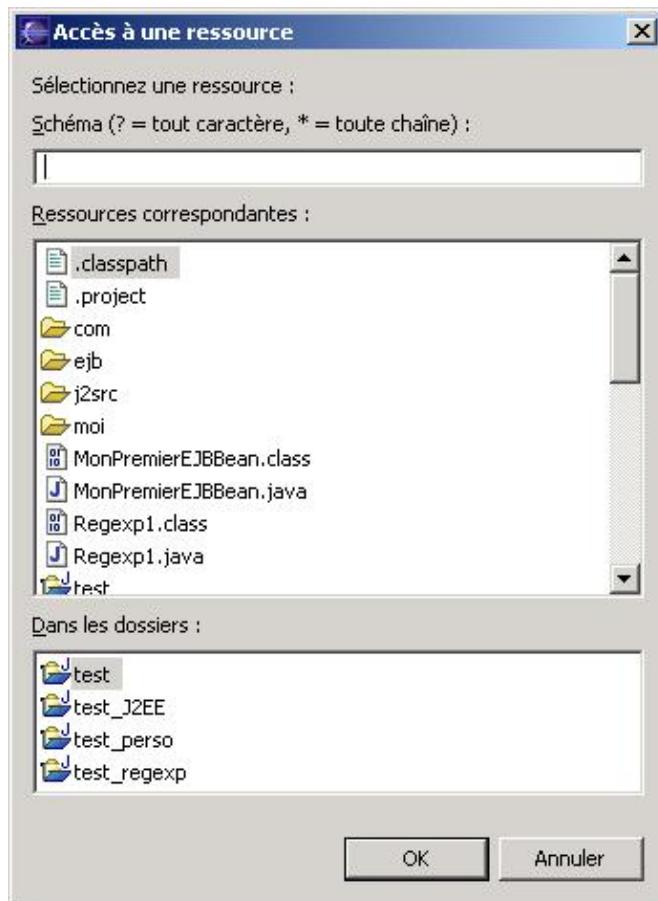


A partir de la vue "Navigateur", il est possible d'ouvrir le fichier sélectionné dans un éditeur :

- avec l'éditeur par défaut associé au type du fichier, il suffit de double cliquer sur le fichier dans le navigateur ou d'utiliser l'option "Ouvrir" du menu contextuel.
- avec un autre éditeur en utilisant l'option "Ouvrir Avec" du menu contextuel

L'association d'un type de fichier avec un éditeur peut être faite dans les préférences.

La vue "Navigateur" contient une option particulièrement pratique pour retrouver une ressource : l'outil "Accéder à". Cet outil permet à partir d'un motif (Pattern) de retrouver les ressources qui respectent le motif dans leur nom. L'option "Accéder à / Ressource" du menu contextuel de la vue "Navigateur" permet d'ouvrir une boîte de dialogue contenant l'outil.



Au fur et à mesure de la saisie du motif, la liste des fichiers correspondant s'affiche.

Il suffit de choisir le fichier et de cliquer sur le bouton "OK" pour fermer la boîte de dialogue et sélectionner le fichier dans la vue "Navigateur".

Par défaut, la vue "Navigateur" affiche tous les projets contenus dans l'espace de travail. Il est possible de limiter la vue à la hiérarchie d'un projet ou d'un dossier en le sélectionnant et en utilisant l'option "Suivant" du menu contextuel.

Les boutons permettent de passer d'un mode à l'autre.

Le menu contextuel propose aussi des options pour copier, déplacer, renommer et supprimer une ressource.

## 4.2. La création de nouvelles entités

Dans Eclipse, on peut créer différents types d'entités qui seront stockées dans l'espace de travail :

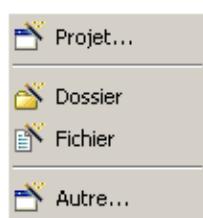
- des projets
- des répertoires pour organiser les projets
- des ressources de différents types qui sont des fichiers

Il existe plusieurs façons de créer ces entités :

- l'option « Nouveau » du menu « Fichier»
- l'option « Nouveau » du menu contextuel de la vue "Navigateur"
- le bouton « Assistant nouveau » dans la barre d'outils

La création est réalisée grâce à un assistant dont le contenu est dynamique en fonction de l'élément à créer.

L'option "Nouveau" du menu "Fichier" ou du menu contextuel de la vue "Navigateur" propose le même sous menu :

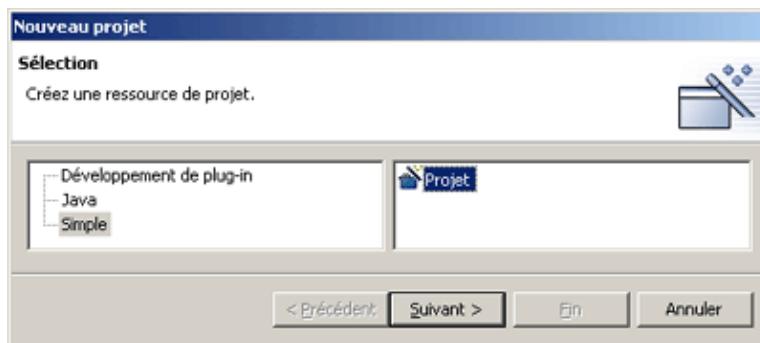


Il est ainsi possible de créer rapidement un projet, un répertoire ou un fichier. Si le fichier est d'un type particulier, un clic sur l'option "Autre" ouvre un assistant qui permet sur sa première page de sélectionner le type de l'entité à créer.

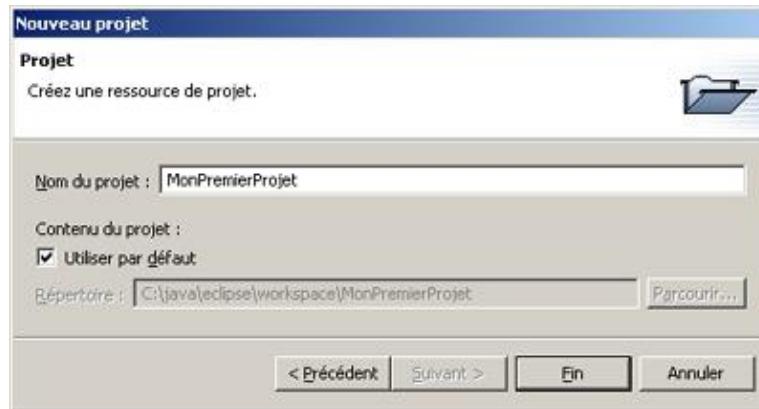
### 4.2.1. La création d'un projet

Le projet est l'unité de base de l'espace de travail. Chaque ressource doit être incluse directement dans un projet ou indirectement dans un répertoire appartenant à un projet.

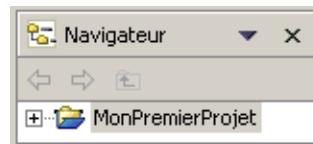
L'assistant permet de sélectionner le type du projet à créer. Il suffit alors de sélectionner la famille, le type du projet et de cliquer sur le bouton "Suivant".



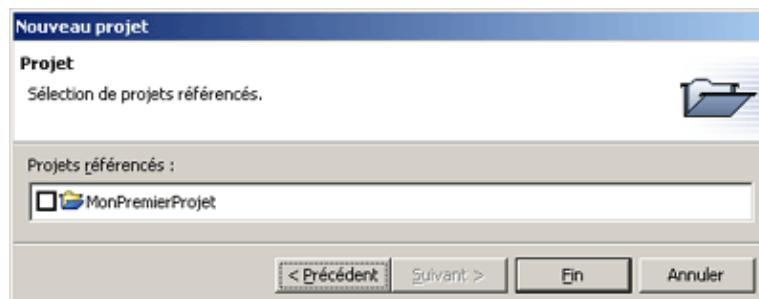
La création se fait grâce à un assistant qui demande le nom du nouveau projet. Ce nom ne doit pas contenir de blanc ou des caractères non alphabétiques.



Un clic sur le bouton "Fin" déclenche la création du projet. Le projet apparaît dans la vue Navigateur.



Si l'espace de travail contient déjà plusieurs projets, il est possible d'associer un ou plusieurs de ceux ci avec le nouveau en cours de création. Pour réaliser cette association, il suffit de cliquer sur le bouton "Suivant" pour afficher le second volet de l'assistant. Il suffit de cocher les projets concernés.



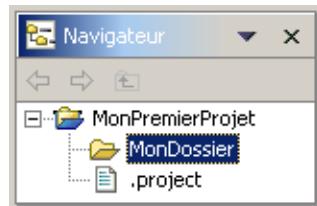
Il est possible de créer des projets particuliers selon le type d'applications à développer.

#### 4.2.2. La création d'un répertoire

L'assistant de création de répertoire permet de créer un répertoire dans un projet. Par défaut, c'est le projet courant qui est sélectionné.

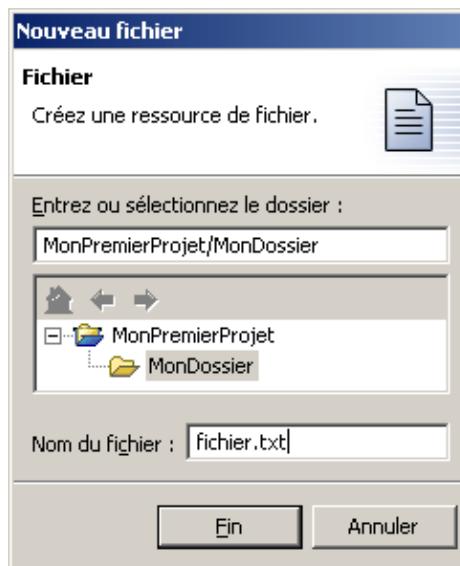


Il suffit ensuite de saisir le nom sans espace ni caractère non alphabétique et de cliquer sur le bouton "Fin". Le nouveau répertoire apparaît dans la vue "Navigateur".

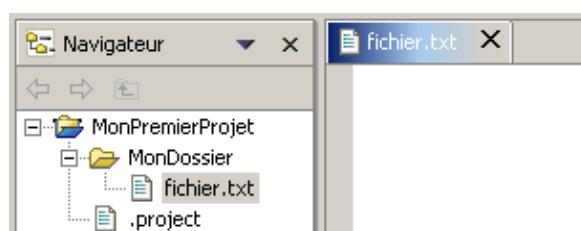


### 4.2.3. La création d'un fichier

L'assistant de création de fichiers permet de choisir le projet et le répertoire dans lequel le fichier sera créé. Une fois cette localisation choisie il suffit de saisir le nom du fichier, son extension et de cliquer sur le bouton "Fin". Ce nom ne doit pas contenir de blanc ou des caractères non alphabétiques.



Si un éditeur est associé au type du nouveau fichier, l'éditeur est ouvert avec le nouveau fichier.



### 4.3. La duplication d'un élément

Dans la vue "Navigateur", pour dupliquer un élément, le plus simple est de faire un cliquer/glisser de l'élément en maintenant la touche Ctrl enfoncee vers son répertoire de destination dans le navigateur.

Il est aussi possible de sélectionner l'élément dans la vue "Navigateur" et d'effectuer une des opérations suivantes :

- appuyer sur la combinaison de touches Ctrl + C
- sélectionner l'option "Copier" du menu contextuel

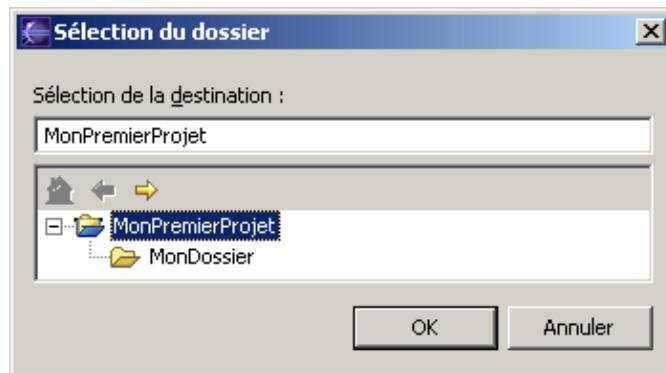
Il suffit alors de sélectionner dans la vue "Navigateur" le répertoire de destination et d'effectuer une des opérations suivantes :

- appuyer sur la combinaison de touches Ctrl + V
- sélectionner l'option "Coller" du menu contextuel

## 4.4. Le déplacement d'un élément

Dans la vue "Navigateur", pour déplacer un élément, le plus simple est de faire un cliquer/glisser de l'élément vers son répertoire de destination dans la vue "Navigateur".

Il est aussi possible de sélectionner l'option "Déplacer" du menu contextuel associé à cet élément et de sélectionner le répertoire de destination dans la boîte de dialogue.

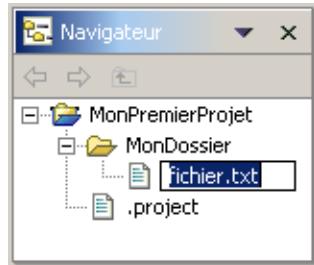


Si le répertoire de destination sélectionné est identique au répertoire d'origine du fichier, une message est affiché.



## 4.5. Renommer un élément

Pour nommer un élément, il suffit de sélectionner l'élément dans la vue "Navigateur", de sélectionner l'option "Renommer" du menu contextuel, saisir le nouveau nom et appuyer sur la touche "entrée".



## 4.6. La suppression d'un élément

Pour supprimer un élément, il suffit de le sélectionner dans la vue "Navigateur" et d'effectuer une des actions suivantes :

- choisir l'option "Supprimer" du menu contextuel de l'élément
- appuyer sur la touche Suppr
- choisir l'option "Supprimer" du menu "Editer"

Une boîte de dialogue demande de confirmer la suppression.



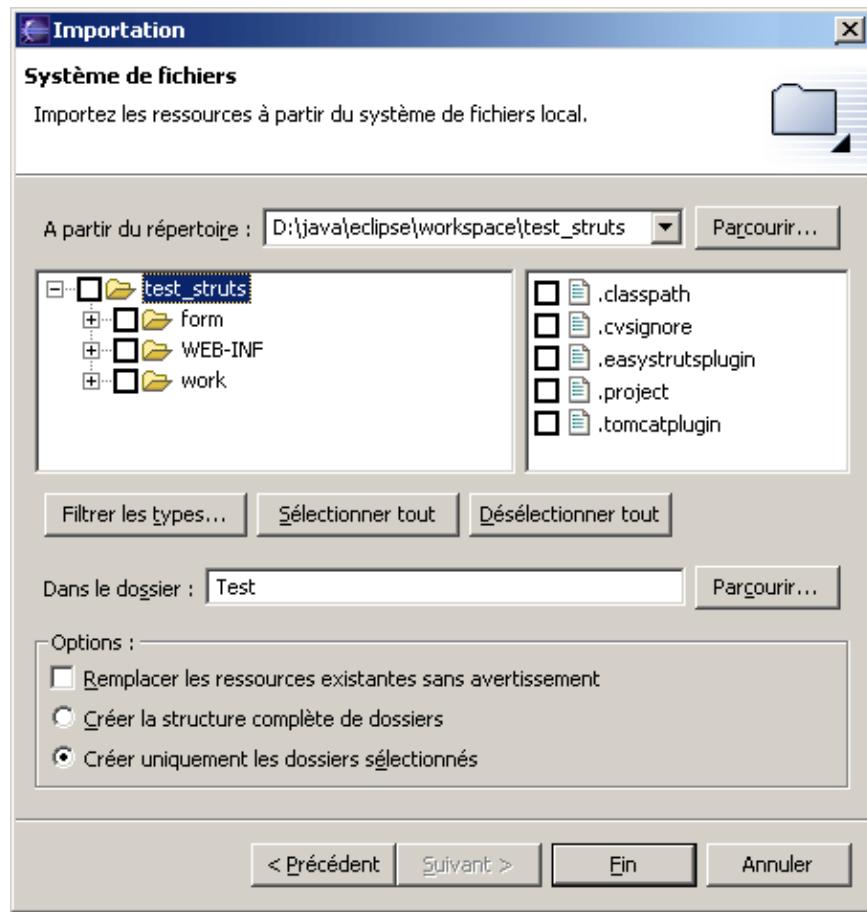
## 4.7. L'importation

L'importation permet d'inclure dans l'espace de travail un certains nombre de fichiers externes. Attention, l'importation ne peut se faire que dans un projet existant.

Il faut utiliser le menu "Fichier/Importer"



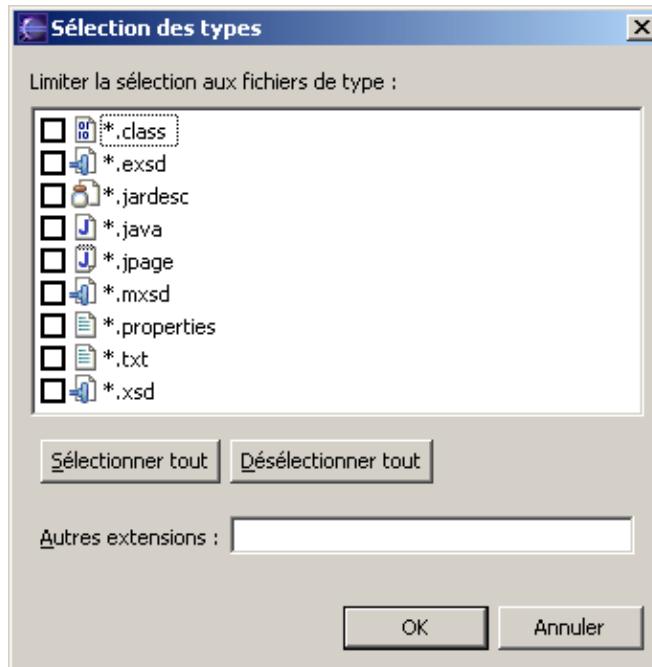
Sélectionnez le type de la source d'importation et cliquez sur le bouton "Suivant"



Selectionnez le répertoire, puis cochez chacun des éléments concernés.

Il est très important de vérifier et de modifier si nécessaire le répertoire de destination qui doit être l'un des projets de l'espace de travail.

En cliquant sur le bouton "Filtrer les types ...", une boîte de dialogue permet de sélectionner les fichiers concernés à partir de leurs extensions.



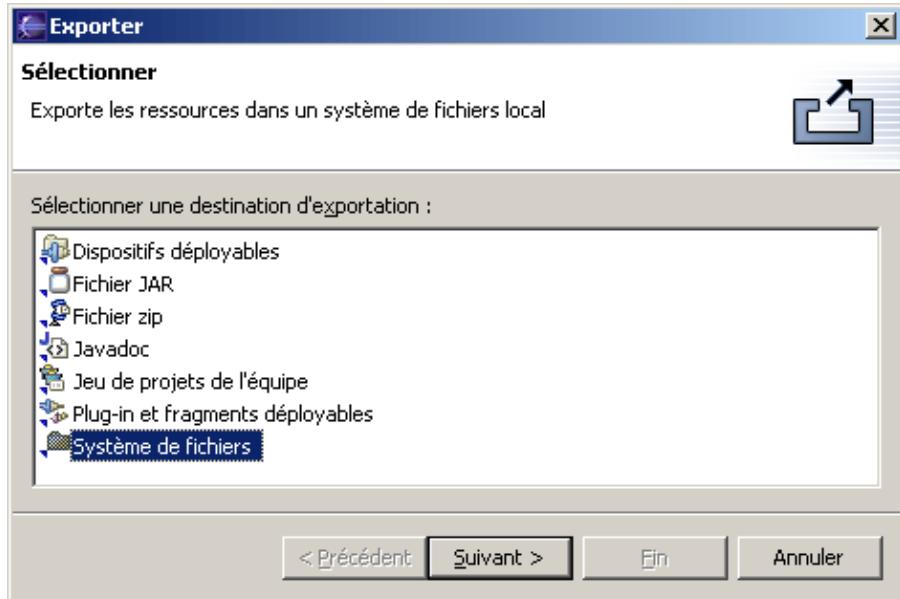
Il suffit de sélectionner les extensions parmi celles proposées, de saisir éventuellement d'autres extensions et de cliquer sur le bouton "OK". Le filtre est alors directement appliqué sur la sélection.

Une fois la sélection terminée, il suffit de cliquer sur le bouton "Fin" pour lancer l'importation.

Au cas où une ressource existerait déjà dans la destination, un message demande la confirmation du remplacement.

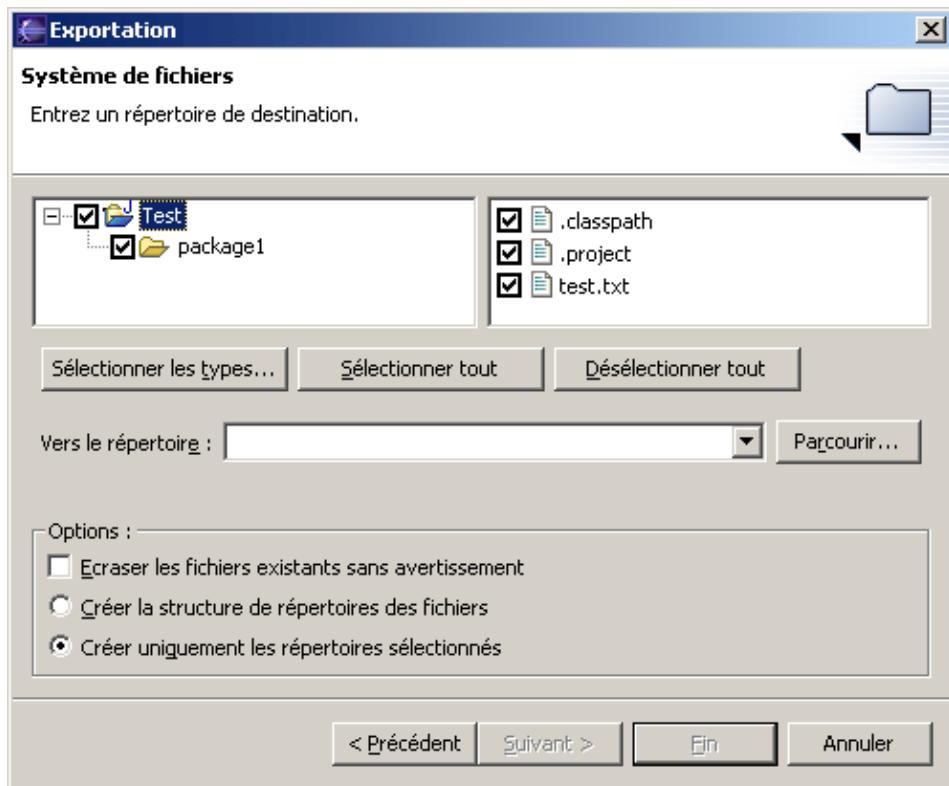
## 4.8. L'exportation

Pour exporter tout ou partie du workspace, il faut utiliser le menu "Fichier/Exporter".



L'assistant demande de sélectionner le format d'exportation.

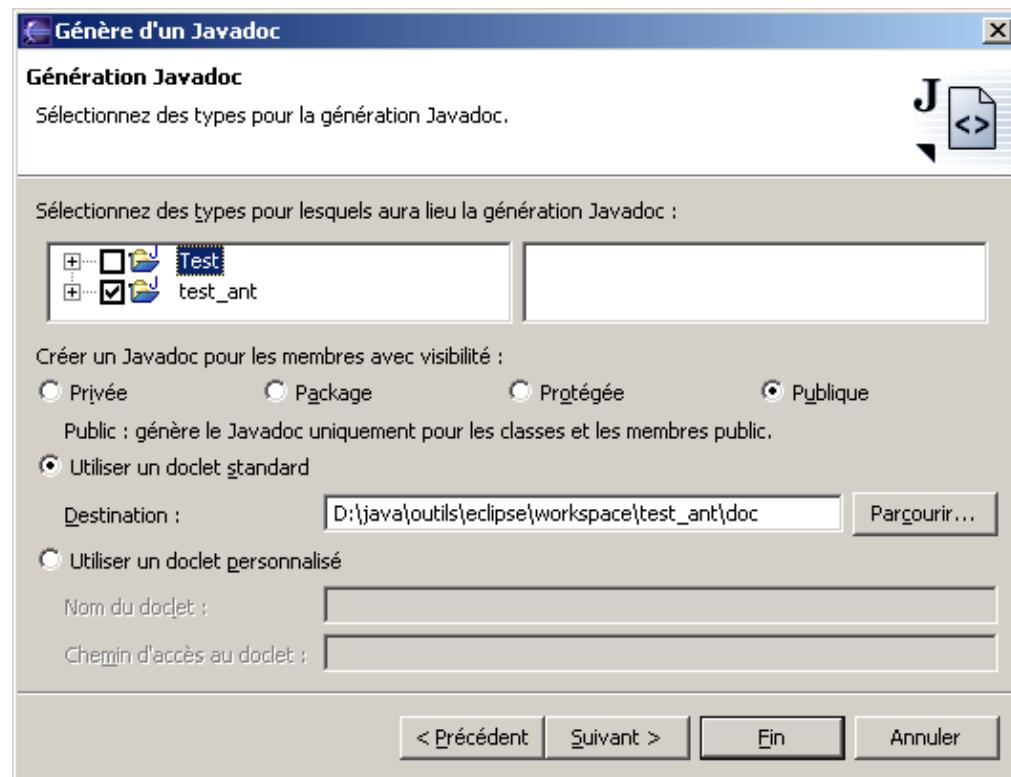
Si le format choisi est "Système de fichiers", l'assistant demande les informations nécessaires : les fichiers à exporter et le répertoire de destination



Attention : pour que la structure des répertoires soit conservée dans la cible, il faut obligatoirement que les

répertoires soient sélectionnés.

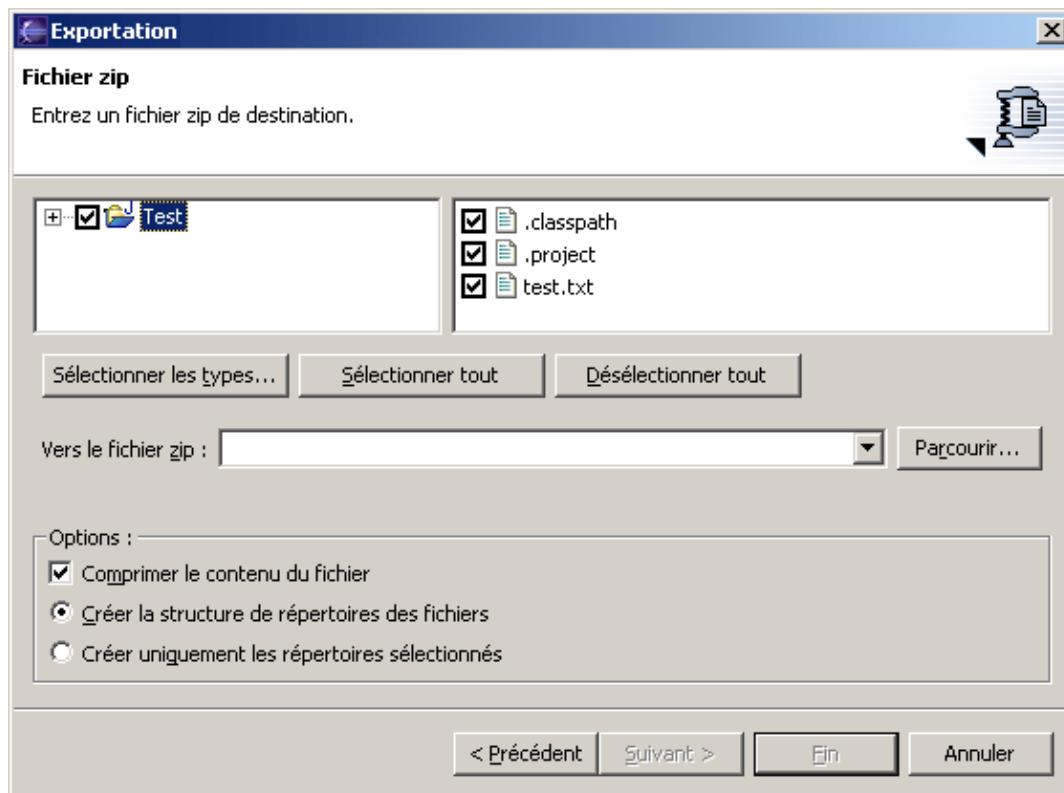
Si le format choisi est "Javadoc", l'assistant demande les informations nécessaires : les fichiers à exporter, le répertoire de destination et les options à utiliser lors de la génération



Les deux pages suivantes permettent de préciser des options pour l'outil Javadoc.

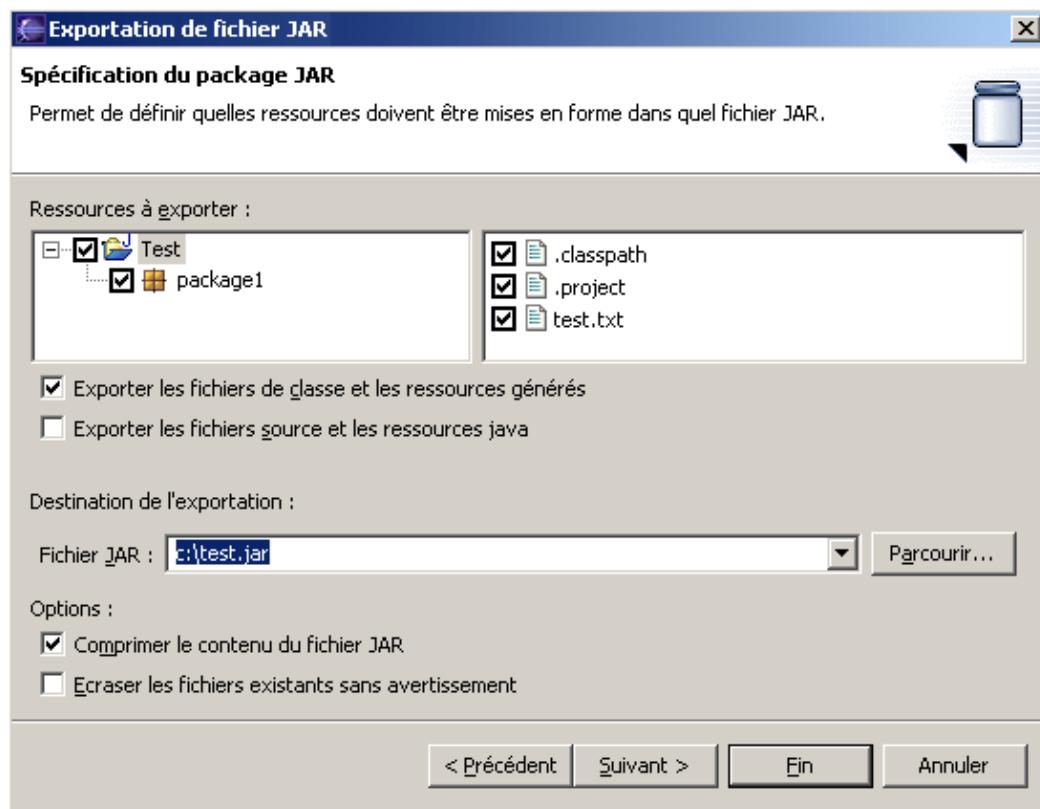
Un clic sur le bouton "Fin" permet de générer la documentation.

Si le format choisi est "Fichier Zip", l'assistant demande les informations nécessaires : les fichiers à exporter et le nom du fichier zip à générer.

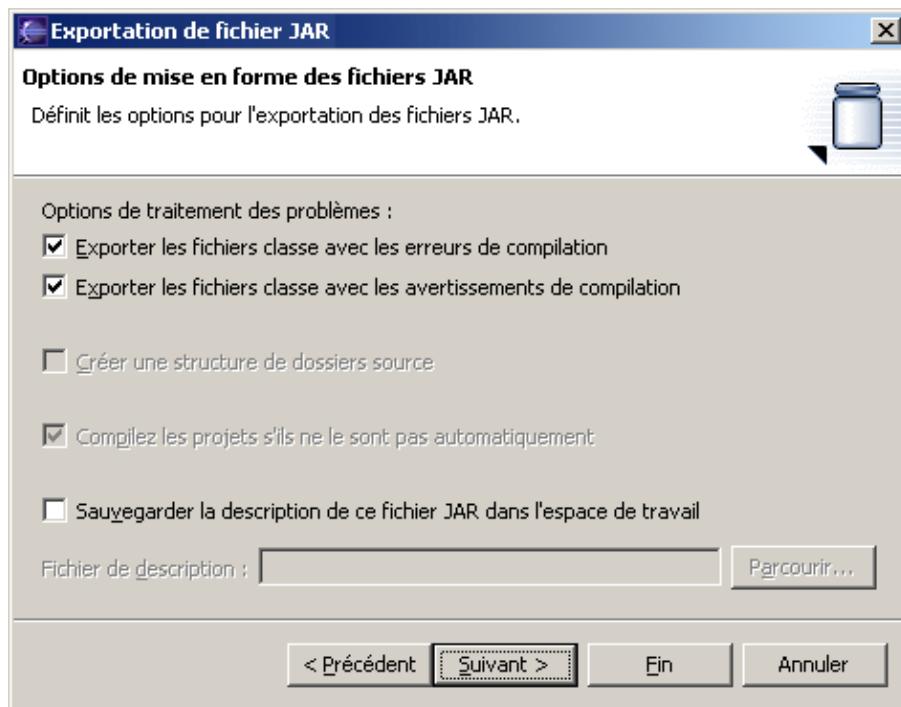


Un clic sur le bouton "Fin" permet de créer le fichier zip contenant tous les éléments sélectionnés.

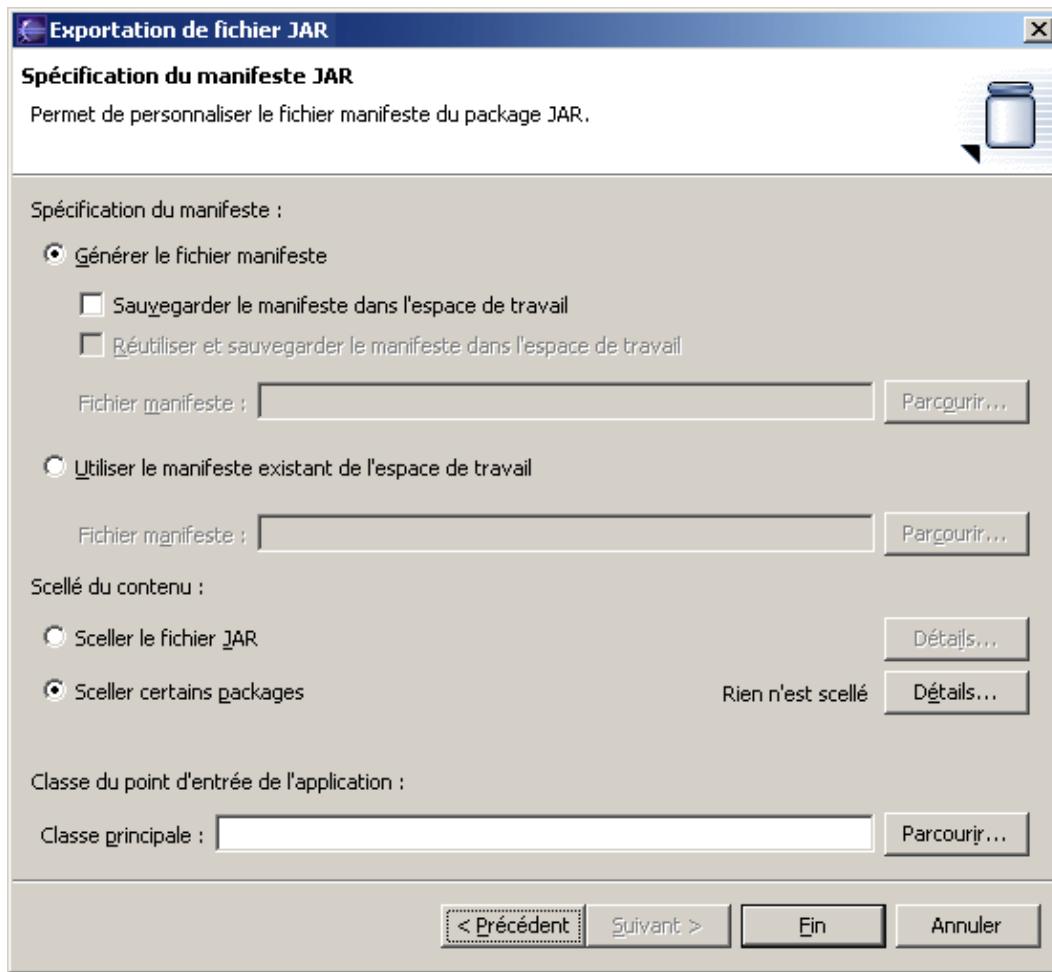
Si le format est "Fichier Jar", l'assistant demande les informations nécessaires : le ou les projets à exporter, le fichier jar à créer et les options à utiliser.



Un clic sur le bouton "Suivant" affiche une nouvelle page de l'assistant pour préciser certaines options.



Un clic sur le bouton "Suivant" affiche une nouvelle page de l'assistant pour préciser le fichier manifest.

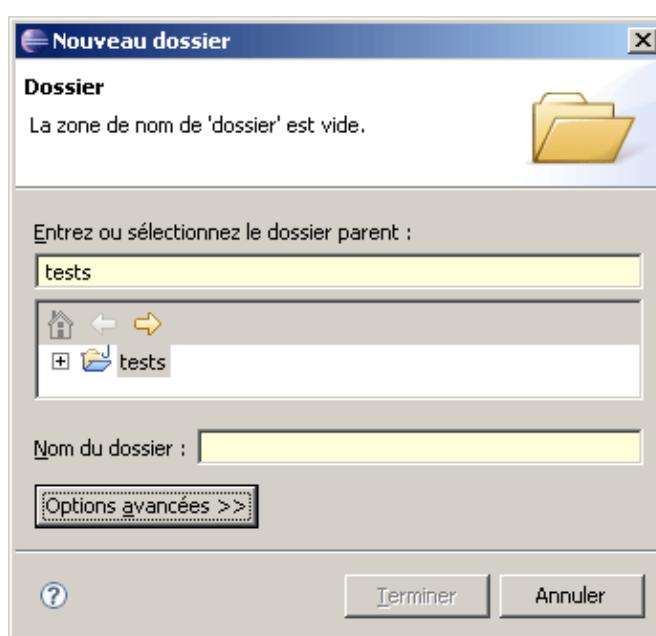


## 4.9. Lier des ressources

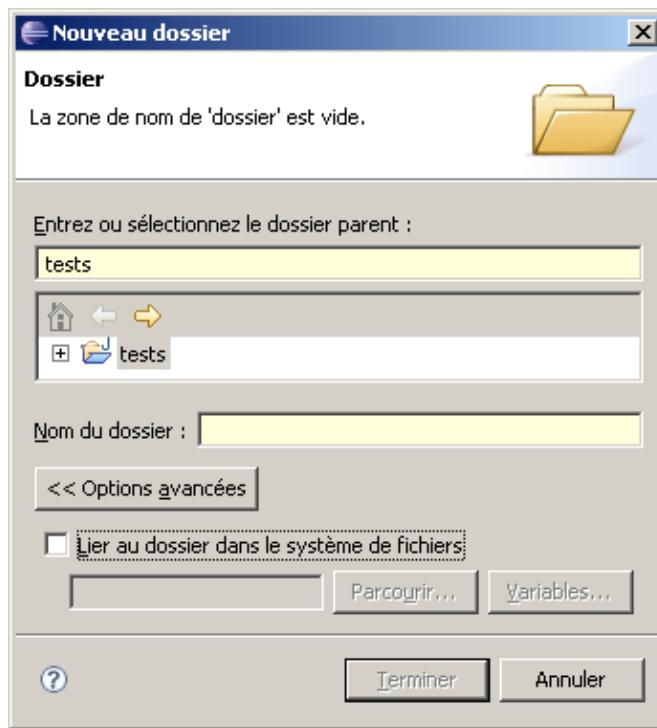


Il est possible de lier des ressources au projet.

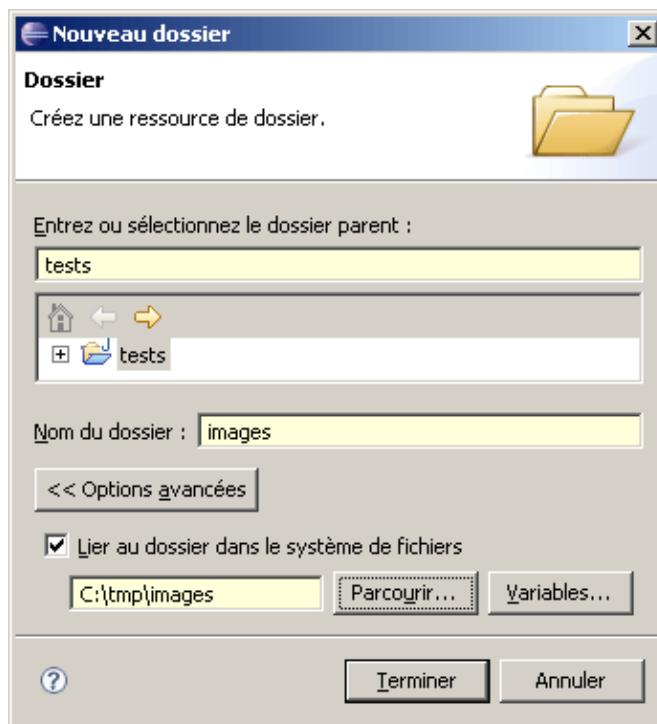
Exemple avec un répertoire : créez une nouvelle entité de type dossier



Cliquez sur le bouton « Options avancées »



Cochez l'option « Lier au dossier dans le système de fichiers », cliquez sur le bouton « Parcourir » pour sélectionner le répertoire concerné.



Cliquez sur le bouton « Terminer »



La ressource liée apparaît avec une petite icône ☒

## 4.10. L'option « Fermer les projets non associés »



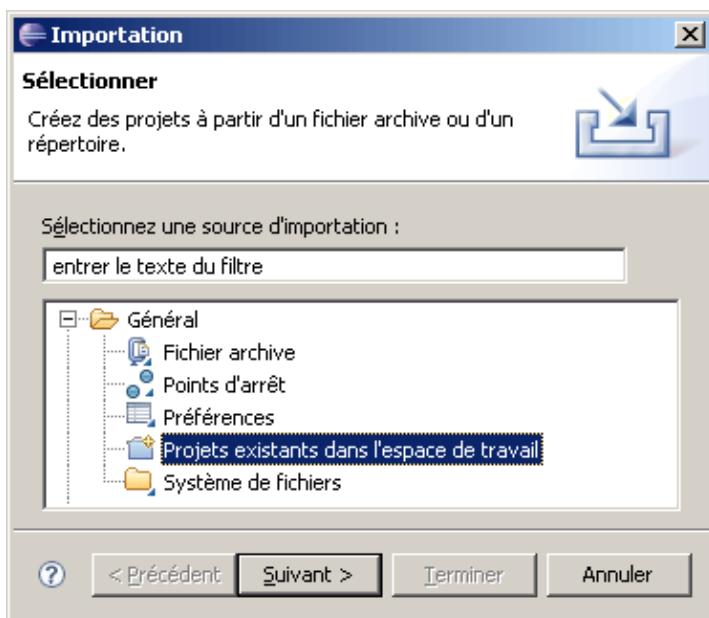
L'option « Fermer les projets non associés » du menu contextuel d'un projet permet de fermer les éléments ouverts dans le plan de travail qui ne concernent pas le projet courant.

Cette option n'est accessible que si au moins un élément concerné est ouvert.

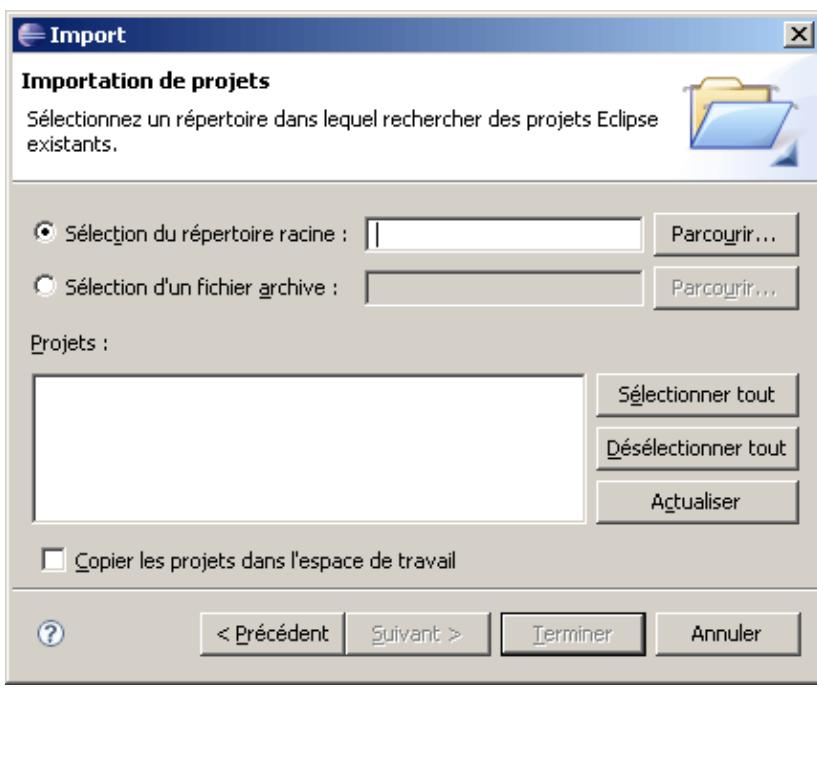
## 4.11. Importer une copie d'un projet



Lors de l'importation d'un projet existant (en utilisant l'option « Importer » du menu principal et en sélectionnant « Général / Projets existants dans l'espace de travail », il est possible de créer une copie du projet.



Cliquez sur le bouton « Suivant »



Pour créer une copie du projet, il faut cocher la case « Copier les projets dans l'espace de travail ».

## 5. Les fonctions pratiques du plan de travail

# Chapitre 5

Eclipse fournit dans le plan de travail plusieurs outils très pratiques qui permettent :

- d'effectuer des recherches
- de gérer une liste de tâches à faire
- de gérer une liste de signets d'éléments
- de comparer des éléments

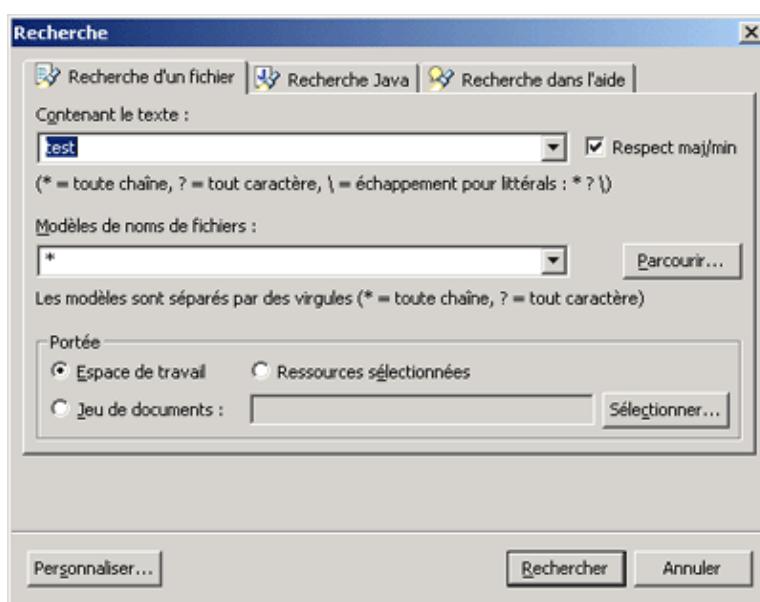
### 5.1. La fonction de recherche

Cette fonction de recherche permet d'obtenir une liste d'éléments qui contiennent une chaîne désignée par un motif.

Elle peut se faire dans tous les fichiers, dans les fichiers source Java ou dans l'aide en ligne.

#### 5.1.1. La recherche dans les fichiers

Pour effectuer une recherche, il faut cliquer sur l'icône  de la barre d'outils du plan de travail. Une boîte de dialogue permet de saisir les critères de recherche.



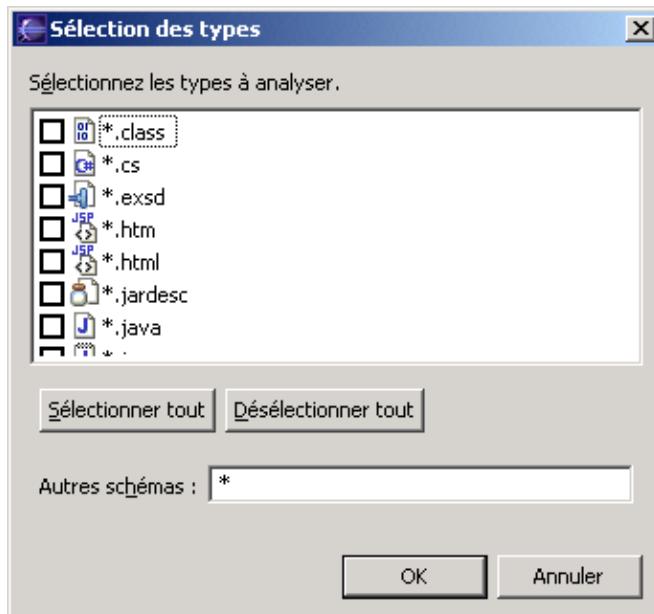
L'onglet "Recherche d'un fichier" permet de faire une recherche de fichiers contenant un texte respectant un motif. Ce motif peut être saisi ou sélectionné dans la liste déroulante à partir des précédents motifs recherchés.

Il est possible de saisir les caractères recherchés et d'utiliser trois caractères dont la signification est particulière :

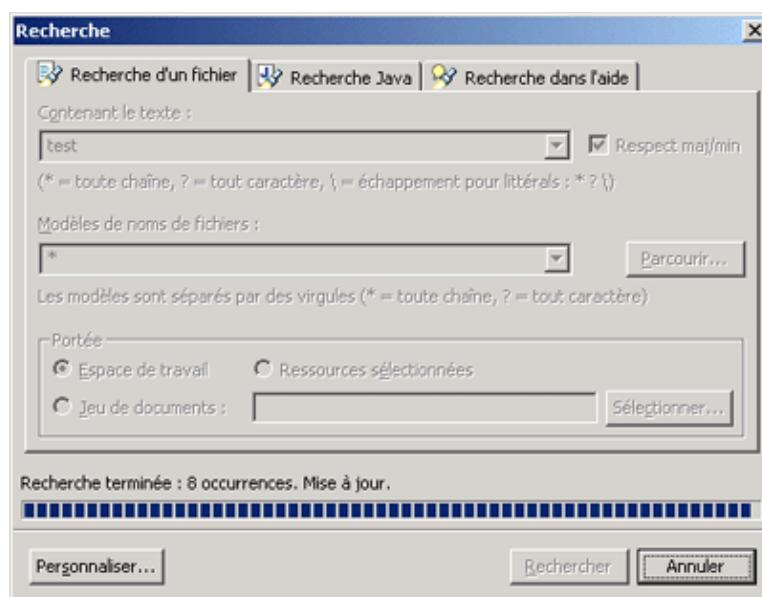
- \* : représente zéro ou plusieurs caractères quelconques
- ? : représente un caractère quelconque
- \ : permet de déspecialiser le caractère \*, ?, et \

Il est possible de vouloir tenir compte de la casse en cochant la case "Respect maj/min".

Il est aussi possible de restreindre la recherche à certains fichiers en précisant un motif particulier. Un clic sur le bouton "Parcourir" ouvre une boîte de dialogue qui permet de sélectionner un ou plusieurs types prédefinis.



Pour lancer la recherche, il suffit de cliquer sur le bouton "Rechercher".



Une barre de progression indique l'évolution de la recherche et le nombre de fois où le motif est trouvé. Un clic sur le bouton "Annuler" permet d'interrompre la recherche.



Il est possible d'utiliser les expressions régulières pour effectuer une recherche. Pour cela, il faut cocher la case correspondante dans la boîte de dialogue.

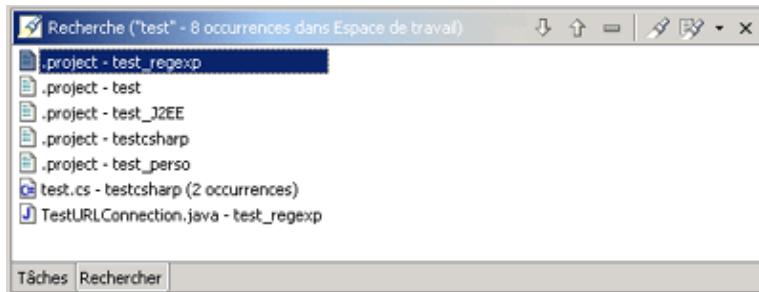


Dans la zone de saisie du mot à rechercher, l'appui sur la combinaison de touches Ctrl+espace ouvre un assistant qui facilite la saisie d'une expression régulière.



### 5.1.2. L'exploitation des résultats de recherche

Une fois la recherche terminée, la vue "Recherche" affiche les éléments contenant le motif et le nombre de fois où le motif a été trouvé dans chaque élément.

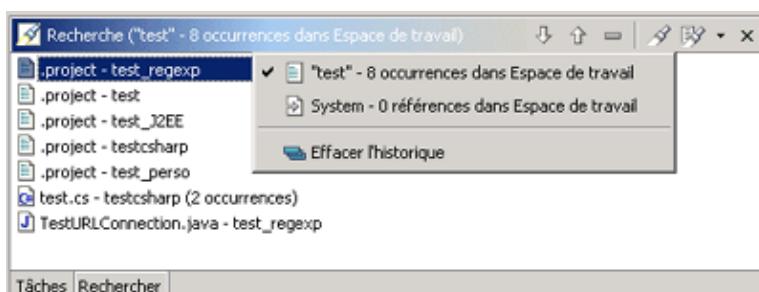


Le bouton permet de passer à l'occurrence suivante quelque soit l'élément qui la contienne. Lors du changement de l'élément qui contient l'occurrence, celui ci est ouvert dans l'éditeur.

Il est possible de supprimer une ou plusieurs occurrences dans la vue "Recherche". Le menu contextuel propose plusieurs options en fonction de la situation actuelle :

- "Supprimer l'occurrence sélectionnée" : cette option permet de supprimer l'occurrence courante de l'élément en cours
- "Supprimer les occurrences en cours" : permet de supprimer toute les occurrences de l'élément et l'élément de la liste
- "Supprimer toutes les occurrences" : permet de supprimer tous les éléments

La vue "Recherche" affiche le résultat de la recherche courante mais elle mémorise aussi les précédentes recherches. Pour afficher les résultats des précédentes recherches, il suffit de sélectionner la recherche en utilisant le bouton . Un menu affiche la liste des précédents motifs de recherche et le nombre d'occurrences trouvées. La recherche courante est cochée.

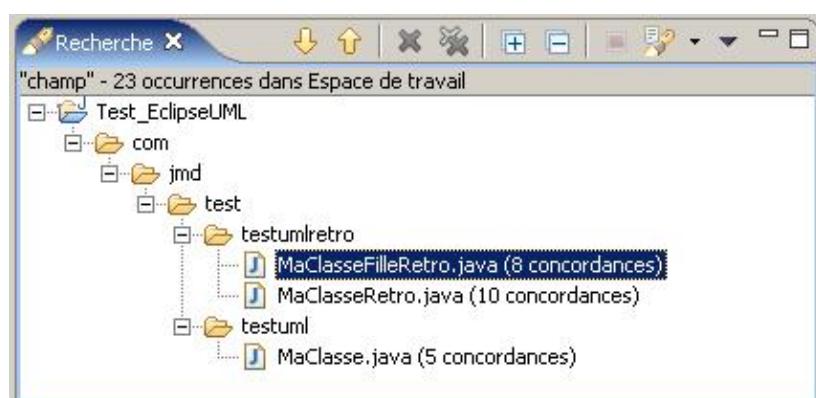


Il est toujours possible de réitérer la recherche en utilisant l'option "Nouvelle recherche" du menu contextuel de la vue.

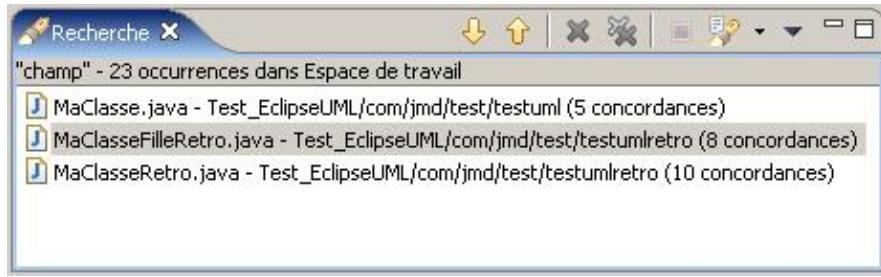


La vue « Recherche » propose deux façons d'afficher les résultats d'une recherche :

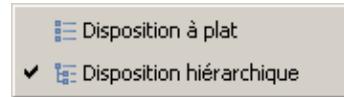
- arborescence (par défaut)



- tableau



Le bouton ouvre un menu déroulant qui permet de sélectionner le mode d'affichage



## 5.2. La liste des tâches

La vue "Tâches" affiche et permet de gérer une liste de tâches à faire. Ces tâches peuvent être de plusieurs types :

- des actions à réaliser
- des erreurs de compilation à corriger
- des points d'arrêt pour le débogage

Ces tâches peuvent être ou non associées à un élément de l'espace de travail. Par exemple, une erreur de compilation est associée à un fichier source.

Lorsqu'une tâche est associée à un élément, le nom de cet élément apparaît dans la colonne "Ressource" et sa localisation dans l'espace de travail dans la colonne "Dans le dossier".

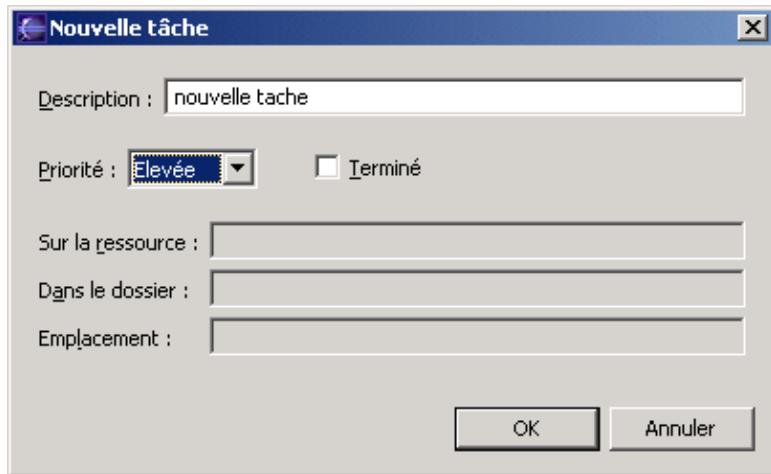
Tâches (3 éléments)					
C	I	Description	Ressource	Dans le dossier	Emplacement
		L'importation javax.ejb ne peut pas être résolue	MonPremierEJBBean.java	test_J2EE/com/moi/ejb	ligne 3
		SessionBean ne peut pas être résolue ou ne corre...	MonPremierEJBBean.java	test_J2EE/com/moi/ejb	ligne 10 dans ...
		Erreur de syntaxe sur le mot clé "else", "case", "d...	TestAssert1.java	test_perso	ligne 21 dans ...

Il est possible d'accéder à l'élément associé à la tâche en double cliquant sur la tâche ou en sélectionnant l'option "Accéder à" du menu contextuel de la tâche. L'élément est ouvert dans l'éditeur avec le curseur positionné sur la ligne associée à la tâche.

### 5.2.1. La création d'une tâche

Pour créer une tâche qui ne soit pas associée à un élément, il suffit de cliquer sur le bouton de la vue.

Une boîte de dialogue permet de saisir la description et la priorité de la nouvelle tâche.



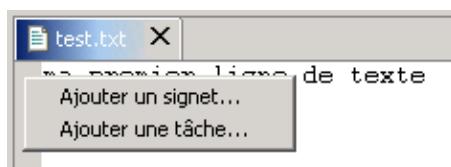
Un clic sur le bouton "OK" crée la nouvelle tâche.

Tâches (4 éléments)					
C	I	Description	Ressource	Dans le dossier	Emplacement
<input checked="" type="checkbox"/>	!	nouvelle tâche			
<input checked="" type="checkbox"/>		L'importation javax.ejb ne peut pas être résolue	MonPremierEJBBean.java	test_J2EE/com/moi/ejb	ligne 3
<input checked="" type="checkbox"/>		SessionBean ne peut pas être résolue ou ne corre...	MonPremierEJBBean.java	test_J2EE/com/moi/ejb	ligne 10 dans ...
<input checked="" type="checkbox"/>		Erreur de syntaxe sur le mot clé "else", "case", "d...	TestAssert1.java	test_perso	ligne 21 dans ...

### 5.2.2. La création d'une tâche associée à un élément

La création d'une tâche associée à un élément ne se fait pas dans la vue "Tâches" mais directement dans un éditeur qui contient l'élément. La tâche est associée à une ligne de l'élément.

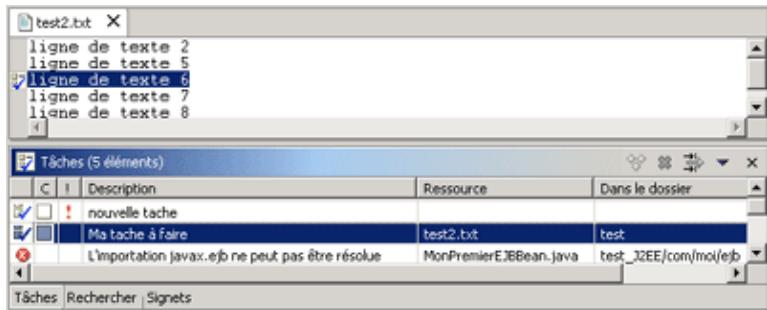
Dans la barre à gauche de l'éditeur, le menu contextuel contient l'option "Ajouter une tâche ..."



Une boîte de dialogue demande de saisir la description de la tâche et de sélectionner la priorité.



Un clic sur le bouton "OK" crée la tâche et une marque particulière apparaît sur la ligne concernée dans la barre de gauche de l'éditeur.



Cette marque reste associée à la ligne même si la position de la ligne dans le fichier change (par ajout ou suppression de lignes dans la ressource).

### 5.2.3. La suppression d'une tâche associée à un élément

Il suffit de sélectionner l'option "Supprimer une tâche" du menu contextuel associé à la marque de la tâche. La marque disparaît et la tâche est supprimée de la liste.

## 5.3. La liste des signets

Les signets (bookmarks) permettent de maintenir une liste d'éléments particuliers dans le but est de permettre d'y accéder rapidement. Pour afficher la vue "Signets", il faut sélectionner l'option "Afficher la vue / Signets" du menu "Fenêtre" du plan de travail.



A partir de la vue "Signets", pour ouvrir un élément dans l'éditeur, il y a trois possibilités :

- double cliquer sur le signet
- sélectionner l'option "Accéder à" du menu contextuel associé au signet
- cliquer sur le bouton une fois le signet sélectionné

Il est aussi possible à partir d'un signet de sélectionner l'élément dans la vue "Navigateur" en utilisant l'option "Afficher dans le navigateur" du menu contextuel.

### 5.3.1. La création d'un signet

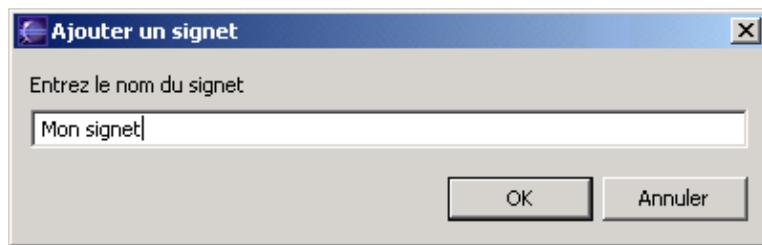
Un signet peut concerner un élément (un fichier) ou plus précisément une composante de cet élément (une position particulière dans le fichier).

Pour créer un signet sur un élément, il suffit de le sélectionner dans la vue "Navigateur" et de sélectionner l'option "Ajouter un signet" du menu contextuel.

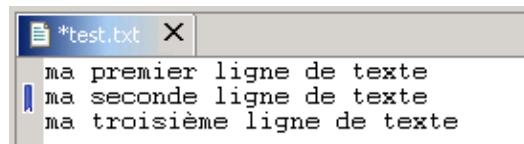
Pour créer un signet sur une ligne de l'élément, il suffit de positionner le curseur sur la ligne désirée dans l'éditeur et de sélectionner l'option "Ajouter un signet" du menu contextuel de la barre de gauche de l'éditeur.



Une boîte de dialogue demande de saisir la description.



Le signet est ajouté dans la liste des signets et une marque est affichée dans la barre de gauche de l'éditeur sur la ligne concernée.



### 5.3.2. La suppression d'un signet

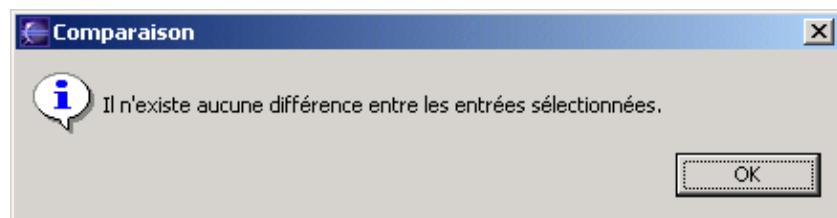
Pour supprimer un signet, il y a trois possibilités :

- dans la vue "Signets", sélectionner le signet et cliquer sur le bouton
- dans la vue "Signets", sélectionner le signet et l'option "Supprimer" du menu contextuel associé au signet
- dans l'éditeur, sélectionner l'option "Supprimer un signet" du menu contextuel associé à l'icône du signet dans la barre de gauche

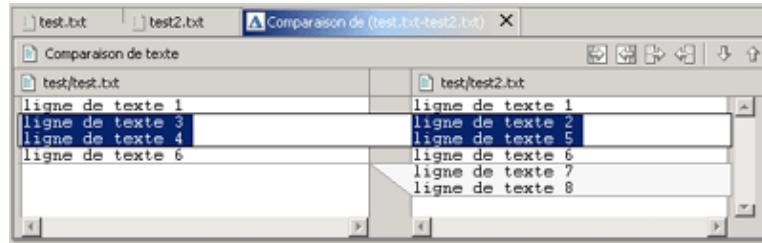
## 5.4. La comparaison d'éléments

Le plan de travail dispose d'un outil pratique pour comparer le contenu de deux éléments. Pour réaliser cette comparaison, il faut sélectionner ces deux éléments en maintenant la touche Ctrl enfoncee dans la vue "Navigateur" et sélectionner l'option "Comparer / Réciproquement" du menu contextuel.

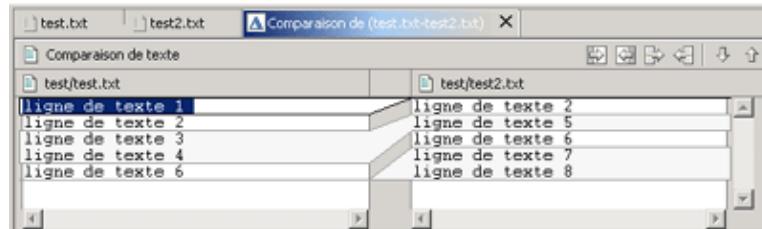
Si les deux fichiers sont identiques, une boîte de dialogue s'affiche :



Si les deux fichiers possèdent des différences, un éditeur particulier s'ouvre. Cet éditeur spécial pour les comparaisons, affiche chaque ligne des deux fichiers dans deux colonnes. Une colonne centrale permet de voir de façon graphique les différences grâce à des lignes.



Dans la barre centrale, les lignes en gris foncé sont identiques, les lignes en blanc sont des différences entre les deux fichiers.



La vue de comparaison de fichier contient une barre d'outils qui permet de naviguer dans les différences et de les reporter pour effectuer une synchronisation sélective.



La flèche vers le haut et le bas permet de naviguer dans les différences respectivement la suivante et la précédente. Les quatre premiers boutons permettent respectivement :

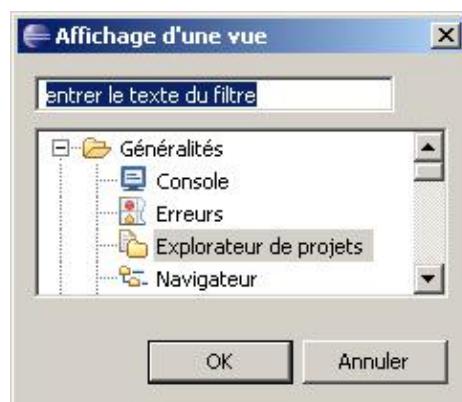
- de copier tout le document de gauche dans le document de droite
- de copier tout le document de droite dans le document de gauche
- de reporter la différence courante de gauche dans le document de droite
- de reporter la différence courante de droite dans le document de gauche

## 5.5. La vue « Explorateur de projets »



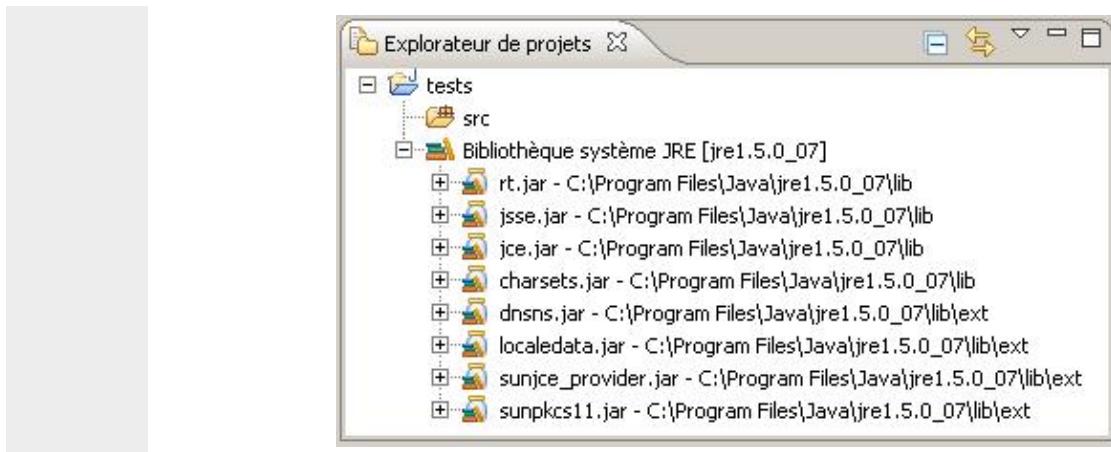
La vue explorateur de projets affiche le contenu des projets de façon hiérarchique en fonction du type de projet.

Pour l'afficher, il faut utiliser l'option « afficher la vue / autres » du menu principal de la fenêtre.



Il faut sélectionner l'élément « Explorateur de projets » dans « Généralités ».

Exemple avec un projet Java



## 6. L'aide dans Eclipse

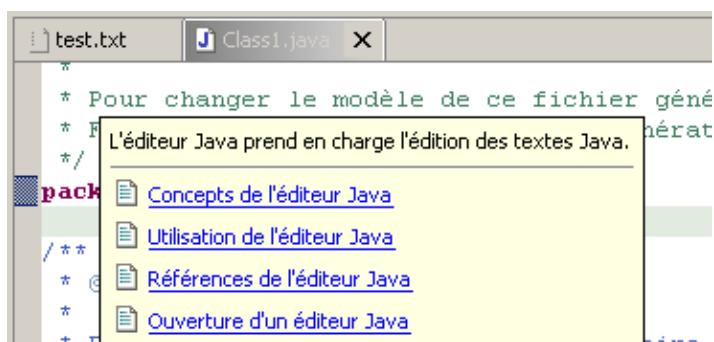
# Chapitre 6

### 6.1. L'aide en ligne

L'aide en ligne est disponible dans toute l'interface de l'IDE au moyen de la touche F1. Cette aide est contextuelle en fonction de l'élément sélectionné dans l'interface au moment de l'appui sur la touche.

Le choix des sujets se rapportant à l'élément courant est affiché dans une info bulle. Il suffit de cliquer sur l'élément sélectionné pour que l'aide en ligne correspondante s'affiche.

Exemple avec l'appui sur F1 dans l'éditeur de code Java



En appuyant sur la touche F1, une vue affiche l'aide contextuelle



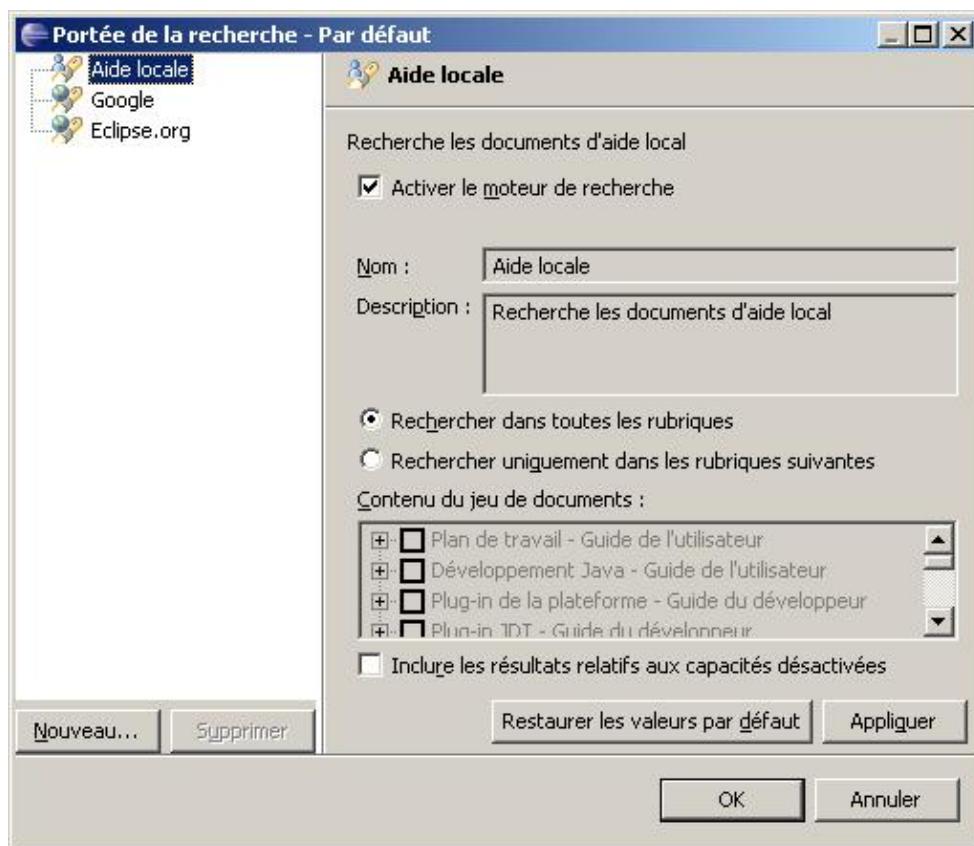
En cliquant sur « Recherche », il est possible de demander une recherche dans l'aide en ligne, dans Google et sur le site Eclipse.org. Il est aussi possible de demander cette recherche en utilisant directement l'option « Recherche » du menu « Aide ».



En cliquant sur « Portée de la recherche », il est possible de modifier la portée de la recherche par défaut.

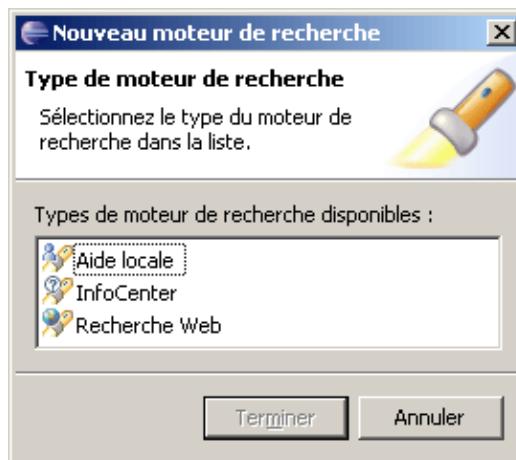


En cliquant sur « Paramètres avancés », il est possible de modifier les options pour les différentes cibles de recherche.



Dans l'aide en ligne, il est possible limiter les sujets dans laquelle la recherche sera effectuée en cochant « Rechercher uniquement dans les rubriques suivantes » et en sélectionnant les sujets concernés.

Il est possible d'ajouter une nouvelle source de recherche en cliquant sur le bouton « Nouveau ».



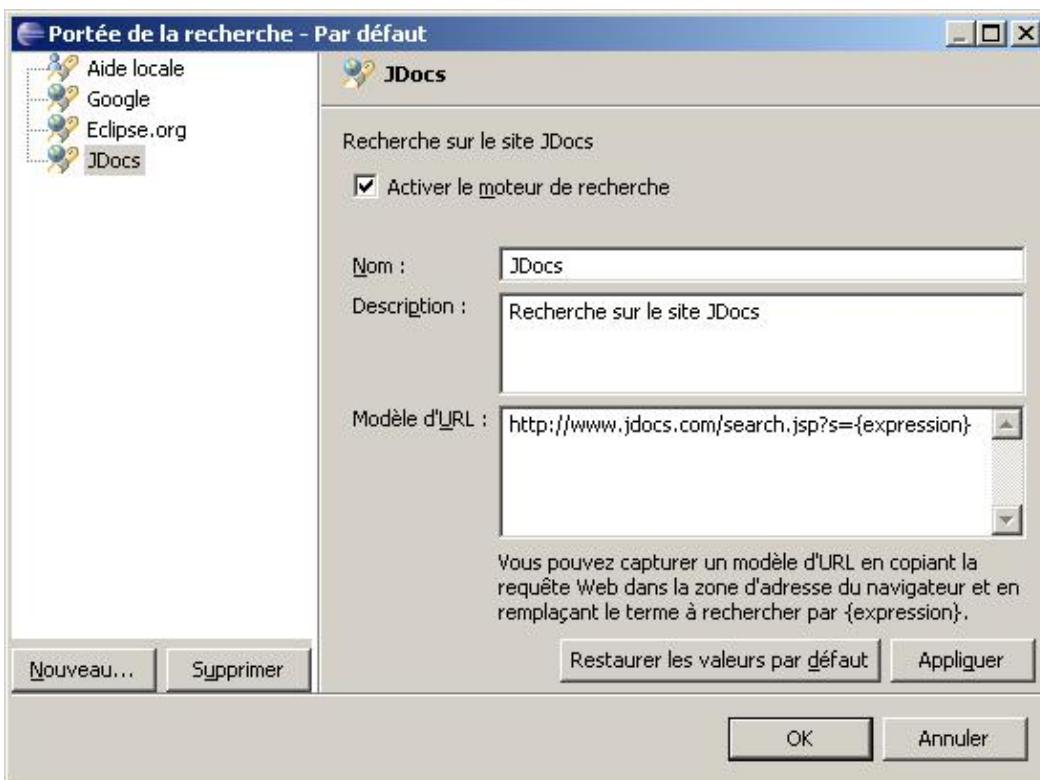
Pour ajouter une source sur le web, il suffit de sélectionner « Recherche Web » et de cliquer sur le bouton « Terminer »

La boîte de dialogue permet de saisir les informations utiles pour effectuer la recherche.



Il suffit alors de saisir le nom, la description et l'url à utiliser pour réaliser la recherche. Il faut utiliser l'expression {expression} pour désigner l'expression à rechercher lors de l'appel de l'url.

Exemple :



Une fois les informations saisies, il suffit de cliquer sur le bouton « OK ».



Pour lancer une recherche, il suffit de saisir l'expression recherchée et de cliquer sur le bouton « Aller »



Par défaut, la recherche se fait en local.



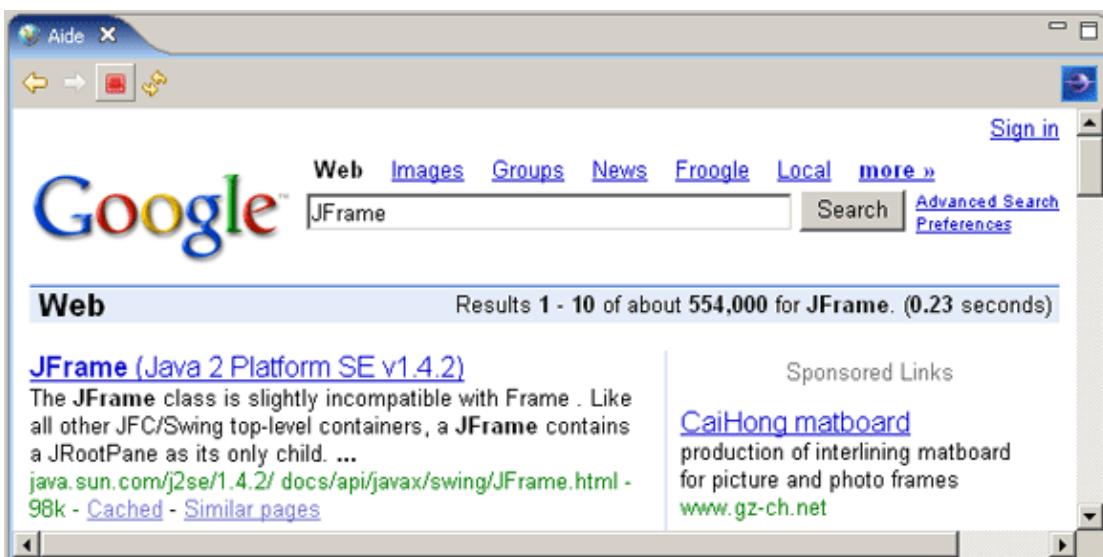
Pour lancer une recherche sur une source web, il faut ouvrir l'onglet de la source et cliquer sur Recherche Web

Un navigateur s'ouvre avec l'url de la source demandée

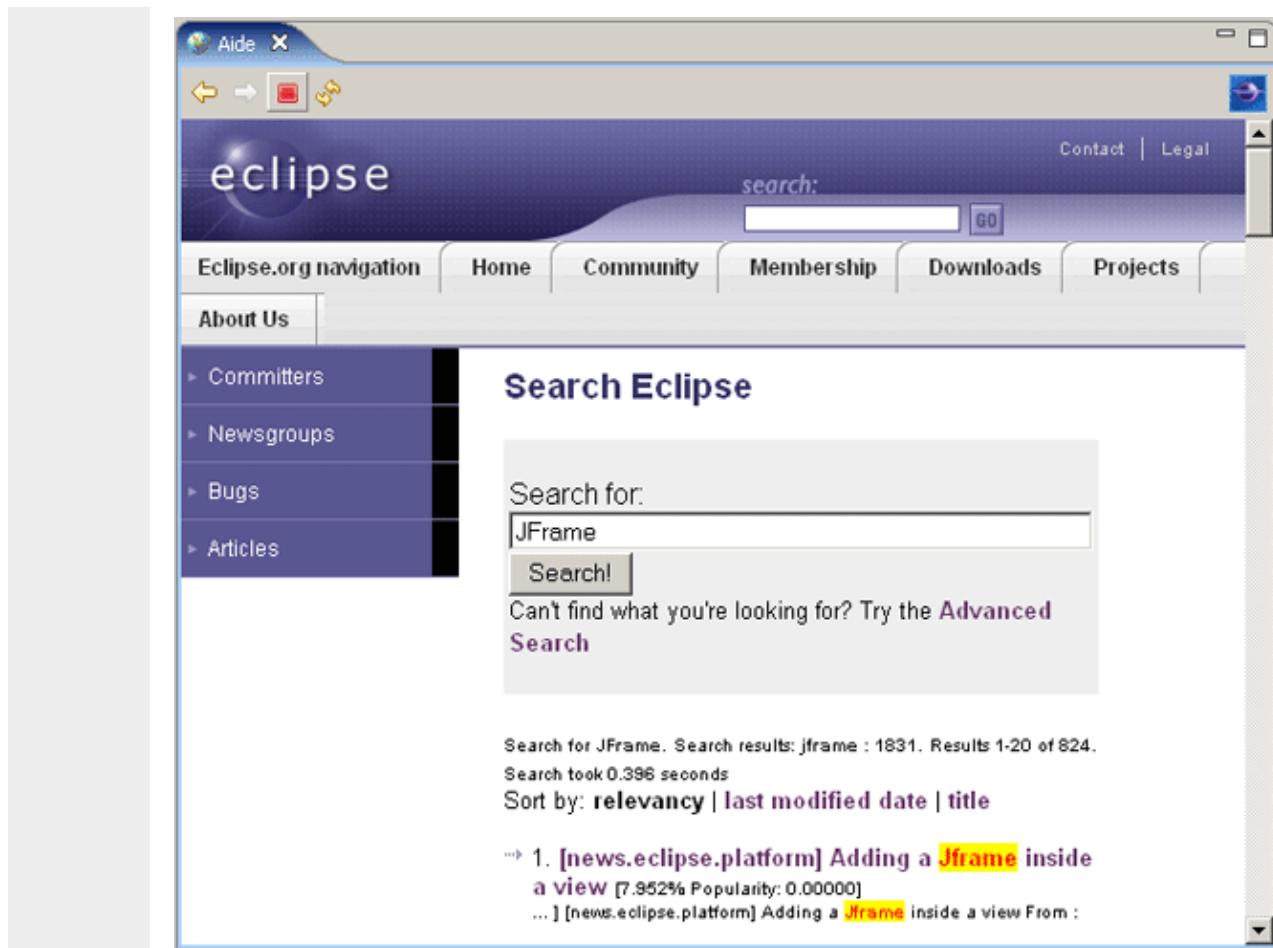


Le principe est le même pour demander une recherche sur Google directement dans Eclipse, il suffit de cliquer sur « Recherche web » dans l'onglet Google de la vue « Aide ».

Le navigateur intégré d'Eclipse est ouvert avec la recherche demandée sur Google

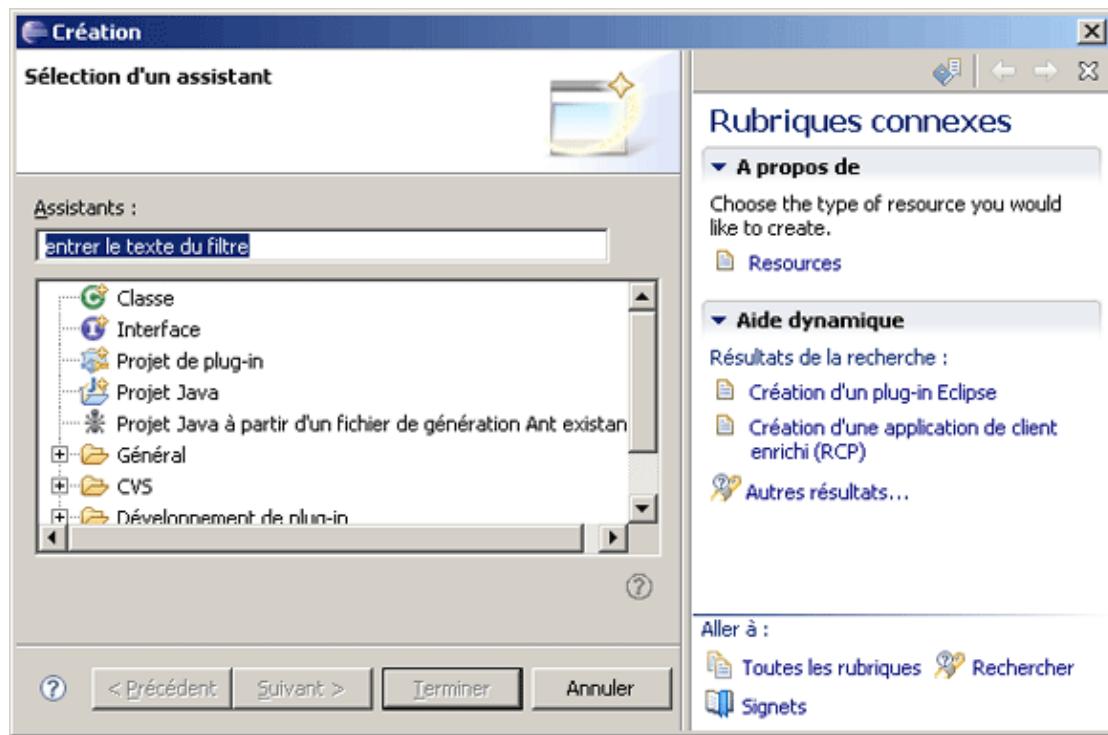


Il est aussi possible de demander la recherche sur le site Eclipse, il suffit de cliquer sur « Recherche web» dans l'onglet Eclipse.org de la vue « Aide »



Certaines boîtes de dialogue possèdent l'icône  en bas à gauche : elle permet un accès à l'aide contextuelle. L'appui sur la touche F1 est équivalent.

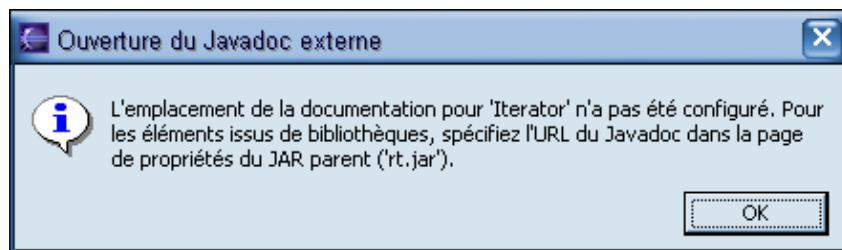
L'aide contextuelle apparaît dans la même fenêtre.



## 6.2. L'aide Javadoc

Si la configuration de l'IDE est correcte, il est possible d'accéder directement à la page Javadoc d'un élément java en plaçant le curseur sur cet élément et en appuyant sur F2.

Si l'accès à la documentation Javadoc pour l'élément n'est pas configuré un message d'erreur s'affiche.



Pour configurer l'IDE, il faut sélectionner la ressource qui contient l'élément dans la vue "Packages" :

- un projet
- un fichier .jar

L'exemple ci dessous permet d'associer la doc du JDK avec le fichier rt.jar

Il faut sélectionner le fichier rt.jar du projet dans la vue "Packages" et sélectionner le menu contextuel "Propriétés".

Il faut sélectionner "Emplacement Javadoc" puis cliquer sur parcourir pour sélectionner le répertoire qui contient le fichier package-list de la documentation.

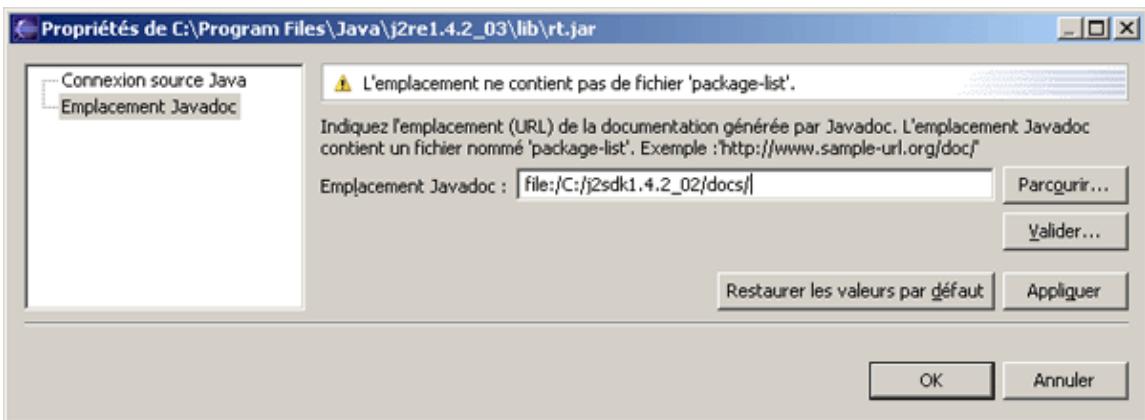


Le bouton "Valider" permet de vérifier si l'emplacement sélectionné contient les éléments nécessaires.



Enfin, il faut cliquer sur le bouton "OK" pour valider les changements.

Si le chemin n'est pas correct, un message d'erreur est affiché.

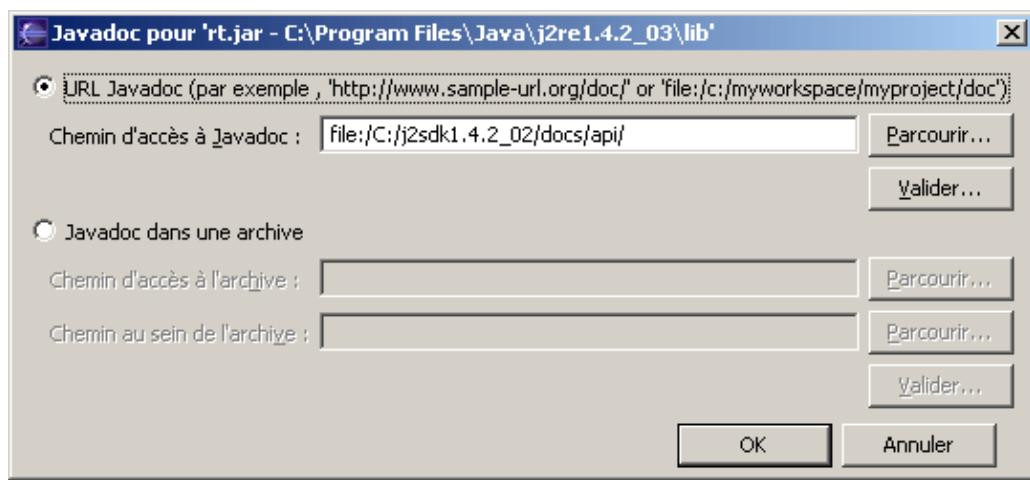


Il est alors possible dans l'éditeur de code Java de positionner le curseur sur un élément contenu dans un des emplacements Javadoc pour que l'aide en ligne affiche la page concernée par l'élément dans la documentation Javadoc.

Ce processus est applicable à toutes les API dont la documentation Javadoc est disponible.



Il n'est plus utile de décompresser le contenu d'une documentation Javadoc pour pouvoir l'associer à une bibliothèque.



### 6.3. Le plug-in Java docs de Crionics

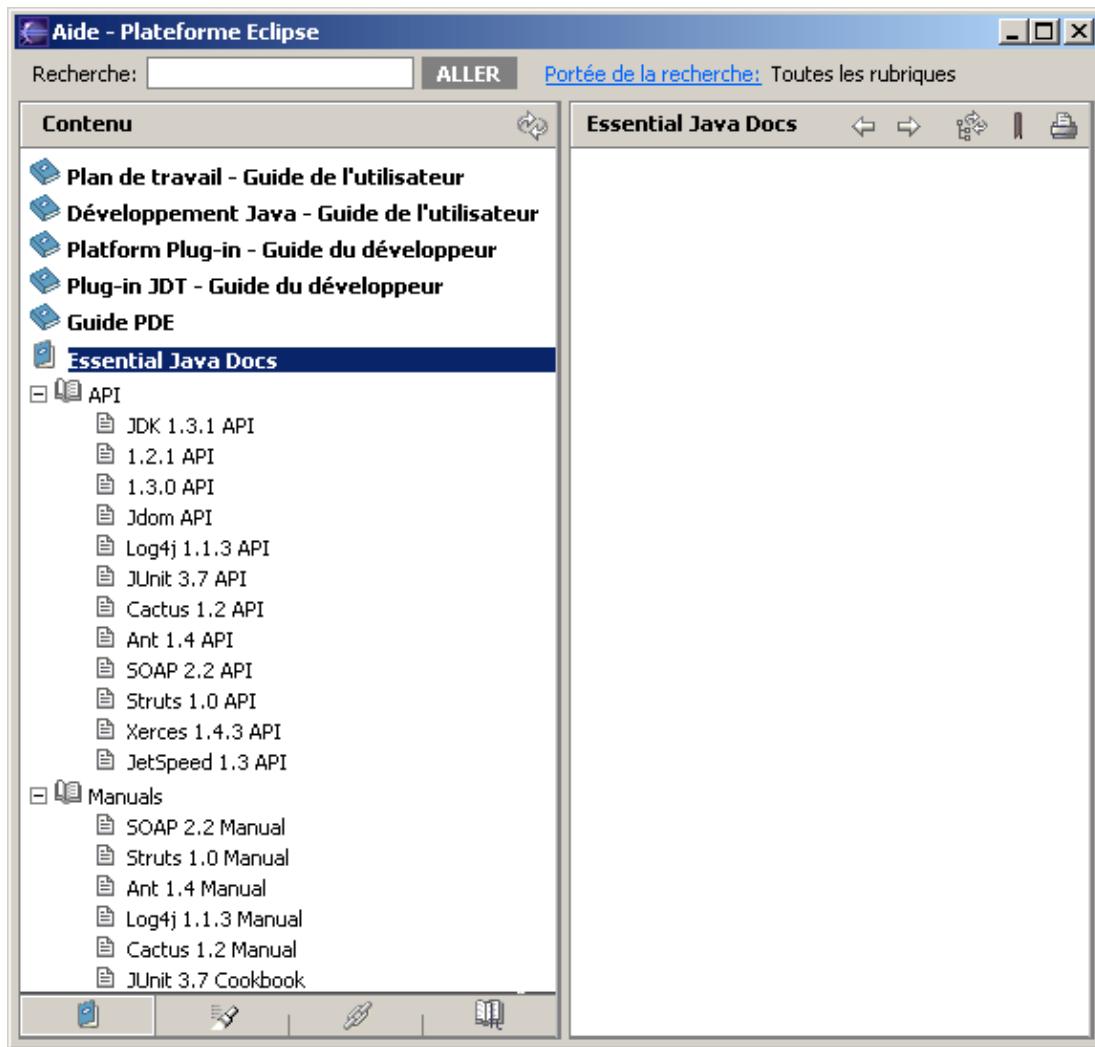
La société Crionics propose un plug-in qui s'intègre dans l'aide en ligne d'Eclipse et qui contient la documentation du JDK 1.3 et de quelques API open source.

Ce plug-in crée une entrée nommée "Essentials Java Docs" dans la table des matières de l'aide en ligne. Cette documentation regroupe la documentation du JDK 1.3.1, de quelques API open source fréquemment utilisée ainsi que quelques manuels pour ces API.

Ce plug-in est téléchargeable sur le site de Crionics (environ 30 Mo):

<http://www.cronicons.com/products/opensource/eclipse/com.cronicons.java.doc.zip>

Pour l'installer, il suffit de décompresser son contenu dans le répertoire plugins d'intallation d'Eclipse.



## **Partie 2 : le développement en Java**

# **Partie 2 :**

# **le**

# **développement**

# **avec Java**

Cette seconde partie est chargée de présenter les bases de l'utilisation d'Eclipse pour le développement avec Java.

Elle comporte les chapitres suivants :

- ◆ Le Java Development Tooling (JDT) : Détaille le JDT qui fournit des outils pour permettre le développement avec Java
- ◆ Déboguer du code Java : détaille la perspective Débogage dédiée à la recherche et à la correction des bugs.
- ◆ Le refactoring : détaille les puissantes fonctionnalités de refactoring proposées par Eclipse
- ◆ Ant et Eclipse : présente l'utilisation de l'outil Ant avec Eclipse
- ◆ JUnit et Eclipse : présente l'utilisation de JUnit avec Eclipse pour réaliser l'automatisation des tests unitaires.

## 7. Le Java Development Tooling (JDT)

# Chapitre 7

Le Java Development Tooling (JDT) est inclus dans Eclipse pour fournir des outils de développement en Java. Il inclut plusieurs plug-ins et apporte :

- les perspectives "Java" et "Navigation Java"
- les vues "Packages" et "Hierarchie"
- les éditeurs "Java" et "Scrapbook"
- les assistants : pour créer de nouveaux projets, packages, classes, interfaces, ...

Dans l'espace de travail, il définit un projet de type particulier pour les projets Java. L'arborescence de ces projets contient un fichier particulier nommé .classpath qui contient la localisation des bibliothèques utiles à la compilation et à l'exécution du code.

### 7.1. Les projets de type Java

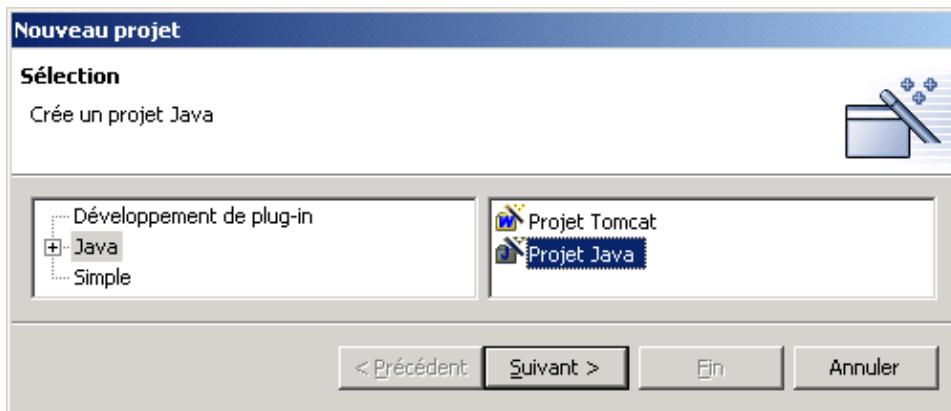
Pour pouvoir développer des entités en Java, il faut les regrouper dans un projet de type Java.

#### 7.1.1. La création d'un nouveau projet Java

Dans la perspective "Java", il y a plusieurs possibilités pour lancer l'assistant de création d'un nouveau projet :

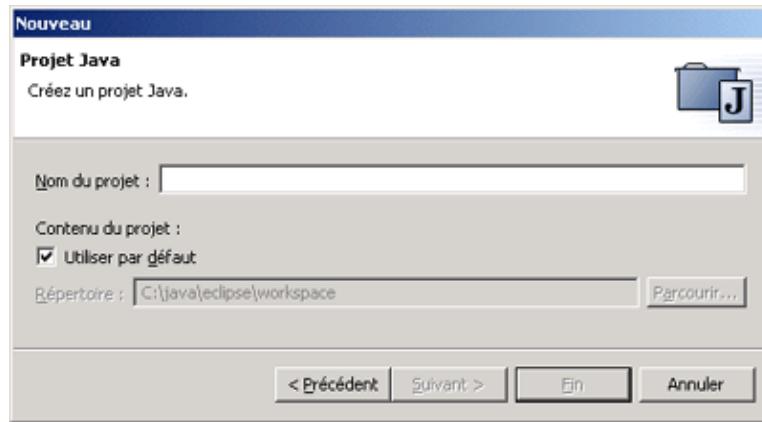
- sélectionner l'option "Projet" du menu "Fichier/Nouveau"
- sélectionner l'option "Nouveau/Projet" du menu contextuel de la vue "Packages"

L'assistant demande le type de projet à créer.

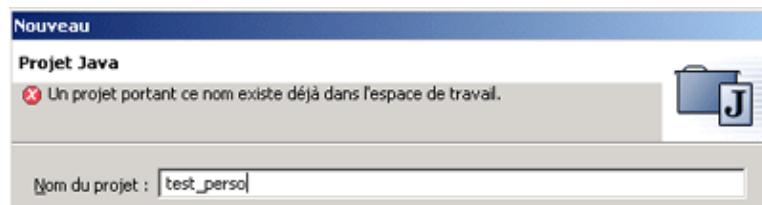


Pour demander directement la création d'un projet "Java", il suffit de cliquer sur l'icône  de la barre d'outils.

L'assistant demande le nom du projet.

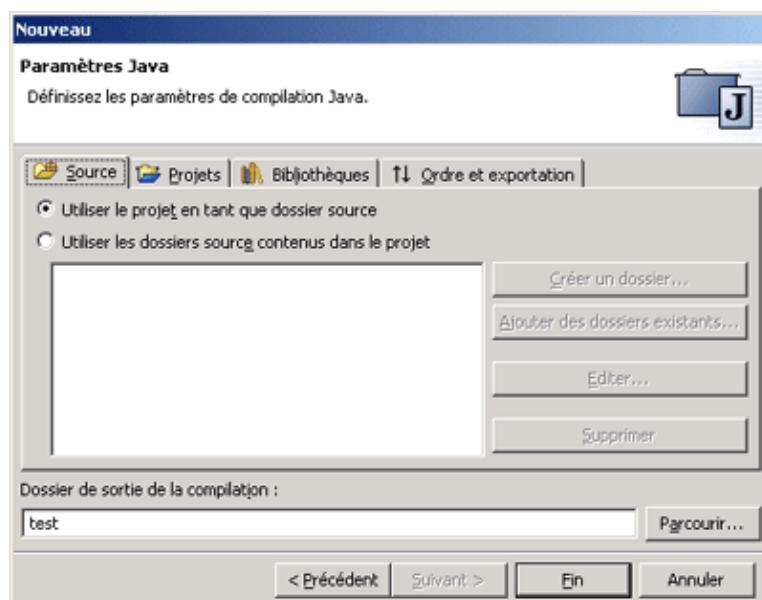


Ce nom de projet ne doit pas déjà être utilisé dans l'espace de travail courant sinon un message d'erreur est affiché.



En cliquant sur le bouton "Fin", le projet est créé avec des paramètres par défaut.

Pour modifier certains paramètres avant la création du projet, suffit de cliquer sur le bouton "Suivant" :



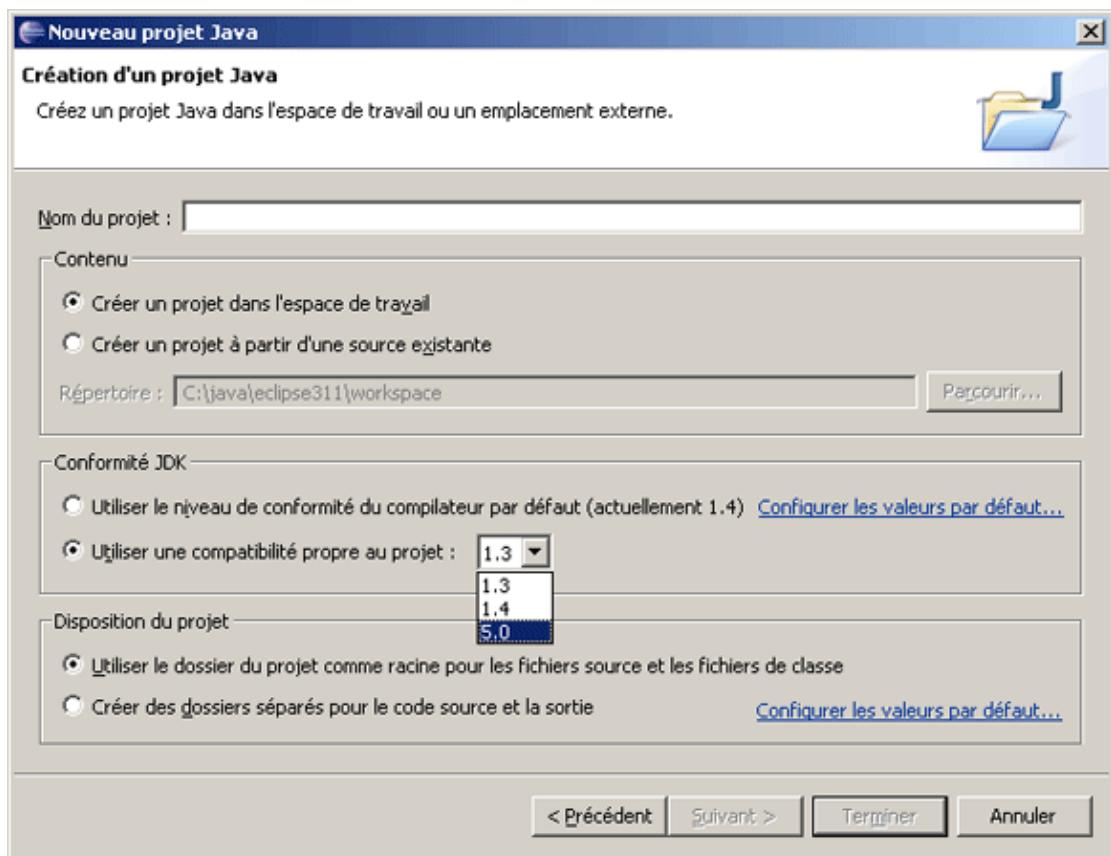
La modification de ces paramètres sera détaillée dans la section suivante. Une fois les paramètres modifiés, cliquer sur le bouton "Fin". Le projet apparaît dans la vue "Packages" de la perspective.



La version 3.1 d'Eclipse propose quelques fonctionnalités intéressantes lors de la création d'un projet Java.



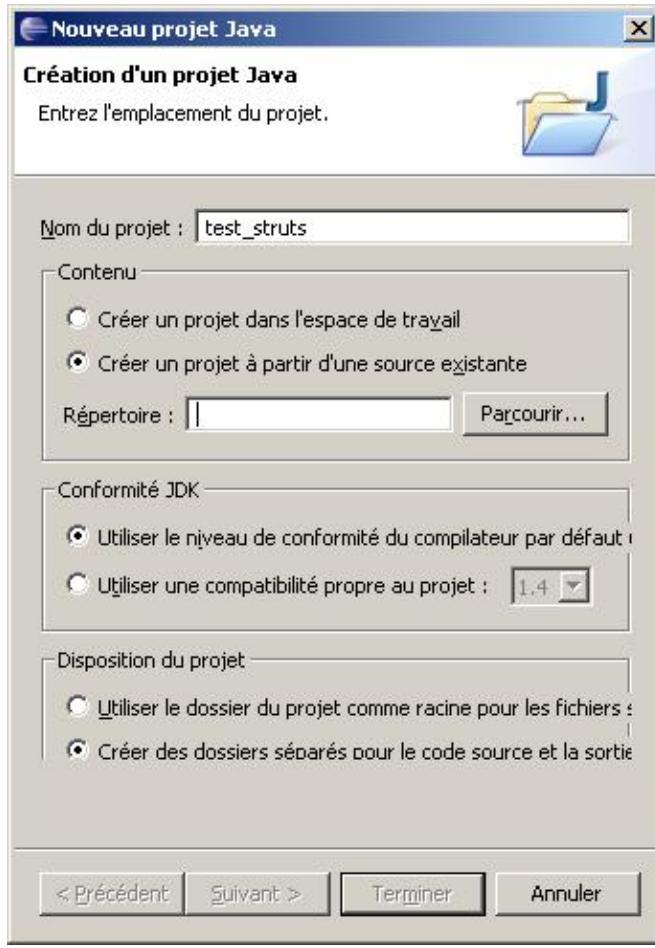
Sélectionnez "Projet Java" et cliquez sur le bouton "Suivant".



La première page de l'assistant propose des options pour faciliter la création du nouveau projet :

- le contenu du projet : précise si le projet est un nouveau projet ou si le projet est importé
- la conformité JDK : précise le JDK utilisé par le JDK
- disposition du projet : précise si les sources et les binaires compilés sont mélangés à la racine ou stockés dans des répertoires dédiés.

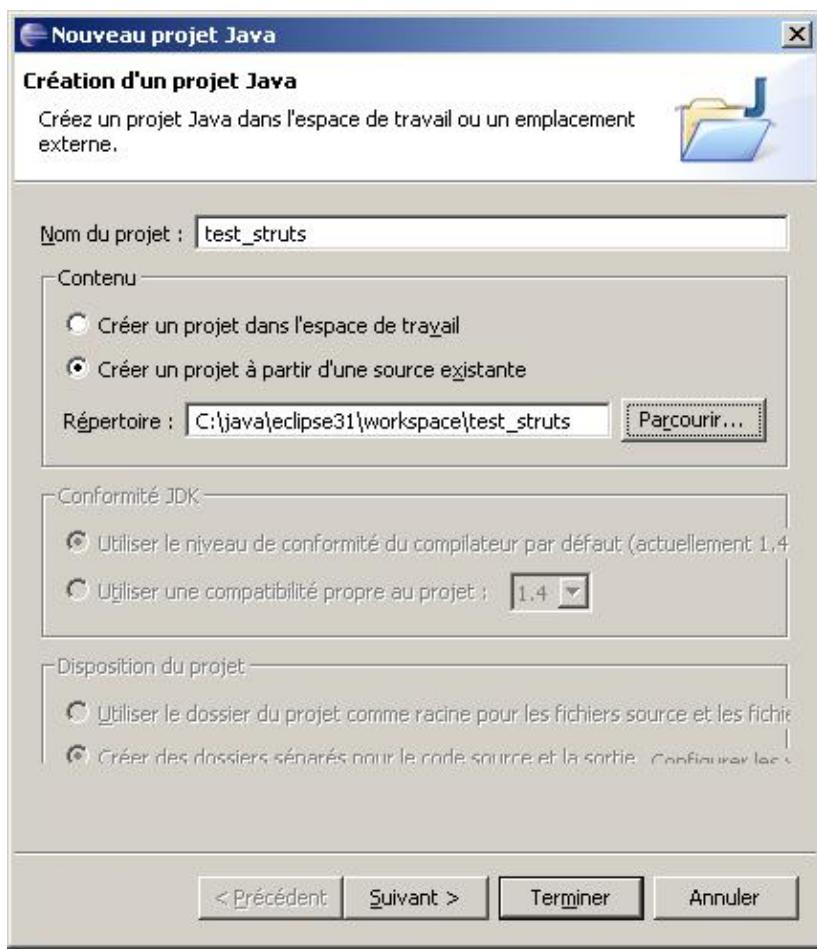
A partir de cette page, il est possible de demander la création d'un projet à partir de source existantes : dans ce cas, le projet n'est pas créé physiquement dans le workspace.



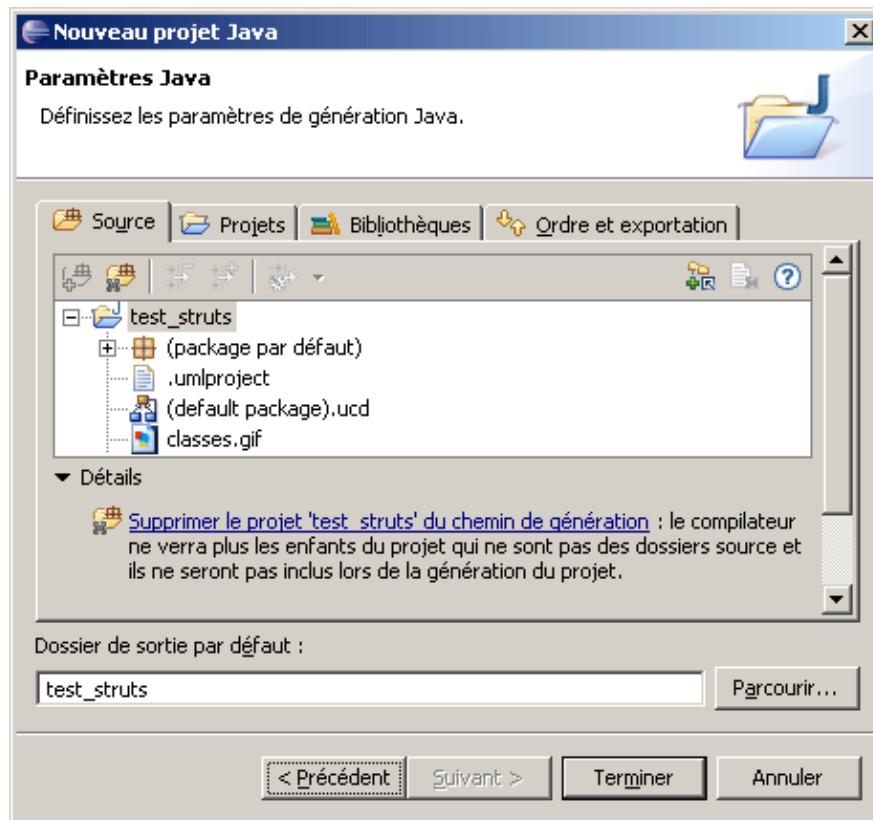
Pour cela, il faut cliquer sur « Créer un projet à partir d'une source existante » et cliquer sur le bouton « Parcourir ».



Sélectionnez le répertoire et cliquez sur le bouton « OK »

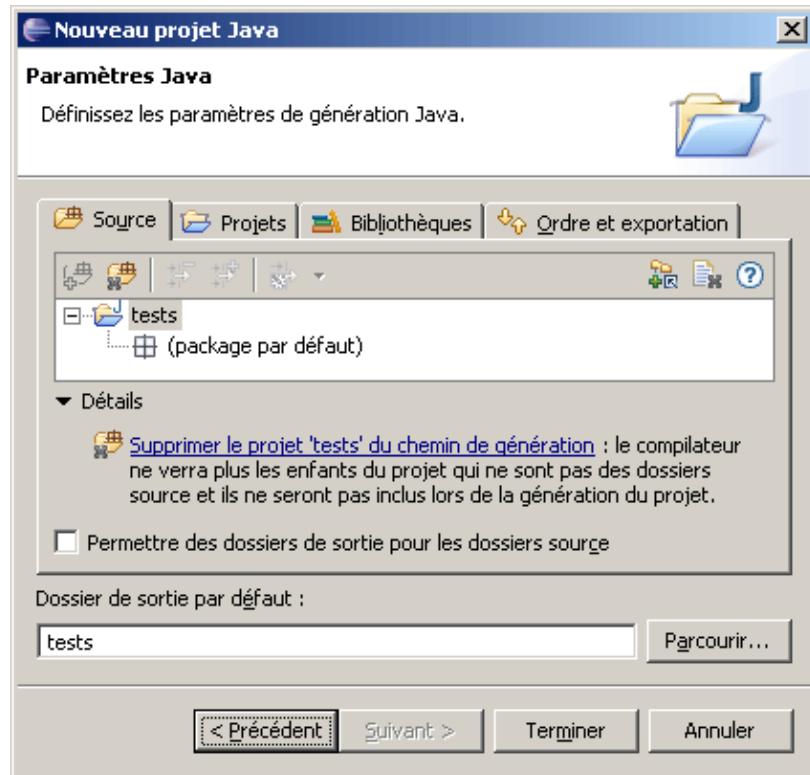


Cliquez sur le bouton « Suivant » : le projet est ajouté dans le workspace et la page suivante de l'assistant est affichée.

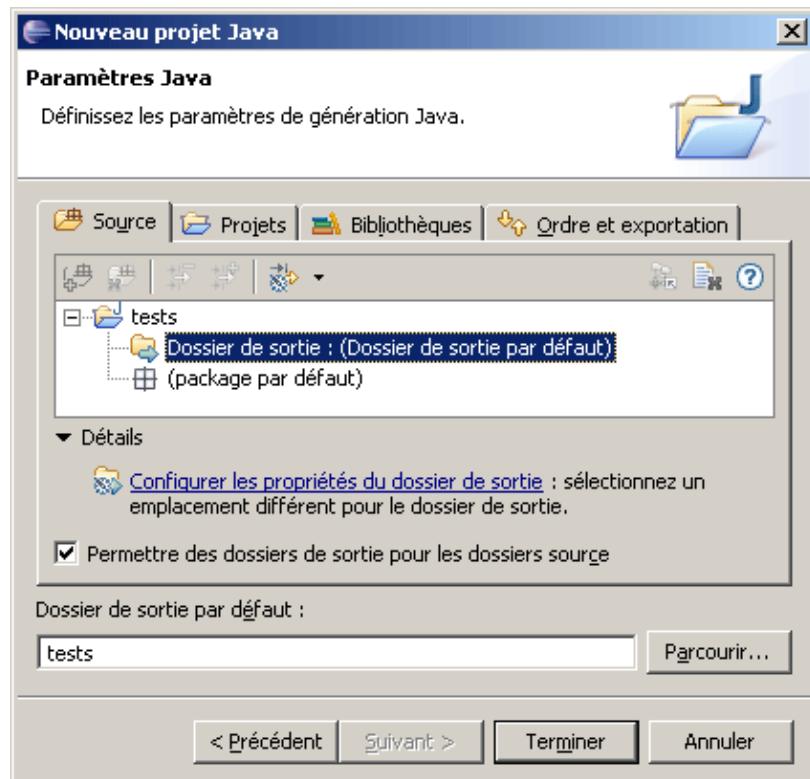


Cliquez sur le bouton « Terminer ».

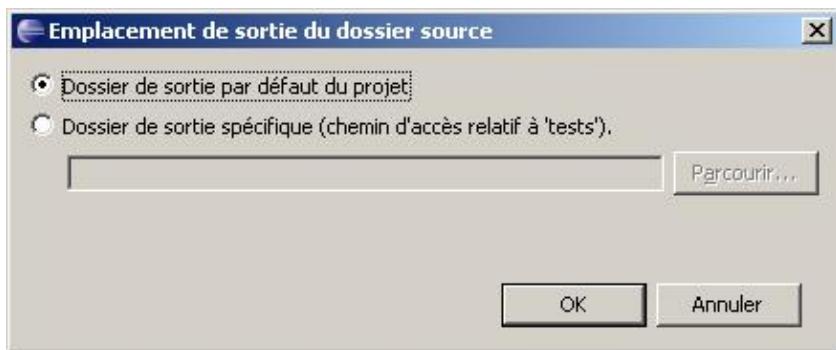
Il est possible de modifier la disposition du projet après sa création en utilisant les propriétés du projet.



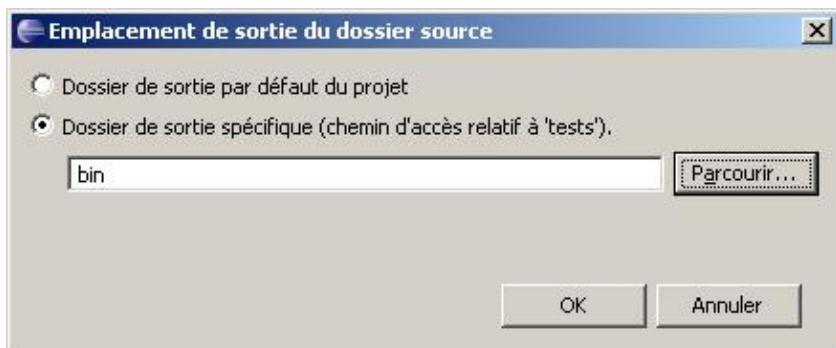
Cocher « Permettre des dossiers de sortie pour les dossiers source »



Sélectionnez dans l'arborescence « Dossier de sortie » et cliquer sur le lien « Configurer les options du dossier de sortie »



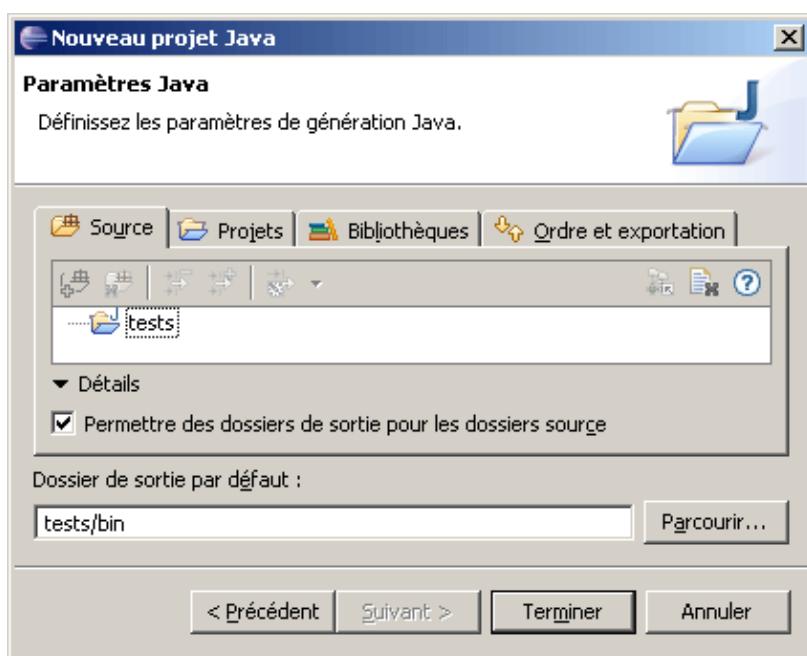
Pour préciser un répertoire particulier, sélectionnez « Dossier de sortie spécifique »



Saisissez ou sélectionnez le nom du répertoire et cliquez sur le bouton « OK »



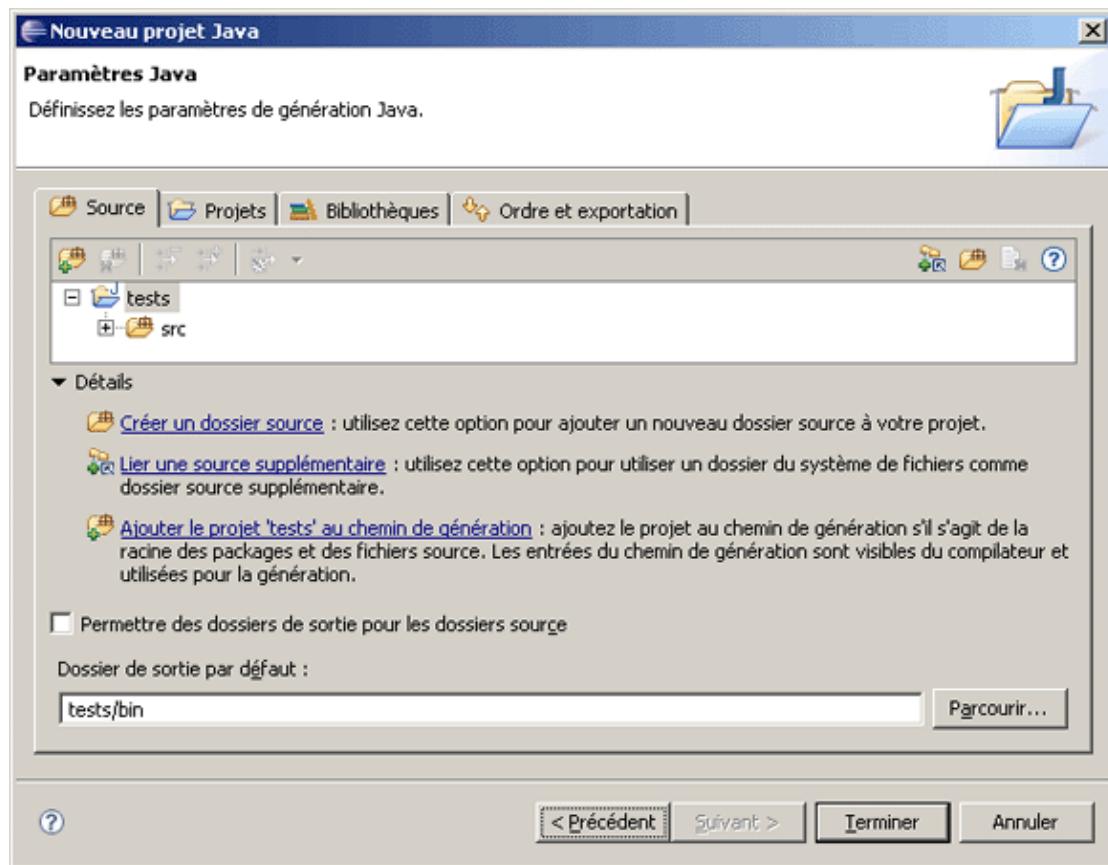
Cliquez sur le bouton « OK »



Cliquez sur le bouton « Terminer ».



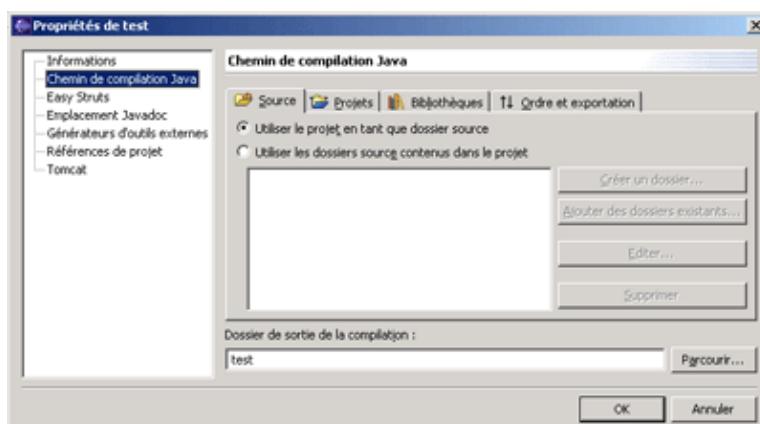
L'onglet "Source" de la page "Paramètres Java" s'est enrichi de deux options : "Lier une source supplémentaire" et "Ajouter le projet au chemin de génération".



### 7.1.2. Les paramètres d'un projet Java

Les principaux paramètres d'un projet peuvent être modifiés :

- lors de l'utilisation de l'assistant à la création du projet
- en sélectionnant le menu contextuel "Propriétés" sur le projet sélectionné dans la vue "Packages" et de choisir "Chemin de compilation Java"



Les propriétés "Chemin de compilation Java" sont regroupées dans quatres onglets :

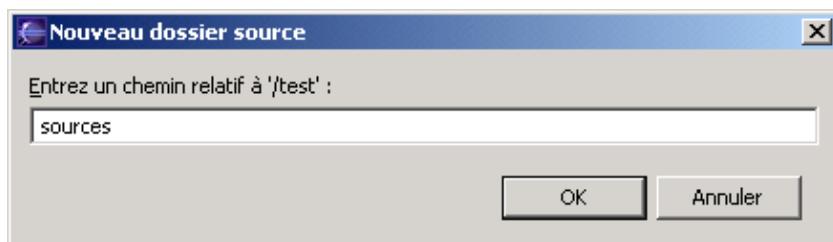
Onglet	Rôle
Source	

	Permet de préciser le répertoire qui va contenir les sources et celui qui va contenir le résultat des compilations
Projets	Permet d'utiliser d'autre projet avec le projet courant
Bibliothèques	Permet d'ajouter des bibliothèques au projet
Ordre et exportation	Permet de préciser l'ordre des ressources dans la classpath

L'onget "Source" permet de préciser le répertoire qui va contenir les sources : par défaut, c'est le répertoire du projet lui même (l'option "utiliser le dossier projet en tant que dossier source" est sélectionné).

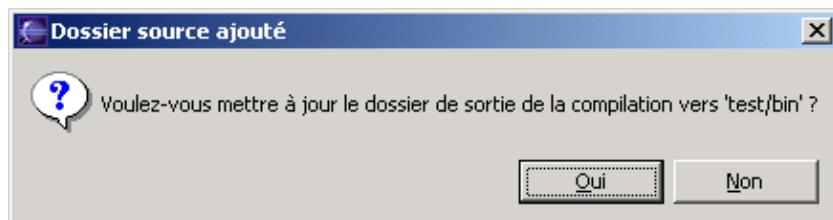
Pour stocker les ressources dans un répertoire dédié, il faut sélectionner l'option "Utiliser les dossiers sources contenus dans le projet". La liste permet de sélectionner le ou les répertoires.

Le bouton "Créer un dossier" ouvre une boîte de dialogue qui demande le nom du répertoire.



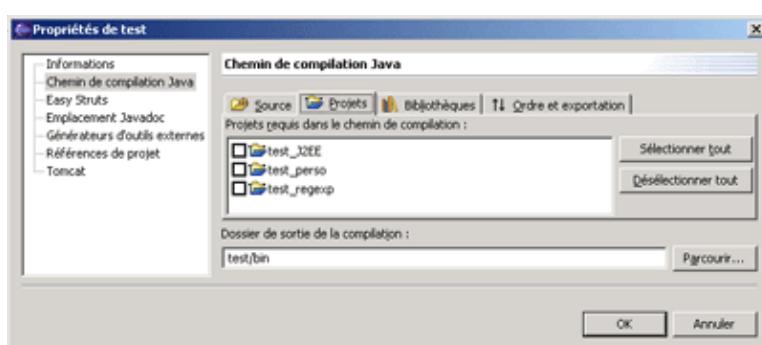
Il suffit de saisir le nom, par exemple "sources" et cliquer sur le bouton "OK"

Par défaut, dès qu'un premier répertoire contenant les sources est sélectionné, Eclipse propose de créer un répertoire bin qui va contenir le résultat des différentes compilations.



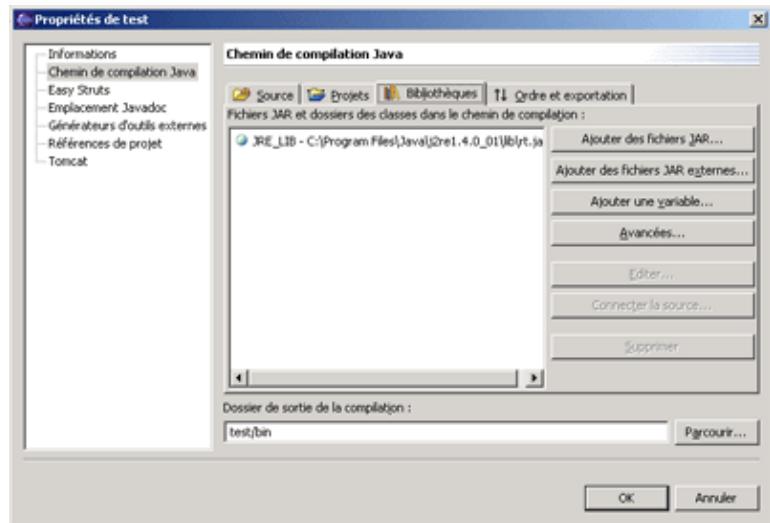
La réponse à la question est libre mais il est préférable de répondre "Oui".

L'onget "Projets" permet d'ajouter des projets contenus dans l'espace de travail au classpath.



Il suffit de cocher les projets à inclure dans le classpath.

L'onget "Bibliothèques" permet d'ajouter des bibliothèques externes au projet notamment des fichiers .jar.



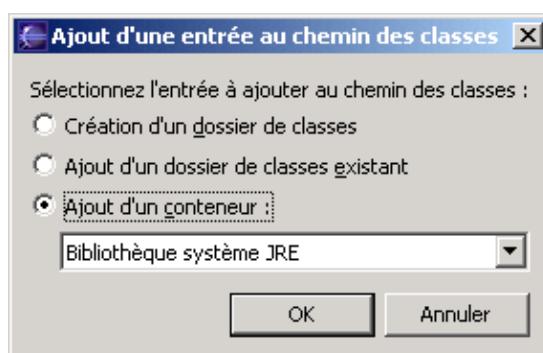
Les bibliothèques incluses dans le classpath du projet courant sont affichées dans la liste.

Pour ajouter une nouvelle bibliothèque contenue dans l'espace de travail, il suffit de cliquer sur "Ajouter des fichiers jar". Pour ajouter des fichiers jar qui ne sont pas contenus dans l'espace de travail, il suffit de cliquer sur le bouton "Ajouter des fichiers jar externes".



Une boîte de dialogue permet de sélectionner le fichier jar. En cliquant sur le bouton "Ouvrir", le fichier jar est ajouté dans la liste.

Le bouton "Avancées ..." permet d'ajouter d'autres entités au classpath notamment des répertoires qui contiennent des fichiers compilés.



Le bouton "Editer" permet de modifier les caractéristiques de la bibliothèque (son chemin d'accès dans le cas d'un fichier jar).

Le bouton "Supprimer" permet de supprimer une bibliothèque du classpath.

L'onglet "Ordre et exportation" permet de modifier l'ordre des bibliothèques dans le classpath, de préciser la cible des éléments générés (le répertoire qui va les contenir) et de définir les ressources qui seront utilisables par les autres projets de l'espace de travail lorsque le projet sera lié avec eux.

## 7.2. La création d'entité

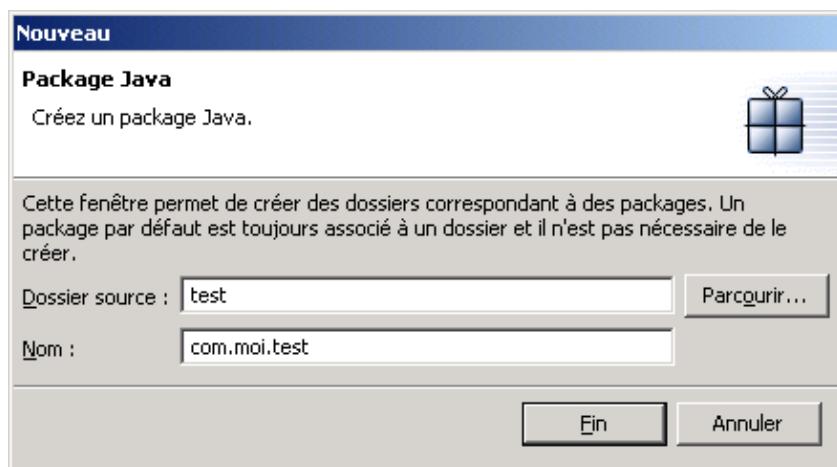
Dans un projet Java, il est possible de créer différentes entités qui entrent dans sa composition : les packages, les classes et les interfaces.

### 7.2.1. La création de packages

Il est possible de créer les packages à l'avance même si ceux ci peuvent être créés automatiquement en même temps qu'une classe qui la contient. Pour créer un nouveau package, il y a plusieurs possibilités :

- cliquer sur la flèche de l'icône  de la barre d'outils de la perspective "Java" et sélectionner "Package"
- cliquer sur l'icône  de la barre d'outils de la perspective "Java"
- sélectionner l'option "Package" du menu "Fichier / Nouveau"

L'assistant demande le nom du package.

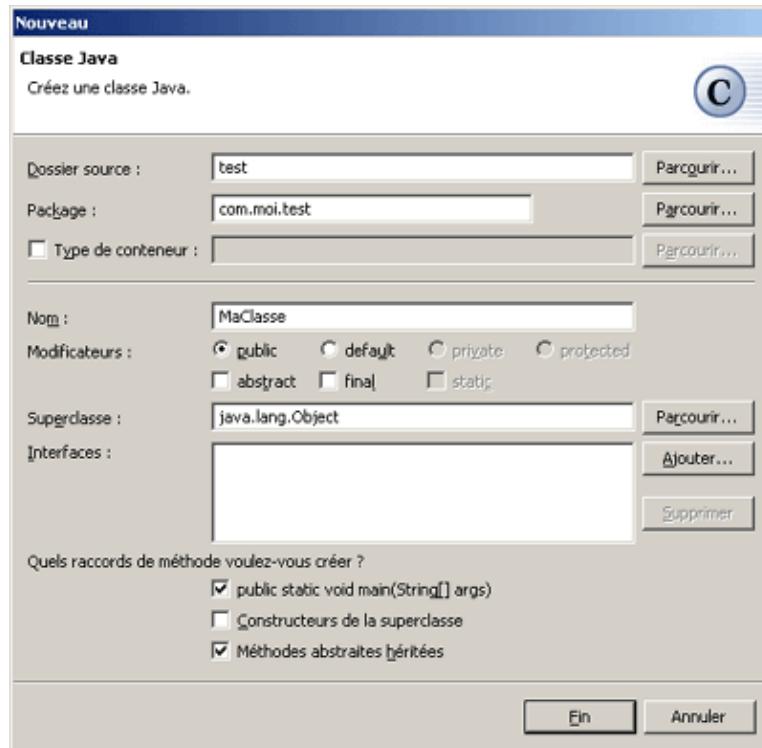


Cliquez sur le bouton "Fin", pour créer le nouveau package. Le package apparaît dans la vue "Packages".

### 7.2.2. La création de classes

Un assistant facilite la création de la classe. Cette création d'une classe peut se faire :

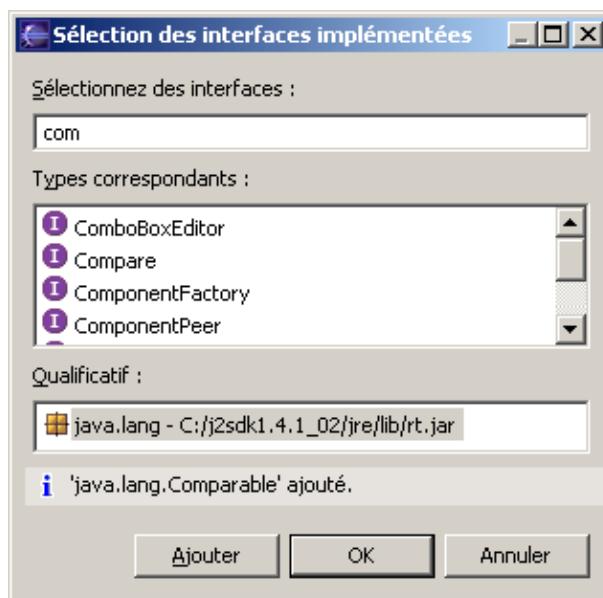
- soit en cliquant sur l'icône  dans la barre d'outils
- soit en sélectionnant l'option Classe du menu "Fichier/Nouveau"



L'assistant demande de renseigner les différentes caractéristiques de la nouvelle classe : le projet et le package d'appartenance, le nom, les modificateurs, la classe mère, les interfaces implémentées. Enfin, il est possible de demander à l'assistant de générer certaines méthodes.

Si un projet ou un package est sélectionné dans la vue "Packages", celui ci est automatiquement repris par l'assistant.

L'ajout d'une interface implémentée se fait en la sélectionnant dans une liste.



Pour ajouter une interface, il suffit de double cliquer dessus ou de la sélectionner et d'appuyer sur le bouton "Ajouter". Une fois toutes les interfaces ajoutées, il suffit de cliquer sur le bouton "OK".

Toutes les méthodes définies dans la ou les interfaces sélectionnées seront présentes dans le code source de la classe générée. Si le package saisi n'existe pas dans le projet, celui ci sera créé en même temps que la classe.

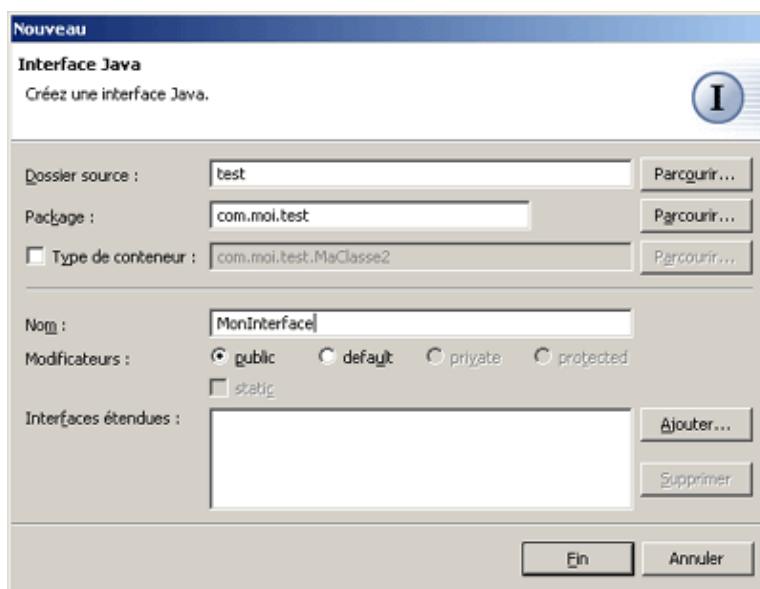
Une fois toutes les données utiles renseignées, il suffit de cliquer sur le bouton "Fin" pour que la classe soit générée et que l'éditeur s'ouvre avec le contenu de son code source.

### 7.2.3. La création d'interfaces

La création d'une interface peut se faire :

- en cliquant sur l'icône  dans la barre d'outils
- en sélectionnant l'option Interface du menu " Fichier/Nouveau "

Un assistant facilite la création de l'interface.



L'assistant demande de renseigner les différentes caractéristiques de la nouvelle interface : le projet et le package d'appartenance, le nom, les modificateurs ainsi que les éventuelles interfaces héritées.

Une fois toutes les données utiles renseignées, il suffit de cliquer sur le bouton " Fin " pour que l'interface soit générée et que l'éditeur s'ouvre avec le contenu de son code source.

### 7.2.4. La création de classe et de package par copier/coller du code source dans l'explorateur de package



Cette fonctionnalité permet de créer une classe et éventuellement le package associé en copiant le code source de la classe à partir du contenu du presse papier (par exemple remplie par la copie du code à partir d'un éditeur externe) et en collant le contenu sur un projet de l'explorateur de package (Ctrl+V ou option « coller » du menu contextuel).

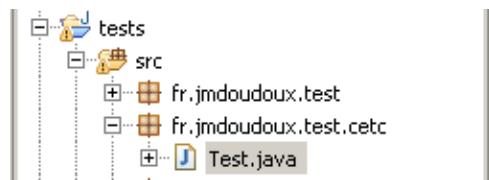
Exemple :

```
package fr.jmdoudoux.test.cetc;

public class Test {

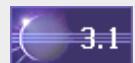
    public static void main(String[] args) {
        System.out.println("test");
    }
}
```

La classe et le package sont automatiquement créés.



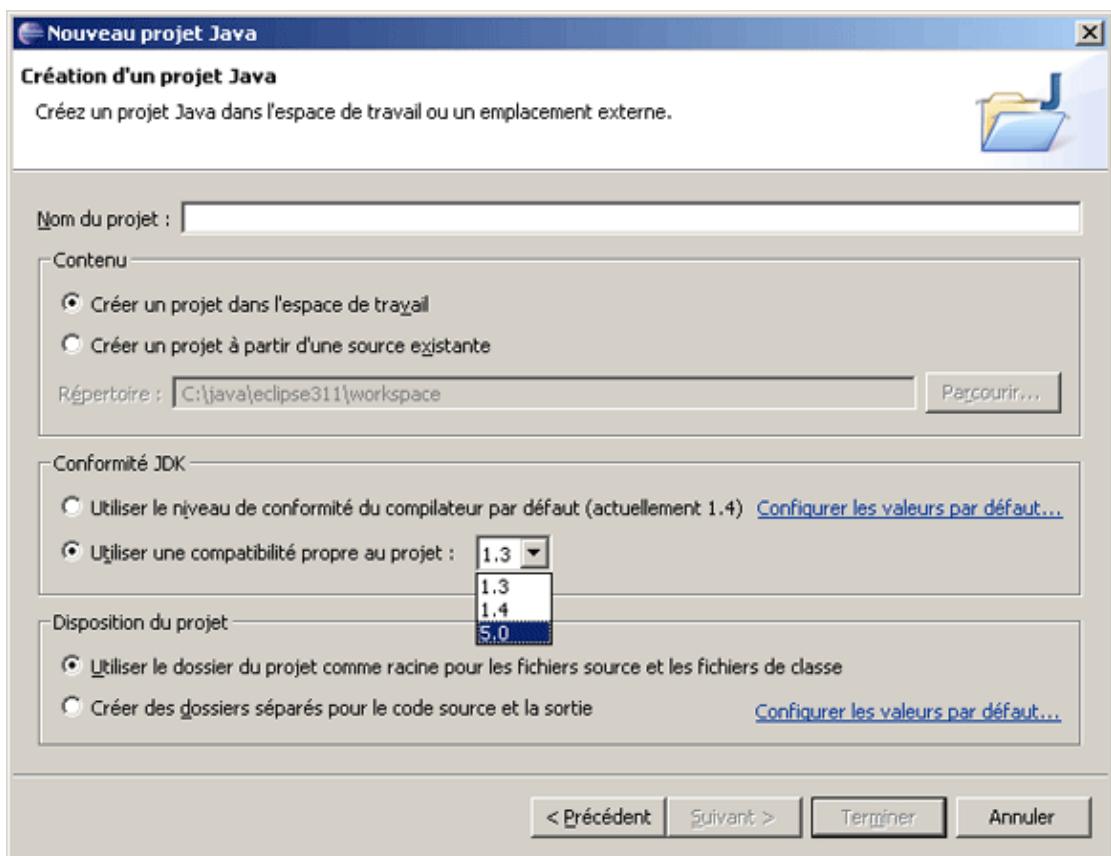
Attention : pour que la création du package puisse être réalisé, il faut absolument que le coller se fasse sur un projet.

### 7.3. Le support de Java 5.0



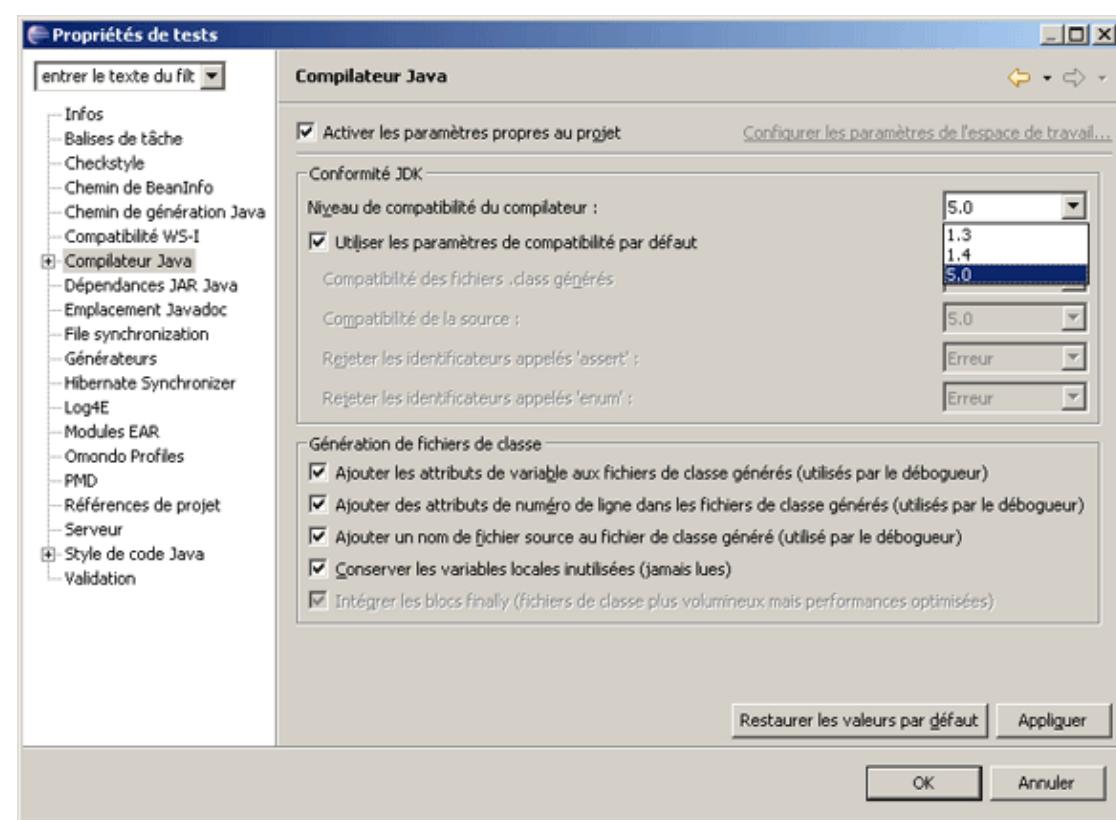
Eclipse 3.1 propose un support complet de Java 5.0.

La version de Java à utiliser dans un projet peut être précisée lors de la création d'un projet :



Par défaut, c'est la version 1.4 qui sera utilisée par défaut. Pour préciser une version différente, il suffit de cocher « Utiliser une compatibilité propre au projet » et de sélectionner la version à utiliser dans la liste déroulante.

Il est possible de modifier la version du JDK à utiliser dans les préférences du projet.



Il suffit de sélectionner « Compilateur Java » dans l'arborescence et de sélectionner la version du JDK à utiliser.

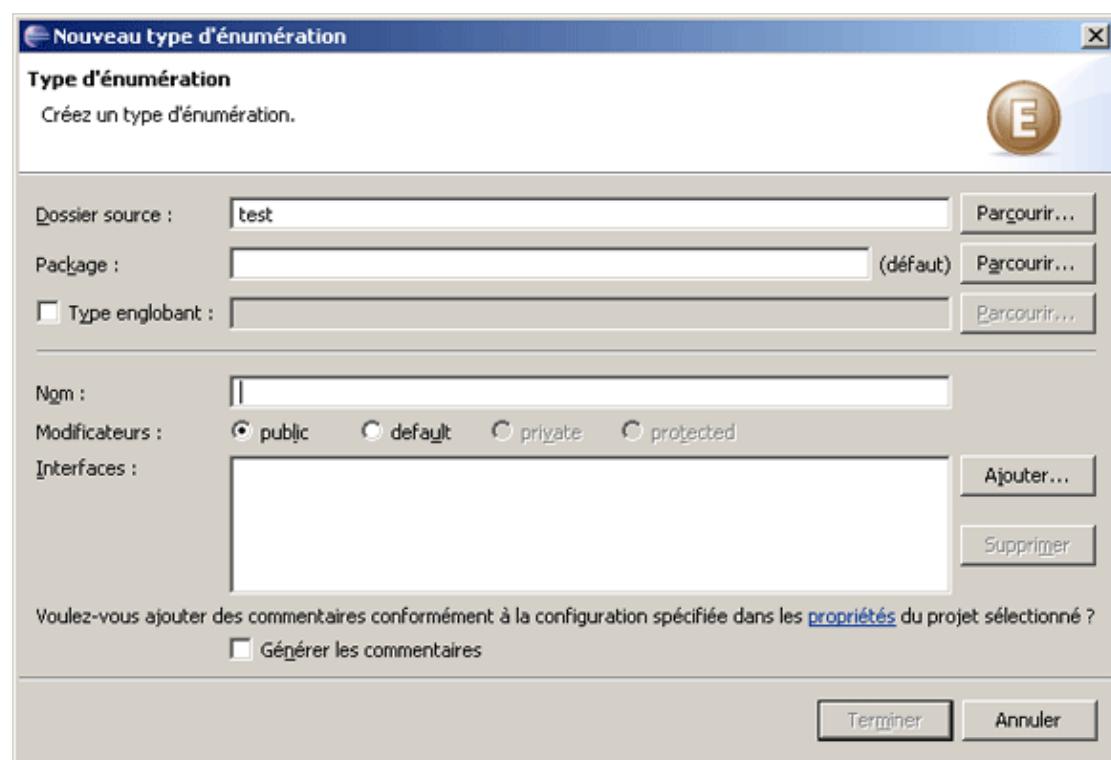


Cliquez sur « Oui » pour recompiler l'intégralité du projet avec la nouveau JDK.

### 7.3.1. La création d'une énumération



Il est possible de créer une énumération en demandant la création d'une entité de type « Java/Enumération ».



Il faut saisir le nom du package et le nom de la classe.

Si le projet n'est pas configuré pour utiliser le JDK 5.0, un message d'erreur est affiché



En cliquant le bouton « Terminer », la classe est générée :

Exemple :

```
package com.jmd.test;

public enum MonStyle {
```

Il suffit alors de compléter la classe avec les différentes valeurs qui composent l'énumération

Exemple :

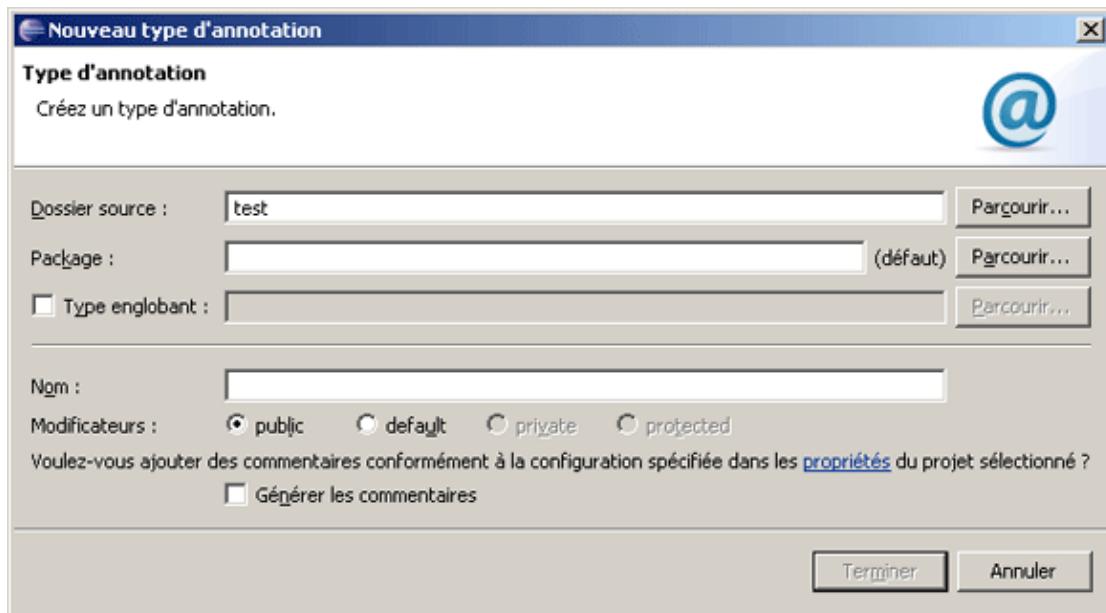
```
package com.jmd.test;

public enum MonStyle {
    STYLE_1, STYLE2, STYLE_3
}
```

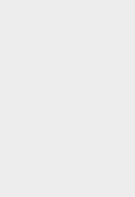
### 7.3.2. La création d'une annotation



Il est possible de créer une annotation en demandant la création d'une entité de type « Java/Annotation ».



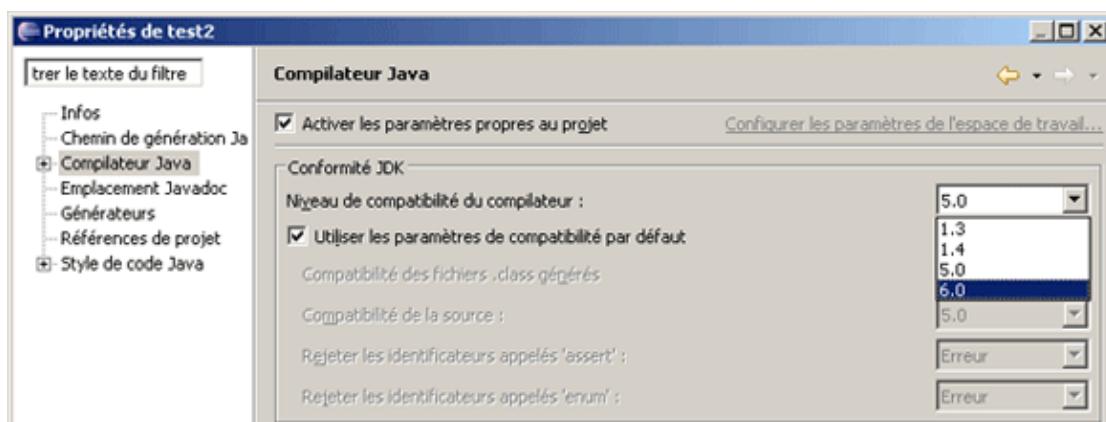
Si le projet n'est pas configuré pour utiliser le JDK 5.0, un message d'erreur est affiché.



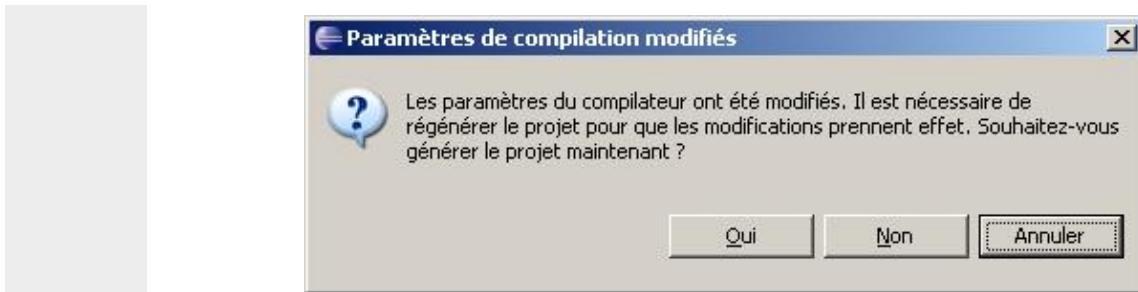
### 7.4. Le support de Java 6.0 par le compilateur



Dans les propriétés du projet, il est possible de préciser dans « Compilateur Java » le support de Java 6.0. Pour modifier la version de Java, cochez la case « Activer les paramètres propres au projet »



Sélectionnez la version 6.0 dans la liste déroulante.



Cliquez sur le bouton « Oui » pour recompiler tous les fichiers sources avec une cible ( target ) 1.6.

Remarque : pour que cela fonctionne correctement, il est nécessaire qu'un JDK 6.0 soit installé sur la machine et configuré dans Eclipse.

## 7.5. Les vues du JDT

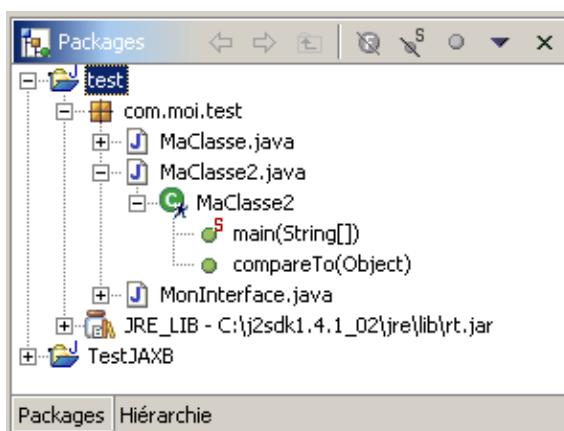
Le JDT contient les vues "Packages" et "Hiérarchie"

### 7.5.1. La vue "Packages"

Cette vue permet d'afficher de façon arborescente le contenu des différents packages définis et utilisés dans chaque projet ainsi que les bibliothèques utilisées par le projet.

Pour les éléments contenant du code source, l'arborescence sous jacente permet de voir les différents membres qui composent l'élément.

Un double clic sur des éléments de l'arborescence, permet d'ouvrir l'éditeur directement sur l'élément sélectionné.



Chaque élément de l'arborescence possède une petite icône en fonction en son type :

Icône	Type de l'élément
	Projet de type Java
	Package
	Elément Java : classe ou interface
	Interface Java

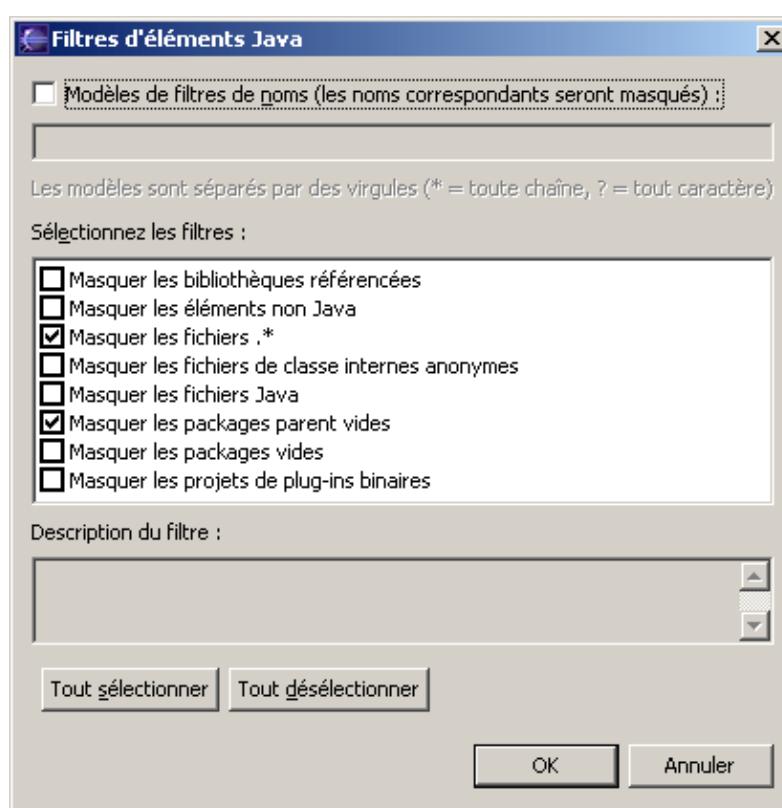
	Classe public Java
	Classe Java pouvant être exécutée (possédant une méthode main())
	Classe protected
	Classe package friendly
	Classe private
	Champ public
	Champ private
	Champ protected
	Champ package friendly
	Méthode public
	Méthode private
	Méthode protected
	Méthode package friendly

Le bouton permet de masquer les champs définis dans les éléments Java.

Le bouton permet de masquer les membres statiques.

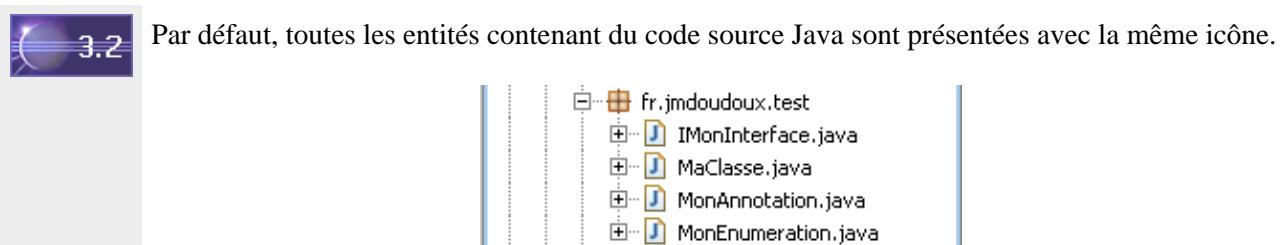
Le bouton permet de masquer tous les membres qui ne sont pas publics.

Il est possible de restreindre les entités affichées dans la vue package. Il suffit de cliquer sur bouton et de sélectionner l'option " Filtres ".

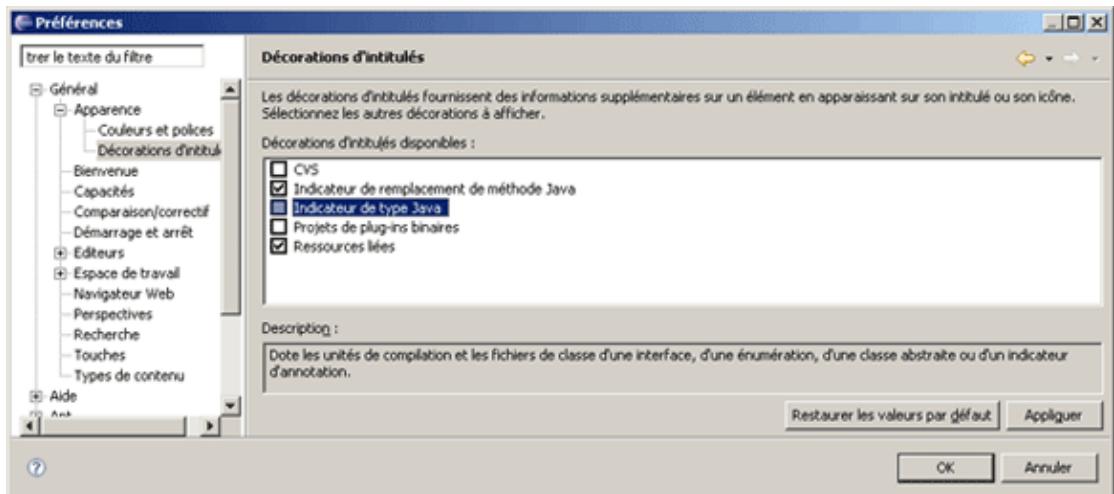


Il suffit de cocher les filtres qui doivent être appliqués.

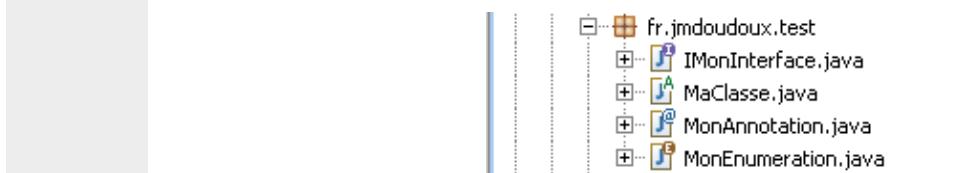
A partir de l'éditeur, il est possible de sélectionner dans la vue "Package", l'élément en cours d'édition en utilisant l'option "Afficher dans la vue package" du menu contextuel.



Il est possible de demander l'utilisation d'une icône dédiée pour les interfaces, les classes abstraites, les annotations et les énumérations en utilisant les Préférences.



Dans l'arborescence « Général / Apparence / Décorations d'intitulés », il suffit de cocher « Indicateur de type Java » et de cliquer sur le bouton « OK ».



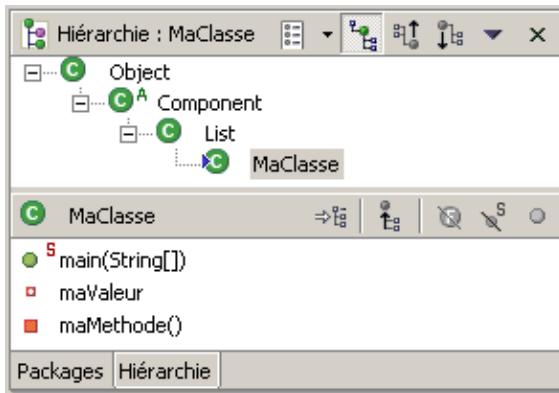
### 7.5.2. La vue "Hiérarchie"

Cette vue affiche la hiérarchie d'un élément. Pour afficher la hiérarchie d'un élément, il y a plusieurs possibilités :

- dans la vue Package sélectionner l'élément et utiliser l'option "Ouvrir la hiérarchie des types" du menu contextuel.
- dans l'éditeur, utiliser le menu contextuel "Ouvrir la hiérarchie des types"

Elle se compose de deux parties :

- une partie supérieure qui affiche la hiérarchie de l'élément.
- une partie inférieure qui affiche la liste des membres de l'élément

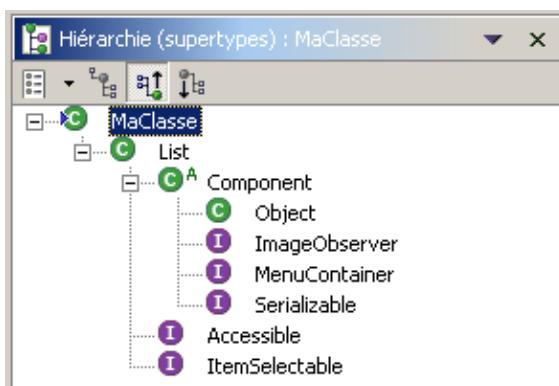


Le bouton permet de sélectionner dans un historique un élément qui a déjà été affiché dans la vue.



Il suffit de sélectionner l'élément concerné et de cliquer sur le bouton "OK".

Le bouton permet d'afficher la hiérarchie des classes mères et des interfaces qu'elles implémentent de l'élément courant.



Le bouton menu permet de changer la présentation de la vue :

- : les deux parties sont affichées horizontalement
- : les deux parties sont affichées verticalement
- : n'affiche que la partie qui présente la hiérarchie

Le bouton permet de masquer les champs.

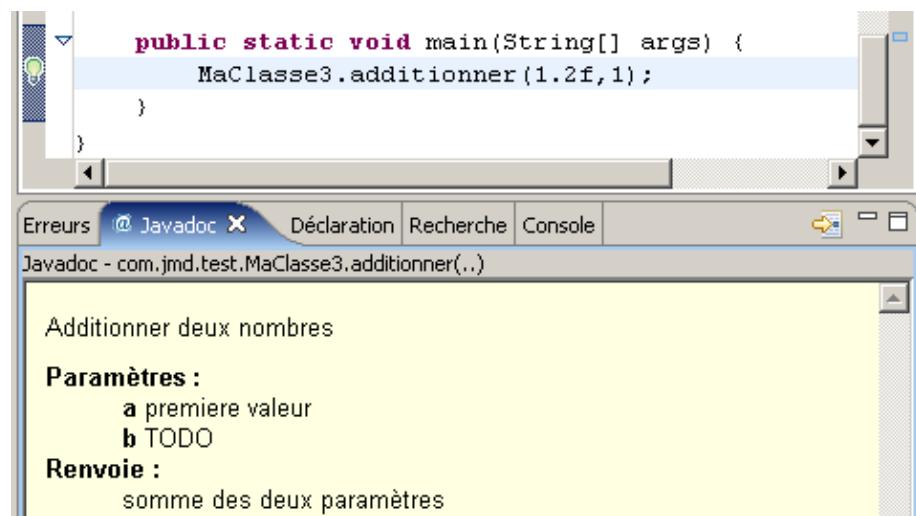
Le bouton permet de masquer les membres statiques.

Le bouton permet de masquer tous les membres qui ne sont pas publics.

### 7.5.3. La vue "Javadoc"



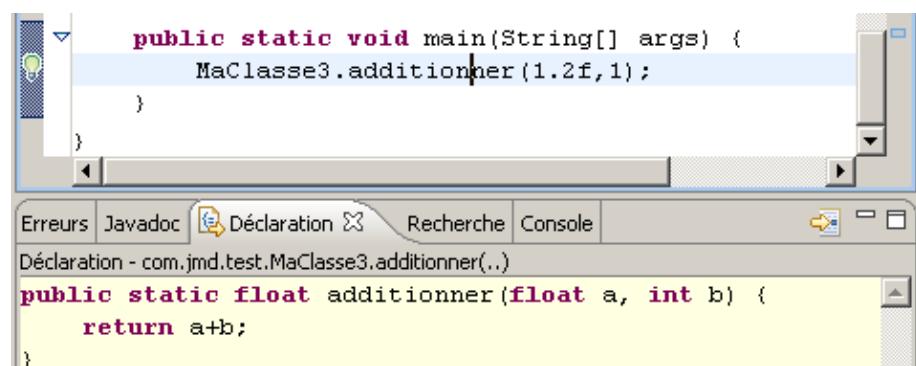
La vue "Javadoc" permet d'afficher la documentation de l'élément sélectionné dans la perspective "Java".



### 7.5.4. La vue "Déclaration"



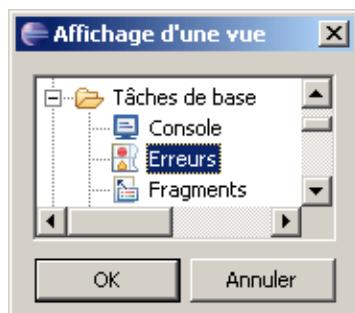
La vue "Déclaration" permet d'afficher le code de l'élément sélectionné dans la perspective "Java".



### 7.5.5. La vue Erreurs

Cette vue affiche les erreurs et en particulier celles concernant les erreurs dans le code source des classes Java.

Pour afficher cette vue, il faut utiliser le menu principal « Fenêtre / Afficher la vue / Autre ... »



Selectionnez « Tâches de base / Erreurs » dans l'arborescence et cliquez sur le bouton « OK »

3 erreur(s), 0 avertissement(s), 0 info(s) (Le filtre a détecté 3 sur 568 éléments)			
Description	Ressource	Dans le dossier	Emplac...
☒ string ne peut pas être résolu en type	ClasseTest.java	tests	ligne 13
☒ Erreur de syntaxe sur le sème "3", supprimez ce sème	ClasseTest.java	tests	ligne 23
☒ Variable locale i en double	TestErreurs.java	tests	ligne 9

Il est possible d'ouvrir l'éditeur directement sur la ligne concernée par l'erreur en double cliquant sur l'erreur concernée.

Il est possible d'appliquer des filtres sur cette vue qui par défaut s'applique à tout le plan de travail.

Il suffit de cliquer sur le bouton pour ouvrir la boîte de dialogue « Filtres ».



Le filtre peut limiter l'affichage des erreurs concernant un type, un ensemble de ressources, contenant tout ou partie d'une description et le niveau de gravité des erreurs.

Il est aussi possible de trier les erreurs dans un ordre particulier. Pour cela, il faut ouvrir le menu déroulant en cliquant sur le bouton et sélectionner l'option « Tri ... ».

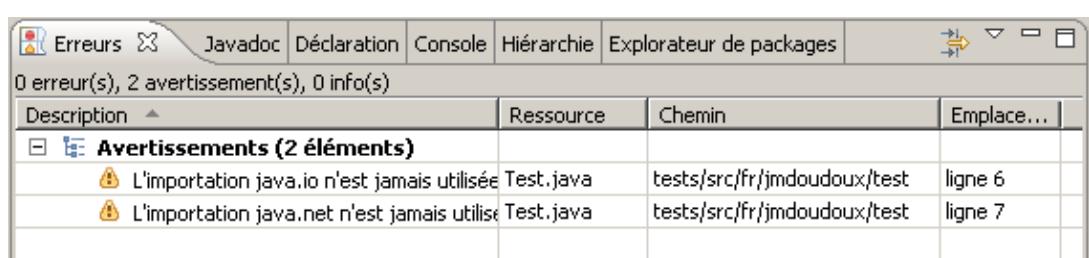


Le tri peut se faire sur quatre critères parmi ceux proposés en précisant pour chaque le sens de l'ordre de tri.

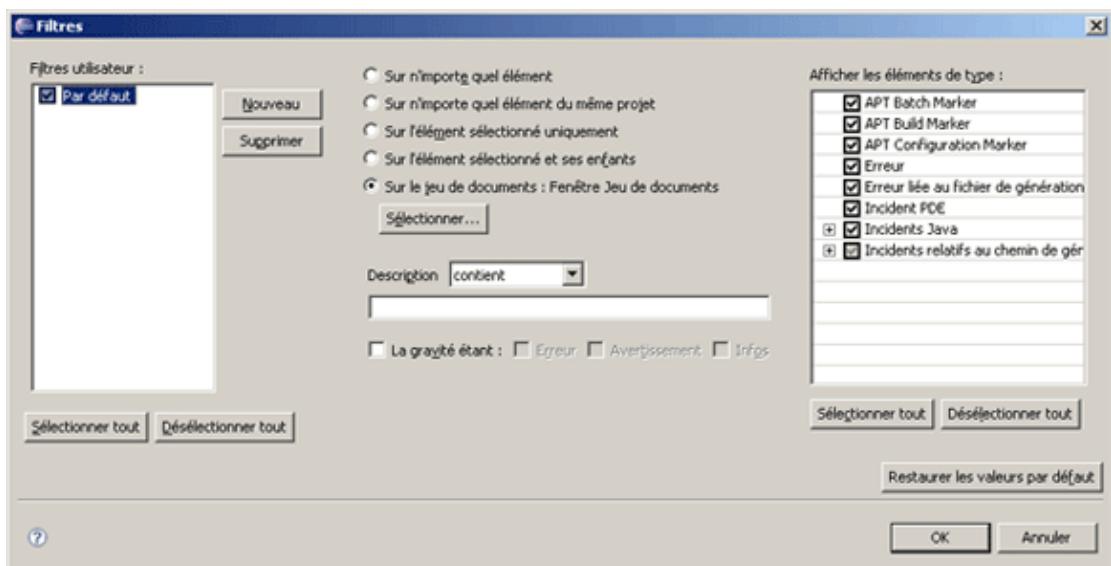
Pour appliquer le tri, il suffit de cliquer sur le bouton « OK ».



La vue affiche les différentes erreurs regroupées par type.



Le bouton permet de gérer les filtres



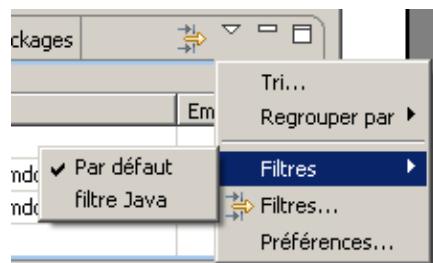
Cliquez sur le bouton « Nouveau »



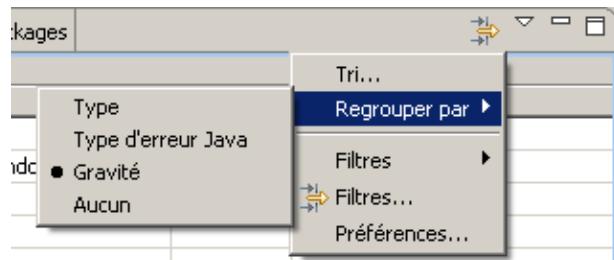
Saisissez le nom du filtre et cliquez sur le bouton « OK »

Saisissez les caractéristiques du filtre et cliquez sur le bouton « OK »

Le bouton affiche le menu déroulant : le filtre créé peut être sélectionné dans le sous menu filtres



Le menu « Regrouper par » permet de sélectionner le type de regroupement



Exemple avec un regroupement par gravité

Erreurs X Javadoc Déclaration Console Hiérarchie Explorateur de packages				
2 erreur(s), 1 avertissement, 0 info(s)				
Description	Ressource	Chemin	Emplace...	
<b>Erreurs (2 éléments)</b>				
Impossible d'établir une référence static	Test.java	tests/src/fr/jmdoudoux/test	ligne 23	
Impossible d'établir une référence static	Test.java	tests/src/fr/jmdoudoux/test	ligne 23	
<b>Avertissements (1 élément)</b>				
Le champ Test.i n'est jamais lu localement	Test.java	tests/src/fr/jmdoudoux/test	ligne 15	

Exemple avec un regroupement par Type

Erreurs X Javadoc Déclaration Console Hiérarchie Explorateur de packages				
2 erreur(s), 1 avertissement, 0 info(s)				
Description	Ressource	Chemin	Emplace...	
<b>Incidents Java (3 éléments)</b>				
Impossible d'établir une référence static	Test.java	tests/src/fr/jmdoudoux/test	ligne 23	
Impossible d'établir une référence static	Test.java	tests/src/fr/jmdoudoux/test	ligne 23	
Le champ Test.i n'est jamais lu localement	Test.java	tests/src/fr/jmdoudoux/test	ligne 15	

## Exemple avec un regroupement par type d'erreur Java

The screenshot shows the Eclipse IDE's Error view window titled "Erreurs". It displays three error entries:

Description	Ressource	Chemin	Emplace...
<b>Erreurs bloquantes (2 éléments)</b>			
Impossible d'établir une référence static Test.java Impossible d'établir une référence static Test.java	tests/src/fr/jmdoudoux/test	ligne 23 ligne 23	
<b>Code inutile (1 élément)</b>			
Le champ Test.i n'est jamais lu localement Test.java	tests/src/fr/jmdoudoux/test	ligne 15	

L'option « Préférences » du menu permet de modifier les paramètres



### 7.5.6. La vue Historique



La vue Historique affiche ensemble les différentes versions des éléments de l'historique local et éventuellement du système de gestion de version connectés au projet.

Pour l'afficher, il faut utiliser l'option « afficher la vue / autres » du menu principal fenêtre.



Il faut sélectionner l'élément « Historique » dans « Équipe ».

## 7.6. L'éditeur de code

Le JDT propose un éditeur dédié au fichier contenant du code Java. Il propose des fonctionnalités particulièrement pratiques pour le développement de code Java notamment :

- la coloration syntaxique
- la complétion de code
- le formatage du code source
- l'importation et l'exportation de code via un assistant
- une forte synergie avec le débogueur

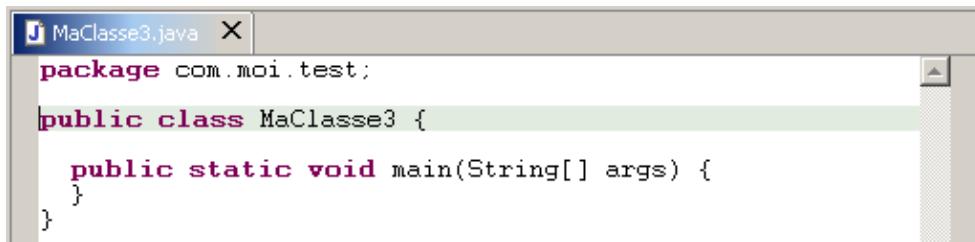
Pour ouvrir un élément dans l'éditeur, il y a deux façons principales :

- double cliquer sur un élément dans la vue "Navigateur"
- double cliquer sur un élément dans la vue "Packages"

L'éditeur peut être invoqué sur un fichier .java ou un fichier .class. Dans le cas d'un fichier .class, si le fichier source est disponible dans l'IDE, alors l'éditeur affiche le contenu du code source.

### 7.6.1. Utilisation de l'éditeur de code

La ligne où se situe le curseur apparaît en gris.

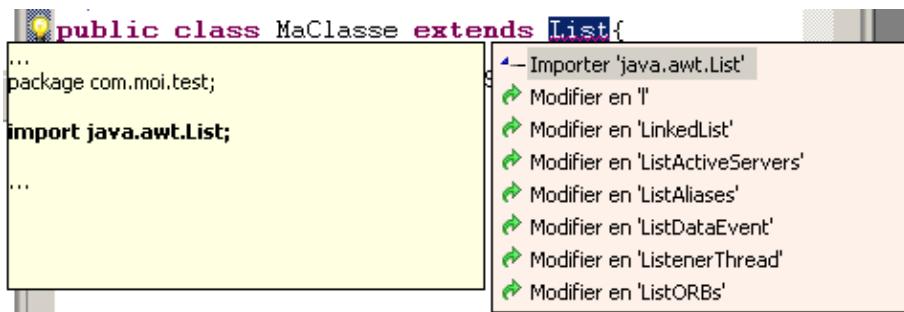


A screenshot of the Eclipse Java editor window titled "MaClasse3.java". The code shown is:

```
package com.moi.test;  
  
public class MaClasse3 {  
    public static void main(String[] args) {  
    }  
}
```

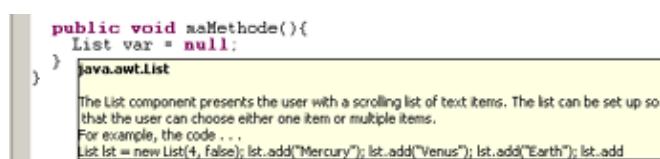
De chaque côté de la zone d'édition, il y a une colonne qui peut contenir de petites icônes pour fournir des informations à l'utilisateur.

Par exemple, si l'on fait hériter une classe d'une classe dont le package n'est pas importé, un clic sur la petite ampoule jaune permet d'obtenir des propositions de corrections.



Il suffit de sélectionner une des actions proposées dans la liste pour que celle-ci soit automatiquement mise en œuvre. Un aperçu des modifications impliquées par l'action sélectionnée est affiché dans une bulle d'aide.

Le bouton  de la barre d'outils permet, s'il est sélectionné, d'afficher une bulle d'aide contenant des informations sur l'élément sous lequel est le curseur.



Une description plus détaillée peut être obtenue en positionnant le curseur sur l'élément et en appuyant sur la touche F2 ou en sélectionnant l'option "Afficher une description de type infobulles" du menu "Editer".



Les méthodes héritées qui sont réécrites sont signalées par une petite icône.

```
MaClasseFille.java
package com.jmd.test;

public class MaClasseFille extends MaClasseMere {

    public MaClasseFille(int valeur) {
        super(valeur);
    }

    public void ajouter(int quantite) {
        super.ajouter(quantite);
    }
}
```

Un clic sur cette petite icône permet d'ouvrir l'éditeur sur la méthode de la classe mère.

```
MaClasseFille.java
this.valeur = valeur;
}

public void ajouter(int quantite){
    valeur += quantite;
}

public int getValeur() {
    return valeur;
}

MaClasseMere.java
```

## 7.6.2. Compléction de code

La compléction de code permet de demander à l'IDE de proposer des suggestions pour terminer le morceau de code en cours d'écriture. Dans l'éditeur Eclipse, pour l'activer, il suffit d'appuyer sur les touches Ctrl et espace en même temps.

```
MaClasse3.java
package com.moi.test;

public class MaClasse3 {

    public static void main(String[] args) {
        System.out.println();
    }
}
```

print(boolean arg0) void - PrintStream  
print(char arg0) void - PrintStream  
print(char[] arg0) void - PrintStream  
print(double arg0) void - PrintStream  
print(float arg0) void - PrintStream  
print(int arg0) void - PrintStream  
print(long arg0) void - PrintStream  
print(Object arg0) void - PrintStream  
print(String arg0) void - PrintStream

Cette fonction peut être appelée alors qu'aucune ou une partie du code à compléter est saisie.

The screenshot shows the Eclipse IDE interface with a Java file named "MaClasse3.java" open. The code contains a main method that prints "bonjour". A code completion dropdown menu is displayed, listing methods from the Object class: clone(), equals(Object arg0), finalize(), hashCode(), and toString().

```
package com.moi.test;

public class MaClasse3 {

    public static void main(String[] args) {
        System.out.println("bonjour");
    }
}
```

- MaClasse3 - com.moi.test
- clone() Object - Object
- equals(Object arg0) boolean - Object
- finalize() void - Object
- hashCode() int - Object
- toString() String - Object

La complétion de code s'adapte au contexte dans lequel elle est appelée. Par exemple, elle peut être appelée pour compléter une clause d'importation. Dans ce cas, elle propose une liste de packages en tenant compte du code déjà saisi.

The screenshot shows the Eclipse IDE interface with the same Java file "MaClasse3.java". The code includes an import statement for the java package. A code completion dropdown menu is shown, listing various sub-packages under java, such as applet, awt, awt.color, awt.datatransfer, awt.dnd, awt.dnd.peer, awt.event, awt.font, and awt.geom.

```
package com.moi.test;

import java.*;

public class MaClasse3 {
    public static void main(String[] args) {
        System.out.println("bonjour");
    }
}
```

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- java.awt.dnd
- java.awt.dnd.peer
- java.awt.event
- java.awt.font
- java.awt.geom

L'éditeur peut générer la structure d'un commentaire Javadoc dans le source. Par exemple, avant une méthode, il suffit de saisir `/**` puis d'appuyer sur la touche " Entrée ".

The screenshot shows the Eclipse IDE interface with the Java file "MaClasse3.java". The code includes a main method. Above it, a Javadoc comment block is being generated, starting with `/**`. The cursor is positioned at the end of the block, after the closing `*/`.

```
package com.moi.test;

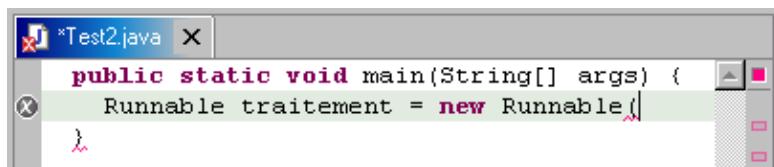
public class MaClasse3 {

    /**
     * 
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("bonjour");
    }
}
```

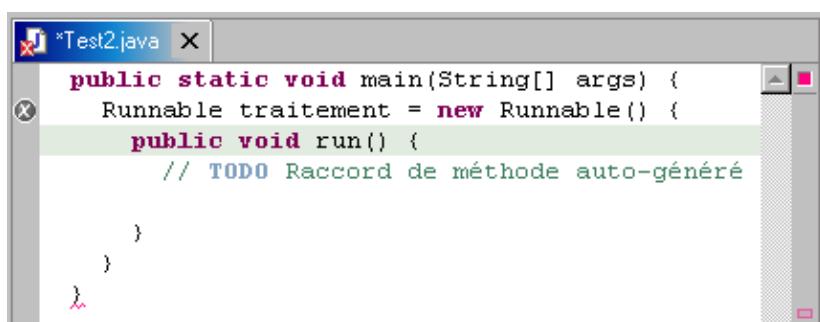
L'appel de la complétion de code en appuyant sur les touches `Ctrl + Espace` permet aussi de faciliter la saisie des commentaires de type Javadoc.



La complétion de code permet de générer une classe interne à partir d'une interface.

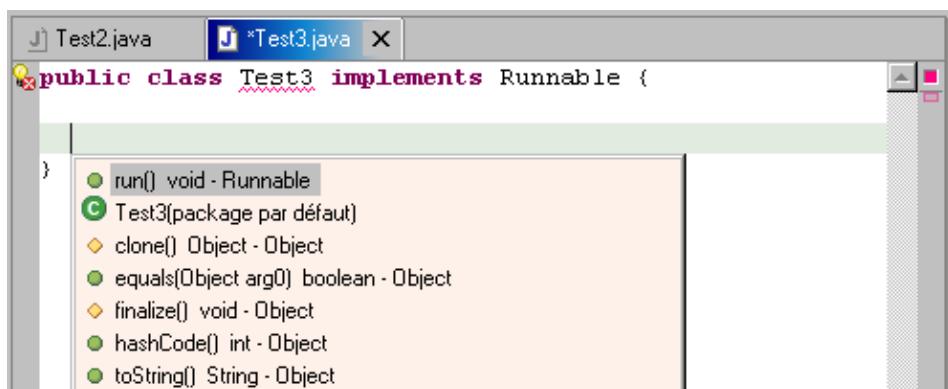


Après la saisie de la parenthèse ouvrante, il suffit d'appuyer sur les touches Ctrl + Espace. L'éditeur va générer la classe interne qui implémente l'interface.



Il suffit de rajouter le ; à la fin de la déclaration de la classe interne et mettre le code nécessaire à la place du commentaire représentant une tâche à faire.

La complétion de code permet aussi de définir des méthodes définies dans une interface implementée dans la classe ou de redéfinir une méthode héritée.



Dans le code du corps de la classe, il suffit d'appuyer directement sur les touches Ctrl + Espace. L'éditeur propose la liste des méthodes à implémenter ou à redéfinir en fonction de la déclaration de la classe.

La signature de la méthode sélectionnée est générée dans le code.

The screenshot shows the Eclipse IDE interface with two tabs open: "Test2.java" and "\*Test3.java". The code editor displays the following Java code:

```
public class Test3 implements Runnable {  
    /* (non-Javadoc)  
     * @see java.lang.Object#toString()  
     */  
    public String toString() {  
        // TODO Raccord de méthode auto-générée  
        return super.toString();  
    }  
}
```

Par défaut, dans le cas de la redéfinition d'une méthode, l'appel à la méthode correspondante de la classe mère est appelée.



Il est possible d'invoquer l'assistant de code en saisissant uniquement les premières lettres capitalisées de l'entité désirée

Exemple :

The screenshot shows the Eclipse IDE interface with a code editor displaying:

```
private void tester() {  
    throw new IAE  
}
```

A code completion dropdown menu is open, listing various exception classes under the category "C" (Constructor):

- C IllegalStateException - java.lang
- C IllegalAccessException - java.lang
- C IllegalAccessError - java.lang
- C IncompleteAnnotationException - java.lang.annotation
- C InstanceAlreadyExistsException - javax.management
- C InvalidActivityException - javax.activity
- C InvalidApplicationException - javax.management
- C InvalidAttributesException - javax.naming.directory

At the bottom of the dropdown, a message reads: "Appuyez sur 'Ctrl+Espace' pour afficher Propositions de modèles".

Cette possibilité peut être désactivée en décocher la case « Afficher les correspondances avec majuscules et minuscules » dans les préférences « Java/Editeur/Affiche l'assistant de contenu ».

L'appel de l'assistant de contenu affiche les propositions regroupées par catégories.

Exemple : saisir do dans l'éditeur de code Java et appuyer sur Ctrl+espace

The screenshot shows the Eclipse IDE interface with a code editor displaying:

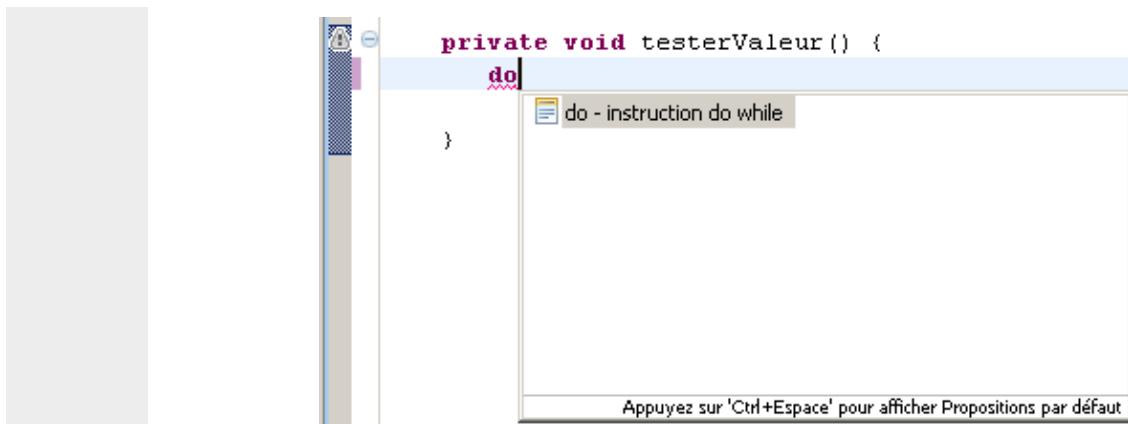
```
private void testerValeur() {  
    do  
}
```

A code completion dropdown menu is open, showing proposals grouped by category:

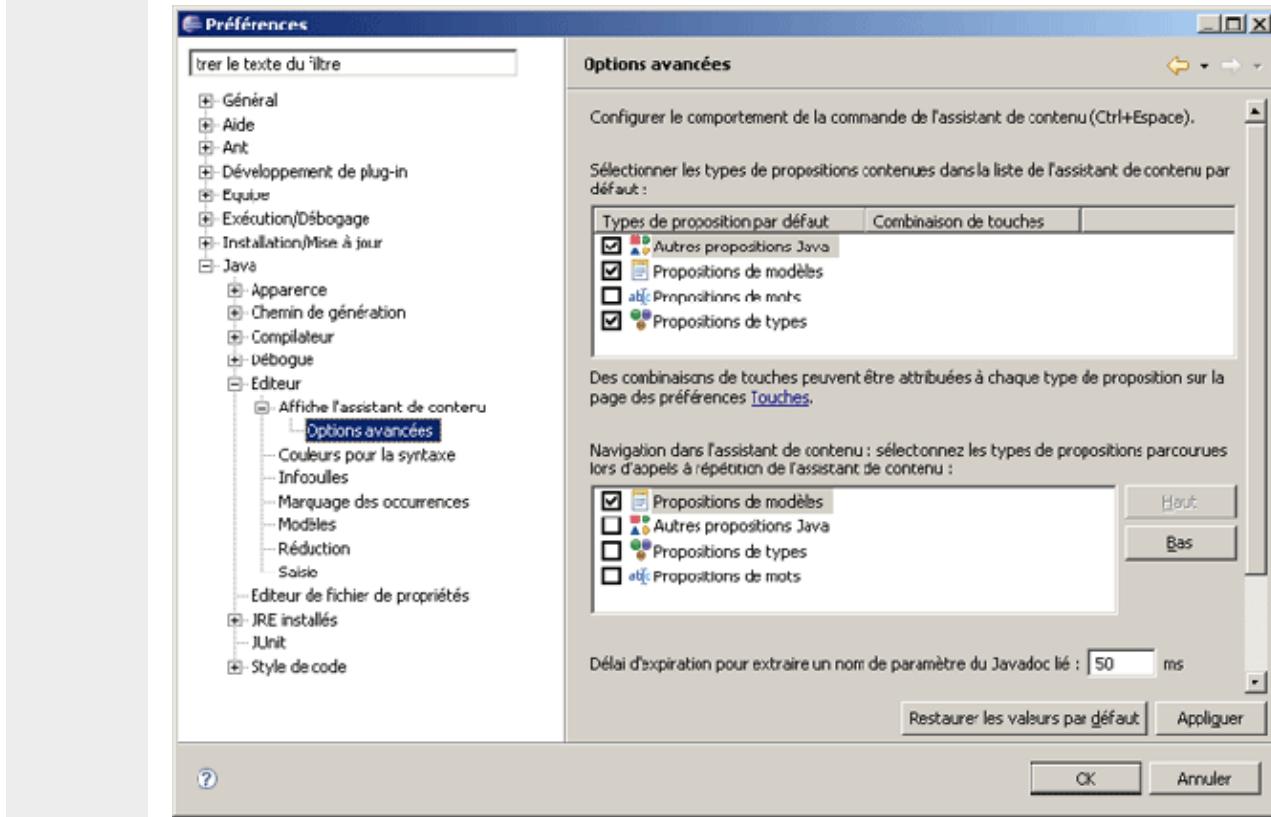
- do - instruction do while
- do
- double
- C Double - java.lang
- I Doc - javax.print
- I DocAttribute - javax.print.attribute
- I DocAttributeSet - javax.print.attribute
- C DocFlavor - javax.print
- C DockingListener - javax.swing.plaf.basic.BasicToolBarUI

At the bottom of the dropdown, a message reads: "Appuyez sur 'Ctrl+Espace' pour afficher Propositions de modèles".

un second appui sur Ctrl+espace affiche une autre catégorie de propositions



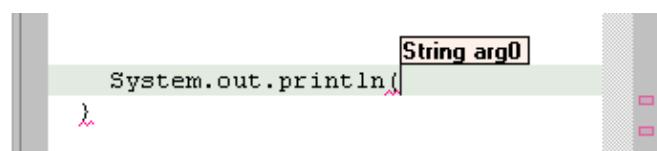
Le mode de fonctionnement peut être configuré dans les préférences



### 7.6.3. Affichage des paramètres sous la forme d'une bulle d'aide

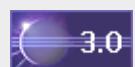
Il est quasiment impossible de retenir les arguments nécessaires à toutes les méthodes de toutes les classes utilisées. L'éditeur d'Eclipse propose d'afficher sous la forme d'une bulle d'aide, les paramètres avec leur type pour la méthode en cours de rédaction dans le code.

Pour utiliser cette fonction, il suffit d'appuyer sur les touches Ctrl + Maj + Espace en même temps dans l'éditeur pendant la saisie d'un ou des paramètres d'une méthode.



Si la méthode est surchargée, alors Eclipse demande de choisir la méthode à utiliser pour ainsi déterminer avec précision les paramètres à afficher dans la bulle d'aide.

#### 7.6.4. Hiérarchie de type dans une bulle d'aide



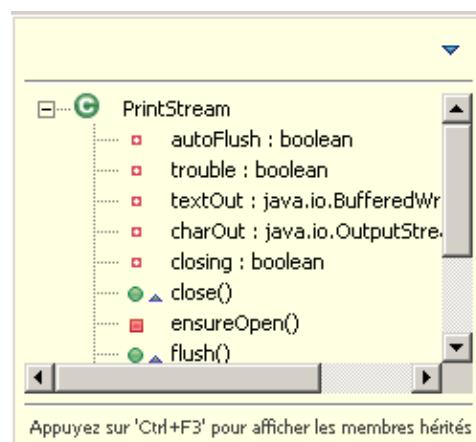
Il est possible demander dans l'éditeur de code d'afficher une bulle d'aide contenant la hiérarchie d'un type. Pour cela, il suffit de mettre le curseur sur un type, une méthode ou un package et d'appuyer sur la combinaison de touches Ctrl+T.



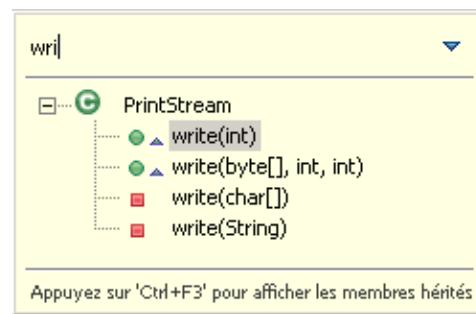
#### 7.6.5. Affichage des membres dans une bulle d'aide



Il est possible demander dans l'éditeur de code d'afficher une bulle d'aide contenant la liste des membres d'un type. Pour cela, il suffit de mettre le curseur sur un type ou une méthode et d'appuyer sur la combinaison de touches Ctrl+F3.



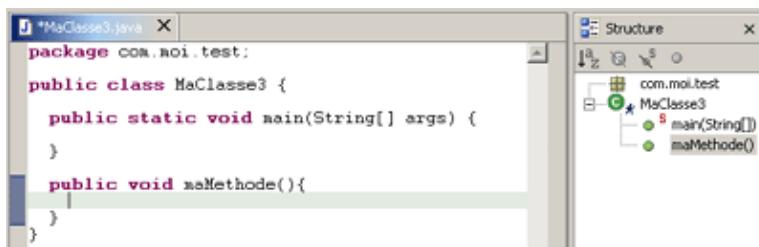
Il est possible de filtrer les éléments en saisissant les premiers caractères des membres à afficher :



#### 7.6.6. L'éditeur et la vue Structure

Il existe un lien entre l'éditeur et la vue "Structure". Si cette vue est visible dans la perspective, dès que le curseur se déplace sur un membre de la classe en cours d'édition, le membre concerné est automatiquement

sélectionné dans la vue "Structure".



Les lignes concernant le membre sélectionné sont marqués par une partie grisée dans la colonne de gauche de l'éditeur.

Les modifications apportées dans le code source (ajout, modification ou suppression de membres) sont automatiquement répercutées dans la vue "Structure".

Le bouton  de la barre d'outils permet de limiter l'affichage dans l'éditeur du membre sélectionné dans la vue "Structure".



Pour réafficher le source complet, il suffit de cliquer de nouveau sur le bouton.

### 7.6.7. La coloration syntaxique

L'éditeur possède une fonction de coloration syntaxique. Par défaut, les éléments sont colorés de la façon suivante :

- les mots clés mots du langage sont colorés en violet gras
- les chaînes de caractères sont en bleu
- les commentaires sont en vert
- les commentaires Javadoc sont en bleu plus clair
- les balises Javadoc sont en gris
- les autres éléments sont en noir

Exemple :

```
package com.moi.test;

import java.awt.List;

/**
 * @author user
 *
 * To change this generated comment edit the !*
 */
public class MaClasse extends List{

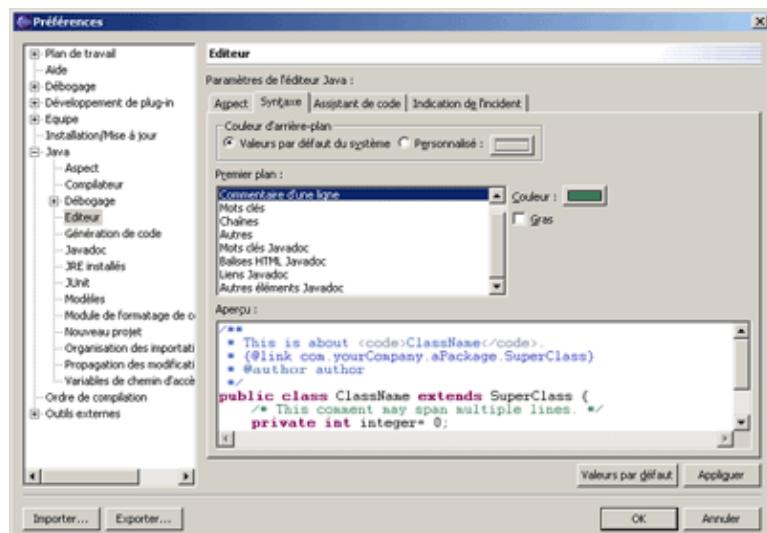
    private int maValeur ;

    public static void main(String[] args) {

        private void maMethode() {}
    }
}
```

Il est possible de modifier ces couleurs par défaut dans les préférences (menu Fenêtres/Préférence)

Il faut sélectionner l'élément Java/Editeur dans l'arborescence. Cet élément possède quatre onglets. L'onglet syntaxe permet de modifier les couleurs.



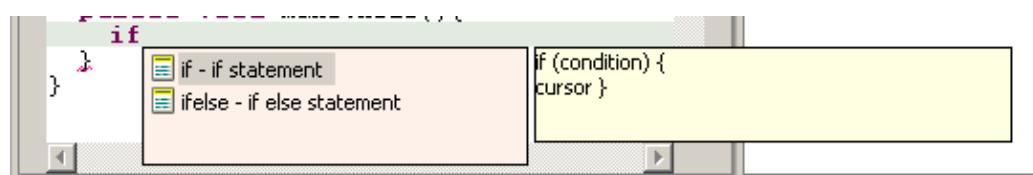
Il suffit de sélectionner l'élément concerné dans la liste déroulante et de sélectionner la couleur qui lui est associée en cliquant sur le bouton couleur. Une boîte de dialogue permet de sélectionner la nouvelle couleur à utiliser.



### 7.6.8. Utilisation des modèles

Il suffit de saisir le nom du modèle et d'appuyer sur les touches Ctrl + espace en même temps

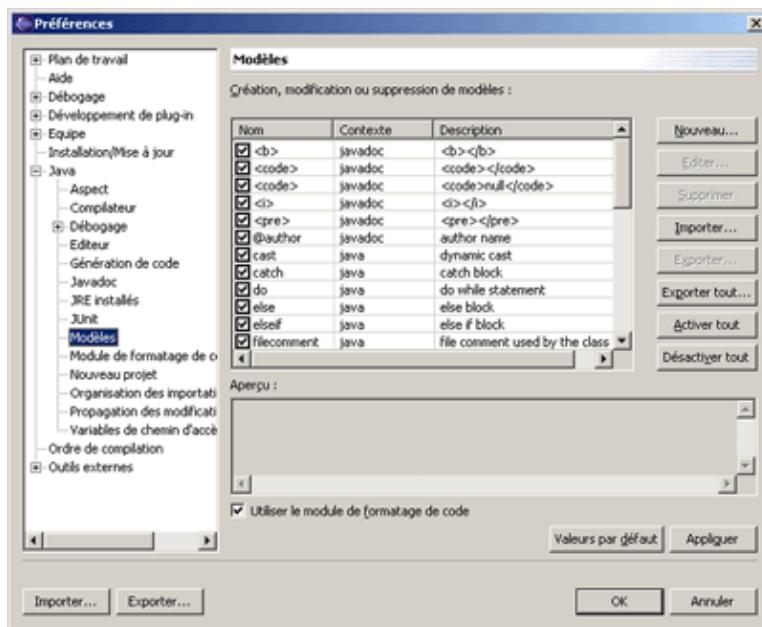
Exemple : avec le modèle if



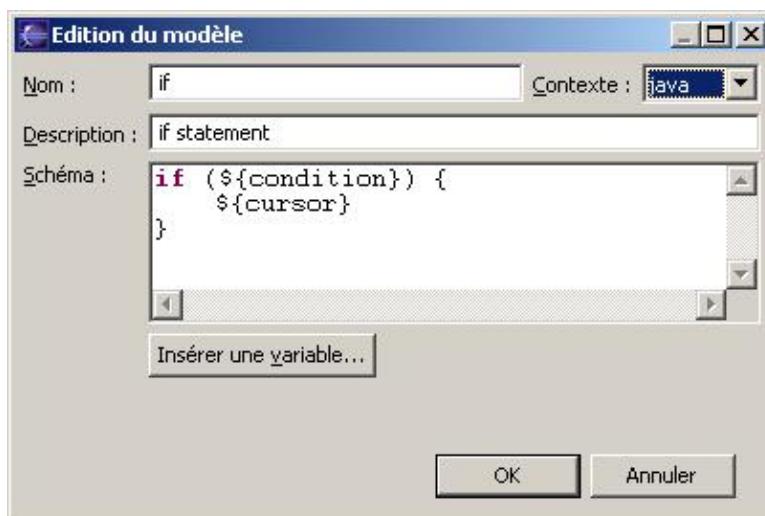
Il suffit de sélectionner le modèle à insérer pour qu'il soit immédiatement insérer dans le code.

Il est possible d'ajouter, de modifier ou de supprimer un modèle en utilisant les préférences (menu Fenêtre/Préférences).

Il suffit de sélectionner dans l'arborescence "Java/Modèles"



Pour modifier un modèle, il suffit de cliquer sur le bouton "Editer"



### 7.6.9. La gestion des importations

Il est possible de faire insérer la clause import pour un élément utilisé dans le code. Pour cela, il suffit de mettre le curseur sur l'élément concerné et de sélectionner l'option "Source/Ajout d'une instruction d'import" ou d'appuyer sur les touches Ctrl + Maj + M.

Si l'élément est déclaré dans un package unique, la clause import est automatiquement générée. Sinon une boîte de dialogue demande de sélectionner l'élément pour lequel la clause import sera générée.



La fonctionnalité "Organisation des importations" est très pratique car elle permet d'insérer automatiquement les clauses imports avec les package requis par le code.

Par exemple : une variable de type Socket est déclarée, sans que le package java.net ne soit importé

```

J *MaClasse3.java X
package com.moi.test;

public class MaClasse3 {

    public static void main(String[] args) {

        public void maMethode(){
            Socket var = null;
        }
    }
}

```

Pour ajouter automatiquement la clause d'importation, il suffit d'utiliser l'option "Source/Organiser les importations" du menu contextuel.

```

J *MaClasse3.java X
package com.moi.test;

import java.net.Socket;

public class MaClasse3 {

    public static void main(String[] args) {

        public void maMethode(){
            Socket var = null;
        }
    }
}

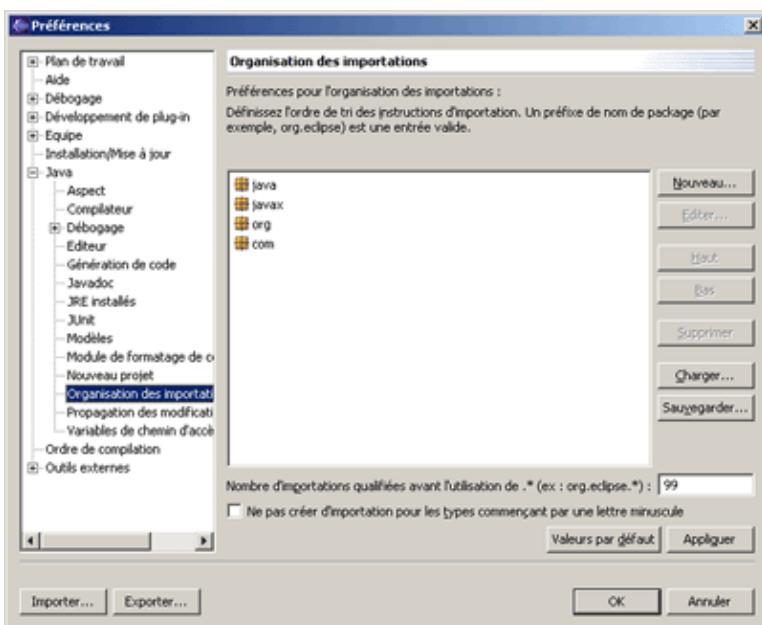
```

La clause import est insérée dans le code. Si un élément est contenu dans plusieurs packages, une boîte de dialogue demande la sélection du type à importer.



Cette fonctionnalité supprime aussi automatiquement les importations qui sont inutiles car aucune classe incluse dans ces packages n'est utilisée dans le code.

Certains réglages de cette fonctionnalité peuvent être effectués dans les préférences (menu "Fenêtre/Préférences"). Il suffit de sélectionner " Java/Organisation des importations " dans l'arborescence.



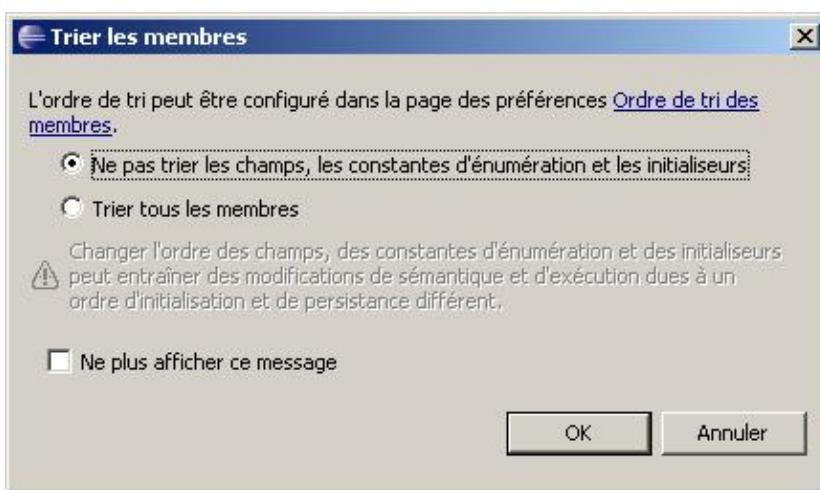
Les importations sont mises à jour lors d'un copier/coller d'une portion de code source. Cette nouvelle fonctionnalité très pratique permet lors d'un copier/coller d'un morceau de code d'une classe dans une autre de mettre à jour les importations requises.

Attention : cette fonctionnalité ne s'applique que si le copier et le coller est fait dans Eclipse.

### 7.6.10. Le tri des membres



Cette fonctionnalité est accessible en utilisant l'option « Source / Trier les membres ... » du menu contextuel de l'éditeur de code Java.



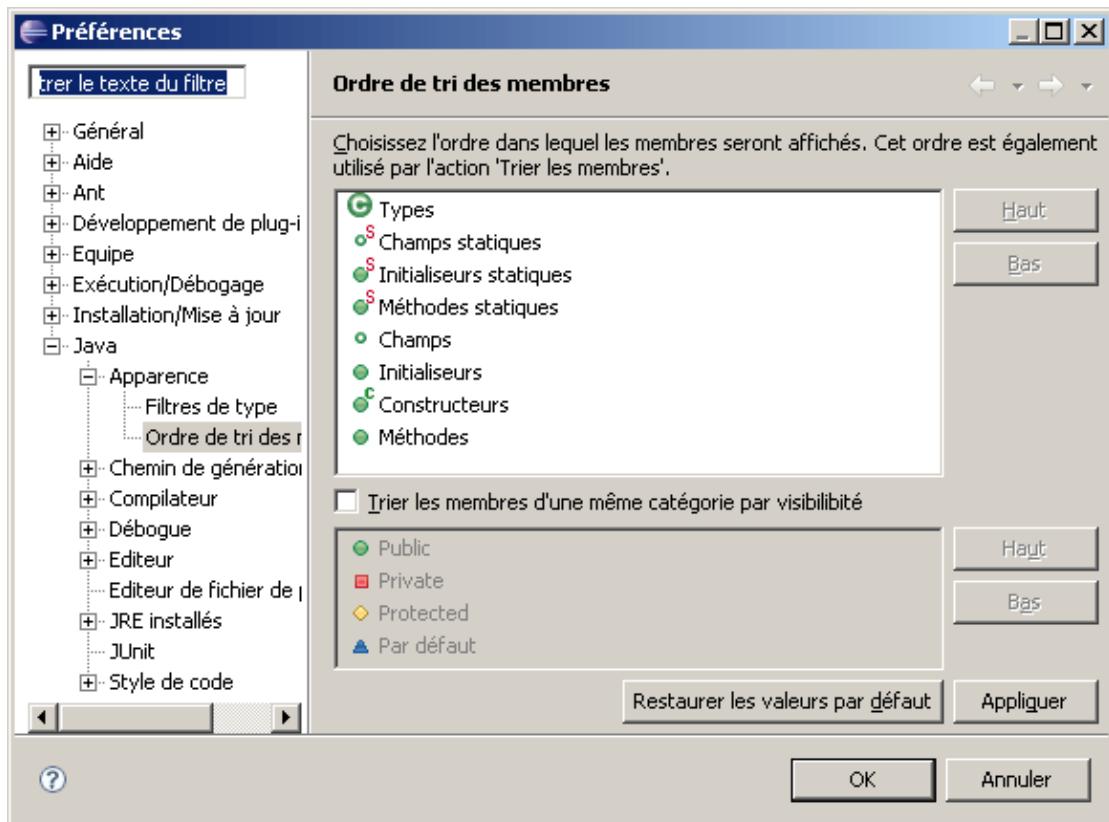
Cette boîte de dialogue permet de sélectionner les membres impactés par le tri.

Cliquez sur le bouton « OK » pour demander le tri des membres du code source.



Cliquez sur le bouton « OK » après avoir pris connaissance du message d'avertissement.

Le lien « Ordre de tri des membres » permet d'ouvrir les préférences sur l'arborescence « Java / Apparence / Ordre de tri des membres ... ».



Pour modifier l'ordre des membres dans le tri, il suffit de sélectionner le type de membres concernés et d'utiliser les boutons « Haut » et « Bas ».

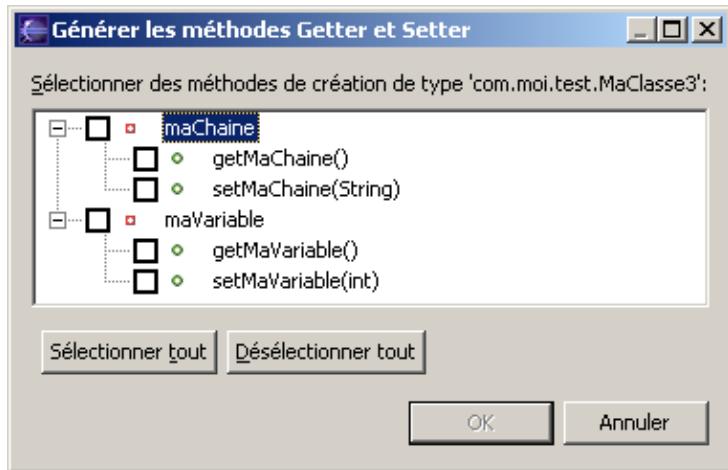
La case à cocher « Trier les membres d'une même catégorie par visibilité » est assez explicite : pour l'activer il suffit de la cocher et de modifier au besoin l'ordre des modificateurs d'accès.

Cliquez sur le bouton « OK » pour valider les modifications.

### 7.6.11. La génération de getter et setter

Il suffit d'utiliser l'option "Source/Générer les méthodes getter et setter" du menu contextuel.

Une boîte de dialogue permet de sélectionner, pour chaque attribut de la classe en cours d'édition, les getters et setters à générer

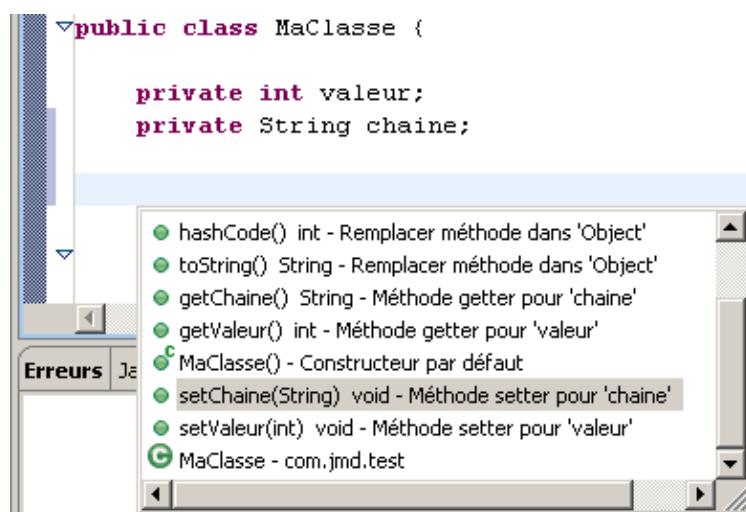


Il suffit de sélectionner les méthodes nécessaires pour qu'une génération par défaut soit effectuée dans le code.



Il est possible de créer directement des méthodes de type getter/setter ou le constructeur par défaut en utilisant la combinaison Ctrl+espace dans le corps d'une classe

Exemple : demander la création d'un setter pour le champ chaîne.



### 7.6.12. La génération des méthodes hashCode() et equals()



Il est possible de demander la génération des méthodes hashCode() et equals() en utilisant, dans l'éditeur de code, l'option « Source / Générer hashCode() et equals() » du menu contextuel.

Exemple :

```
public class Personne {
    private String nom;
    private String Prenom;
}
```

Une boîte de dialogue permet de sélectionner les champs qui vont être utilisés dans le corps de ces méthodes.



Par défaut tous les champs de classe sont sélectionnés.

Cliquez sur le bouton « OK » une fois la sélection effectuée.

Le code de la classe est enrichi des deux méthodes :

#### Exemple :

```
/* (non-Javadoc)
 * @see java.lang.Object#hashCode()
 */
@Override
public int hashCode() {
    final int PRIME = 31;
    int result = 1;
    result = PRIME * result + ((Prenom == null) ? 0 : Prenom.hashCode());
    result = PRIME * result + ((nom == null) ? 0 : nom.hashCode());
    return result;
}

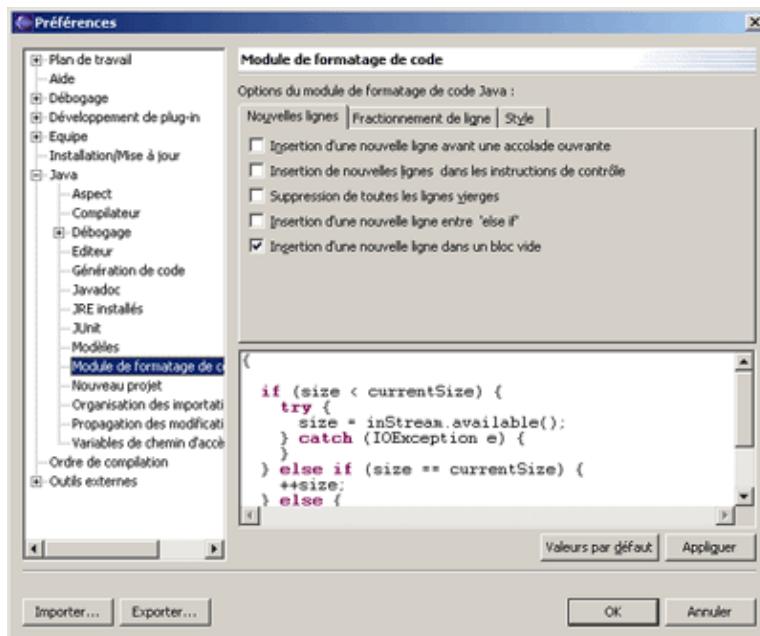
/* (non-Javadoc)
 * @see java.lang.Object>equals(java.lang.Object)
 */
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    final Personne other = (Personne) obj;
    if (Prenom == null) {
        if (other.Prenom != null)
            return false;
    } else if (!Prenom.equals(other.Prenom))
        return false;
    if (nom == null) {
        if (other.nom != null)
            return false;
    } else if (!nom.equals(other.nom))
        return false;
    return true;
}
```

## 7.6.13. Formater le code

L'éditeur peut formater le code selon des règles définies. Pour utiliser cette fonctionnalité, il suffit d'utiliser l'option "Formater" du menu contextuel.

Les règles utilisées pour formater sont définies dans les préférences (menu "Fenêtre/Préférences").

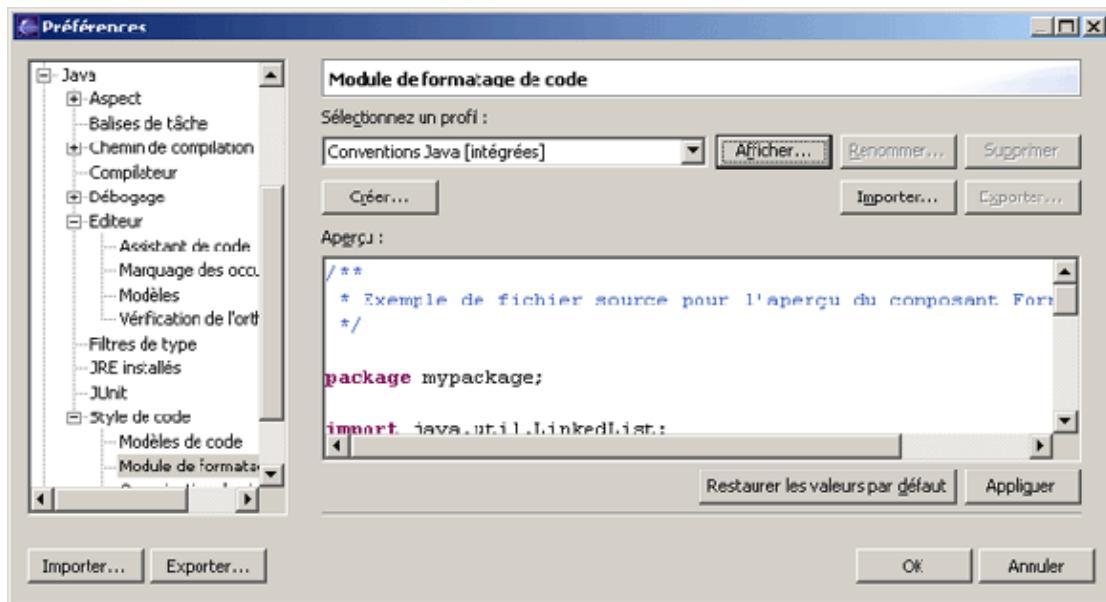
Il faut sélectionner dans l'arborescence, "Java/Module de formatage de code". Les règles de formatage peuvent être définies grâce à trois onglets.



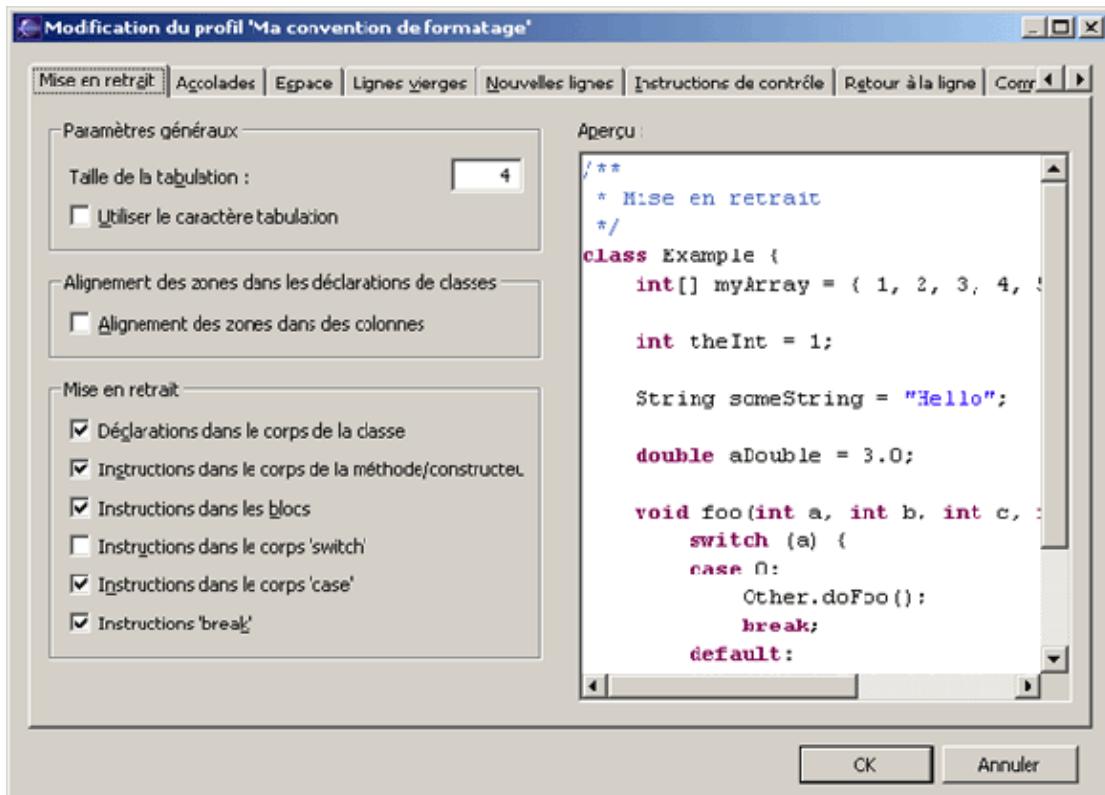
Le module de formatage de code a été entièrement réécrit et propose plusieurs fonctionnalités intéressantes :

- il propose deux profils prédéfinis (convention java et Eclipse 2.1) non modifiable sans duplication
- il permet de créer son propre profil
- il propose un aperçu du style sélectionné
- échanger des profiles entre plusieurs installations d'Eclipse 3 par import/export

Les préférences permettent de mettre en oeuvre ces nouvelles fonctionnalités.



Le bouton « Afficher » permet de voir les différents options qui composent le profil.



Le bouton « Créer » permet de créer son propre style à partir d'un style existant.



Il faut saisir le nom, sélectionner le profil d'origine et cliquer sur le bouton « OK ».

La configuration des paramètres du profil est particulièrement riche et permet un niveau de détails très important.

Dans la perspective Java, il est possible de demander le formatage du code source de tous les fichiers d'un package ou d'un projet en utilisant l'option « Source / Formater » du menu contextuel des éléments.

Une boîte de dialogue demande la confirmation car l'opération ne peut être annulée.



### 7.6.14. Mise en commentaire d'une portion de code

L'éditeur permet de mettre en commentaire une portion de code. Il suffit de sélectionner la portion de code concernée.

```
package com.moi.test;

public class MaClasse10 {

    public void maMethode() {
        int i = 0;
        for(i=0;i<10;i++){
            System.out.println("i = "+i);
        }
    }
}
```

Puis, il faut utiliser l'option " Source / Mettre en commentaires " du menu contextuel de l'éditeur.

```
package com.moi.test;

public class MaClasse10 {

    public void maMethode() {
        // int i = 0;
        // for(i=0;i<10;i++){
        //     System.out.println("i = "+i);
        // }
    }
}
```

Pour supprimer les commentaires sur une portion de code, il suffit de sélectionner la portion de code et d'utiliser l'option " Source / Supprimer la mise en commentaire " du menu contextuel.



Les options de menu « Source / Mettre en commentaire » et « Source / Supprimer les commentaires » sont remplacées par une unique option « Source / Ajout/suppression de la mise en commentaires » qui effectue l'opération en fonction du contexte.

L'option « Source / Mettre en commentaire un bloc » du menu contextuel permet de mettre en commentaire la portion de code sélectionnée grâce à un commentaire de type multi-ligne.

Exemple

```
public static void main(String[] args) {
    System.out.println("bonjour");
    System.out.println("bonjour");
    System.out.println("bonjour");
}
```

Résultat :

```
public static void main(String[] args) {
    /* System.out.println("bonjour");
     * System.out.println("bonjour");
     * System.out.println("bonjour"); */
}
```

Pour utiliser cette fonctionnalité, il est aussi possible d'utiliser la combinaison de touches Ctrl+Maj+/

L'option « Source / Supprimer ma mise en commentaire un bloc » permet l'opération inverse.

### 7.6.15. Protéger une portion de code avec un bloc try/catch

L'éditeur propose une option qui permet automatiquement de rechercher, dans un bloc de code sélectionné, une ou plusieurs exceptions pouvant être levées et de protéger le bloc de code avec une instruction de type try / catch.

Il faut sélectionner le bloc de code à protéger.

```
public void maMethode() {
    String ligne;
    BufferedReader br = null;

    br = new BufferedReader(new FileReader("c:\test.txt"));
    while((ligne = br.readLine()) != null) {
        System.out.println(ligne);
    }
}
```

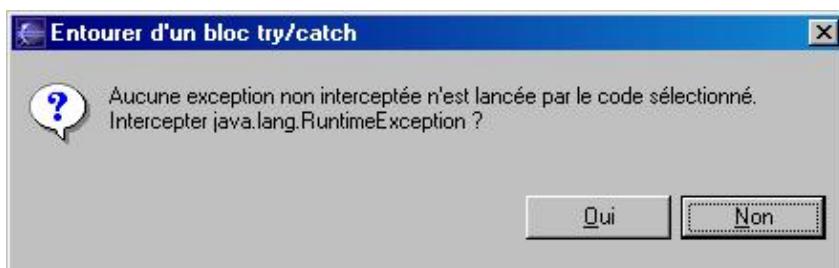
Puis utiliser l'option " Source / Entourer d'un bloc try / catch ". L'éditeur analyse le code et génère la cause try / catch qui va capturer toutes les exceptions qui peuvent être levées dans le bloc de code sélectionné.

```
public void maMethode() {
    String ligne;
    BufferedReader br = null;

    try {
        br = new BufferedReader(new FileReader("c:\\test.txt"));
        while((ligne = br.readLine()) != null) {
            System.out.println(ligne);
        }
    } catch (FileNotFoundException e) {
        // TODO Bloc catch auto-généré
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Bloc catch auto-généré
        e.printStackTrace();
    }
}
```

Chacune des instructions catch est marquée avec une tache " bloc catch auto-généré " pour indiquer au développeur d'ajouter le code nécessaire au traitement de l'exception.

Si le bloc de code ne contient aucun appel de méthode susceptible de lever une exception, une boîte de dialogue demande si l'instruction catch doit capturer une exception de type RuntimeException.



### 7.6.16. La fonctionnalité « Entourer avec »

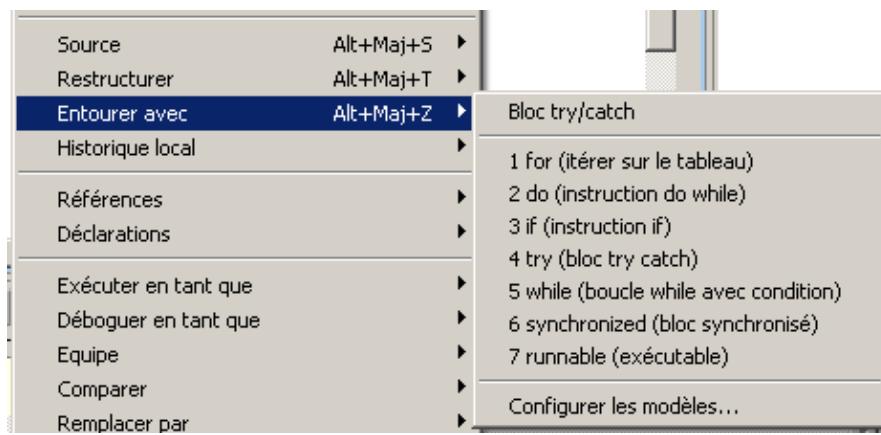


Dans l'éditeur de code Java, il est possible facilement d'entourer une portion de code grâce à la fonctionnalité « Entourer avec ».

Pour cela, sélectionnez une portion de code dans l'éditeur

```
private void testerValeur() {
    System.out.println("test");
}
```

Puis utilisez l'option « Entourer avec » du menu contextuel. Son sous menu propose plusieurs modèles par défaut.



Sélectionnez par exemple « 3 if (instruction if) »

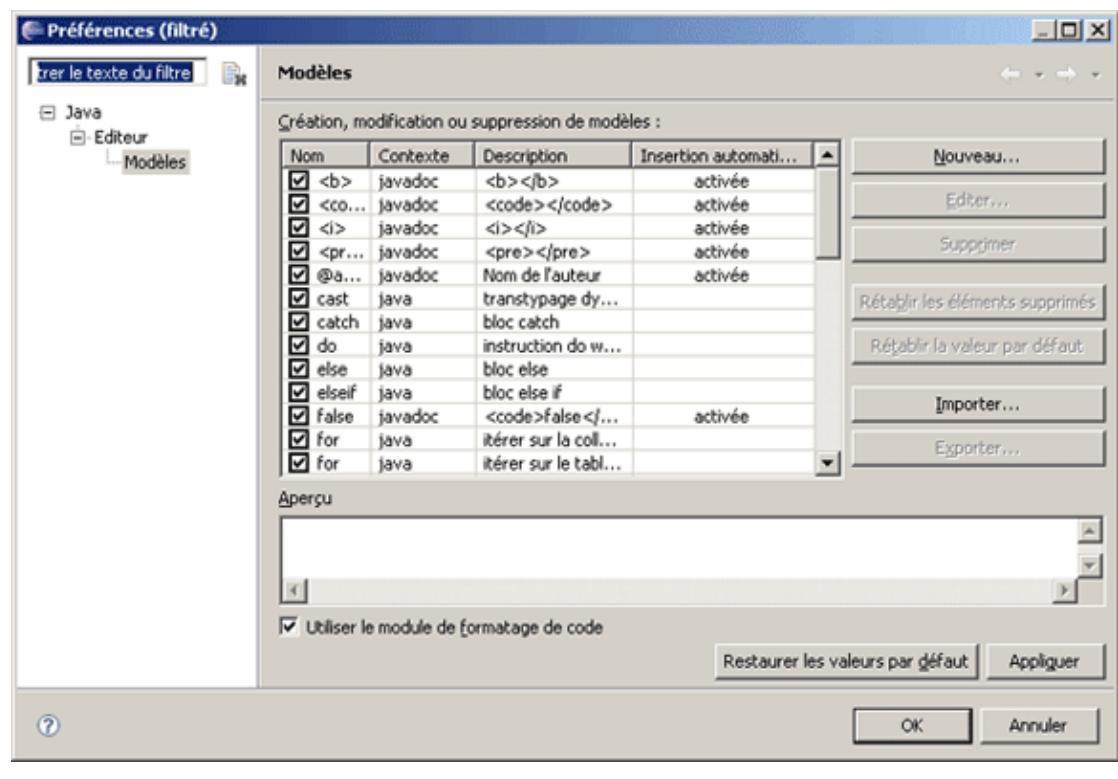
```
private void testerValeur() {
    if (condition) {
        System.out.println("test");
    }
}
```

Il est possible d'accéder directement au sous menu en utilisant la combinaison de touches shift+alt +z

```
private void testerValeur() {
    System.out.println("test");
}

```

Les modèles sont extensibles : pour gérer les modèles, il faut utiliser l'option « Configurer les modèles » du menu contextuel ou ouvrir les préférences et sélectionner « Java / Editeur / Modèles » dans l'arborescence.



Les boutons « Importer » et « Exporter » permettent d'importer et d'exporter les modèles.

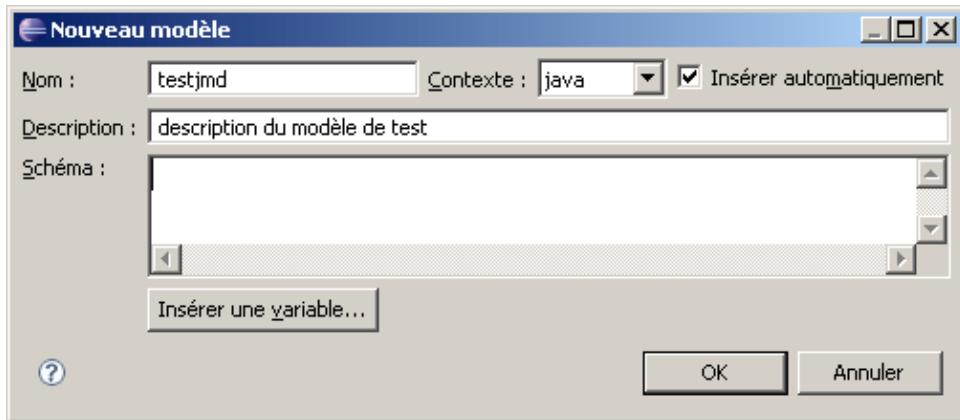
Pour exporter un ou plusieurs modèles, il est nécessaire de les sélectionner au préalable dans la liste afin de pouvoir rendre actif le bouton.

Une boîte de dialogue permet de sélectionner ou de saisir le fichier qui va contenir les données exportées au format xml .

Le bouton « Importer » permet d'importer des modèles sauvegardés dans un fichier xml : il suffit de sélectionner le fichier.

Le bouton « Nouveau » permet de créer un nouveau modèle.

La boîte de dialogue permet de saisir le nom du modèle, une description et le schéma, de sélectionner le contexte et de préciser si l'insertion peut être automatique.



Les modèles existants peuvent être d'une grande aide pour faciliter la compréhension de la rédaction du schéma. Le bouton « Insérer une variable » permet d'insérer une variable prédéfinie.

Une fois toutes les données saisies, cliquez sur le bouton « OK » pour créer le nouveau modèle.

## 7.6.17. Les avertissements

### 7.6.17.1. Détection de l'assignation d'une valeur à un paramètre

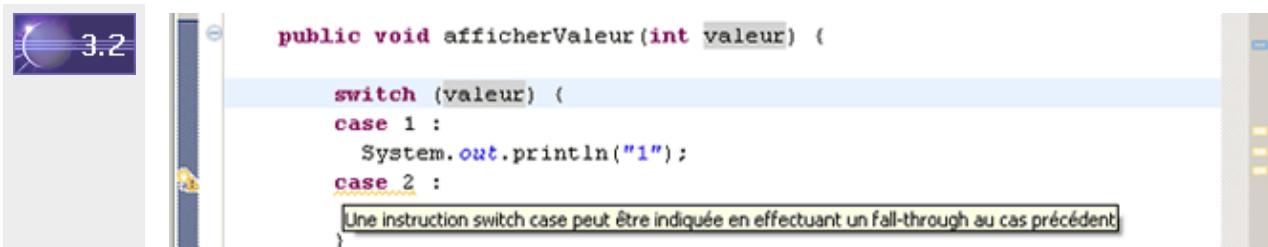


Il est possible de demander au compilateur de signaler les assignements d'une valeur à un paramètre d'une méthode. Par défaut, le compilateur ignore ces actions. Pour les activer, il faut préciser « Avertissement » ou « Erreur » à l'option « Style du code / Affectation de paramètres » dans « Compilateur Java / Erreur avertissements » dans les propriétés du projet.

```
public void afficher(int valeur) {
    valeur = 30;
}
```

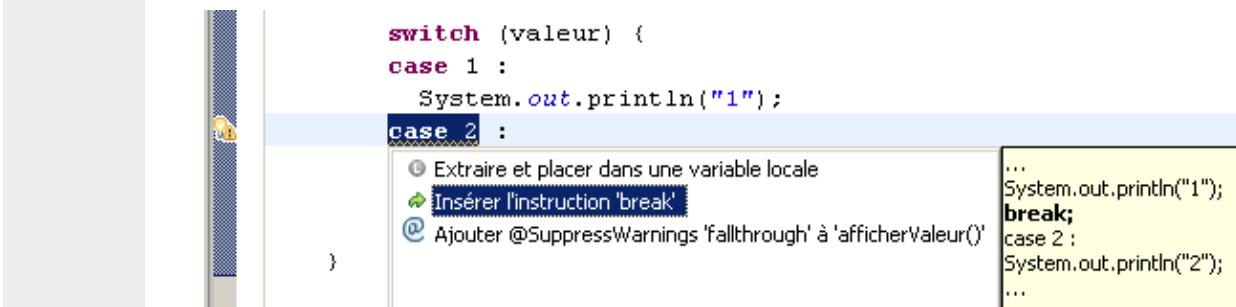
Le paramètre valeur ne doit pas être affecté

### 7.6.17.2. Détection d'un oubli potentiel de l'instruction break



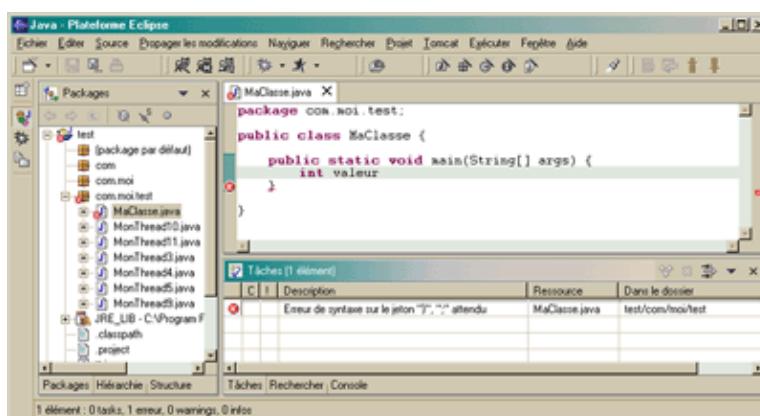
Il est possible de demander au compilateur de signaler les oublis potentiels de l'instruction break dans les cases de l'instruction switch. Par défaut, le compilateur ignore ces actions. Pour les activer, il faut préciser « Avertissement » ou « Erreur » à l'option « Erreurs de programmation possibles / Saut au bloc case suivant » dans « Compilateur Java / Erreur avertissements » dans les propriétés du projet.

Le quick fixe propose d'ajouter l'instruction break manquante.



### 7.6.18. Les erreurs

Lors de la sauvegarde du fichier, celui ci est compilé et les erreurs trouvées sont signalées grâce à plusieurs indicateurs dans différentes vues.



Les erreurs sont signalées par une icône ronde rouge contenant une croix blanche dans les vues suivantes :

- dans l'éditeur, la ou les lignes contenant une ou plusieurs erreurs sont marquées avec cette icône
- dans la vue "Tâches", une entrée pour chaque erreur est créée pour faciliter leur recensement et leur accès
- dans la vue "Packages", tous les éléments allant du fichier source au projet, sont marqués de la petite icône

Dans l'éditeur, le simple fait de laisser le pointeur de la souris sur la petite icône permet d'afficher une bulle d'aide qui précise l'erreur.

The screenshot shows the Eclipse IDE interface with a Java file named "MaClasse.java". The code contains a syntax error: "int valeur" instead of "int value". A red error marker is present at the end of the line. A tooltip message "Erreur de syntaxe sur le jeton '\"', '\"' attendu" is displayed below the cursor. The code is as follows:

```
package com.moi.test;

public class MaClasse {

    public static void main(String[] args) {
        int valeur
    }
}
```

Les boutons et permettent respectivement de se positionner sur l'erreur suivante et sur l'erreur précédente.

Ces mécanismes sont aussi utilisables au cours de la saisie du code Java dans l'éditeur. Eclipse est capable de détecter un certain nombre d'erreurs et de les signaler.

The screenshot shows the Eclipse IDE interface with a Java file named "Test2.java". The code contains a compilation error: "System.out.println("Bonjour");" is highlighted in red. A red error marker is present at the start of the line. A tooltip message "La méthode println(String) n'est pas définie pour le type PrintStream" is displayed below the cursor. The code is as follows:

```
public class Test2 {

    public static void main(String[] args) {
        System.out.println("Bonjour");
    }
}
```

En positionnant le curseur sur la partie de code soulignée en rouge, une bulle d'aide fournit des informations sur l'erreur.

The screenshot shows the Eclipse IDE interface with a Java file named "Test2.java". The code contains a compilation error: "System.out.println("Bonjour");" is highlighted in red. A red error marker is present at the start of the line. A tooltip message "La méthode println(String) n'est pas définie pour le type PrintStream" is displayed below the cursor. The code is as follows:

```
public class Test2 {

    public static void main(String[] args) {
        System.out.println("Bonjour");
    }
}
```

Un clic sur la petite icône en forme d'ampoule jaune, permet d'ouvrir une fenêtre qui propose des solutions de corrections

The screenshot shows the Eclipse IDE interface with a Java file named "Test2.java". The code contains a compilation error: "System.out.println("Bonjour");" is highlighted in red. A red error marker is present at the start of the line. A tooltip message "La méthode println(String) n'est pas définie pour le type PrintStream" is displayed below the cursor. A small yellow lightbulb icon is visible next to the error marker. A context menu is open, showing two options: "Modifier en 'print(..)'" and "Modifier en 'println(..)'". The code is as follows:

```
public static void main(String[] args) {
    System.out.println("Bonjour");
}
```

La seconde fenêtre permet de pré-visualiser les modifications qui seront apportées par la proposition choisie.

La vue erreurs contient aussi les erreurs de compilation du code



Jusqu'à Eclipse 3.1 inclus, la détection d'une erreur empêchait la détection des erreurs suivantes.

```
public class TestErreurs {  
    public void TestA() {  
        int i = 0  
        System.out.println("");  
        TestB();  
    }  
}
```

La détection des erreurs a été améliorée dans version 3.2 d'Eclipse : le même code avec Eclipse 3.2 signale d'autres erreurs.

```
package fr.jmdoudoux.testb;  
public class TestErreurs {  
    public void TestA() {  
        int i = 0  
        System.out.println("");  
        TestB();  
    }  
}
```

## 7.6.19. Masquer certaines portions de code



L'éditeur de code Java propose de masquer certaines portions de code correspondant :

- aux instructions d'importation
- aux commentaires
- aux corps des méthodes

La définition de ces zones est automatique. Le début d'une zone non masquée est signalée par une petit icône dans la barre de gauche de l'éditeur ▾

```
MaClasse.java X
/
 */
public class MaClasse {
    private int valeur;
    public static void main(String[] args) {
        System.out.println("bonjour");
    }
}
```

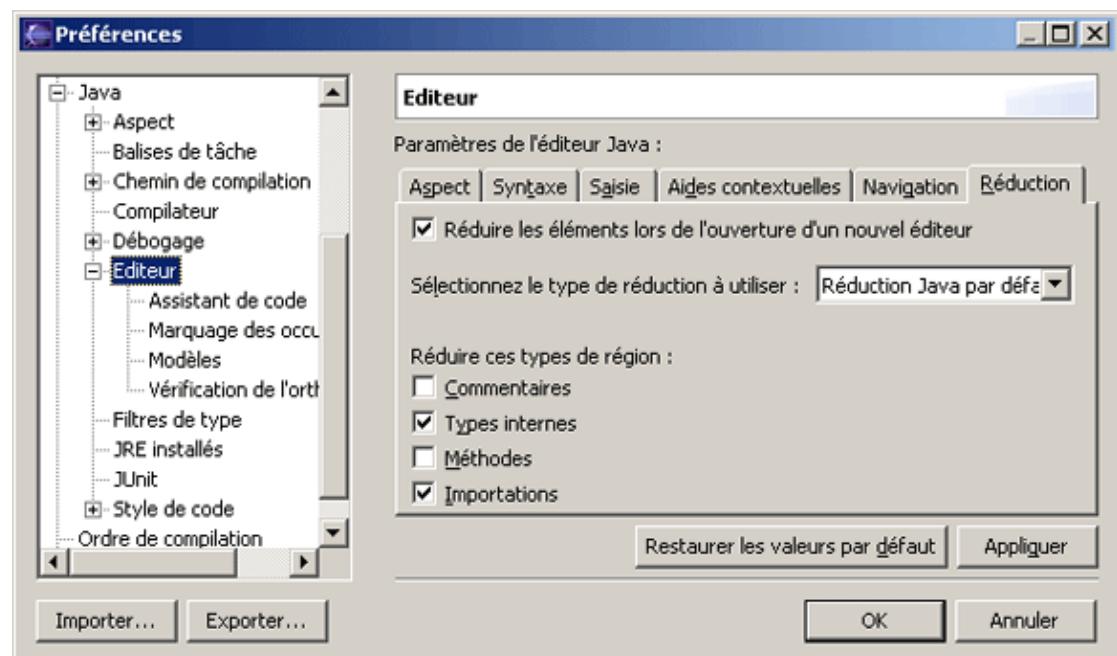
Pour masquer la zone, il suffit de cliquer sur le petit icône ▾. La zone est masquée et l'icône change d'apparence : ▶

```
MaClasse.java X
public class MaClasse {
    private int valeur;
    public static void main(String[] args) {
    }
}
```

En laissant le curseur de la souris sur l'icône ▶, une bulle d'aide affiche le contenu de la zone masquée.

```
private int valeur;
public static void main(String[] args) {
    System.out.println("bonjour");
}
```

Les paramètres de cette fonctionnalité peuvent être modifiés dans les préférences.



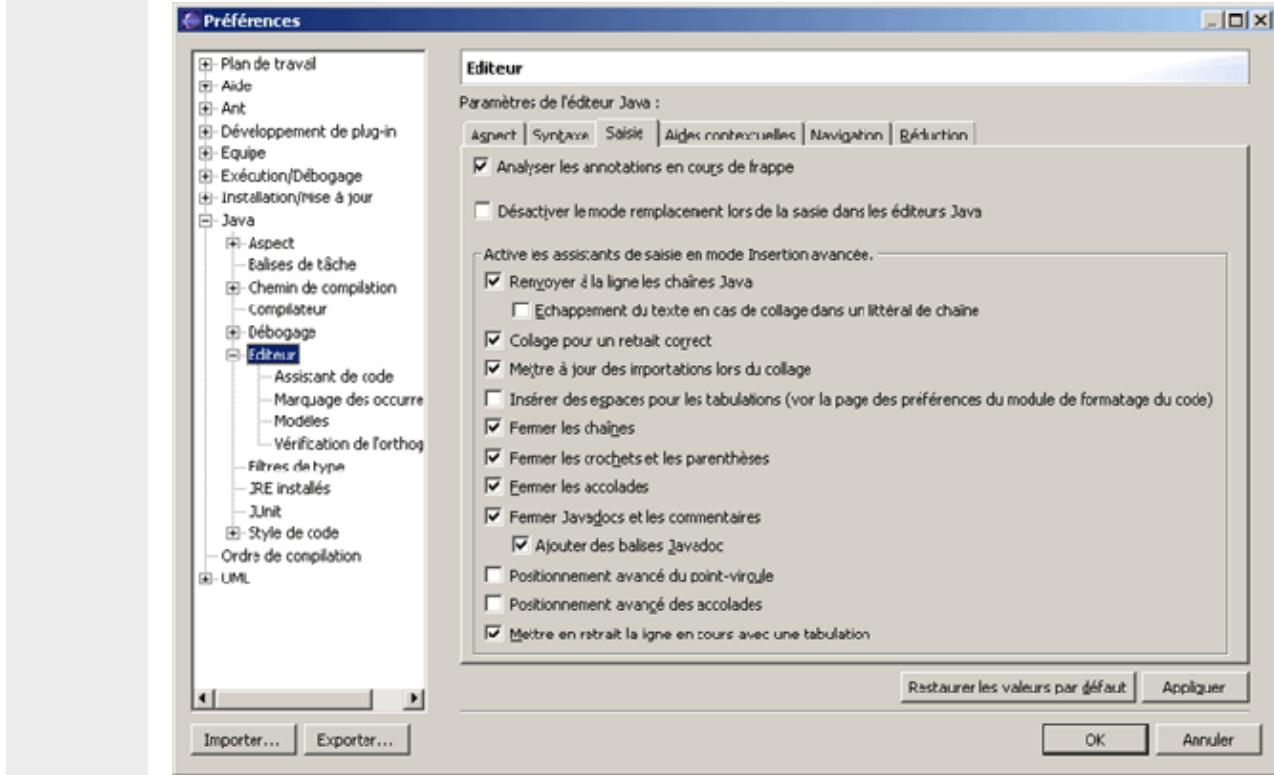
## 7.6.20. Le mode « Insertion Avancée »



Ce mode activé par défaut permet de simplifier la tâche du développeur lors de l'insertion d'une portion de texte dans l'éditeur de code Java. Il permet notamment de fermer automatiquement les parenthèses, les accolades, les guillemets, ...

Le mode de fonctionnement est signalé dans la barre d'état : **Inser...ancée**. Pour basculer du mode normal au mode avancé, il suffit d'utiliser la combinaison de touche Ctrl+Maj+Ins

Le paramétrage de ce mode se fait dans les préférences.



## 7.6.21. Marquer les occurrences trouvées



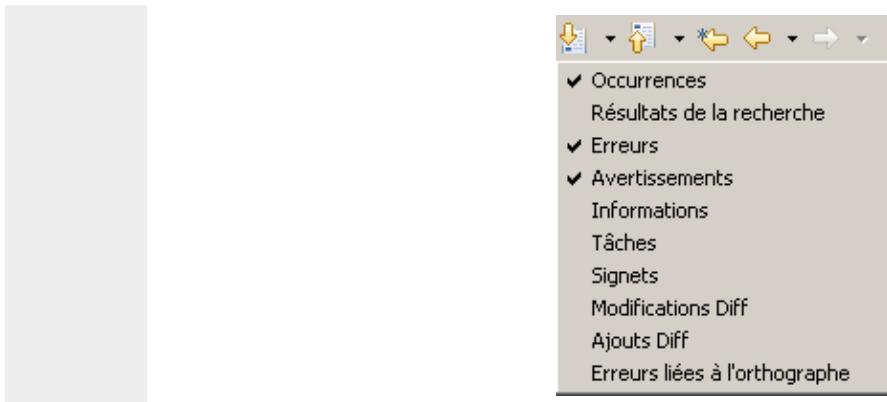
Suite à une recherche, il est possible de demander à l'éditeur de marquer les occurrences trouvées dans le code source en activant le bouton de la barre d'outils ou en utilisant la combinaison de touche Alt+Maj+O.

A screenshot of the Eclipse Java editor showing a piece of code with three occurrences of the string "bonjour". The word "bonjour" is highlighted in yellow, and each occurrence is preceded by a small yellow annotation icon. The code is:

```
public static void main(String[] args) {
    System.out.println("bonjour");
    System.out.println("bonjour");
    System.out.println("bonjour");
```

Les occurrences sont marquées sous la forme d'annotations.

Il est possible d'utiliser les boutons et pour se positionner sur l'occurrence suivante ou précédente. Il faut cependant que les annotations de type occurrences soient cochées dans le menu déroulant associé aux boutons.



### 7.6.22. Marquer les points de sortie d'une méthode



Même si cela n'est pas recommandé, il est possible de mettre plusieurs instruction return dans une méthode.

L'éditeur propose une fonctionnalité pour marquer sous forme d'annotation les points de sortie d'une méthode.

Pour cela, il faut positionner le curseur sur le type de retour de la méthode et activer les boutons



```
private int calculer(int a, int b) {
    if (a == 0) {
        return b;
    }
    if (b == 0) {
        return a;
    }
    return a + b;
}
```

### 7.6.23. Marquer les emplacements où une exception est levée



L'éditeur propose une fonctionnalité qui permet de marquer les méthodes qui lèvent une exception d'un type déclaré dans la signature de la méthode.

Pour cela, il faut placer le curseur sur une des exceptions utilisées dans la clause throws de la déclaration de la méthode et d'utiliser l'option "Rechercher / Occurrences d'exceptions" du menu principal.

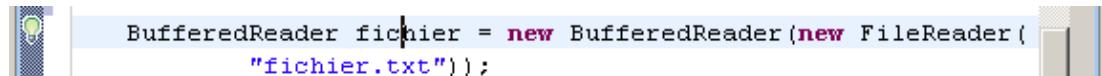
```
private void lecture() throws IOException {
    String ligne = "";
    BufferedReader fichier = new BufferedReader(new FileReader(
            "fichier.txt"));
    while ((ligne = fichier.readLine()) != null) {
        System.out.println(ligne);
    }
    fichier.close();
}
```

## 7.6.24. L'assistant de correction rapide

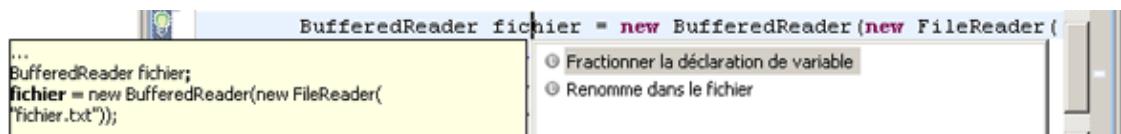


L'éditeur peut faciliter la rédaction du code en proposant un certains nombre d'action selon le contexte.

Pour savoir si une ou plusieurs correction rapide peuvent être proposée, il faut activer l'option dans les préférences, sous l'arborescence « Java / Editeur », sur l'onglet « Aspect », il faut cocher l'option « Activer l'ampoule pour l'assistant rapide » qui par défaut est décochée.



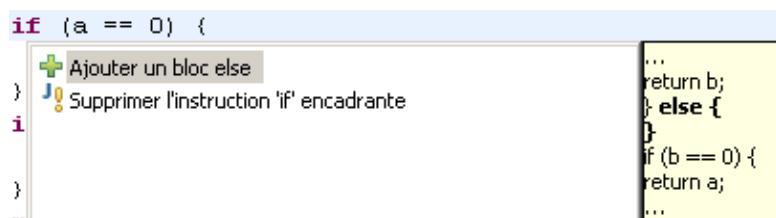
Il suffit alors de cliquer sur l'ampoule ou d'utiliser la combinaison de touche Ctrl+1 pour activer l'assistant.



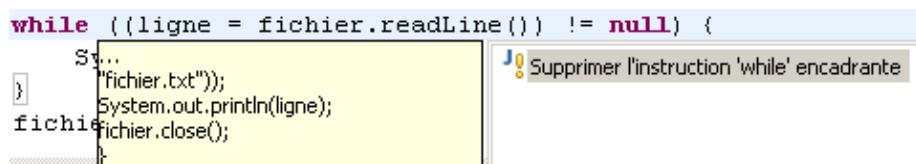
Les opérations proposées par l'assistant le sont fonction du contexte.

Les différentes opérations proposées sont nombreuses et facilitent souvent la réalisation de petites tâches fastidieuses à réaliser manuellement, par exemple :

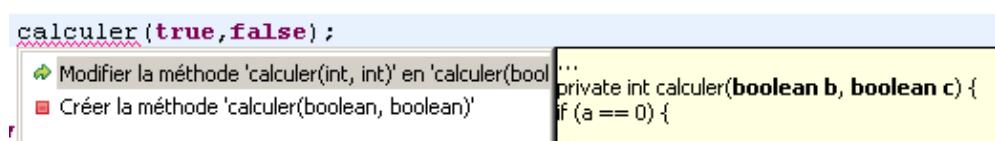
- ajouter un bloc else à une instruction if



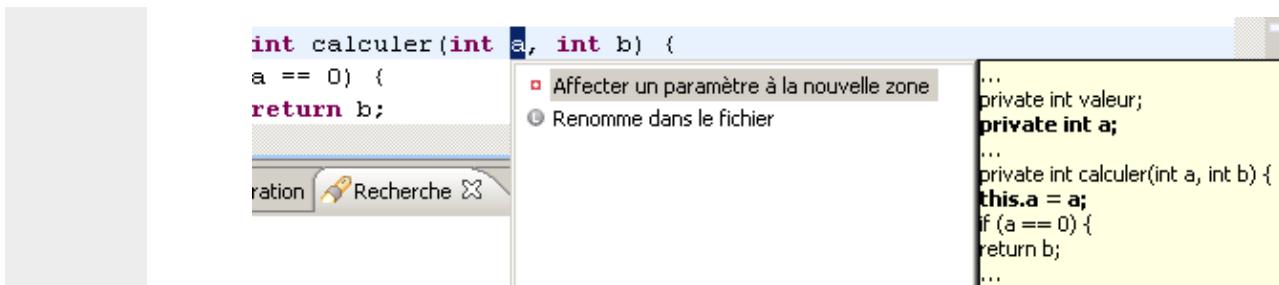
- supprimer une instruction englobante



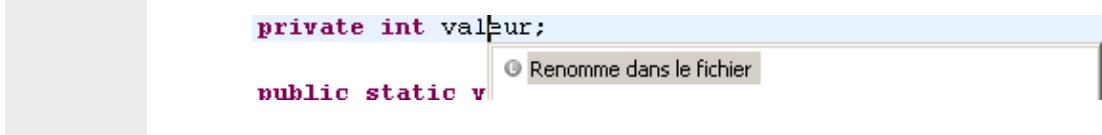
- modifier l'appel ou la méthode en cas de non concordance de paramètres



- créer un nouveau champ qui sera initialisé avec un paramètre de la méthode



- renommer localement un membre



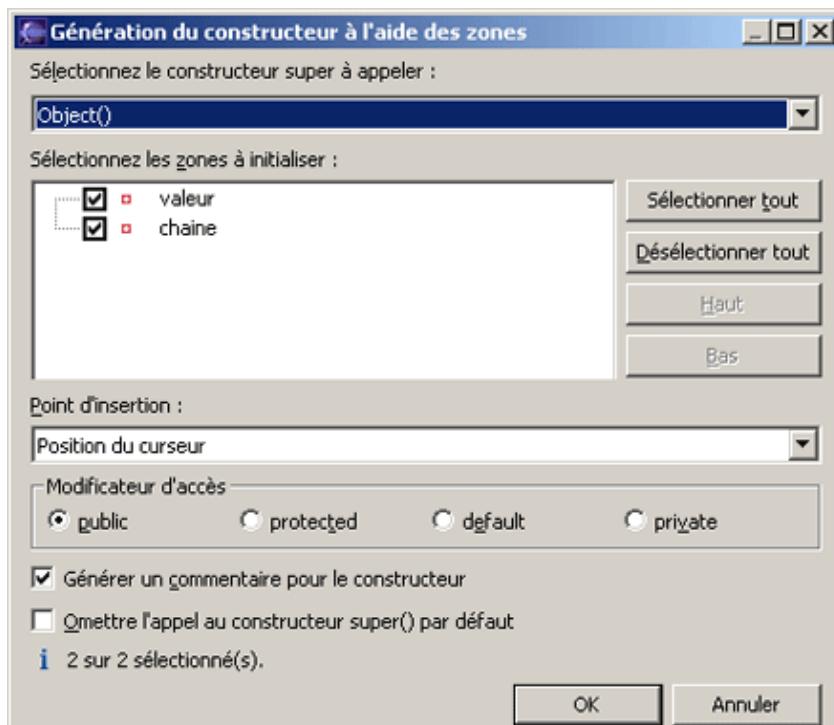
Un quick fixe permet d'utiliser les generics à une collection utilisée avec le type brut.



### 7.6.25. La génération de constructeur



L'option « Source / générer le constructeur à partir des zones ... » du menu contextuel permet de faciliter la création d'un constructeur.

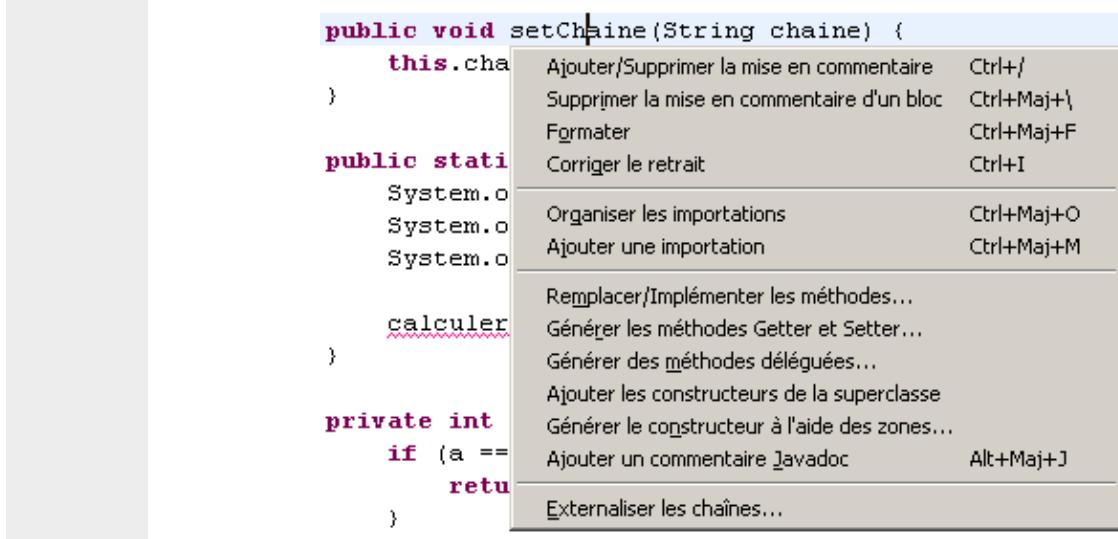


Il est alors possible de sélectionner les paramètres du nouveau constructeur et les différentes options pour que le code du constructeur soit généré.

## 7.6.26. Raccourci clavier pour avoir accès aux fonctions de modification du code source



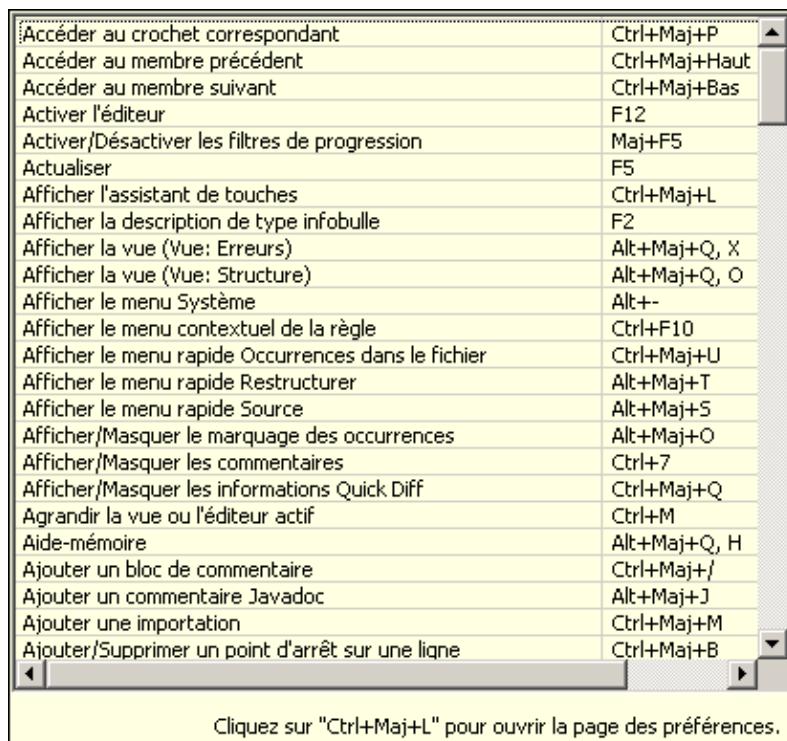
L'éditeur propose un menu contextuel pour accéder rapidement aux fonctions de modification du code source en utilisant la combinaison de touche Alt+Maj+S



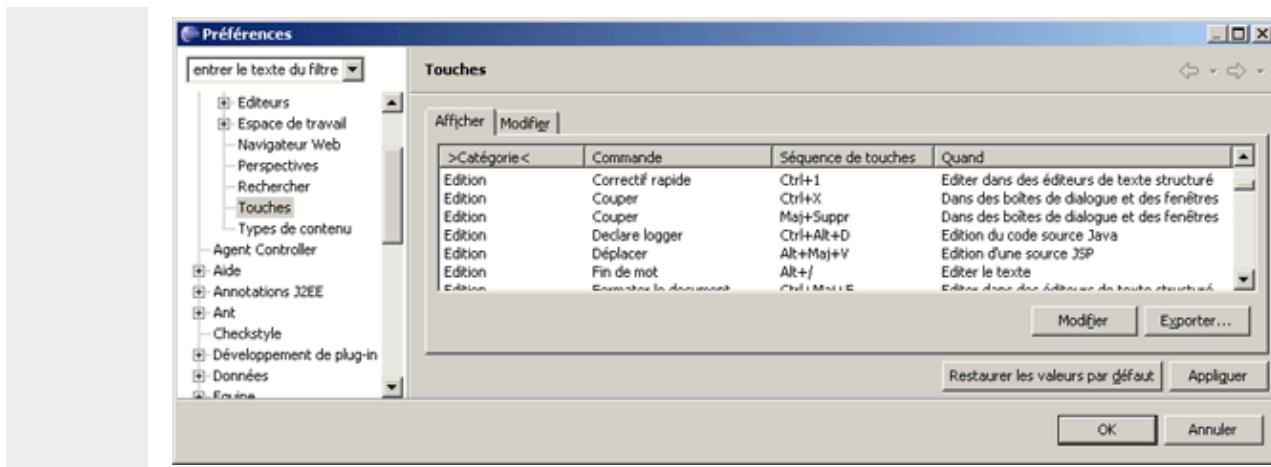
## 7.6.27. Les raccourcis clavier des éditeurs



Dans un éditeur, il suffit d'utiliser la combinaison de touches Ctrl+Shift+L pour obtenir une liste des raccourcis clavier utilisable dans l'éditeur.

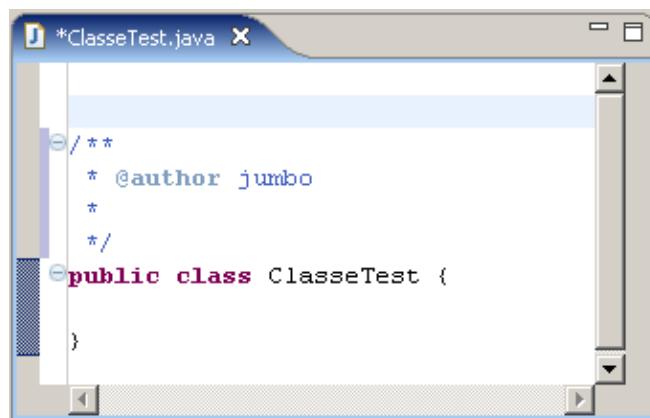


Il suffit de réutiliser la combinaison de touche « Ctrl+Maj+L » pour afficher les préférences concernant les raccourcis clavier des éditeurs.



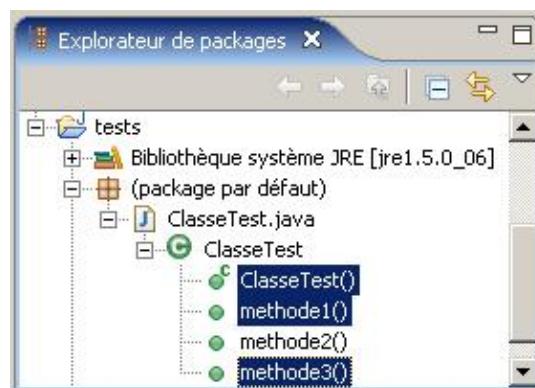
## 7.6.28. La gestion des modèles de commentaires

Dans l'éditeur de code source Java, l'option « Sources / Ajouter commentaires » (Alt + Shift + J) du menu contextuel permet d'insérer un commentaire selon un modèle prédéfini.



Ce modèle peut être paramétré dans les préférences.

Il est possible de demander la génération de commentaires sur plusieurs entités en même temps en sélectionnant dans « l'explorateur de packages » et en utilisant l'option « Source / Ajouter commentaires ».



```
/** * * */ public ClasseTest() { }

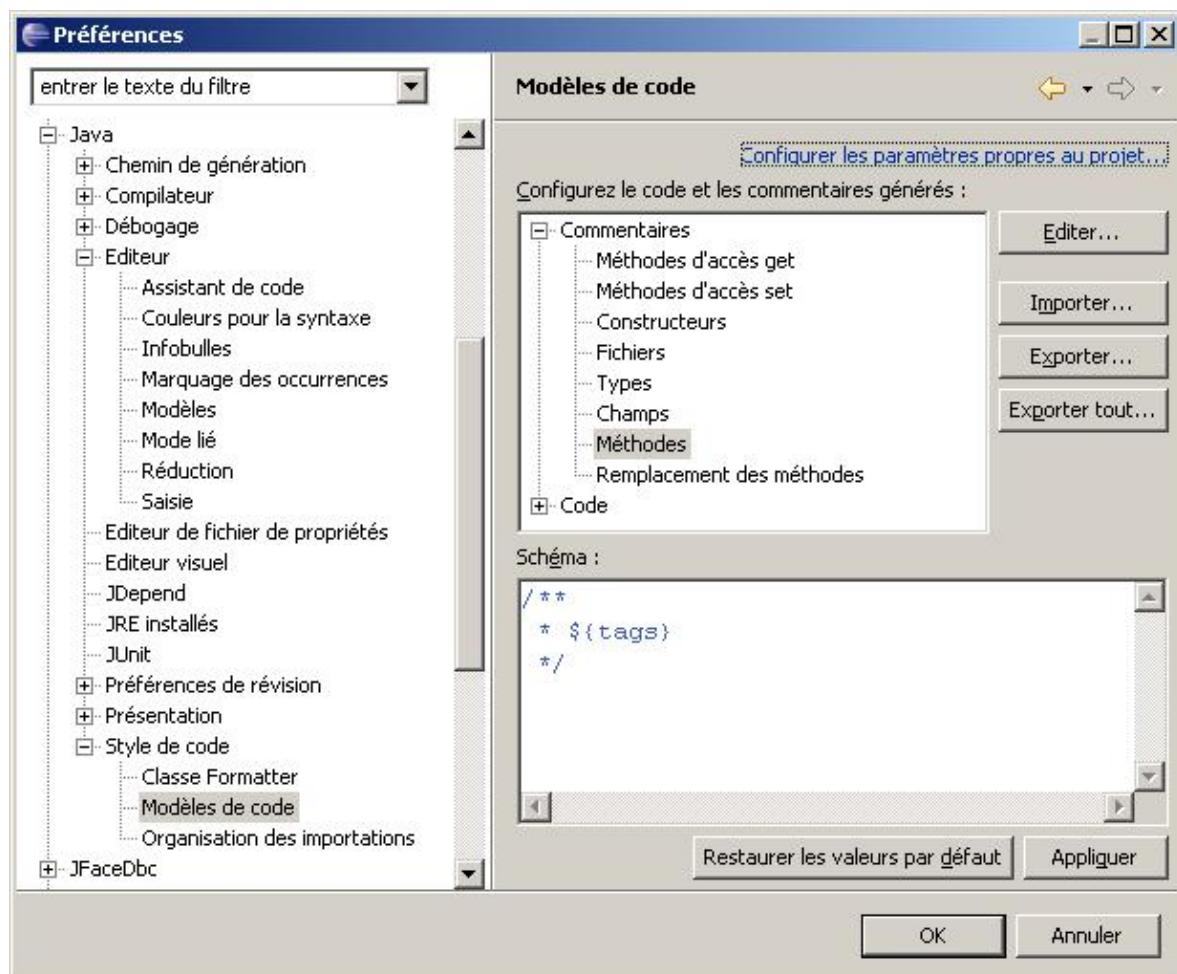
/** * * */ public void methode1() { }

public void methode2() { }

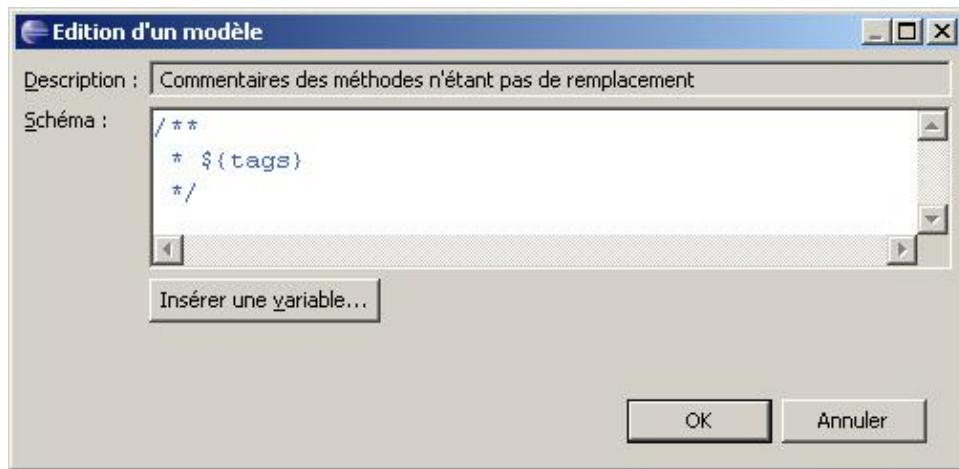
/** * * */ public void methode3() { }

}
```

Les modèles de commentaires sont paramétrables dans les Préférences :



Pour modifier un modèle, il faut le sélectionner dans la liste et cliquer sur le bouton « Editer ... »



Pour insérer une variable, il suffit de cliquer sur le bouton « Insérer une variable ... » ou utiliser la combinaison de touche Ctrl+Espace.



Le contenu de la plupart des variables sera déterminé dynamiquement lors de la génération du commentaire.

Une fois la saisie terminée, il faut cliquer sur le bouton « OK » pour valider les modifications.

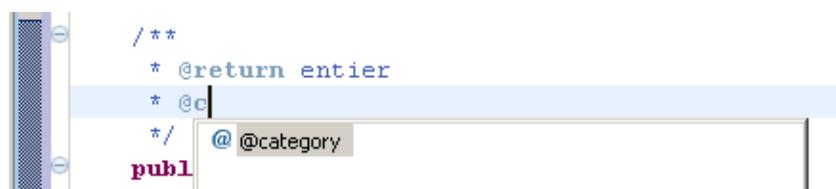
Le modèle sélectionné ou tous les modèles peuvent être exporté et importé grâce aux boutons dédiés.

### 7.6.29. Le support du tag @category



Le tag Javadoc `@category` est supporté et intégré à Eclipse : ce tag permet de regrouper des membres par catégories.

L'assistant de complétion de code propose ce tag.



#### Exemple :

```
/**  
 * @return entier  
 * @category getter
```

```

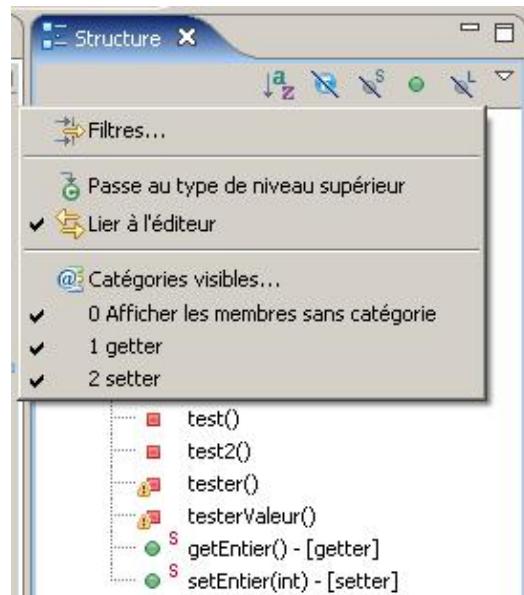
    */
public static int getEntier() {
    return entier;
}

/**
 * @param entier entier à définir
 * @category setter
 */
public static void setEntier(int entier) {
    Test.entier = entier;
}

```

Ce tag, une fois utilisé, est exploité dans plusieurs vues d'Eclipse.

La vue structure affiche la catégorie d'un membre entre crochet. Elle permet d'appliquer un filtre d'affichage sur les membres à partir de leur catégorie en cliquant sur le bouton 

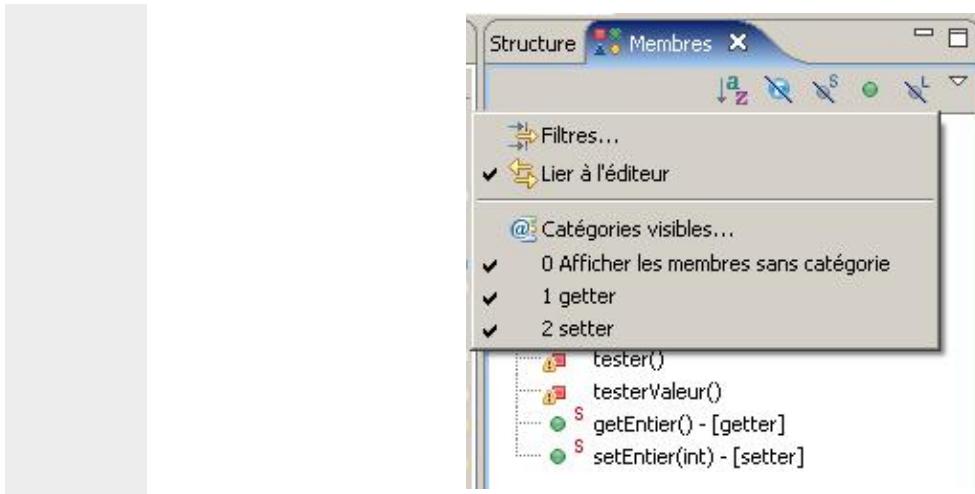


Il est possible de cocher les catégories dont les membres seront affichés. L'option « Catégories visibles » permet de réaliser la même opération mais dans une boîte de dialogue. Celle-ci est particulièrement utile si le nombre de catégories est important.



Cochez les catégories qui doivent être affichées dans la vue et cliquez sur le bouton « OK ».

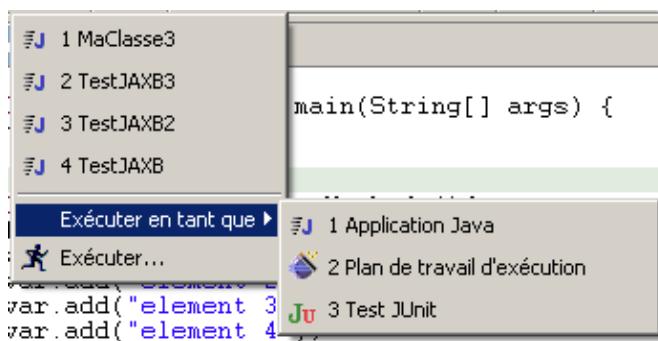
La vue « Membres » offre les mêmes fonctionnalités



## 7.7. Exécution d'une application

Dans la perspective "Java", il est possible d'exécuter une application de plusieurs façons.

Pour exécuter l'application en cours d'édition, il suffit de cliquer sur la flèche du bouton de la barre d'outils.

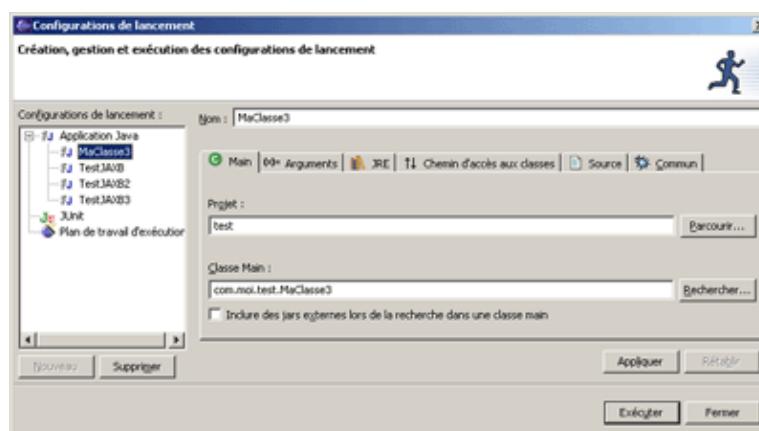


Ce menu déroulant propose différentes options :

- relancer les précédentes exécutions listées dans la première partie du menu
- l'option "Exécuter en tant que" permet de lancer l'application dans trois modes différents (Application java, Test JUnit et plan de travail d'exécution)
- l'option "Exécuter" ouvre une boîte de dialogue pour paramétriser précisément les options d'exécution

L'option "Exécuter en tant que / Application Java" lance la méthode main() d'une application.

L'option "Exécuter" ouvre une boîte de dialogue qui permet de saisir tous les paramètres d'exécution.



La boite de dialogue se compose de six onglets.

L'onglet "Main" permet de sélectionner le projet et la classe de ce projet qui contient la méthode main() à exécuter.

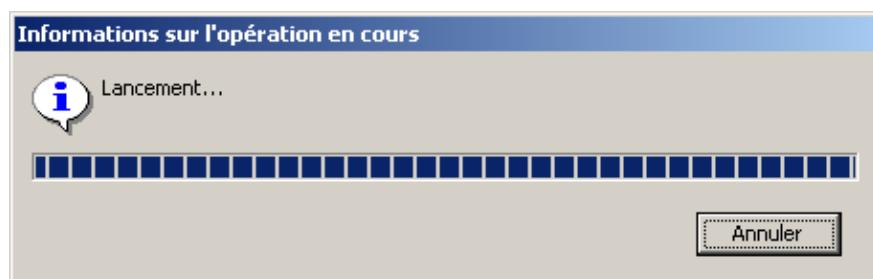
L'onglet "Arguments" permet de préciser les arguments passés à l'application et à la machine virtuelle.

L'onglet "JRE" permet de sélectionner le JRE à utiliser lors de l'exécution.

L'onglet "Chemin d'accès aux classes" permet de modifier la liste et l'ordre des bibliothèques utilisées lors de l'exécution. Cet onglet permet de modifier la liste définie dans le projet qui est celle utilisée par défaut.

L'onglet "Commun" permet de préciser le type de lancement et le mode d'exécution et de débogage.

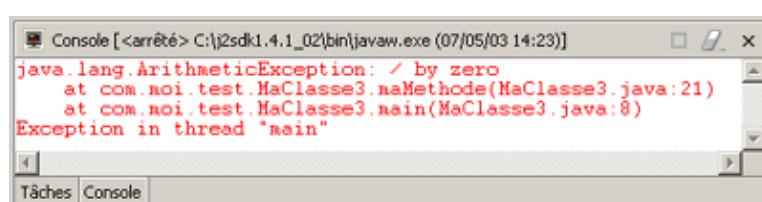
Une fois tous les paramètres voulus renseignés, il suffit de cliquer sur le bouton " Exécuter " pour lancer l'application.



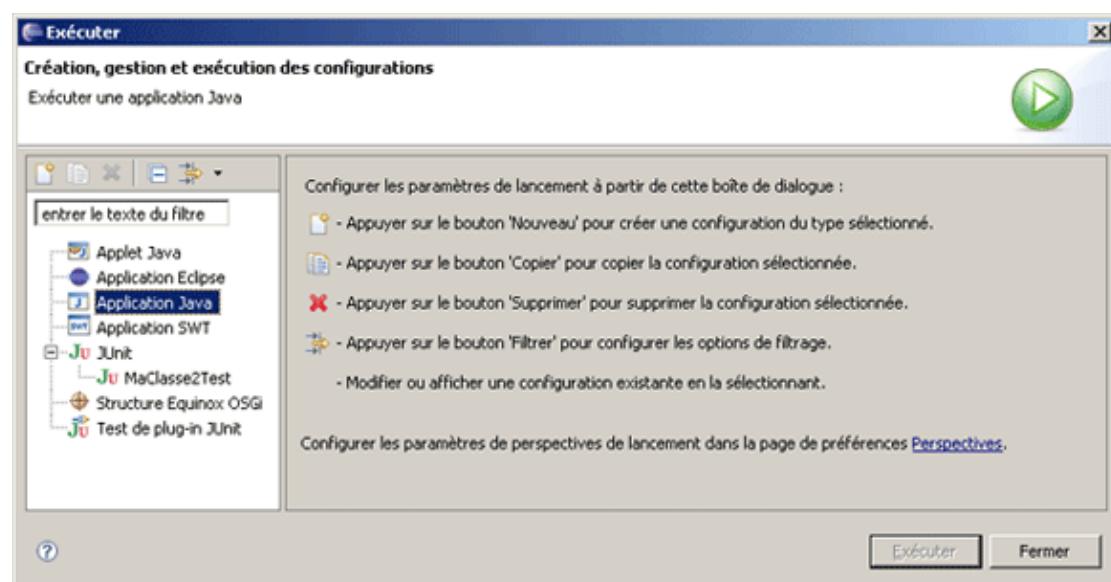
La vue " Console " permet de voir les données qui sont envoyées dans le flux standard de sortie et d'erreurs.



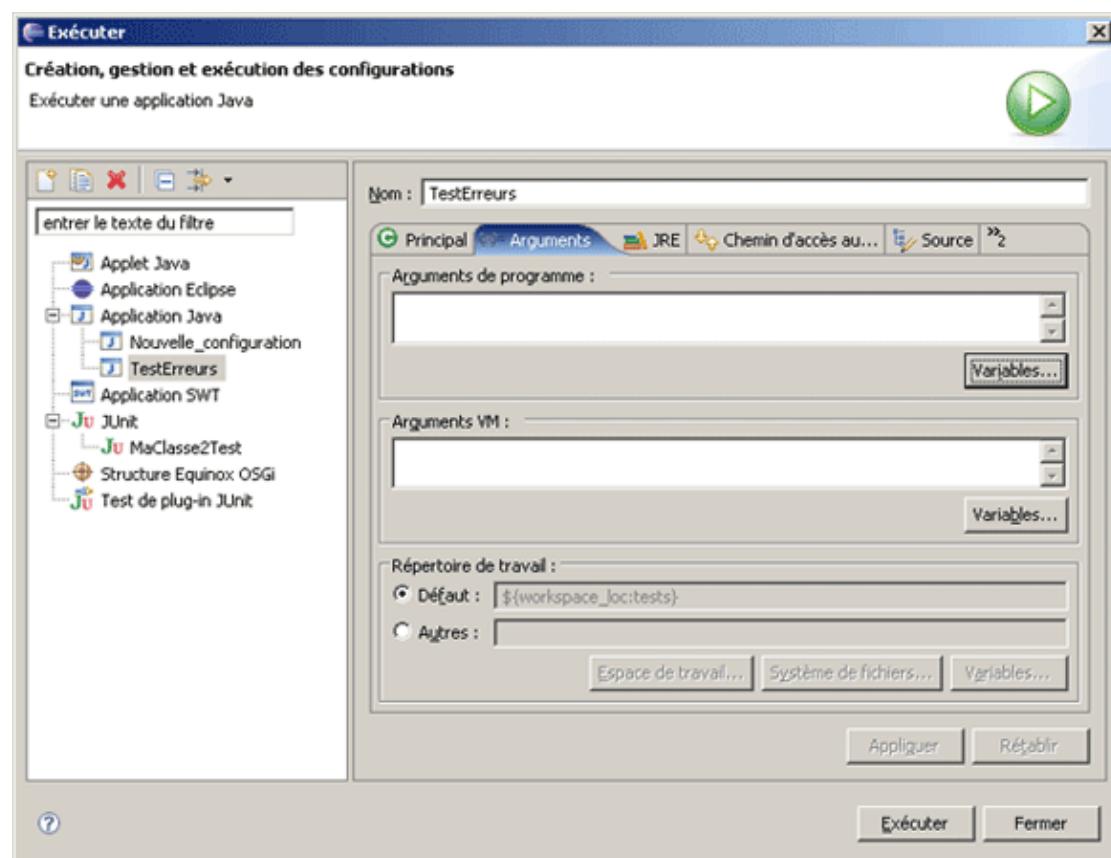
Les messages ayant pour origine une exception sont aussi envoyés dans cette vue.



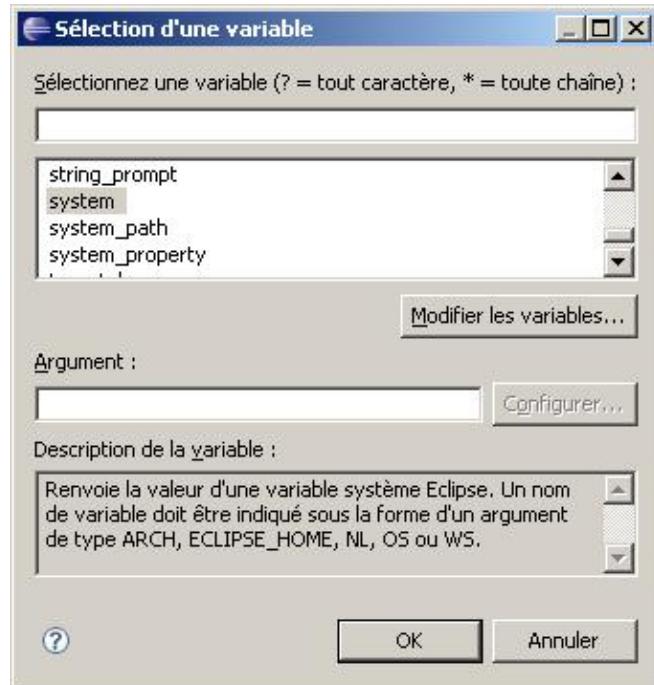
L'ergonomie de la boite de dialogue « Exécuter » a été légèrement revue.



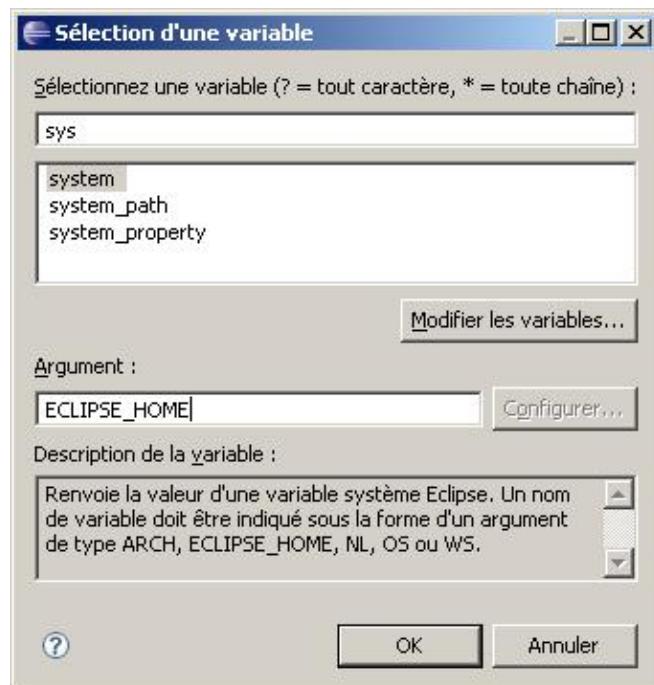
La gestion des configurations se fait via de petit bouton avec des icônes.



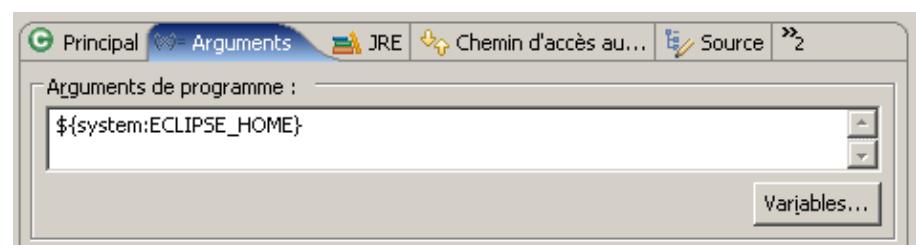
Il est possible de préciser dans les variables une variable d'Eclipse. Il est possible de la saisir directement ou d'utiliser l'assistant de sélection de variable en cliquant que le bouton « Variable ».



Il suffit de sélectionner la variable en s'aideant éventuellement du filtre et de saisir l'argument



Cliquez sur le bouton « OK »



Exemple : source de la classe de test

```
package fr.jmdoudoux.testb;
```

```

public class TestErreurs {

    public static void TestA(String valeur) {
        System.out.println("valeur="+valeur);
    }

    public static void main(String[] argv) {
        TestA(argv[0]);
    }
}

```

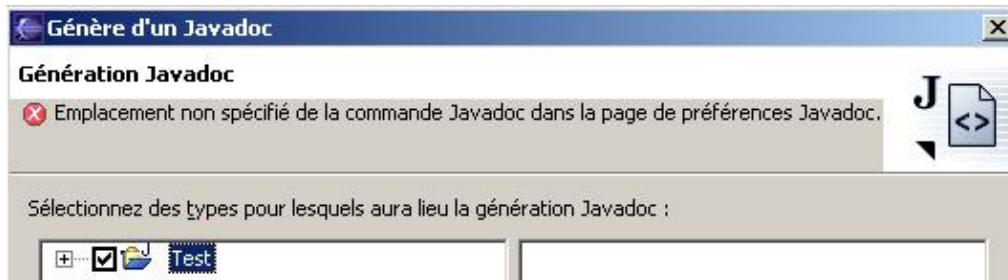
Résultat de l'exécution :

valeur= C:\java\eclipse32

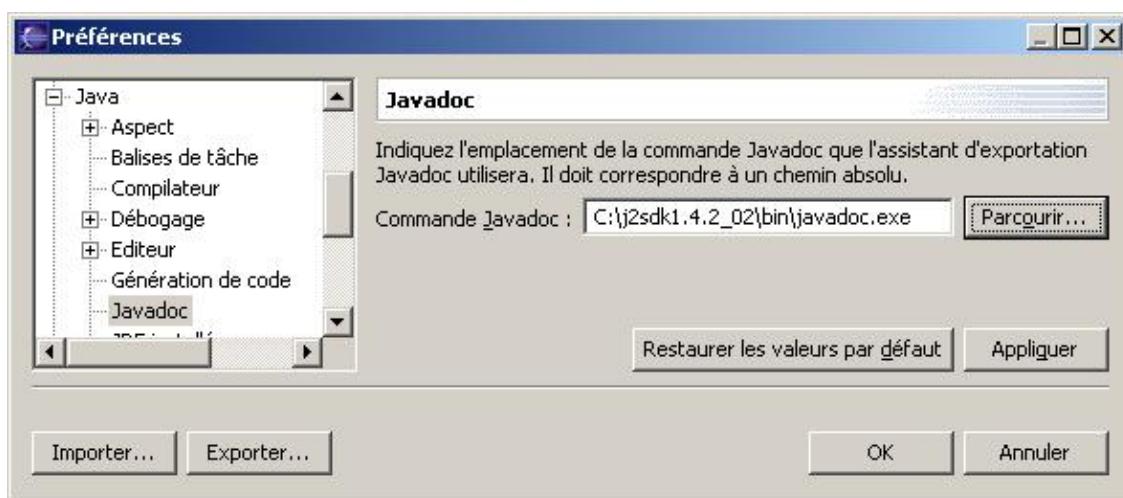
## 7.8. Génération de la documentation Javadoc

Pour demander la génération de la documentation Javadoc, il faut utiliser le menu "Projet/Générer/Les javadoc".

Pour utiliser cet option, il faut obligatoirement que les préférences concernant Javadoc soit renseignées, sinon un message d'erreur empêche l'utilisation de l'assistant

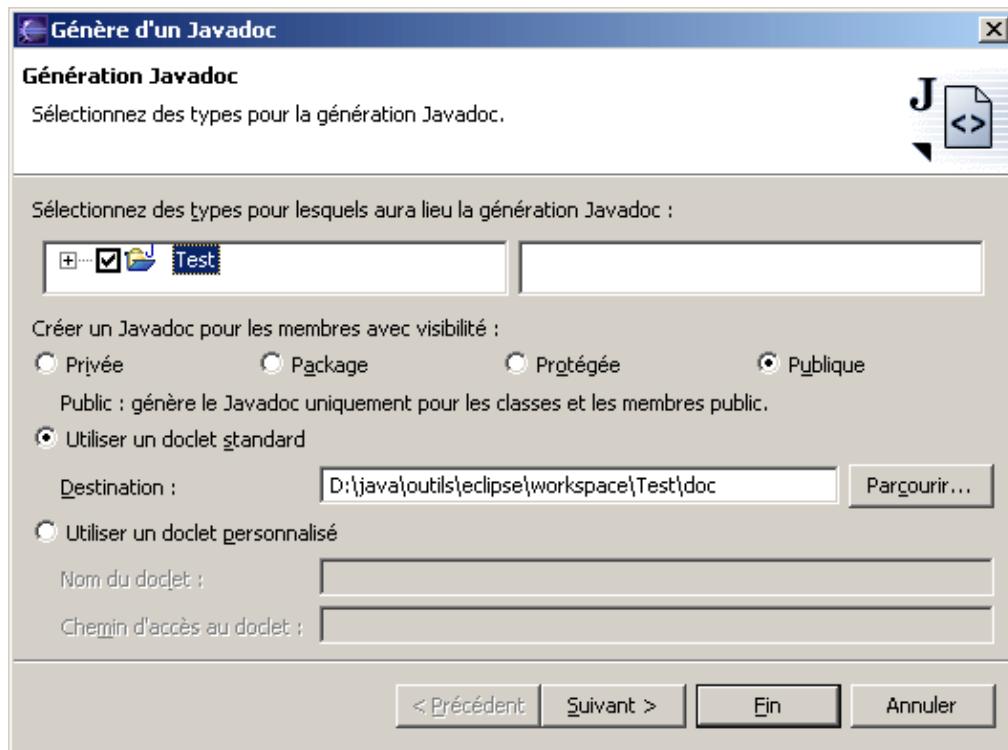


Pour résoudre le problème, il faut aller dans l'arborescence "Java/Javadoc" des préférences, cliquer sur le bouton "Parcourir" et sélectionner le fichier javadoc.exe du JDK à utiliser.



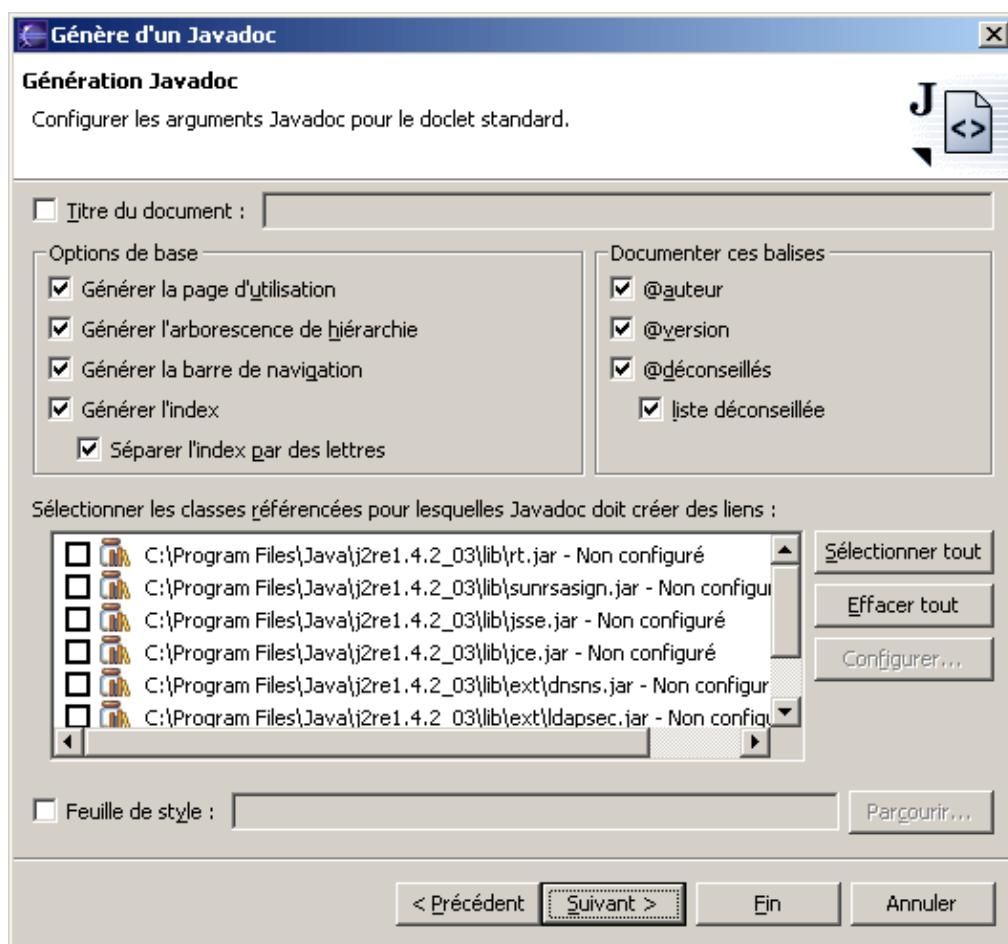
Cliquez sur le bouton "OK" pour valider les modifications.

La génération de la documentation au format Javadoc se fait avec un assistant. Il faut lui indiquer : le ou les projets concernés, la visibilité des membres à inclure, le doclet à utiliser et le répertoire de destination.



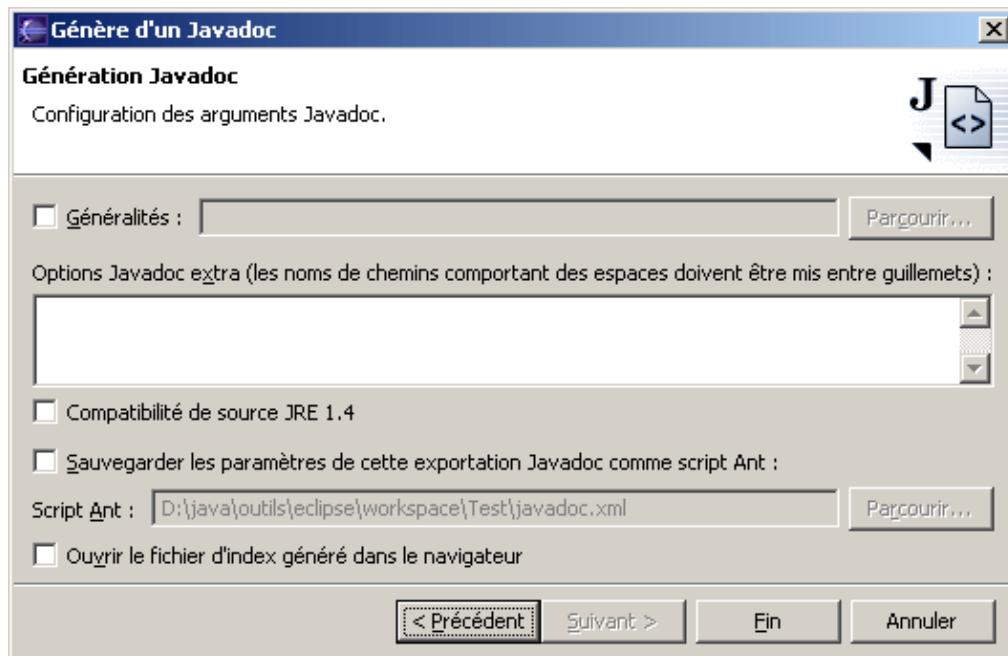
Cliquez sur le bouton "Suivant"

La page suivante de l'assistant permet de préciser des options pour l'outil Javadoc.



Une fois les options configurées, cliquez sur le bouton "Suivant".

La page suivante de l'assistant permet de préciser d'autres options.

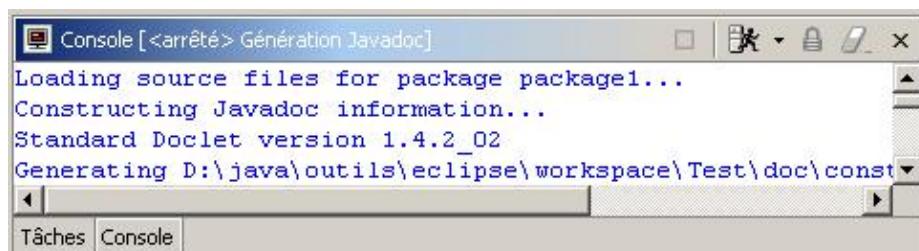


En cliquant sur "Fin", la génération de la documentation est effectuée. L'assistant demande si l'emplacement javadoc du projet doit être mis à jour.

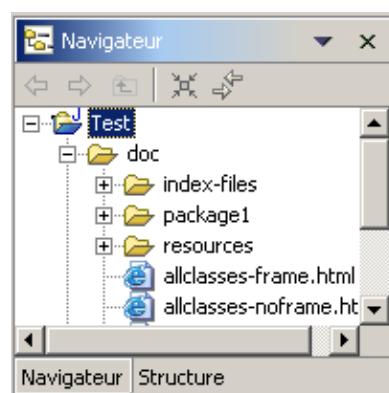


Il est conseillé de répondre "Oui" pour permettre d'avoir accès à cette documentation à partir de l'éditeur en appuyant sur les touches "Shift" + "F2".

Le détail de la génération est affiché dans la vue "Console".

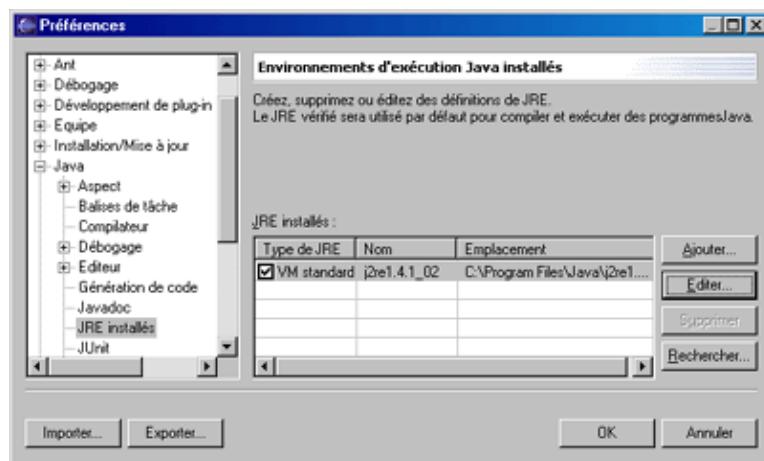


Les fichiers apparaissent dans la vue "Navigateur".

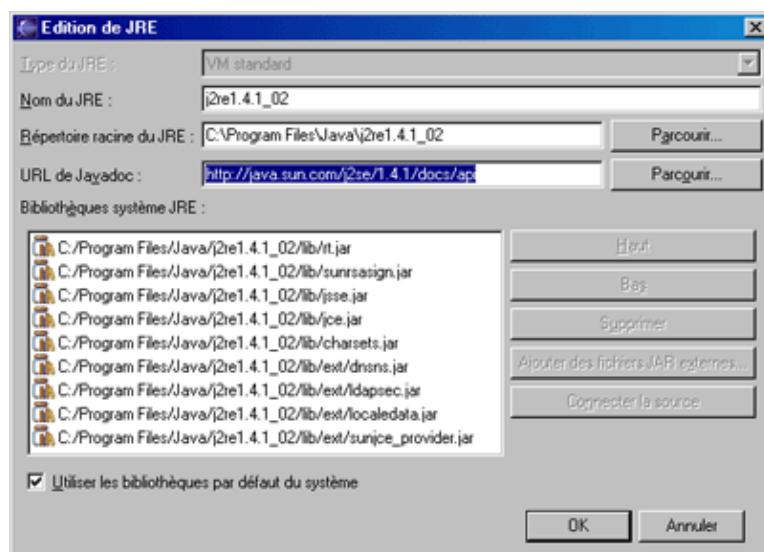


## 7.9. Définition du JRE à utiliser

Eclipse est capable de travailler avec plusieurs JRE. Dans l'arborescence " Java/JRE installé " des préférences, il est possible de définir plusieurs JRE installés sur la machine.



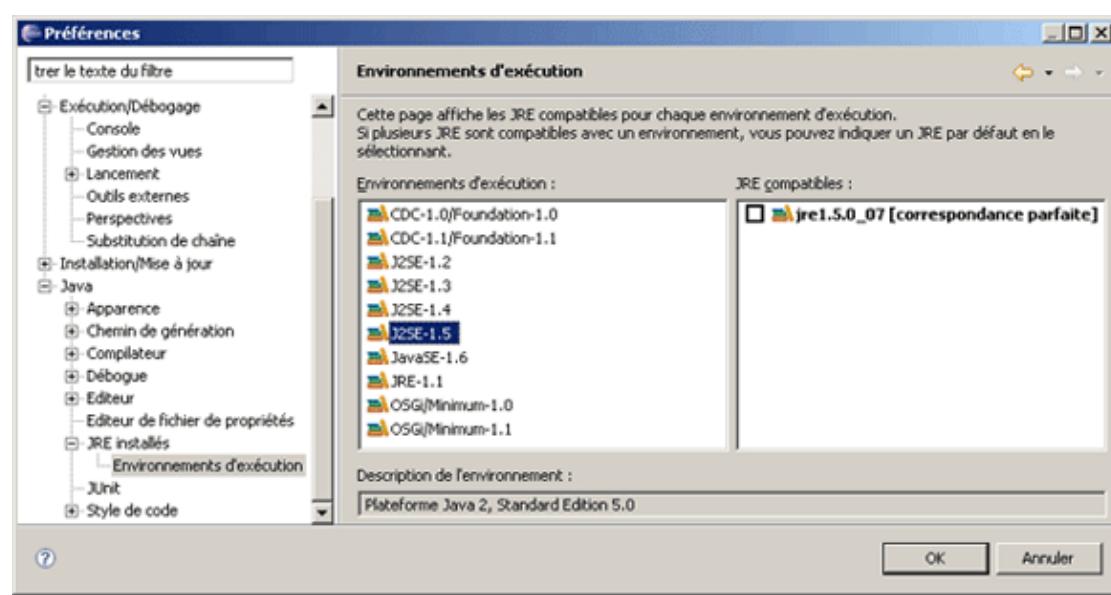
Un clic sur le bouton "Editer" permet de modifier les données du JRE défini dans Eclipse.



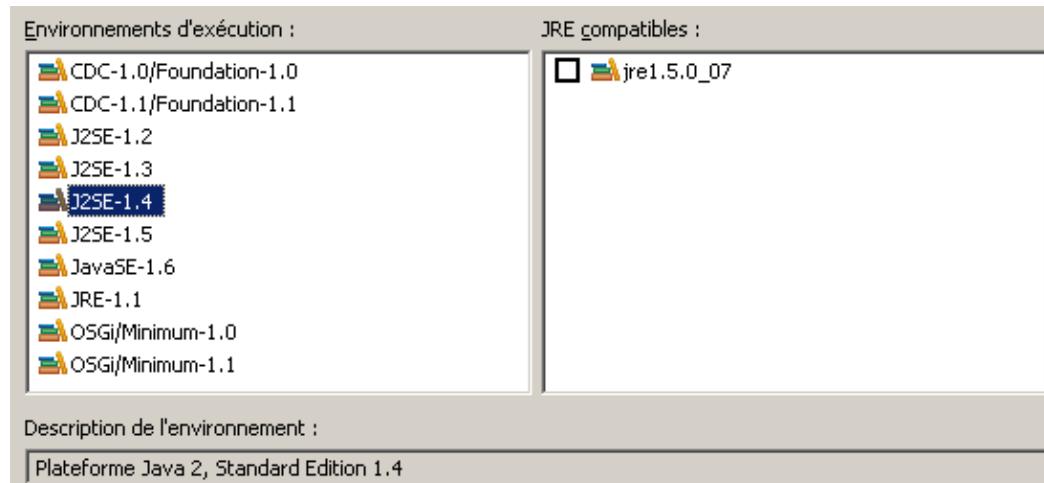
## 7.10. Les environnements d'exécution



Dans les préférences, il est possible de définir pour un environnement d'exécution le JRE compatibles à utiliser. Pour chaque environnement, la liste des JRE compatibles est affichée.



Les JRE affichés en gras avec la mention « [correspondance parfaite] » indique une parfaite adéquation avec l'environnement d'exécution.



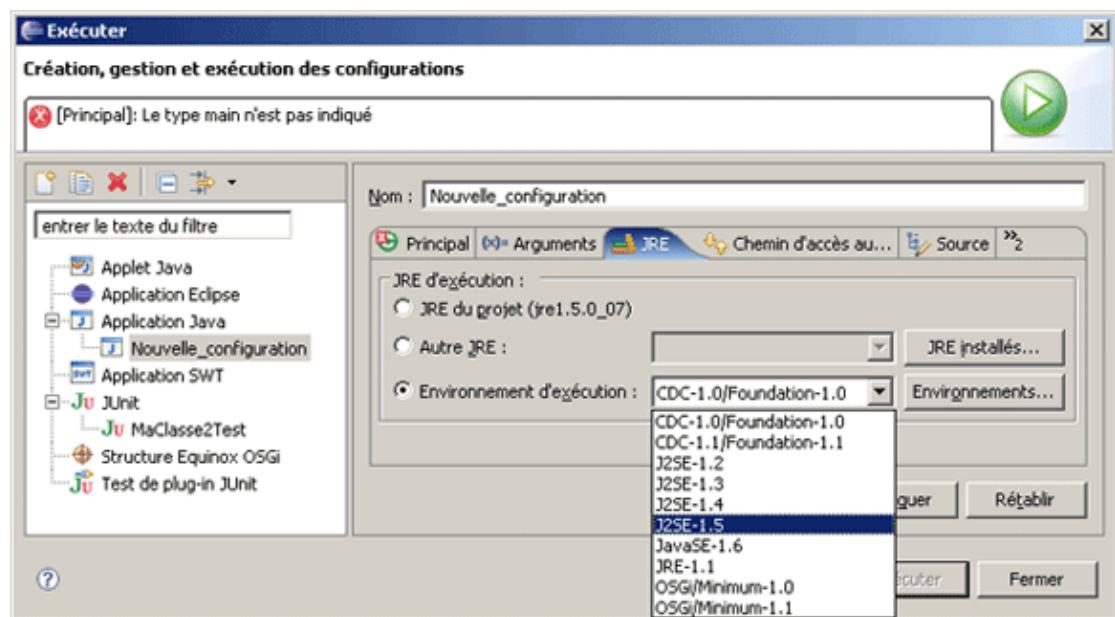
Il est aussi possible qu'aucun JRE compatible ne corresponde à l'environnement d'exécution.



Pour définir le JRE par défaut de l'environnement d'exécution, il suffit de le cocher.

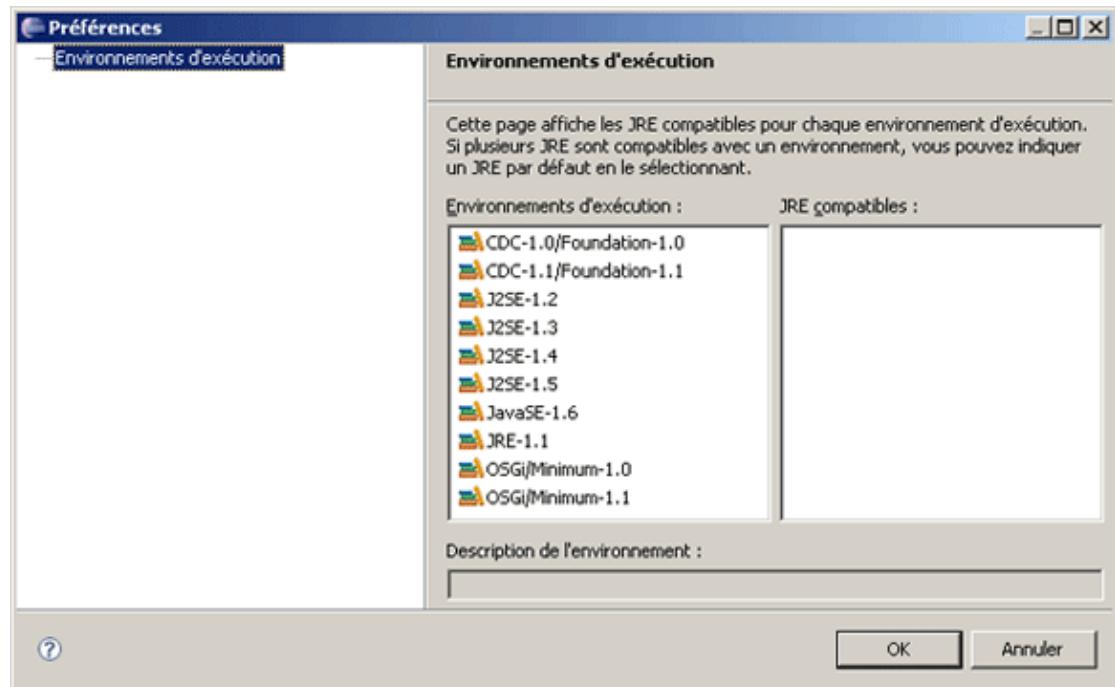
L'utilisation d'un environnement d'exécution est plus souple que d'utiliser directement un JRE.

Il est ainsi possible de préciser un environnement d'exécution dans une configuration d'exécution



Il suffit dans l'onglet « JRE » de cocher le choix « Environnement d'exécution » et de sélectionner dans la liste l'environnement souhaité.

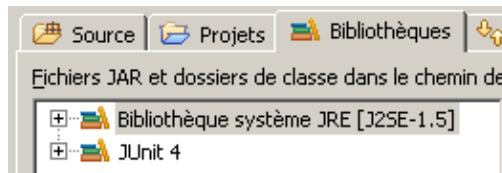
Le bouton « Environnements ... » ouvrir directement les préférences des environnements d'exécution.



Les environnements d'exécution sont aussi utilisables dans les options de « chemin de génération Java » du projet. Dans les propriétés du projet, dans l'onglet « Bibliothèque » de l'option « Chemin de génération Java », sélectionnez « Bibliothèque système JRE » et cliquez sur le bouton « Editer ...».



La boîte de dialogue permet de sélectionner un environnement d'exécution.

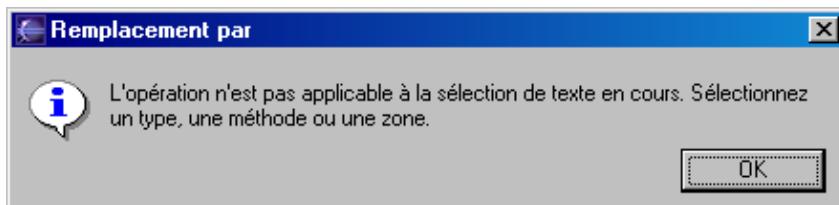


L'environnement d'exécution est précisé entre les crochets à la place du JRE.

## 7.11. Utilisation de l'historique local

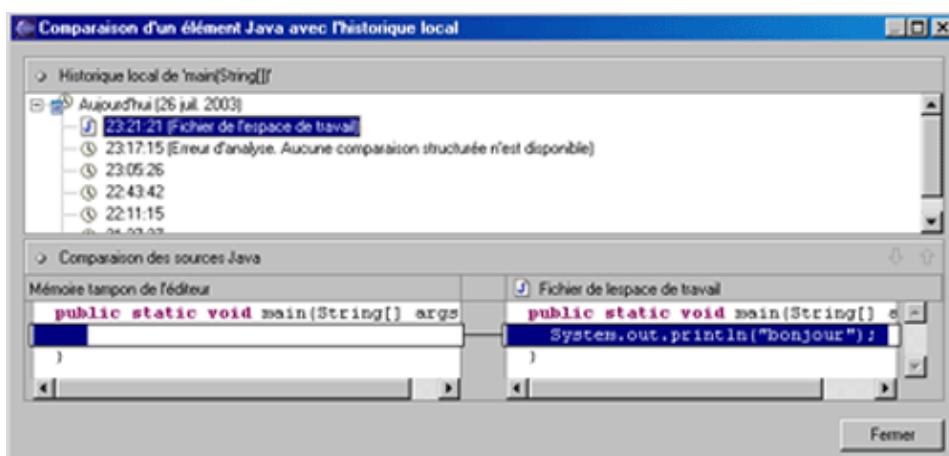
L'historique local est une fonctionnalité proposée par Eclipse pour conserver un certain nombre de versions du code pour chaque élément contenu dans l'espace de travail.

Pour pouvoir utiliser l'historique local, il faut placer le curseur sur un élément de code sinon un message d'erreur est affiché :



L'option "Historique local" du menu contextuel propose 4 options :

- " Comparer à ... "

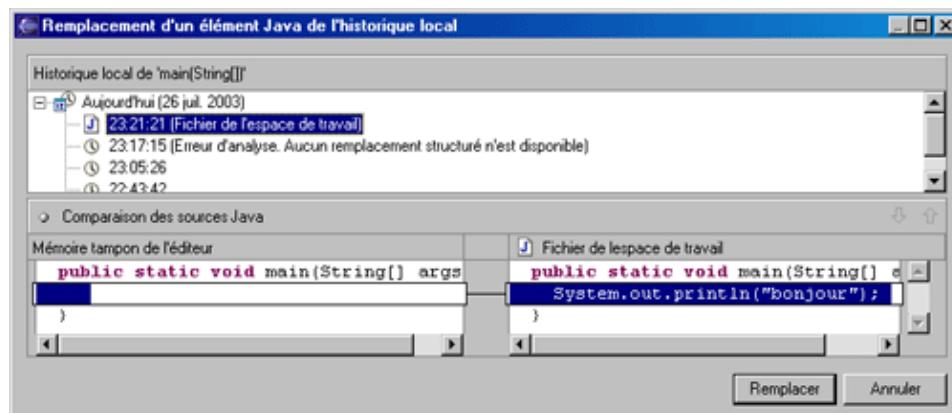


Cette option permet de comparer la version contenue dans l'éditeur avec celles contenues dans l'historique. Il n'est pas possible de reporter les modifications avec cette option.

- "Remplacer par l'élément précédent"

Cette option permet de remplacer la version de l'éditeur par la dernière contenue dans l'historique : elle correspond à la dernière sauvegarde.

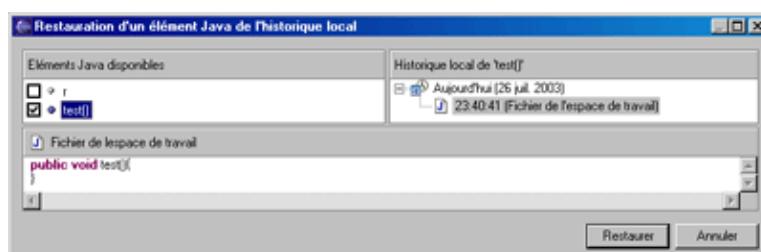
- "Remplacer par ... "



Il suffit de sélectionner la version désirée et de cliquer sur le bouton "Remplacer".

- "Restaurer à partir de ... "

Cette option permet de restaurer des éléments contenus dans l'historique mais qui ne sont plus présents dans l'éditeur de code.



Il suffit de cocher le ou les éléments et de sélectionner la version de l'historique à restaurer et de cliquer sur le bouton "Restaurer".

## 7.12. Externaliser les chaînes

Eclipse possède une fonction permettant d'automatiser l'externalisation des chaînes de caractères dans un fichier de ressource afin de faciliter l'internationalisation de la classe traitée.

L'exemple de cette section utilise le code source suivant :

```

J MaClasse50.java X
package com.moi.test;

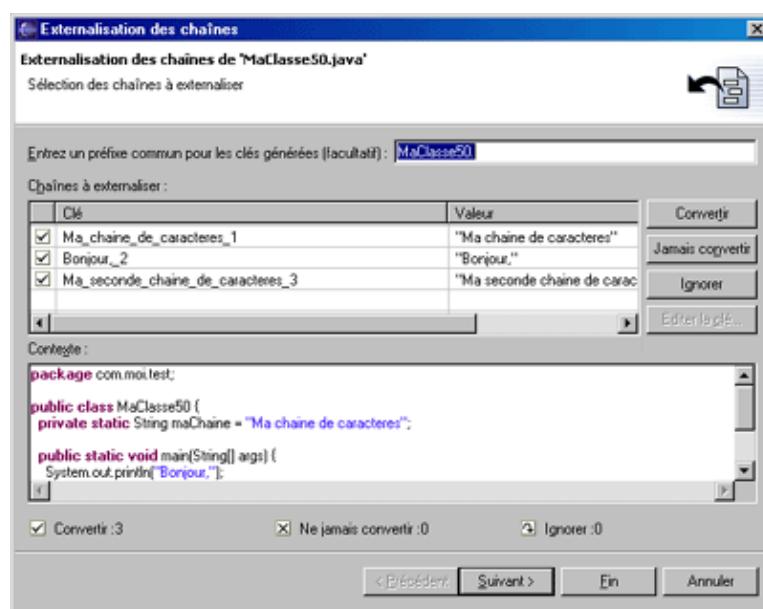
public class MaClasse50 {
    private static String maChaine = "Ma chaine de caracteres";

    public static void main(String[] args) {
        System.out.println("Bonjour,");
        System.out.println(maChaine);
        System.out.println("Ma seconde chaine de caracteres");
    }
}

```

Dans la vue "Package", il faut sélectionner le fichier source puis utiliser l'option " Source / Externaliser les chaînes " du menu contextuel.

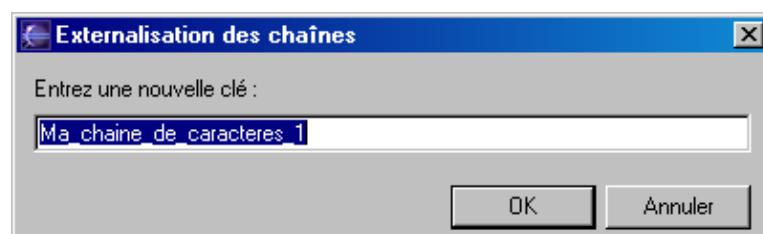
Eclipse analyse le code source à la recherche de chaînes de caractères et affiche l'assistant " Externalisation des chaînes ".



La première page de l'assistant permet de sélectionner les chaînes de caractères à traiter détectées par Eclipse.

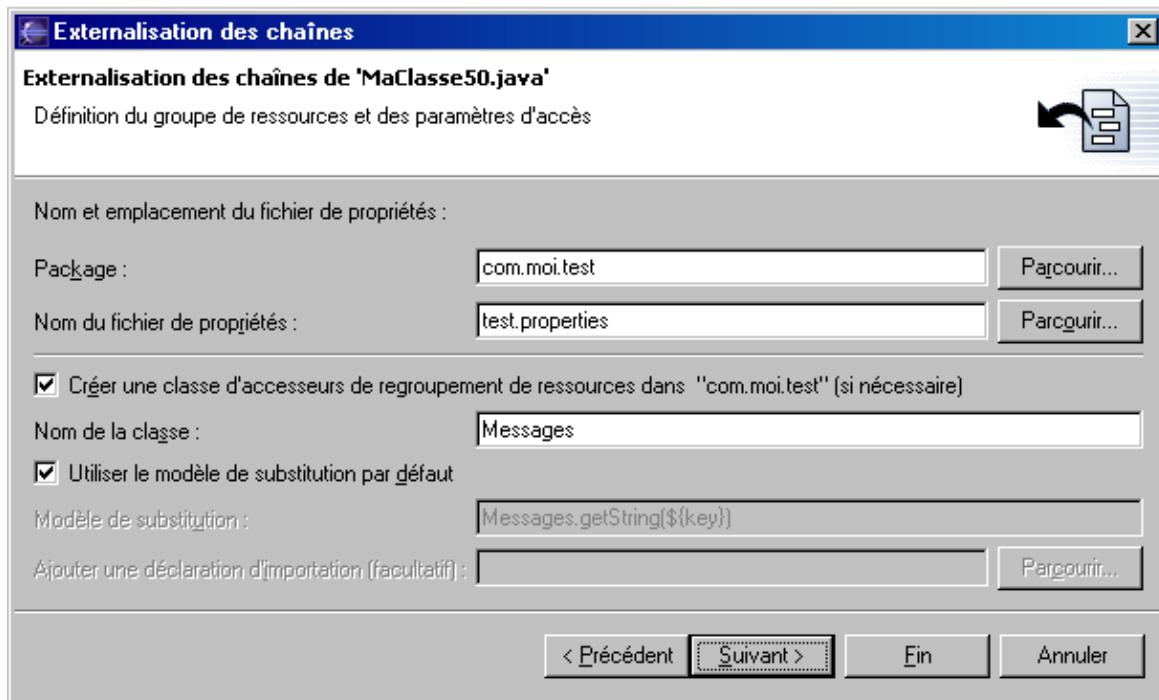
Pour chaque chaîne, il est possible de changer le nom de la clé associée à la chaîne de caractères et de préciser si la chaîne doit être traitée ou non.

Pour modifier la clé, il est possible de cliquer sur la clé et de saisir le nouveau nom ou de sélectionner la ligne de la chaîne et de cliquer sur le bouton " Editer la clé "

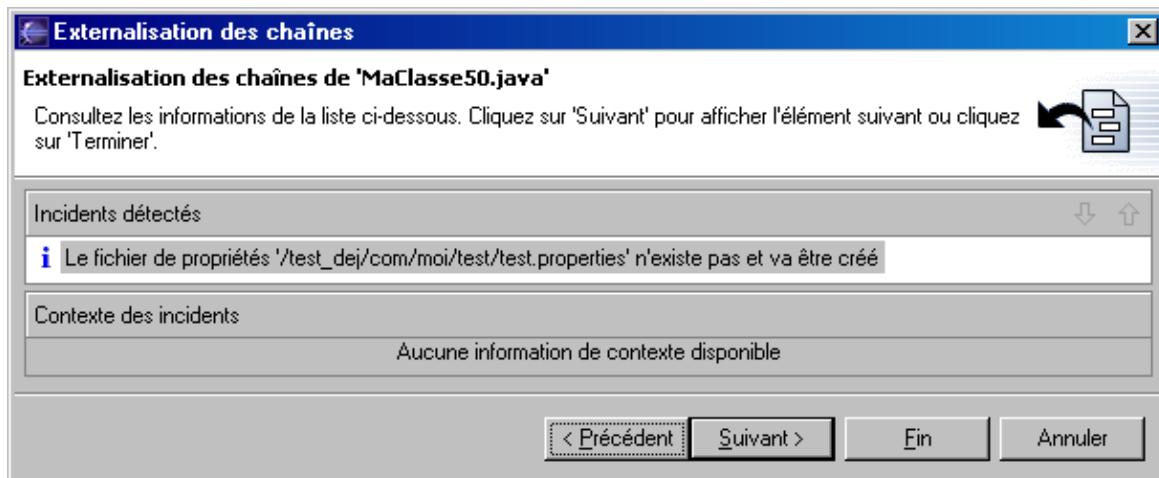


Pour indiquer si la chaîne doit être traitée, il est possible de cliquer plusieurs fois sur la case à cocher de la ligne correspondante pour obtenir le symbole correspondant à la valeur voulue ou de cliquer sur le bouton " Convertir ", " Jamais convertir " ou " Ignorer " après avoir sélectionné la ligne désirée.

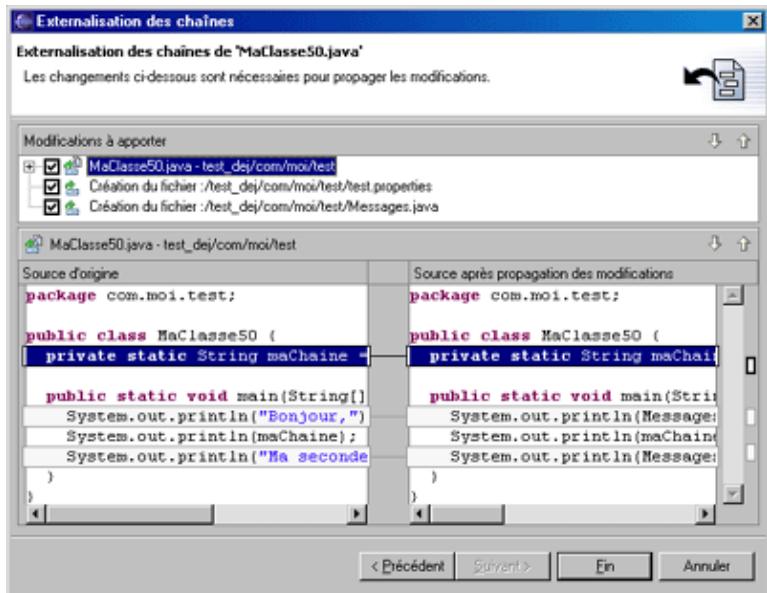
Un clic sur le bouton " Suivant " permet de préciser des informations sur le fichier de ressource qui sera généré.



Un clic sur le bouton " Suivant " affiche une liste des problèmes détectés.



Un clic sur le bouton " Suivant " permet d'afficher une page qui prévisualise les modifications qui vont être apportées.



Il est possible de sélectionner tout ou partie des modifications en les cochant.

Un clic sur le bouton "Fin" met en oeuvre les modifications.

### 7.12.1. Rechercher les chaînes externalisées erronées



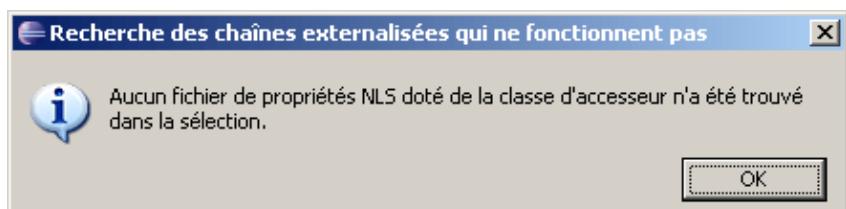
Cette fonctionnalité permet de rechercher des clés dans un fichier de ressource qui ne soient pas utilisées ou qui sont en double.

Exemple :

```
public static void main(String[] args) {
    // TODO Raccord de méthode auto-généré
    System.out.println(Messages.getString("chaine0")); // $NON-NLS-1$
    System.out.println("chaine 2"); // $NON-NLS-1$
    System.out.println(Messages.getString("chaine2")); // $NON-NLS-1$
    System.out.println("chaine 4");
}
```

Pour utiliser cette fonctionnalité, il faut sélectionner préalablement un fichier de ressource, un package ou un projet et utiliser l'option « Source / rechercher les chaînes externalisées qui ne fonctionnent pas ».

Il est nécessaire qu'un moins un fichier de ressource soit présent dans la sélection sinon un message est affiché.



Exemple de fichier message.properties :

```
chaine0=chaine 1
chaine1=chaine 2
```

```
chaine2=chaine 3  
chaine1=chaine 4
```

L'exécution de cette fonctionnalité sur l'exemple affiche le résultat suivant :

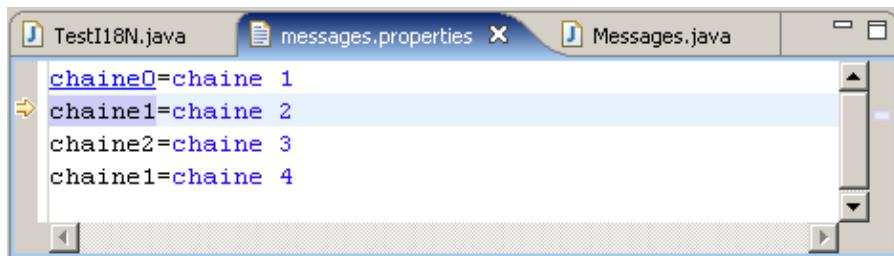


Un double clic sur un des résultats de recherche permet d'ouvrir le fichier de ressources en sélectionnant la première clé concernée par le problème.

### 7.12.2. Recherche de l'utilisation d'une clé

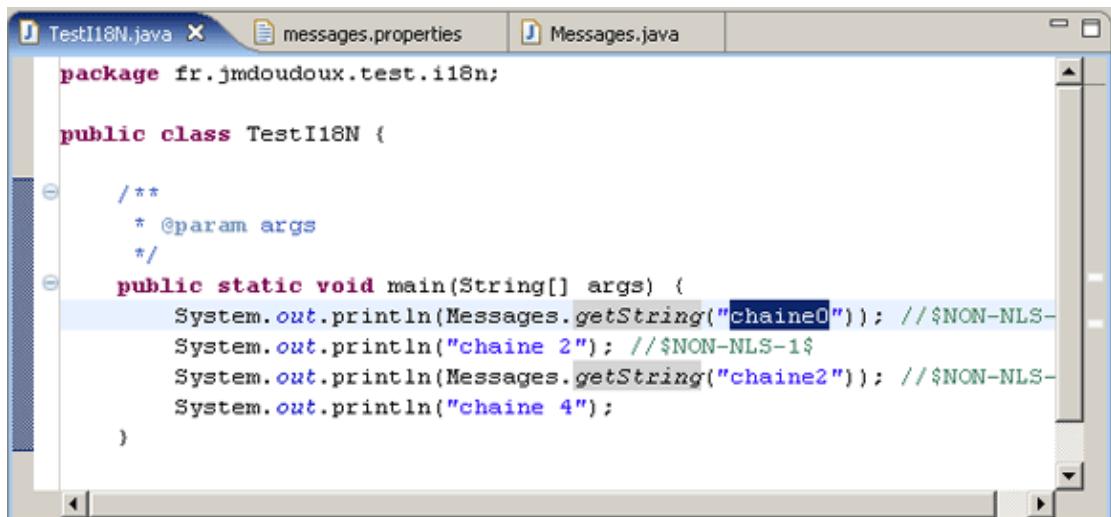


Il est possible de trouver dans le code où est utilisée une clé. Pour cela, il faut ouvrir le fichier de ressource, positionner le curseur de la souris sur la clé en maintenant la touche Ctrl enfoncée (un lien hypertexte s'affiche sous la clé)

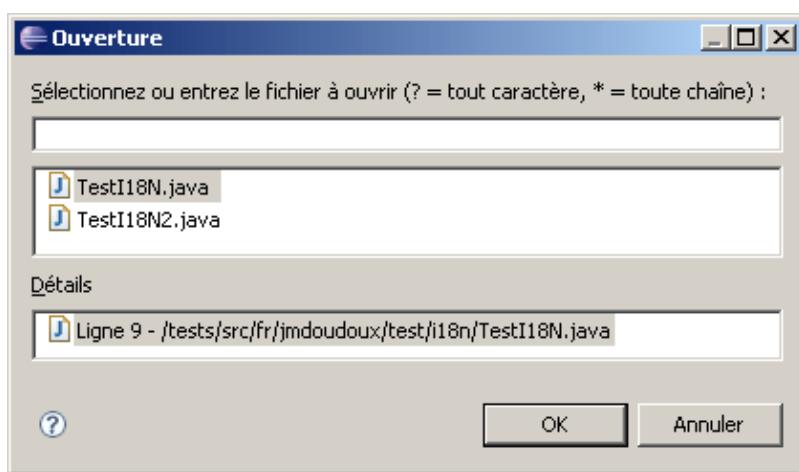


Il suffit alors de cliquer sur le bouton gauche de la source pour qu'Eclipse recherche les utilisations.

Si la clé n'est utilisée qu'une seule fois alors la classe concernée s'ouvre dans l'éditeur ou la clé est utilisée.



Si la clé est utilisée dans plusieurs classes, une boîte de dialogue permet de sélectionner celle qui doit être affichée.

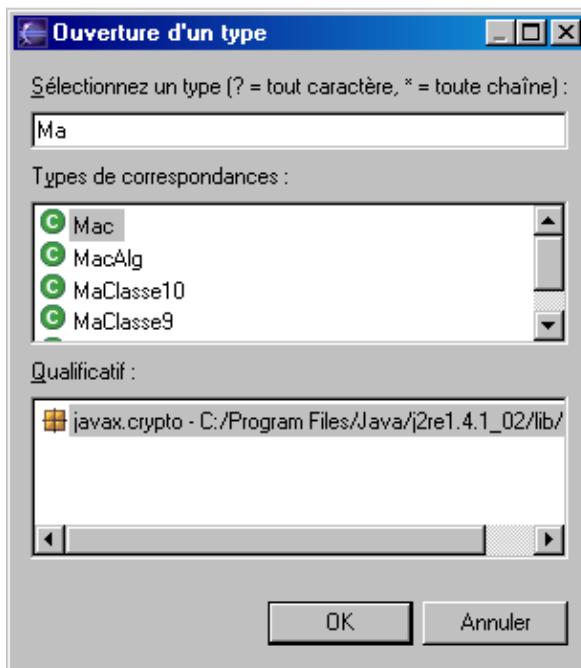


La zone détails affiche les lignes concernées.

Il suffit de sélectionner la classe désirée et de cliquer sur le bouton « OK ».

## 7.13. Ouverture d'un type

Le bouton  de la barre d'outils permet de lancer l'outil "Ouverture d'un type". Cet outil est particulièrement pratique pour rechercher et ouvrir dans l'éditeur le code d'une classe dont on connaît tout ou partie du nom.



Il suffit de saisir le début du nom de la classe ou de l'interface pour que la liste des entités répondant au critère se construise de façon incrémentale.

Il est aussi possible de saisir un motif à l'aide des caractères ? pour représenter un caractère quelconque unique et \* pour représenter aucun ou plusieurs caractères quelconques.

La zone qualificatif affiche le ou les packages où l'entité est définie. Ce nom de package est suivi du nom du projet si l'entité est définie dans l'espace de travail ou du nom du fichier qui contient la version compilée pour une entité externe.

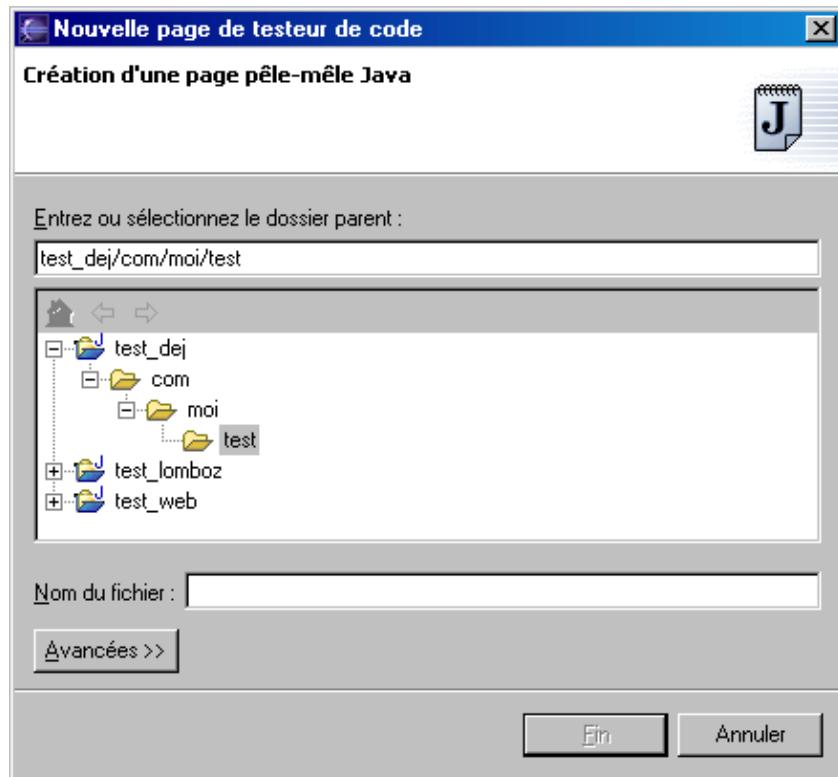
Une fois l'élément voulu sélectionné, un clic sur le bouton "OK" ouvre l'éditeur avec cette entité.

## 7.14. Utilisation du scrapbook

Le scrapbook, traduit par "page de testeur de code", est une fonctionnalité qui permet de tester des morceaux de code dans une machine virtuelle. Cette fonctionnalité très intéressante était déjà présente dans l'outil Visual Age d'IBM.

Pour pouvoir l'utiliser, il faut créer une nouvelle "page de testeur de code", en utilisant une des possibilités d'Eclipse pour créer une nouvelle entité.

Comme pour la création de toute nouvelle entité, un assistant permet de recueillir les informations nécessaires à la création du scrapbook.



Une "page de testeur de code" est physiquement un fichier contenant du code java et ayant une extension .jpage

La seule page de l'assistant permet de sélectionner le répertoire qui va contenir le fichier ainsi que son nom. Par défaut, l'extension .jpage est ajoutée.

Un clic sur le bouton "Fin" permet de générer le fichier et d'ouvrir l'éditeur avec son contenu.

Le grand avantage est de pouvoir tester des morceaux de code sans avoir à créer une classe et une méthode main() et de bénéficier de fonctionnalités particulières pour tester ce code.

Exemple :

```
int i = 0;
i += 10;
System.out.println("Bonjour : "+i);
```

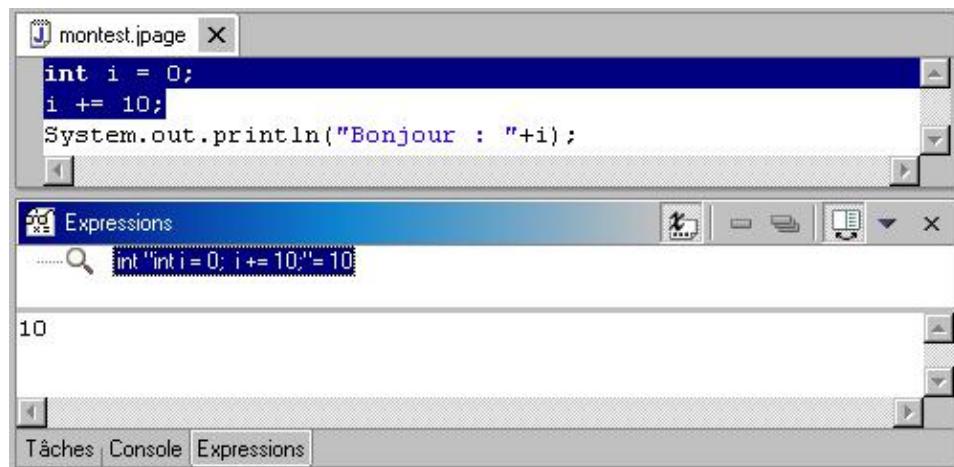
Lors de l'édition du code contenu dans le scrapbook, la barre d'outils est enrichie de plusieurs boutons qui permettent d'utiliser les principales fonctionnalités du scrapbook.

Bouton	Rôle
	Permet d'exécuter un morceau de code et d'évaluer le résultat de l'exécution
	Permet d'afficher dans l'éditeur de code, le résultat de l'exécution d'un morceau de code
	Permet d'exécuter un morceau de code et d'afficher le résultat dans la console
	Permet d'arrêter l'exécution dans la machine virtuelle
	Permet de définir les importations nécessaires

Les trois premiers boutons ne sont utilisables que si un morceau de code est sélectionné dans le scrapbook. Le quatrième bouton n'est utilisable que si une machine virtuelle exécute du code du scrapbook.

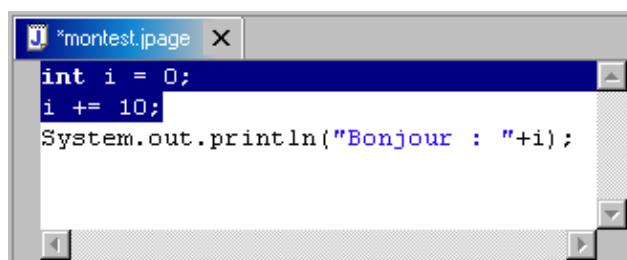
La fonction " Inspecter " permet de visualiser, dans la vue " Expressions ", les valeurs des objets contenus dans le code sélectionné.

Pour la mettre en oeuvre, il suffit de sélectionner un morceau de code dans l'éditeur du scrapbook et de cliquer sur le bouton ou d'utiliser l'option " Inspecter " du menu contextuel.



La fonction " Affichage du résultat de l'exécution " permet d'exécuter un morceau de code et d'afficher le résultat de l'exécution dans l'éditeur juste après la fin de la sélection du morceau de code.

Pour mettre en oeuvre cette fonctionnalité, il faut sélectionner un morceaux de code dans l'éditeur du scrapbook.



Il faut ensuite cliquer sur le bouton ou d'utiliser l'option " Afficher " du menu contextuel.

```
int i = 0;
i += 10;(int) 10
System.out.println("Bonjour : "+i);
```

L'affichage insère le type entre parenthèse et la valeur du résultat dans l'éditeur.

La fonction " Exécuter " permet d'exécuter d'un morceau de code et d'afficher le résultat dans la vue " Console ". Pour mettre en oeuvre cette fonctionnalité, il faut sélectionner un morceau de code dans l'éditeur du scrapbook.

```
int i = 0;
i += 10;
System.out.println("Bonjour : "+i);
```

Il faut ensuite cliquer sur le bouton ou utiliser l'option " Exécuter " du menu contextuel.

Console [org.eclipse.jdt.internal...pbookMain sur l'hôte local :5106] x

Bonjour : 10

Tâches Console Expressions

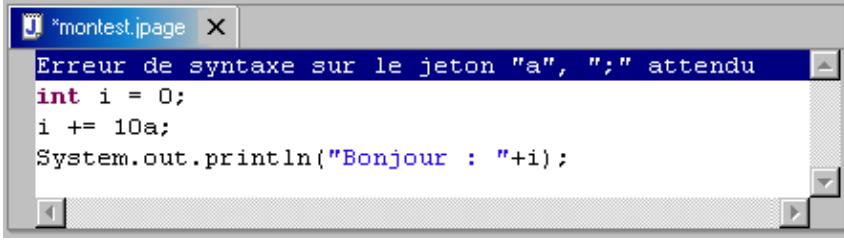
Lors de l'exécution de code dans le scrapbook, une machine virtuelle dédiée est lancée pour cette exécution. Pour pouvoir relancer une exécution, il faut arrêter la machine virtuelle précédemment lancée. L'arrêt de cette machine virtuelle peut se faire en cliquant sur le bouton .

Lors de l'exécution de code dans le scrapbook, si une erreur de syntaxe est détectée, celle ci est signalée directement dans le code de l'éditeur

Exemple :

```
int i = 0;
i += 10a;
System.out.println("Bonjour : "+i);
```

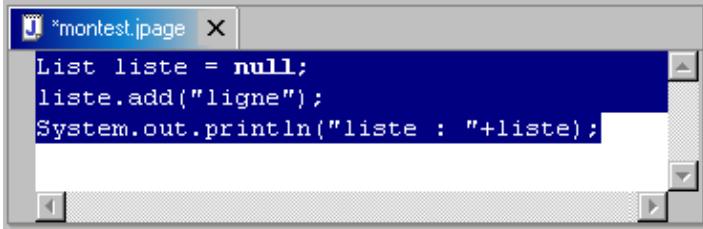
Lors de l'exécution de ce morceau de code, l'erreur suivante est affichée dans l'éditeur



The screenshot shows the Eclipse Java editor with a file named "montest.jpage". The code contains a syntax error:`int i = 0;
i += 10a;
System.out.println("Bonjour : "+i);`A red squiggly underline is under the character 'a' in the line `i += 10a;`, indicating a syntax error.

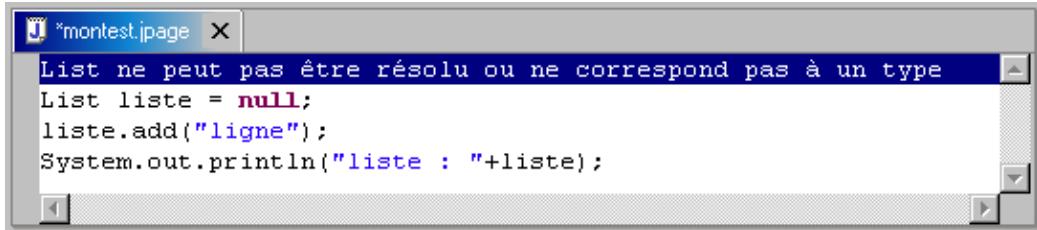
La structure d'un fichier java classique n'étant pas respecter dans le scrapbook, la gestion des clauses d'import est gérée de façon particulière.

Exemple :



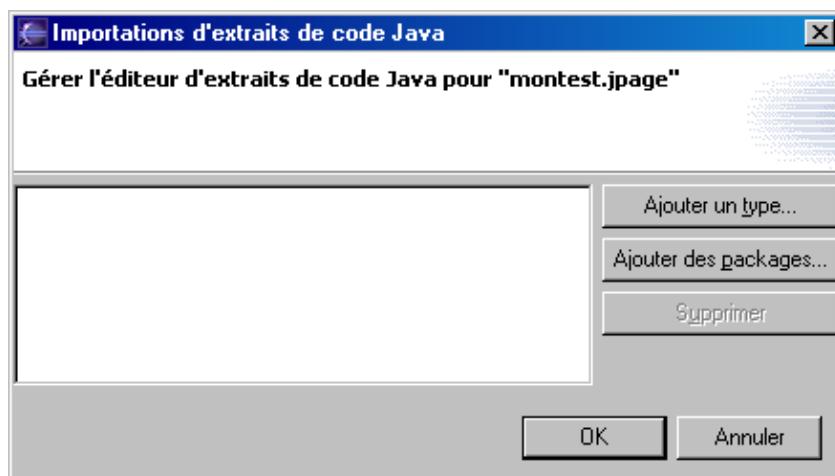
The screenshot shows the Eclipse Java editor with a file named "montest.jpage". The code contains a type resolution error:`List liste = null;
liste.add("ligne");
System.out.println("liste : "+liste);`The word `List` is highlighted in blue, indicating it is unresolved.

L'exécution de ce morceau de code génère l'erreur suivante :

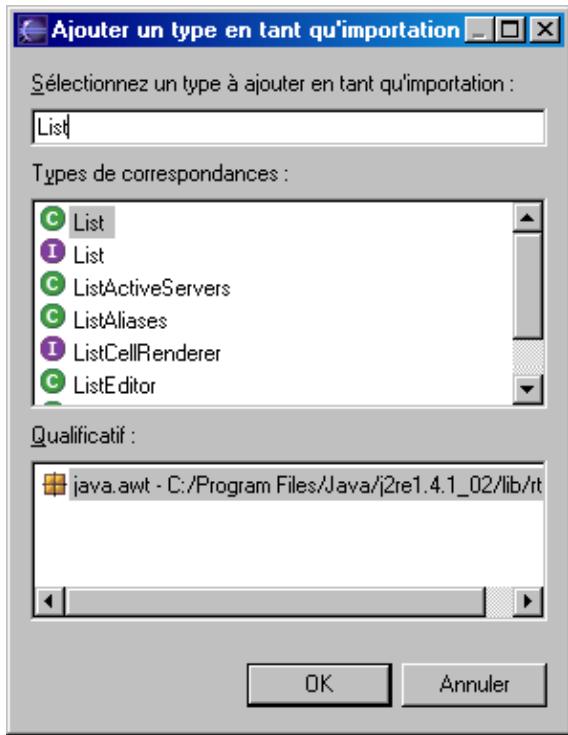


The screenshot shows the Eclipse Java editor with a file named "montest.jpage". The code contains a type resolution error:`List ne peut pas être résolu ou ne correspond pas à un type
List liste = null;
liste.add("ligne");
System.out.println("liste : "+liste);`The message `List ne peut pas être résolu ou ne correspond pas à un type` is displayed at the top of the editor.

Pour ajouter une clause import, il cliquer sur le bouton  ou utiliser l'option " Définit les importations " du menu contextuel.

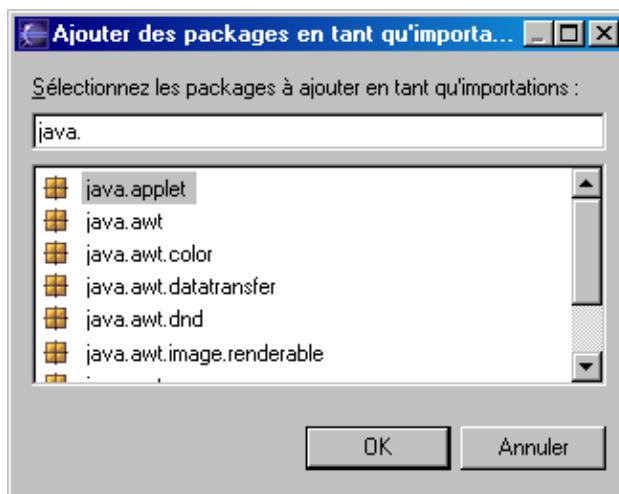


Un clic sur le bouton " Ajouter un type " permet d'importer une classe ou une interface dans le scrapbook.



La zone de saisie du type permet une recherche incrémentale dans toutes les entités définies dans le chemin de compilation du projet.

Un clic sur le bouton " Ajouter un Package " permet d'importer un package



La zone de saisie du package permet une recherche incrémentale du package désiré parmi tous ceux définis dans le chemin de compilation du projet.

Enfin pour supprimer l'importation d'une entité, il suffit de la sélectionner et de cliquer sur le bouton " Supprimer ".

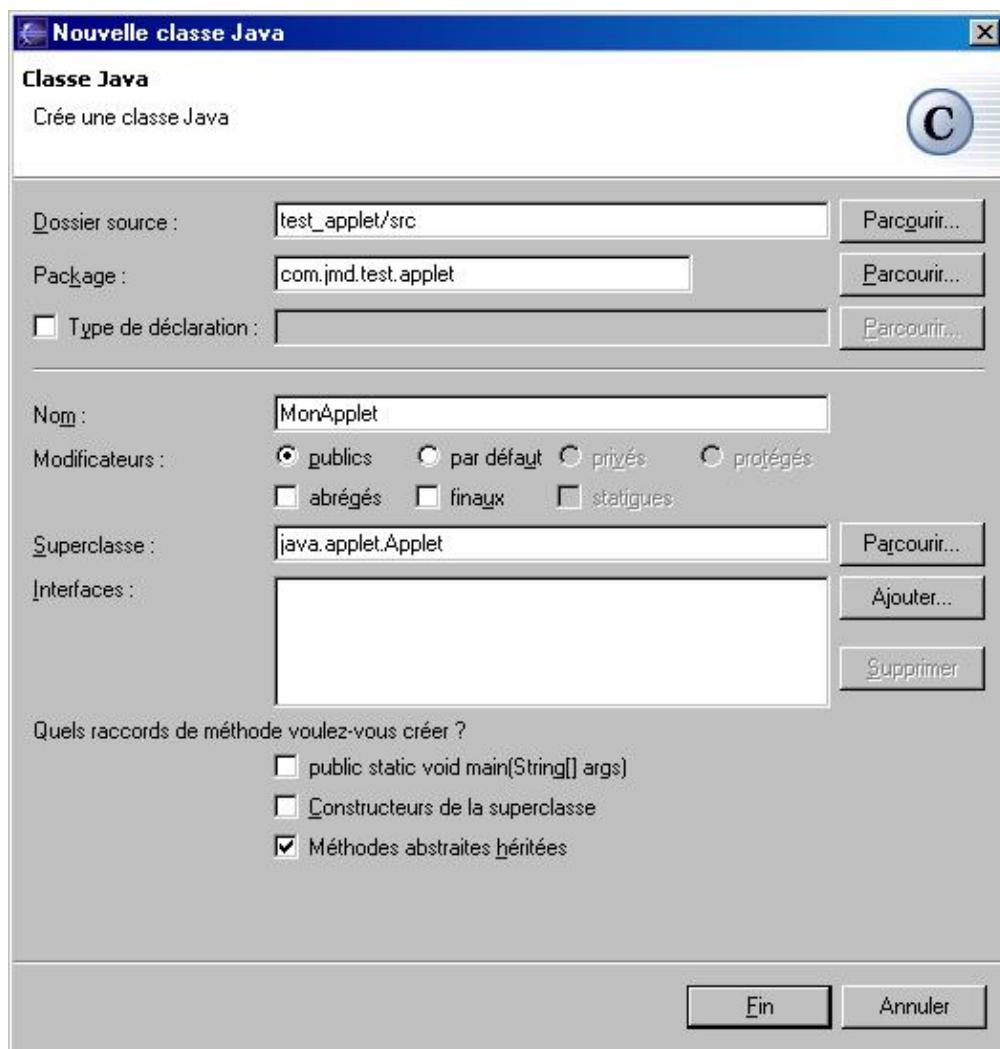
L'assistant de code, est bien sûre disponible dans l'éditeur du scrapbook toujours en utilisant la combinaison de touches "Ctrl" + "Espace".

The screenshot shows the Eclipse Java code editor with the file "montest.java" open. A code completion tooltip is displayed over the variable declaration "Array liste = null;". The tooltip lists several suggestions starting with 'Array':

- Array - java.lang.reflect
- ArrayList - java.util
- Arrays - java.util
- ArrayIndexOutOfBoundsException - java.lang
- ArrayStoreException - java.lang
- ArrayPersistenceDelegate - java.beans

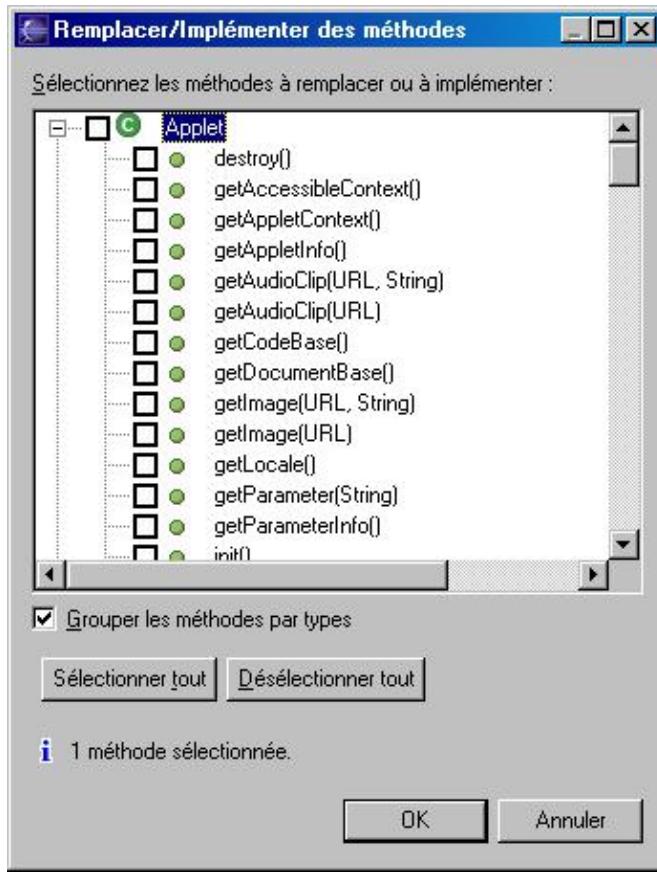
## 7.15. Le développement d'applets

Il faut créer un nouvelle classe qui hérite de la classe `java.applet.Applet`



La nouvelle classe est créée et l'éditeur de code ouvre son source.

Pour faciliter la réécriture des méthodes utiles, il est possible d'utiliser l'option « source / Remplacer/Implémenter les méthodes ... » du menu contextuel de l'éditeur de code.



Il suffit de cocher chaque méthode désirée et de cliquer sur le bouton « Ok ».

Le code source est enrichi avec la ou les méthodes sélectionnées :

#### Exemple :

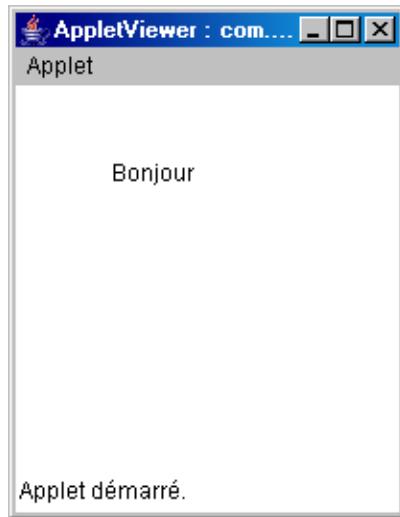
```
/* (non-Javadoc)
 * @see java.awt.Component#paint(java.awt.Graphics)
 */
public void paint(Graphics arg0) {
    // TODO Raccord de méthode auto-généré
    super.paint(arg0);
}
```

Il suffit d'insérer le code dans ces méthodes.

#### Exemple :

```
/* (non-Javadoc)
 * @see java.awt.Component#paint(java.awt.Graphics)
 */
public void paint(Graphics arg0){
    super.paint(arg0);
    arg0.drawString("Bonjour", 50, 50);
}
```

Pour exécuter une applet, il faut utiliser l'option « Exécuter en tant que / Applet » du bouton « Exécuter » de la barre d'outils.

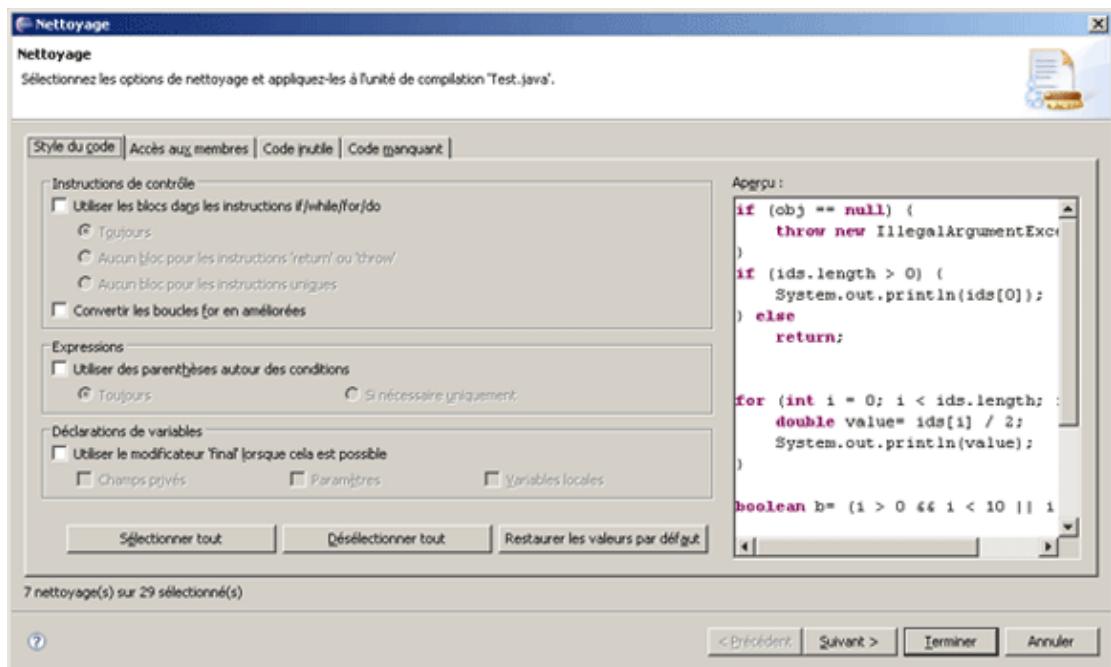


## 7.16. Le nettoyage du code



Le nettoyage du code peut s'appliquer sur un projet, un package ou un fichier. Il suffit de sélectionner l'un de ces éléments et d'utiliser l'option « Source / Nettoyer ... » du menu contextuel.

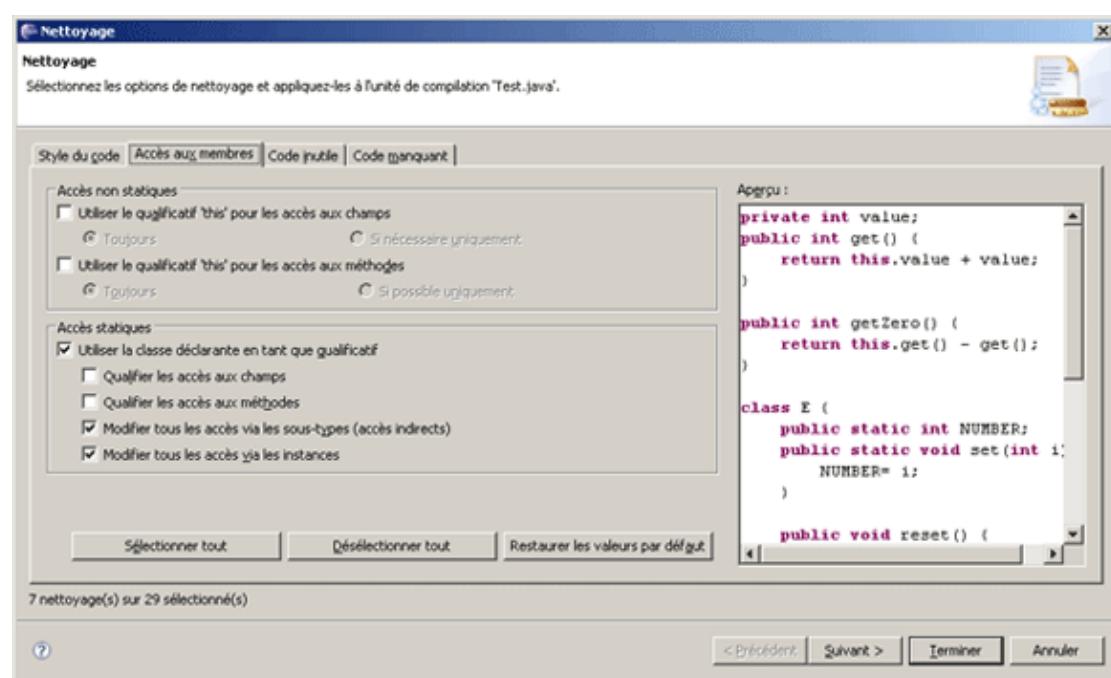
Les fonctionnalités offertes par cet assistant sont nombreuses : la partie « Aperçu » permet d'avoir une idée sur les possibilités de chaque fonctionnalité.



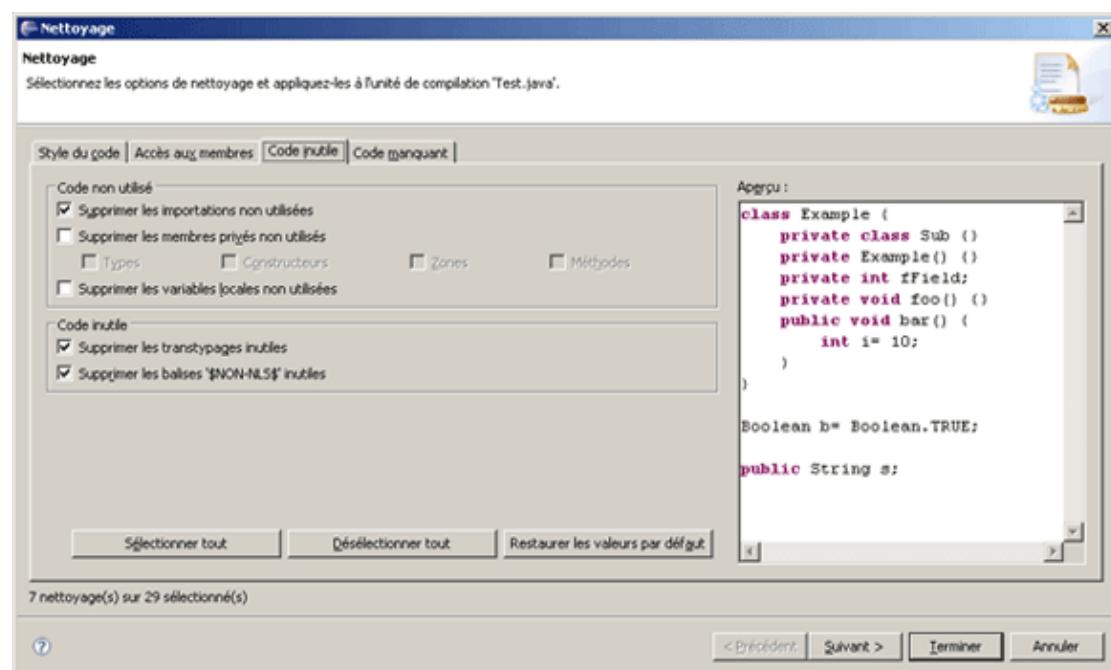
L'option « Utiliser les blocs dans les instructions if/ while /for/do » permet de définir le bloc contenant une instruction avec des accolades.

L'option « Convertir les boucles for améliorées » permet de convertir les boucles au format Java 5.0.

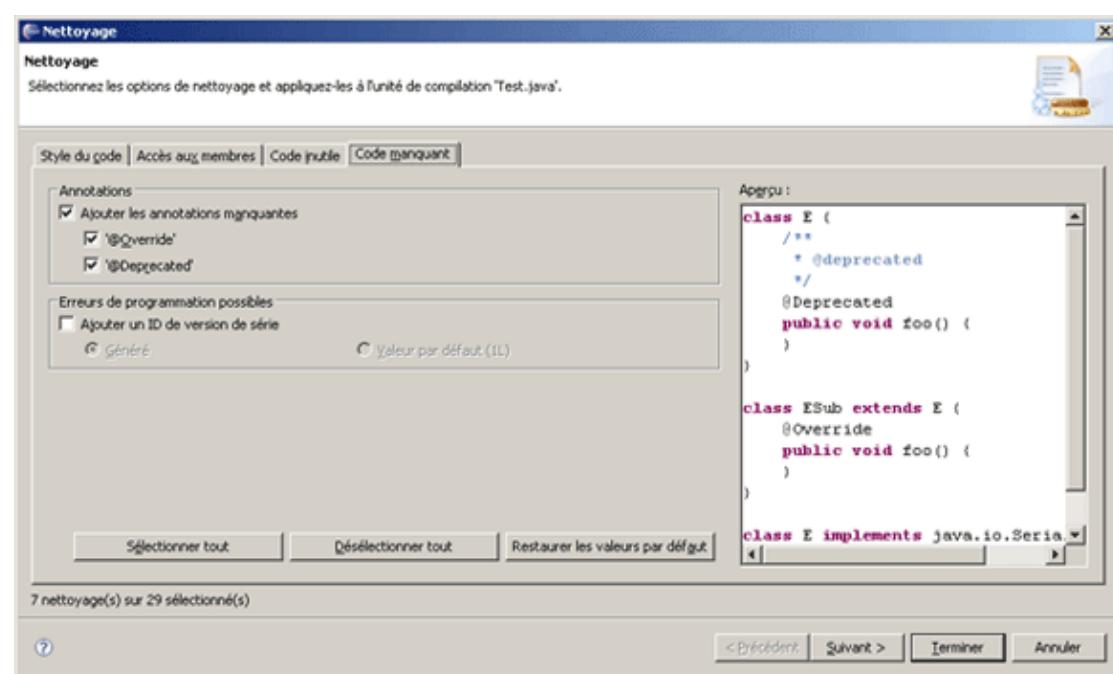
Le libellé de l'option « Utiliser le modificateur final lorsque cela est possible» est assez explicite.



Ces options permettent de modifier l'accès aux membres.

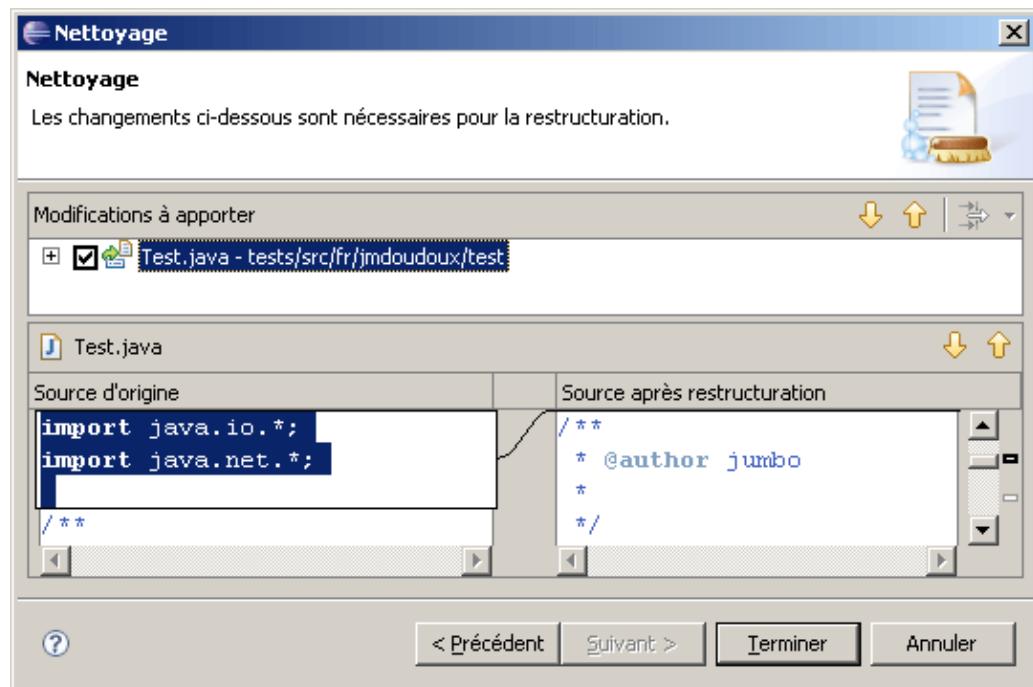


Ces options permettent de supprimer certaines déclarations inutiles.



Les options de l'onglet « Code manquant » permet d'ajouter les annotations manquantes définies par Java 5.0 et de générer un ID pour les classes sérialisables.

Une fois les nettoyages sélectionnés, il faut cliquer sur le bouton « Suivant »



Un aperçu des modifications est affiché. Chaque modification peut être validée ou non en utilisant les cases à cocher dans « Modifications à apporter »

Cliquer sur le bouton « Terminer » pour exécuter les modifications.

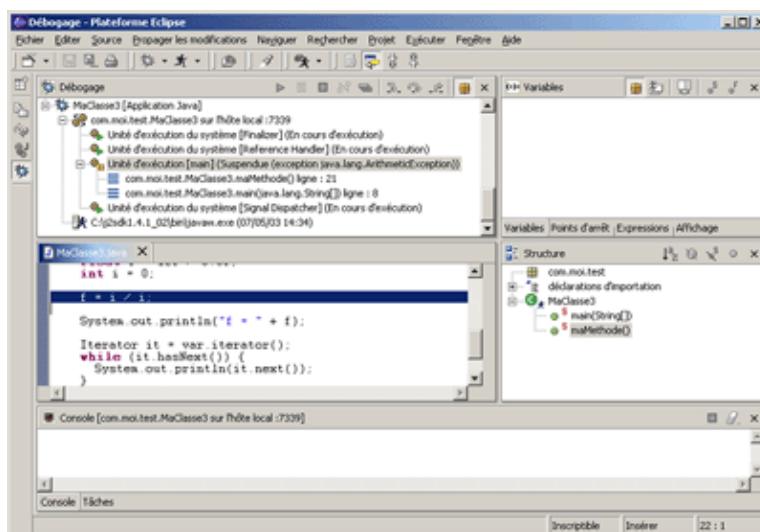
## 8. Déboguer du code Java

# Chapitre 8

### 8.1. La perspective "Débogage"

Pour déboguer du code Java, Eclipse propose une perspective dédiée : la perspective "Débogage".

Celle-ci est automatiquement affichée lorsqu'une application est lancée sous le contrôle du débogueur en utilisant le bouton  de la barre d'outils. Son principe de fonctionnement est identique au bouton d'exécution situé juste à côté de lui.



Par défaut, la perspective "Débogage" affiche quelques vues aussi présentes dans la perspective Java (les vues "Structure" et "Console") ainsi que l'éditeur de code Java.

Elle affiche aussi plusieurs vues particulièrement dédiées au débogage.

### 8.2. Les vues spécifiques au débogage

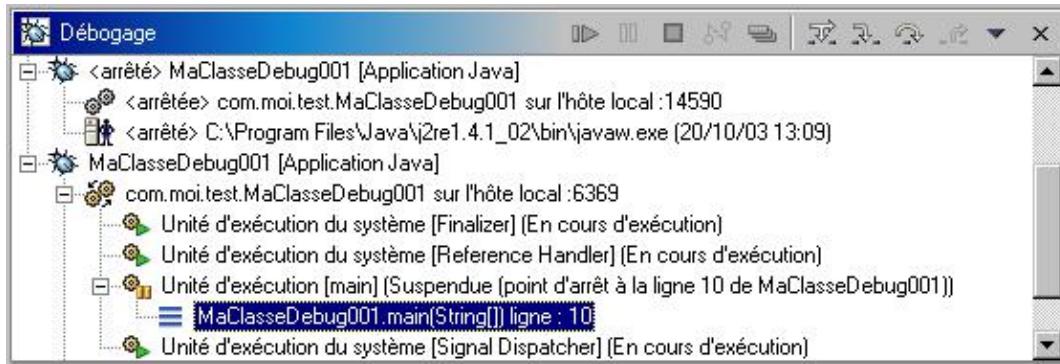
Les vues spécifiques au débogage sont :

- la vue "Débogage" : affiche sous la forme d'une arborescence, les différents processus en cours d'exécution ou terminés
- la vue "Variables" : affiche les variables utilisées dans les traitements en cours de débogage
- la vue "Points d'arrêts" : affiche la liste des points d'arrêts définis dans l'espace de travail
- la vue "Expressions" : permet d'inspecter une expression en fonction du contexte des données en cours d'exécution

- la vue "Affichage" : permet d'afficher le résultat de l'évaluation d'une expression

### 8.2.1. La vue "Débogage"

Cette vue affiche sous la forme d'une arborescence, les différents processus en cours d'exécution ou terminés.



Cette vue possède plusieurs icônes qui permettent d'agir sur les éléments affichés.

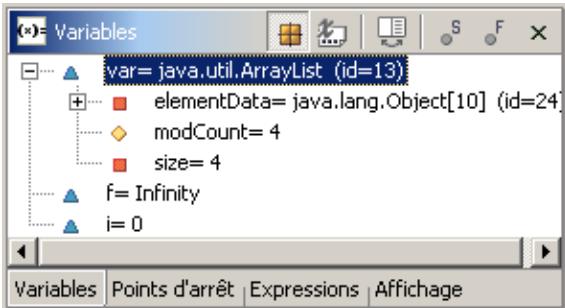
Icône	Rôle
	Reprendre l'exécution précédemment interrompue
	Interrompre l'exécution du processus
	Demande de mettre fin au processus
	Enlève de la liste tous les processus qui sont terminés
	Exécute la ligne courante et arrête l'exécution sur la première ligne de code incluse dans la première méthode de la ligne courante (raccourci : F5)
	Exécute la ligne courante et arrête l'exécution avant la ligne suivante (raccourci : F6)
	Exécute le code de la ligne courante jusqu'à la prochaine instruction return de la méthode (raccourci : F7)
	Affiche ou non le nom pleinement qualifiés des objets

L'exécution d'un processus est automatiquement suspendu dans les cas suivants :

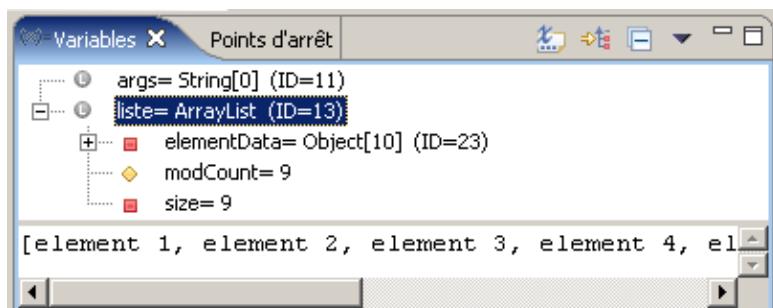
- un point d'arrêt est rencontré
- une exception non capturée est propagée jusqu'au sommet de la pile d'appel

### 8.2.2. La vue "Variables"

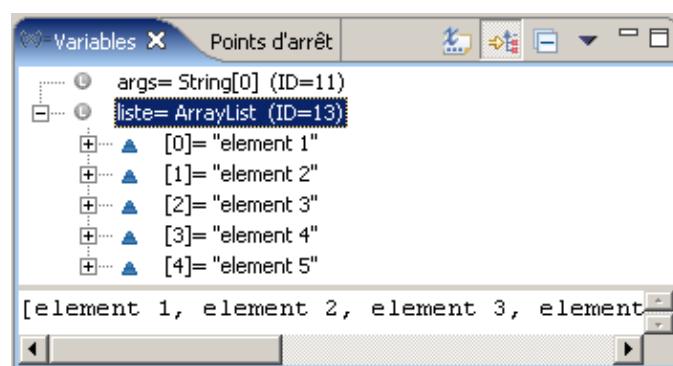
Cette vue permet de visualiser les valeurs des variables utilisées dans les traitements en cours de débogage. Ces valeurs peuvent être unique si la variable est de type primitive ou former une arborescence contenant chacun des champs si la variable est un objet.



La liste des éléments d'une collection est affichée dans la vue « Variable » si la variable inspectée est une collection.

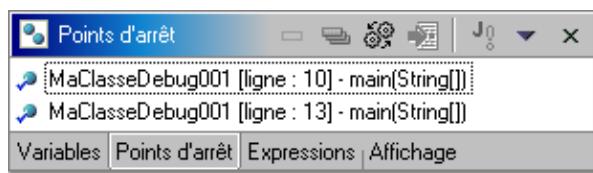


Le bouton permet d'afficher le contenu des éléments de la collections.



### 8.2.3. La vue "Points d'arrêts"

Cette vue permet de recenser tous les points d'arrêts définis dans l'espace de travail.

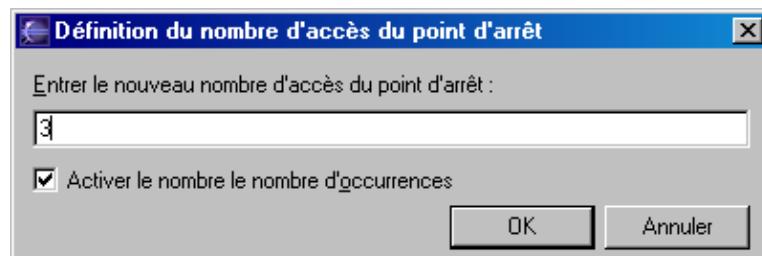


Un double clic sur un des points d'arrêts permet d'ouvrir l'éditeur de code directement sur la ligne ou le point d'arrêt est défini. Cette action est identique à l'utilisation de l'option "Accéder au fichier" du menu contextuel ou à un clic sur le bouton de la barre de titre de la vue.

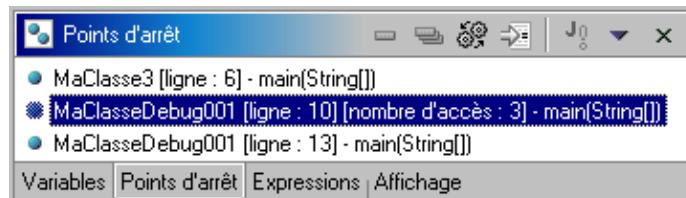
Il est possible grâce au menu contextuel de contrôler les points d'arrêts.

L'option "Nombres d'occurrences" permet de préciser le nombre fois ou le point d'arrêt exécuté sera ignoré avant qu'il n'interrrompe l'exécution. Ceci est particulièrement pratique lorsque l'on débogue du code contenu dans une boucle et que l'on connaît l'itération qui pose problème.

Lors du clic sur l'option "Nombres d'occurrences", une boîte de dialogue permet de préciser le nombre de passage sur le point d'arrêt à ignorer.



Ce nombre est indiqué dans la vue "Points d'arrêts", précédé du libellé "nombre d'accès".

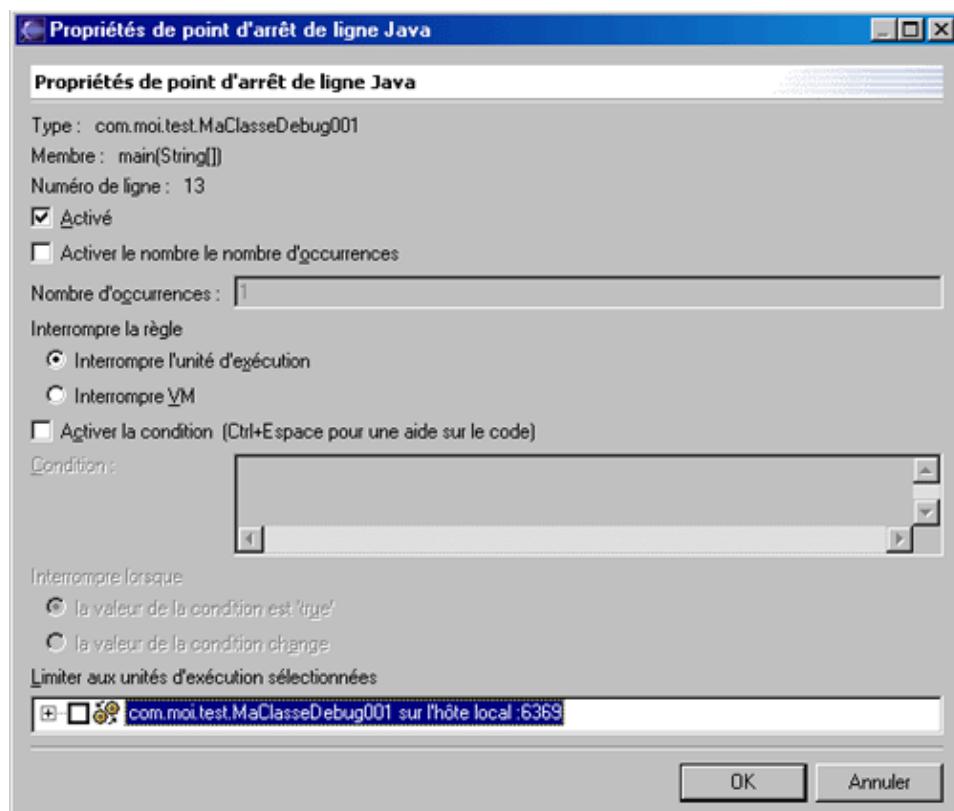


Il est possible d'activer ou de désactiver le point d'arrêt sélectionné respectivement grâce à l'option "Activer" ou "Désactiver".

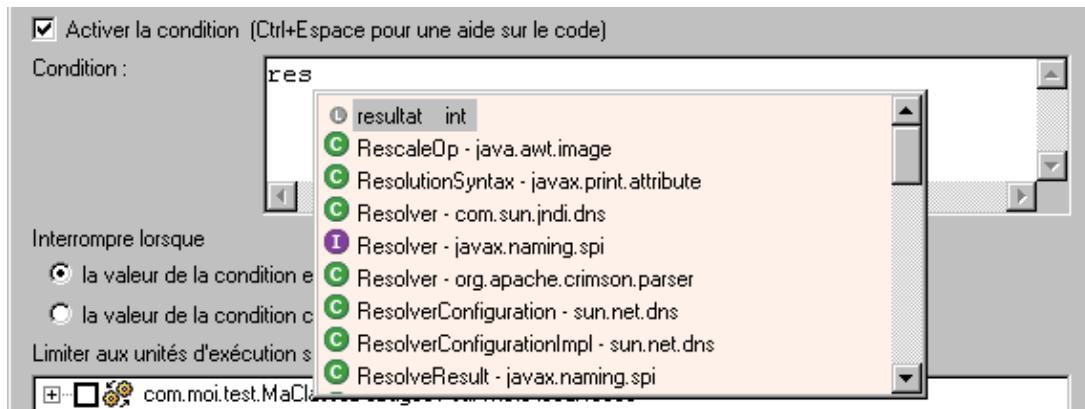
L'option "Supprimer" permet de supprimer le point d'arrêt sélectionné. Il est aussi possible d'utiliser le bouton de la barre de titre de la vue.

L'option "Supprimer tout" permet de supprimer tous les points d'arrêts définis. Il est aussi possible d'utiliser le bouton de la barre de titre de la vue.

L'option "Propriétés ..." permet d'ouvrir une boîte de dialogue pour régler les différents paramètres du point d'arrêt sélectionné.

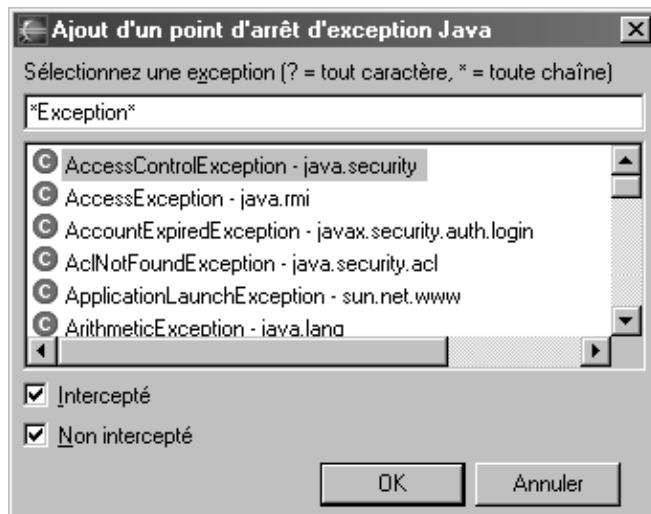


Un de ces paramètres les plus intéressants est la possibilité de mettre une condition d'activation du point d'arrêt. Il suffit pour cela de cocher la case "Activer la condition" et de la saisir dans la zone de texte prévue à cet effet. Dans cette zone de texte, l'assistant de complétion de code est utilisable.

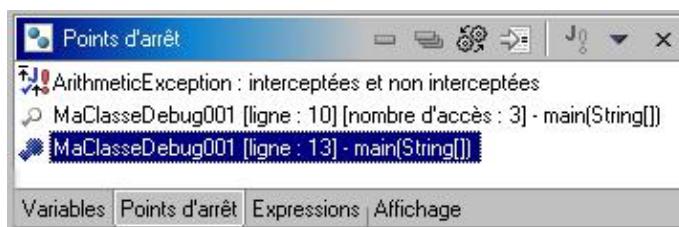


Dans cette vue, il est aussi possible de demander l'arrêt de l'exécution non pas à l'exécution d'une ligne de code mais à la lever d'une exception particulière. Pour cela, il suffit de cliquer sur le bouton .

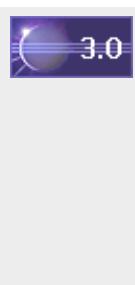
Une boîte de dialogue permet de sélectionner l'exception concernée.



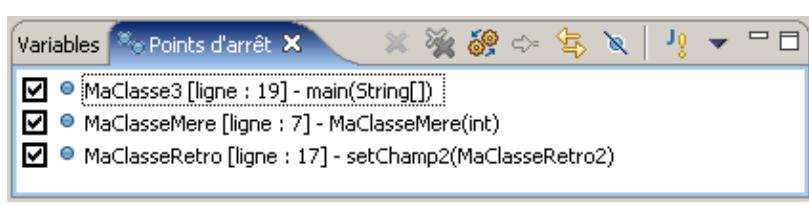
Le nouveau point d'arrêt est affiché dans la vue "Point d'arrêts".



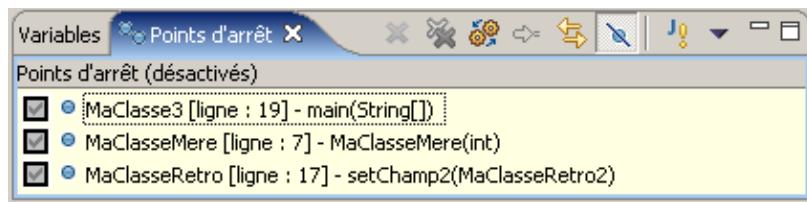
Il est possible de préciser si l'arrêt se fait sur une exception interceptée, non intercepté ou dans les deux cas. Ce type de point d'arrêt possède les mêmes propriétés que les points d'arrêts liés à une ligne de code.



Une case à cocher devant chaque point d'arrêt permet d'activer ou non un point d'arrêt.

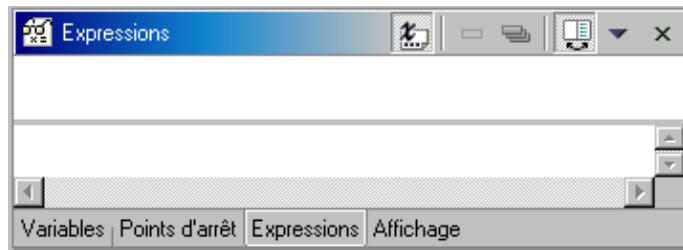


Le bouton  permet de demander la désactivation de tous les points d'arrêts enregistrés dans l'espace de travail.



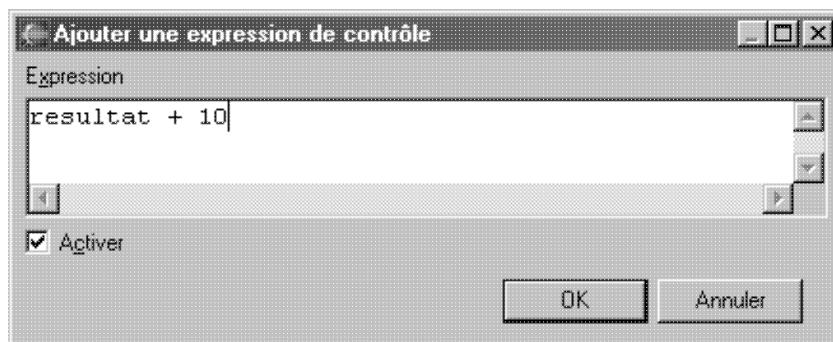
### 8.2.4. La vue "Expressions"

La vue "Expressions" permet d'inspecter la valeur d'une expression selon les valeurs des variables dans les traitements en cours de débogage.

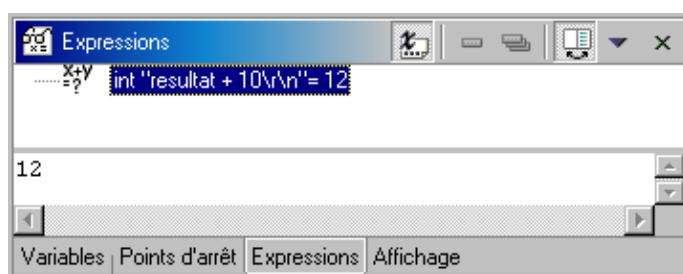


Pour ajouter une nouvelle expression, il suffit d'utiliser l'option  "Ajouter une expression de contrôle Java" du menu contextuel.

Une boîte de dialogue permet de saisir l'expression.



La vue "Expressions" affiche le résultat de l'évaluation en tenant compte du contexte d'exécution.



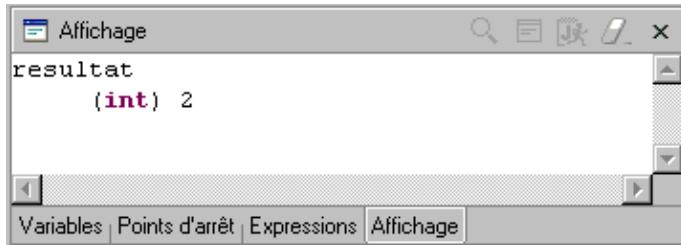
L'option "Réévaluer l'expression de contrôle" permet de recalculer la valeur de l'expression.

L'option "Editer l'expression de contrôle" permet de modifier l'expression.

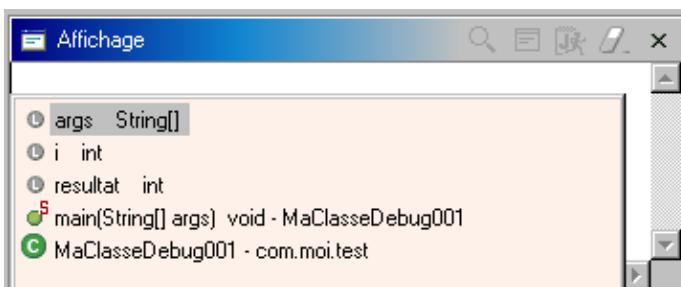
## 8.2.5. La vue "Affichage"

La vue "Affichage" permet d'afficher le résultat de l'évaluation d'une expression.

La valeur de l'expression à afficher peut avoir plusieurs origines. Le plus pratique est de sélectionner un morceau de code dans l'éditeur et d'utiliser l'option "Afficher" du menu contextuel. Le résultat de l'évaluation est affiché dans la vue "Affichage".



Il est aussi possible d'ajouter directement une expression dans la vue en utilisant l'option "Assistant de contenu" du menu contextuel.



Il faut sélectionner le membre ou la variable concerné.

Pour obtenir des informations sur sa valeur, il suffit de sélectionner l'expression dans la vue. Diverses actions sont alors disponibles avec les boutons dans la barre de titre de la vue ou dans les options du menu contextuel :

- inspecter : exécution et évaluation dans la vue "Expressions"
- afficher : afficher le résultat dans la vue "Affichage"
- exécuter : exécuter l'expression sélectionnée. Il est ainsi possible de modifier la valeur d'une variable

## 8.3. Mise en oeuvre du débogueur

La mise en oeuvre du débogueur reste comparable à celle proposée par d'autres IDE : mise en place d'un point d'arrêt, exécution plus ou moins rapide dans le débogueur pour arriver à cibler le problème.

### 8.3.1. Mettre en place un point d'arrêt

Pour placer un point d'arrêt, il suffit dans l'éditeur de double cliquer dans la barre de gauche pour faire apparaître une icône ronde bleue.

```
MaClasse.java X
package com.moi.test;

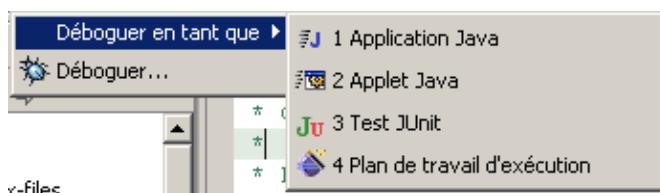
public class MaClasse {

    public static void main(String[] args) {
        int valeur;
        valeur = 10;
        System.out.println("valeur = "+valeur);
    }
}
```

### 8.3.2. Exécution dans le débogueur

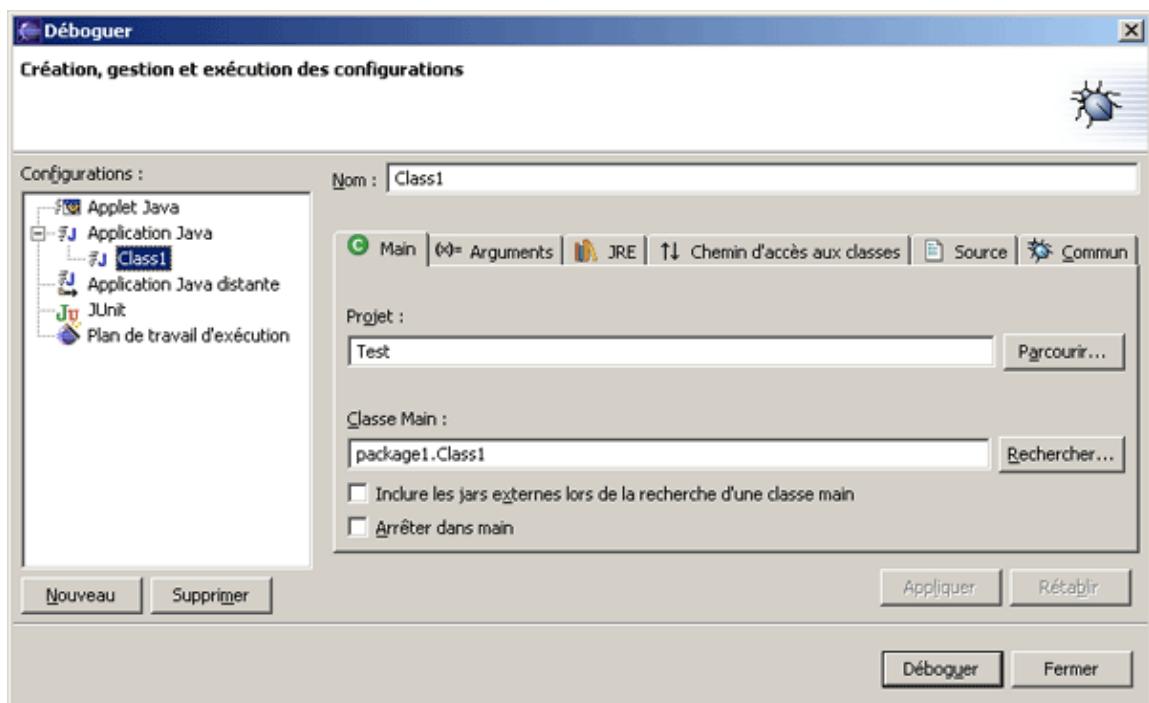
Pour lancer l'exécution dans le débogueur, il faut ouvrir le menu en cliquant sur flèche du bouton .

Un menu déroulant propose de déboguer les dernières applications exécutées ou de lancer le débogage de la source courante en proposant deux options de menu :



Le plus simple pour lancer le débogage d'une application est de sélectionner l'option « Déboguer en tant que / Application Java »

L'option "Déboguer ..." permet de fournir des paramètres précis en vue de l'exécution d'une application sous le débogueur. Un assistant permet de sélectionner la classe et de préciser les paramètres.



Il ne reste plus qu'à mettre en oeuvre les différentes fonctionnalités proposées par les vues de la perspective pour déboguer le code.

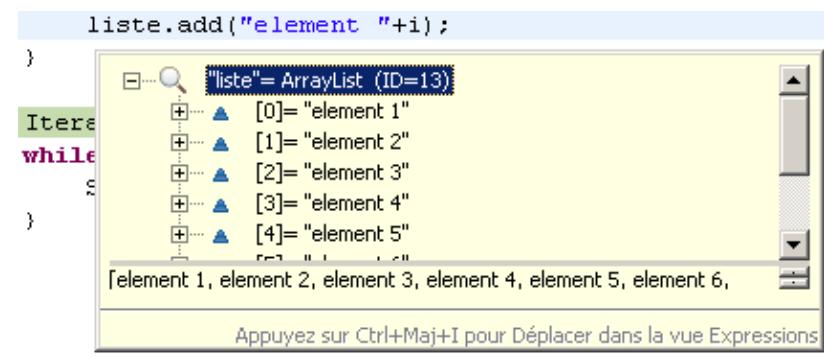


Lors d'une demande d'inspection d'une variable dans l'éditeur affichant le code, une bulle d'aide affiche les valeurs de la variable.



Pour accéder à la vue expression, il suffit de réutiliser la combinaison de touche Ctrl+Maj+I.

Dans la fenêtre affichant le code, l'inspection d'une variable ouvre une bulle d'aide qui affiche en popup les informations sur la collection



### 8.3.3. Exportation / Importation des points d'arrêt



Il est possible d'exporter dans un fichier les points d'arrêts en utilisant l'option Exporter du menu principal « Fichier ». Sélectionnez « Général / Points d'arrêt »



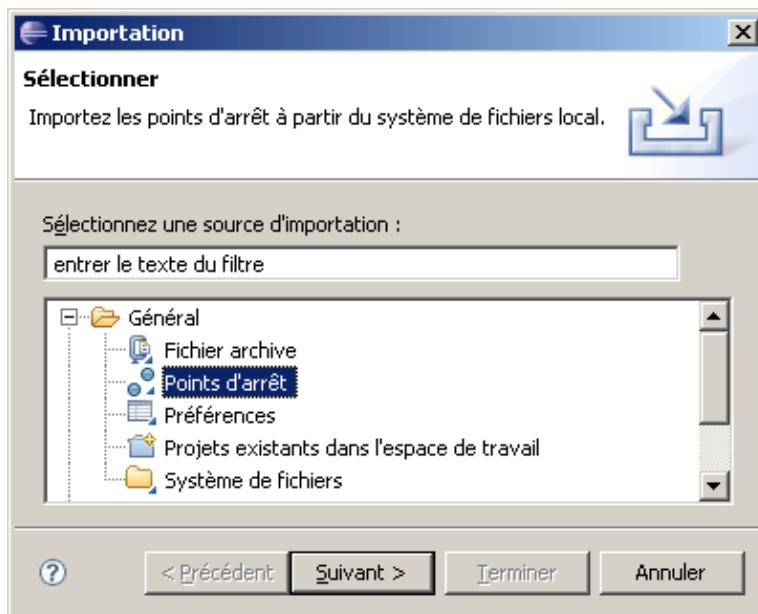
Cliquez sur le bouton « Suivant »



Il faut sélectionner les points d'arrêts à inclure dans le fichier, saisir le nom du fichier et cliquer sur le bouton « Terminer »

L'importation se fait en utilisant l'option « Importation » du menu principal Fichier

Il suffit de sélectionner « Général / Points d'arrêt »



Cliquez sur le bouton « Suivant »

## 9. Le refactoring

# Chapitre 9

Eclipse intègre de puissantes fonctionnalités pour faciliter le refactoring. Le refactoring consiste à modifier la structure d'un fichier source et si nécessaire à propager ces modifications dans les autres fichiers sources pour maintenir autant que possible l'intégrité du code existant.



Un menu nommé « Propager les modifications » permet d'utiliser certaines de ces fonctionnalités : il est présent dans le menu principal et dans de nombreux menus contextuels. Chacune de ces fonctionnalités est applicable sur un ou plusieurs types d'entités selon le cas : projets, classes et membres d'une classe. Certaines de ces fonctionnalités possèdent un raccourci clavier.

Type de modification	Fonctionnalités	Entité(s) concernée(s)	Raccourci clavier
Structure du code	Renommer	projets, packages, classes, champs, méthodes, variables locales, paramètres	Alt+Maj+R
	Déplacer	projets, packages, classes, méthodes et champs statiques	Alt+Maj+V
	Changer la signature de la méthode	méthodes	
	Convertir une classe anonyme en classe imbriquée	classes anonymes	
	Convertir un type imbriqué au niveau supérieur	classes imbriquées	
Structure au niveau de la classe	Transférer	méthodes ou champs	
	Extraire	méthodes ou champs	
	Extraire une interface	classes	
	Utiliser le supertype si possible	classes	
Structure à l'intérieur d'une classe	Intégrer	méthodes, constantes et variable locales	Alt+Maj+I
	Extraire la méthode	morceau de code sélectionné	Alt+Maj+M
	Extraire la variable locale	morceau de code sélectionné pouvant être transformé en variable	Alt+Maj+L
	Extraire une constante		

		morceau de code sélectionné pouvant être transformé en constante	
	Convertir la variable locale en zone	variables locales	
	Encapsuler la zone	champs	

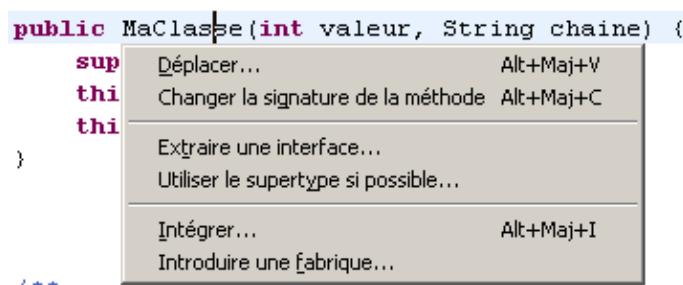


3.1 Le menu « Propager les modifications » permettant d'utiliser les fonctionnalité de refactoring a été renommé en «Restructurer».



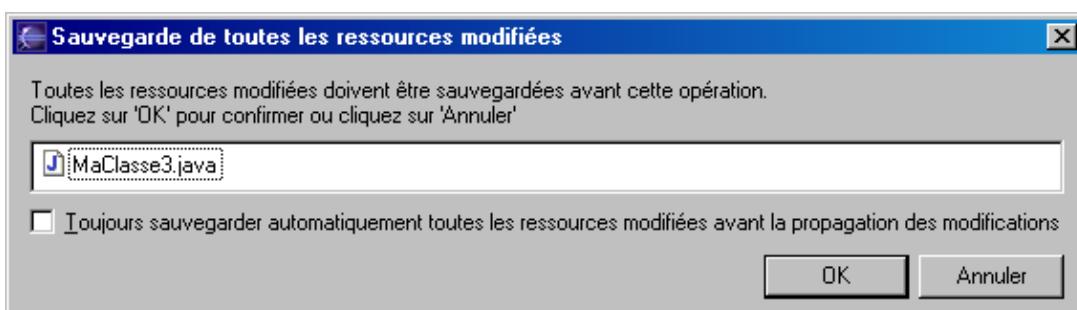
3.0 L'accès aux fonctions de refactoring est contextuel : seules les fonctions applicables dans le contexte courant peuvent être activées.

L'éditeur propose un menu contextuel pour accéder rapidement aux fonctions de refactoring en utilisant la combinaison de touche Alt+Maj+T



Les fonctions proposées le sont en fonction du contexte (le type d'entité sur lequel le curseur est placé).

Avant de pouvoir utiliser ces fonctionnalités, il faut sauvegarder l'élément qui sera modifié sinon un message d'erreur est affiché.

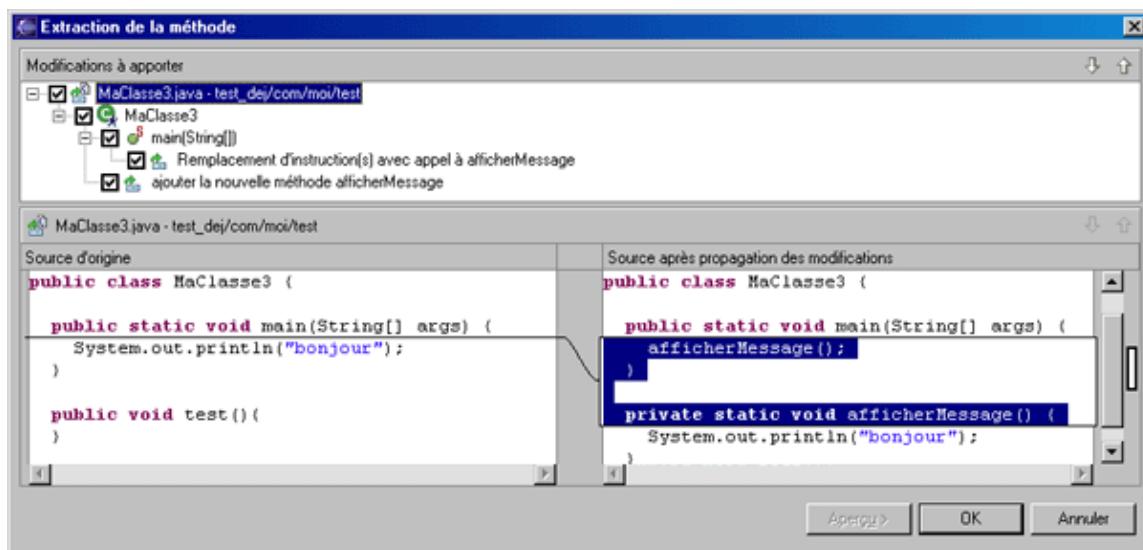


L'usage de ces fonctions utilise toujours le même principe de fonctionnement :

1. sélection de l'élément concerné par la fonction de refactoring
2. appel de la fonction en utilisant l'option du menu « Propager les modifications »
3. renseigner les informations utiles à la fonction dans la boîte de dialogue
4. pré-visualiser et valider individuellement les modifications qui seront apportées
5. demander la mise en oeuvre des modifications en cliquant sur « Ok » (ceci peut être fait sans demander la prévisualisation)

Les fonctionnalités du refactoring sont accessibles à partir des vues "Packages" et "Structure" ou de l'éditeur de code Java selon l'entité concernée par la modification. Dans les vues, il suffit de sélectionner le ou les éléments concernés. Dans l'éditeur de code, il faut sélectionner le nom de l'élément ou de positionner le curseur dessus avant d'appeler la fonctionnalité.

Toutes les fonctionnalités utilisent un assistant qui propose toujours à la fin un bouton « Aperçu > » permettant d'ouvrir une boîte de dialogue qui permet de pré-visualiser chacune des modifications résultant de l'usage de la fonction.



Une arborescence permet d'afficher chacun des éléments qui sera modifié ainsi que toutes les modifications pour ces éléments. Il est possible de cocher ou non tout ou partie des modifications pour qu'elles soient appliquées ou non.

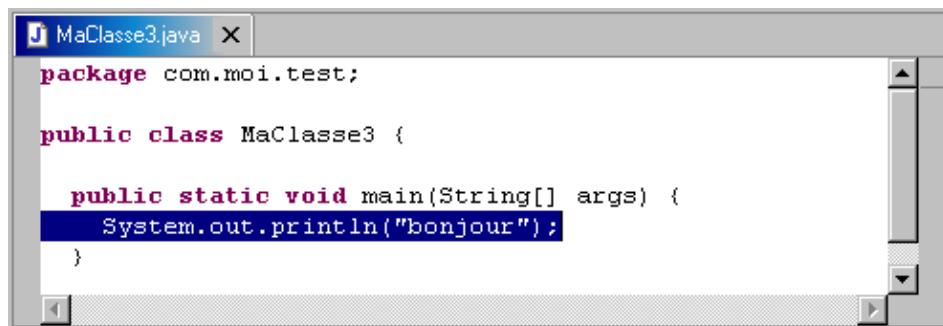
La partie inférieure présente, pour l'élément sélectionné, le code source actuel à gauche et le code source tel qu'il sera après l'application de la fonction à droite.

Un clic sur le bouton « OK » permet de mettre en oeuvre toutes les modifications qui sont cochées.

## 9.1. Extraction d'une méthode

Cette fonction permet de transférer un morceau de code dans une méthode qui sera créée pour l'occasion.

Pour l'utiliser, il faut sélectionner le morceau de code.



Il faut sélectionner l'option « Extraire la méthode ... » du menu principal ou du menu contextuel « Propager les modifications ». Une boîte de dialogue permet de saisir le nom de la méthode et de sélectionner le modificateur d'accès.



Un clic sur le bouton « OK » permet de mettre en oeuvre automatiquement les modifications.

```

package com.moi.test;

public class MaClasse3 {

    public static void main(String[] args) {
        afficherMessage();
    }

    private static void afficherMessage() {
        System.out.println("bonjour");
    }
}

```

Dans le cas où le code sélectionné contient une ou plusieurs variables, la boîte de dialogue contient en plus la liste des variables détectées dans le code.

```

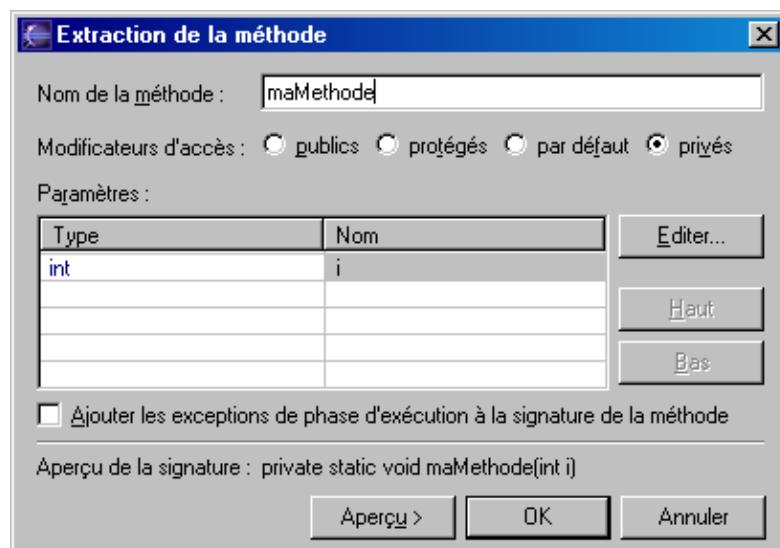
package com.moi.test;

public class MaClasse4 {

    public static void main(String[] args) {
        int i = 0;

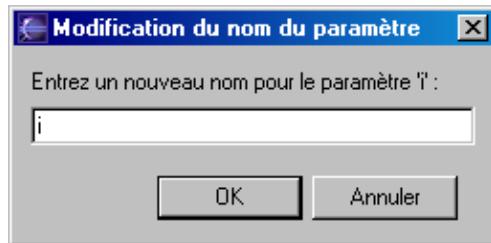
        i += 4;
        System.out.println("i = "+i);
    }
}

```

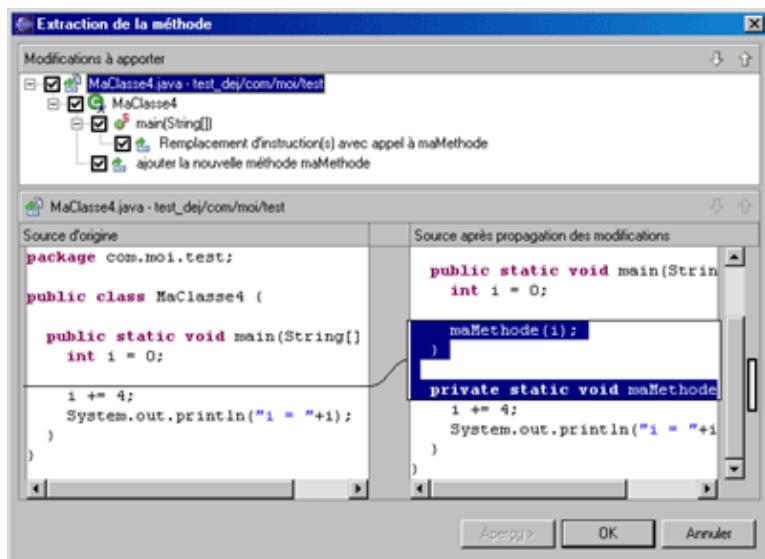


Il faut saisir le nom de la méthode, sélectionner son modificateur d'accès et éventuellement modifier les paramètres puis cliquer sur le bouton "OK".

La liste des variables détectées dans le code est affichée dans la liste des paramètres. Pour modifier le nom d'un des paramètres, il suffit de le sélectionner et de cliquer sur le bouton « Editer... ».



Le bouton « Aperçu » permet de voir et de valider les modifications qui seront apportées.

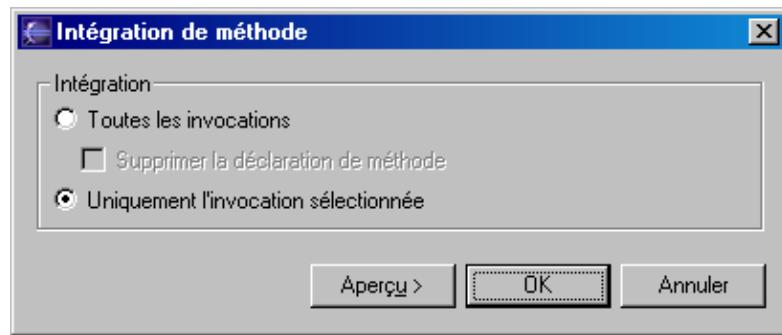


## 9.2. Intégrer

L'intégration permet de remplacer l'appel d'une méthode par le code contenu dans cette méthode.

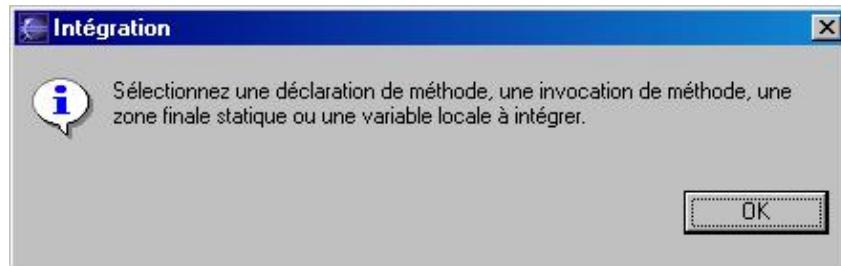
A screenshot of the Eclipse Java editor showing the file "MaClasse3.java". The code defines a class "MaClasse3" with a "main" method that calls "afficherMessage()", and a "test" method that also calls "afficherMessage()". The "afficherMessage()" call is highlighted with a blue selection bar. The editor interface includes tabs for "MaClasse3.java" and "X", and a status bar at the bottom.

Puis il faut utiliser l'option « Propager les modifications/Intégrer... » du menu principal ou contextuel.



En fonction du contexte d'appel, une boîte de dialogue permet de sélectionner la portée des modifications.

Si le code sélectionné ne correspond pas à une méthode, un message d'erreur est affiché.



### 9.3. Renommer

La fonction "Renommer" permet d'attribuer un nouveau nom à une entité présente dans l'espace de travail. Le grand intérêt de réaliser une telle opération via cette fonctionnalité plutôt que de le faire à la "main" et qu'elle automatise la recherche et la mise à jour de toute les références à l'élément dans l'espace de travail.

Les exemples de cette section utilisent les deux classes suivantes :

```
MaClasse11.java X MaClasse12.java
package com.moi.test;

public class MaClasse11 {

    public static void main(String[] args) {
        MaClasse12 mc = new MaClasse12();
        mc.afficherMessage("Bonjour");
    }
}
```

```
MaClasse11.java MaClasse12.java X
package com.moi.test;

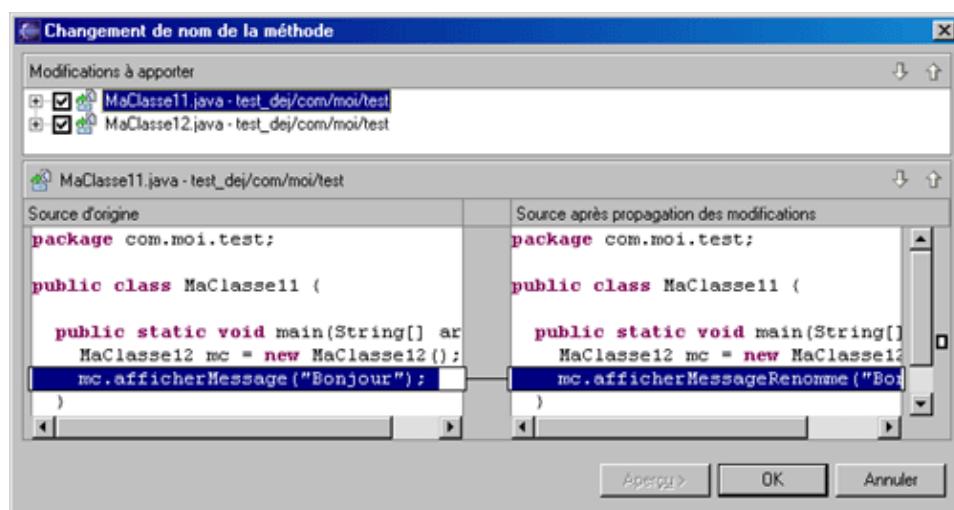
public class MaClasse12 {
    public void afficherMessage(String msg) {
        System.out.println(msg);
    }
}
```

Le renommage concerne l'entité sélectionnée dans la vue "Package" ou "Structure" ou sous le curseur dans l'éditeur de code Java.

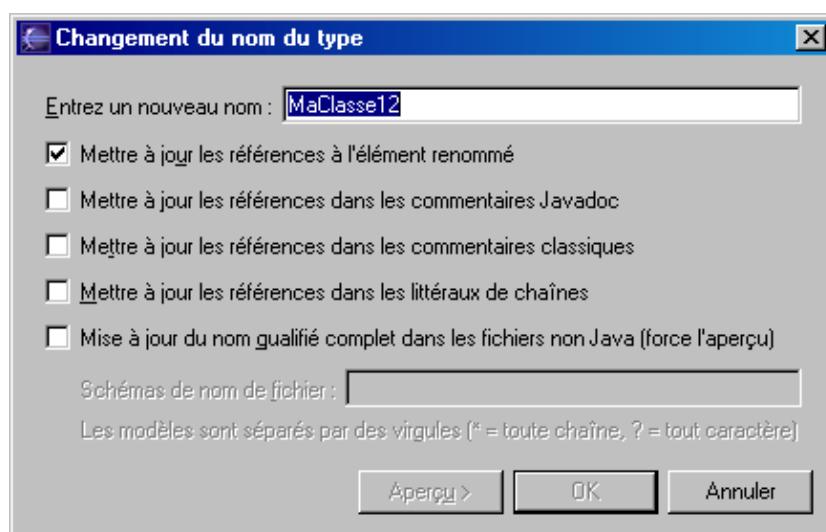
Par exemple, pour renommer une méthode, il suffit de positioner le curseur sur le nom de la déclaration d'une méthode dans le code source et sélectionner l'option « Propager les modifications / Renommer » du menu contextuel.



Il suffit alors de saisir le nouveau nom. Eclipse permet de renommer la méthode dans sa classe de définition mais remplace aussi tous les appels de la méthode contenue dans l'espace de travail.

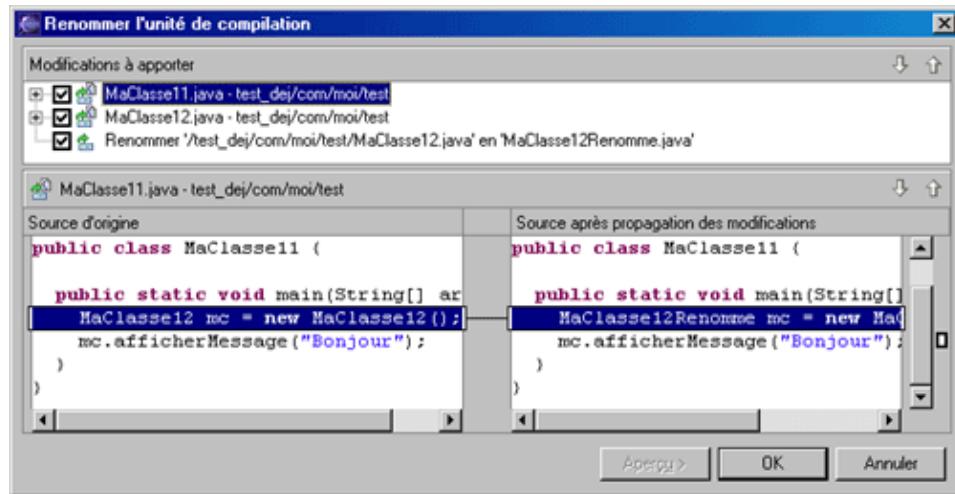


Pour renommer une classe, il y a deux possibilités : selectionner la classe dans la vue « Packages » ou positionner le curseur sur la définition du nom de la classe dans l'éditeur. Puis il faut utiliser l'option « Renommer » du menu contextuel « Propager les modifications ».

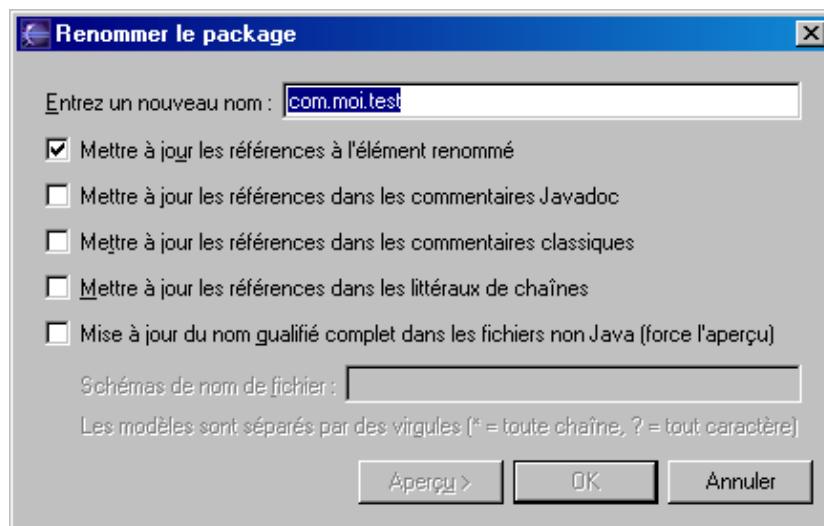


Il est possible de préciser les types d'entités qui doivent être mise à jour.

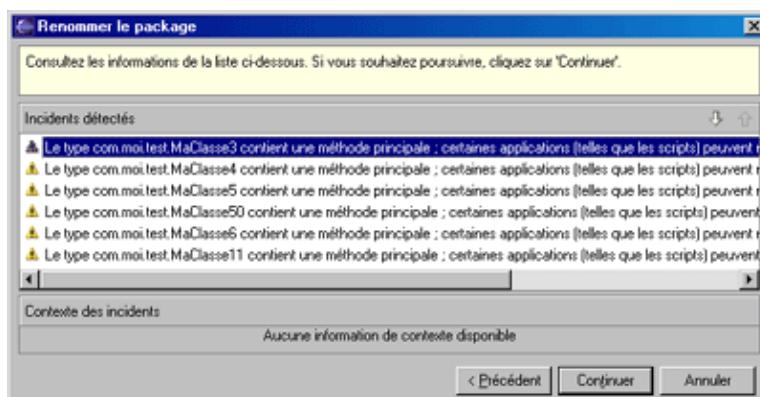
A la validation, Eclipse effectue toutes les modifications nécessaires suite au renommage de la classe, notamment le remplacement à tous les endroits dans l'espace de travail où la classe est utilisée.



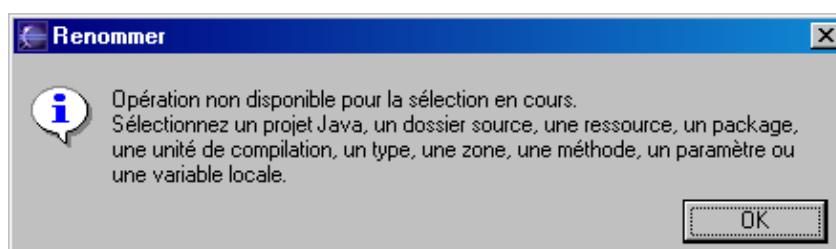
Le principe est le même pour renommer un package.



Si Eclipse détecte des problèmes potentiels lors de l'analyse de la demande, il affiche dans une boîte de dialogue la liste de ces problèmes et demande un confirmation avant de poursuivre les traitements.



Si la sélection ne correspond à aucun élément renommable, un message d'erreur est affiché.





Lors d'une opération de refactoring de renommage, l'assistant propose de mettre en oeuvre un délégué.

Exemple :

```
Exemple :  
  
package fr.jmdoudoux.test;  
  
public class MaClasse1 extends MaSousClasseMere {  
    public MaClasse1(int valeur) {  
        this.valeur = valeur;  
    }  
  
    public int additionner(int val) {  
        return valeur+val;  
    }  
}
```

Sélectionnez une méthode et utiliser l'option « Restructurer / Renommer ... » du menu contextuel.



Saisissez le nouveau nom de la méthode et cochez la case « Conserver la méthode originale en tant que délégué de la méthode renommée ».



Par défaut la case à cocher « Marquer comme étant déprécié » est cochée.

Cliquez sur le bouton « OK »

### Exemple :

```
package fr.jmdoudoux.test;

public class MaClasse1 extends MaSousClasseMere {

    public MaClasse1(int valeur) {
        this.valeur = valeur;
    }

    /**
     * @deprecated Utilisez {@link #additionnerRenomme(int)} à la place
     */
    public int additionner(int val) {
        return additionnerRenomme(val);
    }

    public int additionnerRenomme(int val) {
        return valeur+val;
    }
}
```

La nouvelle est renommée avec le nom fournie et l'ancienne méthode est définie avec l'attribut deprecated : elle appelle simplement la nouvelle méthode.

Cette fonctionnalité permet de garder une compatibilité ascendante.

Elle est mise en oeuvre dans plusieurs outils de refactoring tel que Renommer, Changer la signature de la méthode ou Introduire le paramètre.

Lors d'une opération de refactoring de type Renommer, il est possible de demander le renommage des entités dont le nom est similaire à l'entité renommée.

### Exemple :

```
package fr.jmdoudoux.test ;

public class Fenetre  {

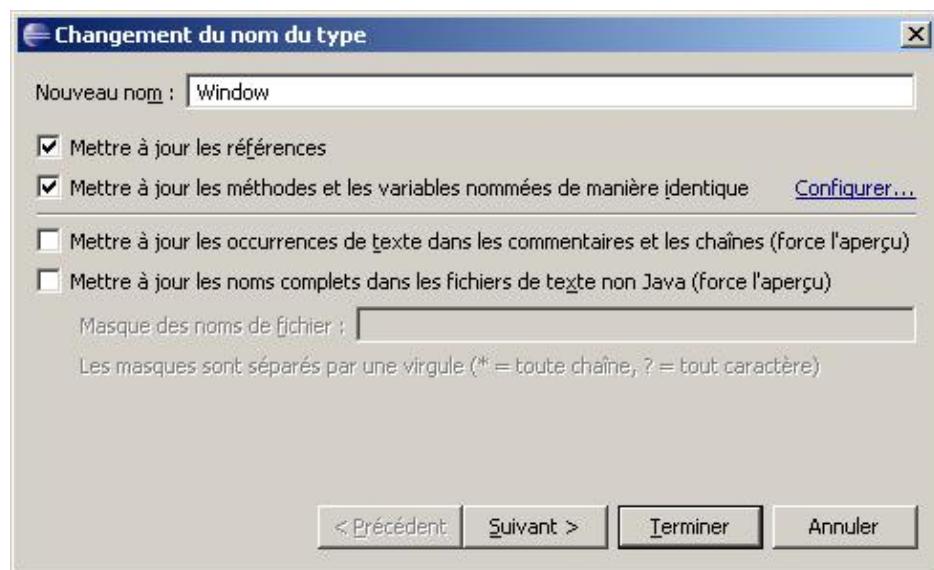
    public Fenetre() {}

    public Fenetre creerFenetre() {
        Fenetre fenetre = new Fenetre();
        return fenetre ;
    }

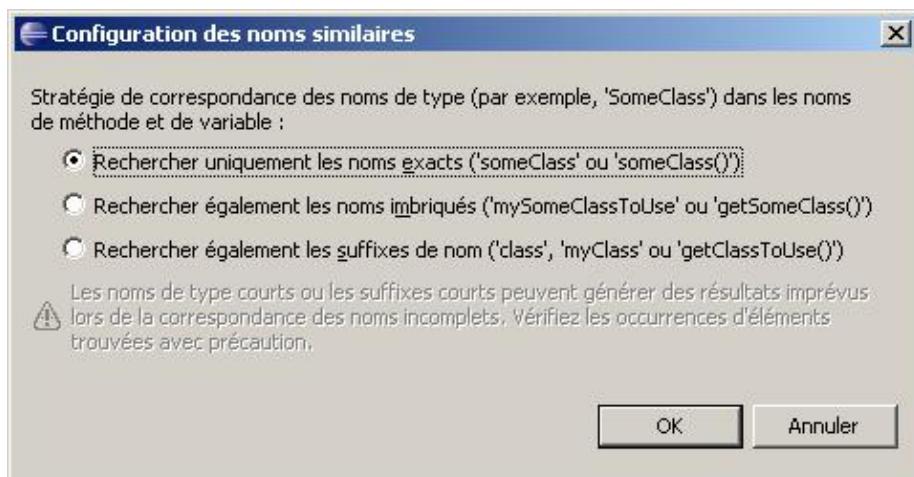
    public void testerFenetre(Fenetre fenetre) {
        System.out.println(" Fenetre =" + fenetre);
    }
}
```

Sélectionnez la classe et utilisez l'option « Restructurer / Renommer » du menu contextuel.

Saisissez le nouveau nom et cochez la case « Mettre à jour les méthodes et les variables nommées de manières identiques ».



Cliquez sur le lien « Configurer ... » pour préciser les paramètres de recherche et de remplacement.



Sélectionnez l'option désirée et cliquez sur le bouton « OK ».

Voici le résultat avec l'option par défaut :

#### Exemple :

```
package fr.jmdoudoux.test ;  
  
public class Window {  
  
    public Window() {}  
  
    public Window creerFenetre () {  
        Window window = new Window();  
        return window;  
    }  
  
    public Window creerFenetreModale () {  
        Window window = new Window();  
        return window;  
    }  
  
    public void testerFenetre (Window window) {  
        System.out.println(" Fenetre =" +window);  
    }  
}
```

Voici le résultat avec l'option « Rechercher également les noms imbriqués » :

Exemple :

```
package fr.jmdoudoux.test;

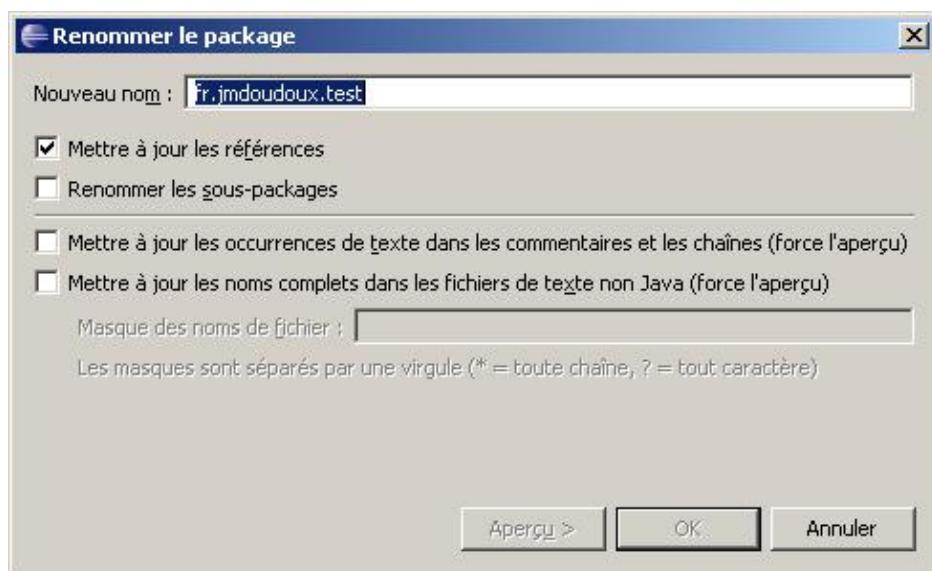
public class Window {

    public Window() {}

    public Window creerWindow () {
        Window window = new Window();
        return window;
    }
    public Window creerWindowModale () {
        Window window = new Window();
        return window;
    }

    public void testerWindow (Window window) {
        System.out.println(" Fenetre =" +window);
    }
}
```

Lors du renommage d'un package, il est possible de demander le renommage des sous packages

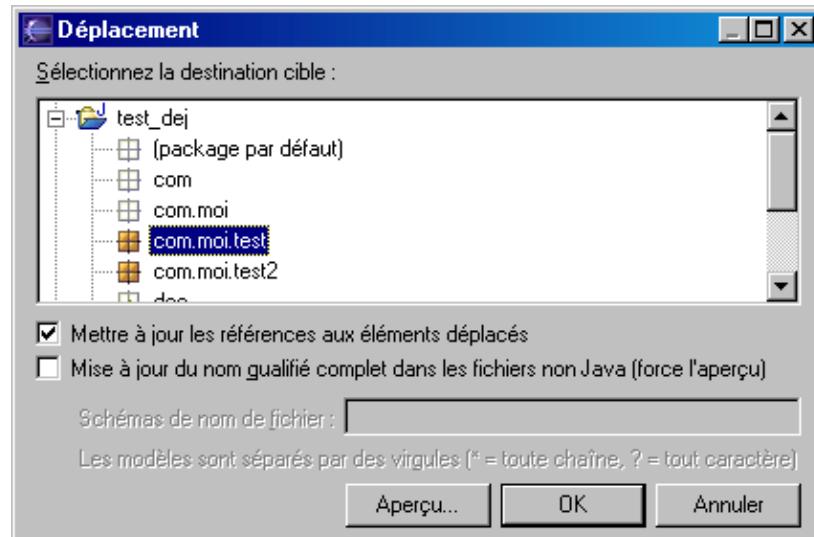


Il suffit de cocher la case « Renommer les sous packages » pour que tous les sous packages soient automatiquement renommés lors de l'opération.

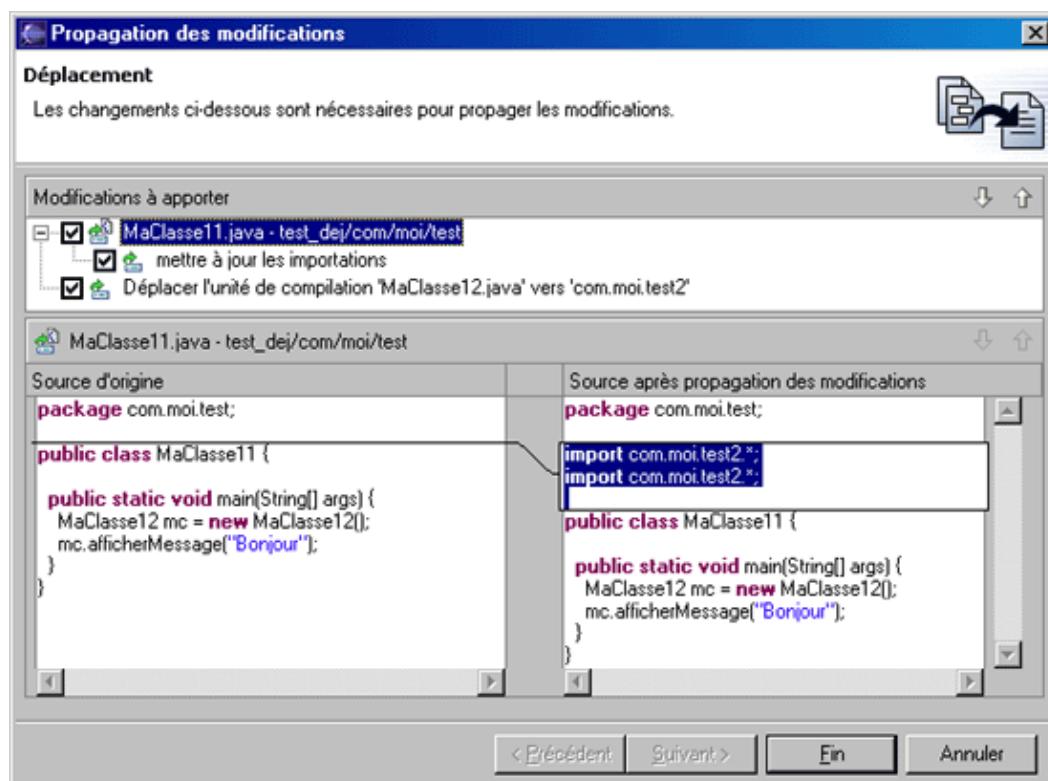
## 9.4. Déplacer

La fonction "Déplacer" permet de déplacer certains éléments précis notamment une classe dans l'espace de travail pour la déplacer dans un autre package. La grande intérêt de réaliser une telle opération via cette fonctionnalité est d'automatiser la recherche et la mise à jour de toutes les références à l'élément dans l'espace de travail.

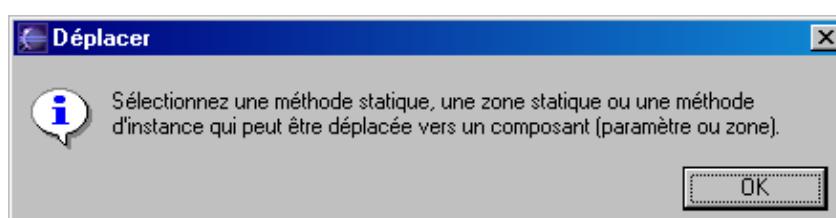
Par exemple, pour déplacer une classe dans un autre package, il faut sélectionner la classe dans la vue "Packages" et de sélectionner l'option "déplacer" du menu "Propager les modifications"



En plus du déplacement de la classe, les clauses import correspondantes sont modifiées dans les classes qui utilisent la classe déplacée.

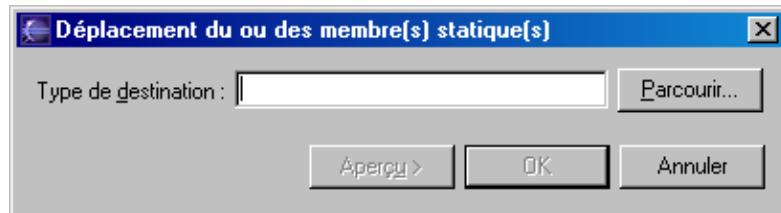


Si le code sélectionné dans l'éditeur de code Java ne correspond pas à une entité concernée par cet action, un message d'erreur est affiché.



Dans l'éditeur de code il faut sélectionner ou mettre le curseur sur un élément statique, avant d'utiliser la fonctionnalité.

Une boîte de dialogue permet de sélectionner la classe dans laquelle la méthode va être transférée.



Le bouton "Parcourir" permet d'ouvrir une boîte de dialogue pour sélectionner la classe de destination.

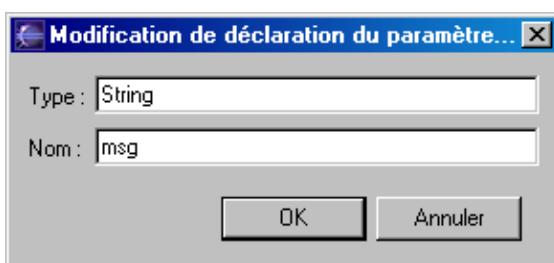
## 9.5. Changer la signature de la méthode

Cette fonction permet de modifier la signature d'une méthode : changer les attributs de visibilités, les paramètres (ajouter/supprimer un paramètre ou modifier le nom, le type ou l'ordre des paramètres) ou le type de retour de la méthode.

Pour utiliser cette fonction, il faut sélectionner dans l'éditeur le nom d'une méthode et appeler la fonction par le menu contextuel. Une boîte de dialogue permet de modifier les éléments qui constituent la signature de la méthode.



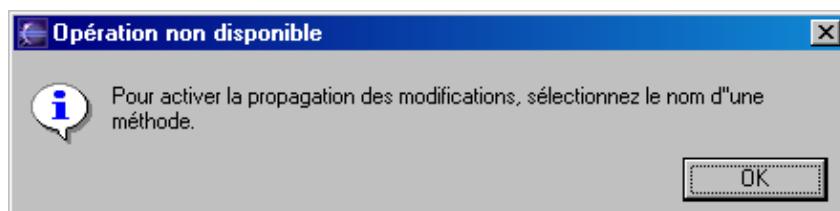
Les actions réalisées par les boutons "Ajouter" et "Supprimer" sont assez explicite. Le bouton "Editer" permet de modifier le paramètre sélectionné.



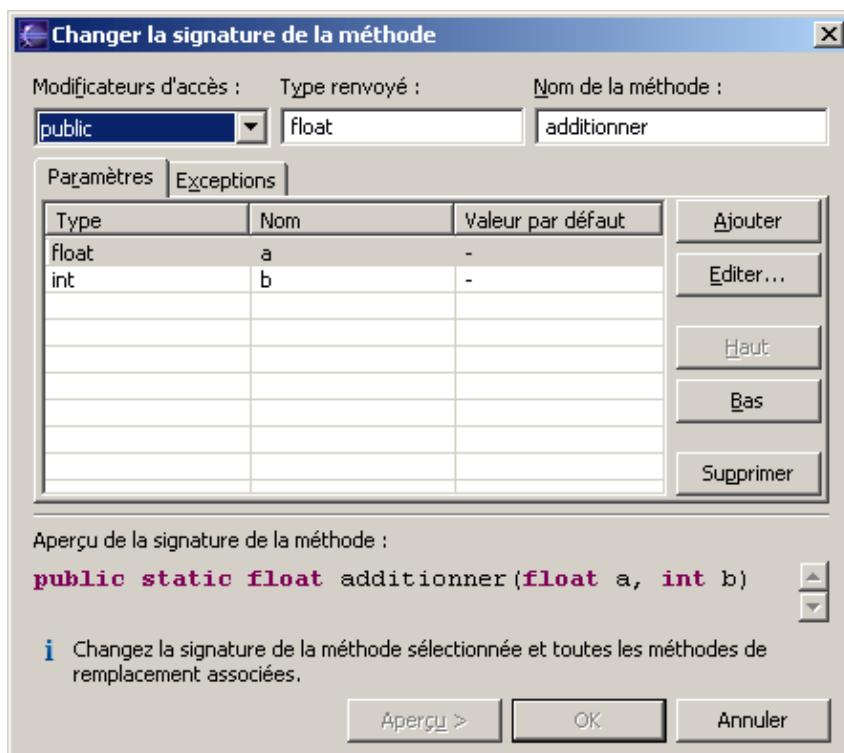
Les boutons "Haut" et "Bas" permettent de modifier l'ordre du paramètre sélectionné.

La valeur par défaut est importante car elle sera utilisée lors de la modification de l'utilisation de la méthode, pour éviter les erreurs de compilation.

Si le code sélectionné dans l'éditeur de code lors de l'appel de la fonction ne correspond pas à une entité concernée par cet action, un message d'erreur est affiché.

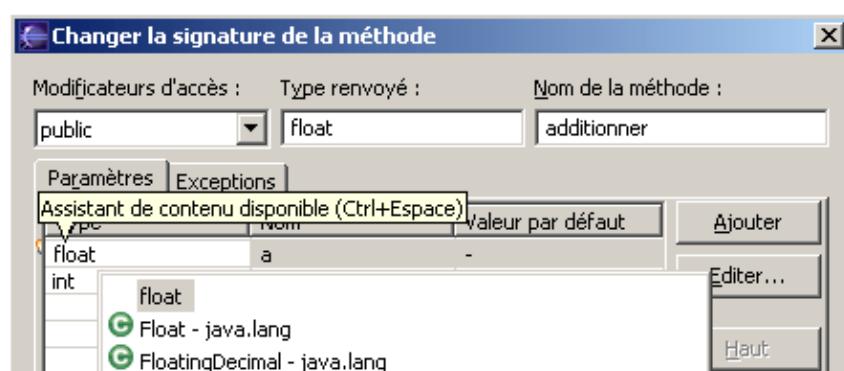


Cette fonction a été améliorée avec plusieurs fonctionnalités.



La boîte de dialogue permet maintenant aussi de renommer le nom de la méthode.

Le type peut être recherché grâce à un assistant en utilisant la combinaison de touche Ctrl+Espace.



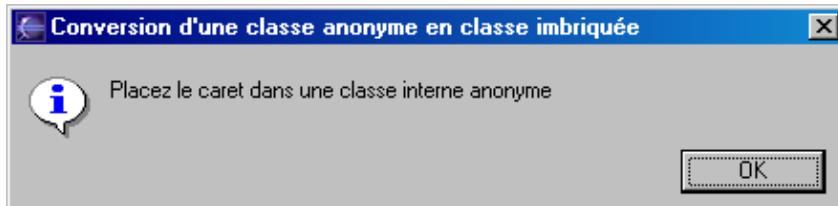
L'onglet "Exceptions" permet de modifier les exceptions propagées par la méthode.

Lors des mises à jours liées à cette fonctionnalité, la documentation Javadoc de la méthode sera mise à jour.

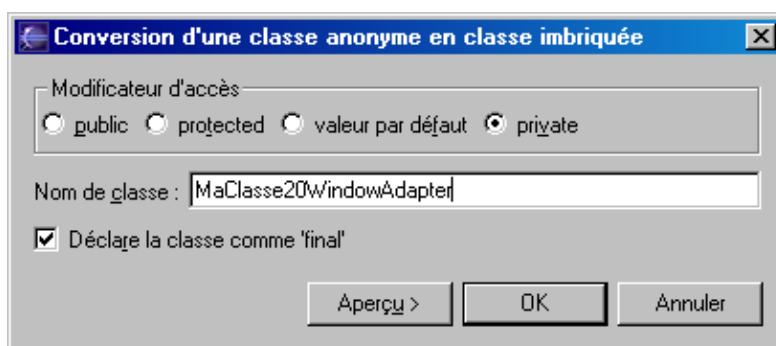
## 9.6. Convertir une classe anonyme en classe imbriquée

Cette fonction permet de transformer une classe anonyme souvent utilisée pour créer des listeners en une classe imbriquée.

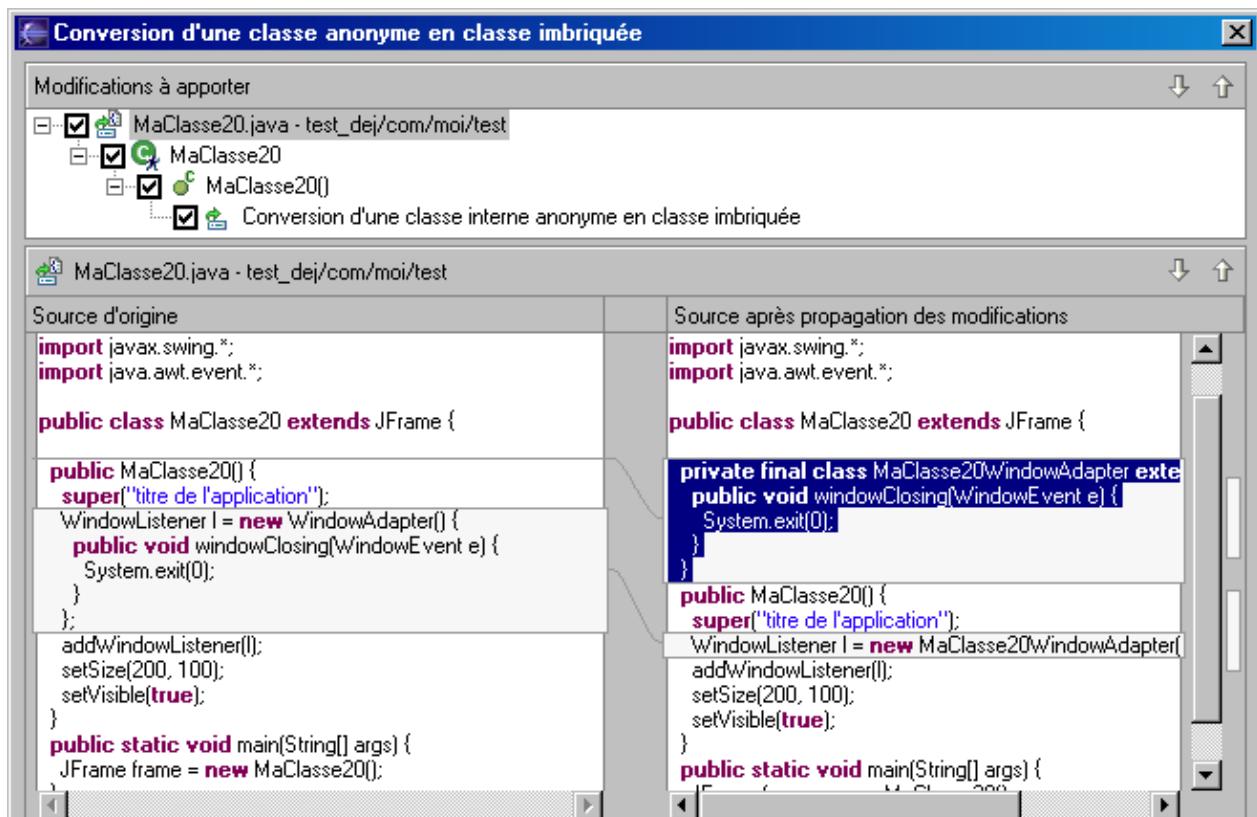
Pour utiliser cette fonction, il faut placer le curseur dans le code de la classe anonyme. Si tel n'est pas le cas lors de l'appel de la fonction, une message d'erreur est affiché.



Une boîte de dialogue permet de saisir les informations concernant la classe qui sera générée.



La nouvelle classe imbriquée est créée en tenant compte des informations saisies.

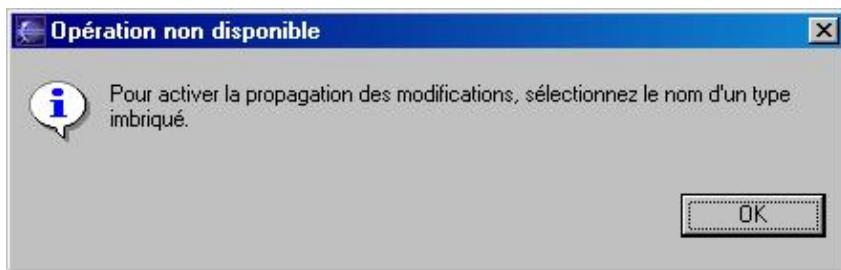


## 9.7. Convertir un type imbriqué au niveau supérieur

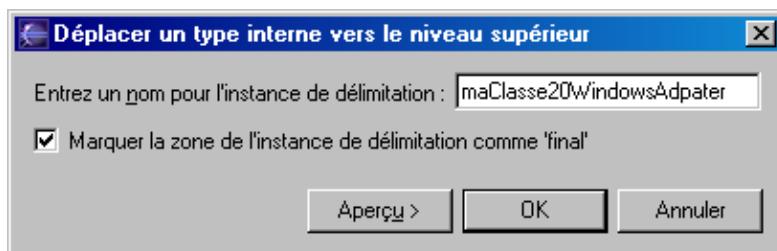
Cette fonction permet de convertir une classe imbriquée en une classe définie dans un fichier particulier.

Pour utiliser cette fonction, il faut sélectionner une classe imbriquée dans la vue "Packages" ou "Structure" ou sélectionner dans l'éditeur de code le nom d'une classe imbriquée.

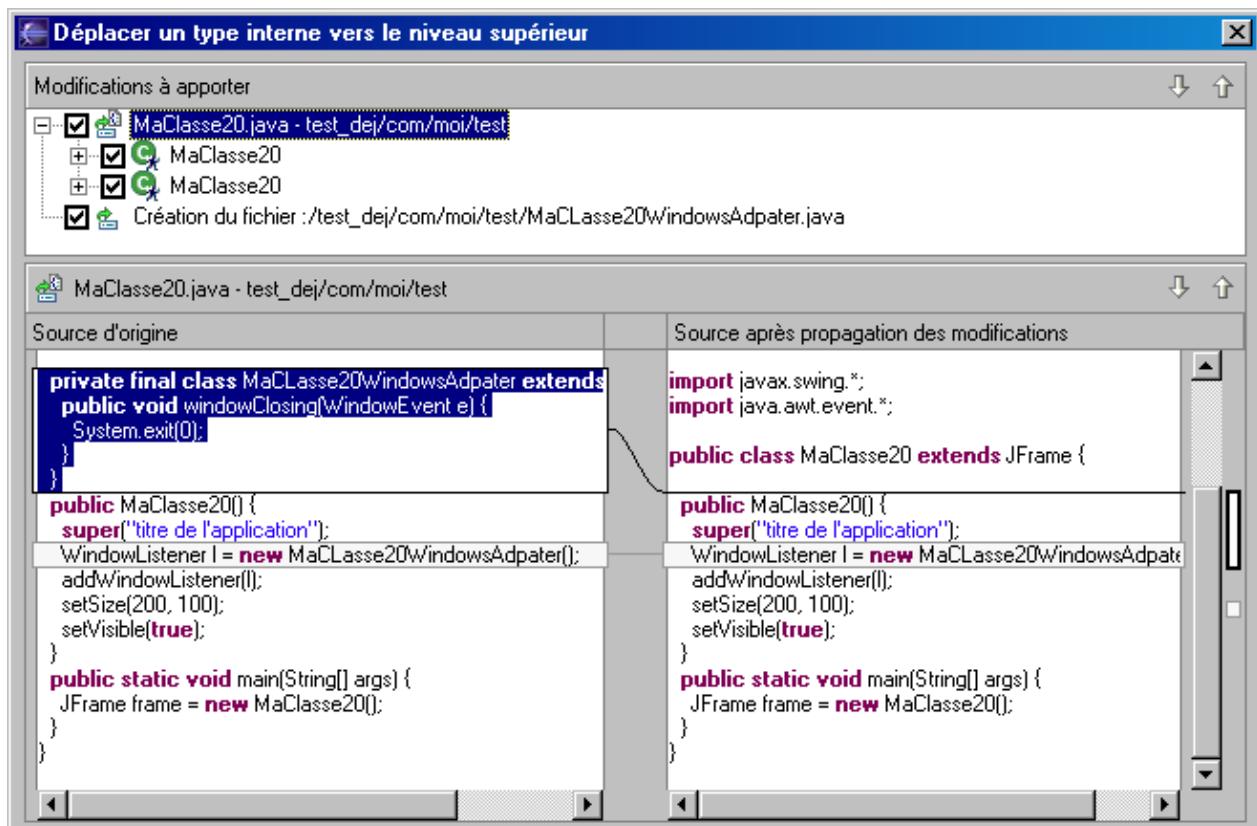
Si le code sélectionné ne correspond pas à une entité concernée par cet action, un message d'erreur est affiché.



Lors de l'appel de la fonction, une boîte de dialogue permet de préciser des informations sur la classe qui va être déplacée.



Voici un exemple de l'aperçu de l'utilisation de cet fonction

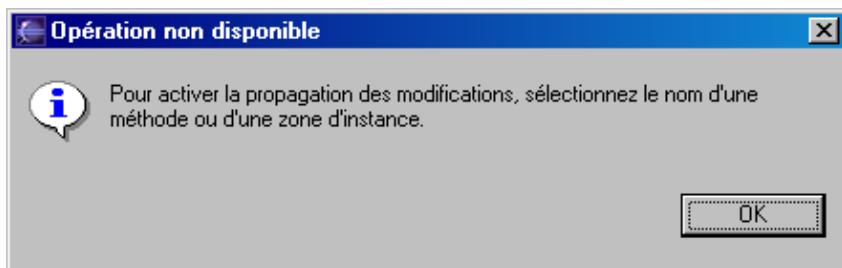


Le nom pour l'instance de délimitation permet de stocker une instance de la classe englobante si la classe avait besoin de l'accès à des membres de l'ancienne classe englobante.

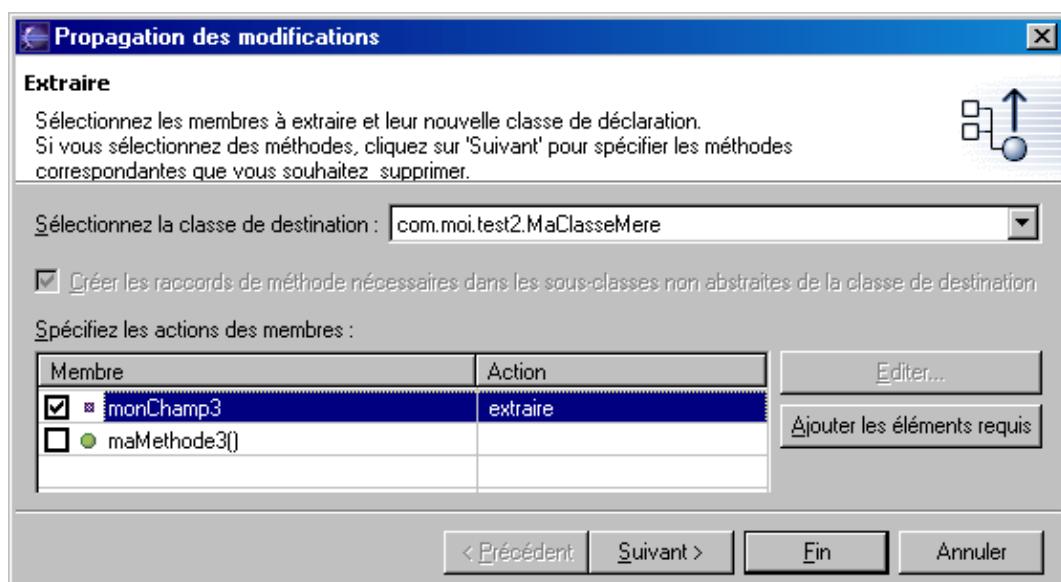
## 9.8. Extraire

Cette fonction permet de déplacer la déclaration d'un membre d'une classe fille dans une de ces classes mères.

Pour utiliser cette fonction, il faut sélectionner un membre d'une classe fille ou positionner le curseur sur un de ces membres sinon un message d'erreur est affiché.



Une boîte de dialogue permet de sélectionner le ou les membres concernés dont celui à partir duquel la fonction a été appelée.

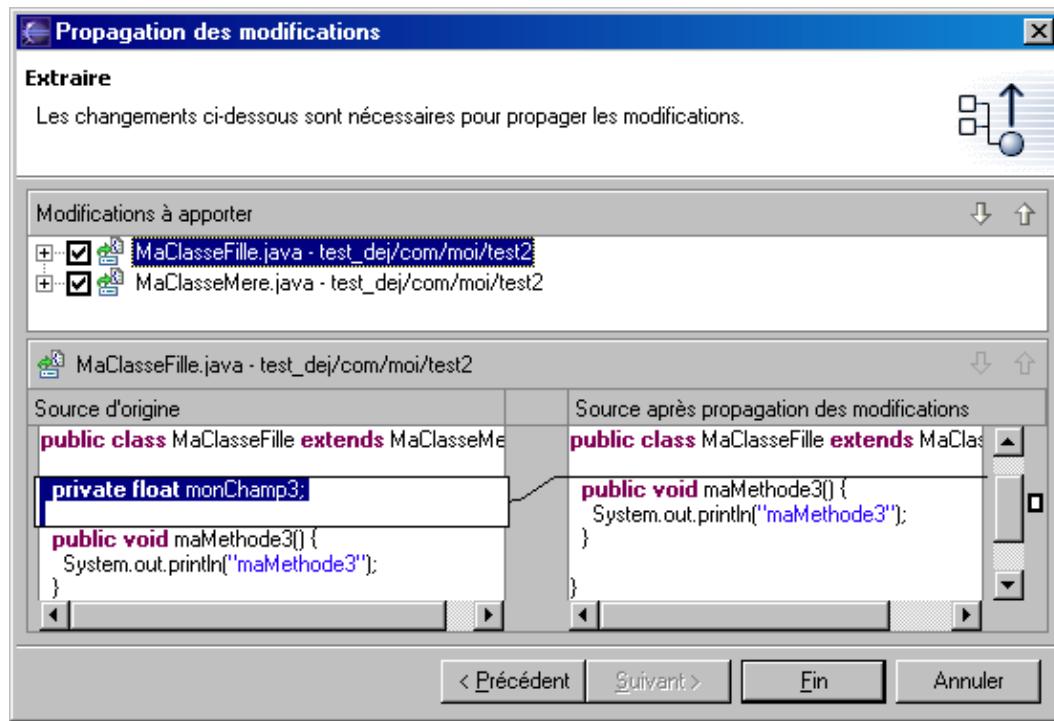


Il suffit de cocher le ou les membres concernées et de sélectionner la classe mère dans la liste déroulante.

Pour les méthodes, le bouton "Editer" permet de sélectionner l'action qui sera réalisée.



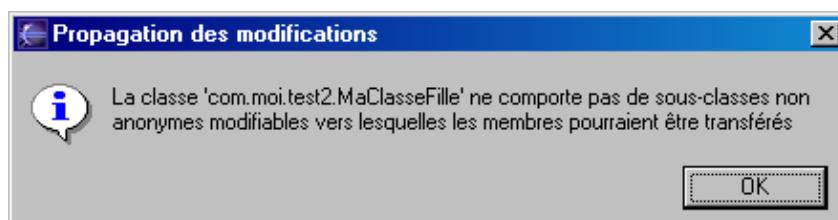
L'action "extraire" permet de déplacer la déclaration et le code de la méthode dans la classe mère sélectionnée. L'action "déclarer abstrait dans la déclaration" permet de déclarer la classe choisie abstraite avec une déclaration de la méthode elle aussi abstraite. Dans ce dernier cas, le code de la méthode est laissé dans la classe fille en tant que rédéfinition.



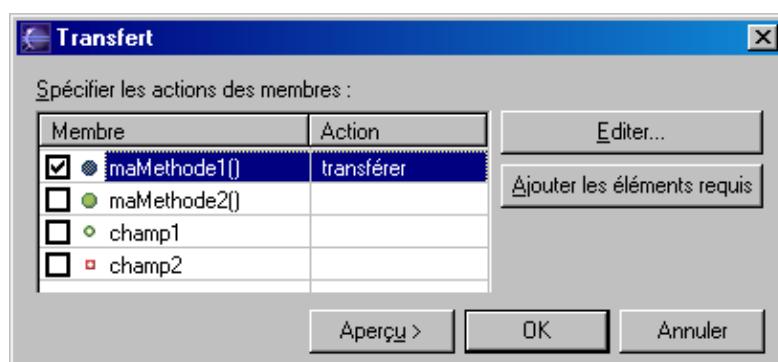
## 9.9. Transferer

Cette fonction permet de transferer une méthode dans les classes filles.

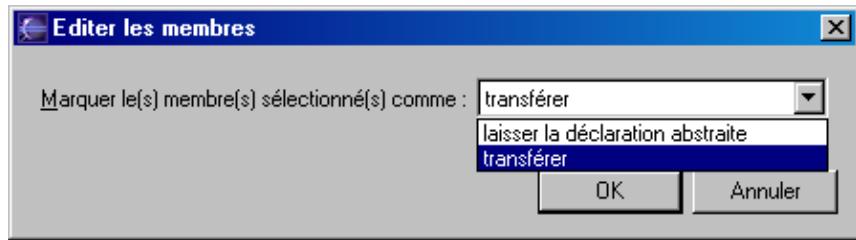
Pour utiliser cette fonction il faut sélectionner un membre d'une classe possèdant au moins une classe fille ou mettre le curseur sur un de ces membres dans l'éditeur de code. Si la fonction est appelée sur une classe qui ne possède pas de classe fille, un message d'erreur est affiché.



Une boîte de dialogue permet de sélectionner le ou les membres concernés par le transfert.



Pour une méthode, le bouton "Editer" permet de sélectionner l'action qui sera réaliser.



L'action "transfert" permet de déplacer la déclaration et le code de la méthode dans la ou les classes filles. L'action "laisser la déclaration abstraite" permet de laisser la déclaration abstraite de la méthode dans la classe mère.

## 9.10. Extraire une interface

Cette fonction permet de définir à postériori une interface à partir d'une ou plusieurs méthodes définies dans une classe. Il faut sélectionner dans le code le nom d'une classe.

```

package com.moi.test;

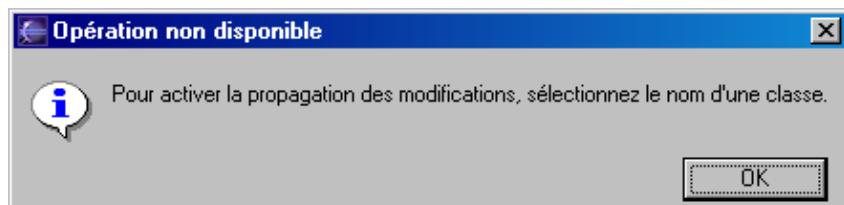
public class MaClasse6 {

    public static void main(String[] args) {
    }

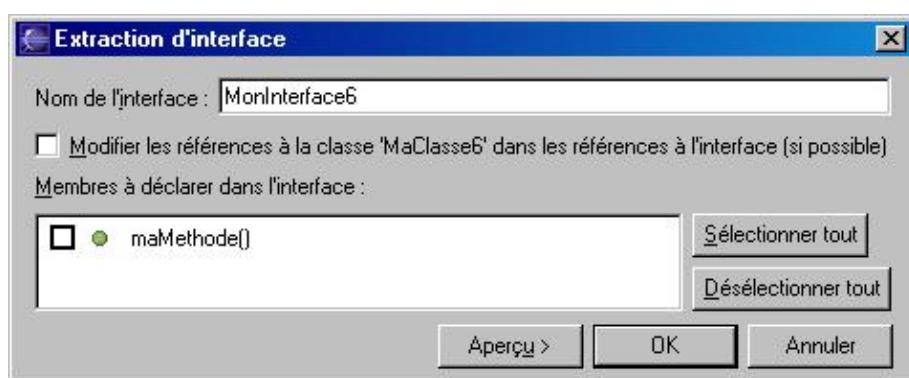
    public int maMethode() {
        return 0;
    }
}

```

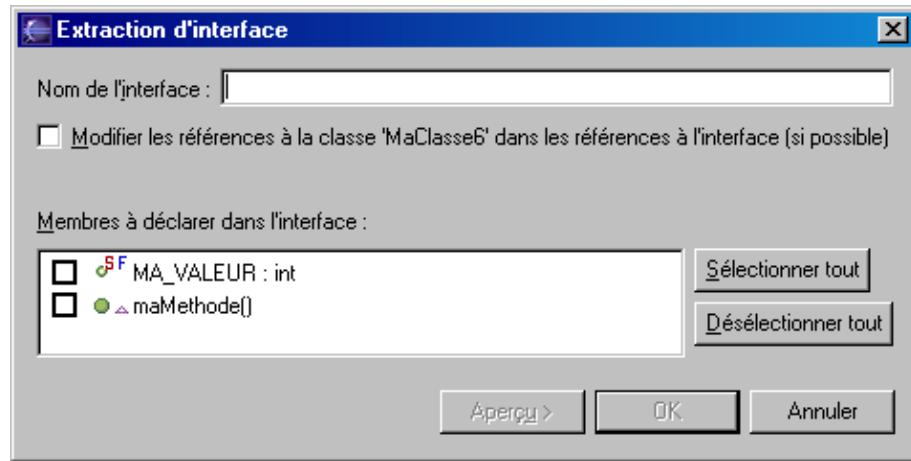
Si la partie sélectionnée ne correspond pas à un nom de classe, un message d'erreur est affiché.



Il suffit ensuite d'utiliser l'option « Extraire une interface » du menu « Propager les modifications ». Une boîte de dialogue permet de préciser le nom de l'interface et de sélectionner les membres à déclarer dans l'interface.



Pour respecter les spécifications du langage Java, les membres qu'il est possible d'intégrer dans l'interface sont des méthodes et des constantes publiques, par exemple :



Le résultat est la création de l'interface et son implémentation par la classe sélectionnée.

```

package com.moi.test;

public interface MonInterface6 {
    public abstract int maMethode();
}

package com.moi.test;

public class MaClasse6 implements MonInterface6 {

    public static void main(String[] args) {}

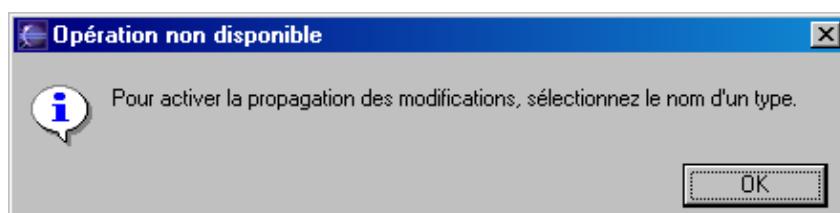
    public int maMethode() {
        return 0;
    }
}

```

## 9.11. Utiliser le supertype si possible

Cette fonction permet de remplacer l'utilisation d'un type par un de ses super types si celui ci en possède.

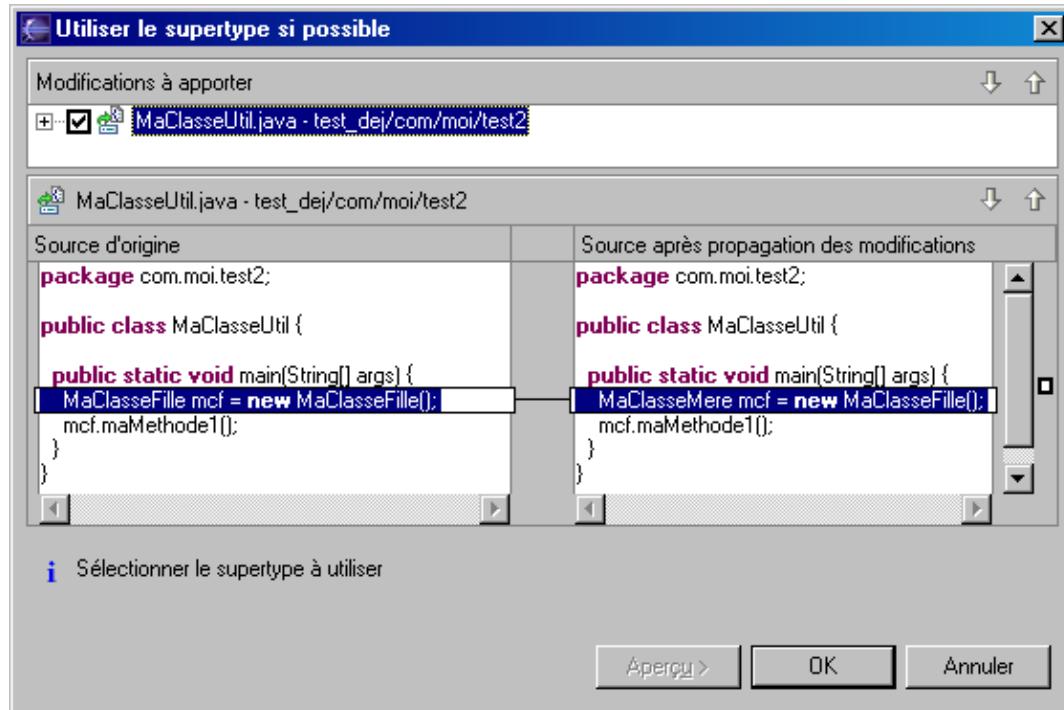
Pour utiliser cette fonction, il faut sélectionner un type dans le code de l'éditeur ou mettre le curseur sur ce dernier sinon un message d'erreur est affiché :



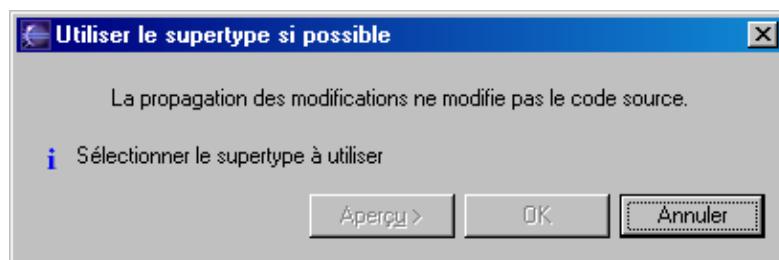
Lors de l'appel de cette fonction, une boîte de dialogue permet de sélectionner une des classes mère du type sélectionné.



La pré-visualisation permet de voir les modifications qui seront apportées dans le code.



Si l'utilisation du super type sélectionné n'est pas possible (par exemple car une méthode utilisée n'est pas définie dans le type sélectionné), aucune modification n'est faite dans le code et l'aperçu affiche le message suivant :



## 9.12. Convertir la variable locale en zone

La fonction « Convertir la variable locale en zone » permet de transformer une variable locale à une méthode en un champ de la classe.

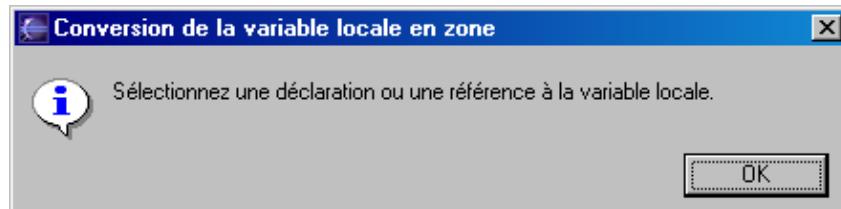
Pour utiliser cette fonction, il faut sélectionner dans le code une variable locale, puis utiliser l'option « Convertir la variable locale en zone » du menu « Propager les modifications ».

```
MaClasse8.java
package com.moi.test;

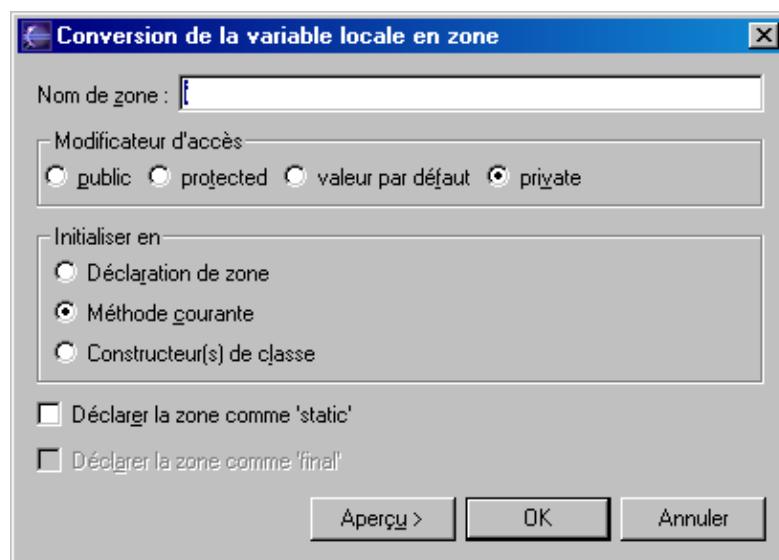
public class MaClasse8 {

    public void maMethode() {
        int i = 0;
    }
}
```

Si la sélection ne correspond pas à une variable locale, un message d'erreur est affiché.



Une boîte de dialogue permet de préciser le nom du champ, le modificateur d'accès et l'endroit dans le code où le champ sera initialisé.



Le résultat de l'exécution de la fonction est le suivant :

```
MaClasse8.java
package com.moi.test;

public class MaClasse8 {

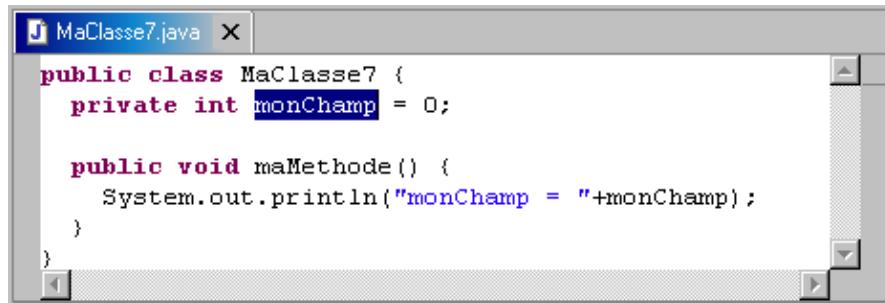
    private int i;

    public void maMethode() {
        i = 0;
    }
}
```

## 9.13. Encapsuler la zone

Cette fonction permet d'encapsuler un attribut en générant un getter et éventuellement un setter.

Il faut sélectionner dans le code un champ de la classe.

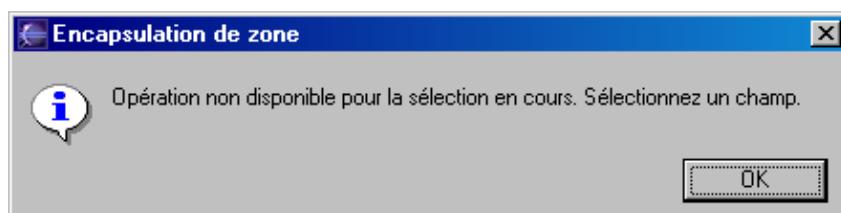


```
public class MaClasse7 {
    private int monChamp = 0;

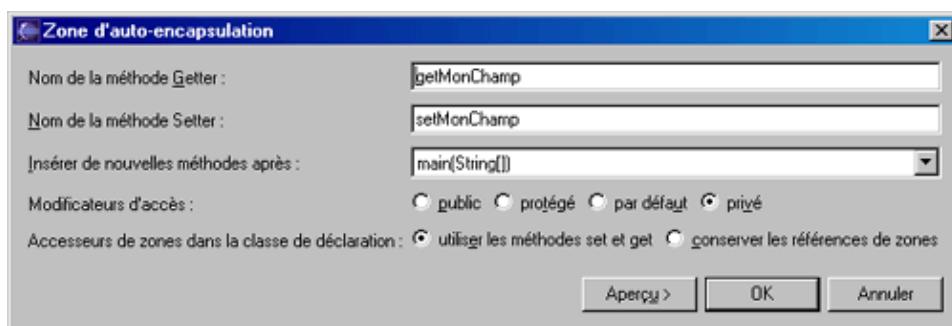
    public void maMethode() {
        System.out.println("monChamp = "+monChamp);
    }
}
```

Il faut ensuite utiliser l'option « Encapsuler la zone » du menu « Propager les modifications ».

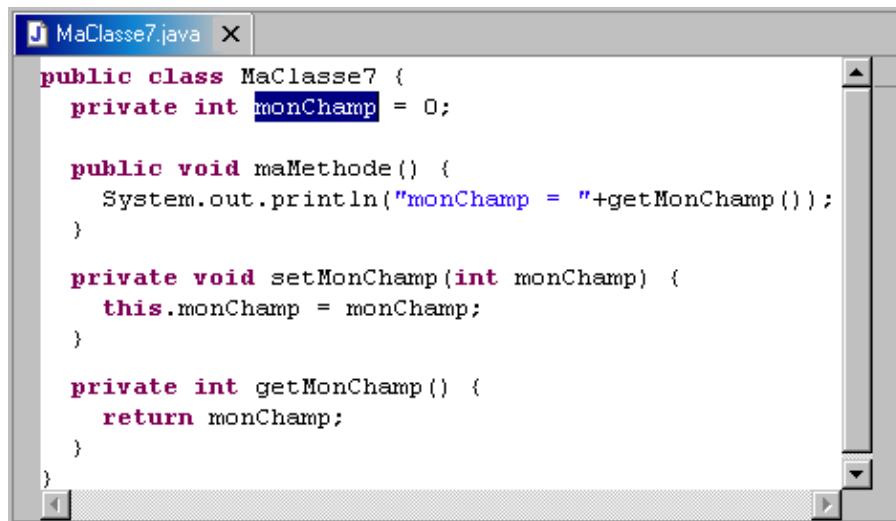
Si la sélection dans le code ne correspond pas à un champ de la classe, un message d'erreur est affiché.



Une boîte de dialogue permet de préciser les options pour la génération du getter et du setter



Le résultat de l'exécution génère le getter et le setter et remplace l'utilisation du champ par ces méthodes.



```
public class MaClasse7 {
    private int monChamp = 0;

    public void maMethode() {
        System.out.println("monChamp = "+getMonChamp());
    }

    private void setMonChamp(int monChamp) {
        this.monChamp = monChamp;
    }

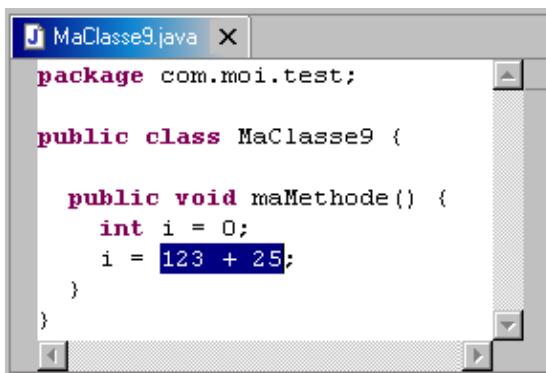
    private int getMonChamp() {
        return monChamp;
    }
}
```

En sélectionnant l'option « conserver les références de zones », ces méthodes ne sont pas utilisées dans la classe où le champ est utilisé.

## 9.14. Extraire la variable locale

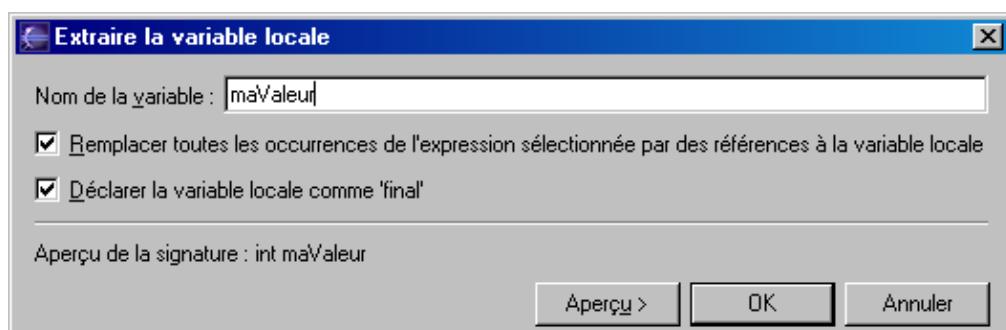
Cette fonction permet de définir une constante à partir d'une expression.

Il faut sélectionner une expression dans le code source.



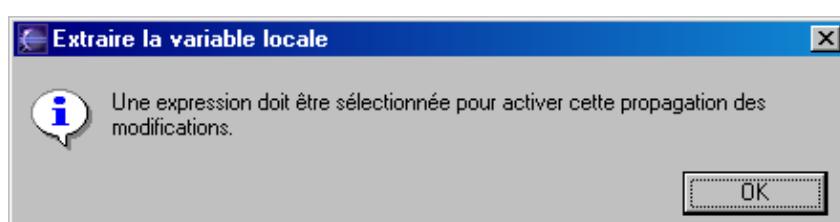
A screenshot of the Eclipse Java editor showing a Java file named "MaClasse9.java". The code contains a public class definition with a method "maMethode" that contains an assignment statement: "i = 123 + 25;". This line is highlighted with a blue selection bar.

Puis utiliser l'option « Extraire la variable locale » du menu « Propager les modifications ».



Une boîte de dialogue permet de saisir le nom de la variable, de préciser si toutes les occurrences doivent être modifiées et si la variable doit être une constante.

Si la sélection dans le code source est incorrecte, un message d'erreur est affiché.



Le résultat de l'exécution de cette fonction est le suivant :

```
MaClasse9.java
package com.moi.test;

public class MaClasse9 {

    public void maMethode() {
        int i = 0;
        final int maValeur = 123 + 25;
        i = maValeur;
    }
}
```

## 9.15. Extraire une constante

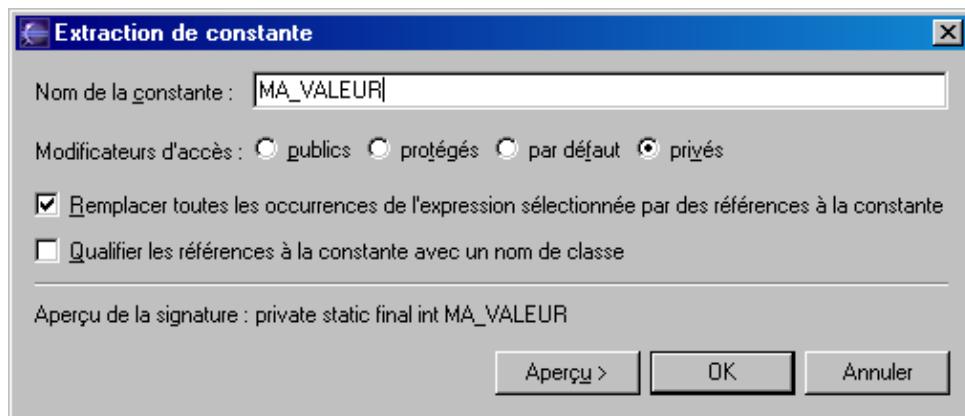
Cette fonction permet de définir une constante à partir d'une valeur en dur utilisée dans le code. Il suffit de sélectionner cette valeur dans le code source.

```
MaClasse5.java
package com.moi.test;

public class MaClasse5 {

    public static void main(String[] args) {
        int i = 0;
        i +=123;
    }
}
```

Puis d'utiliser l'option « Extraire une constante ... » du menu « Propager les modifications ». Une boîte de dialogue apparaît pour préciser le nom de la constante à créer et son modificateur d'accès.



Il est possible de préciser si toutes les occurrences de la valeur doivent être remplacées dans le code source.

Il est aussi possible de préciser l'utilisation de la qualification de la constante avec le nom de la classe dans laquelle elle est définie.



```
J MaClasse5.java X
package com.moi.test;

public class MaClasse5 {

    private static final int MA_VALEUR = 123;
    public static void main(String[] args) {
        int i = 0;
        i +=MA_VALEUR;
    }
}
```

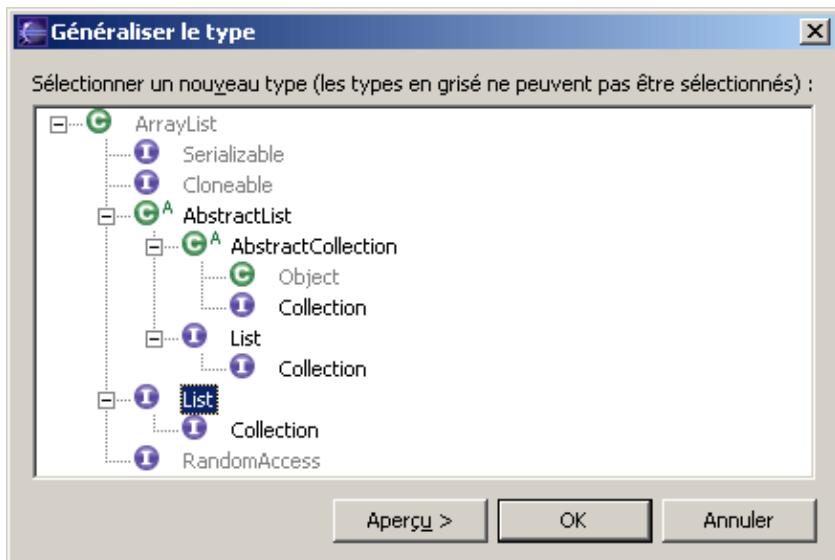
## 9.16. Généraliser le type

Cette fonction permet de remplacer le type d'une variable par un de ces super types. Il faut positionner le curseur sur la déclaration d'une variable, d'un champ, d'un paramètre ou d'une valeur de retour.

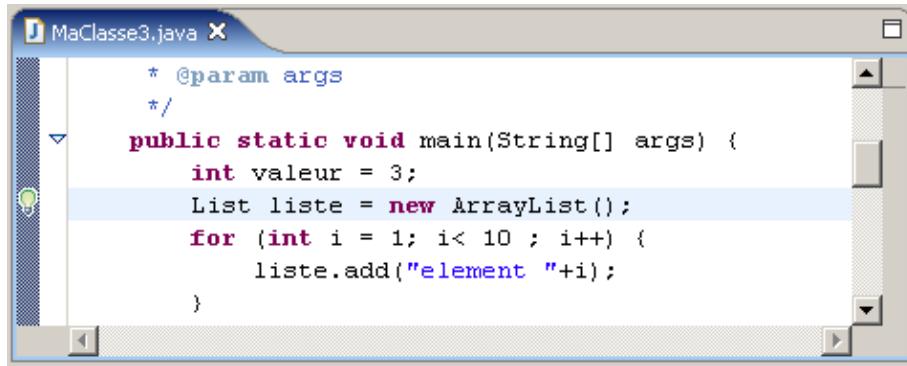


```
J MaClasse3.java X
*/
public static void main(String[] args) {
    int valeur = 3;
    ArrayList liste = new ArrayList();
    for (int i = 1; i< 10 ; i++) {
        liste.add("element "+i);
    }
}
```

Une boîte de dialogue affiche la liste des super types.



Il suffit de sélectionner le type désiré et de cliquer sur le bouton « Ok » pour que le code soit mise à jour.



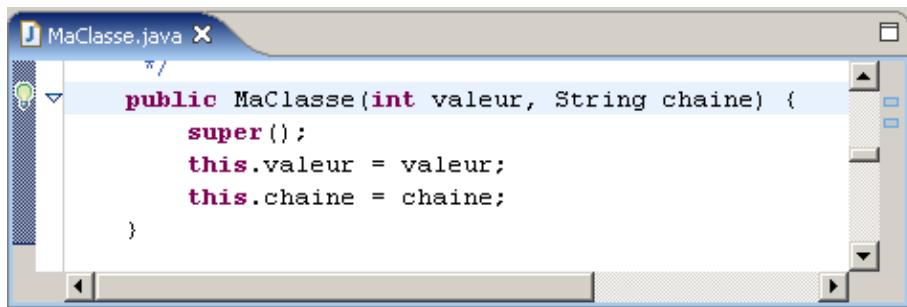
```
* @param args
*/
public static void main(String[] args) {
    int valeur = 3;
    List liste = new ArrayList();
    for (int i = 1; i < 10 ; i++) {
        liste.add("element "+i);
    }
}
```

Cette fonctionnalité est très utile car elle facilite la mise en oeuvre d'une bonne pratique de programmation consistant à utiliser un super type comme type d'une variable lorsque cela est possible.

## 9.17. Introduire une fabrique



Cette fonction permet d'obliger l'utilisation d'une fabrique pour créer une occurrence de la classe.

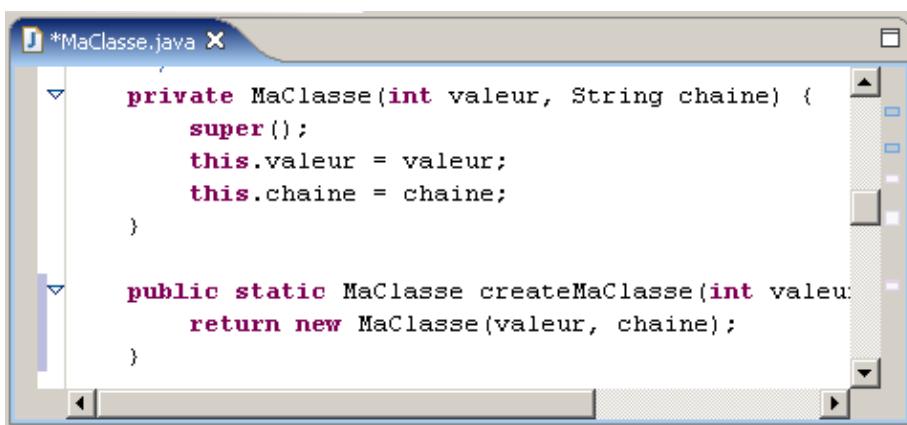


```
/*
public MaClasse(int valeur, String chaine) {
    super();
    this.valeur = valeur;
    this.chaine = chaine;
}
```

Il faut positionner le curseur sur un constructeur puis appeler la fonction.



Le code est modifié en créant la méthode de fabrication et en rendant le constructeur privé.



```
private MaClasse(int valeur, String chaine) {
    super();
    this.valeur = valeur;
    this.chaine = chaine;
}

public static MaClasse createMaClasse(int valeur, String chaine) {
    return new MaClasse(valeur, chaine);
}
```

## 9.18. Introduire un paramètre

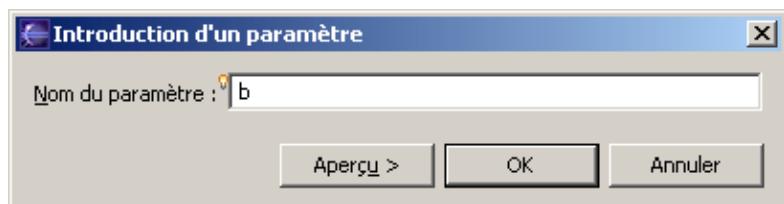


Cette fonction permet de remplacer une expression utilisée dans une méthode par un paramètre fourni à cette méthode.

Il faut positionner le curseur sur une expression littérale et appeler la fonction.

```
public static float additionner(float a) {  
    return a+1;  
}
```

Une boîte de dialogue permet de saisie le nom du paramètres



En cliquant sur le bouton « Ok », le paramètre est ajouté dans la signature de la méthode et la valeur littérale est remplacée par le paramètre.

```
public static float additionner(float a, int b)  
    return a+b;  
}
```

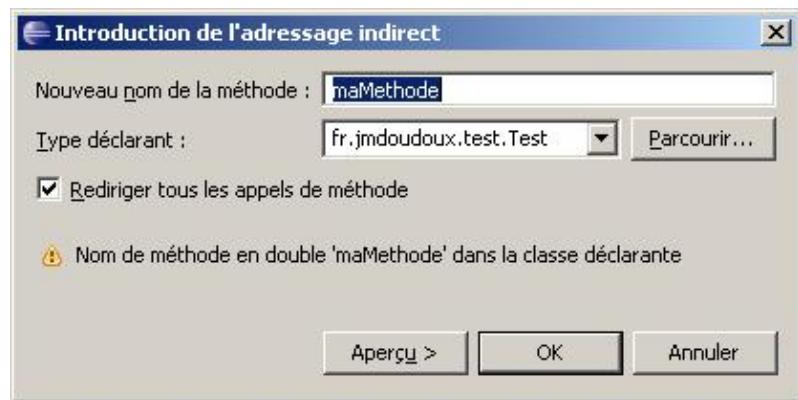
## 9.19. Introduction de l'adressage indirect



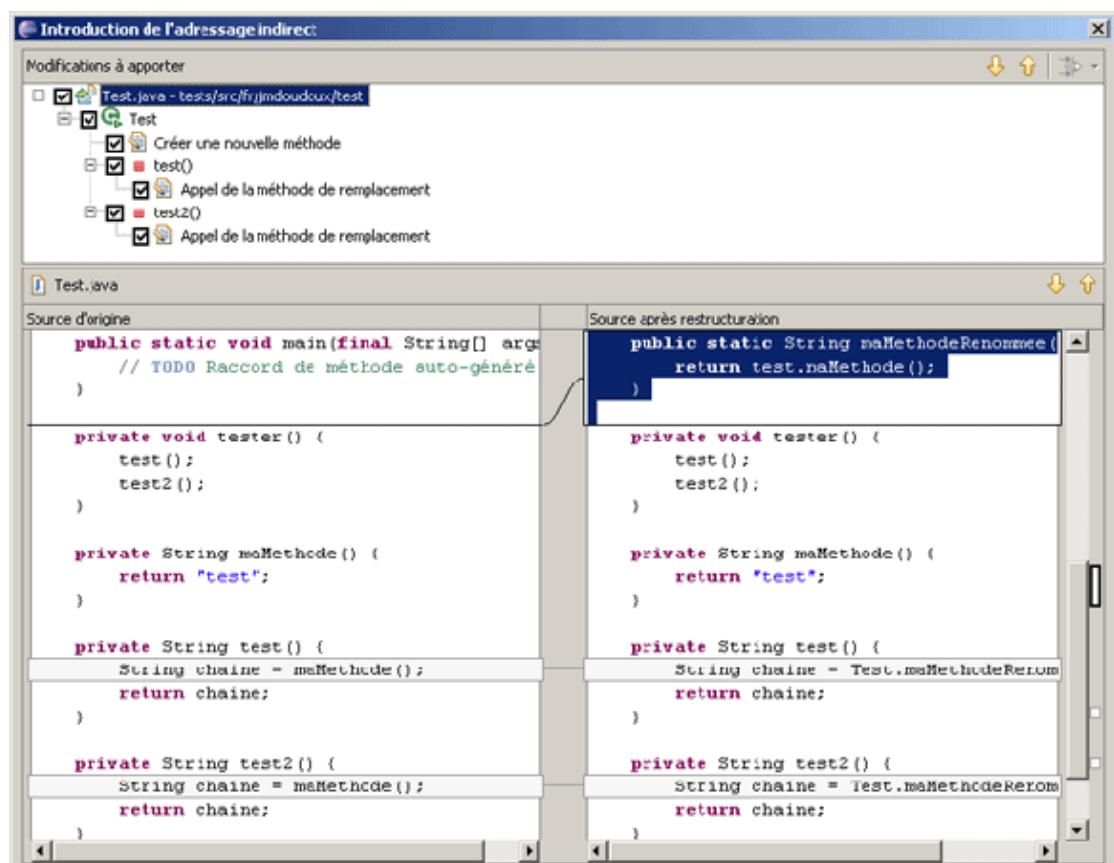
Ce refactoring permet de rediriger l'appel d'une méthode vers une nouvelle méthode qui appelle elle-même la méthode courante. Tous les appels actuels sont remplacés par un appel à la nouvelle méthode.

```
private String maMethode() {  
    return "test";  
}  
  
private String test() {  
    String chaine = maMethode();  
    return chaine;  
}  
  
private String test2() {  
    String chaine = maMethode();  
    return chaine;  
}
```

Sélectionnez une méthode dans le code et utilisez l'option « Restructurer / Introduction de l'adressage indirect ... »



Saisissez le nouveau de la méthode et cliquez sur le bouton « OK ».



Cliquez sur le bouton « OK » pour exécuter le refactoring .

```
public static String maMethodeRenommee(Test test) {
    return test.maMethode();
}

private void tester() {
    test();
    test2();
}

private String maMethode() {
    return "test";
}

private String test() {
    String chaine = Test.maMethodeRenommee(this);
    return chaine;
}

private String test2() {
    String chaine = Test.maMethodeRenommee(this);
    return chaine;
}
```

## 9.20. Extraire une super classe



Ce refactoring permet de créer une super classe à partir de plusieurs classes sous réserve que ces classes possèdent des membres communs.

Les classes utilisées dans cette section sont les suivantes :

Exemple :

```
package fr.jmdoudoux.test;

public class MaClasse1 extends MaClasseMere{
    private int valeur = 100;

    public MaClasse1(int valeur) {
        this.valeur = valeur;
    }

    public void afficher() {
        System.out.println("valeur="+valeur);
    }

    public int additionner(int val) {
        return valeur+val;
    }
}

package fr.jmdoudoux.test;

public class MaClasse2 extends MaClasseMere{
    private int valeur = 200;

    public MaClasse2(int valeur) {
        this.valeur = valeur;
    }
}
```

```

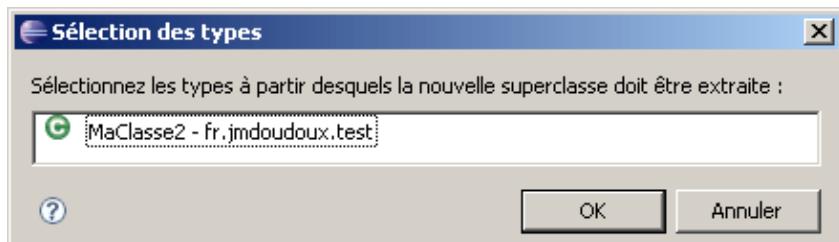
public void afficher() {
    System.out.println("valeur="+valeur);
}

public int Calculer() {
    return valeur + 10;
}
}

```

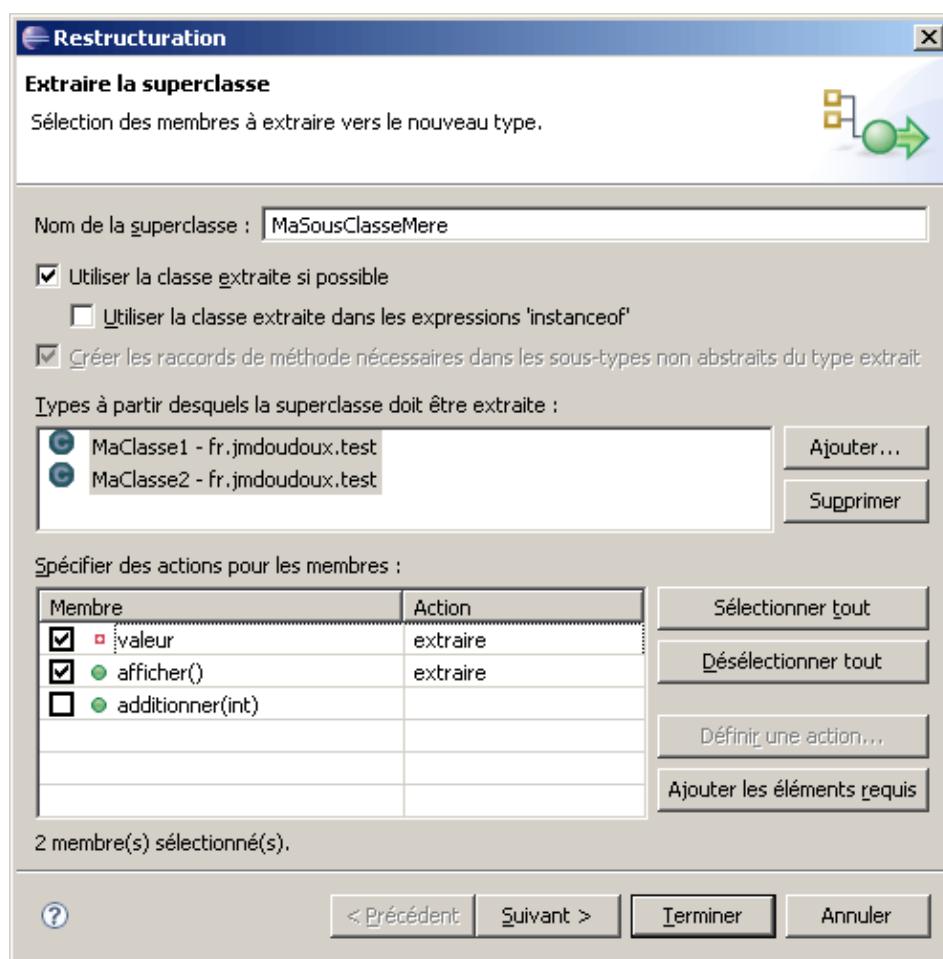
Sélectionnez la première classe (MaClasse1 par exemple) et utilisez l'option « Restructurer / Extraire une super classe » du menu contextuel

Saisissez le nom de la super classe à créer puis cliquez sur le bouton « Ajouter »

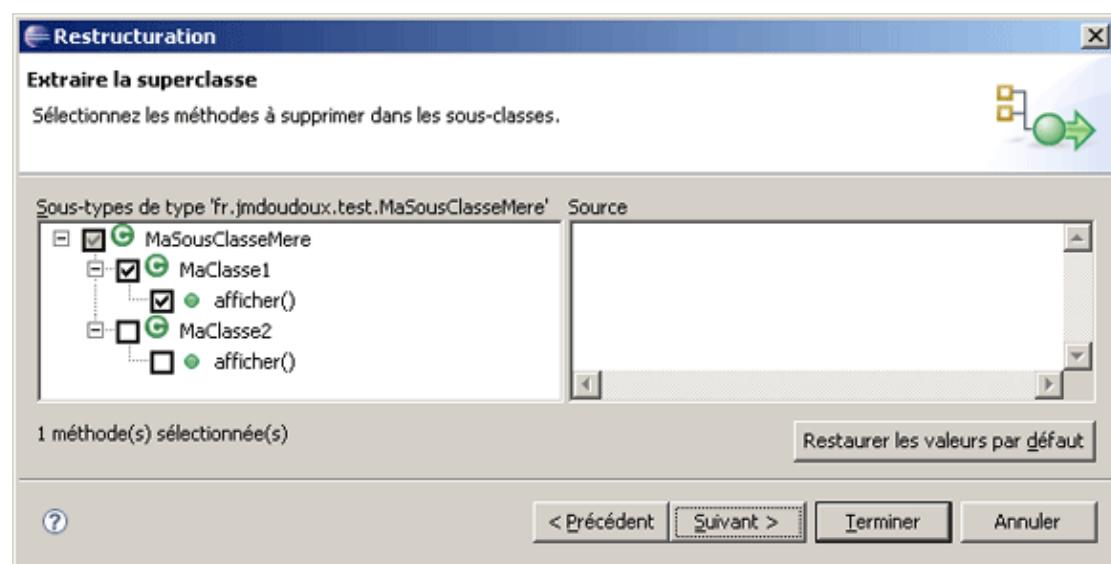


Sélectionnez la ou les classes concernées par ce refactoring et cliquez sur le bouton « OK »

Cochez les membres de la première classe à intégrer dans la classe mère.



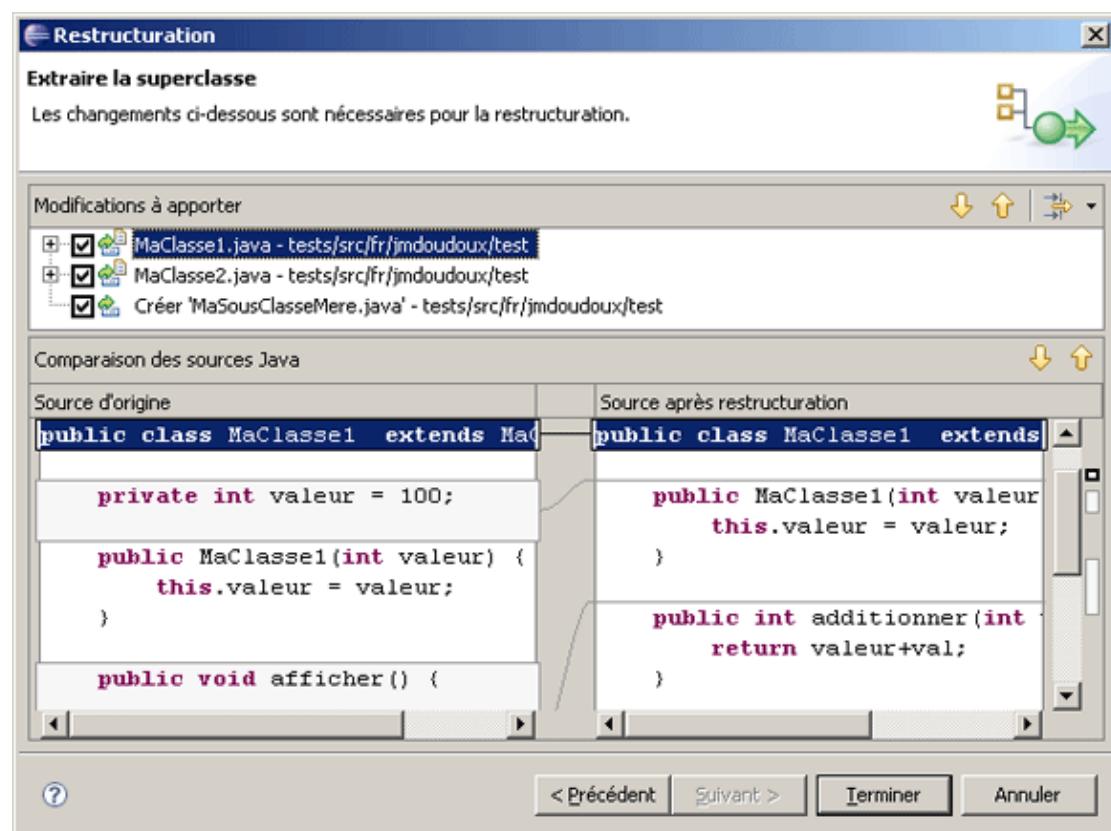
Cliquez sur le bouton « Suivant »



Cette page de l'assistant permet de sélectionner les méthodes à supprimer dans les classes filles issues du refactoring .

La partie source permet de voir le code source de la méthode et ainsi de sélectionner les méthodes concernées par le refactoring.

Cliquez sur le bouton « Suivant » pour avoir un aperçu des modifications



Cliquez sur le bouton « Terminer »

La super classe est créée :

Exemple :

```
package fr.jmdoudoux.test;
```

```

public class MaSousClasseMere extends MaClasseMere {
    protected int valeur = 100;

    public MaSousClasseMere() {
        super();
    }

    public void afficher() {
        System.out.println("valeur=" + valeur);
    }
}

```

et les classes filles sont modifiées

#### Exemple :

```

package fr.jmdoudoux.test;

public class MaClasse1 extends MaSousClasseMere {

    public MaClasse1(int valeur) {
        this.valeur = valeur;
    }

    public int additionner(int val) {
        return valeur+val;
    }
}

package fr.jmdoudoux.test;

public class MaClasse2 extends MaSousClasseMere {

    public MaClasse2(int valeur) {
        this.valeur = valeur;
    }

    public int Calculer() {
        return valeur + 10;
    }
}

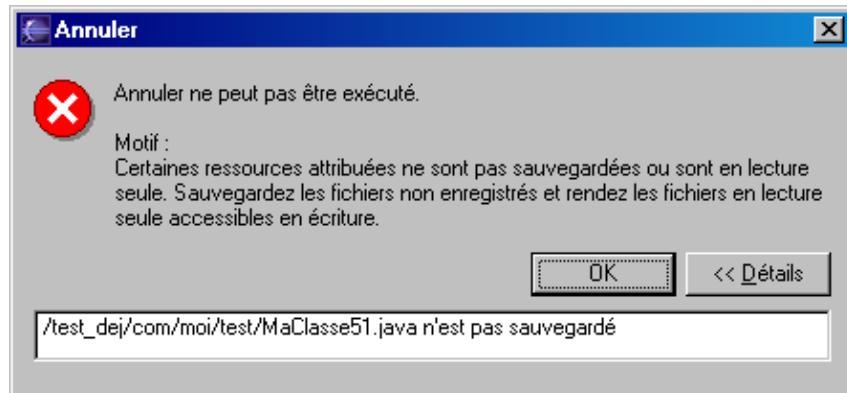
```

## 9.21. Annuler ou refaire une opération de refactoring

Le menu "Propager les modifications" possède deux options "Annuler" et "Rétablir" qui ont un rôle similaire aux deux options du même nom dans le menu "Edition". L'utilisation de ces deux options est cependant dédiées à l'annulation ou le rétablissement des modifications opérées par la dernière opération de refactoring.

Ces deux options ne sont plus disponibles dès que des modifications supplémentaires sont effectués et sauvegardées dans au moins un des fichiers modifiés par l'opération de refactoring.

Si un des fichiers modifiés par l'opération de refactoring est modifié sans être sauvégarde avant l'utilisation d'une de ces deux options, un message d'erreur est affiché.



## 9.22. L'historique de restructuration

Sélectionnez la restructuration et cliquez sur le bouton « Supprimer ».



Cliquez sur le bouton « Oui »

## 10. Ant et Eclipse

# Chapitre 10

Ant est un projet du groupe Apache-Jakarta. Son but est de fournir un outil écrit en Java pour permettre la construction d'applications (compilation, exécution de tâches post et pré compilation ...). Ces processus de construction sont très importants car ils permettent d'automatiser des opérations répétitives tout au long du cycle de vie de l'application (développement, tests, recette, mise en production ...).

Ant pourrait être comparé au célèbre outil make sous Unix. Il a été développé pour fournir un outil de construction indépendant de toute plate-forme car écrit avec et pour Java. Il repose sur un fichier de configuration XML qui décrit les différentes tâches qui devront être exécutées par l'outil. Ant fournit un certain nombre de tâches courantes qui sont codées sous forme d'objets développés en Java.

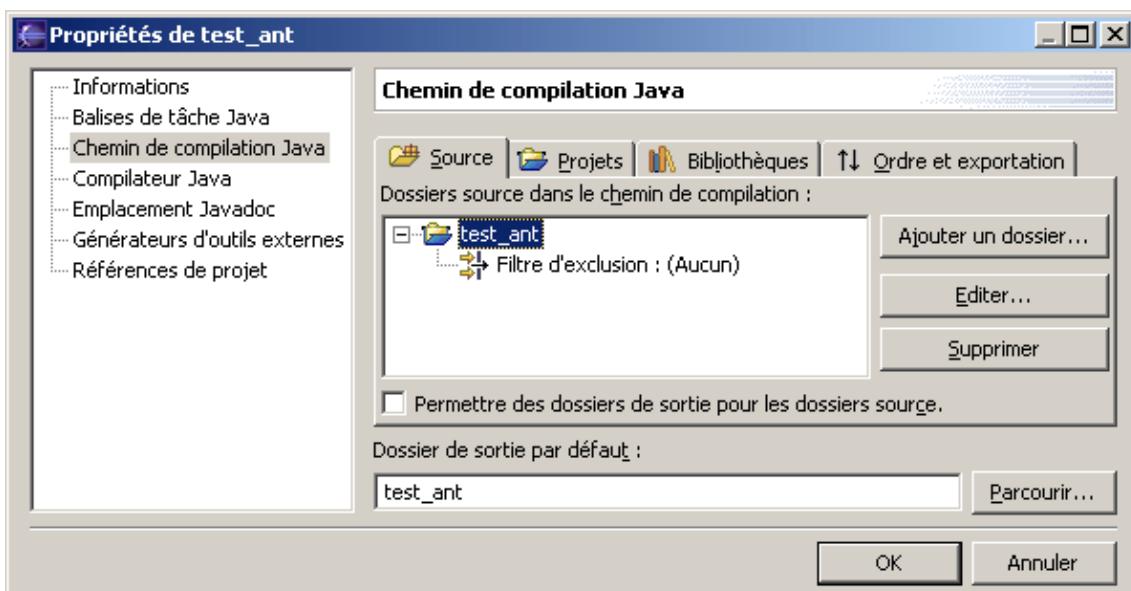
Le fichier de configuration contient un ensemble de cibles (target). Chaque cible contient une ou plusieurs tâches. Chaque cible peut avoir une dépendance envers une ou plusieurs autres cibles pour pouvoir être exécutée.

Pour obtenir plus de détails sur l'utilisation de Ant, il est possible de consulter la documentation de la version courante à l'url suivante : <http://jakarta.apache.org/ant/manual/index.html>

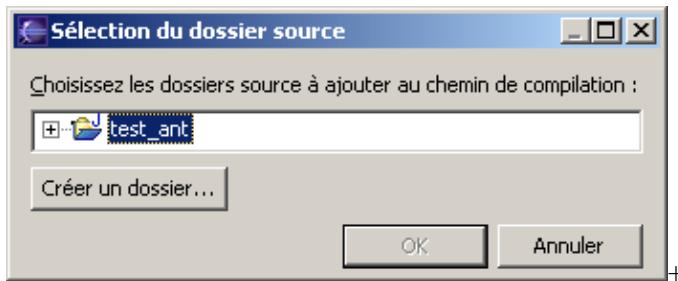
### 10.1. Structure du projet

Pour utiliser Ant, il faut organiser différemment la structure du projet si celui-ci utilise la structure par défaut d'un projet Java. Par défaut, les fichiers sources .java et leurs homologues compilés .class sont dans le même répertoire à la racine du projet.

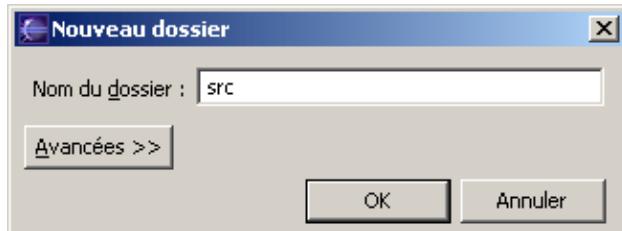
Il faut mettre les sources dans un répertoire et les fichiers .class dans un autre. Ce changement peut être fait dans l'onglet "Source" des propriétés du projet.



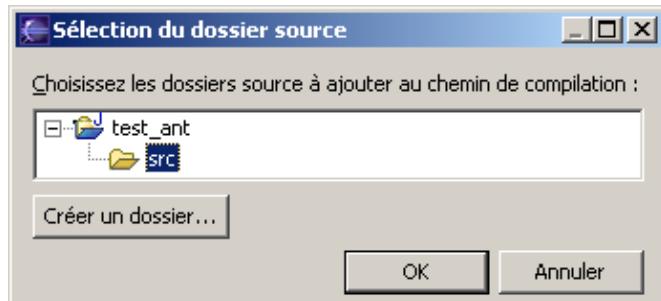
Cliquez sur le bouton "Ajouter un dossier".



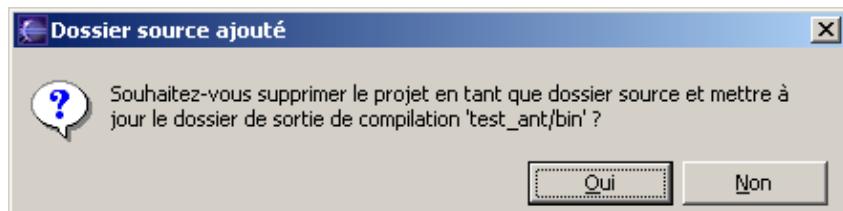
Cliquez sur le bouton "Créer un dossier".



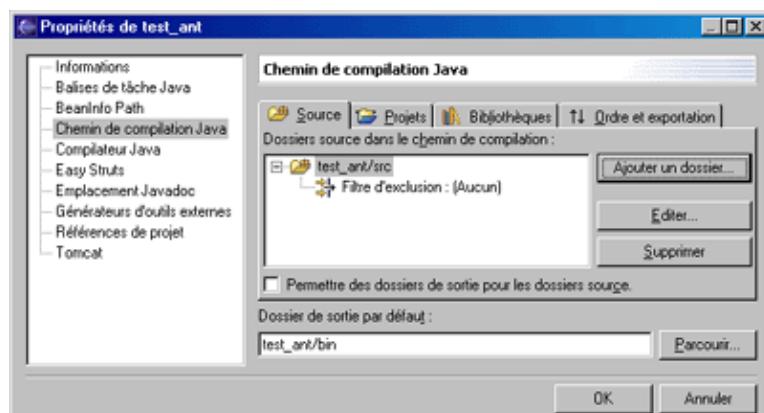
Il faut saisir le nom du répertoire qui va contenir les sources (par exemple src) et cliquer sur le bouton "OK".



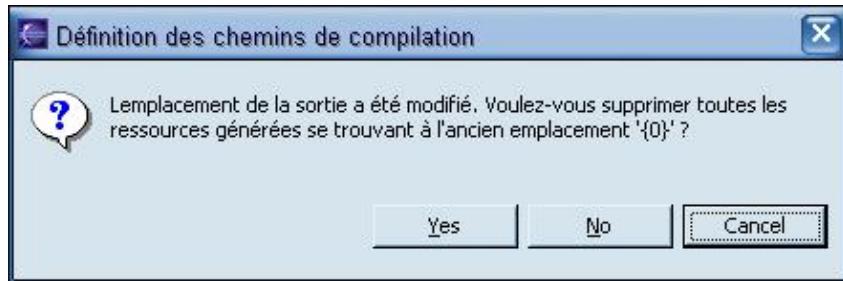
Cliquez sur le bouton "OK".



En cliquant sur "Oui", Eclipse va automatiquement créer un répertoire bin qui va contenir le résultat des compilations des sources.



Cliquez sur le bouton "OK".



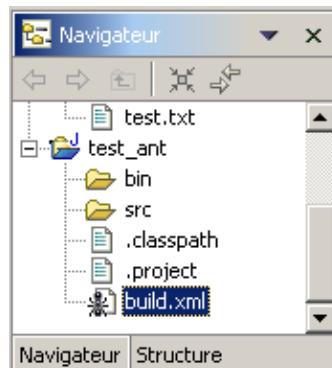
Cliquez sur le bouton "Yes".

Il faut ensuite déplacer les fichiers .java existant dans le répertoire src en effectuant un copier/coller dans la vue "Navigateur".

Il faut ensuite créer un répertoire build contenant deux sous dossiers : lib et doc. Ces dossiers vont contenir respectivement les fichiers de distribution générés (.jar, .war, .ear selon le type d'application) et la documentation des classes au format Javadoc.

## 10.2. Création du fichier build.xml

Les ordres de générations sont fournis à Ant sous la forme d'un fichier au format xml nommé build.xml. Il faut créer ce nouveau fichier à la racine du projet.



Le fichier est automatiquement reconnu comme étant un fichier de configuration pour Ant : une icône particulière contenant une fourmi est associée au fichier.

Il suffit ensuite d'éditer le fichier pour insérer les paramètres d'exécution.

### Exemple : afficher un message

```
<?xml version="1.0"?>
<project name="TestAnt1" default="bonjour">
    <target name="bonjour">
        <echo message="Premier test avec Ant! "/>
    </target>
</project>
```

Un éditeur particulier est dédié à l'édition du fichier build.xml de Ant. Il propose notamment un achèvement du code pour les tags en utilisant la combinaison de touches Ctrl + espace.

```

<?xml version="1.0"?>
<project name="TestAnt1" default="bonjour">
<target name="bonjour">
<echo message="Premier test avec Ant!"/>
</target>
</pr>

```

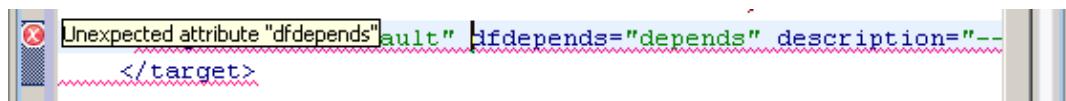
La vue structure affiche l'arborescence du fichier.



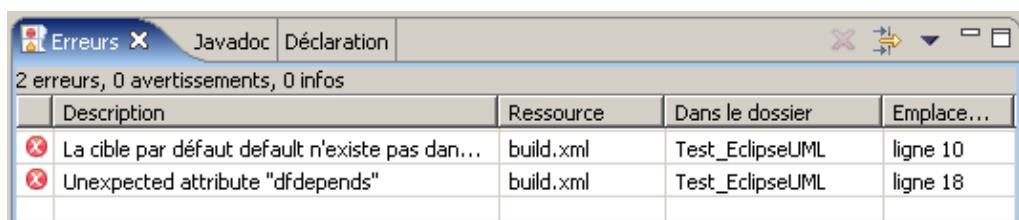
Une fois le contenu du fichier saisi, il suffit de l'enregistrer.



Les erreurs dans le fichier de configuration sont signalées dans l'éditeur avec possibilité d'obtenir une bulle d'aide précisant le problème en laissant le curseur de la souris sur la petite icône rouge avec une croix blanche.



À la sauvegarde du fichier de configuration, les erreurs persistantes sont signalées dans la vue « Erreurs »

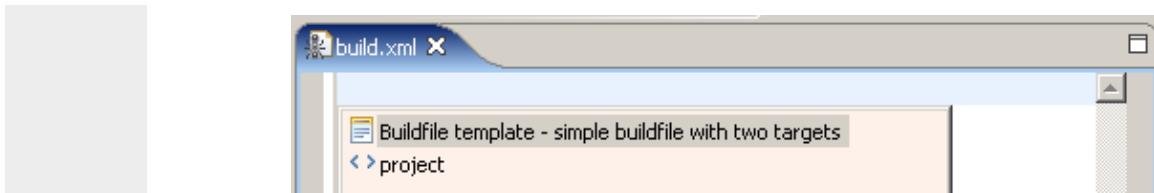


L'éditeur de fichier de configuration de Ant propose une compléction de code avec une bulle d'aide qui permet de fournir des informations sur l'entité sélectionnée.



L'éditeur propose des modèles de code utilisable via la combinaison de touches Ctrl+espace.

Par exemple, à la création d'un fichier build.xml vide, il est possible de demander l'insertion d'un modèle contenant deux cibles. Une bulle d'aide fournit un aperçu du code qui sera inséré.



```
<target name="default" depends="depends" description="--> description">
<java>
</tar> <> java
<!--
<javac srcdir="src" deskdir="ds" classpath="jarfile" debug="on" />
javac - add a javac task
```

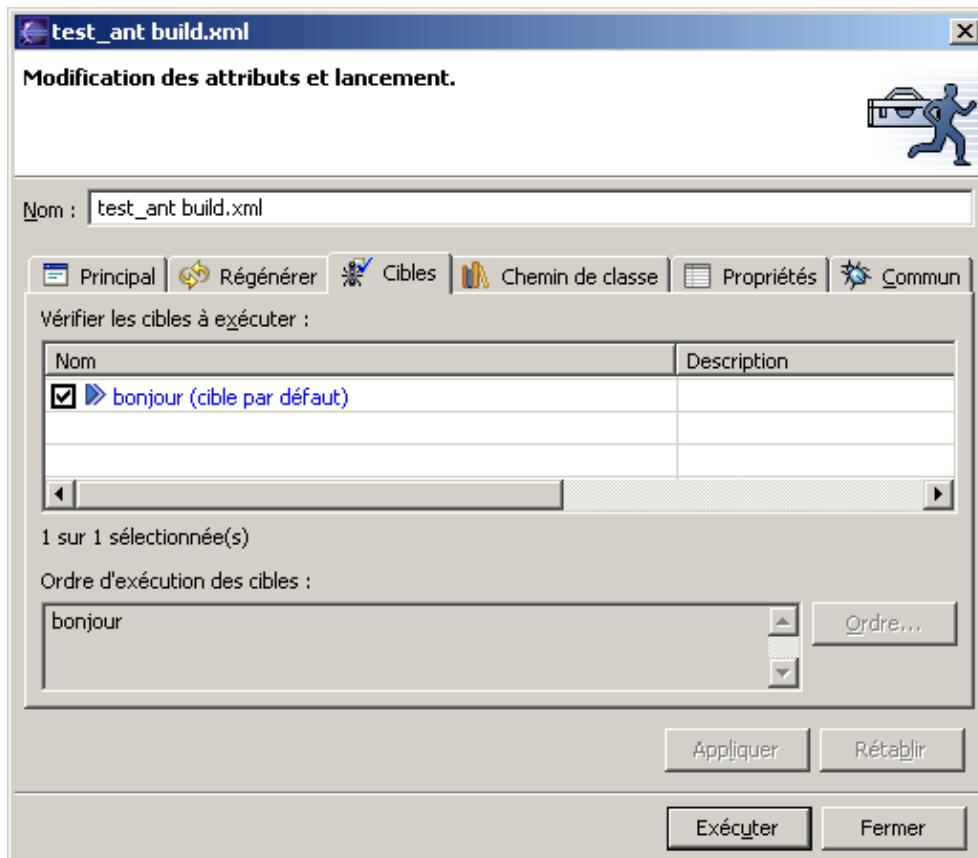
L'édition de fichier de configuration d'Ant peut formater le code source XML du fichier de configuration en sélectionnant l'option « Formater » du menu contextuel ou en utilisant la combinaison de touches Ctrl+Maj+F.

L'exécution de Ant se fait par défaut dans une machine virtuelle dédiée et non plus dans la machine virtuelle dans laquelle Eclipse s'exécute.

### 10.3. Exécuter Ant

Pour exécuter Ant avec un fichier build.xml, il suffit dans la vue "Navigateur" ou "Packages" de sélectionner ce fichier build.xml et d'utiliser l'option "Exécuter Ant" du menu contextuel.

Une boîte de dialogue s'ouvre. Elle permet de modifier quelques paramètres externes à Ant et de lancer l'exécution.



Nom	Description
<input checked="" type="checkbox"/> bonjour (cible par défaut)	

1 sur 1 sélectionnée(s)

Ordre d'exécution des cibles :

bonjour

Ordre...

Appliquer

Rétablissement

Exécuter

Fermer

Par défaut, la tâche définie par défaut dans le fichier build.xml est sélectionnée.

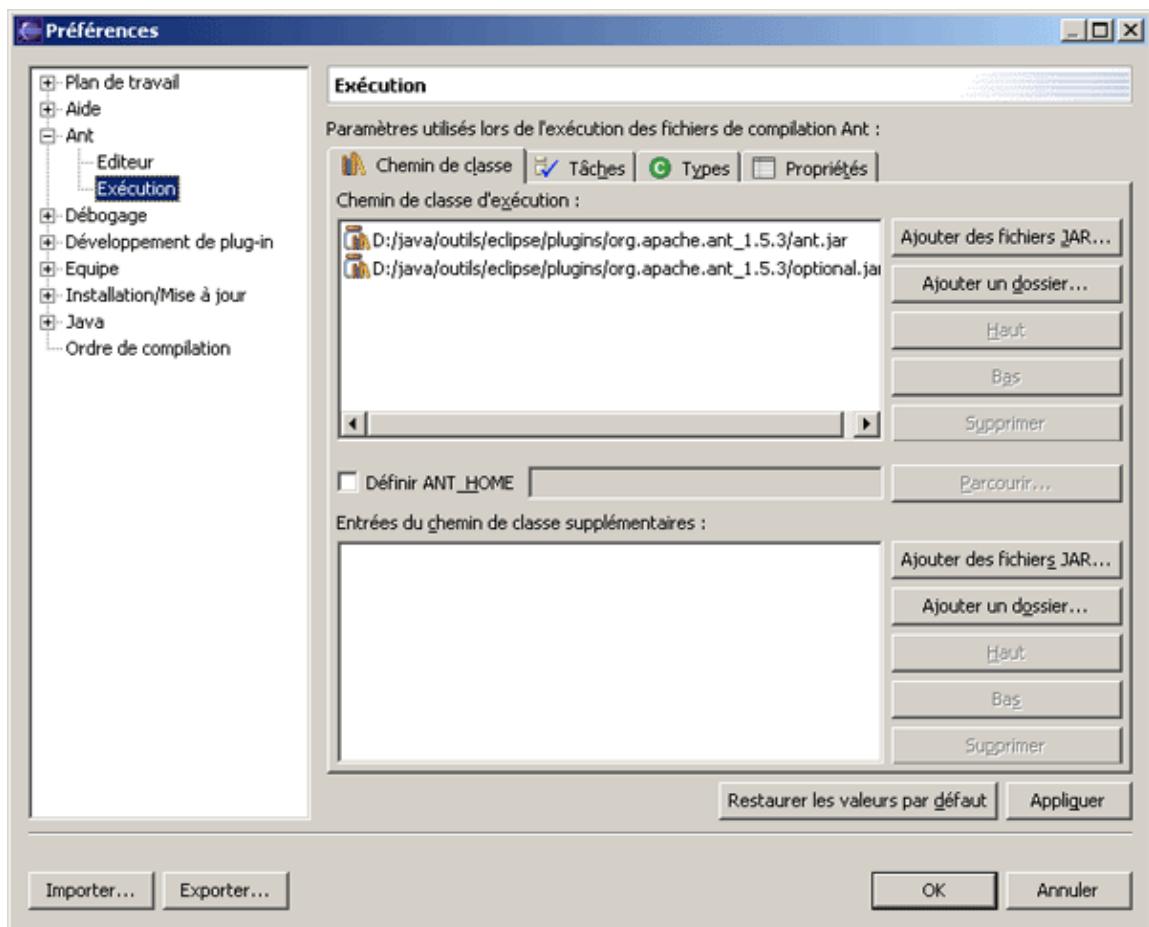
Pour lancer l'exécution, il suffit de cliquer sur le bouton "Exécuter".

Le résultat de l'exécution s'affiche dans la vue "Console"

Exemple :
<pre>Buildfile: I:\eclipse\workspace\test_junit\build.xml  bonjour: [echo] Premier test avec Ant! BUILD SUCCESSFUL Total time: 401 milliseconds</pre>

## 10.4. Les paramètres

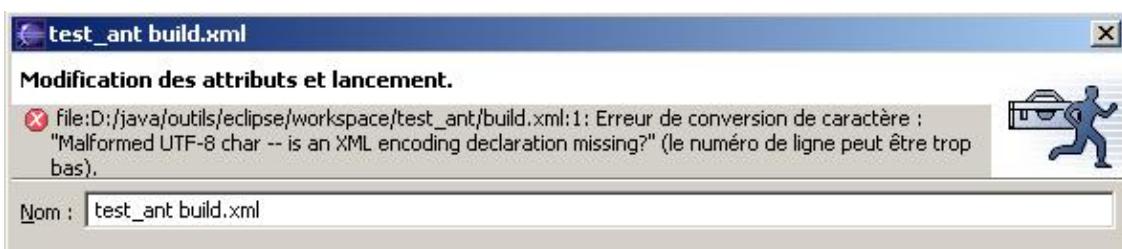
Dans les préférences, il est possible de préciser des paramètres par défaut utilisés lors de l'édition d'un fichier ant ou de son exécution.



## 10.5. Résolution des problèmes

Plusieurs problèmes peuvent survenir lors de l'utilisation de Ant. Voici une solution pour quelques uns d'entre eux.

### 10.5.1. Utilisation de caractères accentués



Il faut ajouter l'attribut encoding avec le jeux de caractères utilisés dans le prologue du fichier build.xml.

Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

### 10.5.2. Impossible de lancer la tâche javadoc

Exemple :

```
Buildfile: I:\eclipse\workspace\test_junit\build.xml
init:
    [echo] Generation numero : 7 du July 2 2003
compil:
doc:
    [javadoc] Generating Javadoc
    [javadoc] Javadoc execution
    [javadoc] BUILD FAILED: file:I:/eclipse/workspace/test_junit/build.xml:37:
        Javadoc failed: java.io.IOException: CreateProcess: javadoc.exe -d "I:\eclipse
        \workspace\test_junit\build\doc" -use -package -classpath "I:\eclipse\s
        tartup.jar;I:\eclipse\workspace\test_junit\junit.jar" -version
        -author "I:\eclipse\workspace\test_junit\src\MaClasse.java" "I:\eclipse
        \workspace\test_junit\src\MaClasse2.java" error=2
Total time: 681 milliseconds
```

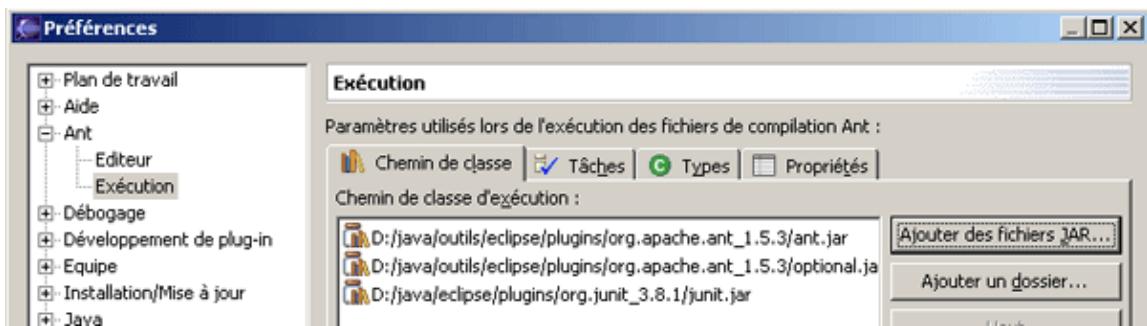
Il faut vérifier la présence de l'outil dans les répertoires désignés par la variable d'environnement PATH du système d'exploitation. Dans le cas de Javadoc sous Windows, il faut s'assurer que le répertoire %JAVA\_HOME%\bin soit inséré dans la variable PATH. Si cette dernière doit être modifiée, il faut arrêter et relancer Eclipse après la modification pour que celle ci soit prise en compte.

### 10.5.3. Impossible d'utiliser la tâche JUnit

Exemple :

```
Buildfile: I:\eclipse\workspace\test_junit\build.xml
init:
    [echo] Generation numero : 13 du July 2 2003
compil:
test:
    [junit] BUILD FAILED: file:I:/eclipse/workspace/test_junit/build.xml:62:
        Could not create task or type of type: junit.
        Ant could not find the task or a class this task relies upon.
```

Dans les préférences, il faut rajouter le fichier junit.jar dans l'onglet "Classpath" de l'arborescence "Ant/Runtime"



## 10.6. Un exemple complet

### Exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<project name="TestAnt1" default="all">
    <description>
        Génération de l'application
    </description>

    <property name="bin" location="bin"/>
    <property name="src" location="src"/>
    <property name="build" location="build"/>
    <property name="doc" location="${build}/doc"/>
    <property name="lib" location="${build}/lib"/>
    <property name="junit_path" value="junit.jar"/>

    <target name="init" description="Initialisation">
        <tstamp/>
        <buildnumber file="numerobuild.txt" />
        <echo message="Generation numero : ${build.number} du ${TODAY}" />
    </target>

    <target name="compil" depends="init" description="Compilation">
        <javac srcdir="${src}" destdir="${bin}">
            <classpath>
                <pathelement path="${java.class.path}" />
                <pathelement location="${junit_path}" />
            </classpath>
        </javac>
    </target>

    <target name="all" depends="init, compil, test, doc"
           description="Generation complete">
        <echo message="Generation complete." />
    </target>
    <target name="doc" depends="compil" description="Generation documentation">
        <javadoc destdir="${doc}" author="true" version="true" use="true"
               package="true">
            <fileset dir = "${src}">
                <include name="**/*.java"/>
                <exclude name="**/*Test*"/>
            </fileset>
            <classpath>
                <pathelement path="${java.class.path}" />
                <pathelement location="${junit_path}" />
            </classpath>
        </javadoc>
    </target>

```

```

</javadoc>
</target>

<target name="test" depends="compil" description="Executer tests avec JUnit">
  <junit fork="yes" haltonerror="true" printsummary="on">
    <formatter type="plain" usefile="false" />
    <test name="ExecuterLesTests"/>
    <classpath>
      <pathelement location="${bin}"/>
      <pathelement location="${junit_path}"/>
    </classpath>
  </junit>
</target>
</project>

```

### Résultat de l'exécution :

```

Buildfile: I:\eclipse\workspace\test_junit\build.xml
init:
[echo] Generation numero : 16 du July 2 2003
compil:
test:
[junit] Running ExecuterLesTests
[junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0,02 sec
[junit] Testsuite: ExecuterLesTests
[junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0,02 sec
[junit]
[junit] Testcase: testCalculer took 0,01 sec
[junit] Testcase: testCalculer took 0 sec
[junit] Testcase: testSommer took 0 sec
doc:
[javadoc] Generating Javadoc
[javadoc] Javadoc execution
[javadoc] Loading source file I:\eclipse\workspace\test_junit\src\MaClasse.java...
[javadoc] Loading source file I:\eclipse\workspace\test_junit\src\MaClasse2.java...
[javadoc] Constructing Javadoc information...
[javadoc] Standard Doclet version 1.4.1
[javadoc]
[javadoc] Building tree for all the packages and classes...
[javadoc] Building index for all the packages and classes...
[javadoc] Building index for all classes...
all:
[echo] Generation complete.
BUILD SUCCESSFUL
Total time: 4 seconds

```

## 10.7. L'éditeur de fichiers Ant



Eclipse 3.1 propose un éditeur dédié à la mise à jour de script Ant. Cet éditeur est utilisé par défaut utilisé pour éditer les scripts Ant. Il propose des fonctionnalités qui facilitent leur rédaction.

Dans un fichier build.xml vide, il est possible de demander la création un fichier xml type avec deux targets en utilisant la combinaison de touches Ctrl+espace

```
<?xml version="1.0"?>
<!-- =====
   27 janv. 2006 00:15:15

   project
   description

   jumbo
   =====
<project name="project" default="default">
   <description>
      description
   </description>

   <!-- =====
      target: default
   ===== -->
```

Chaque partie qui compose le fichier build.xml est marquée avec une petite icône dans la partie gauche. Chacune de ces icônes permet de masquer la partie qu'elle désigne. Lorsque le curseur passe au dessus de cette partie, une ligne désigne la partie.

```
<?xml version="1.0"?>
⊖<!-- =====
   27 janv. 2006 00:19:50

   project
   description

   jumbo
   =====
⊖<project name="project" default="default">
⊖<description>
   description
</description>
```

Un clic sur l'icône permet de masquer la partie concernée.

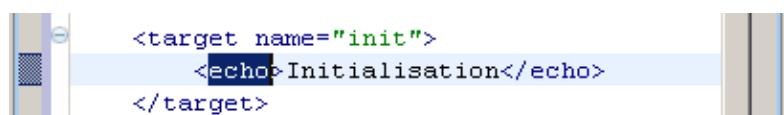
```
<?xml version="1.0"?>
⊕<!-- =====
⊖<project name="project" default="default">
⊖<description>
   description
</description>
```

L'icône se transforme en un petit signe plus : . En cliquant, sur cette icône, la partie masquée est de nouveau affichée. En laissant le curseur de la souris sur cette icône, la partie masquée est affichée dans une bulle d'aide.

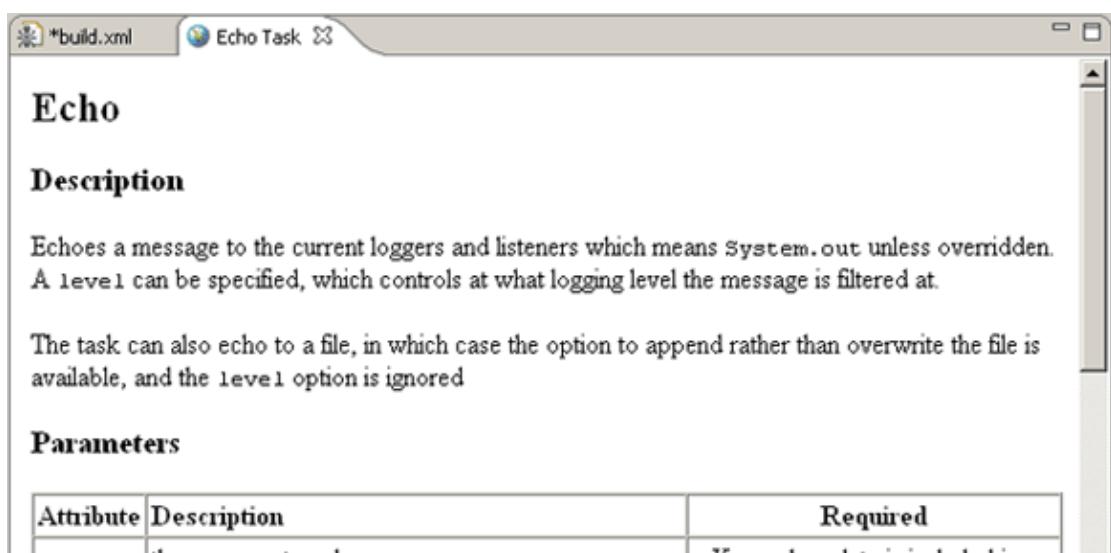


The screenshot shows an Eclipse code editor window titled '\*build.xml'. The XML code contains several sections that are collapsed, indicated by a '+' icon next to the opening tag. One section is expanded, showing the text: '<!-- =====--> 27 janv. 2006 00:19:50 project description jumbo =====-->'.

Il est possible d'obtenir de l'aide à partir de l'éditeur. Il suffit de sélectionner un tag et d'appuyer sur la combinaison de touches Shift + F2.



Attention, une connexion internet est nécessaire pour utiliser en oeuvre cette fonctionnalité. Le navigateur interne s'ouvre avec une aide en ligne concernant le tag.

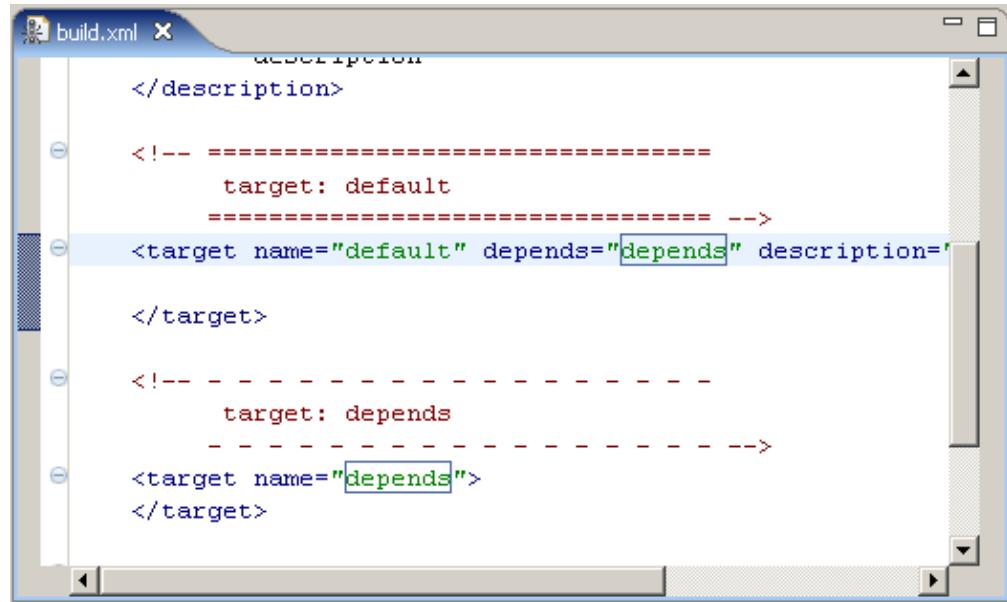


The screenshot shows the Eclipse Help browser window titled 'Echo Task'. The page has a title 'Echo' and a section 'Description' which includes the text: 'Echoes a message to the current loggers and listeners which means System.out unless overridden. A level can be specified, which controls at what logging level the message is filtered at.' Below this is another section: 'The task can also echo to a file, in which case the option to append rather than overwrite the file is available, and the level option is ignored'. At the bottom is a table titled 'Parameters' with one row:

Attribute	Description	Required
message	the message to echo	Yes, unless data is included in a

Cette documentation permet d'obtenir un descriptif du tag, un détail de ces paramètres et des exemples.

L'éditeur est capable de synchroniser les modifications apportées à un élément : pour cela il faut positionner le curseur sur l'élément et d'utiliser l'option « Renommer dans le fichier » du menu contextuel.



```
</description>

<!-- =====
      target: default
===== -->
<target name="default" depends="depends" description="

</target>

<!-- -----*
      target: depends
-----*>
<target name="depends">
</target>
```

Toutes les occurrences sont marquées par un cadre. La modification de l'élément entraîne automatiquement la modification de toutes ces utilisations dans le fichier.

L'option « Formater » du menu contextuel permet de mettre en forme le code source du script Ant.

## 10.8. Le débogage de fichiers Ant



Cette section sera développée dans une version future de ce document

# Chapitre 11

La version 2.1 d'Eclipse intègre la possibilité d'utiliser JUnit directement dans l'IDE.

JUnit est un framework open source pour réaliser des tests unitaires sur du code Java. Le principal intérêt est de s'assurer que le code répond toujours au besoin même après d'éventuelles modifications.

Le but est d'automatiser les tests. Ceux ci sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ces résultats.

Avec Junit, l'unité de tests est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes :

- création d'une instance de la classe et de tout autre objet nécessaire aux tests
- appel de la méthode à tester avec les paramètres du cas de test
- comparaison du résultat obtenu avec le résultat attendu : en cas d'échec, une exception est levée

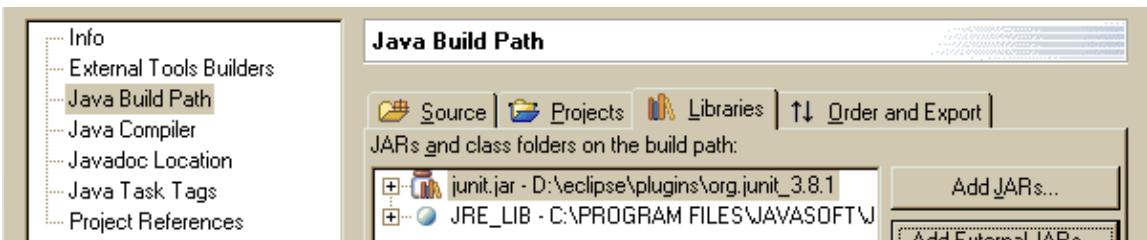
### 11.1. Paramétrage de l'environnement

Pour pouvoir utiliser JUnit dans un projet, il faut ajouter le fichier junit.jar dans le classpath du projet. Il faut pour cela :

- sélectionner les propriétés du projet
- dans l'onglet librairies, cliquer sur le bouton « Add External Jar »
- sélectionner le fichier junit.jar dans le répertoire plugins/org.junit\_3.8.1 du répertoire d'Eclipse



Eclipse ajoute le fichier junit.jar au classpath



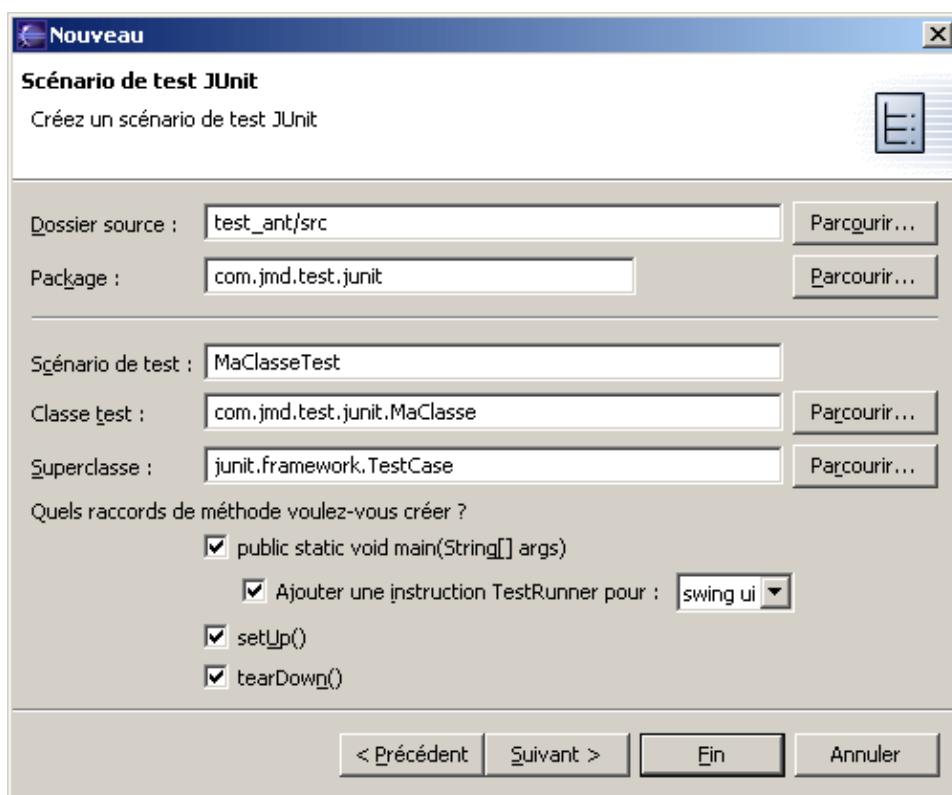
Dans ce chapitre, la classe suivante est utilisée comme classe à tester avec JUnit.

Exemple :

```
package com.moi.test.junit;
public class MaClasse {
    public int additioner(int a, int b) {
        return a + b;
    }
}
```

## 11.2. Ecriture des cas de tests

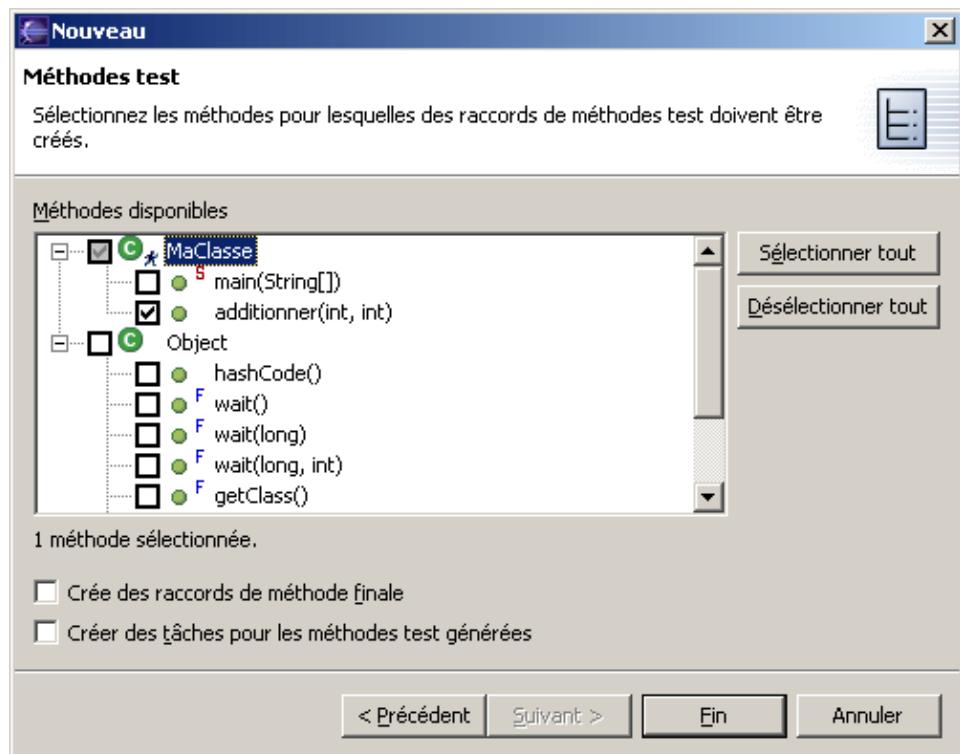
Pour utiliser JUnit, il faut créer une classe qui va contenir les cas de test. Il faut créer une nouvelle entité de type "Java / JUnit / Scénario de test".



Si le fichier junit.jar n'est pas inclus dans le classpath du projet, un message d'erreur est affiché et il est impossible de poursuivre l'exécution de l'assistant.



Cliquez sur le bouton "Suivant".



Il faut compléter la classe générée selon les besoins : par exemple, ajouter un attribut qui va contenir une instance de la classe à tester, ajouter l'instanciation de cette classe dans la méthode setUp() et libérer cette instance dans la méthode tearDown().

Il faut ajouter les traitements nécessaires dans les méthodes testXXX() en utilisant l'API de JUnit.

#### Exemple :

```
package com.moi.test.junit;

import junit.framework.TestCase;

public class MaClasseTest extends TestCase {
    private MaClasse maClasse = null;

    public MaClasseTest(String arg0) {
        super(arg0);
    }

    public static void main(String[] args) {
        junit.swingui.TestRunner.run(MaClasseTest.class);
    }

    protected void setUp() throws Exception {
        super.setUp();
        maClasse = new MaClasse();
    }

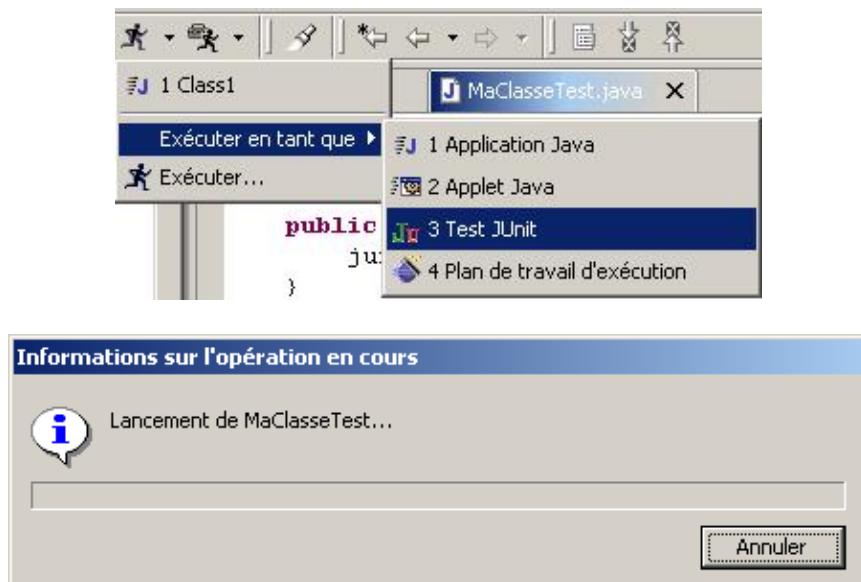
    protected void tearDown() throws Exception {
        super.tearDown();
        maClasse = null;
    }

    public void testAdditioner() {
        assertTrue(maClasse.additionner(2,2) == 4);
    }
}
```

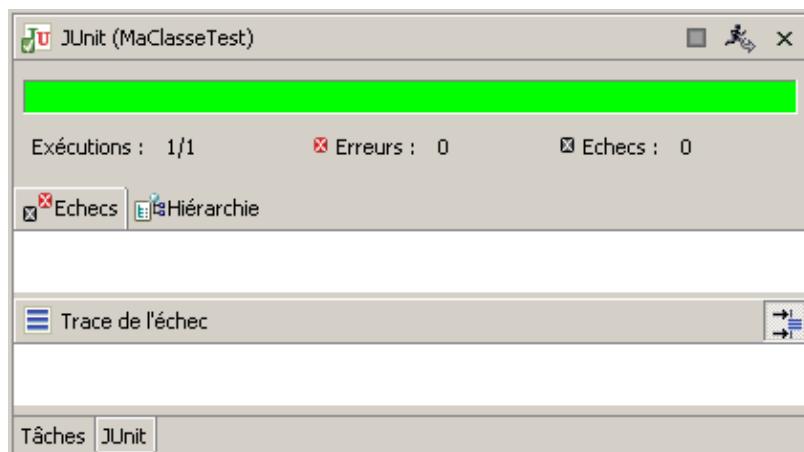
JUnit utilise l'instrospection pour exécuter les méthodes commençant par test.

### 11.3. Exécution des cas de tests

Pour exécuter les tests, il faut exécuter la classe en tant que « Test JUnit ».



Eclipse exécute les tests et affiche le résultat dans une vue dédiée.



Si tous les cas de tests ont été exécutés avec succès, une ligne verte est affichée.

Cette vue contient deux onglets :

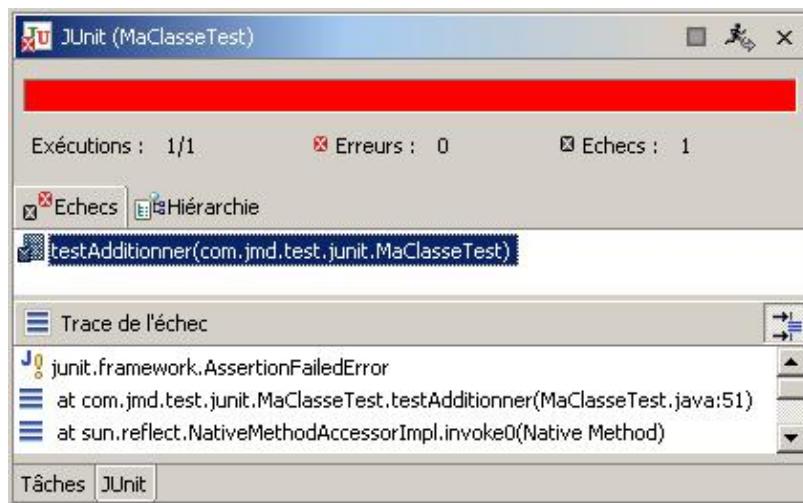
- « Failures » : contient la liste des cas de tests qui ont échoués
- « Hierarchy » : contient une arborescence des cas de tests

Dans le cas où un ou plusieurs tests échouent, la ligne est rouge.

Exemple : si le cas test suivant est ajouté :

```
public void testAdditioner() {  
    assertTrue(maClasse.additioner(2,2) == 4);  
    assertTrue(maClasse.additioner(2,3) == 4);  
}
```

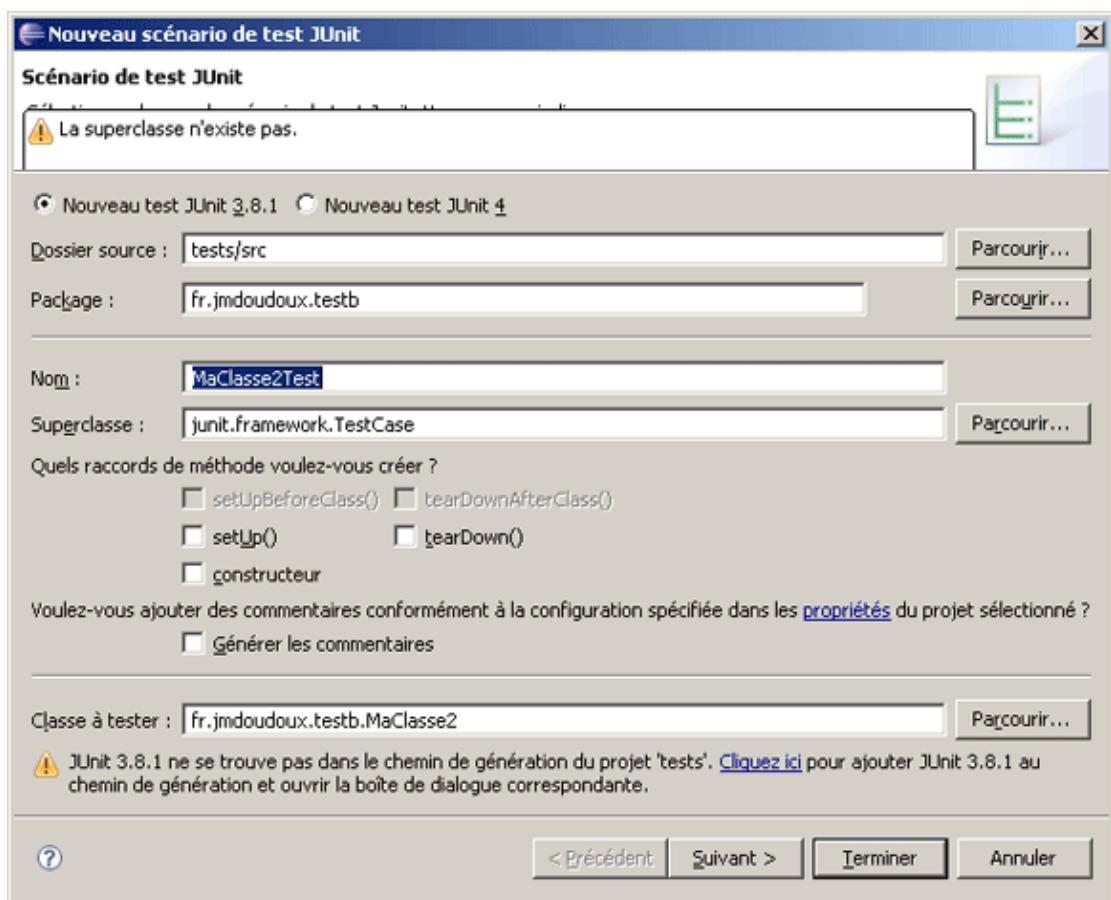
Une exécution de l'exemple précédent permet d'avoir le résultat suivant :



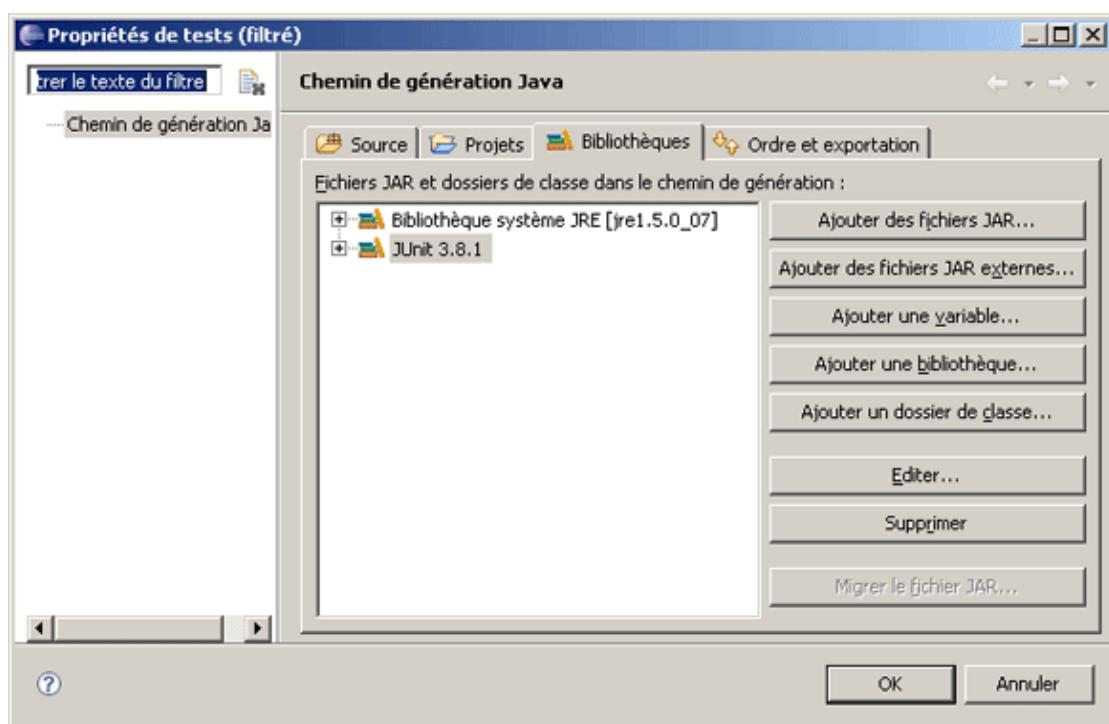
## 11.4. Le support de JUnit 4



Eclipse 3.2 propose un support de la version 4 de JUnit. L'assistant de création d'un scénario de test permet de sélectionner la version de JUnit à utiliser.

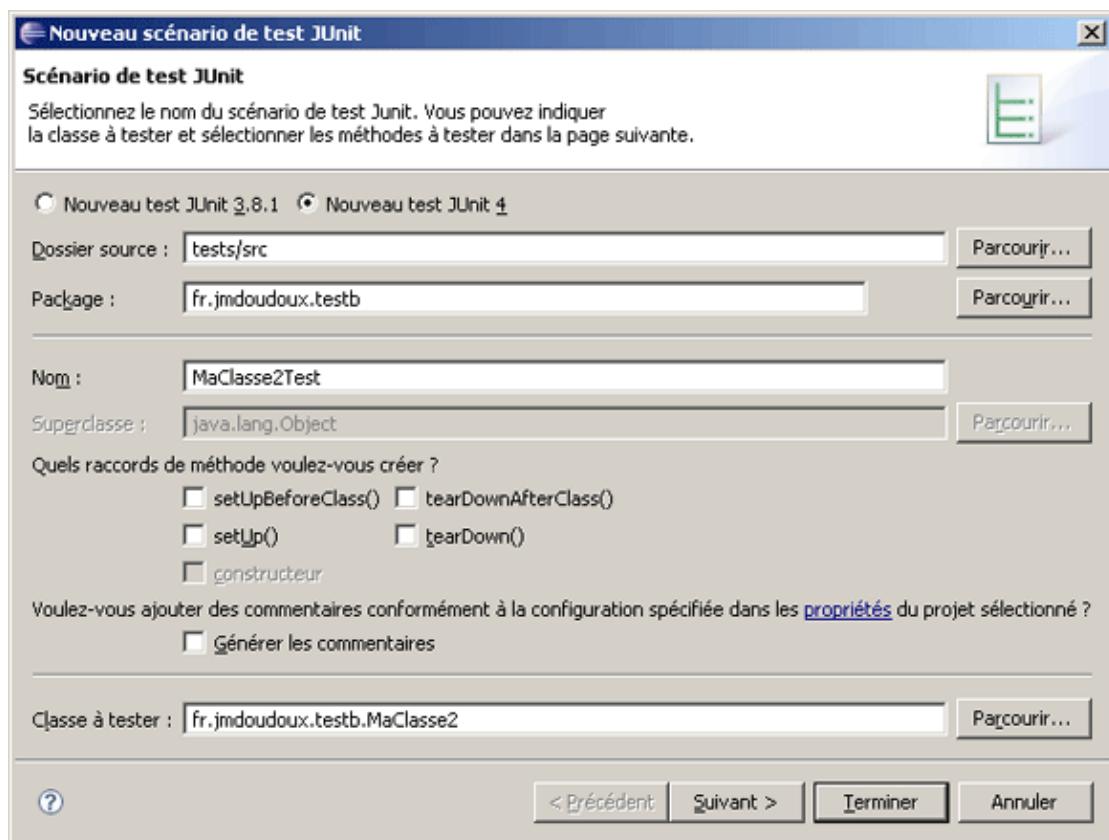


Pour ajouter la bibliothèque au classpath du projet, il suffit de cliquer sur le lien « Cliquez ici »

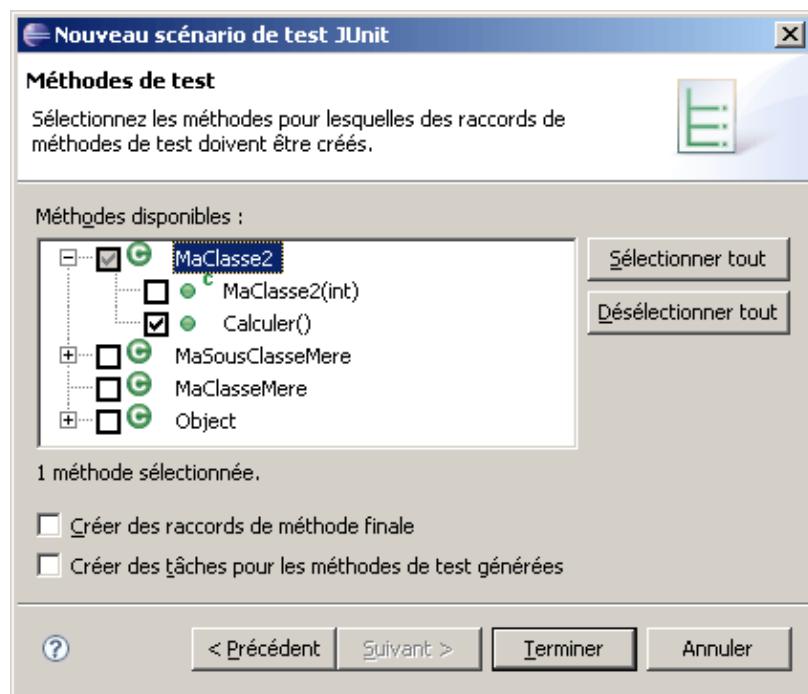


Le fichier jar correspondant est ajouté et il suffit simplement d'appuyer sur le bouton « OK » pour valider l'ajout de la bibliothèque.

Le principe est identique lorsque c'est la version 4.0 de JUnit qui est sélectionnée.



Cliquez sur le bouton « Suivant ».



Sélectionnez les méthodes à tester dans le scénario et cliquez sur le bouton « Terminer »

```

package fr.jmdoudoux.testb;

import static org.junit.Assert.*;

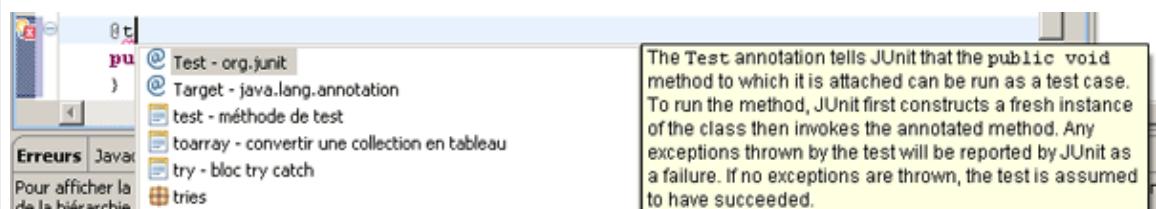
public class MaClasse2Test {

    @Test
    public void testCalculer() {
        fail("Non implémenté actuellement");
    }
}

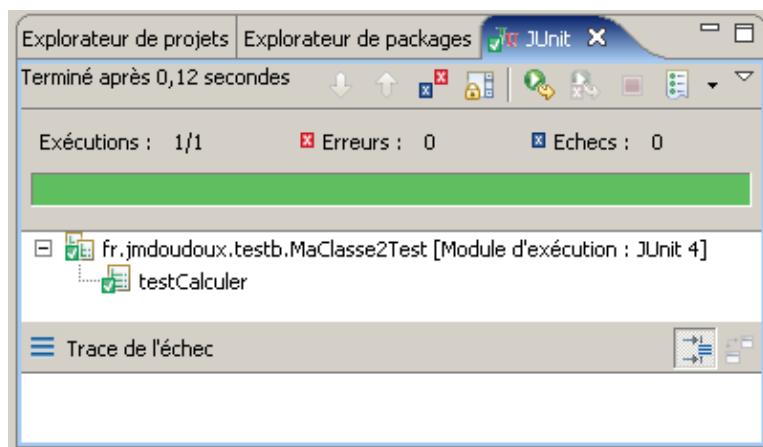
```

La classe générée prend en compte les fonctionnalités de la version 4 notamment les annotations.

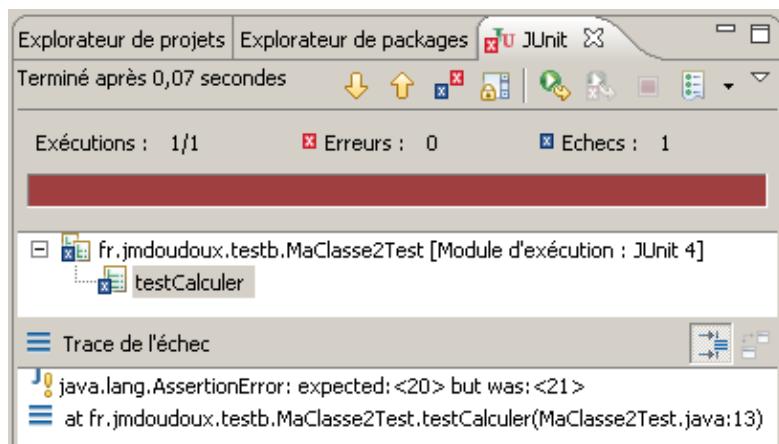
L'assistant de code propose l'annotation @test.



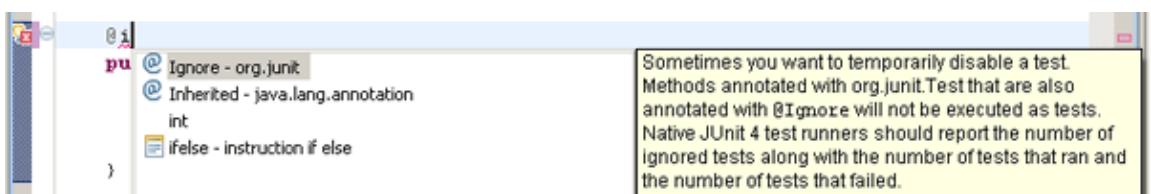
Le menu « Exécuter » propose automatiquement l'option « Exécuter en tant que / Test JUnit ».



En cas d'échec, les erreurs sont précisées dans la partie « Trace de l'échec ».



L'assistant de code propose aussi l'annotation `@Ignore` pour ne pas prendre en compte une méthode de test.



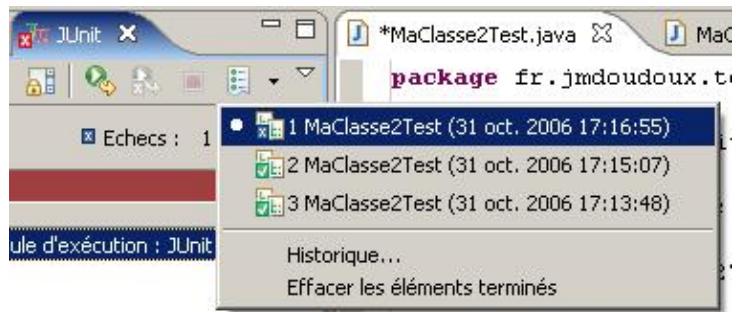
La vue JUnit propose un historique des tests effectués : pour afficher la liste des tests effectués, il suffit de cliquer sur le bouton .



Il suffit de sélectionner le test à exécuter et de cliquer sur le bouton « OK ».

Il est possible de supprimer un test en le sélectionnant et en cliquant sur le bouton « Supprimer » ou de supprimer tous les tests en utilisant le bouton « Supprimer tout »

En cliquant sur la petite flèche du bouton, la liste apparaît directement dans le menu déroulant.



L'option « Historique ... » permet d'ouvrir la boîte de dialogue « Exécutions de test ».

L'option « Effacer les éléments terminés » permet de vider l'historique.

# Partie 3 : les fonctions avancées d'Eclipse

Cette troisième partie présente des fonctionnalités avancées d'Eclipse.

Elle comporte les chapitres suivants :

- ◆ La gestion de la plate-forme : détaille les fonctionnalités proposées pour gérer les plug-ins et la plate-forme Eclipse.
- ◆ CVS 2.0 et Eclipse 2.1 : détaille l'utilisation de CVS 2.0 avec Eclipse 2.1.
- ◆ CVS 2.5 et Eclipse 3.0 : détaille l'utilisation de CVS 2.5 avec Eclipse 3.0.
- ◆ Subversion et Eclipse : détaille l'utilisation de subversion avec Eclipse 3.0.

## 12. La gestion de la plate-forme

# Chapitre 12

Eclipse est conçu pour être un outil modulaire. De nombreux modules (plug-ins) sont fournis avec Eclipse mais il est aussi très facile d'en ajouter d'autres développés par la communauté ou des sociétés commerciales.

L'installation d'un plug-in est souvent très simple car elle consiste essentiellement à décompresser l'archive qui contient le plug-in pour que le contenu soit insérer dans le répertoire "plugins" où est installé Eclipse.

Eclipse 2.0 propose une fonctionnalité qui permet d'installer ou de mettre à jour des plug-ins via le web.

### 12.1. Informations sur les plug-ins installés

Pour obtenir des informations sur la plate-forme, il faut utiliser le menu "Aide / A propos de plateforme Eclipse".

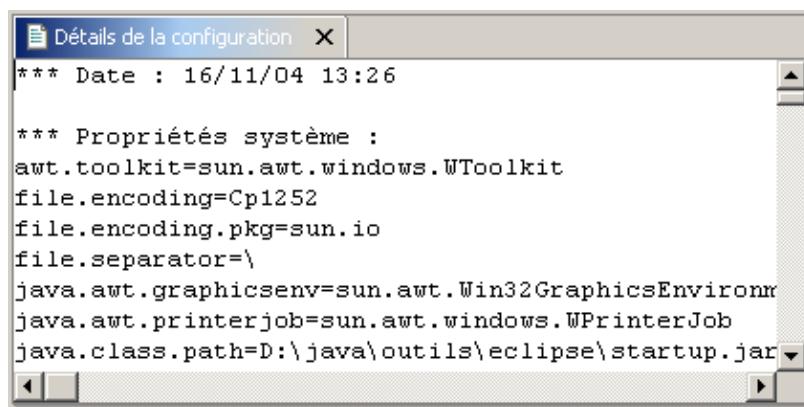


Cette boîte de dialogue affiche des informations générales sur Eclipse et permet de sélectionner le type d'informations supplémentaires souhaitées.

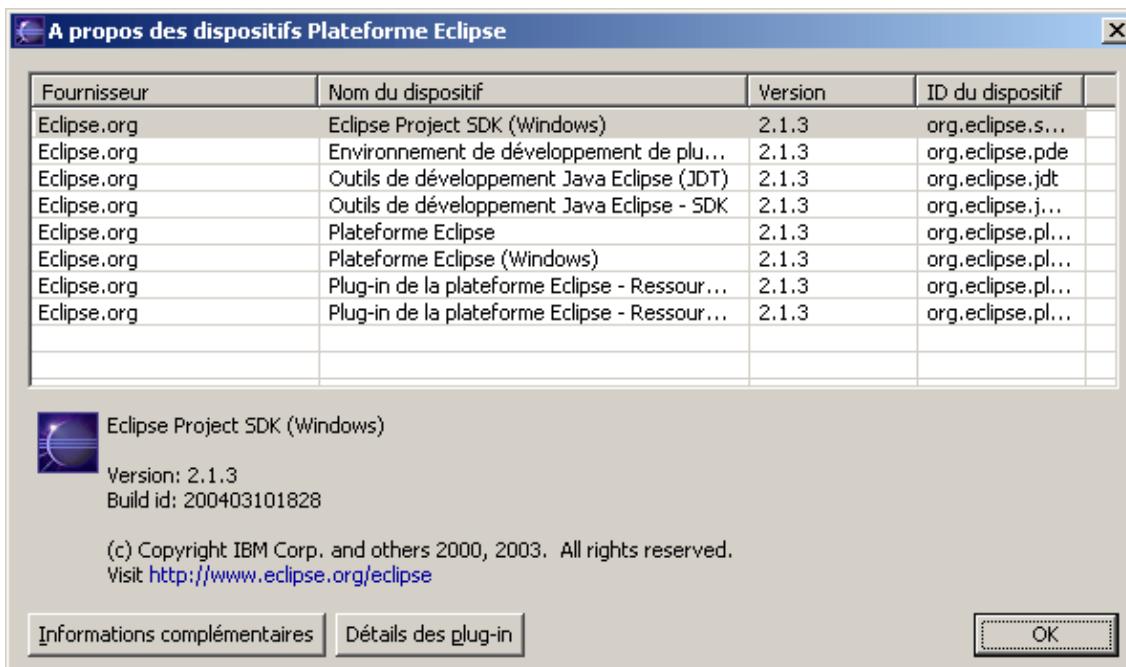
Cliquez sur le bouton "Détails des plug-ins" pour obtenir une liste de tous les plug-ins installés.



Un clic sur le bouton "Détails de la configuration" affiche un fichier dans la vue éditeur contenant des informations détaillées sur la configuration en cours d'utilisation.



Un clic sur le bouton "Détails des dispositifs", permet d'avoir des informations sur les éléments de base d'Eclipse.



## 12.2. Installation du plug-in JadClipse sous Eclipse 1.0

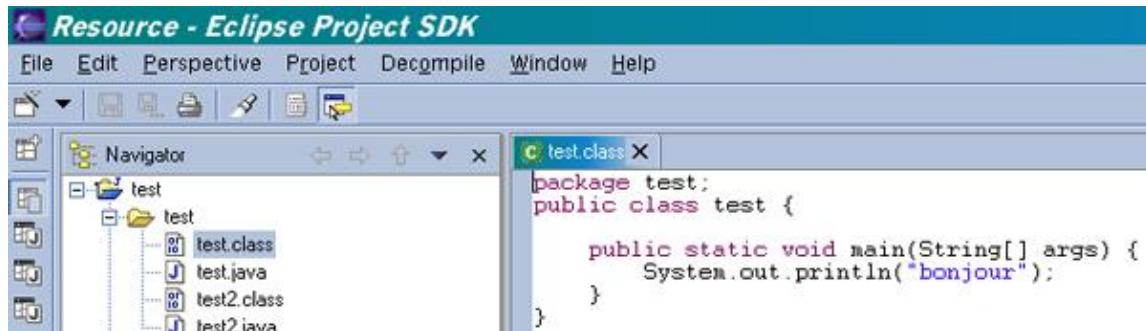
Jadclipse est un plug-in qui permet d'utiliser le décompilateur Jad dans Eclipse.

Il suffit de télécharger le fichier zip contenant le module à l'url <http://sourceforge.net/projects/jadclipse/>. Il faut aussi avoir installer l'outil Jad.

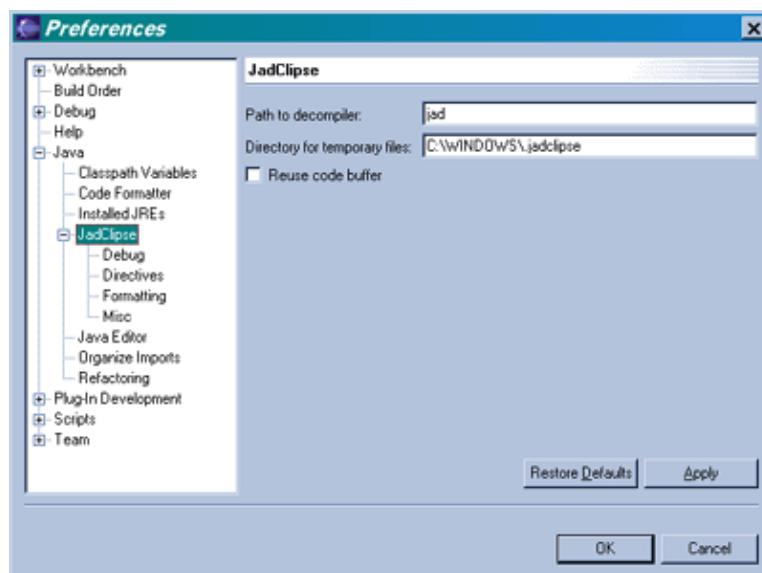
Pour installer le plug-in, il suffit simplement de décompresser le fichier dans le répertoire "plugins" du répertoire principal d'Eclipse.

Pour que le module soit pris en compte et configurer automatiquement, il suffit de lancer ou relancer Eclipse.

A l'ouverture d'un fichier .class, le menu « Decompile » apparaît dans le menu principal. Il faut cocher l'option « Engage Jadclipse ». Chaque fichier .class ouvert est automatiquement décompilé et le code source est affiché dans l'éditeur.



Le plug-in est parfaitement intégré dans Eclipse : les options du plug-in sont modifiables dans l'arborescence "Java / Jadclipse" des préférences (menu "Window / Preferences").



## 12.3. La mise à jour des plug-ins avec Eclipse 2.0

Eclipse 2.0 propose des fonctionnalités pour permettre la mise à jour ou l'installation de plug-ins en utilisant le web.

### 12.3.1. La perspective « Installation / Mise à jour »

La perspective « Installation / Mise à jour » permet la gestion des mises à jour. Pour l'afficher, il suffit de sélectionner cette perspective ou de sélectionner l'option « Mise à jour des logiciels / Gestionnaire des mises à jour » du menu "Aide".

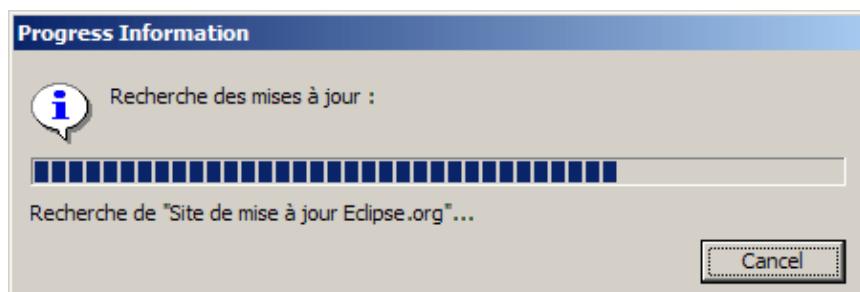


La vue "Mises à jour des dispositifs" recense la liste des sources d'où peuvent être téléchargées les mises à jour.

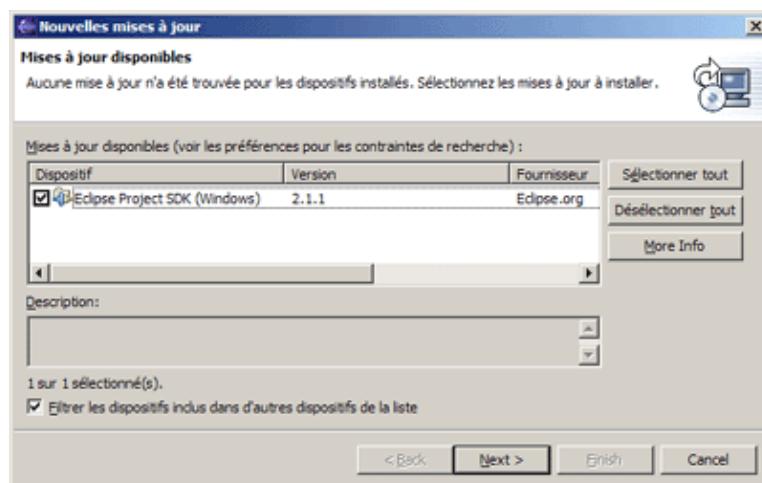
### 12.3.2. Recherche et installation des mises à jour

Il faut utiliser l'option "Mise à jour des logiciels" du menu "Aide".

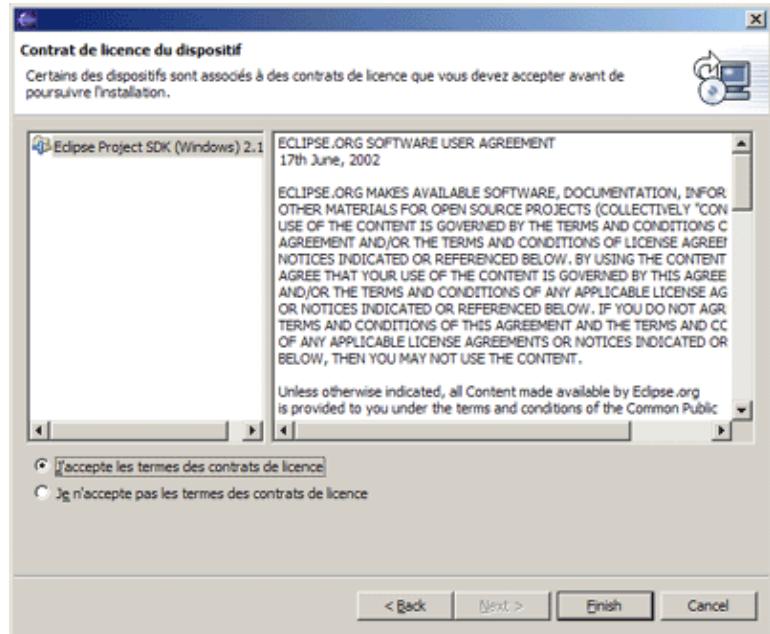
Un assistant recherche les mises à jour pour les plug-ins dont le site est enregistré dans Eclipse.



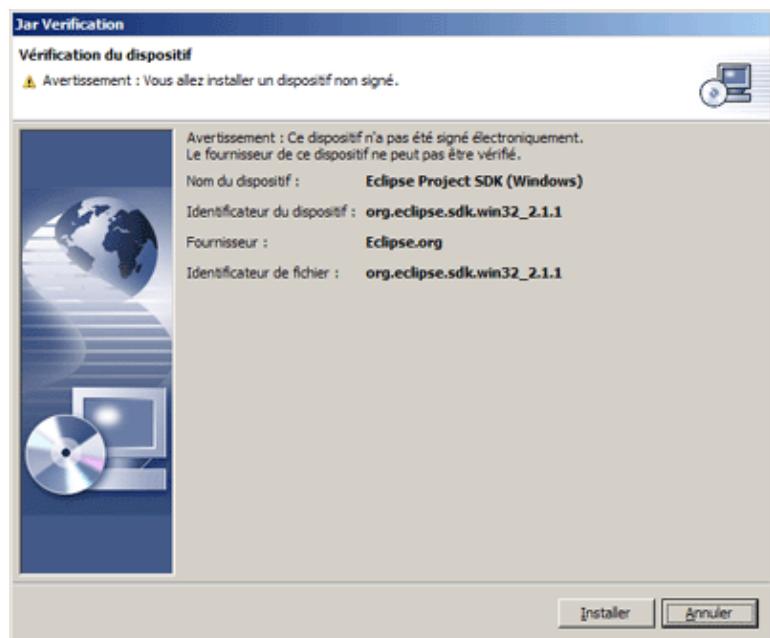
L'exemple ci dessous illustre la mise à jour d'une version 2.1 d'Eclipse avec la version 2.1.1.



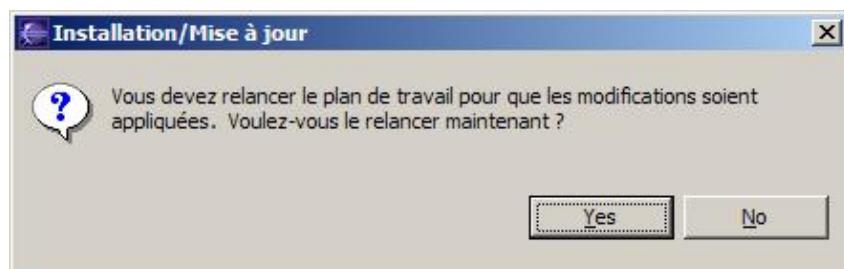
Cliquez sur le bouton "Next"



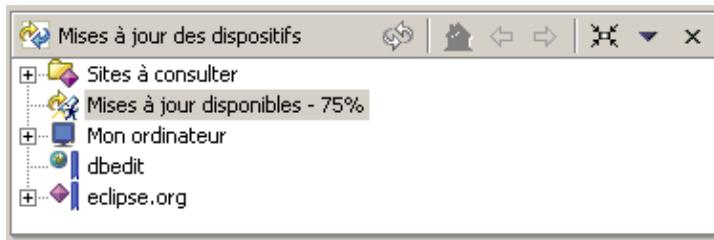
Lisez la licence et si vous l'acceptez, Cliquez sur le bouton "J'accepte les termes des contrats de licence" puis sur le bouton "Finish"



Cliquez sur le bouton "Installer". Une fois l'installation terminée, il faut relancer Eclipse.



Pour rechercher les mises à jour, il est aussi possible dans la perspective "Installation/mise à jour" de cliquer sur "Mise à jour disponible" dans la vue "Mises à jour des dispositifs".



La progression et le détail des mises à jour trouvées sont affichés dans la vue "Aperçu"

A screenshot of the Eclipse Device Updates preview view. The title bar says "Aperçu". The main area is titled "Recherche - Mises à jour disponibles" with a sub-instruction: "Recherche des mises à jour pour les dispositifs installés dans cette application. Les mises à jour correspondent à des dispositifs possédant le même identificateur que ceux installés mais avec des versions plus récentes." It includes instructions for controlling search scope and notes about preferences. Below this are tabs for "Paramètres de la requête" (selected) and "Paramètres de portée". A message says "Dernière recherche : non disponible" and an "Annuler" button. A progress bar indicates a search in progress: "Recherche des mises à jour : Recherche dans "New update site"...". The bottom section shows a table titled "Résultats de la recherche de dispositif... jour de dispositifs disponibles (9)" with columns "Dispositif" and "Version". The results list several Eclipse components with their versions: DbEdit Database Utilities (1.0.2), Eclipse Java Development Tools (2.1.3), Eclipse JDT Plug-in Developer Res... (2.1.3), and others.

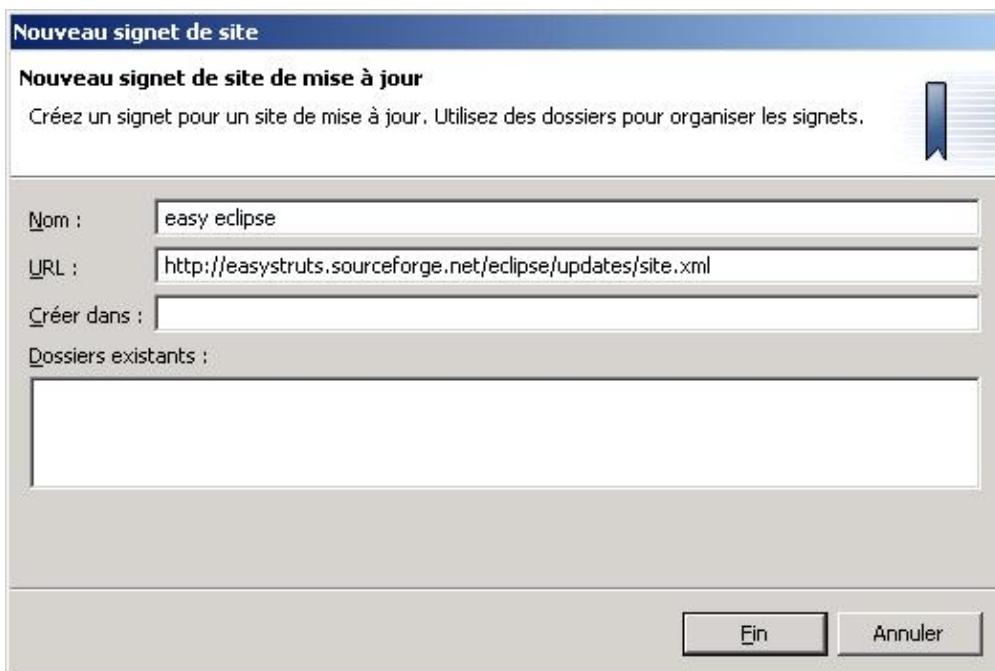
### 12.3.3. Installation d'un nouveau plug-in

L'exemple ci dessous met en oeuvre l'installation du plug-in Easy Struts. Pour d'autres plug-in permettant leur mise à jour par le réseau, la procédure est similaire.

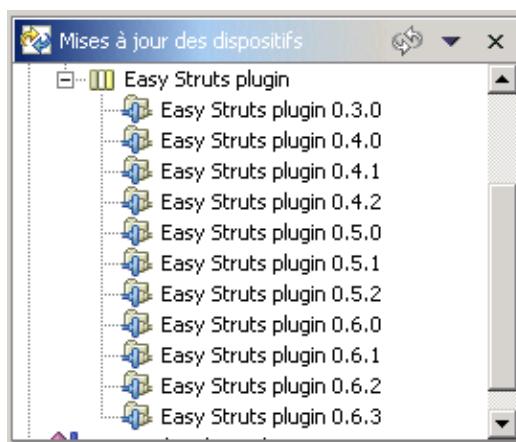
Pour ajouter une nouvelle source, il suffit de sélectionner l'option « Nouveau / Signet du site ... » du menu contextuel de la vue « Mises à jour des dispositifs ».



Une boîte de dialogue permet de saisir un nom et l'URL du signet.



Il faut saisir le nom et l'url du fichier site.xml désiré puis cliquer sur le bouton "Fin".



L'application va lire le fichier et affiche les plug-ins disponibles dans l'arborescence. Il suffit de sélectionner la version désirée. La vue "Aperçu" affiche le details du plug-in sélectionné.

Pour installer la version sélectionnée, cliquer sur le bouton "Install" dans la vue « Aperçu ».



Cliquer sur le bouton "Next". Lire et accepter la licence, puis cliquer sur le bouton Next

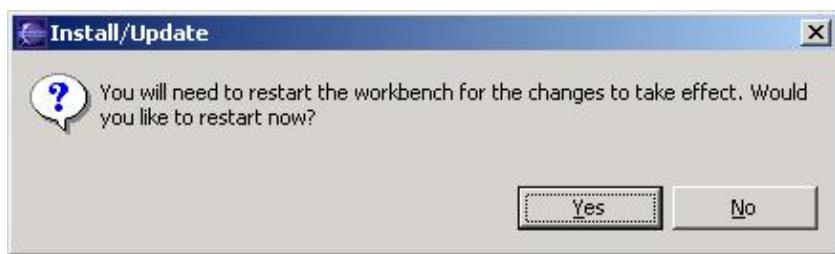


Sélectionner la cible d'installation et cliquer sur le bouton «Finish».

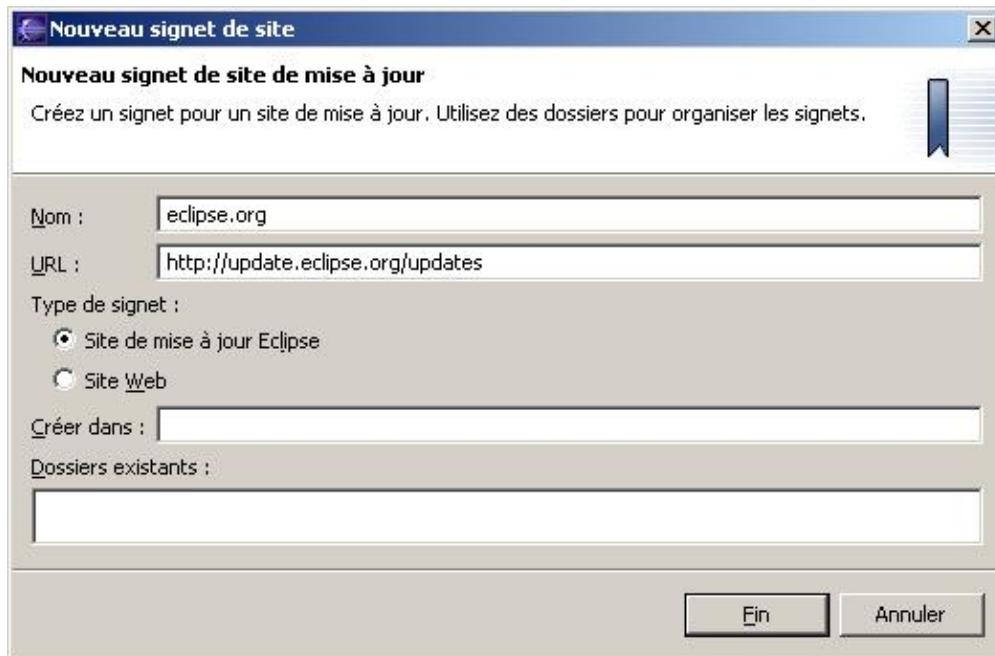


Il faut confirmer l'installation, car le package n'est pas signé, en cliquant sur le bouton « Install »

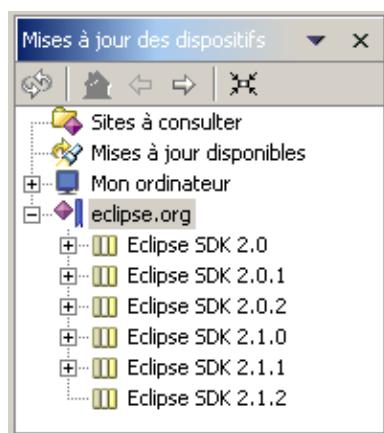
Le plug-in est téléchargé et installé. L'assistant propose de redémarrer le plan de travail pour que les modifications soient prises en compte. Cliquez sur le bouton "Yes".



Il peut être intéressant d'ajouter le site officiel d'Eclipse pour ajouter des plug-ins qui ne sont pas fournis en standard.



Une fois le nouveau signet configuré, il suffit de naviguer dans l'arborescence en fonction de la version de la plate-forme pour pouvoir installer un ou plusieurs plug-ins parmi ceux proposés.



#### 12.3.4. Sauvegarde et restauration d'une configuration

A chaque mise à jour, Eclipse enregistre la configuration.

Il est aussi possible de demander la sauvegarde explicite de la configuration : il faut, dans la perspective

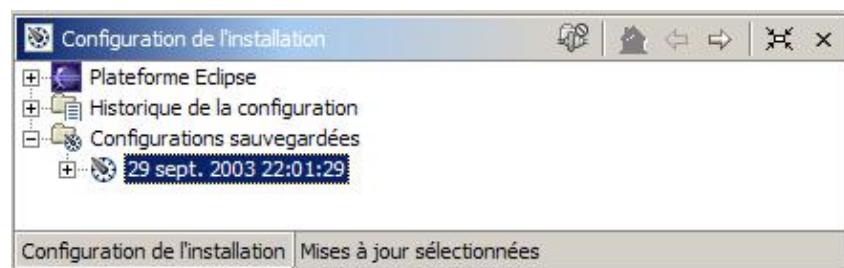
« Installation / Mise à jour », sélectionner « Plate-forme Eclipse » dans la vue « Configuration de l'installation »



Il suffit de cliquer sur le bouton « Sauvegarder » dans la vue « Aperçu »



Pour restaurer une configuration, il suffit de sélectionner, dans la vue « Configuration de l'installation », la sauvegarde automatique (dans « historique de la configuration) ou la sauvegarde manuelle (dans « Configuration sauvegardées»).



Il suffit ensuite de cliquer sur le bouton « Restaurer » de la vue « Aperçu ».

**Aperçu**

**29 sept. 2003 22:01:29**

**Date de création**  
29 sept. 2003 22:01:29

**Configuration actuelle**  
Non

**Activités**  
La liste suivante présente les activités qui ont donné lieu à la création de cette configuration :

Date	Cible	Action
29 sept. 2003 22:23:42	29 sept. 2003 22:01:29	Conservé

**Récupération de la configuration**  
Vous pouvez restaurer l'une des configurations compatibles précédentes à partir de la configuration actuelle. Les configurations sont compatibles si elles ne sont pas séparées par une réconciliation système complète. La restauration ne sera que partielle si certains des dispositifs qui font partie de la configuration souhaitée ont été supprimés depuis.

Pour restaurer cette configuration, cliquez sur : **Restaurer**

**Sauvegarde de la Configuration**  
Les configurations peuvent être sauvegardées de façon à pouvoir être restaurées à une date ultérieure. Les configurations sauvegardées apparaissent dans le dossier **Configurations sauvegardées** dans la vue **Configuration de l'installation**.

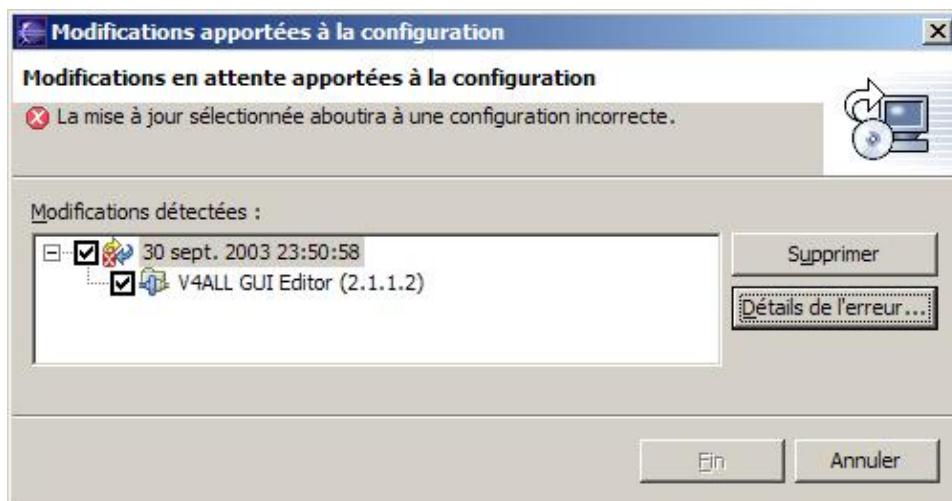
Pour sauvegarder cette configuration, cliquez sur : **Sauvegarder**

Cette opération nécessite un redémarrage de l'application.

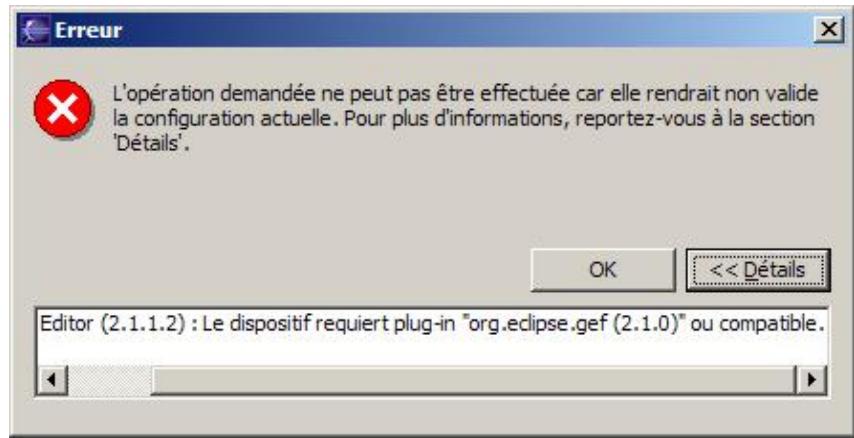
### 12.3.5. Résolution des problèmes de dépendances

Il est possible que la mise à jour ou l'installation d'un plug-in échoue notamment pour des questions de dépendances vis-à-vis d'un autre plug-in.

Dans ce cas, la mise à jour est signalée avec une croix blanche dans un rond rouge.

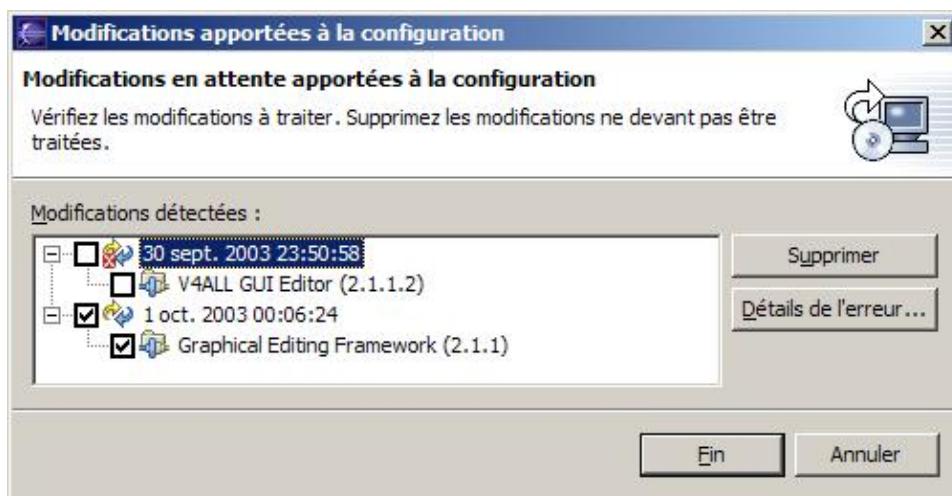


Pour avoir plus d'informations, il suffit de sélectionner la mise à jour et de cliquer sur le bouton « Détails de l'erreur ».

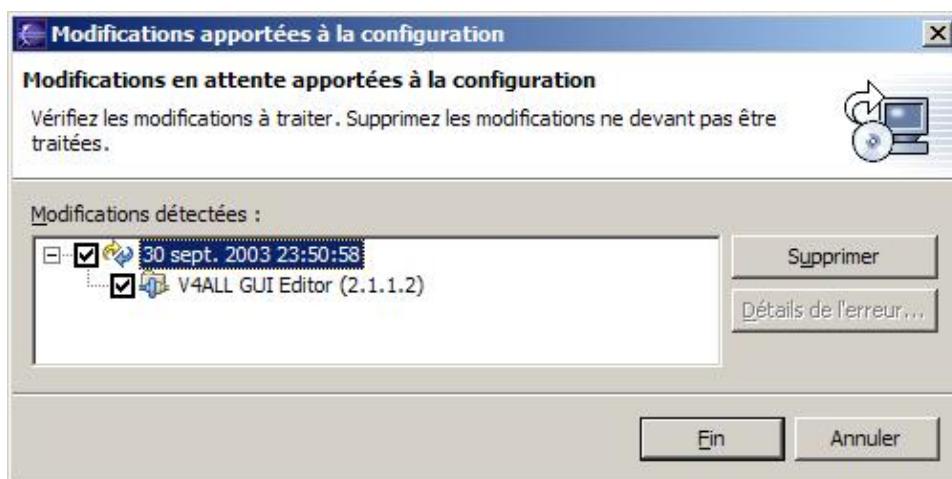


Dans l'exemple ci-dessus, il manque un plug-in ou la version installée n'est pas correcte.

Pour résoudre le problème, il suffit de quitter Eclipse, d'installer le plug-in (en le décompressant dans le bon répertoire) et de relancer Eclipse.



Il suffit alors de décocher la mise à jour posant problème et de cliquer sur le bouton « Fin » afin de mettre à jour le plug-in manquant. Cette opération nécessite le redémarrage d'Eclipse.



Suite à ce redémarrage, le plug-in erroné n'apparaît plus en erreur et peut être installé en cliquant sur le bouton « Fin ».

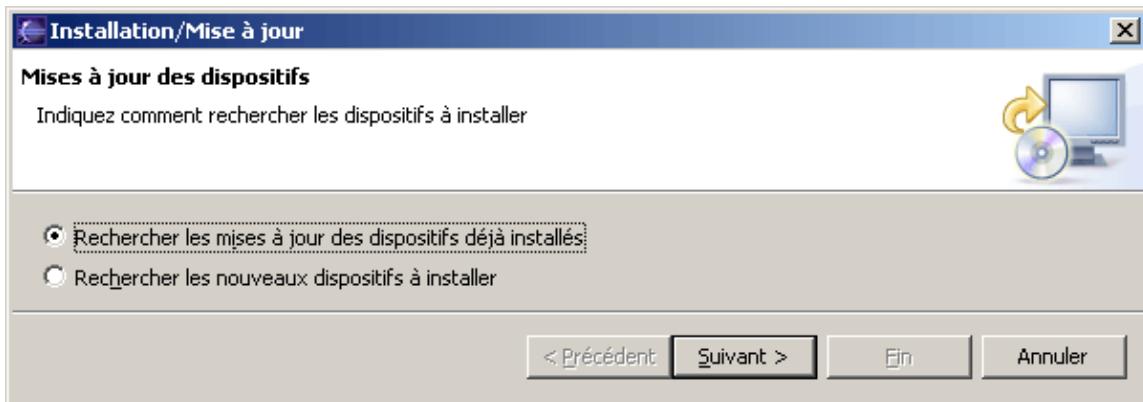
## 12.4. La mise à jour des plug-ins avec Eclipse 3.0

La perspective "Installation et mise à jour" est supprimée et l'option "Mise à jour de logiciels" du menu "Aide" propose maintenant deux options :

- "Rechercher et installer ..."
- "Configuration du produit ..."

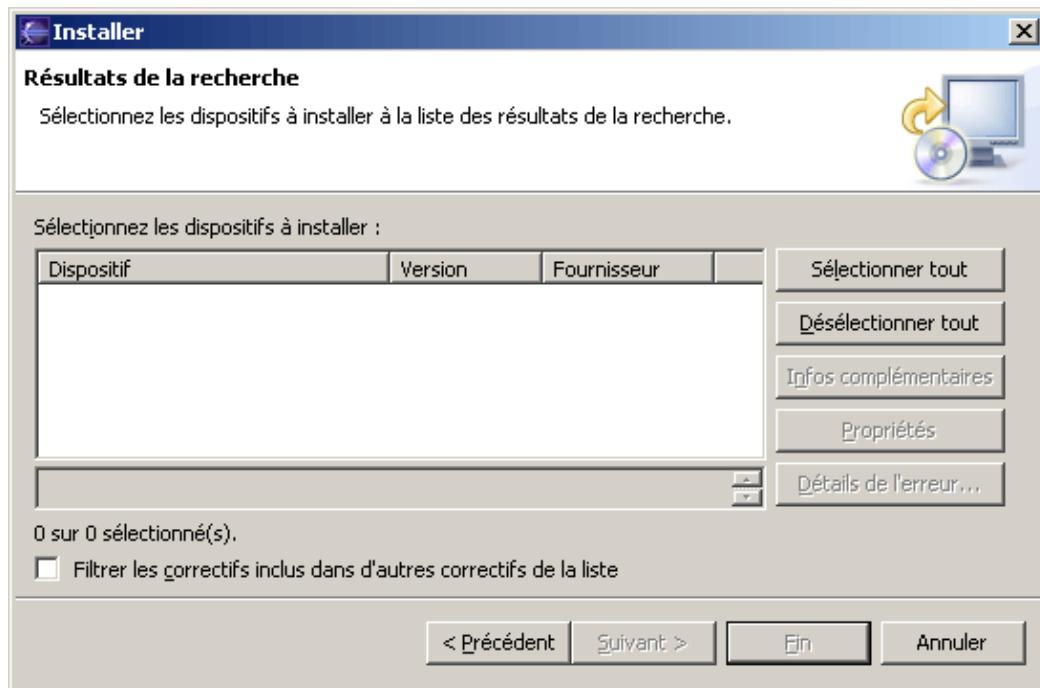
### 12.4.1. Recherche et installation de plug-ins

L'option "Rechercher et installer ..." permet de rechercher et d'installation un plug-in ou une nouvelle version d'un plug-in installé grâce à un assistant.

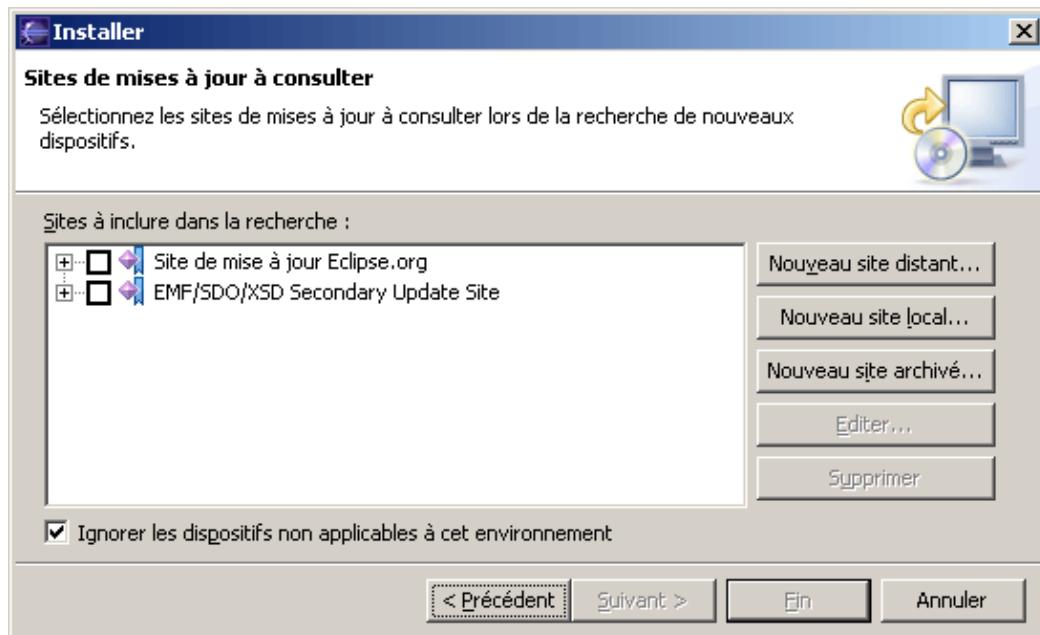


La première page permet de sélectionner le type d'élément à rechercher.

L'option "Rechercher les mises à jour des dispositifs déjà installés" permet de rechercher dans les sites enregistrés, des nouvelles versions des plug-ins installés.

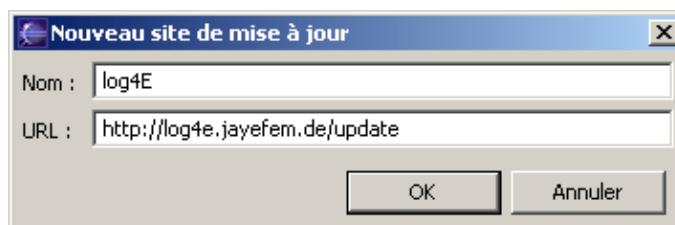


L'option "Rechercher les nouveaux dispositifs à installer" permet de gérer la liste des sites proposant des plug-ins et ainsi d'obtenir de nouveaux plug-ins.



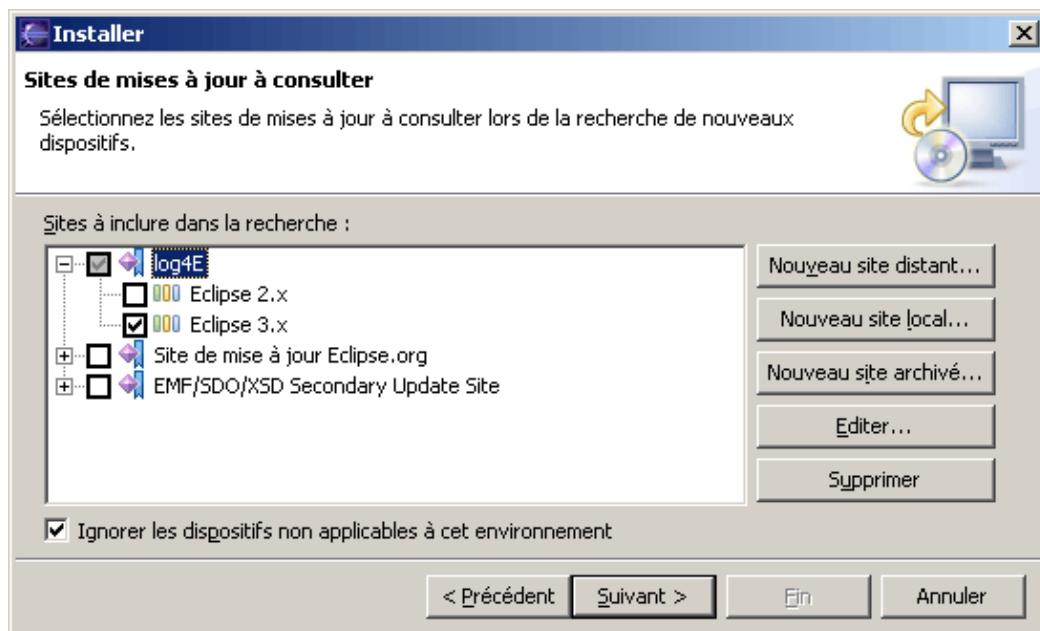
Cette page permet de sélectionner les sites concernés par la recherche. Il est possible d'ajouter de nouveaux sites grâce aux boutons de droite.

Un clic sur le bouton "Nouveau site distant ..." ouvre une boîte de dialogue qui permet de saisir les informations le nouveau site. Il suffit de saisir un nom et l'url du site.

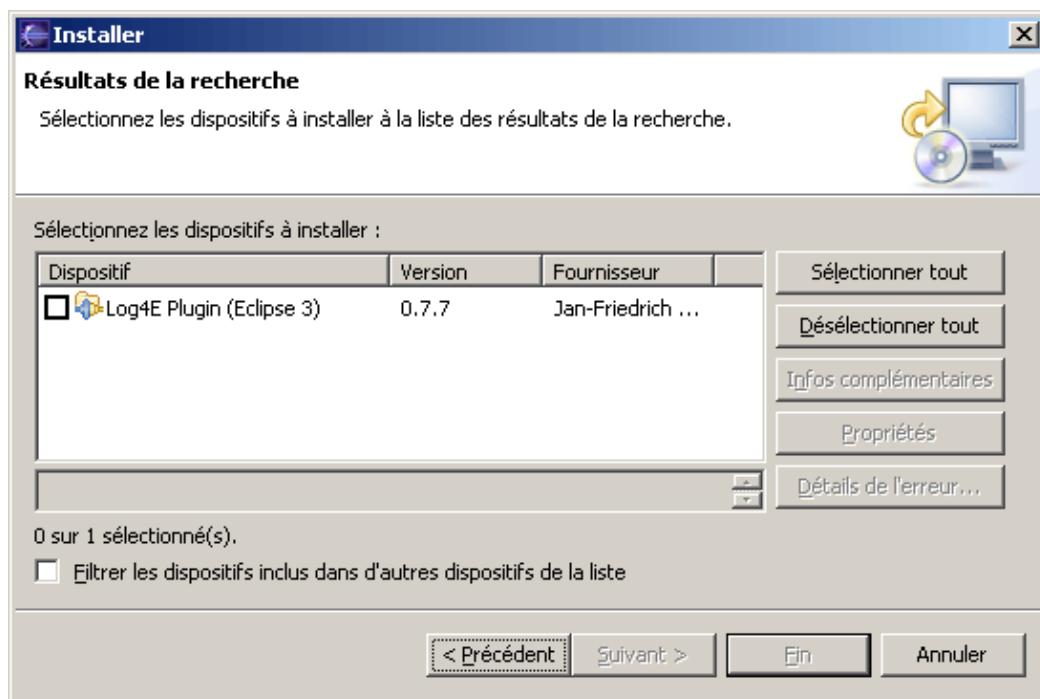


Cliquez sur le bouton "OK"

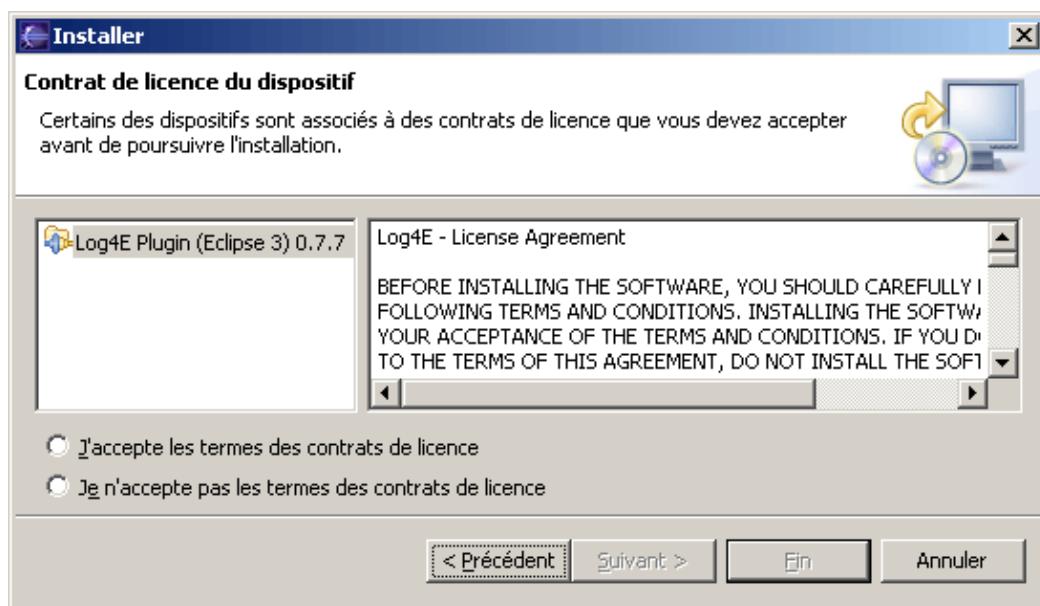
Le nouveau site est ajouté dans la liste. Il faut ouvrir l'arborescence du nouveau site pour sélectionner les dispositifs désirés.



Un clic sur le bouton "Suivant" permet de lancer la recherche. Les résultats de la recherche sont affichés dans la page suivante.



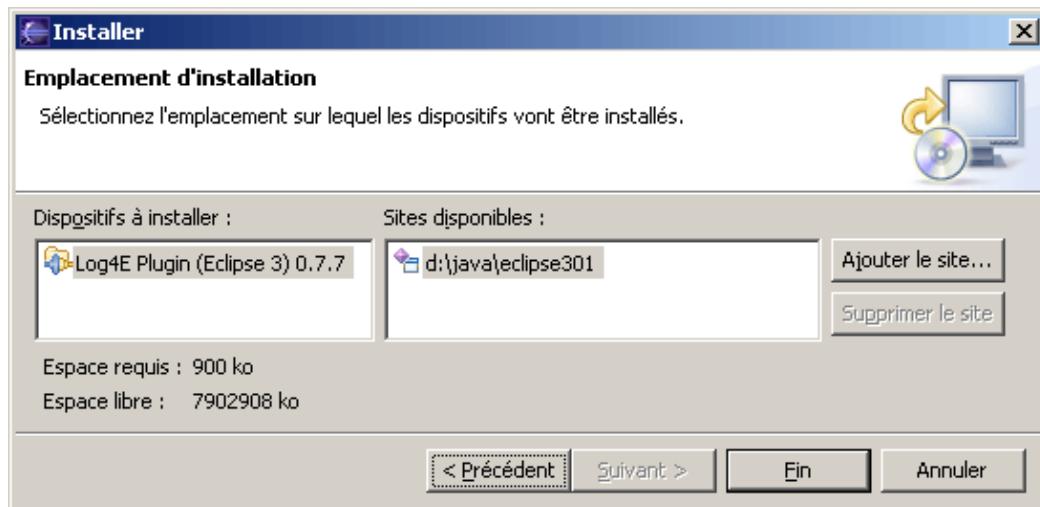
Il suffit alors de sélectionner le ou les plug-ins désirés puis de cliquer sur le bouton "Suivant".



La page suivante permet de lire et d'accepter la licence pour poursuivre les traitements.

Si vous acceptez la licence, cliquez sur le bouton "J'accepte les termes des contrats de licence" puis sur le bouton "Suivant".

La page suivante permet de sélectionner l'emplacement de l'installation du plug-in



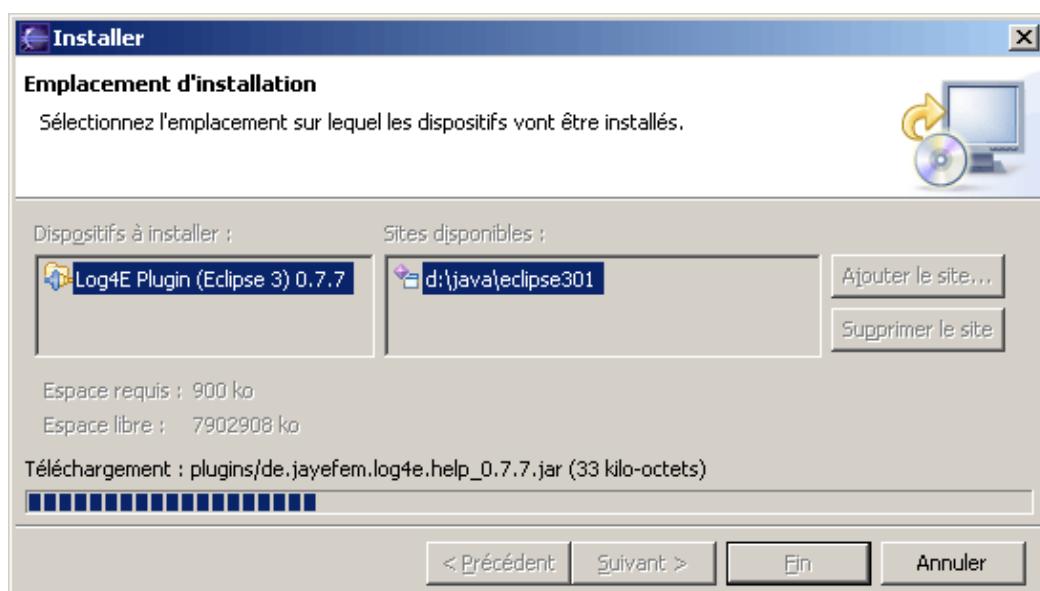
Cliquez sur le bouton "Fin".

Si le plug-in n'est pas signé, une demande de confirmation d'installation est demandée

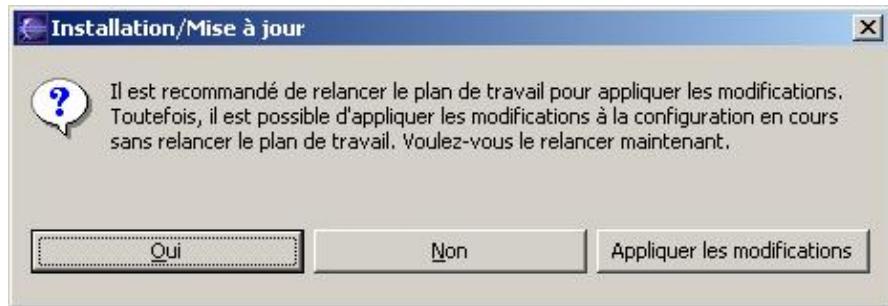


Dans ce cas, cliquez sur le bouton "Installer".

Les fichiers sont téléchargés et installés.



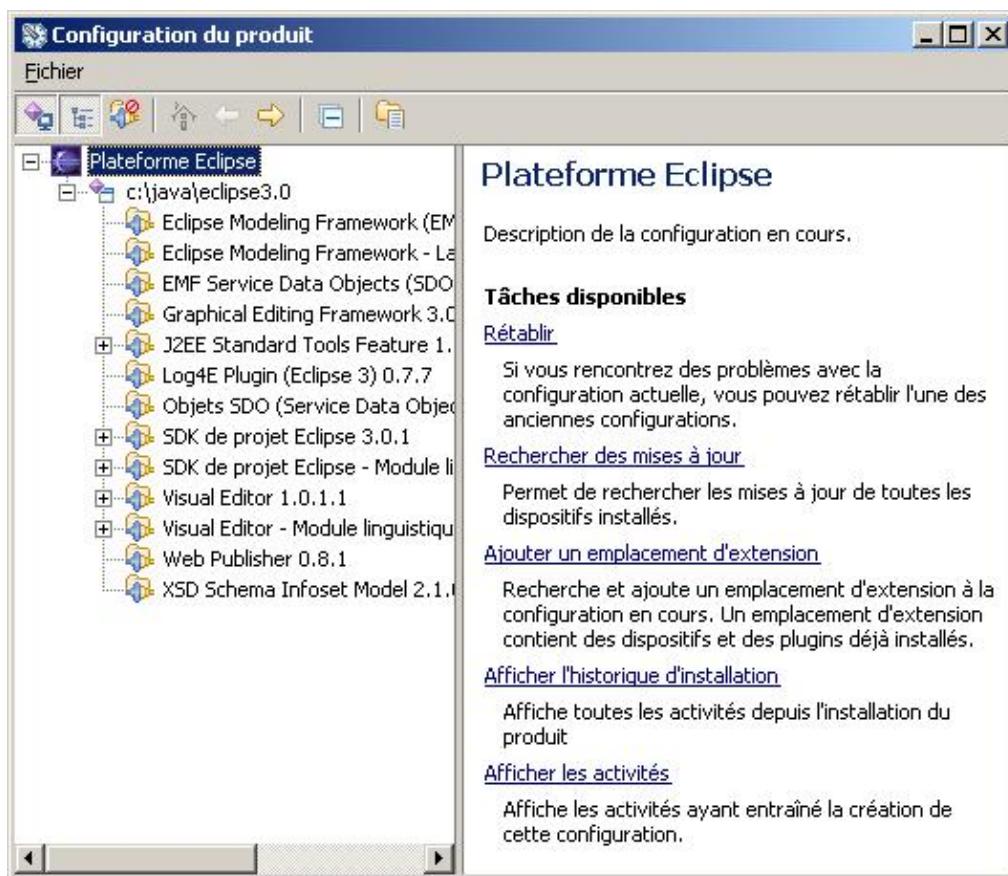
Une fois ces opérations terminées, le système recommande de redémarrer la plate-forme.



Il est donc recommandé de cliquer sur le bouton "Oui" surtout pour pouvoir utiliser immédiatement le plug-in fraîchement installé.

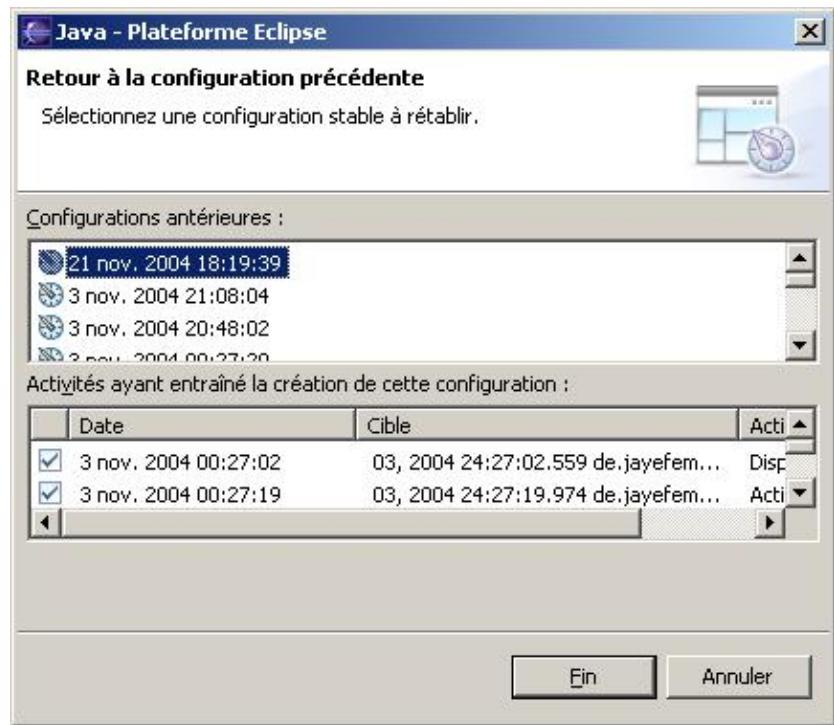
#### 12.4.2. La configuration du produit

L'option Configuration du produit permet une gestion détaillée de la plate-forme.



En sélectionnant la racine de l'arborescence "Plateforme Eclipse", plusieurs tâches sont disponibles.

La tâche "Rétablir" permet de restaurer une ancienne configuration.



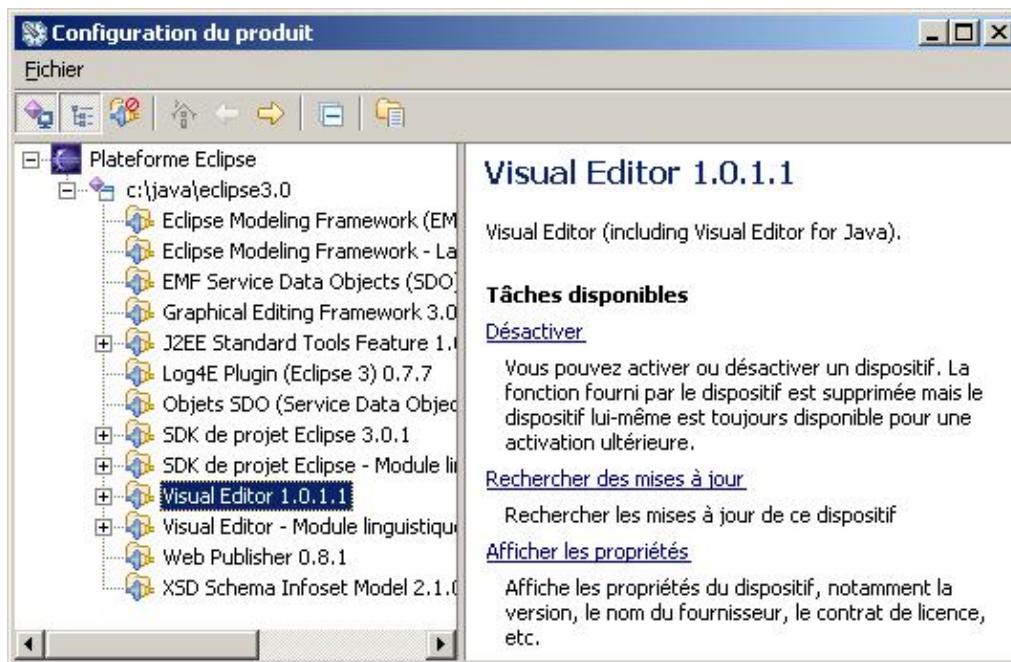
Il suffit de sélectionner la configuration désirée et de cliquer sur le bouton "Fin".

La tâche "Afficher l'historique d'installation" affiche une page web contenant l'historique des installations. Le bouton permet aussi de demander l'exécution de cette tâche.

The screenshot shows a Mozilla Firefox window titled 'Install-Log - Mozilla Firefox'. The main content is a page titled 'Historique d'installation'. The page text states: 'La liste ci-dessous représente l'historique des opérations de mise à jour depuis l'installation du produit. Chaque opération est associée à une date et aux activités effectuées. Les opérations sont triées par date et par ordre croissant.' Below this is a table of installation history:

Date / Heure	Cible	Action	Statut
20 août 2004 22:27:58	file:/c:/java/eclipse3.0/	site-install	success
<b>20 août 2004 22:48:00</b>			
20 août 2004 22:47:20	de.jayefem.log4e_0.7.5	feature-install	success
20 août 2004 22:48:00	de.jayefem.log4e_0.7.5	feature-enable	success
<b>3 nov. 2004 00:04:08</b>			
2 nov. 2004 23:49:20	02, 2004 23:49:20.186 org.eclipse.emf_2.0.1	feature-install	success
2 nov. 2004 23:57:43	02, 2004 23:57:43.449 org.eclipse.emf_2.0.1	feature-enable	success
Terminé			

Pour chaque plug-in installé, il est possible de réaliser plusieurs tâches en le sélectionnant :

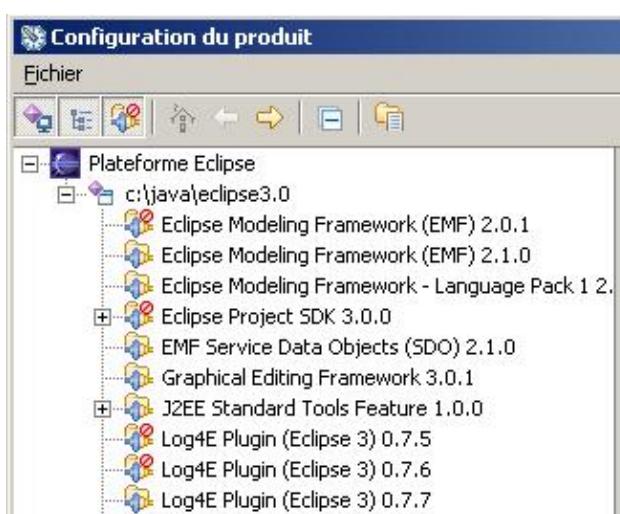


La tâche "Désactiver" permet désactiver le plug-in après une confirmation



Il faut redémarrer la plateforme que la désactivation soit effective.

Par défaut, les plug-ins désactivés ne sont pas affichés. Il faut cliquer sur le bouton pour les afficher dans l'arborescence.



La tâche "Activer" permet de réactiver le plug-in après une configuration. La relance de la plate-forme n'est pas obligatoire mais fortement recommandée.

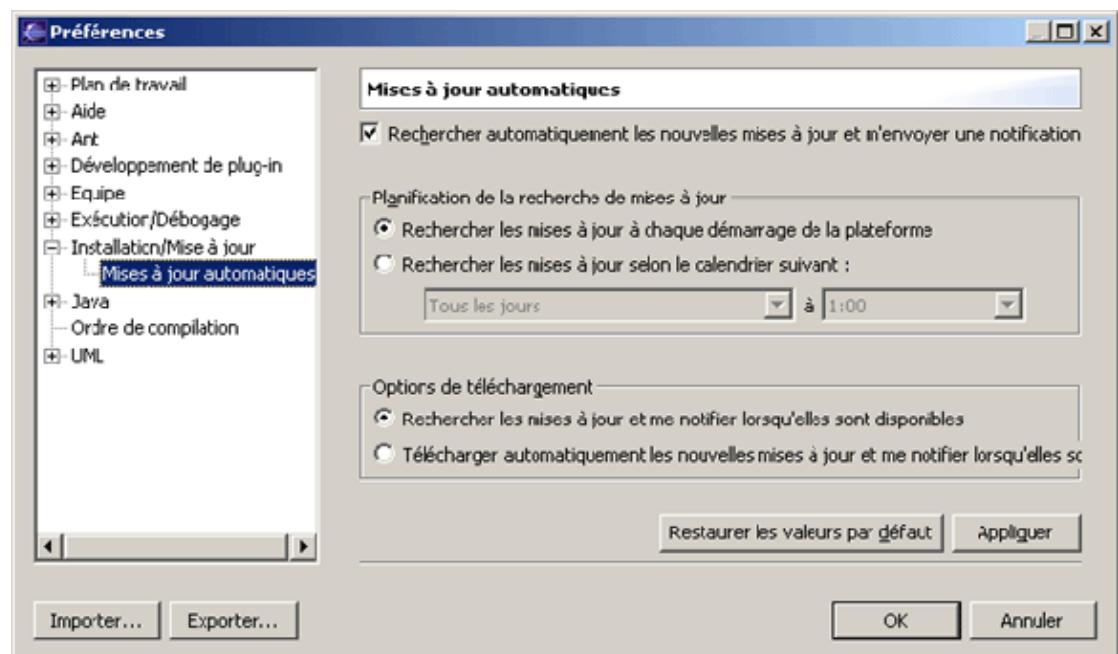
La tâche "Désinstaller" permet de désinstaller le plug-in après confirmation



La relance de la plate-forme n'est pas obligatoire mais fortement recommandée.

### 12.4.3. Mises à jour automatiques

 Il est possible de paramétriser une fréquence de mises à jour automatiques. Pour cela dans les préférences, il faut cocher l'option "Rechercher automatiquement les nouvelles mises à jour et m'envoyer une notification"

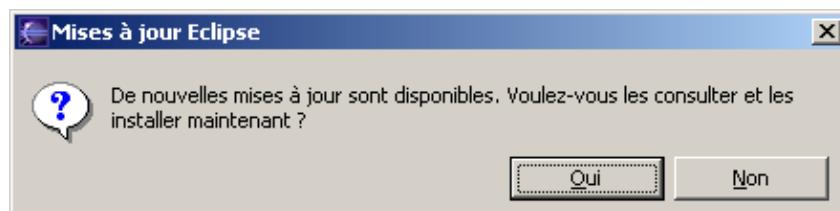


Il est alors possible de paramétriser la planification et les options de téléchargement pour une mise à jour automatique des éléments enregistrés dans le système de mise à jour de l'application.

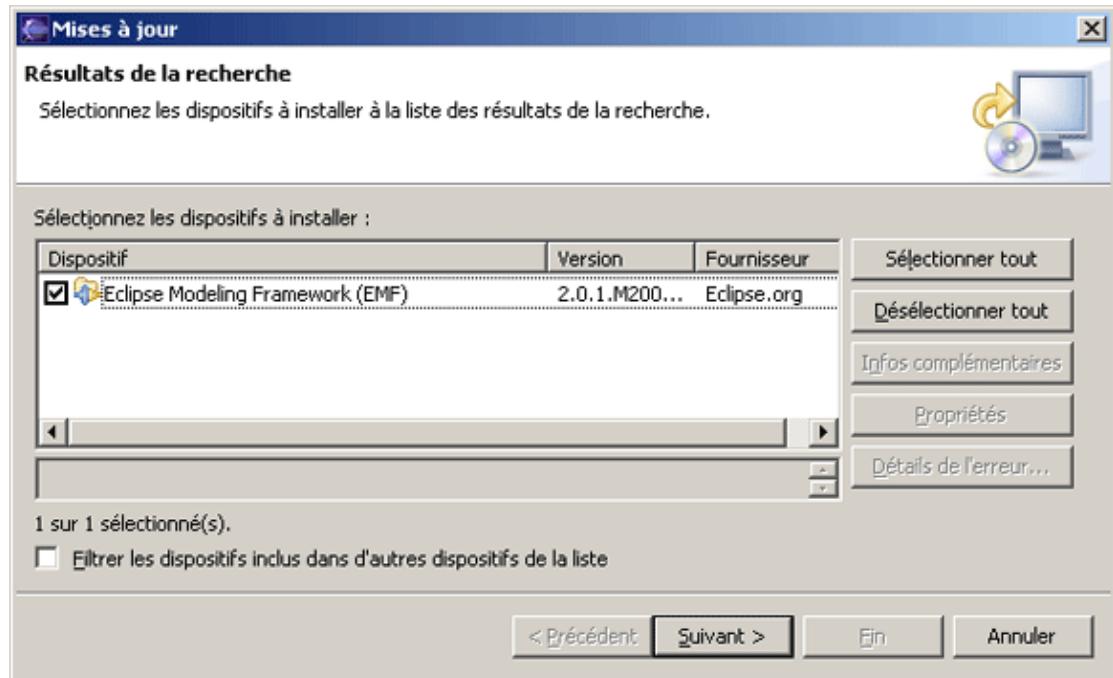
Il est ainsi possible de préciser deux choses :

- le moment de la recherche automatique : soit à chaque démarrage de l'application soit à une périodicité et une heure précise.
- informer de la disponibilité de mises à jour ou télécharger automatiquement les mises à jour trouvées et informer de la fin de leur téléchargement

Si le système détecte des mises à jour, une boîte de dialogue est affichée à l'issu des traitements de recherche :



Un clic sur le bouton "Oui" permet d'afficher une boîte de dialogue recensant les mises à jour trouvées.



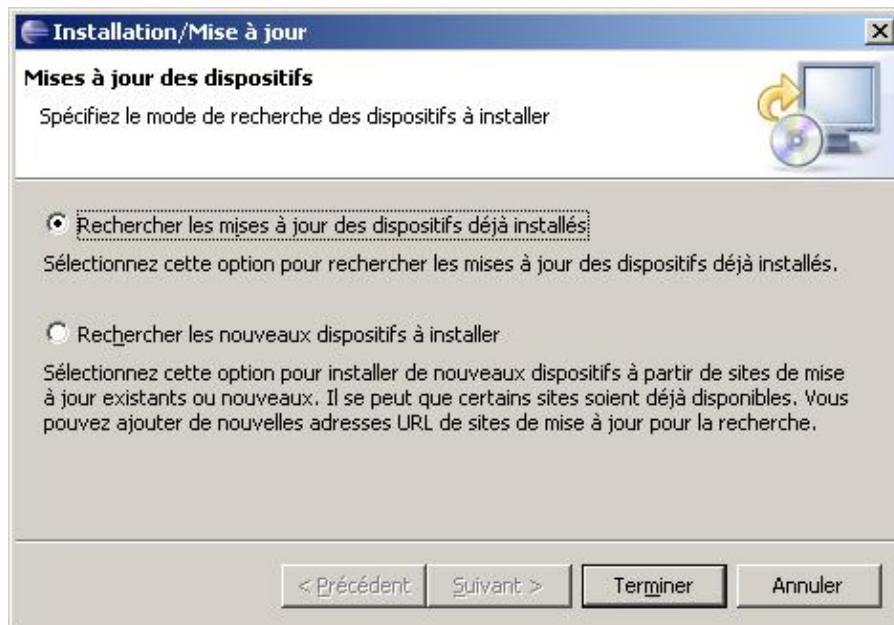
Il faut sélectionner le ou les éléments à mettre à jour et de cliquer sur le bouton "Suivant".

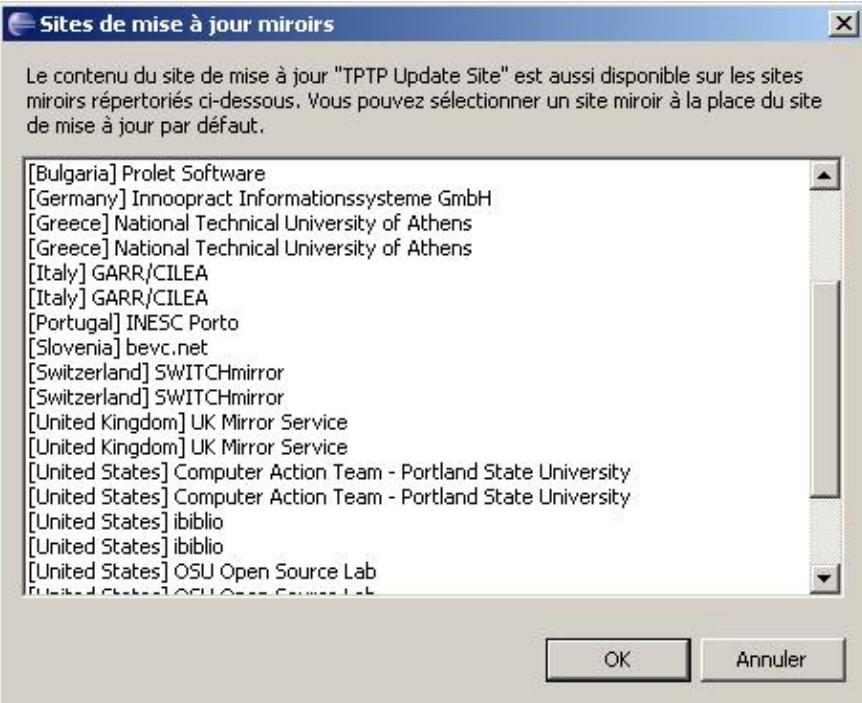
Il suffit enfin de suivre les étapes de l'assistant pour obtenir et installer le ou les dispositifs choisis.

#### 12.4.4. Mise à jour de la plate-forme via des miroirs



La mise à jour de la plate-forme se faire à partir de miroir. Lors d'une demande mise à jour, une boîte de dialogue permet de sélectionner le miroir à utiliser.





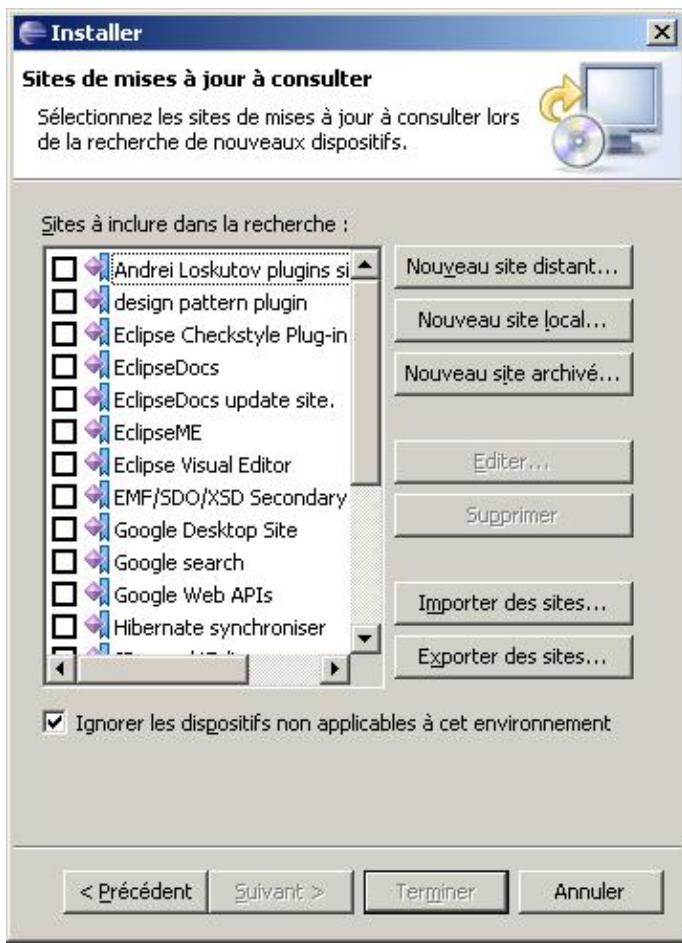
Sélectionnez le miroir et cliquez sur le bouton « OK »



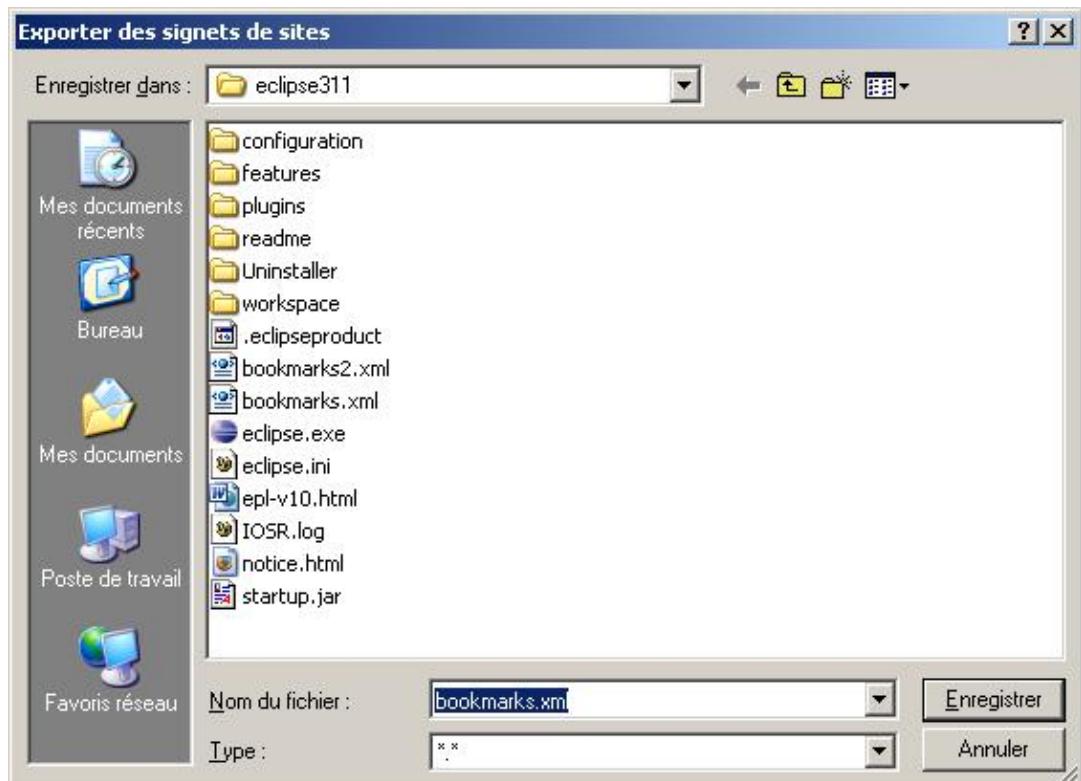
Si aucune mise n'est trouvée alors un message est affiché



Il est possible d'exporter/importer une liste des signets de sites de mise à jour



En cliquant sur le bouton « Exporter des sites », une boîte de dialogue permet de sélectionner le fichier qui va contenir la liste.



Le fichier créé est au format xml.

### Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<bookmarks>
    <site name="Log4E" url="http://log4e.jayefem.de/update"
        web="false" selected="true" local="false"/>
    <site name="Hibernate synchroniser"
        url="http://www.binamics.com/hibernatesync"
        web="false" selected="true" local="false"/>
    <site name="JFaceBDC"
        url="http://jfacedbc.sourceforge.net/update-site/site.xml"
        web="false" selected="true" local="false"/>
    <site name="QJ-Pro"
        url="http://qjpro.sourceforge.net/update_site"
        web="false" selected="true" local="false"/>
    <site name="EclipseME"
        url="http://eclipseme.org/updates/"
        web="false" selected="true" local="false"/>
    <site name="Eclipse.org update site"
        url="http://update.eclipse.org/updates/3.0"
        web="false" selected="false" local="false"/>
</bookmarks>
```

Pour importer une liste, il suffit de cliquer sur le bouton « Importer des sites » et de sélectionner le fichier contenant la liste.

## 13. CVS 2.0 et Eclipse 2.1

# Chapitre 13

CVS (Concurrent Versions System) est un outil libre de gestion de versions. Initialement développé pour Unix, une version pour windows NT/2000 de CVS peut être téléchargée à l'url <http://www.cvsnt.org/>

Toutes les données sont stockées dans un référentiel (repository). Chaque modification d'une ressource gérée par CVS associe à cette entité un numéro de révision unique.

Une version contient un ensemble de ressource, chacune ayant une révision particulière pour la version correspondante.

CVS ne verrouille pas ces ressources. Deux développeurs peuvent créer chacun une révision d'une même ressource. La fusion des deux versions est à la charge d'un des développeurs.

Eclipse propose une perspective pour utiliser CVS dans un projet.

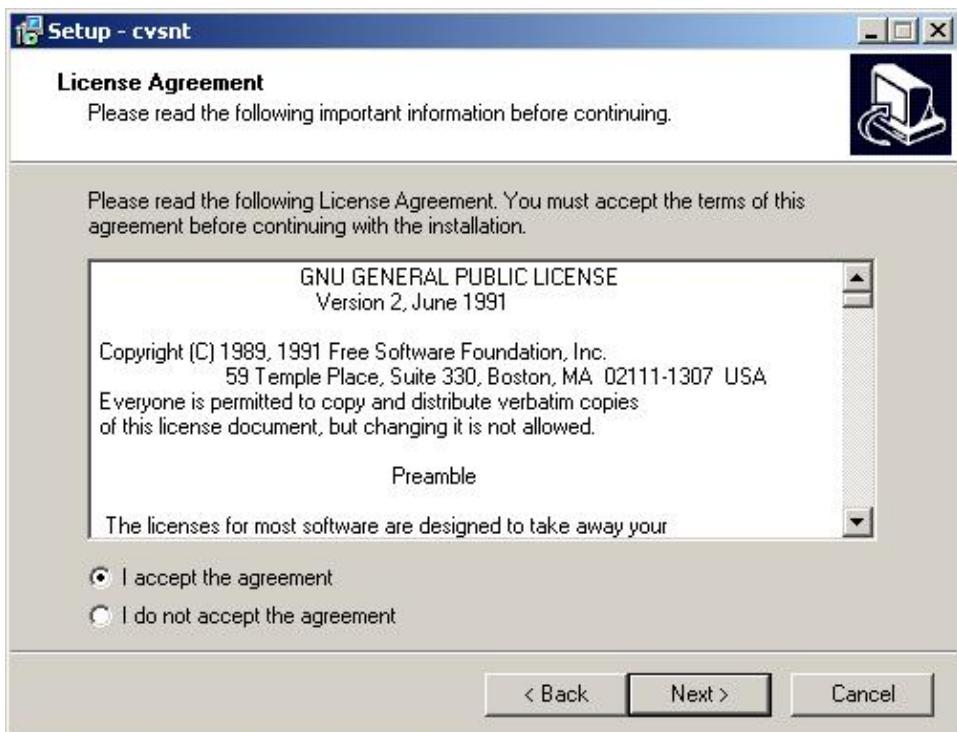
### 13.1. Installation de cvsnt

Il faut créer deux répertoires, par exemple c:\cvs\cvsrepo et c:\cvs\cvstemp

Exécuter le programme d'installation cvsnt-2.0.0.exe



Cliquez sur le bouton «Next»

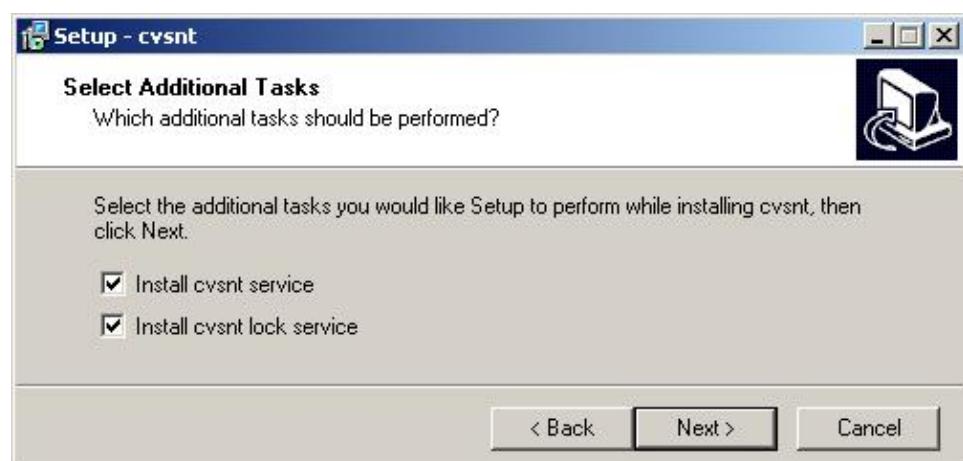
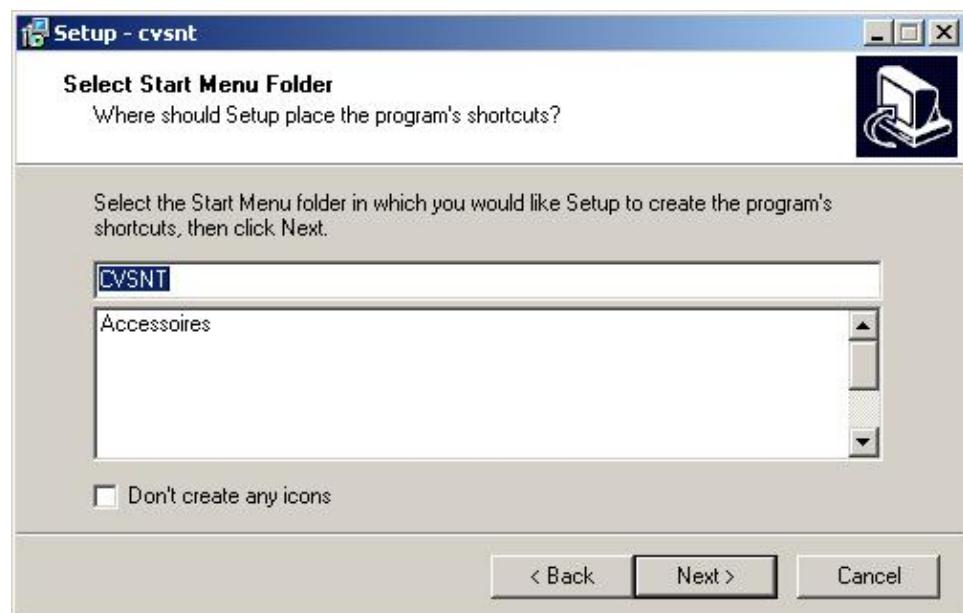
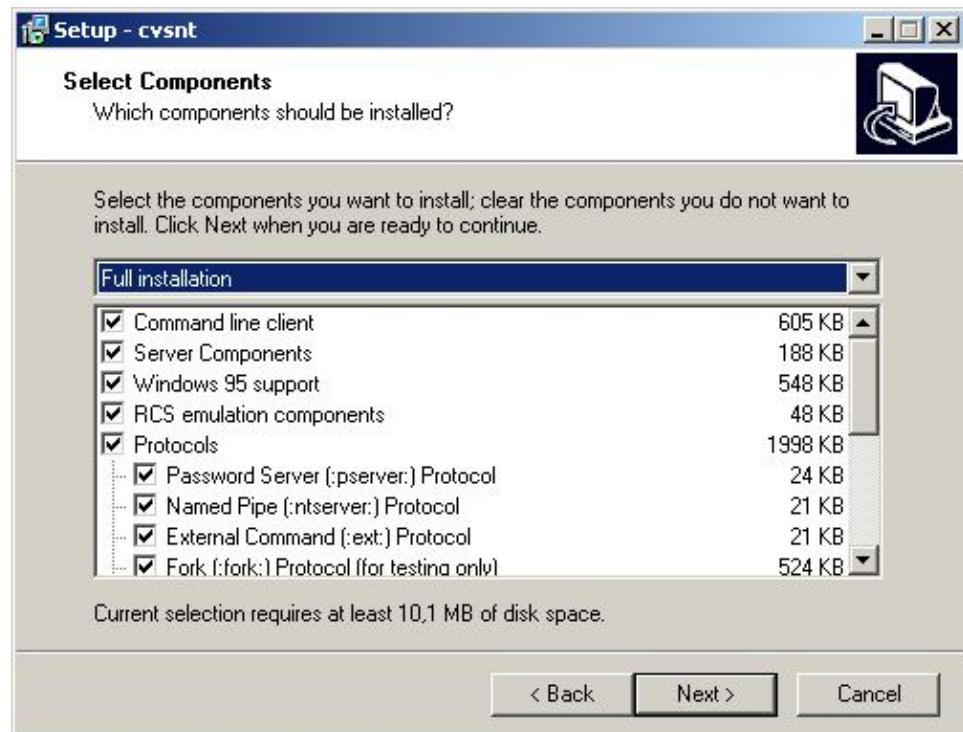


Lisez la licence et si vous l'acceptez, cliquez sur le bouton «Next»

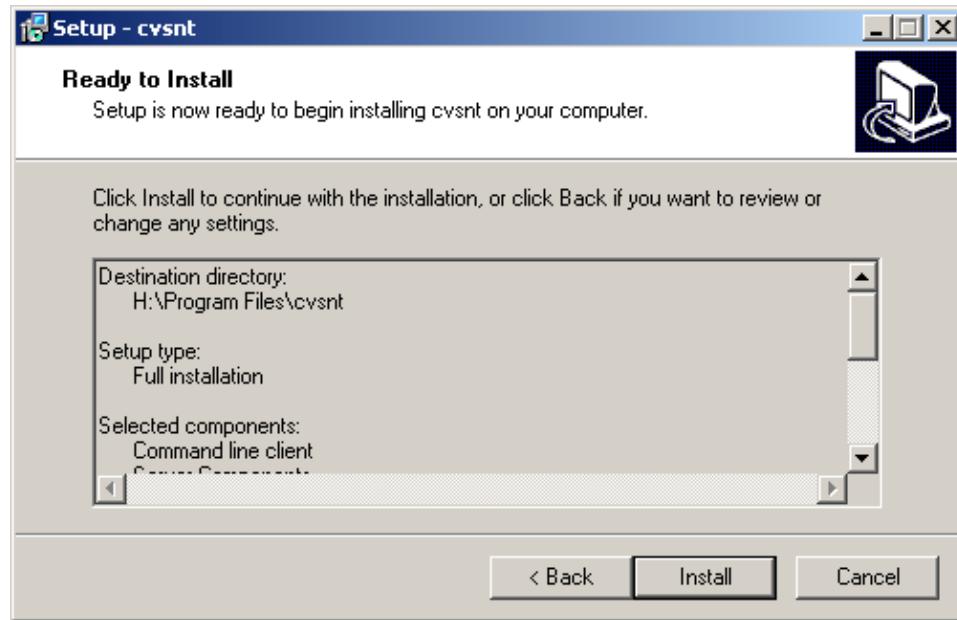


Sélectionnez le répertoire d'installation et cliquez sur le bouton «Next»

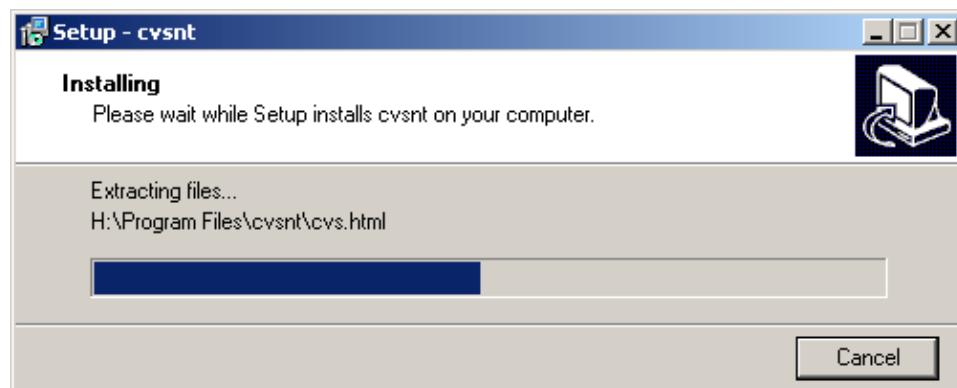




Cliquez sur le bouton «Next»



Cliquez sur le bouton «Install»



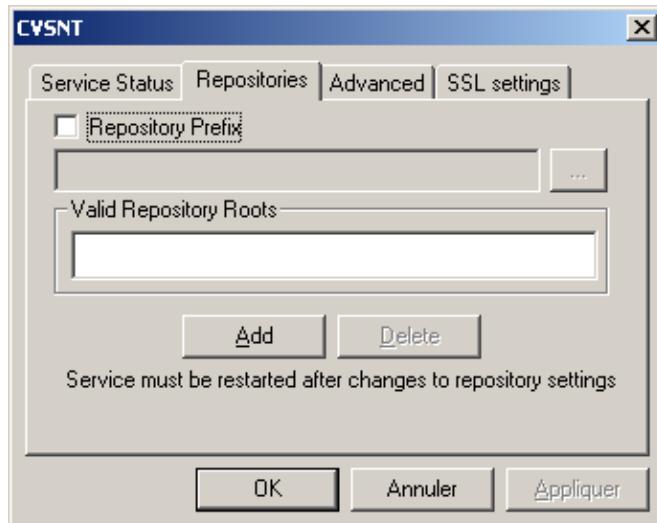
Une fois l'installation terminée, cliquez sur le bouton «Finish».



Exécutez «le service control panel» en cliquant sur l'icône CVS for NT dans le menu "Démarrer/Programmes/CVTNT" ou dans le panneau de configuration

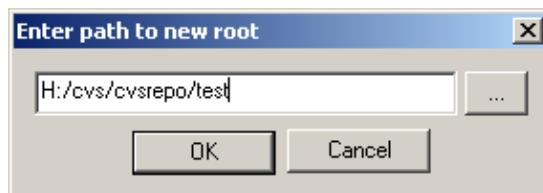


Sélectionnez l'onglet «Repositories»

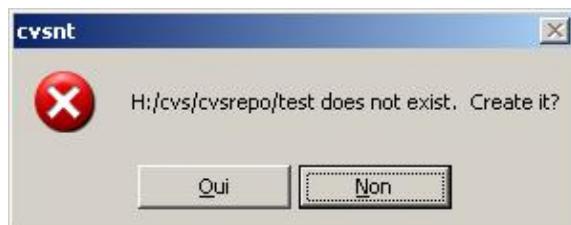


Cliquez sur «Repository prefix» et sélectionnez le répertoire cvsrepo précédemment créé.

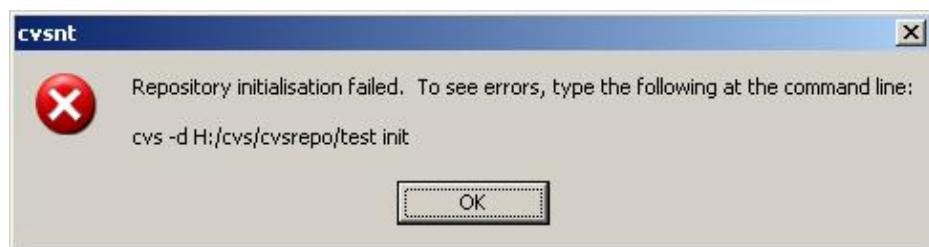
Cliquez sur le bouton «Add»



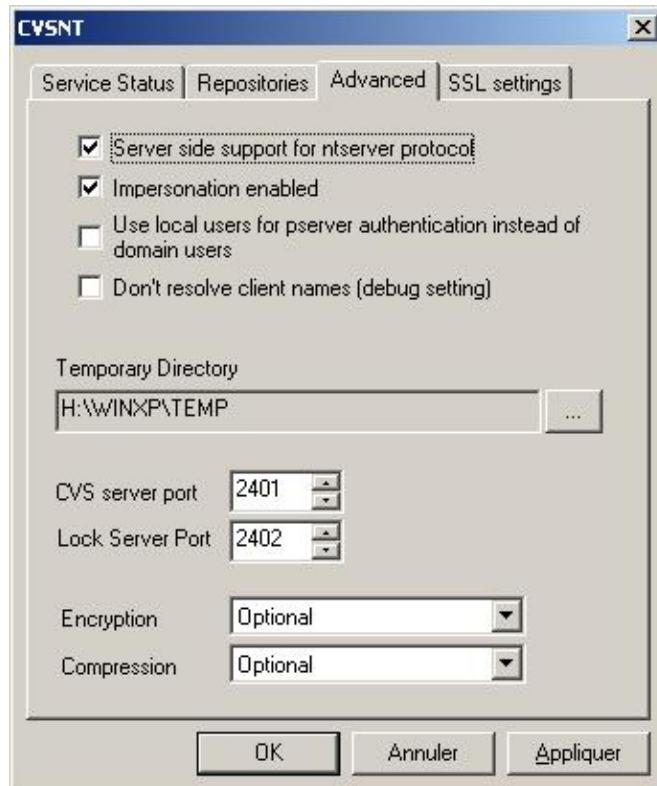
Saisissez le nom du répertoire et cliquez sur le bouton «OK»



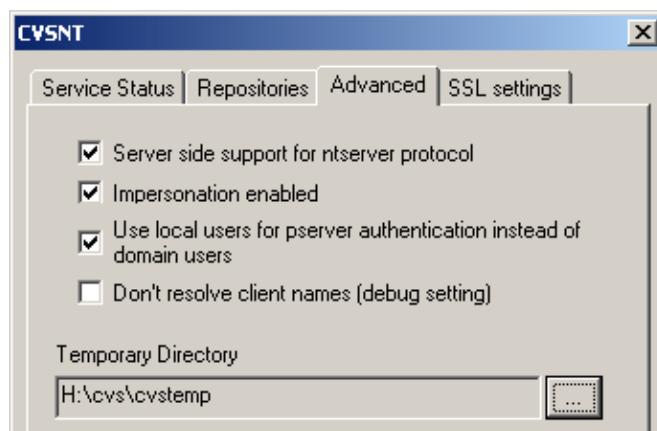
Cliquez sur le bouton «Oui»



Sélectionnez l'onglet «Advanced» et cliquez sur le bouton "..."



Sélectionner le répertoire temporaire précédemment créé



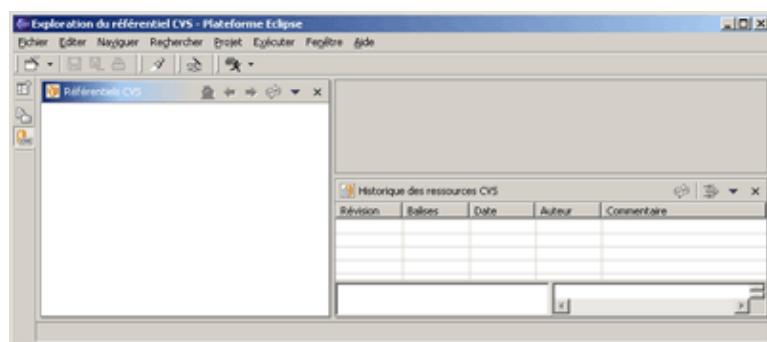
Sur l'onglet «Service Status», cliquez sur le bouton «Start»



Cliquez sur le bouton Ok

## 13.2. La perspective CVS

La perspective «Exploration du référentiel CVS» permet de gérer les échanges et le contenu des projets stockés sous CVS.

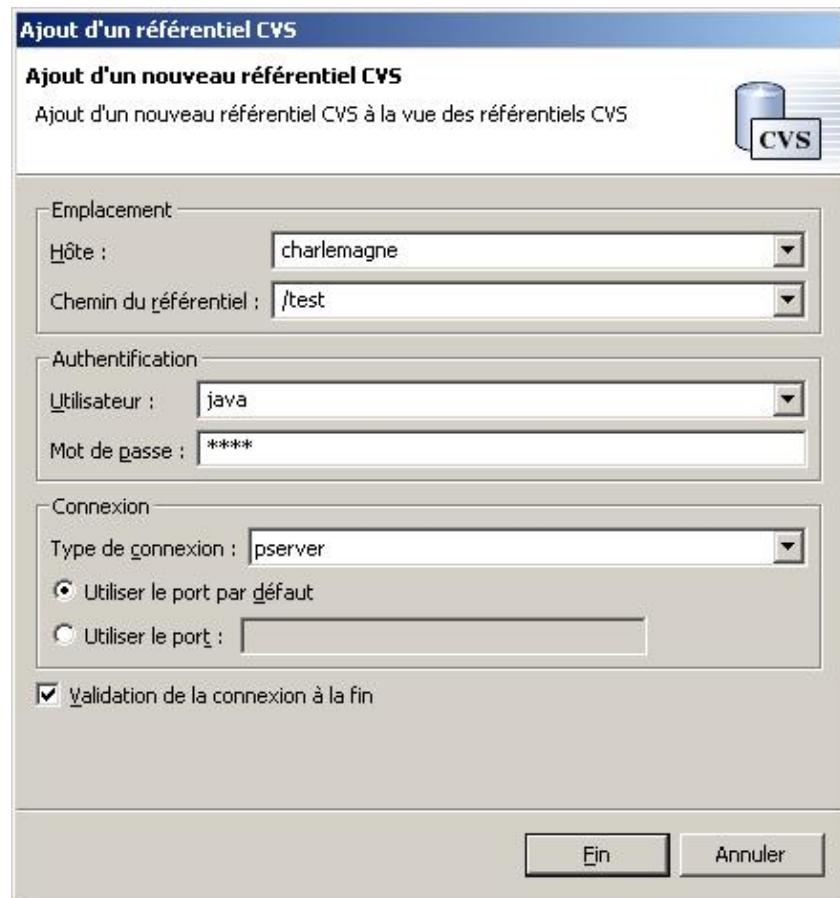


### 13.2.1. La création d'un emplacement vers un référentiel

Dans la vue "Référentiels CVS", sélectionner l'option "Nouveau/Emplacement du référentiel" du menu contextuel.



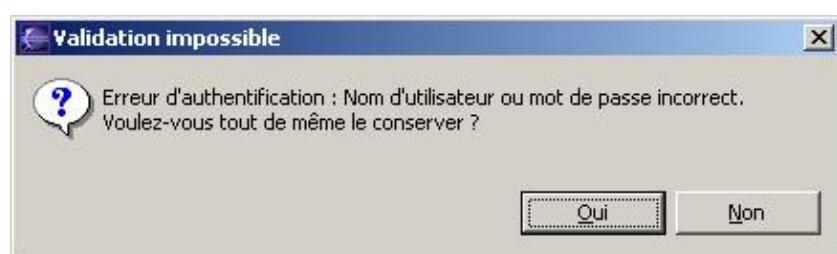
Une boîte de dialogue s'ouvre pour définir un nouvel emplacement. Un emplacement contient uniquement les informations sur une connexion.



Il faut renseigner le nom de la machine, le chemin du référentiel, le nom de l'utilisateur, son mot de passe (celui de windows) et le type de connexion (utilisez pserver) puis cliquer sur le bouton "Fin"

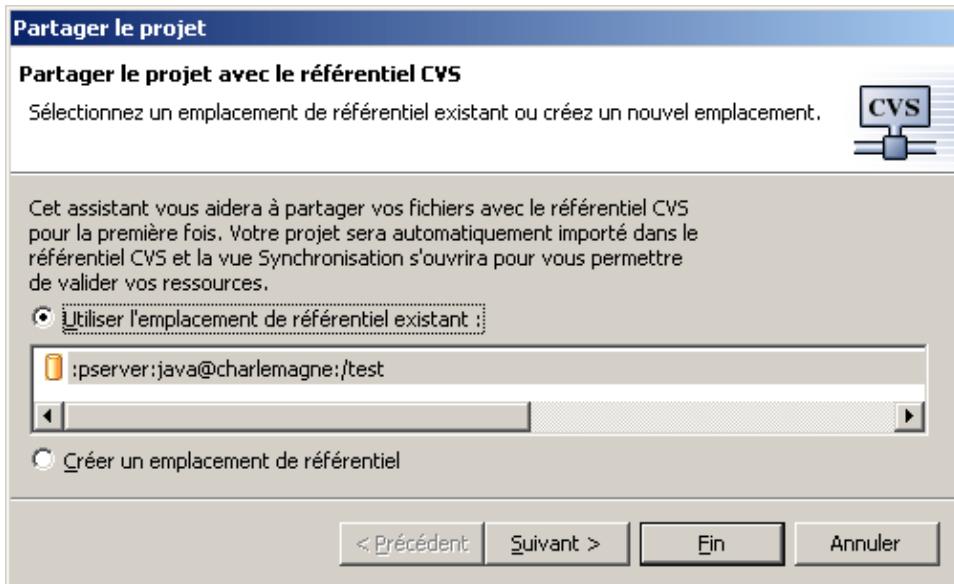


Si la connexion ne peut être établie, un message d'erreur est affiché

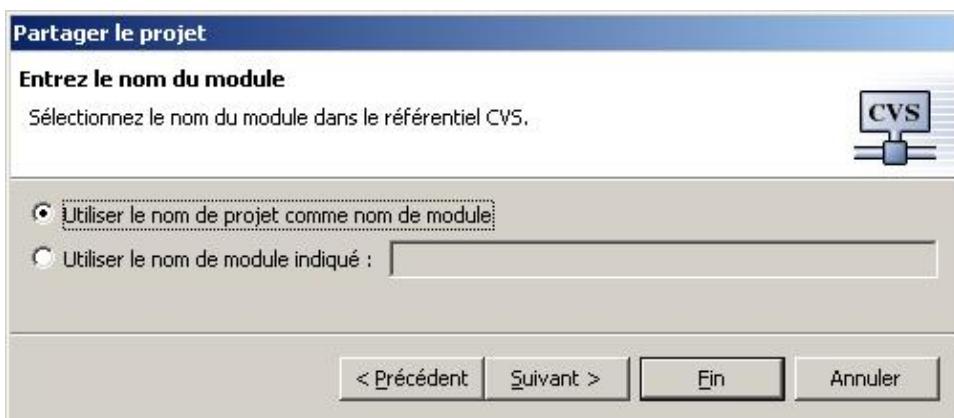


### 13.2.2. Partager un projet

Il faut sélectionner un projet dans une vue et sélectionner l'option "Equipe/Partager le projet" du menu contextuel.



Cliquez sur le bouton "Suivant".



Cette étape permet de donner un nom au module : soit celui du projet Eclipse soit un nom spécifique à préciser. Cliquez sur le bouton "Suivant".

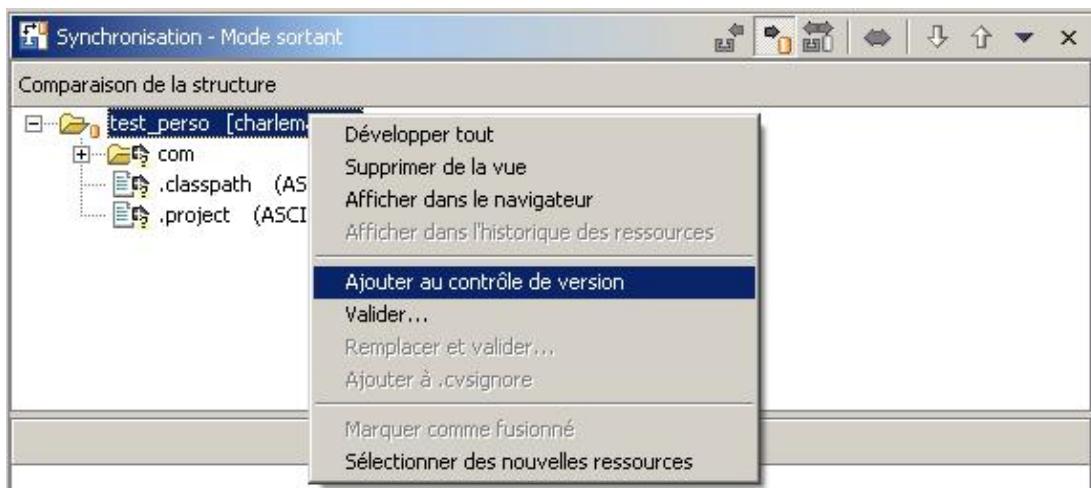


Cliquez sur le bouton "Fin".

La vue "Synchronisation - Mode sortant" affiche les fichiers qui ont été modifiés

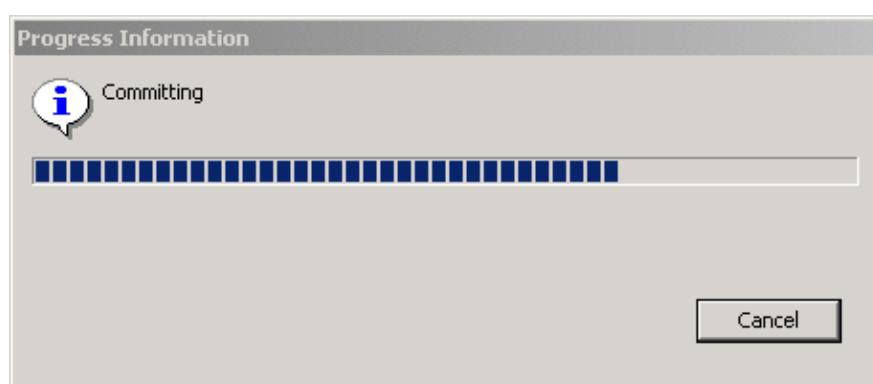


Dans cette vue, il faut sélectionner le projet et activer l'option "Ajouter au contrôle de version" du menu contextuel.



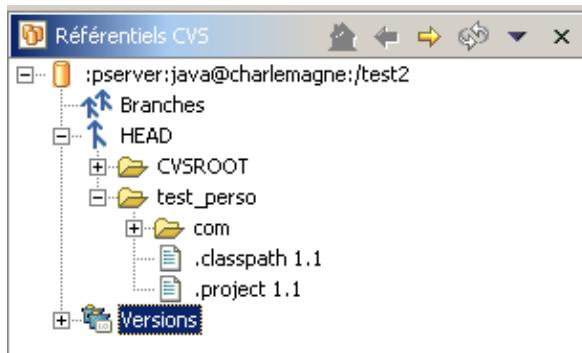
Une fois le traitement effectué, activer l'option "Valider" du menu contextuel

Une boîte de dialogue demande la saisie d'un commentaire



### 13.2.3. Voir le projet dans la perspective CVS

Pour voir le projet, il faut rafraîchir la vue «Referentiel CVS» en cliquant sur le bouton ou en activant l'option "Régénérer la vue" dans le menu contextuel.



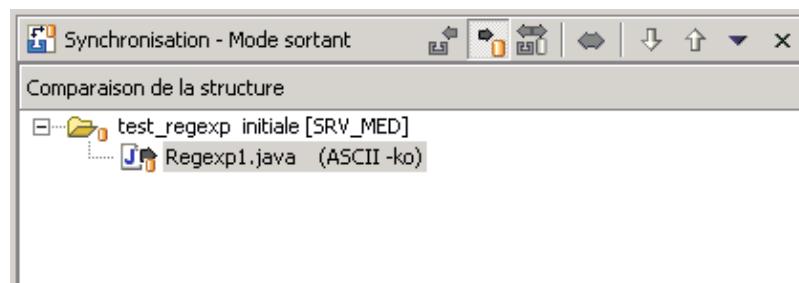
Remarque : si l'arborescence du projet n'est pas affichée, il suffit de cliquer sur le bouton et de cocher l'option "Afficher les dossiers".

### 13.3. L'utilisation des révisions

#### 13.3.1. Créer une révision

Lorsqu'une ressource est modifiée et sauvegardée localement dans l'espace de travail, il est possible d'enregistrer ces modifications dans CVS sous la forme d'une révision.

Il suffit de sélectionner la ressource et d'activer l'option "Equipe/Synchroniser avec le référentiel". La vue "Synchronisation" s'affiche avec la ressource modifiée.



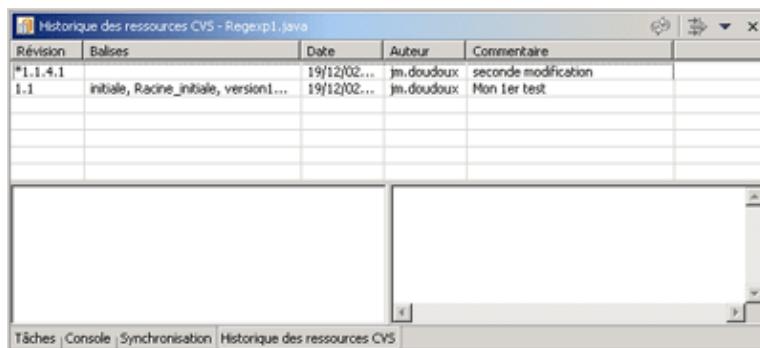
Dans cette vue, il suffit de sélectionner la ressource et d'activer l'option "Valider". Il est possible de saisir un commentaire.

La vue "Synchronisation" permet la synchronisation entre les ressources locales (dans l'espace de travail) et celles contenues dans CVS. Les trois premiers boutons permettent de préciser le sens de la synchronisation :

- mode entrant : modifications contenues dans le référentiel à intégrer dans l'espace de travail
- mode sortant : modifications contenues dans l'espace de travail à intégrer dans le référentiel
- mode entrant/sortant : modifications dans l'espace de travail et le référentiel à intégrer dans l'un et l'autre

### 13.3.2. Gestion des révisions

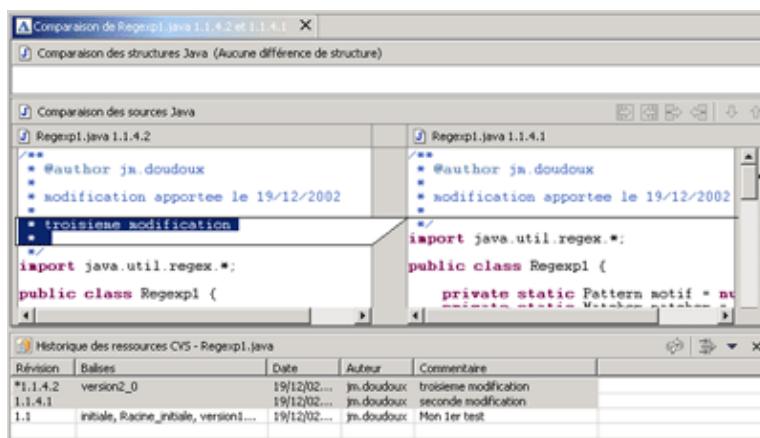
Il faut sélectionner une ressource et activer l'option "Equipe/Afficher dans l'historique des versions". La vue "Historique des ressources CVS" s'affiche en contenant les différentes révision de la ressource.



La révision courante est précédée d'une petite étoile.

Pour remplacer la ressource par celle correspondant à une autre révision, il suffit de sélectionner la révision et d'activer l'option "Obtenir une révision avec des marqueurs" du menu contextuel. Il faut ensuite cliquer sur le bouton "Ok" lors du message d'avertissement et la révision courante est modifiée.

Il est possible de comparer le contenu de deux révisions. Pour cela, il suffit de sélectionner les deux révisions en maintenant la touche Ctrl enfoncee et d'activer l'option "Comparer" du menu contextuel.

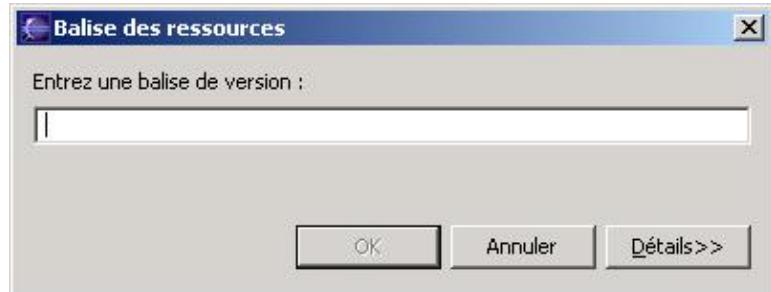


## 13.4. La gestion des versions d'un projet

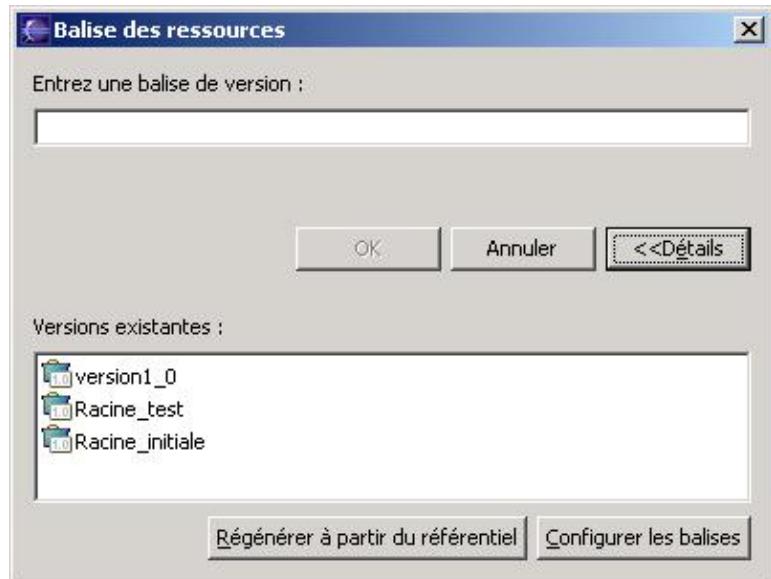
### 13.4.1. La création d'une version d'un projet

Il faut sélectionner le projet et activer l'option "Equipe/Baliser en tant que version" du menu contextuel.

Une boite de dialogue demande le nom de la balise



Un clic sur le bouton "Détails" permet de voir les versions existantes.



Il suffit de saisir le nom et de cliquer sur le bouton "Ok"

La version apparaît dans la vue "Référentiels CVS"

### 13.5. Obtenir une version dans le workspace

Dans la vue "Référentiels CVS", il suffit de cliquer sur la version concernée et d'activer l'option "Réserver en tant que projet" du menu contextuel.

Si le projet est déjà présent dans le workspace, un message demande la confirmation pour le remplacement.

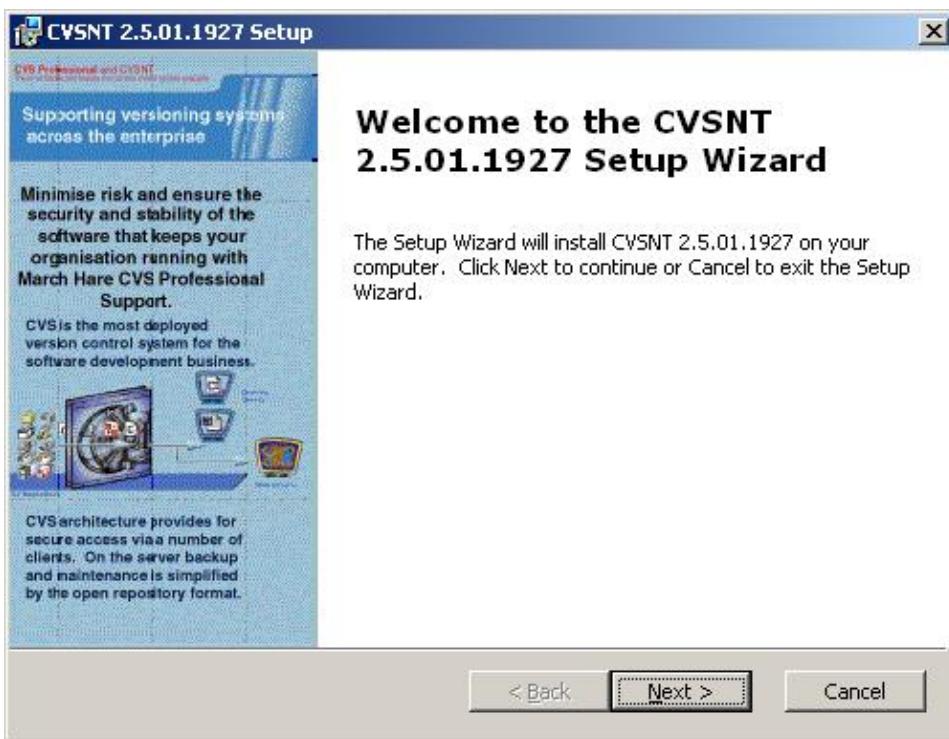


## 14. CVS 2.5 et Eclipse 3.0

# Chapitre 14

### 14.1. Installation et configuration de CVS 2.5

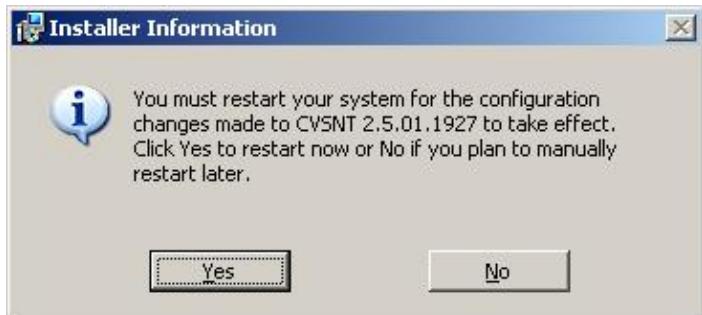
Il faut télécharger le fichier cvsnt-2.5.01.1927.msi sur le site <http://www.cvsnt.com/cvspro/> et l'exécuter



Il suffit de suivre les différentes étapes de l'assistant pour réaliser l'installation

- Sur la page d'accueil, cliquez sur le bouton « Next »
- Sur la page « End-User License Agreement », sélectionnez « I accept the terms in the Licence Agreement » et cliquez sur le bouton « Next »
- Sur la page « Choose Setup Type », cliquez sur le bouton « Complete »
- Sur la page « Ready to Install », cliquez sur le bouton « Install »
- Les fichiers sont copiés sur le système
- Cliquez sur le bouton « Finish »

Une boîte de dialogue informe de la nécessite de redémarrer le système.



Cliquez sur le bouton « Yes » pour redémarrer le système.

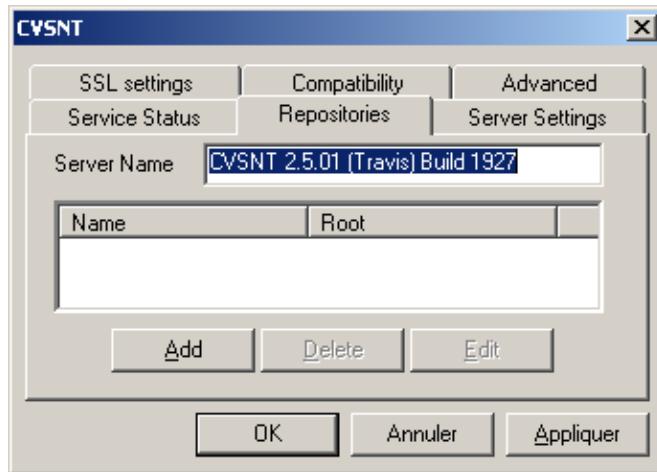
Une petite icône est ajoutée dans le panneau de configuration



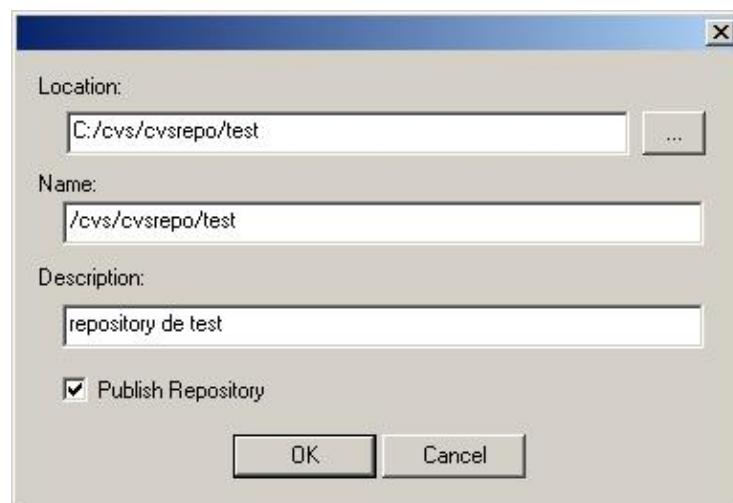
Un double clic sur cette icône permet d'accéder à la configuration de CVSNT.



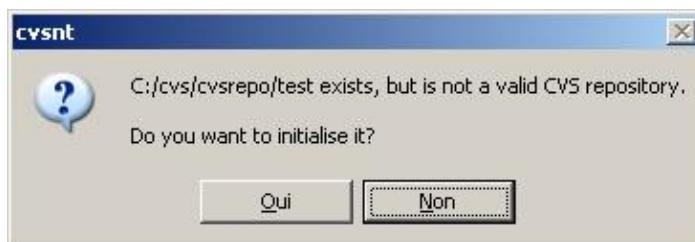
Cliquez sur l'onglet « Repositories »



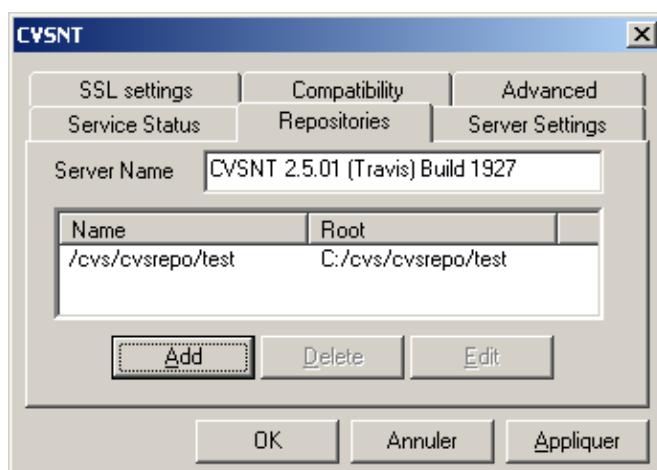
Cliquez sur le bouton « Add »



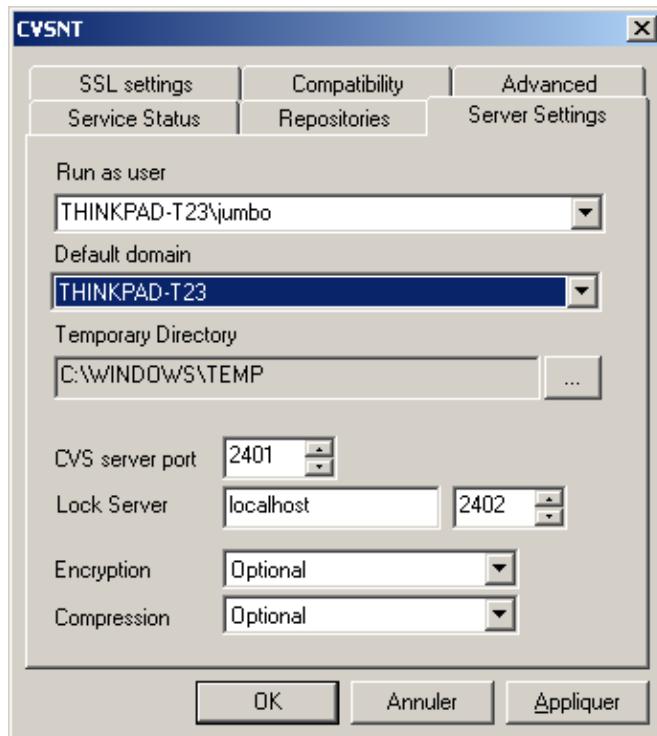
Sélectionner l'emplacement de stockage des fichiers, saisissez le nom du référentiel et la description puis cliquez sur le bouton « Ok ».



Cliquez sur le bouton « Oui » pour procéder à l'initialisation du référentiel



Sélectionnez l'onglet « Server Settings »

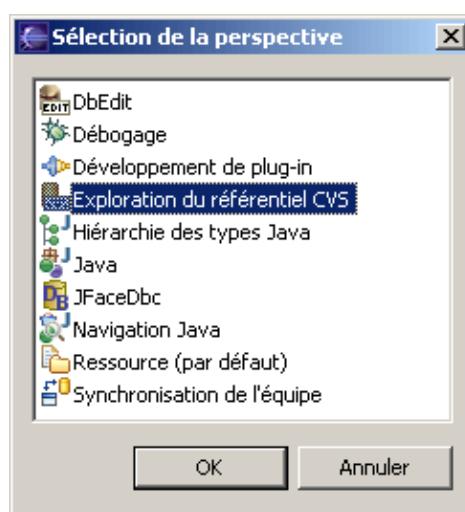


Sélectionnez les informations utiles et cliquez sur le bouton « Appliquer »

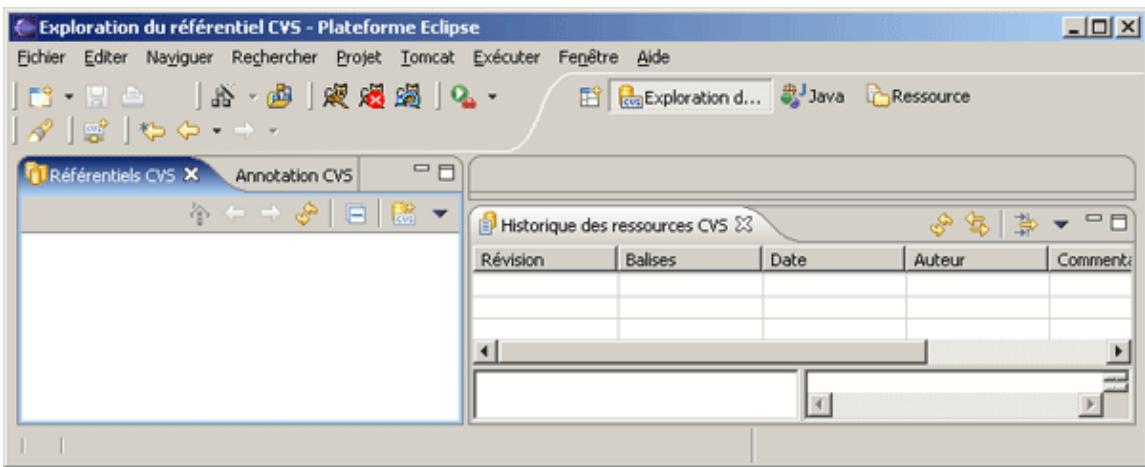
Sélectionnez l'onglet « Service Status » et démarrez les deux services proposés ci ceux-ci sont arrêtés.

## 14.2. La perspective « Exploration du référentiel CVS »

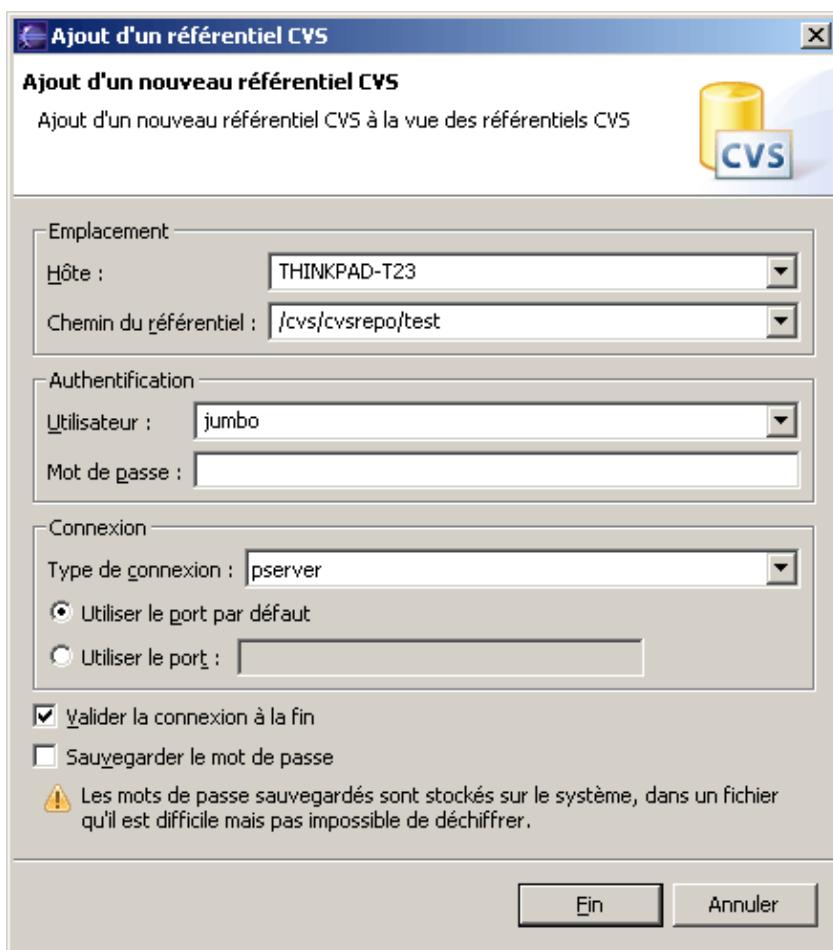
Pour ouvrir une perspective, il faut sélectionnez le menu « Fenêtre/Ouvrir la perspective/Autre ».



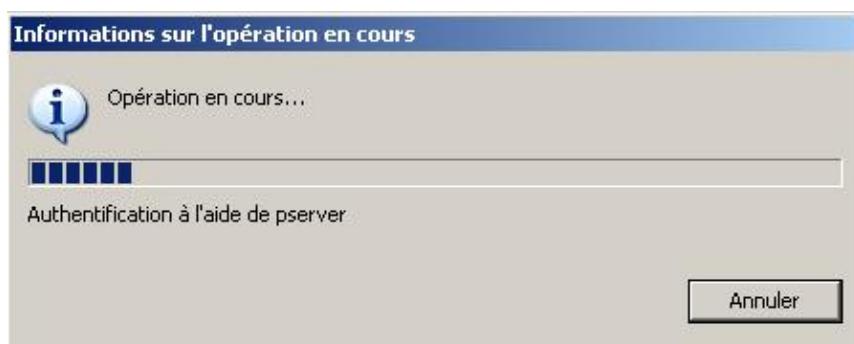
Sélectionnez l'option « Exploration du référentiel CVS » et cliquez sur le bouton « OK » pour ouvrir la perspective.



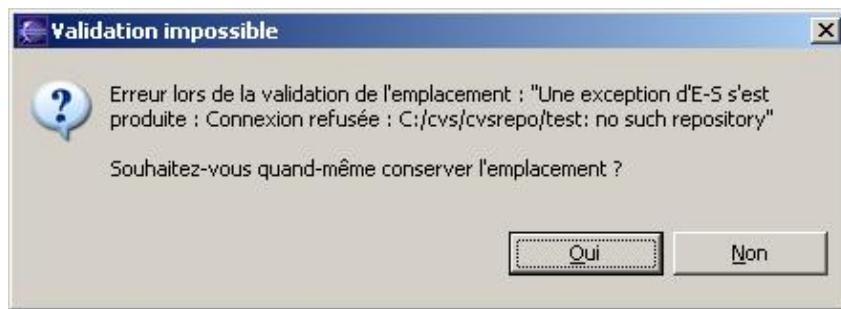
Dans la vue « Référentiels CVS », sélectionnez l'option « Créez/Emplacement du référentiel » du menu contextuel.



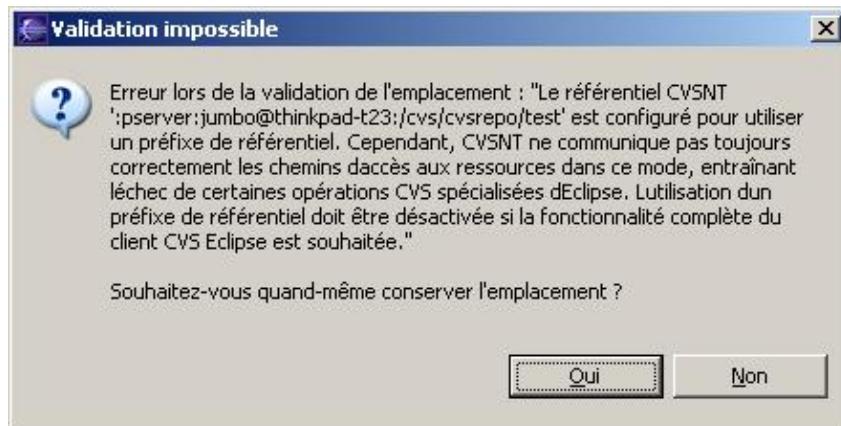
Saisissez le nom de l'hôte, le chemin du référentiel, nom de l'utilisateur et son mot de passe pour se connecter à CVS, le type de connexion. Cliquez sur le bouton « Fin » pour demander la connexion



Un message d'erreur informe de l'inexistante du référentiel si ce dernier n'est pas trouvé.

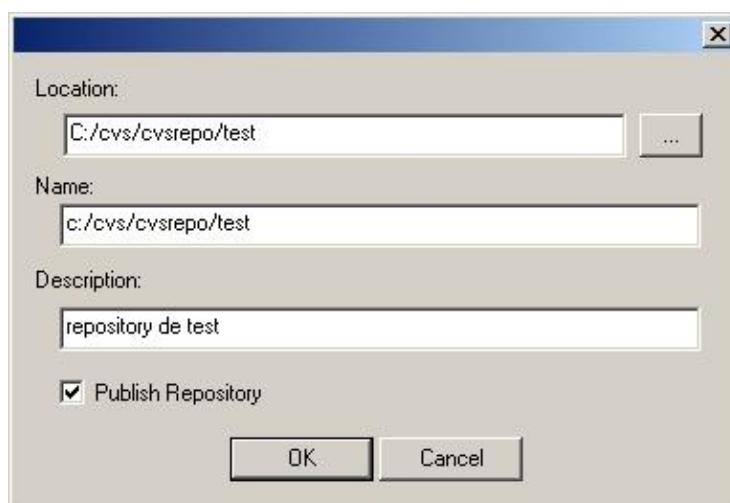


Comme CVS est configuré pour utiliser un préfixe, un message d'erreur est affiché.

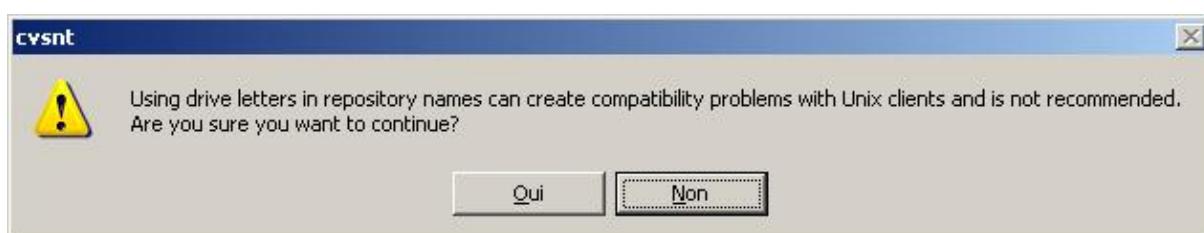


Cliquez sur le bouton « Non »

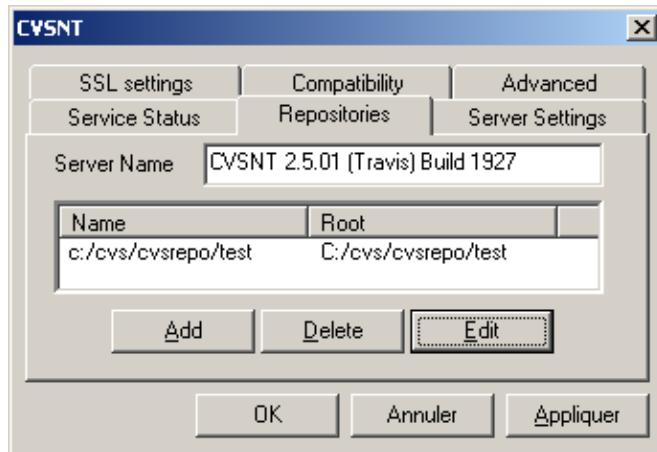
Il est alors nécessaire de modifier le nom du référentiel dans la configuration de CVSNT en précisant le chemin complet du référentiel dans son nom.



Cliquez sur le bouton « OK »

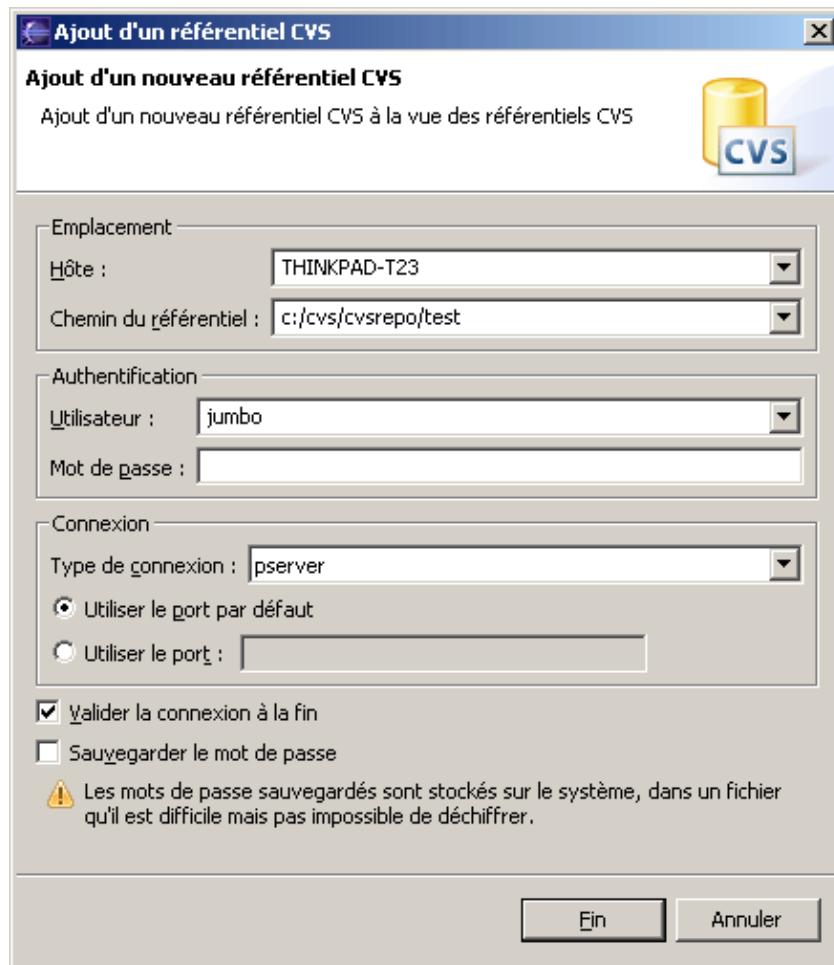


Cliquez sur le bouton « Oui »



Cliquez sur le bouton « Appliquer » puis sur le bouton « OK »

Il suffit alors de modifier les paramètres du référentiel sous Eclipse



Le chemin du référentiel doit contenir le chemin complet. Cliquez sur le bouton « Fin »

Si le mot de passe saisi est incorrect, une boîte de dialogue permet de le ressaisir.



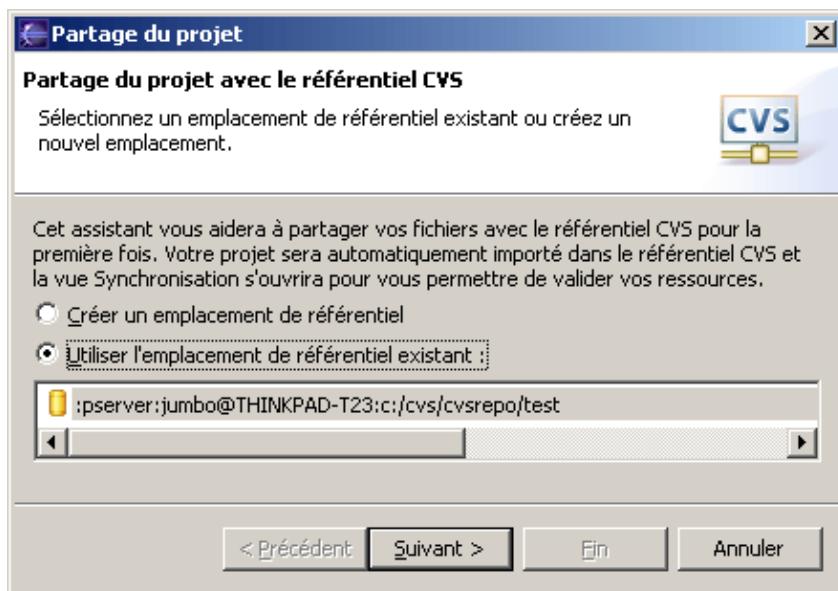
Le nouveau chemin de référentiel est ajouté dans la vue « Référentiels CVS ».



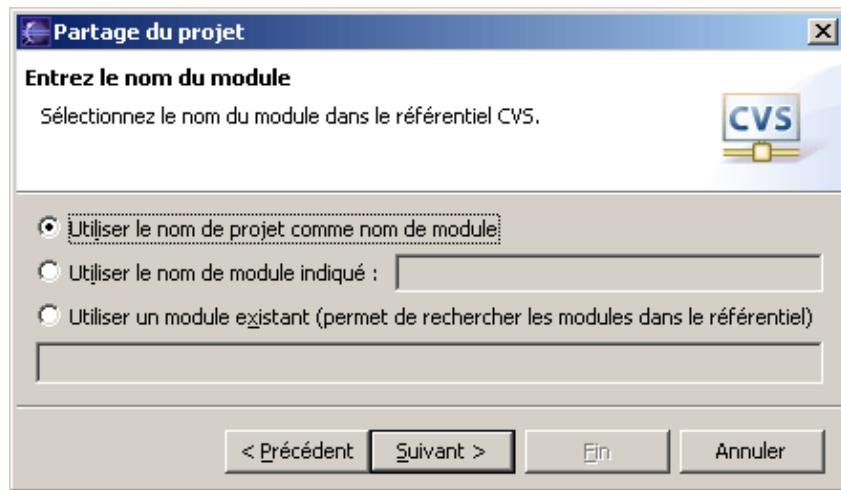
Un chemin de référentiel n'est pas une connexion mais des informations nécessaires à cette connexion réalisée lorsque que les traitements en ont besoin.

### 14.3. Ajouter un projet au référentiel

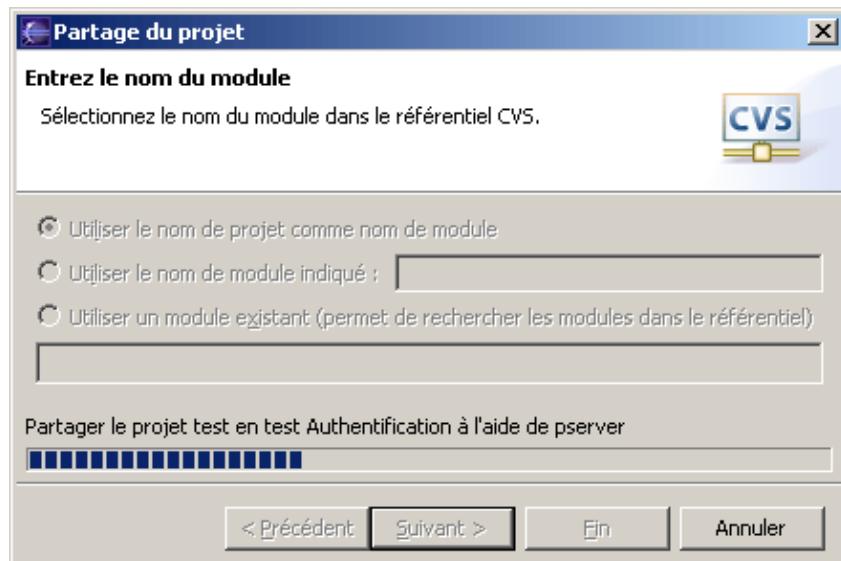
Pour ajouter un projet au référentiel, il faut utiliser l'option « Equipe/Partager le projet ... » du menu contextuel du projet



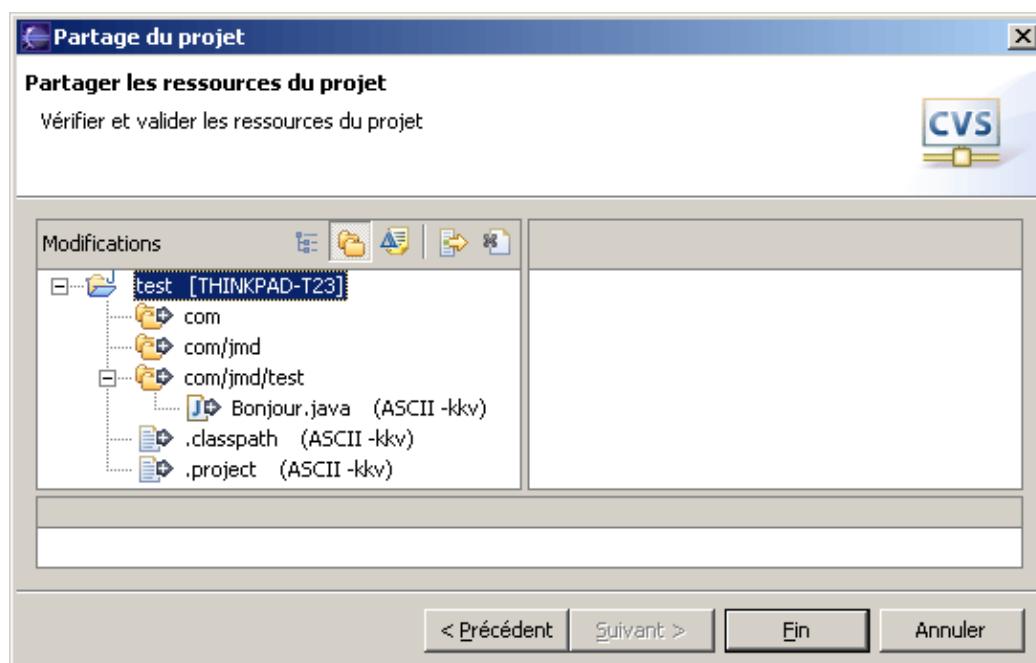
Sélectionnez « Utiliser l'emplacement de référentiel existant » et cliquez sur le bouton « Suivant »



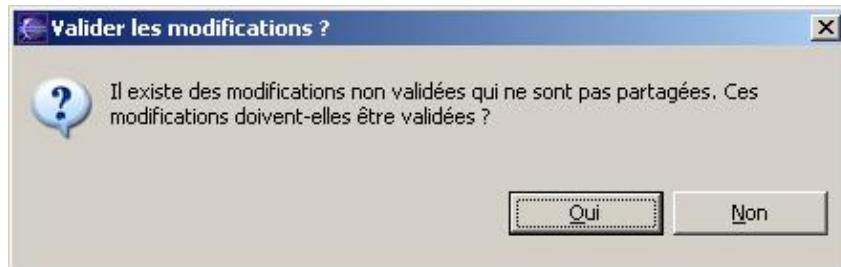
Sélectionnez « Utiliser le nom de projet comme nom de module » et cliquez sur le bouton « Suivant »



La page suivante de l'assistant permet de voir les fichiers qui seront ajoutés au référentiel.



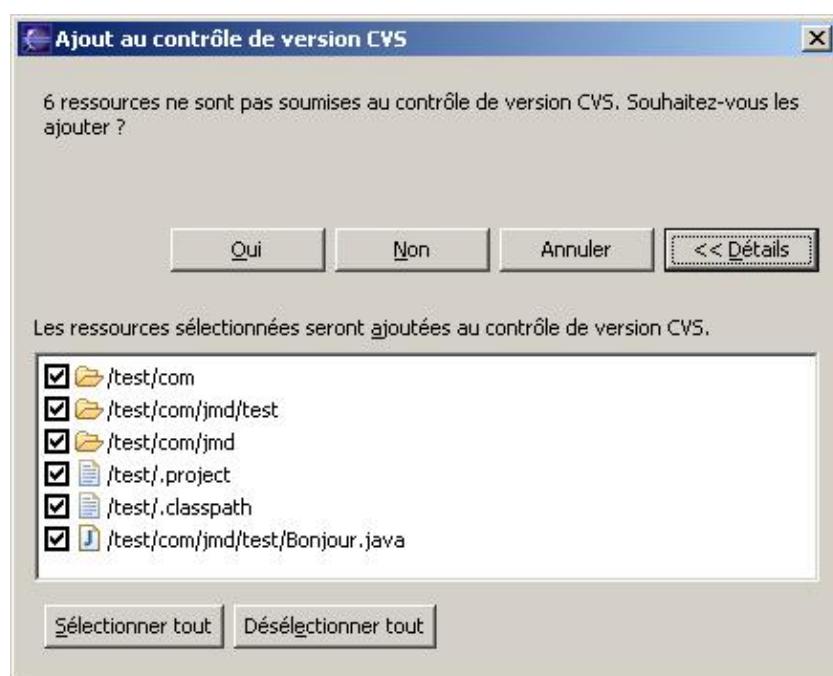
Cliquez sur le bouton « Fin »



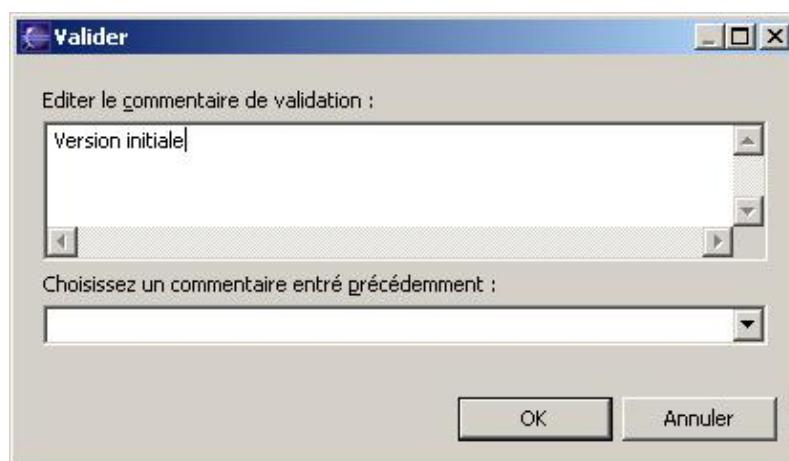
Cliquez sur le bouton « Oui »



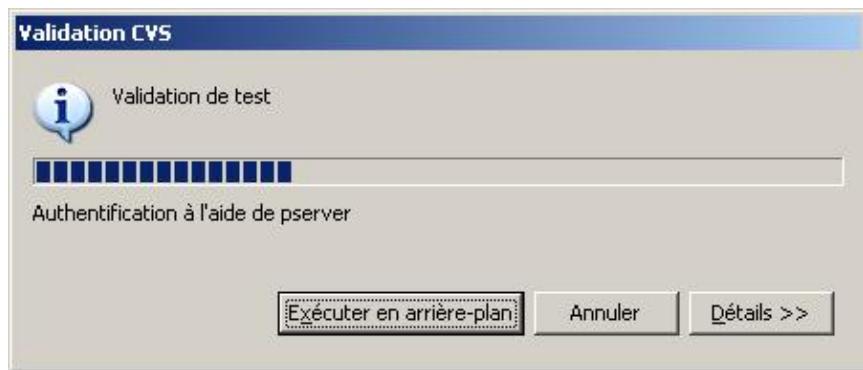
Cliquez sur le bouton « Détails » pour obtenir la liste des ressources à ajouter dans le référentiel.



Cliquez sur le bouton « Oui »



Saisissez le commentaire et cliquez sur le bouton « OK »



Une boîte de dialogue affiche la progression des traitements.



Les projets qui sont connectés à un référentiel sont identifiables grâce à plusieurs informations :

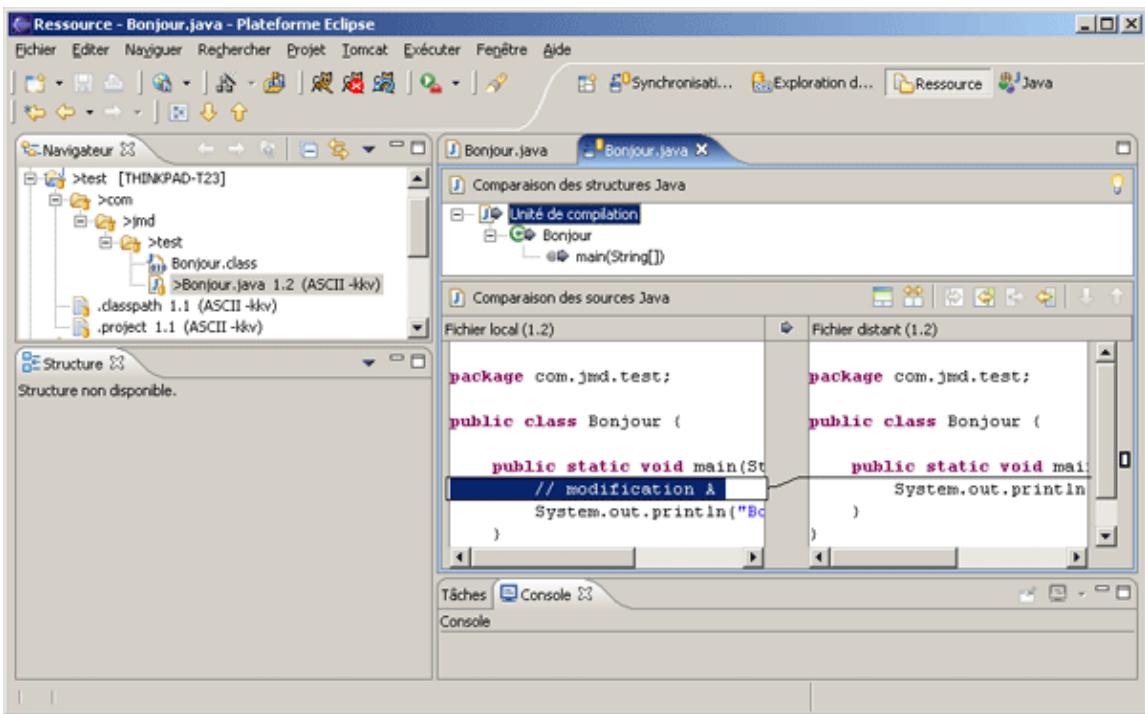
- le projet et les éléments inclus dans le référentiel possèdent une icône avec un petit cylindre jaune
- le nom du projet est suivi du nom du référentiel entre crochets
- le nom des éléments du projet sont suivis de la version et du type de fichiers entre parenthèses

## 14.4. Reporter des modifications dans le référentiel

Dès qu'un élément inclus dans le référentiel est modifié, ce dernier et le nom de l'ensemble de ces éléments pères sont précédés d'un caractère « > » permettant de rapidement visualiser les fichiers modifiés.



Pour reporter les modifications, il est possible de le faire au niveau de chaque éléments ou au niveau du projet en utilisant l'option « Equipe/Synchroniser avec le référentiel » du menu contextuel.



La vue affiche les différences entre la version locale et la version contenue dans le référentiel.

Pour reporter les modifications dans le référentiel, il faut utiliser l'option « Equipe/Valider » de l'élément comparé.



Une boîte de dialogue permet de saisir le commentaire de validation. Cliquez sur le bouton « OK ».

Comme la version locale et la version dans le référentiel sont identiques, les caractères <> disparaissent.



## 14.5. Déconnecter un projet du référentiel

Pour supprimer un référentiel, il faut sélectionner l'option « Ignorer l'emplacement » du menu contextuel

Il est obligatoire qu'aucun projet ne soit connecté sur ce dernier sinon un message d'erreur empêche l'opération.



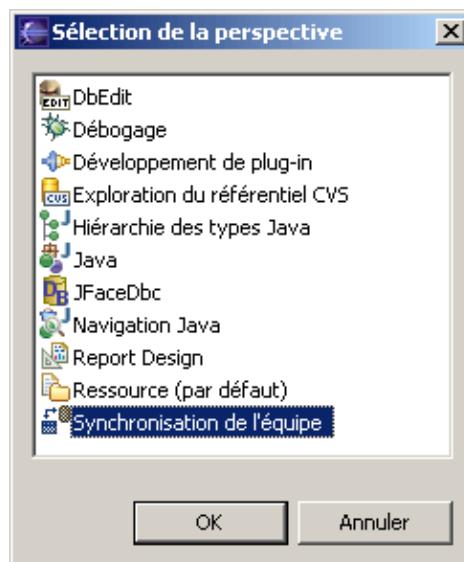
Pour déconnecter un projet, il faut utiliser l'option « Equipe/Déconnecter ... » du menu contextuel du projet.



Cliquez sur le bouton « Oui »

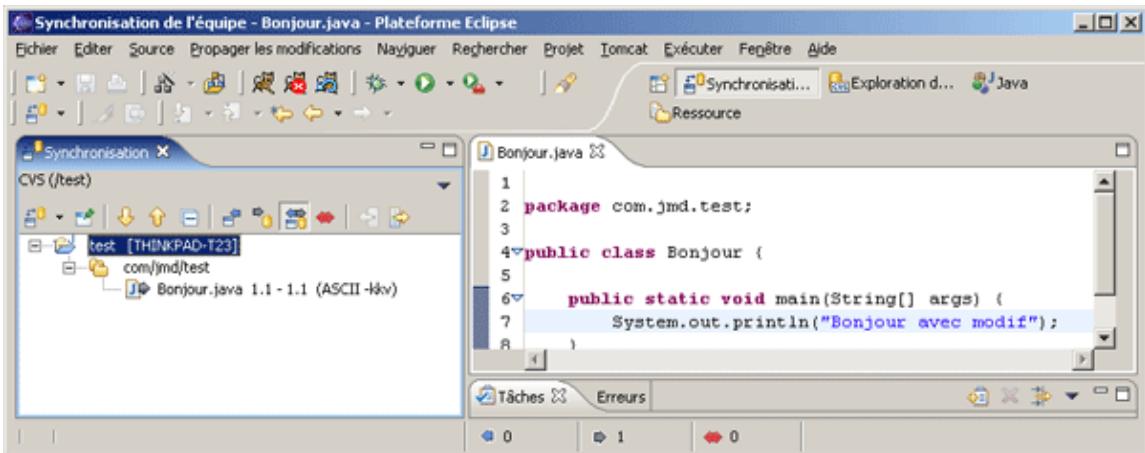
## 14.6. La perspective Synchronisation de l'équipe

Pour ouvrir cette perspective, il faut utiliser l'option « Fenêtre/Ouvrir la perspective/Autre » et sélectionner l'option « Synchronisation de l'équipe »



Cette perspective est aussi utilisée lors de l'utilisation de l'option « Equipe/Synchroniser avec le référentiel ... »

La vue « Synchronisation » permet de synchroniser les sources contenues dans l'espace de travail et le référentiel de CVS.



Les modifications sont signalées avec de petites icônes :

	Pour les modifications contenues dans le référentiel à intégrer dans l'espace de travail
	Pour les modifications contenues dans l'espace de travail à intégrer dans le référentiel
	Pour signaler des modifications effectuées dans l'espace de travail et dans le référentiel

La vue synchronisation propose plusieurs modes pour faciliter la synchronisation :

	Mode entrant : des modifications sont présentes dans le référentiel mais pas encore en local
	Mode sortant : modifications locales qui ne sont pas encore enregistrées dans le référentiel
	Mode entrant/sortant : utilisation des deux modes précédents simultanément
	Mode « en conflit » : des modifications locales existent et des modifications différentes sont présentes dans le référentiel

Les modifications selon le mode courant sont affichées :



Si le mode courant ne contient aucune modification, il propose de passer dans un autre mode.



Pour connaître l'historique des versions, il faut utiliser dans la vue « Synchronisation » l'option « Afficher dans l'historique des versions » du menu contextuel d'un élément.

La vue « Historique des ressources CVS » s'affiche.

The screenshot shows the Eclipse interface with the title bar "Tâches Erreurs Historique des ressources CVS". The main area displays a table for the file "Bonjour.java" with columns: Révision, Balises, Date, Auteur, and Commentaire. The table contains five rows of data:

Révision	Balises	Date	Auteur	Commentaire
1.4		14/06/05 08:23	jumbo	Modification B
*1.3		14/06/05 08:17	jumbo	<b>modification A</b>
1.2		13/06/05 22:17	jumbo	Version 2
1.1		02/05/05 22:19	jumbo	Version initiale

Below the table is a text input field containing "modification A" with scroll bars on either side.

La version correspondant à celle dans l'espace de travail est signalée avec une petite étoile.

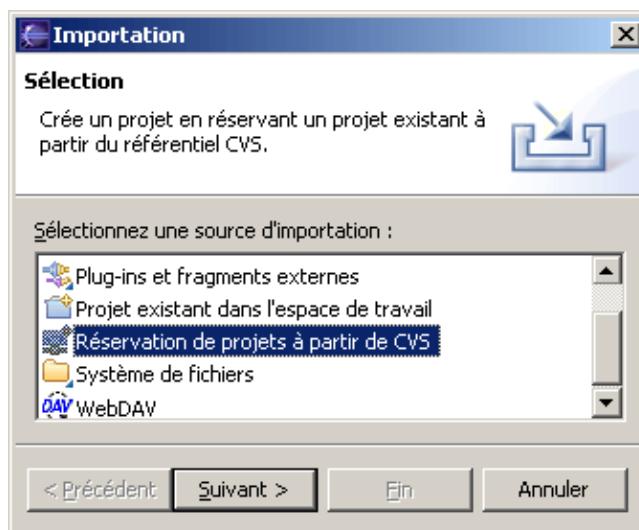
A partir de cette vue, il est possible de récupérer dans l'espace de travail n'importe quelle version en la sélectionnant et en utilisant l'option « Obtenir le contenu » du menu contextuel.

The screenshot shows the same Eclipse interface as the previous one, but the row for revision 1.3 is now selected, indicated by a blue highlight. The table data remains the same as in the first screenshot.

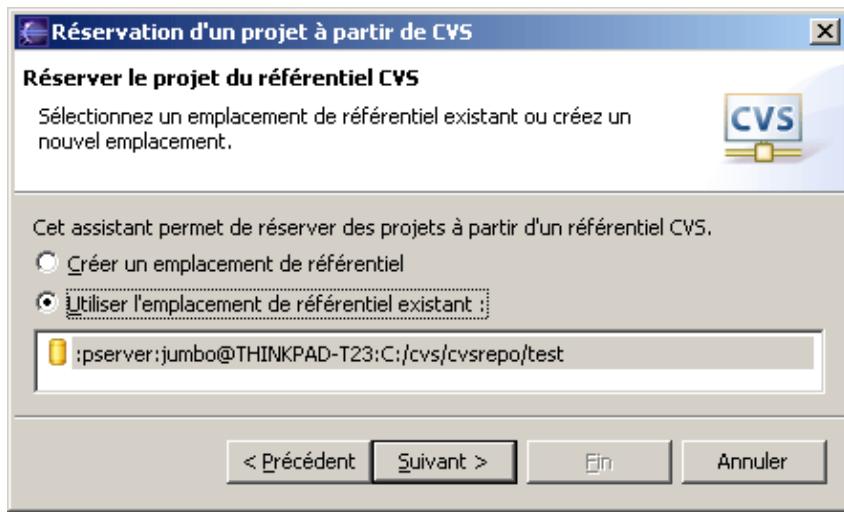
## 14.7. Importation d'un projet à partir de CVS

Il est possible à n'importe quel utilisateur pouvant se connecter au référentiel d'importer un projet dans son espace de travail pour le modifier.

Il faut utiliser l'option « Fichier/Importer ...» du menu principal.



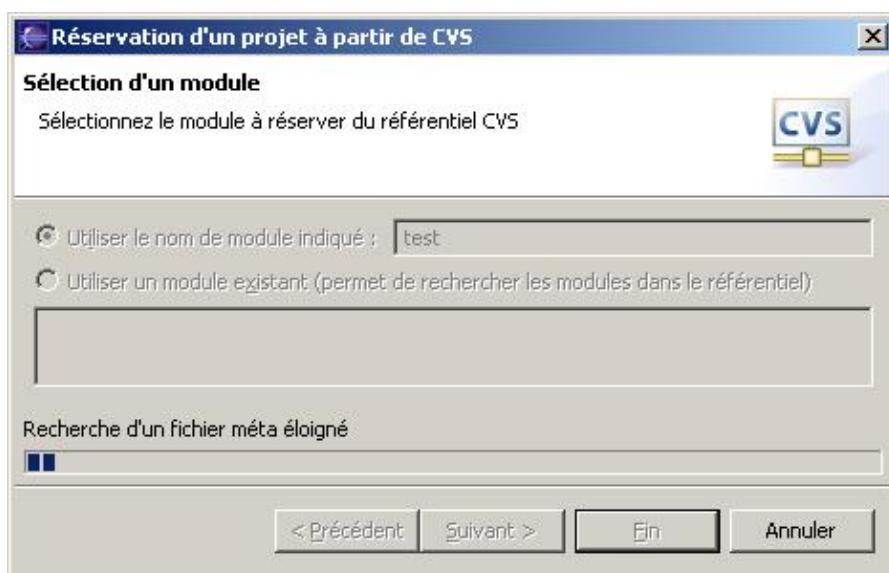
Sélectionnez « Réservation de projets à partir de CVS » puis cliquez sur le bouton « Suivant ».



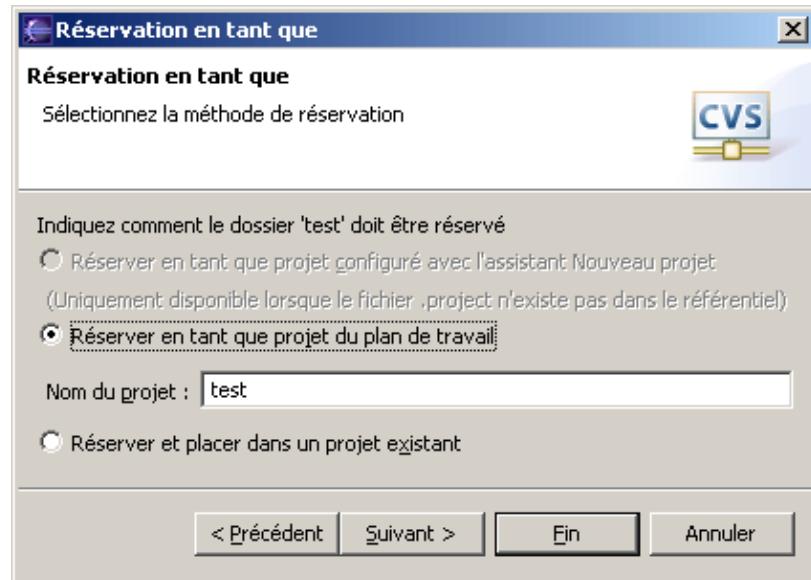
Le référentiel est sélectionné par défaut, cliquez sur le bouton « Suivant ».



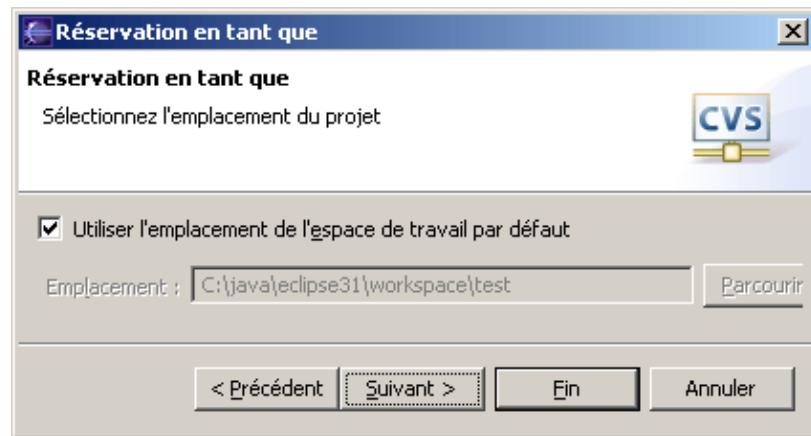
Saisir le nom du module et cliquez sur le bouton « Suivant ». Il est aussi possible de cocher la case « Utiliser un module existant » pour demander à l'assistant de rechercher les modules inclus dans le référentiel et de permettre de sélectionner ceux concernés.



L'assistant se connecte au référentiel



Cliquez sur le bouton « Suivant »

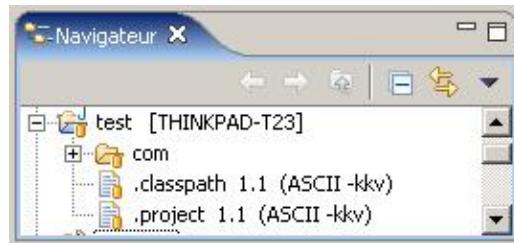


Cliquez sur le bouton « Suivant »



Sélectionnez la balise HEAD et cliquez sur le bouton « Fin »

Le projet est importé dans l'espace de travail.



## 14.8. Versionner un projet

Il faut utiliser l'option « Equipe/Baliser en tant que version... » dans la vue « Navigateur » ou « Packages »

Si toutes les modifications ne sont pas validées, un message de confirmation est demandé à l'utilisateur.



Une boîte de dialogue permet de saisir le nom de la balise de version (ce nom ne doit pas contenir d'espaces)



Il suffit de saisir le nom de la balise et de cliquer sur le bouton « OK »

Révision	Balises	Date	Auteur	Commentaire
*1.4	v1	14/06/05 08:23	jumbo	Modification B
1.3		14/06/05 08:17	jumbo	modification A
1.2		13/06/05 22:17	jumbo	Version 2
1.1		02/05/05 22:19	jumbo	Version initiale

Below the table, there is a small window showing the details for revision 1.4, which is 'Modification B'.

## 15. Subversion et Eclipse

# Chapitre 15

SubVersion (SVN) est un système de control de versions open source plus récent que CVS et qui propose des fonctionnalités manquantes dans l'historique système de contrôle de versions open source.

Ce chapitre propose

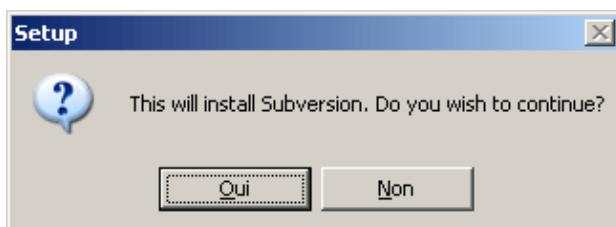
- d'installer et paramétrier subversion sous Windows
- d'installer le plug-in subclipse
- de mettre en oeuvre le plug-in subclipse

	Version utilisée dans cette section
SubVersion	1.3
Eclipse	3.0.1
J2RE	1.5.0_07
Plug-in Subclipse	1.0.1

### 15.1. Installation de subversion sous Windows

Téléchargez le fichier svn-1.3.0-setup.exe sur le site <http://subclipse.tigris.org/>.

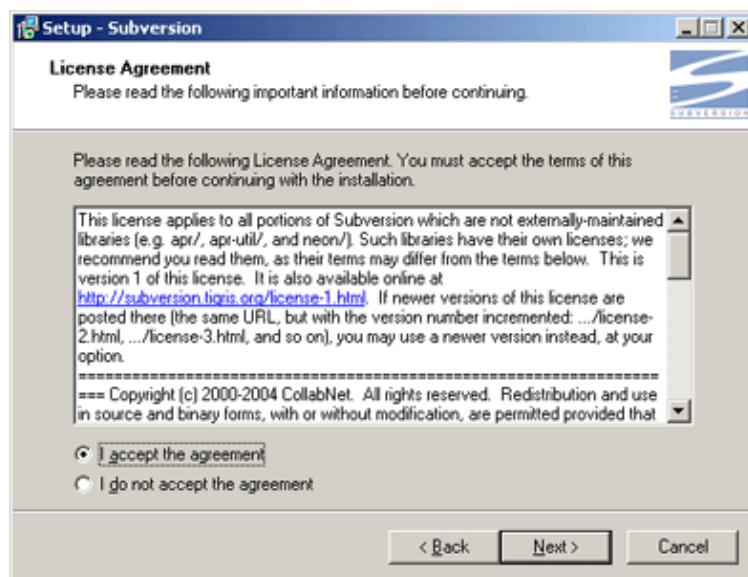
Exécutez le fichier svn-1.3.0-setup.exe



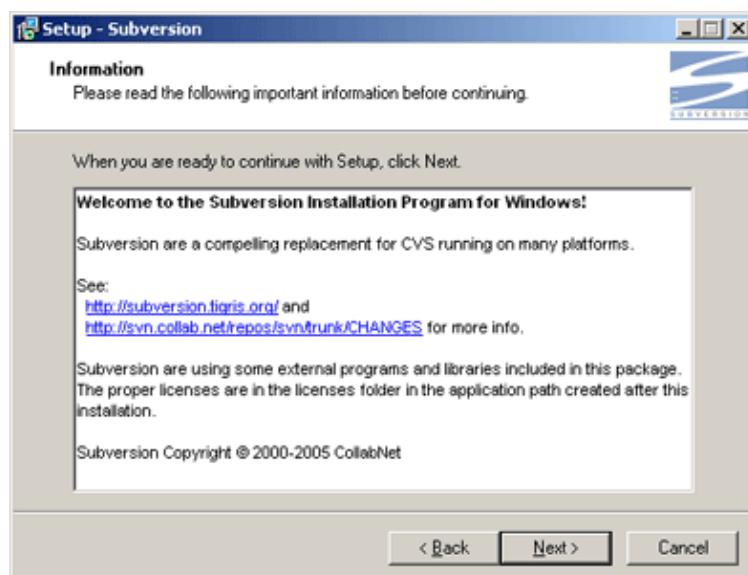
Cliquez sur le bouton « Oui »



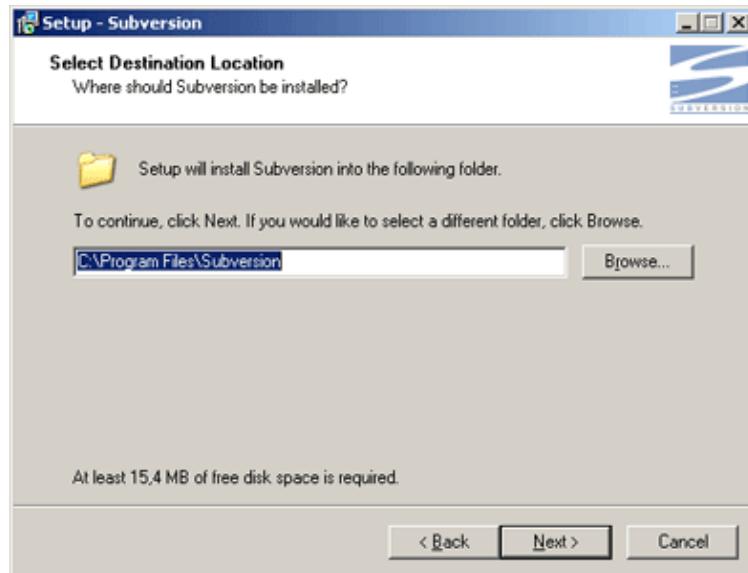
Cliquez sur le bouton « Next »



Lire la licence et si vous l'acceptez cliquez sur « I accept the agreement » puis sur le bouton « Next »



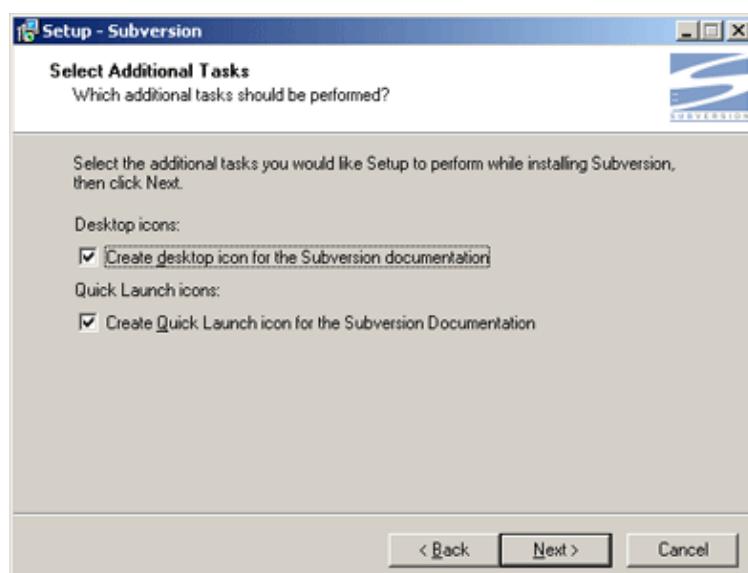
Cliquez sur le bouton « Next »



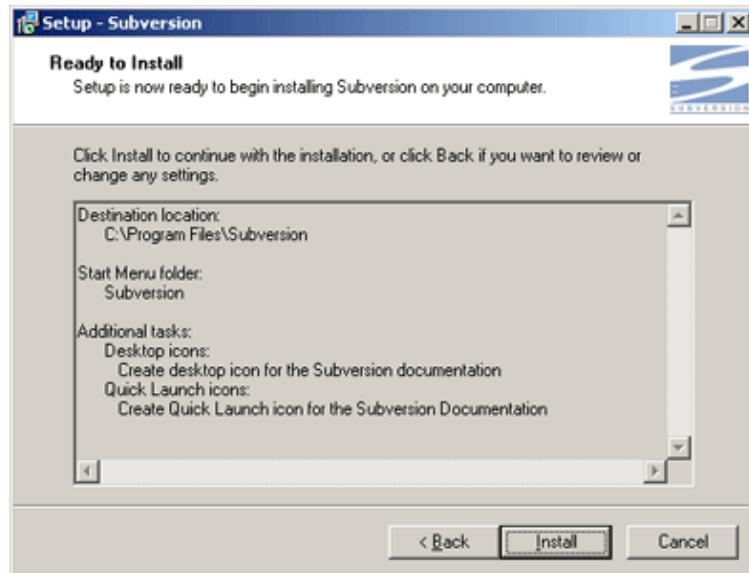
Cliquez sur le bouton « Next »



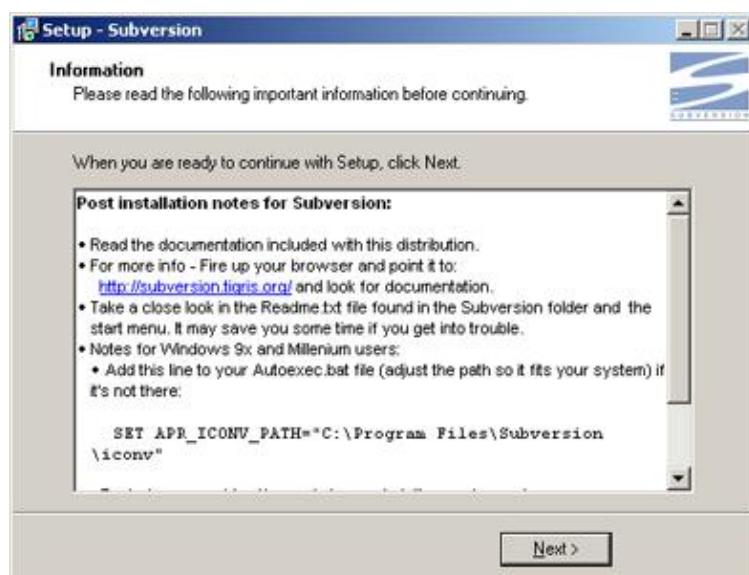
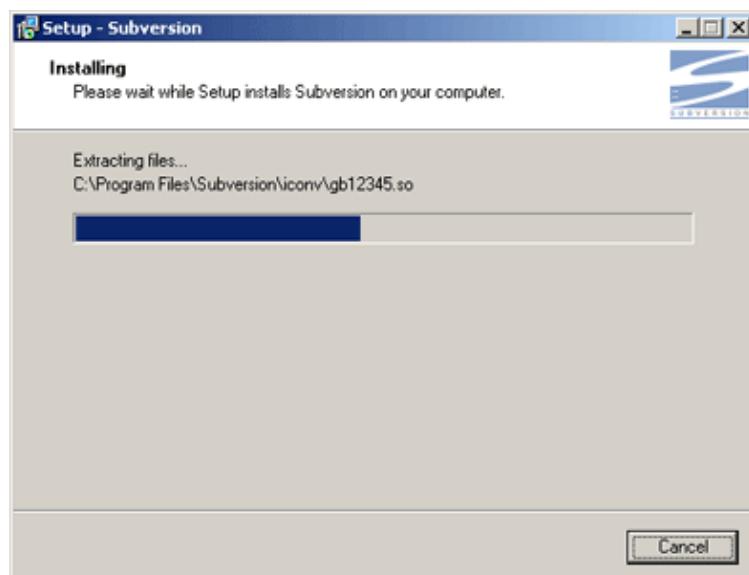
Cliquez sur le bouton « Next »



Cliquez sur le bouton « Next »



Cliquez sur le bouton « Install »

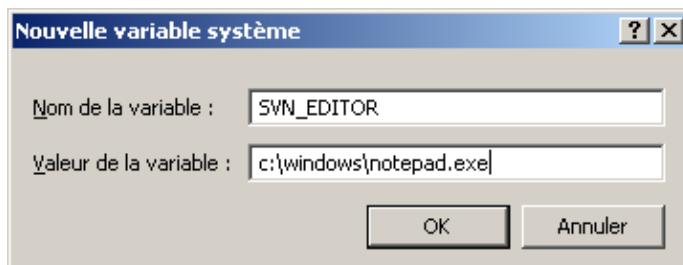


Cliquez sur le bouton « Next »



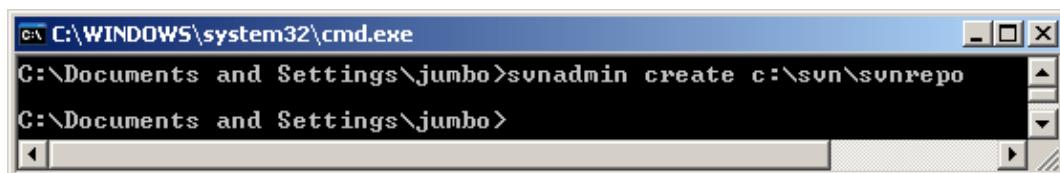
Cliquez sur le bouton « Finish »

Ajouter une variable d'environnement SVN\_EDITOR avec comme valeur le chemin vers l'outil notepad (c:\windows\notepad.exe)

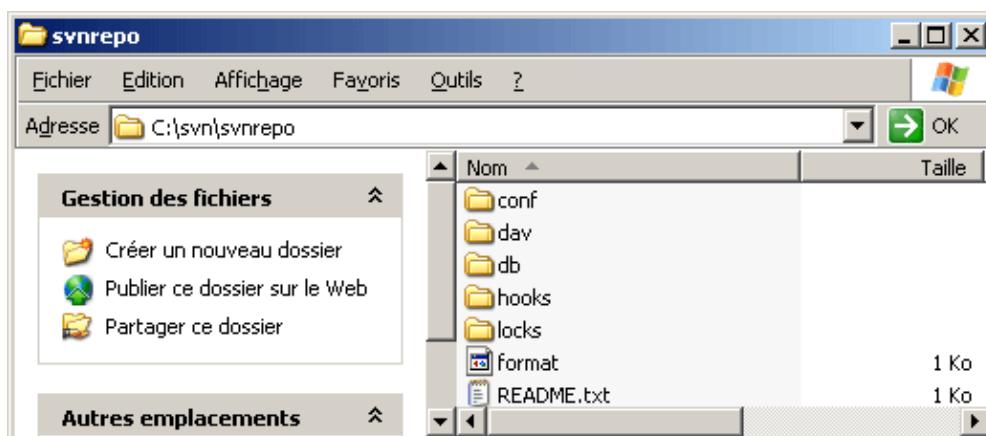


Il faut créer un répertoire qui va contenir le repository, par exemple C:\svn\svnrepo

Ouvrez une boîte de commande Dos et exécuter la commande svnadmin create c:\svn\svnrepo



Le repository est créé : il est composé de plusieurs sous répertoires.



Dans le fichier svnserve.conf du sous répertoire conf, il faut supprimer les commentaires sur les lignes ci

dessous

Exemple :

```
...
# [general]
...
# anon-access = read
# auth-access = write
...
# password-db = passwd
...
```

Pour cela, il faut supprimer le caractère # en début de chacune des quatre lignes.

Il faut ensuite éditer le fichier passwd du sous répertoire conf pour supprimer le commentaire sur la ligne ci dessous.

Exemple :

```
...
# [users]
...
```

Il faut ensuite ajouter dans cette section le ou les utilisateurs avec leur mot de passe.

Exemple :

```
...
[users]
# harry = harryssecret
# sally = sallyssecret
jumbo = jumbomdp
...
```

Ouvrez une boîte de commandes Dos et exécutez la commande svnserve --daemon --root c:\svn\svnrepo

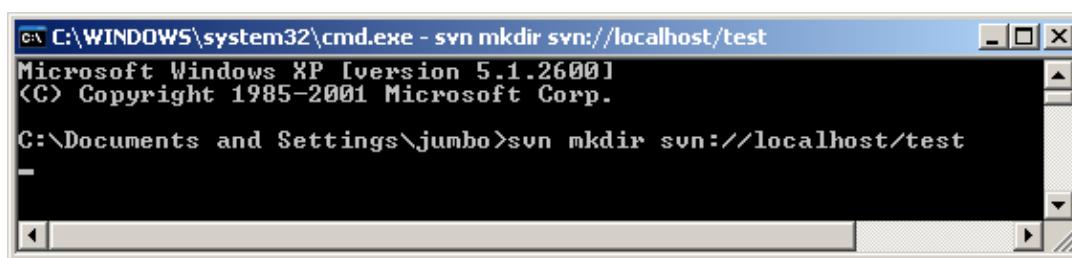


Avec le service pack 2 de Windows XP, une alerte de sécurité est affichée

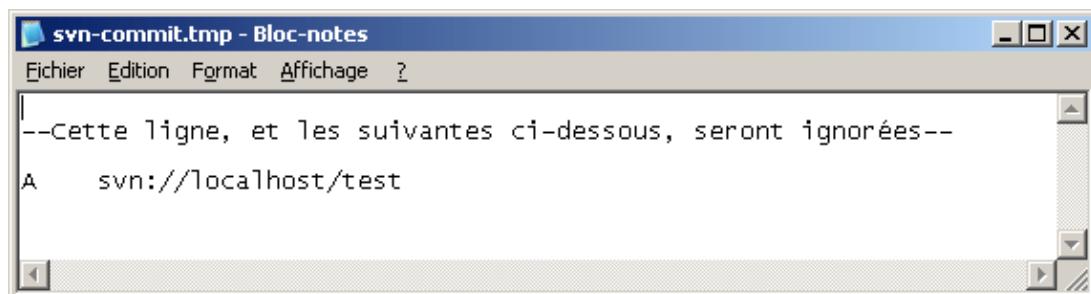


Cliquez sur le bouton « Débloquer »

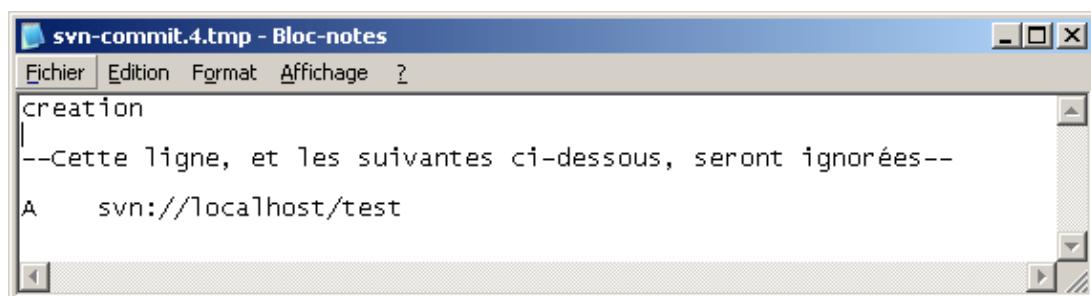
Il faut ensuite créer un projet, nommé par exemple test, en ouvrant une boîte de commandes Dos pour exécuter la commande ci dessous



L'éditeur de texte s'ouvre



Saisissez sur la première ligne un commentaire



Enregistrez le fichier et fermez l'éditeur

```
ex C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\jumbo>svn mkdir svn://localhost/test
Domaine d'authentification : <svn://localhost:3690> 11bd7939-afe1-4445-9387-1d75
7dd7d2c4
Mot de passe pour 'jumbo' : *****

Révision 1 propagée.

C:\Documents and Settings\jumbo>
```

Saisissez le mot de passe indiqué dans le fichier passwd.

Par défaut, c'est l'utilisateur de Windows qui est utilisé. Pour utiliser un autre utilisateur, il faut simplement appuyer sur la touche entrée lors de la saisie du mot de passe, de saisir le nom de l'utilisateur à utiliser et son mot de passe.

Si subversion n'est pas correctement configuré, un message d'erreur est affiché :

Exemple : la section [header] n'est pas décommentée

```
ex C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\jumbo>svn mkdir svn://localhost/test
svn: C:\svn\svnrepo\conf\svnserve.conf:12: Section header expected
svn: Le message de propagation a été laissé dans un fichier temporaire :
svn:   'svn-commit.4.tmp'

C:\Documents and Settings\jumbo>
```

Exemple : les trois lignes ne sont pas décommentées

```
ex C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\jumbo>svn mkdir svn://localhost/test
svn: Autorisation refusée
svn: Le message de propagation a été laissé dans un fichier temporaire :
svn:   'svn-commit.tmp'
```

Pour obtenir la liste des projets, il faut utiliser la commande svn ls svn://localhost/

```
ex C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\jumbo>svn ls svn://localhost/
test/

C:\Documents and Settings\jumbo>
```

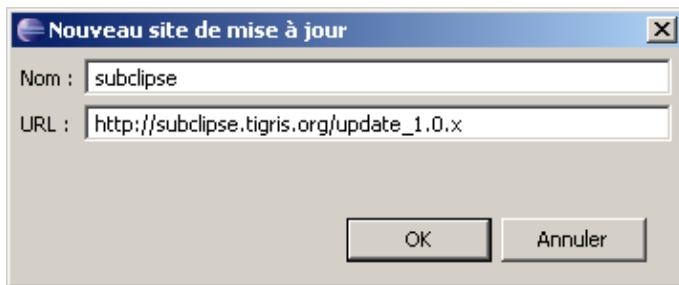
## 15.2. Le plug-in Subclipse

Subclipse est un plug-in qui permet des interactions avec Subversion à partir d'Eclipse.

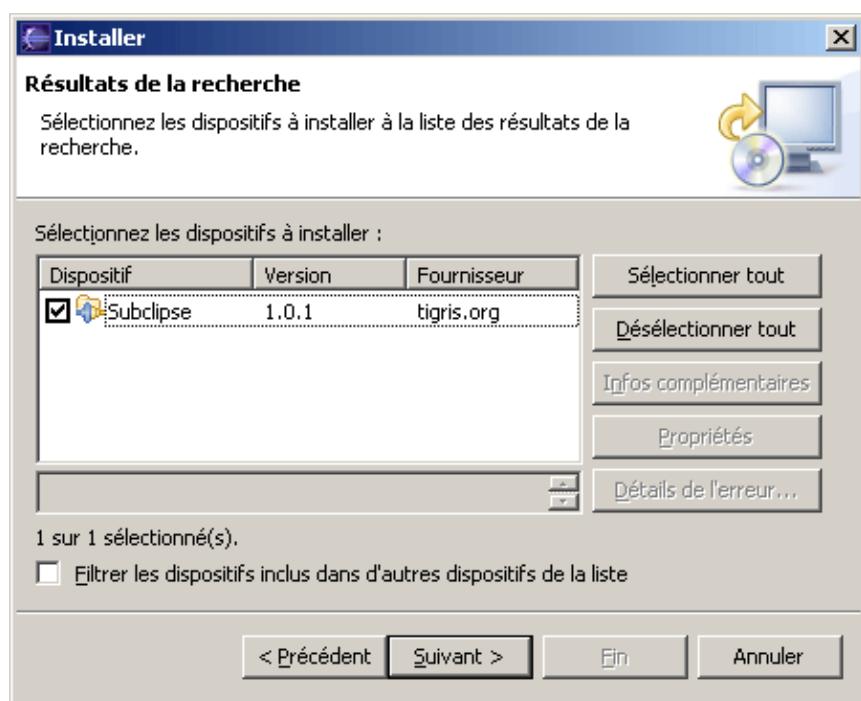
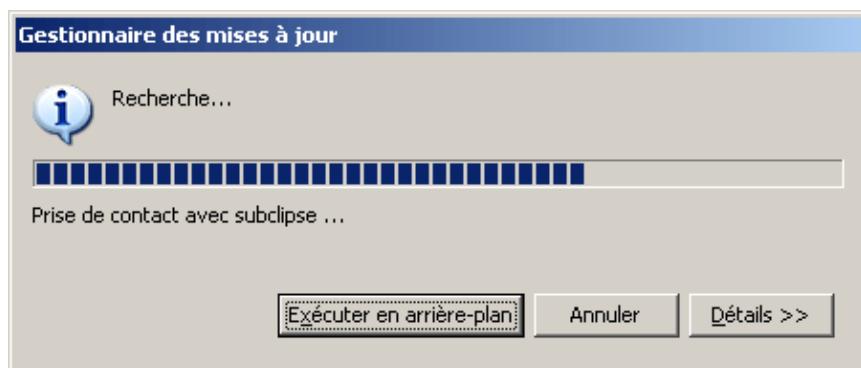
Le site officiel du plug-in est à l'url : <http://subclipse.tigris.org/>

### 15.2.1. Installation de Subclipse

Le plus simple est d'utiliser le gestionnaire de mise à jour pour installer le plug-in

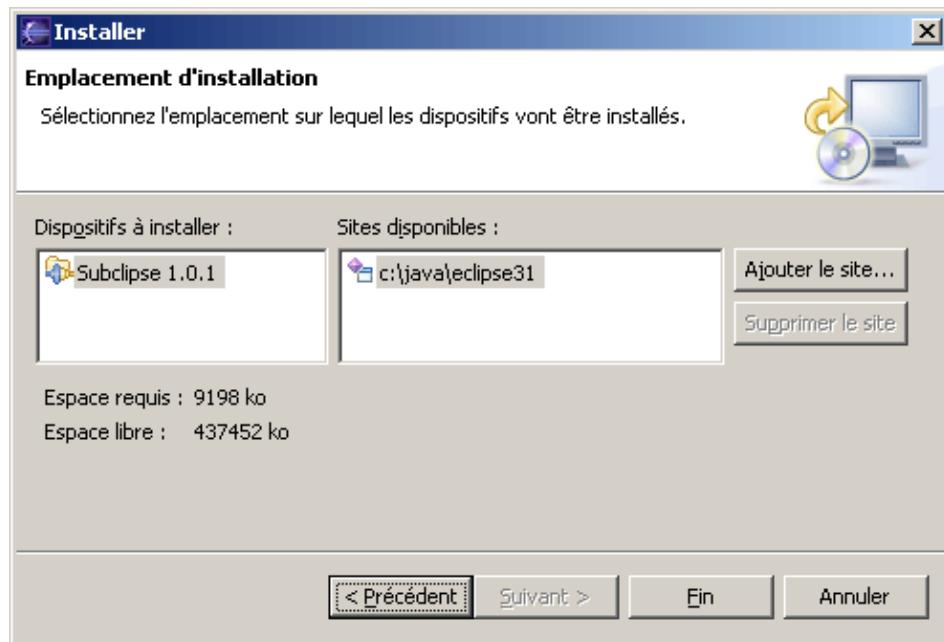


Saisissez le nom du site et son url et cliquez sur le bouton « OK ».



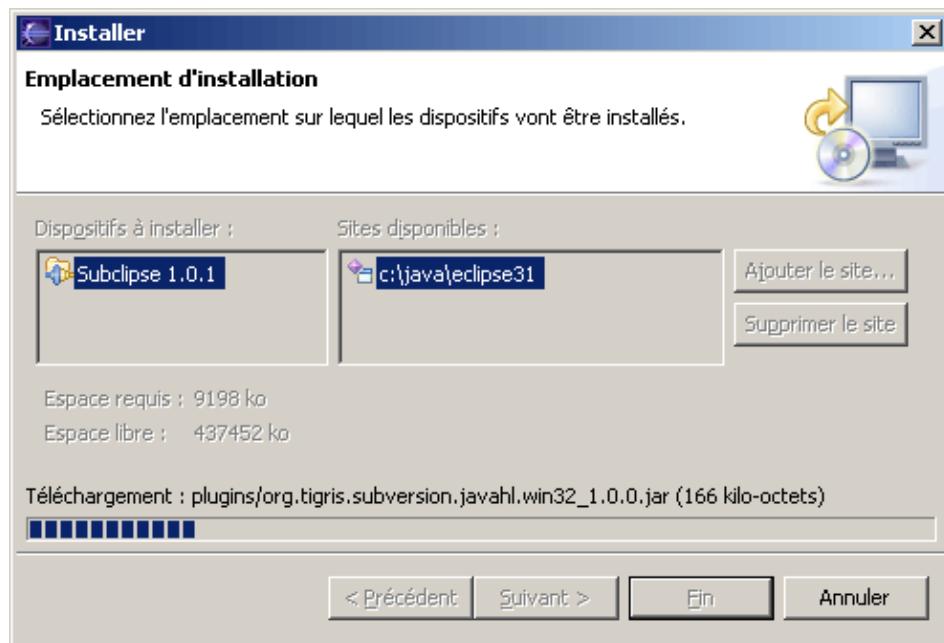
Sélectionnez la version et cliquez sur le bouton « Suivant ».

Lisez et acceptez les termes de la licence en cliquant sur le bouton « Suivant »

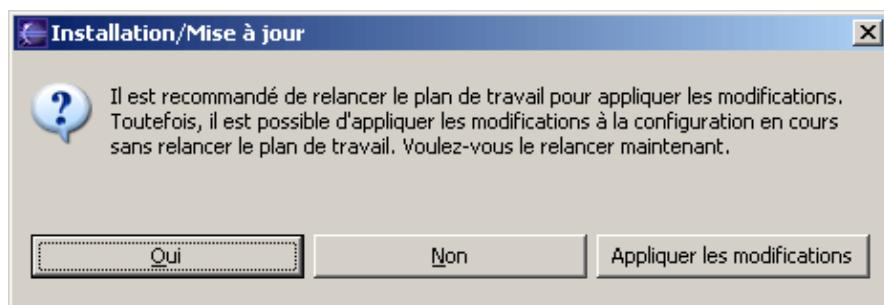


Cliquez sur le bouton « Fin »

Sur la page de vérification du dispositif, cliquez sur le bouton « Installer »



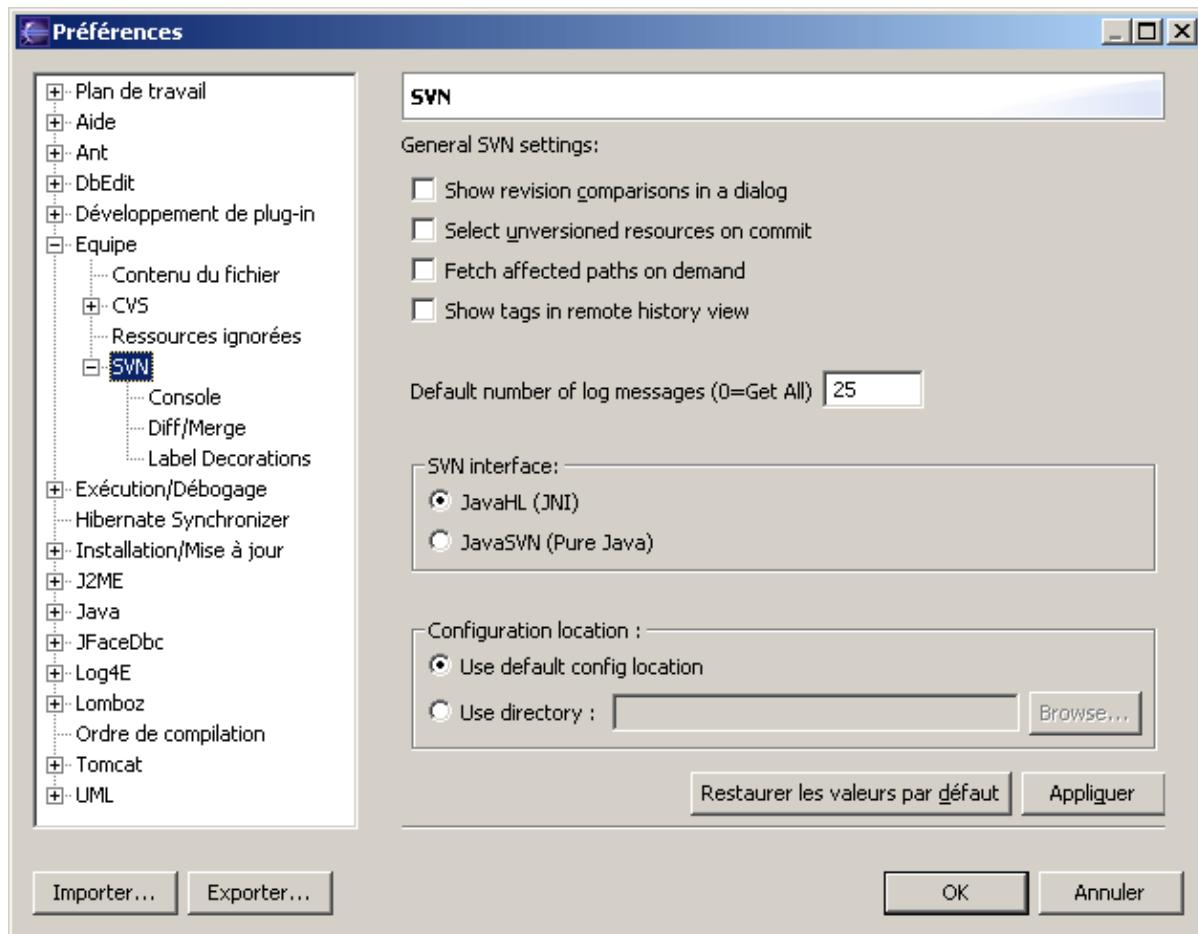
Les fichiers du plug-in sont téléchargés.



Cliquez sur le bouton « Oui » pour relancer Eclipse.

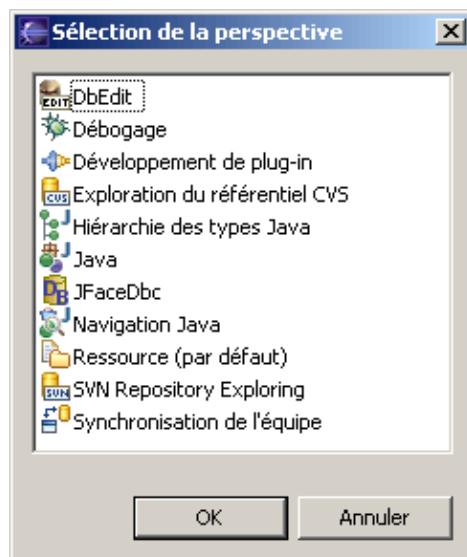
## 15.2.2. Paramétrage du plug-in

Le plug-in subclipse peut être configuré dans les préférences dans la rubrique « Équipe/SVN ».

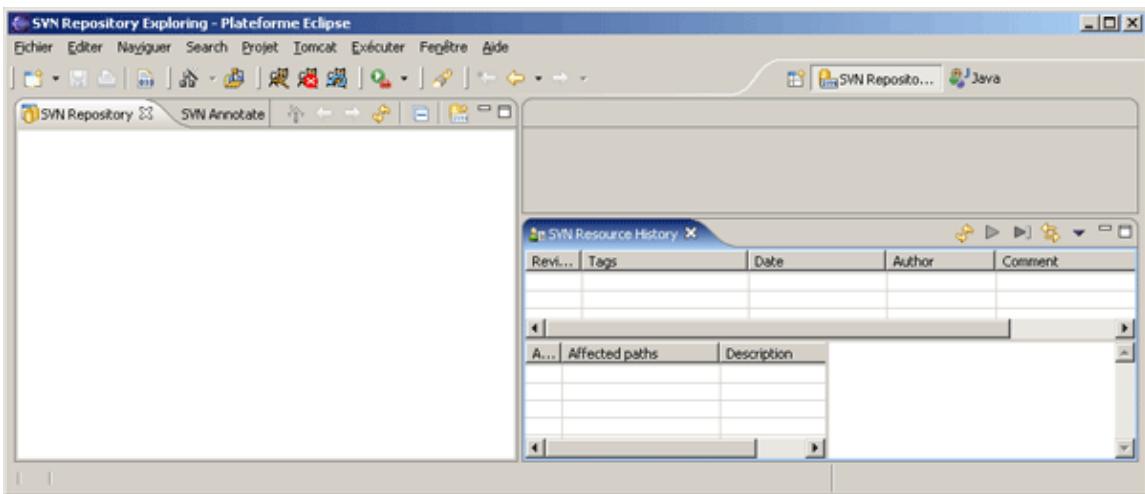


## 15.2.3. Utilisation du plug-in

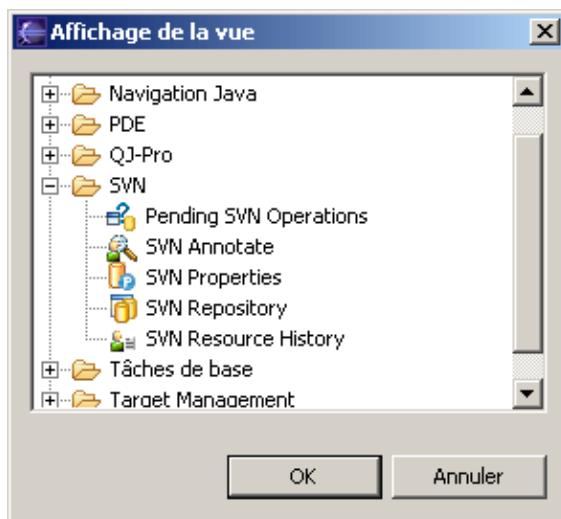
Le plug-in possède sa propre perspective nommée « SVN Repository Exploring »



Il suffit de demander son affichage



La perspective se compose de plusieurs vues.



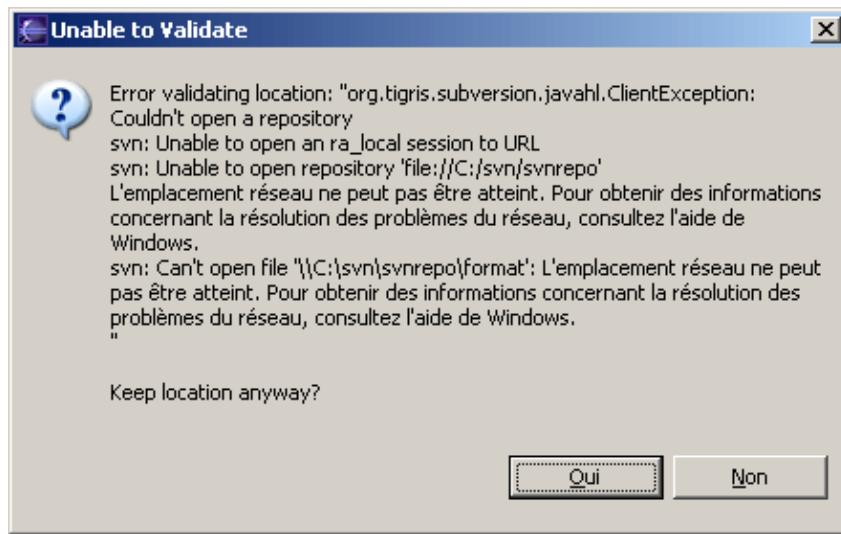
### 15.2.3.1. La connexion à un repository

Dans la vue SVN Repository, cliquez sur le bouton ou sélectionnez l'option « New / Respository Location ... » du menu contextuel.

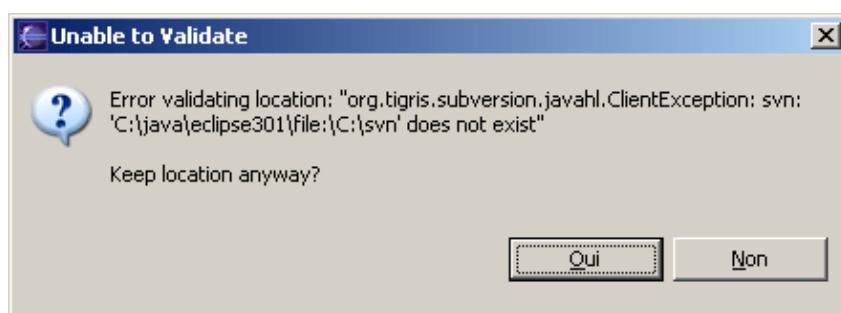


Saisissez l'url du repository et cliquez sur le bouton « Fin ».

Si l'url saisie n'est pas valide, un message d'erreur est affiché



Attention : le protocole `file:///` ne peut pas être utilisé avec le protocole JavaSVN. Toute tentative d'utilisation de ces deux protocoles ensemble affichera une erreur



La vue SVN Repository affiche le repository et les projets qu'il contient



A sa création, un projet est vide : il faut créer les répertoires de base en utilisant l'option « New / Remote Folder » du menu contextuel du projet.



Saisissez le nom du répertoire et cliquez que le bouton « Fin ».

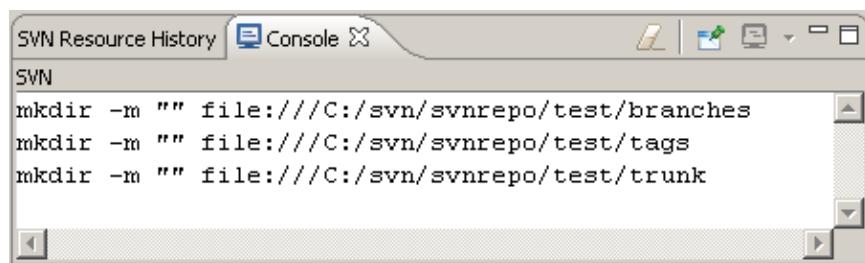
Le bouton suivant permet de saisir un commentaire sur l'opération.



Dans ce cas, saisissez le commentaire et cliquez sur le bouton « Fin ».

Il faut créer les répertoires branches, tags et trunk.

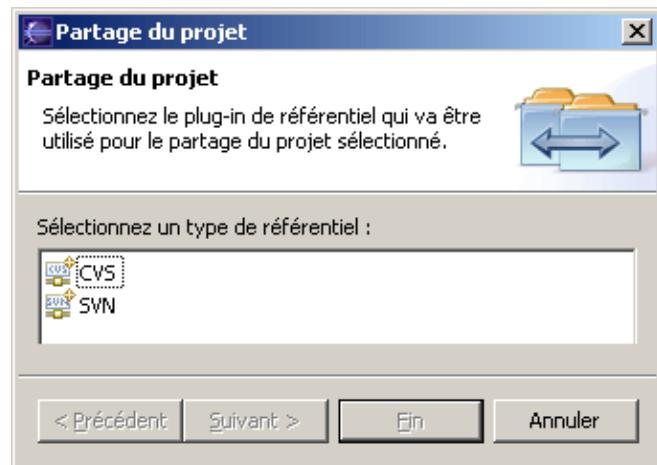
La vue console affiche les commandes subversion exécutées.



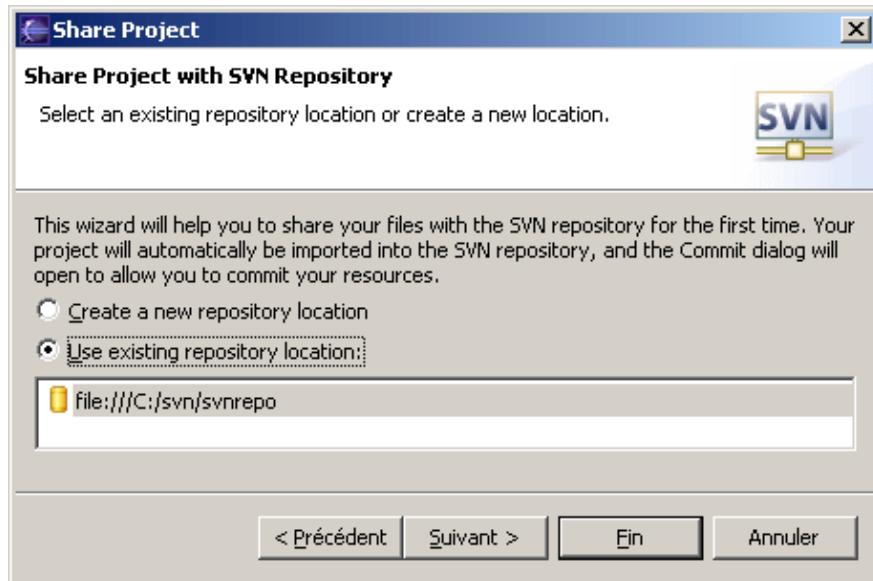
### 15.2.3.2. Ajouter un projet au repository

Dans la perspective Java, sélectionnez un projet dans la vue « Packages » et utiliser l'option « Equipe / Partager le projet ... ».

La première page de l'assistant permet de sélectionner le type de référentiel à utiliser.

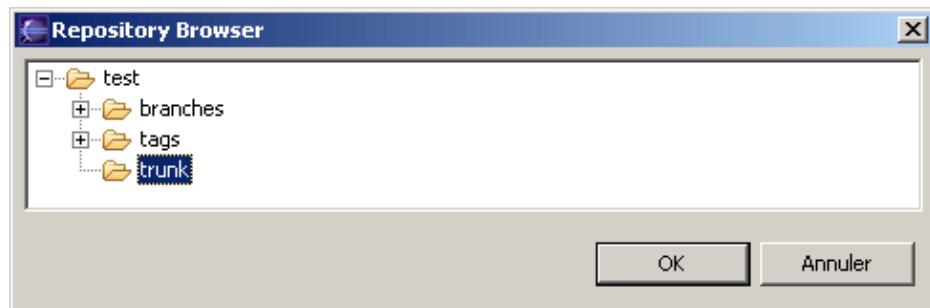


Sélectionnez SVN et cliquez sur le bouton "Suivant". La page suivante permet de sélectionner ou de créer le référentiel qui va contenir le projet.

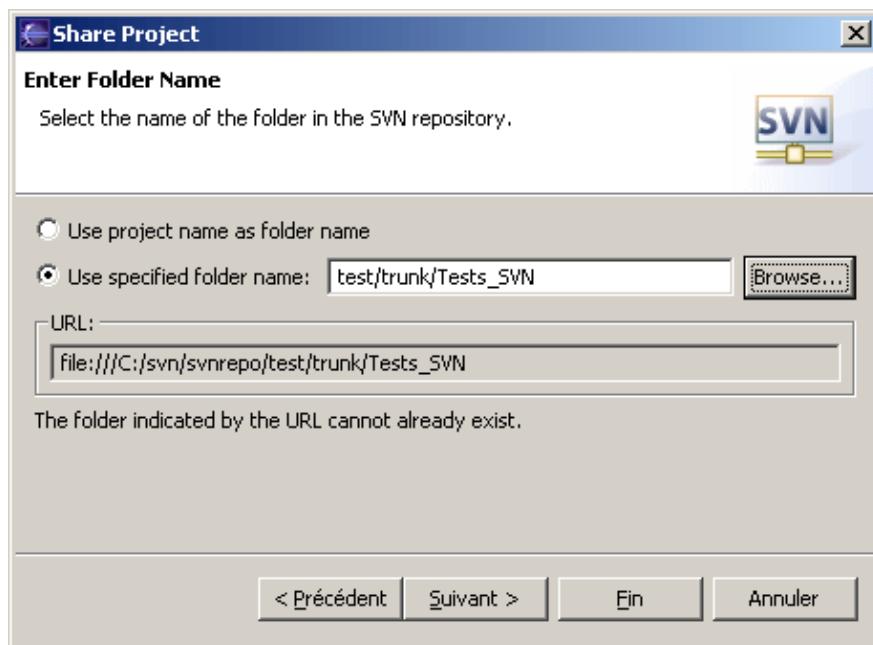


Sélectionnez le référentiel à utiliser et cliquer sur le bouton « Suivant ».

La page suivante permet de sélectionner le répertoire à utiliser. Cliquez sur « Use specified folder name » puis sur le bouton « Browse »



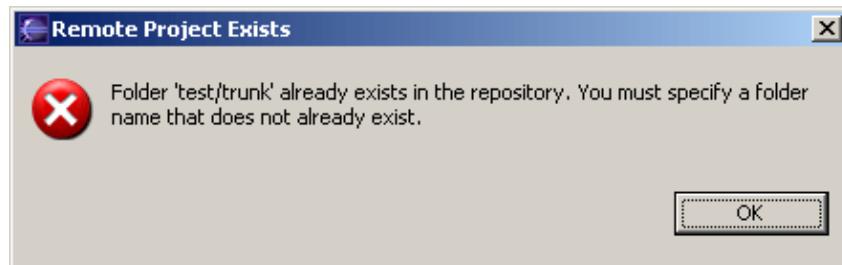
Sélectionnez le répertoire précédemment créé et cliquez sur le bouton « OK ».



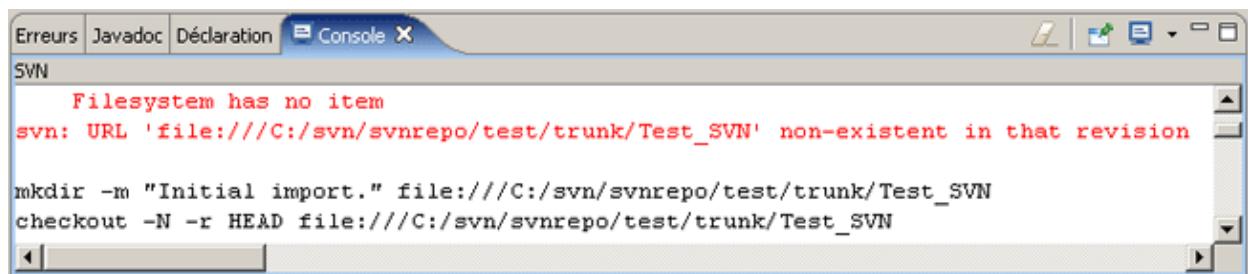
Remplacer « New Folder » par le nom du répertoire à utiliser puis cliquez sur le bouton « Suivant »



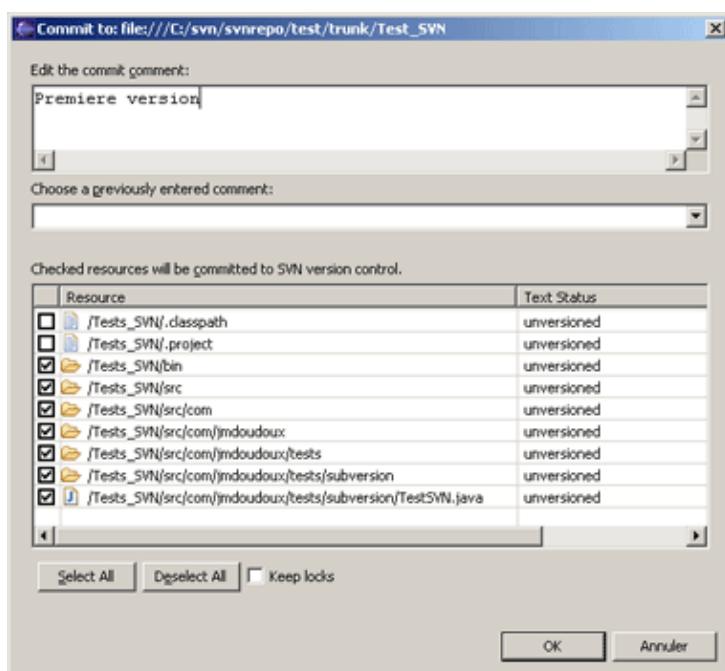
Cliquez sur le bouton « Fin ». Si le répertoire précisé existe déjà alors un message d'erreur est affiché



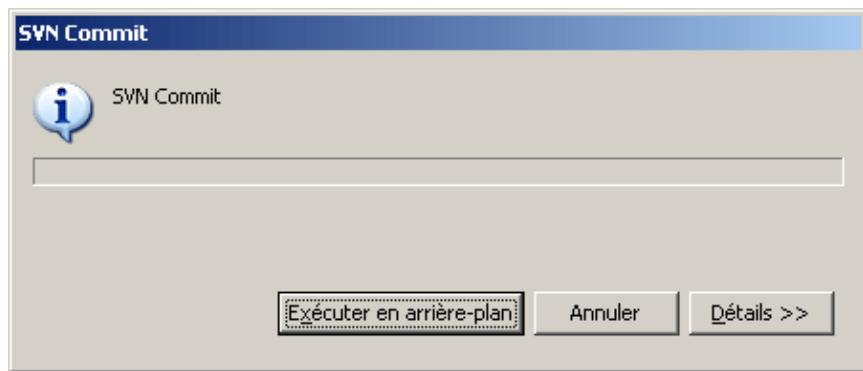
Une boîte de dialogue permet saisir les informations de commit de l'opération.



Les commandes subversion exécutées sont affichées dans la console, puis la boîte de dialogue « Commit » s'affiche.



Saisissez un commentaire, sélectionnez les ressources à ajouter dans le référentiel et cliquez sur le bouton "OK"



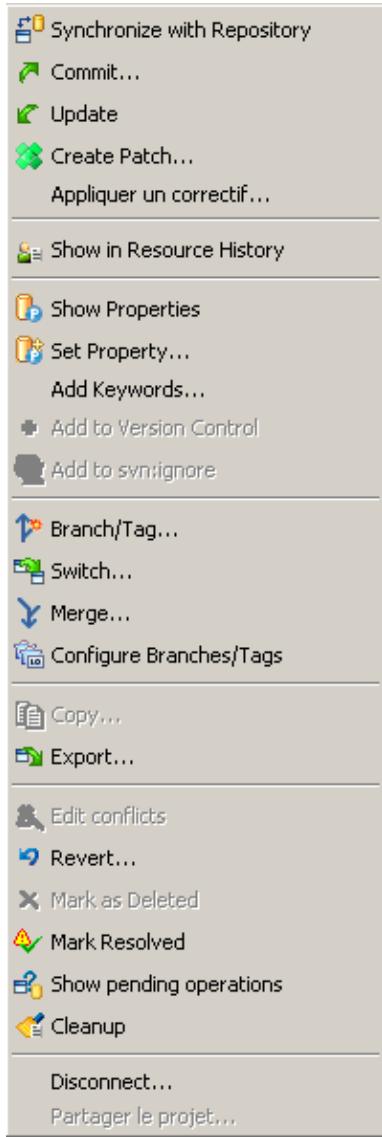
.La console affiche le détail des opérations exécutées.

```
Erreurs Javadoc Déclaration Console X
SVN
mkdir -m "Initial import." file:///C:/svn/svnrepo/test/trunk/Test SVN
checkout -N -r HEAD file:///C:/svn/svnrepo/test/trunk/Test SVN
Checked out revision 5.
add-N C:\java\workspace\Tests SVN\bin
  A C:\java\workspace\Tests SVN\bin
add-N C:\java\workspace\Tests SVN\src
  A C:\java\workspace\Tests SVN\src
add-N C:\java\workspace\Tests SVN\src\com
  A C:\java\workspace\Tests SVN\src\com
add-N C:\java\workspace\Tests SVN\src\com\jmdoudoux
  A C:\java\workspace\Tests SVN\src\com\jmdoudoux
add-N C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests
  A C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests
add-N C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests\subversion
  A C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests\subversion
add -N C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests\subversion\Tests
  A C:\java\workspace\Tests SVN\src\com\jmdoudoux\tests\subversion\Tests
commit -m "Première version" C:/java/eclipse31/workspace/Tests SVN/bin C:/java/eclipse31/workspace/Tests SVN/src
  Adding eclipse31/workspace/Tests SVN/bin
  Adding eclipse31/workspace/Tests SVN/src
  Adding eclipse31/workspace/Tests SVN/src/com
  Adding eclipse31/workspace/Tests SVN/src/com/jmdoudoux
  Adding eclipse31/workspace/Tests SVN/src/com/jmdoudoux\tests
  Adding eclipse31/workspace/Tests SVN/src/com/jmdoudoux\tests\subversion
  Adding eclipse31/workspace/Tests SVN/src/com/jmdoudoux\tests\subversion\1
Transmitting file data ...
Committed revision 6.
```

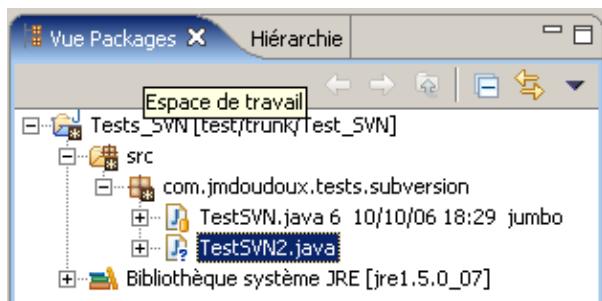
Dans la vue « Packages », les ressources sont affichées de façon à indiquer leur connexion avec le référentiel.



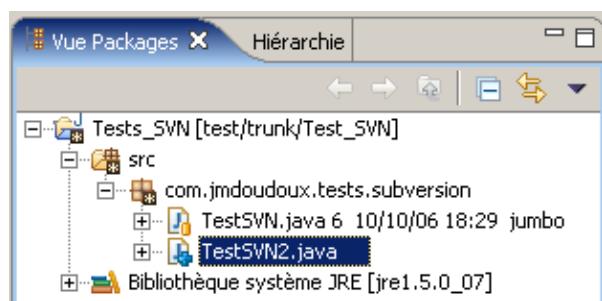
L'option « Equipe » du menu contextuel du projet permet maintenant un accès aux principales commandes de subversion



Par défaut, les nouvelles ressources ne sont pas ajoutées automatiquement au référentiel.

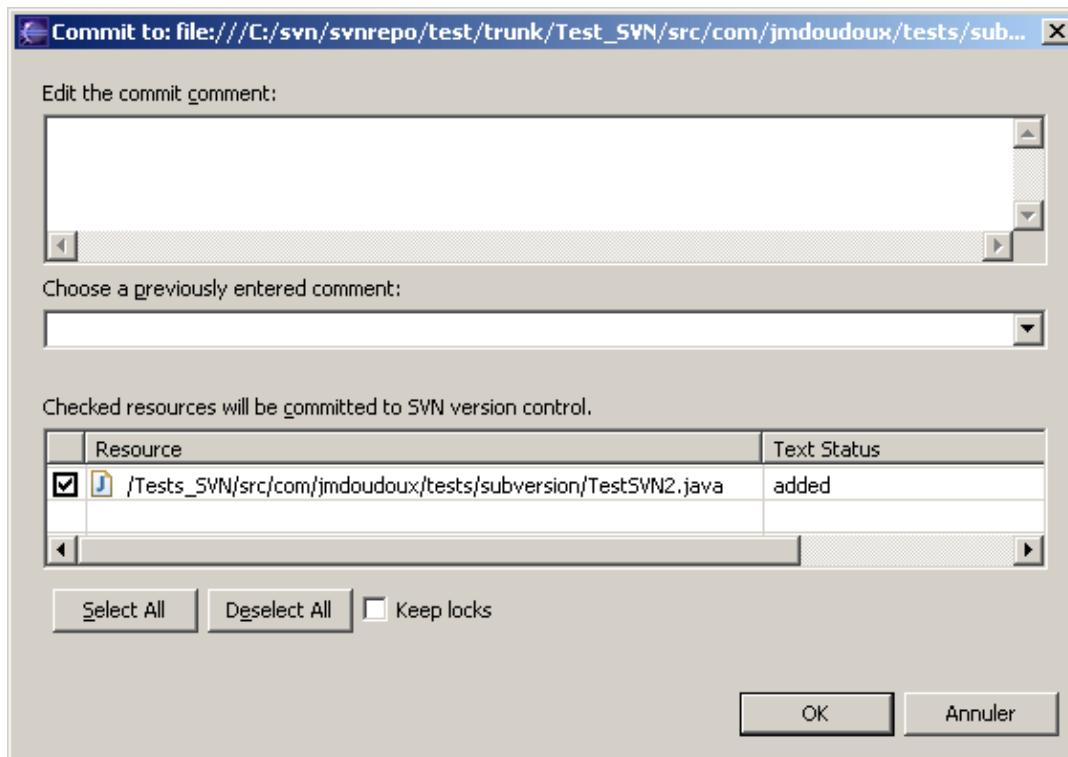


Elles apparaissent un point d'interrogation en bas à droite de leur icône. Elles doivent être ajoutées manuellement en utilisant l'option « Equipe / Add to version control» de leur menu contextuel.



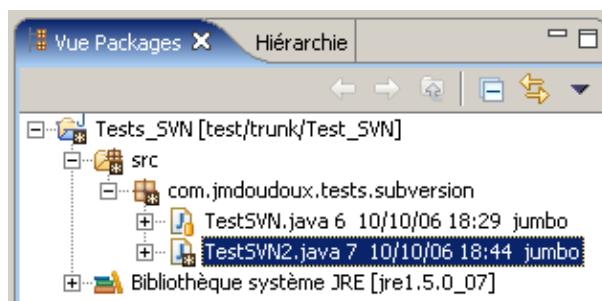
L'icône change en une petite flèche bleue indiquant que la ressource est ajoutée dans le référentiel mais n'est pas commitée.

Pour faire un commit des modifications vers le référentiel, il faut utiliser l'option « Equipe / Commit » du menu contextuel.



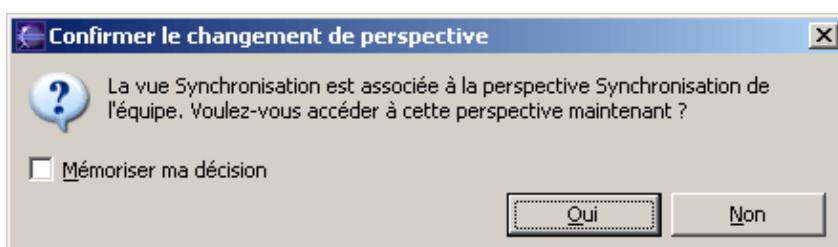
Par défaut les fichiers ajoutés sont sélectionnés. Saisissez un commentaire et cliquez sur le bouton « OK ».

Si des modifications sont apportées à une ressource du référentiel, son icône est modifiée.



### 15.2.3.3. Synchroniser l'espace de travail et le référentiel

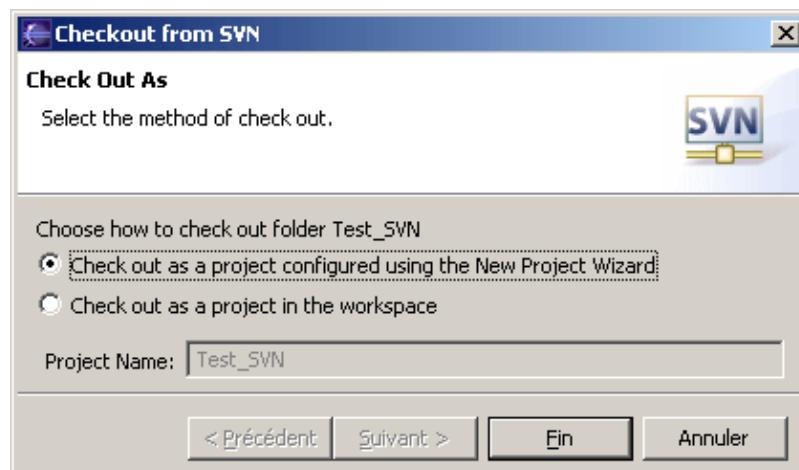
L'option « Equipe / Synchronise with repository » permet d'accéder à la perspective « Synchronisation de l'équipe ».



Cochez la case à cocher « Mémoriser ma décision » et cliquez sur « Oui » si cette boîte de dialogue apparaît.

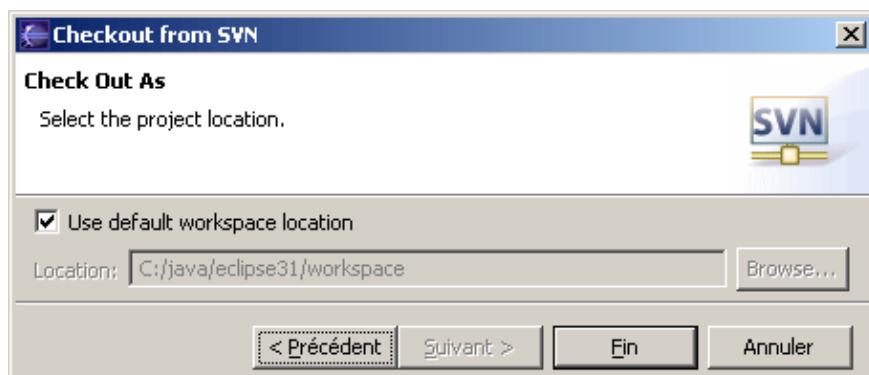
#### 15.2.3.4. Checkout d'un projet

Dans la perspective « SVN Repository Exploring », il est possible de demander le check out complet d'un projet : dans la vue « SVN Repository », il faut sélectionner l'élément et utiliser l'option « Checkout » du menu contextuel.



Pour obtenir les ressources dans un projet existant, sélectionnez « check out as project in the workspace » et cliquez sur le bouton « Suivant ».

La page suivante permet de sélectionner le chemin de l'espace de travail.



Cliquer sur le bouton « Fin »



### **15.3. Le plug in Subversive**

Ce plug-in est téléchargeable à l'url :

<http://www.polarion.org/index.php?page=overview&project=subversive>

## **Partie 4 : le développement avancé avec Java**

# **Partie 4 : le développement avancé avec Java**

Cette quatrième partie présente le développement avec Java nécessitant quelques paramétrage ou l'utilisation de plug-ins tiers.

Elle comporte les chapitres suivants :

- ◆ Des plug-ins pour le développement avec Java : présente quelques plug-ins tiers facilitant le développement avec Java.
- ◆ Le développement d'interfaces graphiques : présente plusieurs plug-ins pour faciliter le développement d'applications graphiques utilisant AWT, Swing ou SWT.
- ◆ Le plug-in TPTP (Test & Performance Tools Platform) : présente le plug-in TPTP (Test & Performance Tools Platform) a pour but de proposer un framework permettant de fournir des services de mesures de performance et d'automatisation de tests.
- ◆ Hibernate et Eclipse : Présente le plug-in Hibernate Synchronizer qui facilite la mise en oeuvre de l'outil de mapping Hibernate.

## 16. Des plug-ins pour le développement avec Java

# Chapitre 16

Il existe de nombreux plug-ins développés par des tiers pour enrichir Eclipse. Certains de ces plug-ins sont spécialement dédiés à faciliter le développement de code avec Java.

Ce chapitre va présenté plusieurs de ces plug-ins :

- Jalopy : ce plug-in permet de mettre en oeuvre l'outil open source du même nom qui permet de formatter le code source
- Log4E : ce plug-in permet de faciliter la mise en oeuvre d'un système de log (log4J, api de logging du JDK 1.4 ou common logging)

### 16.1. Le plug-in Jalopy

Jalopy est un utilitaire open source très pratique qui permet de formatter du code source Java et même de vérifier l'application de normes de codage.

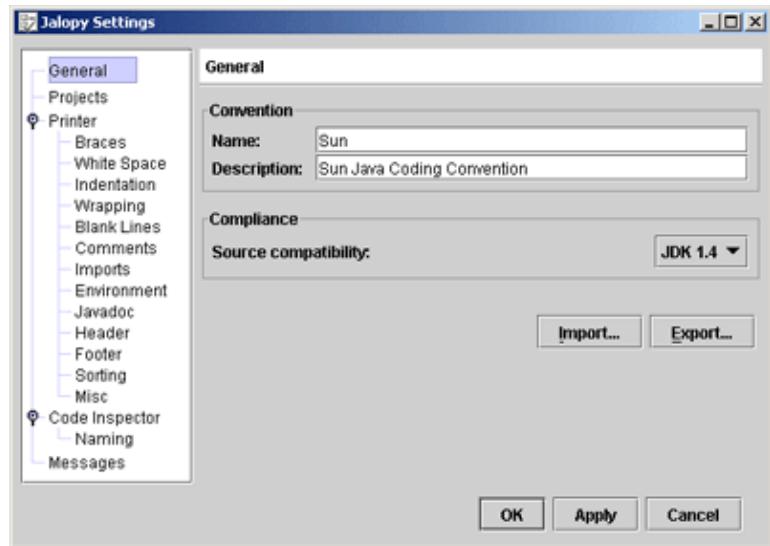
Il permet notamment :

- d'indenter le code
- de générer des modèles de commentaires Javadoc dynamiques en fonction des éléments du code à documenter (par exemple générer un tag @param pour chaque paramètre d'une méthode)
- d'organiser l'ordre des clauses import
- d'organiser l'ordre des membres d'une classe selon leur modificateur
- de vérifier l'application de normes de codage,
- ...

Il existe des plug-ins pour plusieurs IDE dont un pour Eclipse : il suffit de télécharger le fichier jalopy-eclipse-0.2.6.zip sur le site <http://jalopy.sourceforge.net/download.html>

Pour installer le plug-in, il faut décompresser le contenu de l'archive dans un réperoire temporaire et copier le réperoire de.hunsicker.jalopy.plugin.eclipse\_0.2.6 dans le réperoire plugins d'Eclipse.

L'option "Jalopy preferences" du menu "Fenêtre" permet de paramétrer Jalopy



Pour la mise en oeuvre, il suffit d'utiliser le menu contextuel "Format with Jalopy" de l'éditeur de code Java.

#### Exemple :

```
package com.moi.test;

public class TestJalopy {

    public static void main(String[] args) {}

    public void maMethode(int a, int b) {}

    private int maMethode2(int a) { return a; }

    public void maMethode3() {}

}
```

#### Résultat :

```
package com.moi.test;

/**
 * DOCUMENT ME!
 *
 * @author $author$
 * @version $Revision$
 */
public class TestJalopy {
    /**
     * DOCUMENT ME!
     *
     * @param args DOCUMENT ME!
     */
    public static void main(String[] args) {
    }

    /**
     * DOCUMENT ME!
     *
     * @param a DOCUMENT ME!
     * @param b DOCUMENT ME!
     */
    public void maMethode(int a, int b) {
    }

    /**
     * DOCUMENT ME!
     */
}
```

```

public void maMethode3() {
}

/**
 * DOCUMENT ME !
 *
 * @param a DOCUMENT ME !
 *
 * @return DOCUMENT ME !
 */
private int maMethode2(int a) {
    return a;
}
}

```

Le formattage du code source est réalisé en tenant compte des nombreux paramètres définis dans Jalopy.

Il est également possible de demander le formattage de l'ensemble des fichiers source Java contenus dans un ou plusieurs packages ou répertoires. Il suffit de sélectionner le ou les packages ou répertoires et de sélectionner l'option "Format" du menu contextuel.

## 16.2. Log4E

Log4E est un plug-in gratuit dont le but est de faciliter l'utilisation d'une API de logging dans du code java.

Log4E propose en standard d'utiliser 3 API : Log4J, l'api Logging du JDK 1.4 et l'api Common Logging du projet Jakarta.

Le site officiel de ce plug-in est à l'url : [http://log4e.jayefem.de/index.php/Main\\_Page](http://log4e.jayefem.de/index.php/Main_Page)

### 16.2.1. Installation

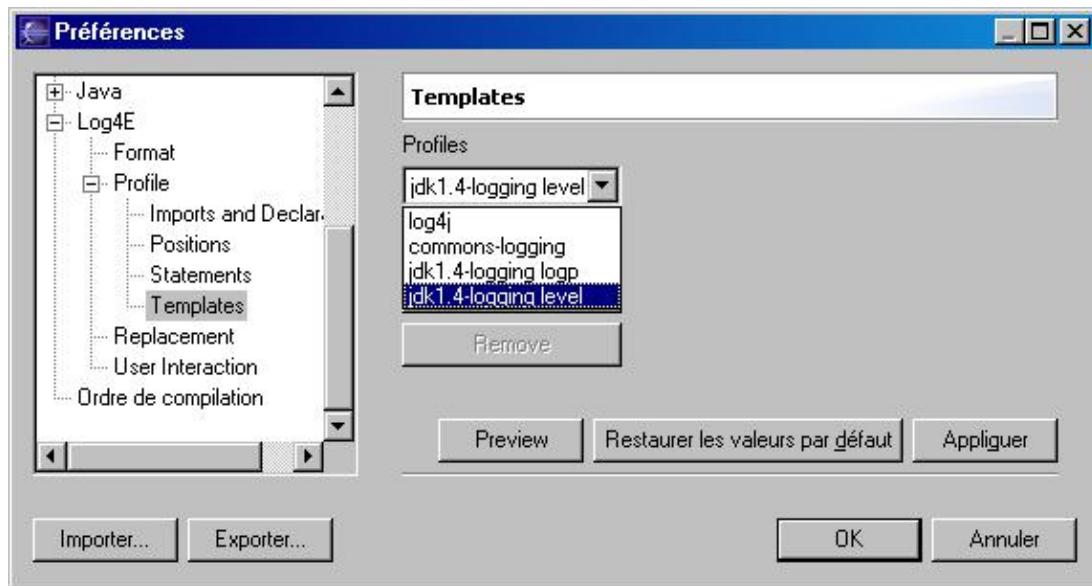
Il y a deux façons d'obtenir le plug-in et de l'installer :

- télécharger l'archive .zip contenant le plug-in et le décompresser dans le répertoire d'installation d'Eclipse
- utiliser la fonctionnalité de mise à jour d'Eclipse avec l'url : <http://log4e.jayefem.de/update>

### 16.2.2. Les paramètres

Le plug-in Log4E possède de nombreux paramètres modifiables dans la fenêtre de gestion des paramètres.

Un des paramètres le plus important se situe à l'arborescence Log4E / Profile / Templates



Le profile permet de sélectionner l'api qui sera utilisée par le plug-in lors de la génération de code pour les opérations demandées.

### 16.2.3. Utilisation

Exemple :

```
package com.jmd.test.java;

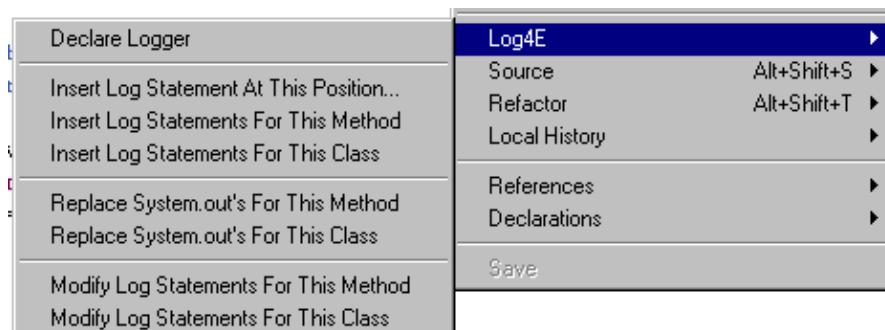
public class TestLog4E {

    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        System.out.println("resultat = " + additionner(a, b));
    }

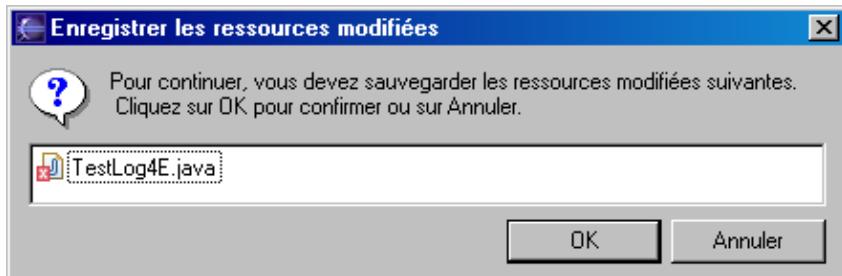
    public static int additionner(int a, int b) {
        int res = a + b;
        return res;
    }
}
```

L'api de logging sélectionnée dans la préférences pour les exemples de cette section est celle du JDK 1.4.

Le menu contextuel de l'éditeur de code se voit ajouter une nouvelle option nommée Log4E.



Avant de pouvoir utiliser une de ces options, il est nécessaire de sauvegarder le code source pour permettre au plug-in de réaliser les modifications.



Cette nouvelle entrée du menu contextuel, propose plusieurs options réparties en 4 groupes.

Toute opérations impliquant une modification du code est soumise à l'approbation de l'utilisation sous la forme d'une vue permettant de comparer le code avant et après modification et d'accepter ou non les modifications.

L'option « Declare Logger » permet de demander l'insertion dans le code des clauses imports ainsi que la déclaration des objets nécessaires à l'utilisation de l'api de logging .

#### Exemple :

```
package com.jmd.test.java;

import java.util.logging.Level;
import java.util.logging.Logger;

public class TestLog4E {

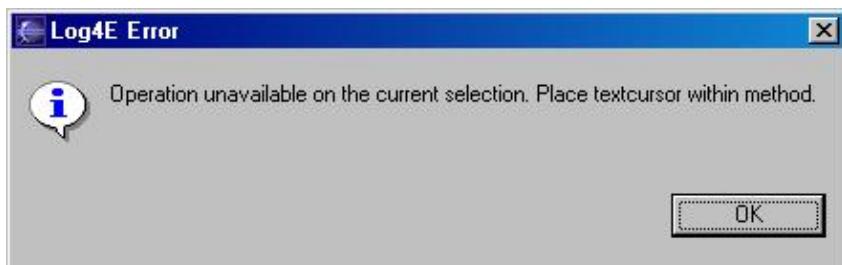
    /**
     * Logger for this class
     */
    private static final Logger logger = Logger.getLogger(TestLog4E.class
            .getName());

    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        System.out.println("resultat = " + additionner(a, b));
    }

    public static int additionner(int a, int b) {
        int res = a + b;
        return res;
    }
}
```

L'option Insert Log Statement At The Position ... permet d'insérer du code pour utiliser l'api de logging.

Pour utiliser cette option, il est nécessaire de placer le curseur de l'éditeur de code dans une méthode, sinon un message d'erreur est affiché.



L'option Insert Log Statement For This Method permet d'insérer automatique des appels à l'api de logging dans la méthode : au début de la méthode, à la fin de la méthode et dans les blocs try/catch.

### Exemple avec la méthode additionner()

```
...
    public static int additionner(int a, int b) {
        if (logger.isLoggable(Level.FINE)) {
            logger.fine("start");
        }
        int res = a + b;
        if (logger.isLoggable(Level.FINE)) {
            logger.fine("end");
        }
        return res;
    }
...
```

Pour utiliser cette option, il est nécessaire de placer le curseur de l'éditeur de code dans une méthode, sinon un message d'erreur est affiché.

L'option `Insert Log Statement For This Class` est identique à l'option précédente mais effectue les traitements sur toute la classe.

L'option `Replace System.out's For This Method` permet de remplacer les utilisations de la classe `System.out` ou `System.err` par l'utilisation de l'api de logging.

Pour utiliser cette option, il est nécessaire de placer le curseur de l'éditeur de code dans une méthode, sinon un message d'erreur est affiché.

L'option `Replace System.out's For This Class` est identique à l'option précédente mais effectue les traitements sur toute la classe.

## 17. Le développement d'interfaces graphiques

# Chapitre 17

### 17.1. Eclipse et SWT

Eclipse est développé en utilisant SWT comme API pour le développement de son interface graphique pour l'utilisateur. Pour le développement de plug-ins, l'utilisation de SWT est donc fortement recommandée mais il est aussi possible d'utiliser Eclipse et SWT pour le développement des applications standalone.

#### 17.1.1. Configurer Eclipse pour développer des applications SWT

Sans plug-ins, il est possible de développer des applications graphiques utilisant SWT avec Eclipse. Bien sûr, l'inconvénient majeur est de devoir écrire tout le code "à la main".

Pour le développement d'applications SWT, il est nécessaire de configurer certains éléments notamment le classpath. Dans cette section, la version de SWT utilisée est la 2.1.

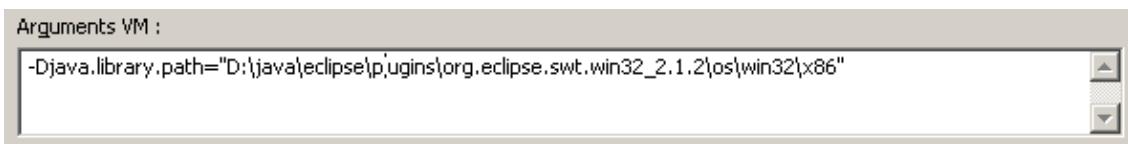
Pour utiliser SWT dans un projet, il faut ajouter dans son classpath le fichier swt.jar. Dans les propriétés du projet, sélectionnez « Chemin de compilation Java » et cliquez sur le bouton « Ajouter des fichiers jar externes ... ».

Une boîte de dialogue permet de sélectionner le fichier swt.jar. Ce fichier est dépendant du système d'exploitation sur lequel l'application va s'exécuter. Par exemple sous Windows, ce fichier se trouve dans le sous répertoire plugins\org.eclipse.swt.win32\_2.1.0\ws\win32\ du répertoire d'installation d'Eclipse. Cliquez sur le bouton « Ok » pour valider les modifications et recompiler le projet.

Pour pouvoir exécuter le projet, il faut que l'application puisse accéder à une bibliothèque native particulière qui sous Windows se nomme swt-win32-2135.dll. Ce fichier se trouve dans le sous répertoire plugins\org.eclipse.swt.win32\_2.1.2\os\win32\x86 du répertoire d'installation d'Eclipse. Le nom et le chemin de cette bibliothèque dépend de la version de SWT qui est utilisée.

Il y a plusieurs solutions possibles pour mettre ce fichier à disposition d'une application :

1. précisez le chemin de la bibliothèque dans les paramètres de la machine virtuelle lors de l'exécution  
Pour cela choisissez, « Exécuter ... » pour ouvrir la boîte de dialogue de configuration d'exécution.  
Sur l'onglet « Arguments ... », saisir  
-Djava.library.path="chemin\_complet\_du\_fichier" et cliquez sur le bouton « Appliquer » puis sur le bouton « Exécuter »



L'inconvénient de cette méthode est que cette configuration doit être effectuée pour chaque configuration de lancement.

2. Ajouter le répertoire qui contient la bibliothèque à la variable PATH sous Windows ou LD\_LIBRARY\_PATH sous Unix.
3. Il est possible de copier la bibliothèque dans un répertoire contenu dans la variable java.library.path. L'inconvénient de cette méthode est qu'il faut recopier la bibliothèque à chaque nouvelle version de SWT utilisée.
4. Il est possible de copier la bibliothèque dans le répertoire racine de l'application. L'inconvénient de cette méthode est qu'il faut recopier la bibliothèque dans chaque projet qui utilise SWT.

Lors de l'exécution, si la bibliothèque ne peut être accédée par l'application, une exception est levée.

Exemple :

```
java.lang.UnsatisfiedLinkError: no swt-win32-2135 in java.library.path
```

Pour une utilisation sous Linux, la configuration est similaire. Il est cependant nécessaire d'ajouter dans le classpath une bibliothèque supplémentaire nommée swt-pi.jar. Ce fichier se trouve au même endroit que le fichier swt.jar dans le répertoire eclipse/plugins/org.eclipse.swt\_gtk\_3.0.0/gs/gtk/

Si ce fichier n'est pas inclus dans le classpath, une exception est levée lors de l'exécution :

Exemple :

```
Exception in thread "main" java.lang.NoClassDefFoundError: org/eclipse/swt/internal/gtk
/OS
    at org.eclipse.swt.internal.Converter.wcsToMbcS(Converter.java:63)
    at org.eclipse.swt.internal.Converter.wcsToMbcS(Converter.java:54)
    at org.eclipse.swt.widgets.Display.<clinit>(Display.java:118)
    at com.jmd.test.swt.TestSWT.main(TestSWT.java:9)
```

Sous Linux, il faut que l'application puisse accéder à aux bibliothèques natives qui se situe dans le répertoire eclipse/plugins/org.eclipse.swt\_gtk\_3.0.0/os/linux/x86

Le plus simple de mettre à jour la variable d'environnement LD\_PATH du système ou d'utiliser l'option de la JVM lors de l'exécution (en adaptant le chemin d'Eclipse) :

-Djava.library.path="/home/java/eclipse/plugins/org.eclipse.swt\_gtk\_3.0.0/os/linux/x86".

Si ces bibliothèques ne peuvent être trouvées lors de l'exécution, une exception est levée.

Exemple :

```
Exception in thread "main" java.lang.UnsatisfiedLinkError: no swt-pi-gtk-3062
in java.library.path
    at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1517)
    at java.lang.Runtime.loadLibrary0(Runtime.java:788)
    at java.lang.System.loadLibrary(System.java:834)
```

## 17.1.2. Un exemple très simple

Exemple :

```
import org.eclipse.swt.*;
import org.eclipse.swt.graphics.*;
import org.eclipse.swt.widgets.*;

public class TestSWT2 {

    public static void main(String[] args) {
        Display display = new Display();
        Shell shell = new Shell(display);
        shell.setText("Test");

        Composite composite = new Composite(shell, SWT.NONE);
        Color couleur = new Color(display,131,133,131);
        composite.setBackground(couleur);

        Label label = new Label(composite, SWT.NONE);
        label.setBackground(couleur);
        label.setText("Saisir la valeur");
        label.setBounds(10, 10, 100, 25);

        Text text = new Text(composite, SWT.BORDER);
        text.setText("mon texte");
        text.setBounds(10, 30, 100, 25);

        Button button = new Button(composite, SWT.BORDER);
        button.setText("Valider");
        button.setBounds(10, 60, 100, 25);
        composite.setSize(140,140);

        shell.pack();
        shell.open();

        while (!shell.isDisposed())
            if (!display.readAndDispatch())
                display.sleep();

        couleur.dispose();
        display.dispose();
    }
}
```

Si l'environnement est correctement configuré comme expliqué dans la section précédente, l'exécution de l'application se fait comme tout autre application.



## 17.2. Le plug-in Eclipse V4all

V4All est un projet open source qui a pour but de développer un plug-in pour Eclipse permettant le développement d'interfaces graphiques avec Swing ou SWT.

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
V4All	2.1.1.9.

### 17.2.1. Installation

V4All requiert l'installation du plug-in GEF (Graphical Editing Framework) en exécutant les étapes suivantes :

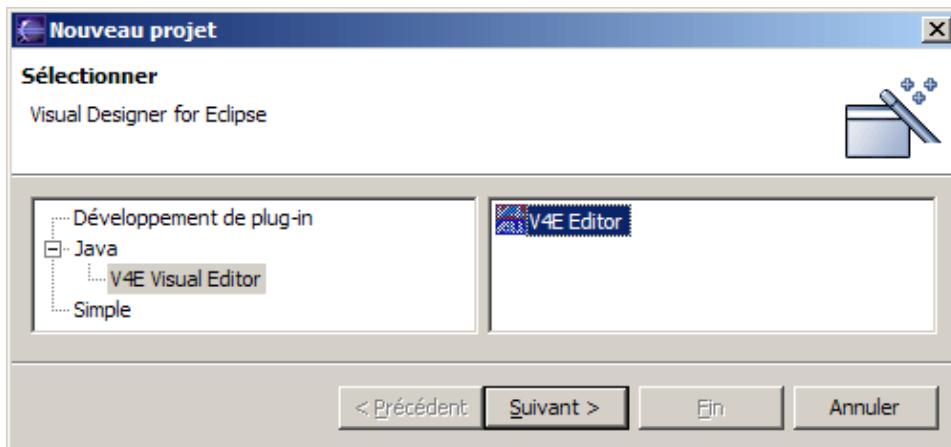
- téléchargez le fichier zip contenant le runtime de GEF sur le site d'Eclipse <http://download.eclipse.org/tools/gef/downloads/>
- décompressez ce fichier dans le répertoire qui contient Eclipse.
- relancez Eclipse et cliquez sur le bouton « Fin » pour valider les modifications apportées par le plug-in
- cliquez sur le bouton « Oui » lors de la demande de relance de l'application.

Pour installer V4All, il faut suivre les étapes suivantes :

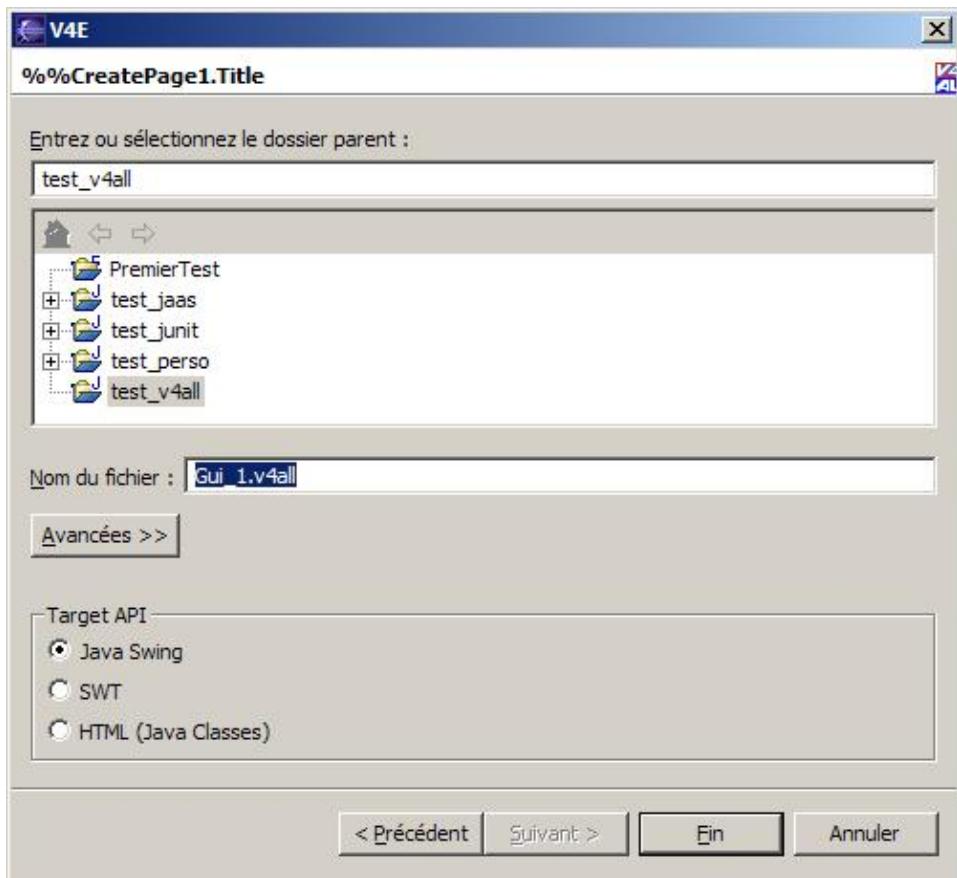
- téléchargez le fichier v4all\_2\_1\_1\_9.zip sur le site <http://sourceforge.net/projects/v4all>
- décompressez son contenu dans le répertoire qui contient Eclipse.
- relancez Eclipse et cliquez sur le bouton « Fin » pour valider les modifications apportées par le plug-in
- cliquez sur le bouton « Oui » lors de la demande de relance de l'application.

### 17.2.2. Utilisation

Pour utiliser V4all, il faut créer ou utiliser un projet Java existant, puis il faut créer un nouveau projet de type « V4E Editor ».

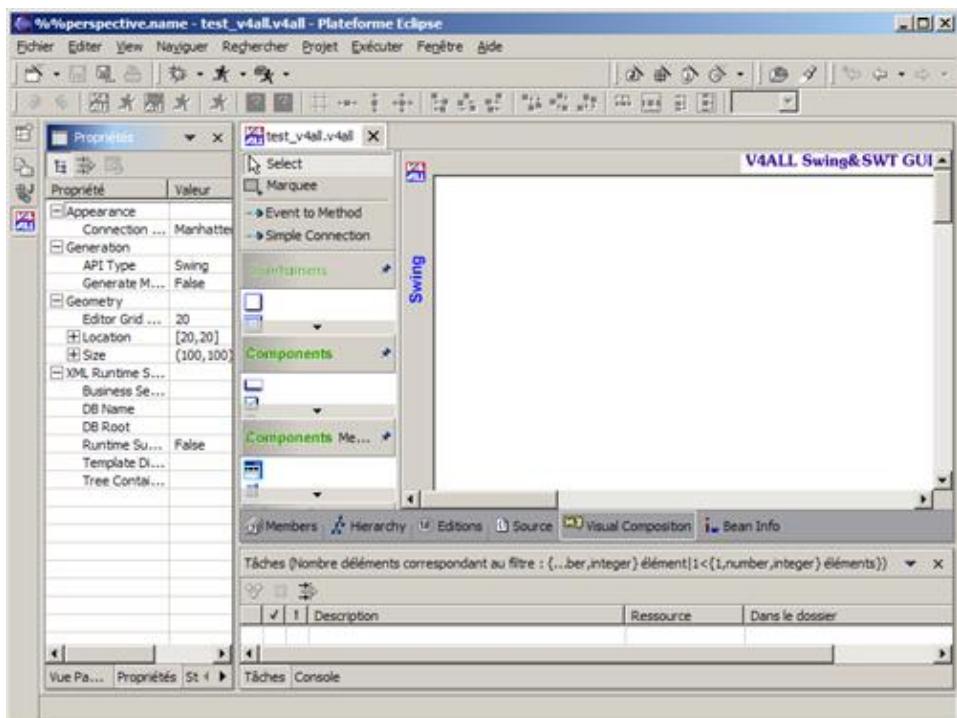


Cliquez sur le bouton « Suivant ».



Un assistant permet de sélectionner le projet Java qui va contenir le code, de saisir le nom du fichier qui va contenir les informations, et de sélectionner l'API à utiliser

Il suffit de cliquer sur le bouton « Fin » pour que la perspective dédiée à V4All s'affiche.



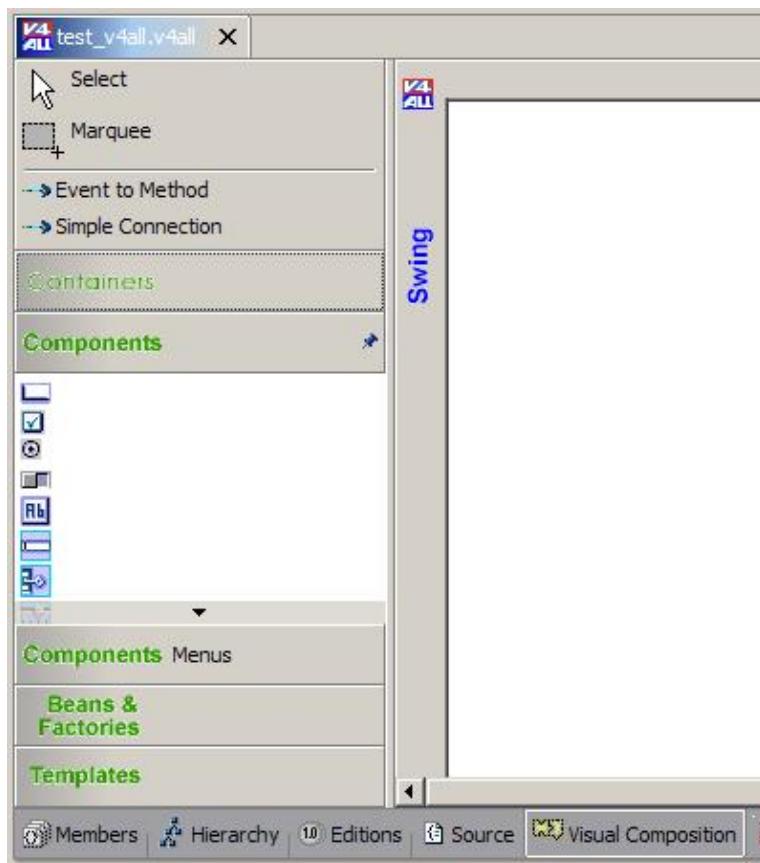
V4All propose son propre éditeur pour réaliser de manière WYSIWYG des interfaces graphiques.

Cet éditeur se compose de deux parties :

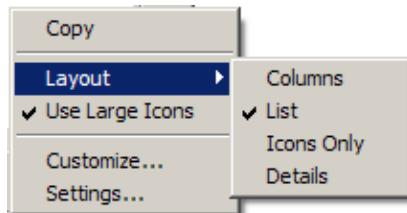
- une barre latérale qui permet de définir une action et de sélectionner un composant

- une zone de travail sur laquelle les composants sont déposés par cliquer/glisser

Par défaut, chacun des éléments de la barre est ouvert, ce qui les rend peu visibles. Pour ouvrir un seul élément, il suffit de double cliquer sur son titre.



L'apparence de la barre peut être configurée pour faciliter son utilisation : le menu contextuel de la barre propose plusieurs options intéressantes :



L'utilisation de l'option « Use large Icons » améliore la visibilité des icônes.

L'option « Layout » permet de sélectionner le mode d'affichage des composants : l'utilisation de l'option « Details » permet d'avoir un libellé associé à l'icône.

### 17.2.3. Un exemple simple

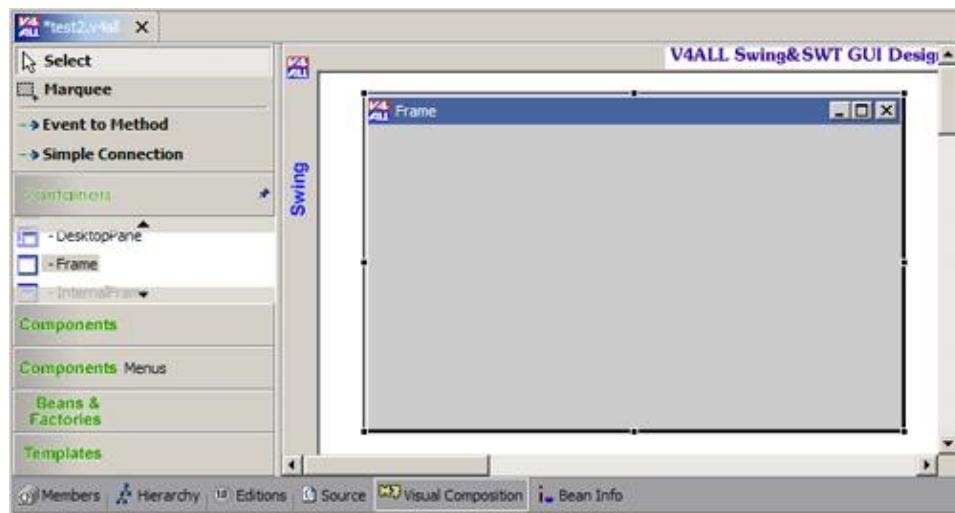
Il faut créer un nouveau projet V4All.

Dans les propriétés de ce projet, mettre la valeur « True » à la propriété « Generation / Generate Main ».

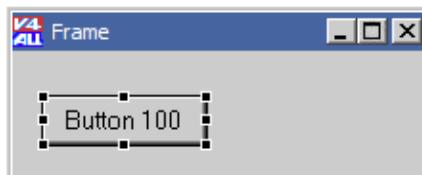
Il faut faire un cliquer/glisser du composant Frame sur la zone d'édition

A partir de ce moment, il est possible de demander la génération du code et son exécution en utilisant l'option « Run » du menu contextuel « Code generation And Run ».

L'application se lance et la fenêtre définie s'affiche : elle est déjà fonctionnelle et il suffit de cliquer sur le bouton de fermeture pour arrêter l'application.



Pour ajouter un bouton, il suffit de faire un cliquer/glisser du composant sur la Frame : le conteneur qui va recevoir le composant change de couleur.



Chaque composant possède de nombreuses propriétés qu'il est possible de modifier dans la vue "Propriétés". Cette vue affiche les propriétés du composant sélectionné dans l'éditeur.

Pour un bouton, il est par exemple possible de modifier le nom du composant avec la propriété « Basic / Bean Name », le texte du bouton avec « Component / Text », ...

Il faut procéder de la même façon pour ajouter une zone de texte (un composant de type TextField).

### 17.3. Eclipse VE

VE (Visual Editor) est un framework dont le but est de faciliter le développement de plug-in permettant la réalisation d'interfaces graphiques. Même si VE est proposé avec une implémentation de référence proposant la conception d'interfaces graphiques reposant sur SWT, Swing ou AWT, le but du framework est de pouvoir proposer à terme la conception d'interfaces graphiques reposant sur d'autres bibliothèques, pas nécessairement développées en ou pour Java.

Produit	Version utilisée dans cette section
Eclipse	3.1.1.
J2RE	1.5.0_06
Eclipse VE	1.1.0.1
Eclipse VE NLPack1 runtime	1.1.0.1

Le site officiel du plug-in VE est à l'url <http://www.eclipse.org/vep/>.

Plusieurs versions du plug-in ont été diffusées :

Version	Date
1.1.01	Septembre 2005
1.1	Juillet 2005
1.0.2.2	Juin 2005
1.0.1.1	Octobre 2004
1.0	Septembre 2004
0.5	Décembre 2003

### 17.3.1. Installation

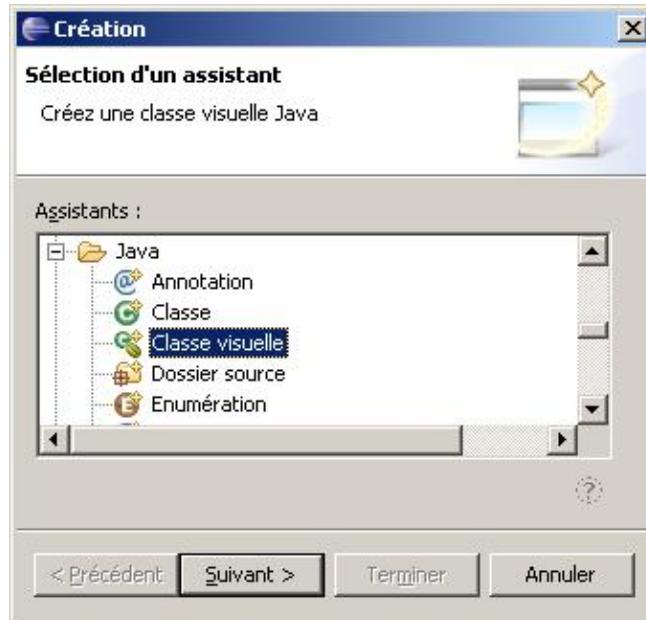
La version 1.1.0.1 du plug-in VE nécessite plusieurs pré-requis pour fonctionner :

- un JRE 1.4.2 minimum
- la version 3.1. minimum d'Eclipse
- et les plug-ins GEF version 3.1 et EMF version 2.1. Ces deux plug-ins sont à télécharger et à installer avant d'installer VE si ils ne sont pas déjà installés.

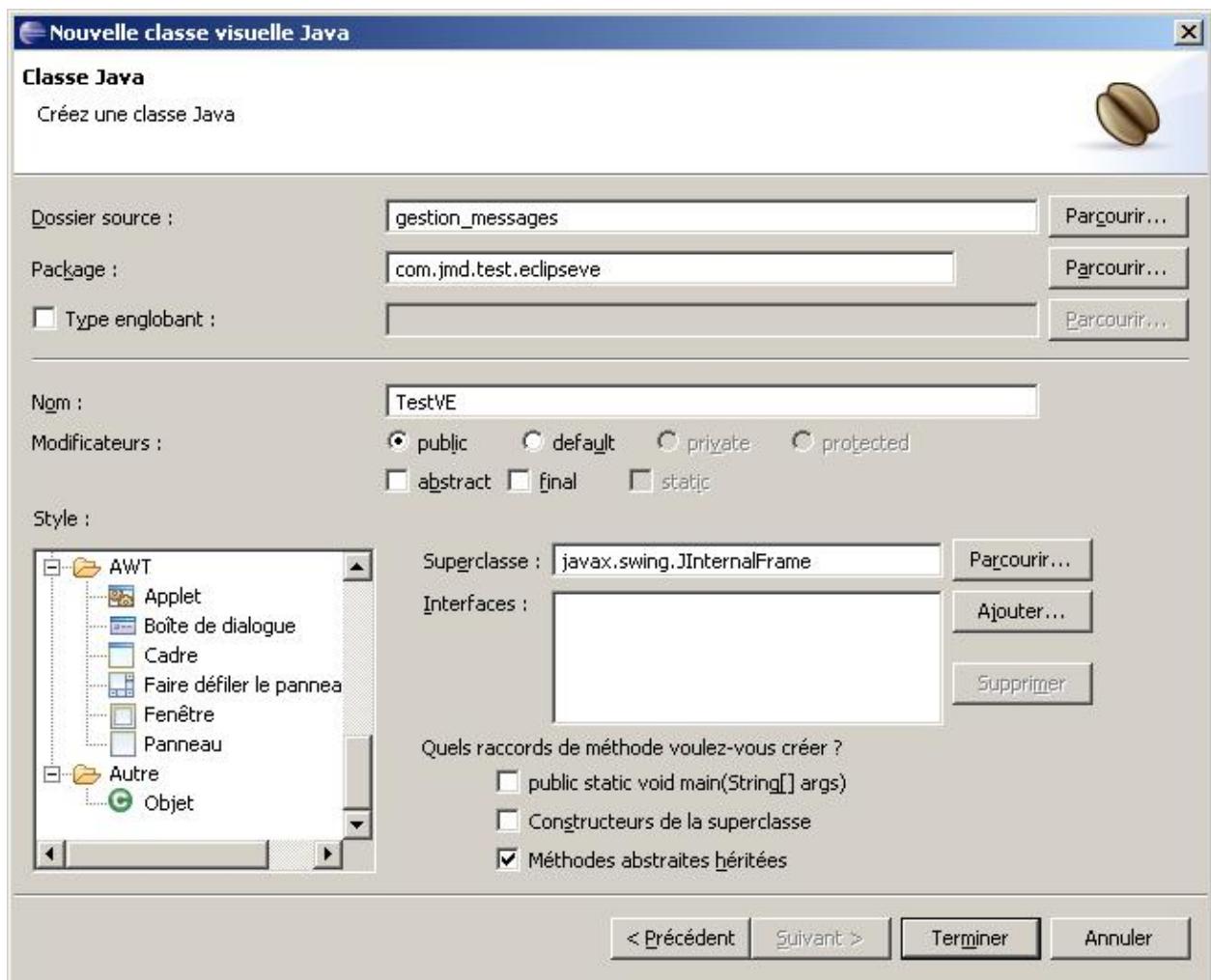
Il faut télécharger les fichiers VE-runtime-1.1.0.1.zip et NLpack1-VE-runtime-1.1.0.1.zip à partir du site officiel du plug-in et décompresser le contenu de ces deux archives dans un répertoire du système. Il suffit alors de copier les répertoires features et plugins décompressés dans le répertoire d'installation d'Eclipse.

### 17.3.2. Mise en oeuvre et présentation rapide

Dans un projet Java existant, il faut créer une nouvelle entité de type «Java / Classe Visuelle».



La première page de l'assistant permet de fournir des informations sur la classe Java qui sera générée.

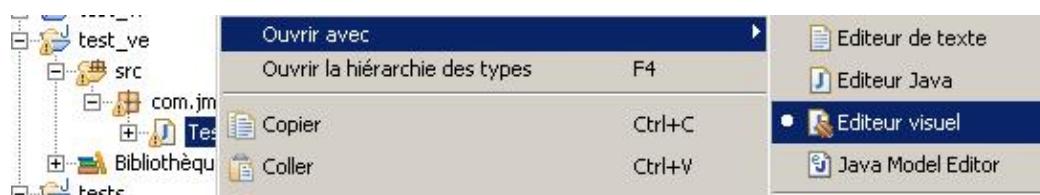


Il faut saisir le nom de la classe et renseigner les informations utiles (packages, style, super classe), puis cliquer sur le bouton « Terminer ».

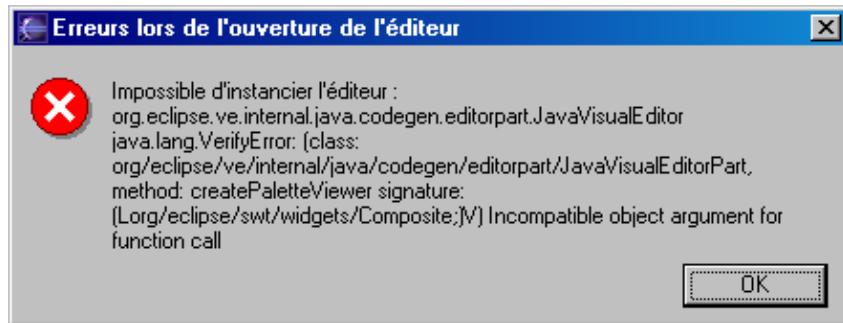
L'assistant a généré une classe Java qui s'ouvre dans un éditeur particulier, l'«Editeur Visuel». Le lancement de cet éditeur nécessite le lancement d'une machine virtuelle dédiée.



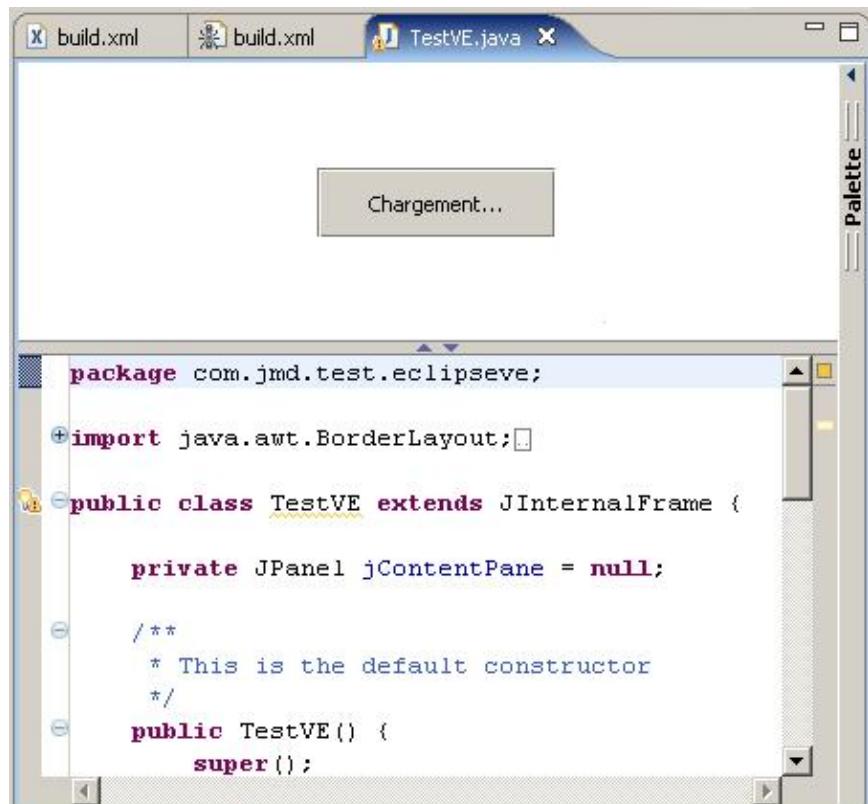
Il est possible d'édition cette classe avec l'éditeur de code Java ou grâce au plug-in VE de l'édition avec l'«Editeur Visuel».



Lors du lancement de cet éditeur visuel, si l'erreur ci dessous s'affiche, c'est que la version du plug-in GEF requise n'est pas installée avec Eclipse.



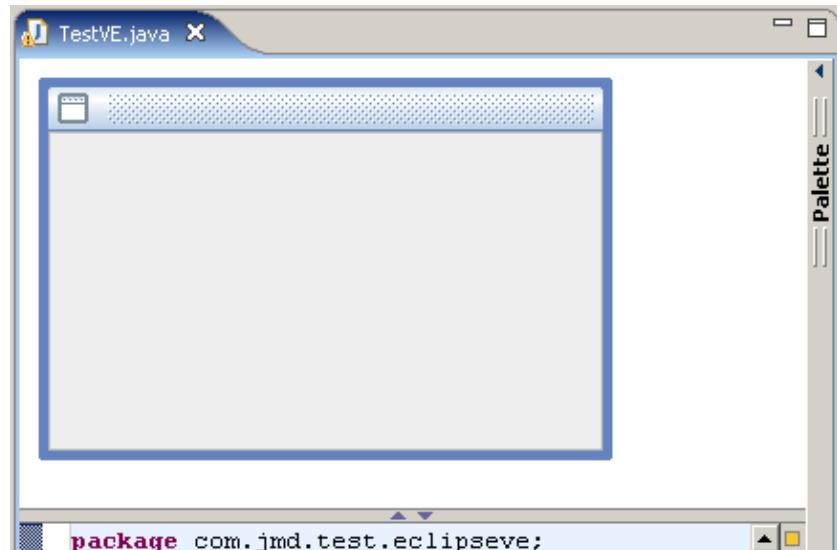
L'éditeur se charge et s'adapte en fonction du code de la classe en cours d'édition.



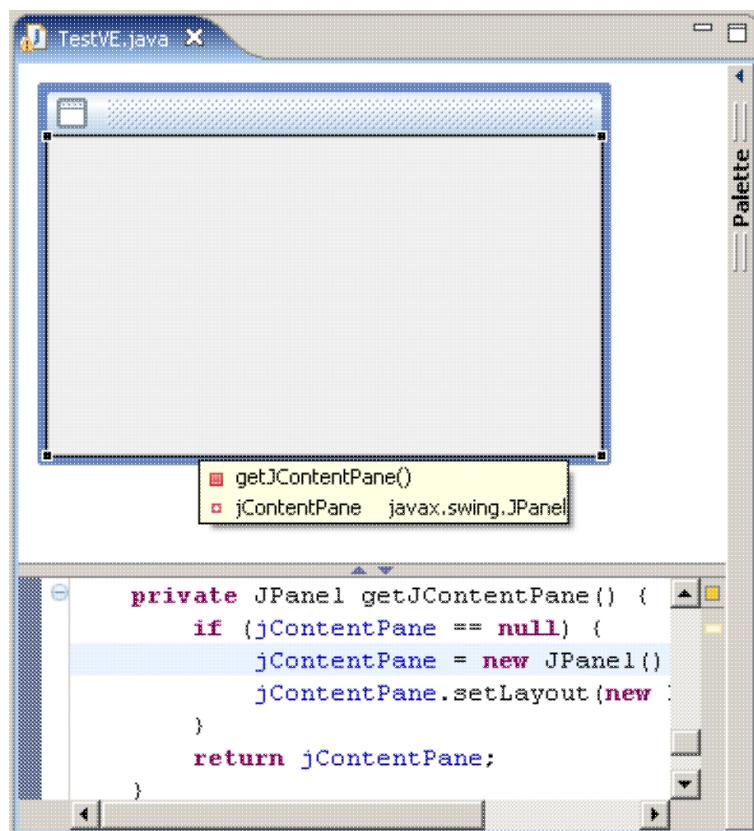
L'éditeur se compose de trois parties :

- En haut, un éditeur Wysiwyg du composant en cours de développement
- En bas, un éditeur de code qui affiche le code java correspondant
- A droite une barre de menu rétractable qui permet de sélectionner les composants à utiliser, regroupés par thèmes

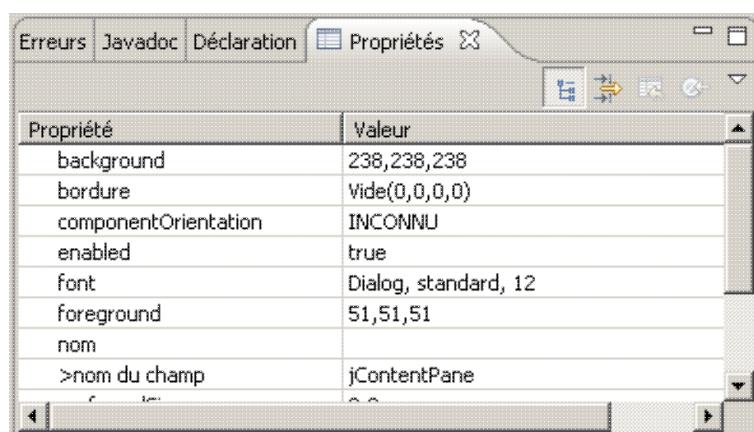
Une fois le chargement terminé, l'éditeur Wysiwyg affiche le rendu du composant en cours d'édition.



La sélection d'un composant dans l'éditeur Wysiwyg affiche le code correspondant dans l'éditeur de code.



La vue « Propriétés » affiche les propriétés du composant actuellement sélectionné dans l'éditeur.



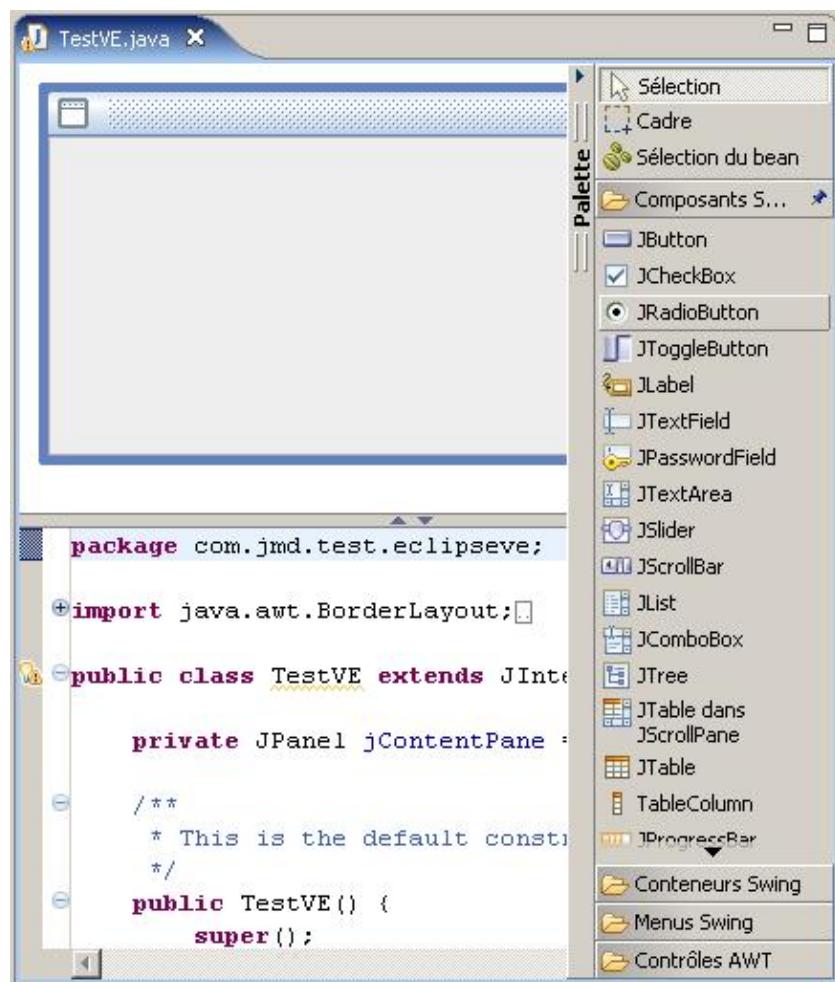
La vue « Beans Java » affiche sous la forme d'une arborescence les différents éléments qui constituent le composant en cours de développement.



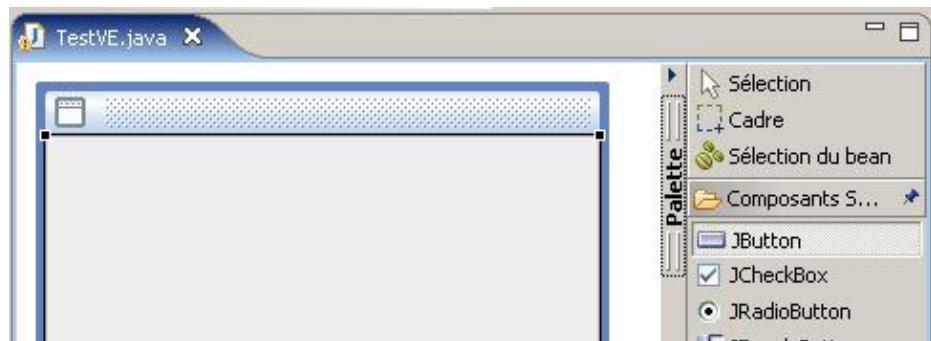
La sélection d'un élément dans l'arborescence sélectionne le composant dans l'éditeur Wysiwyg et ces propriétés sont affichées dans la vue « Propriétés ».

### 17.3.3. L'ajout d'éléments

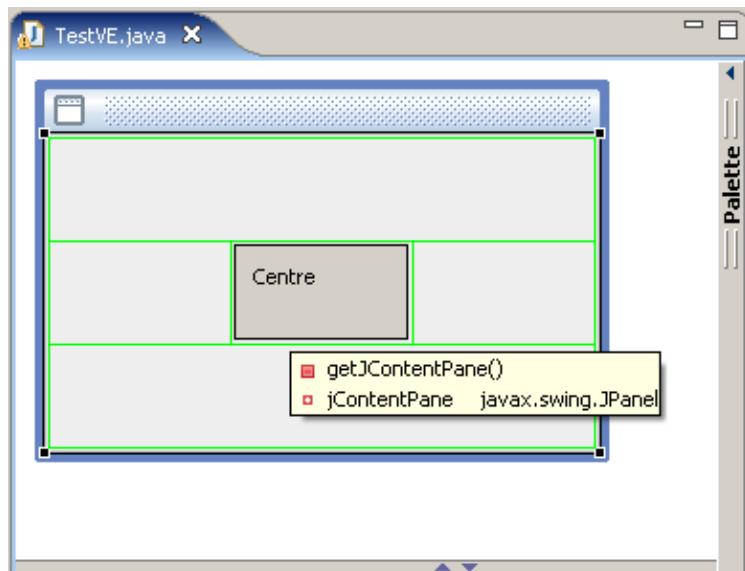
Pour ajouter un nouvel élément dans le composant en cours de développement, il faut afficher la palette en laissant le curseur de la souris sur la barre de menu rétractable.



Il suffit alors de cliquer sur le composant dans la palette



puis de cliquer à sa position sur le composant.

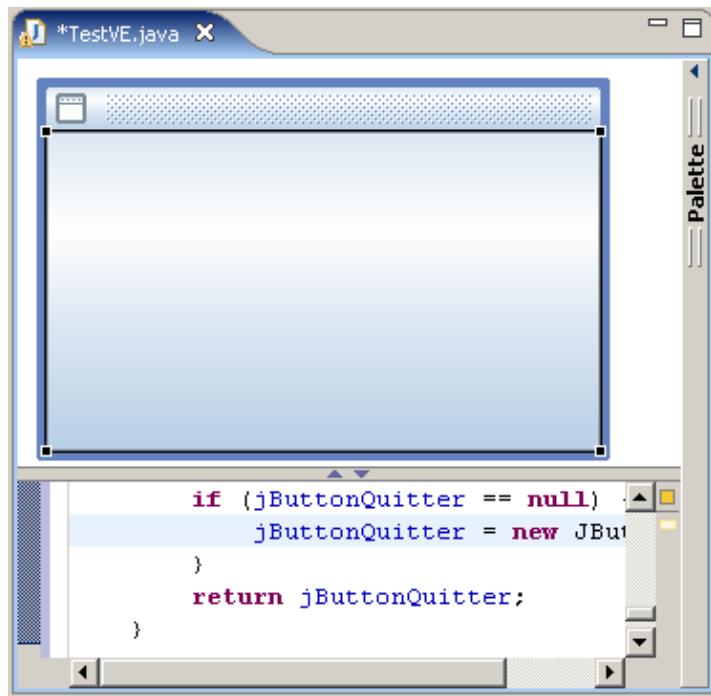


La localisation du nouvel élément est facilitée par l'affichage des zones du gestionnaire de positionnement du composant survolées par la souris (dans l'exemple ci dessus un BorderLayout).

Une fois le composant ajouté, une boîte de dialogue permet de saisir le nom du nouveau composant.

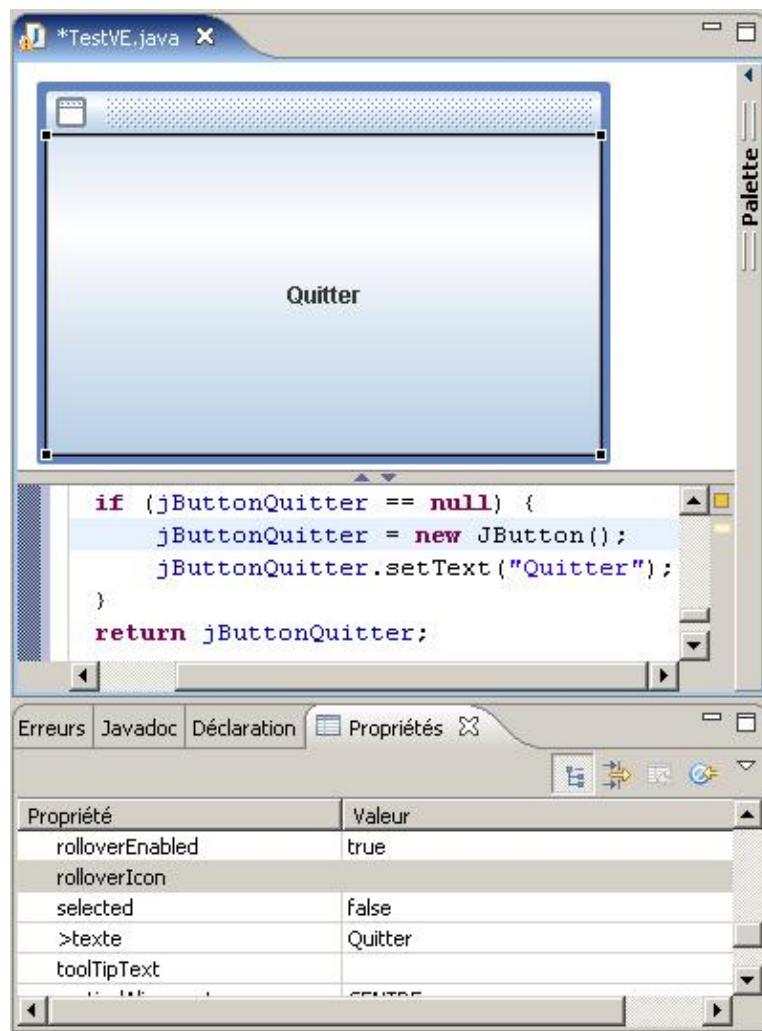


Le code correspondant au composant est ajouté et le composant s'affiche.



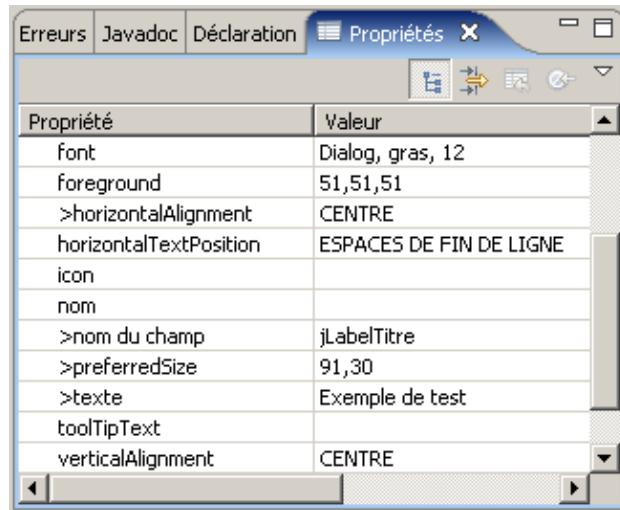
```
if (jButtonQuitter == null) {  
    jButtonQuitter = new JButton();  
}  
return jButtonQuitter;  
}
```

Les propriétés peuvent être mises à jour, par exemple :

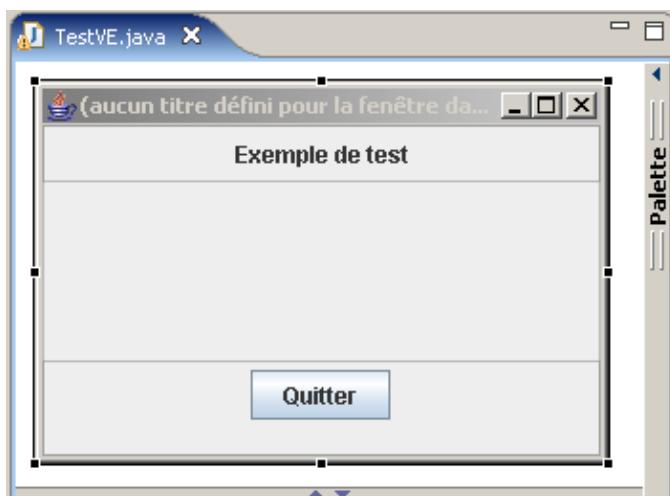


Propriété	Valeur
rolloverEnabled	true
rolloverIcon	
selected	false
>texte	Quitter
toolTipText	

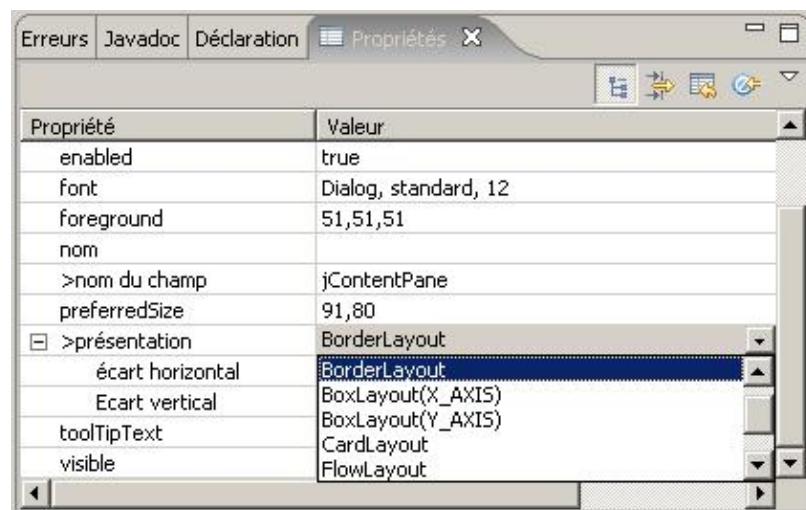
L'ajout d'un autre composant est tout aussi facile : par exemple, ajouter un composant JLabel dans la partie North du gestionnaire de positionnement. Ces propriétés peuvent aussi être modifiées en fonction des besoins :



L'éditeur Wisywig affiche le résultat :

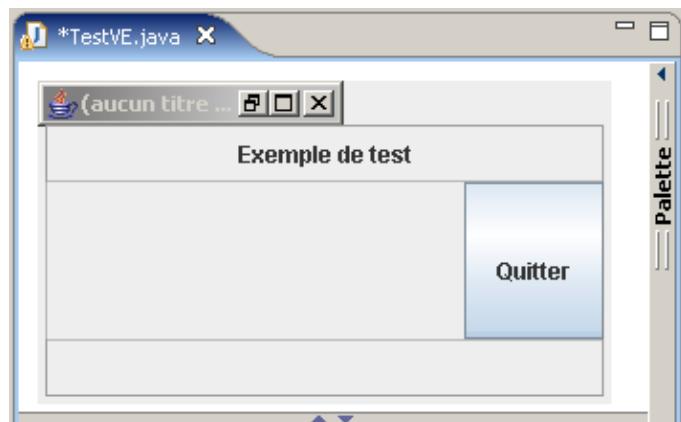
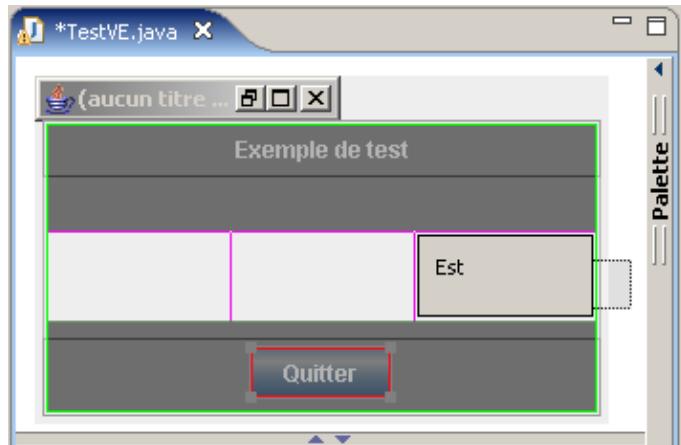


L'ajout d'un conteneur ce fait de la même façon que celle d'un composant. Dans les propriétés, il est possible de modifier le gestionnaire de positionnement défini par défaut pour ce conteneur en modifiant sa propriété layout.

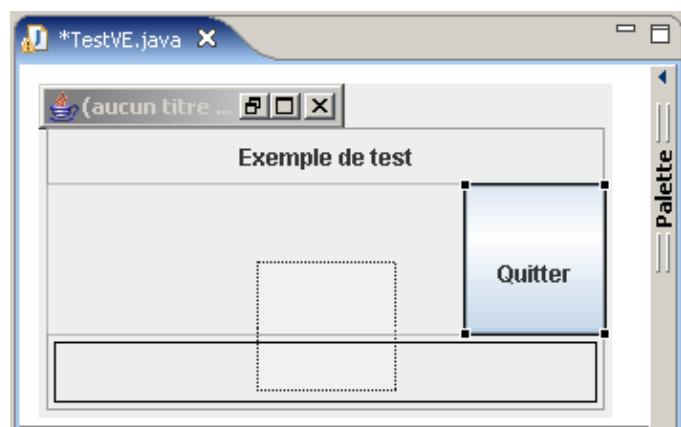


Il est possible de déplacer un composant d'un conteneur vers un autre. Il suffit de faire un cliquer/glisser du composant vers le nouveau conteneur.

Exemple : déplacement du bouton dans la partie East du gestionnaire de positionnement BorderLayout



Exemple : déplacement du bouton dans le panneau inséré dans la partie South



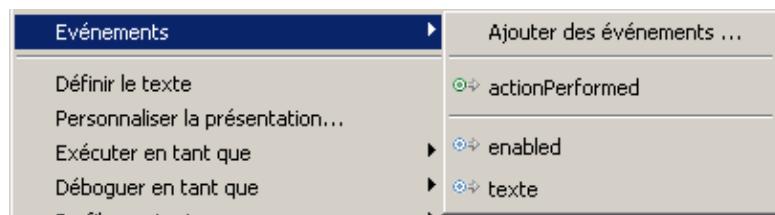
La mise en forme des composants est très intuitive et très facile d'emploi notamment grâce à un affichage particulièrement pratique des gestionnaires de positionnement de chacun des conteneurs.

### 17.3.4. La gestion des événements

Pour ajouter la gestion d'un événement particulier d'un composant, il faut utiliser l'option « Evénements » du menu contextuel de ce composant.

Par exemple, pour ajouter un événement sur le clic du bouton, il faut :

- Cliquer sur le bouton pour le sélectionner
- Utiliser l'option « Evénements » de son menu contextuel
- Et sélectionner l'option « actionPerformed »



Le morceau de code correspondant est automatiquement généré.

```
jButtonQuitter.setText("Quitter");
jButtonQuitter.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        System.out.println("actionPerformed()"); // TODO Au
    }
});
```

The screenshot shows the Java code generated by Eclipse for the JButton's actionPerformed event. It includes the button's text definition and the addition of an ActionListener that prints "actionPerformed()" to the standard output.

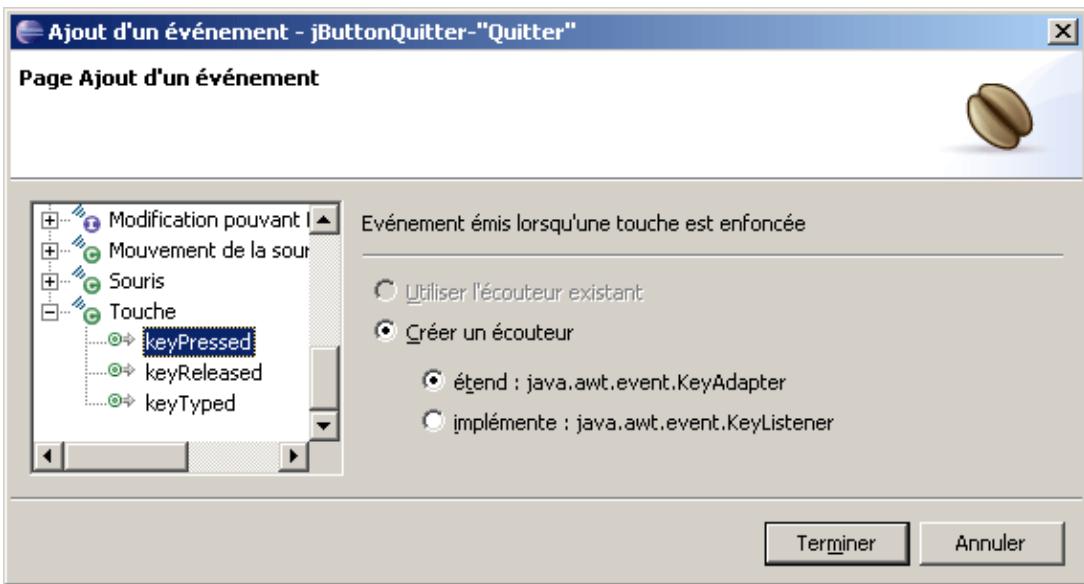
Une tâche de type « A faire » (TODO) est créée pour rappeler au développeur que cette méthode a été générée automatiquement et qu'il faut la compléter avec le code adéquat.

Il est par exemple possible de remplacer l'affichage du nom de la méthode sur la sortie standard par une demande de l'arrêt de l'application.

```
jButtonQuitter.setText("Quitter");
jButtonQuitter.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        System.exit(0);
    }
});
```

The screenshot shows the Java code after modification. The println statement has been replaced by a call to System.exit(0), which exits the application when the button is clicked.

Pour pouvoir définir un gestionnaire pour un des autres gestionnaires d'événements, il suffit d'utiliser l'option « Evénements / Ajouter des événements » du menu contextuel. Une boîte de dialogue permet alors de sélectionner l'événement désiré.

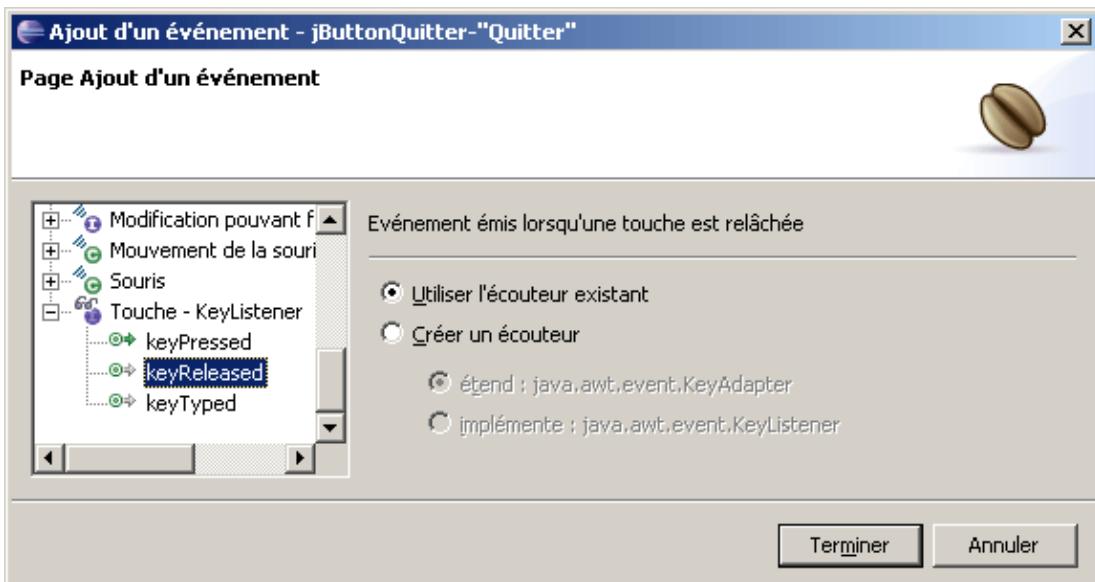


Il suffit alors de sélectionner :

- soit un type d'événement (par exemple Touche) : dans ce cas les trois méthodes correspondant aux trois événements du listener correspondant seront générées
- soit directement un événement : dans ce cas, il est possible de définir quel sera l'origine du gestionnaire d'événement (Listener ou Adapter) selon que l'on veuille ou non une définition par défaut des méthodes à implémenter.

Un fois qu'un gestionnaire est défini, une petite icône en forme de lunette apparaît en haut à gauche du gestionnaire.

Si le gestionnaire d'événement est déjà partiellement implémenté pour un événement particulier, alors lors de la sélection d'un autre événement, VE propose d'utiliser le gestionnaire existant.



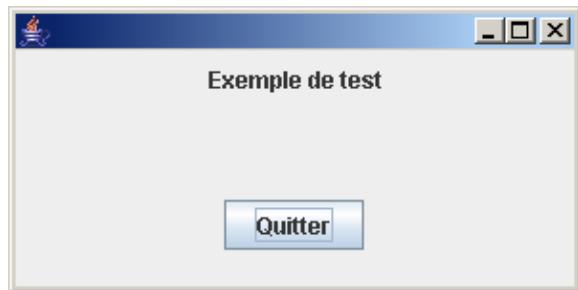
### 17.3.5. Exécution

Si l'application contient une méthode main(), il suffit de demander l'exécution de cette classe.

Dans l'exemple ci dessous, la classe ne possède pas de méthode main(). Il suffit d'écrire une classe qui va instancier un objet de type du composant créé et qui va l'afficher

### Exemple :

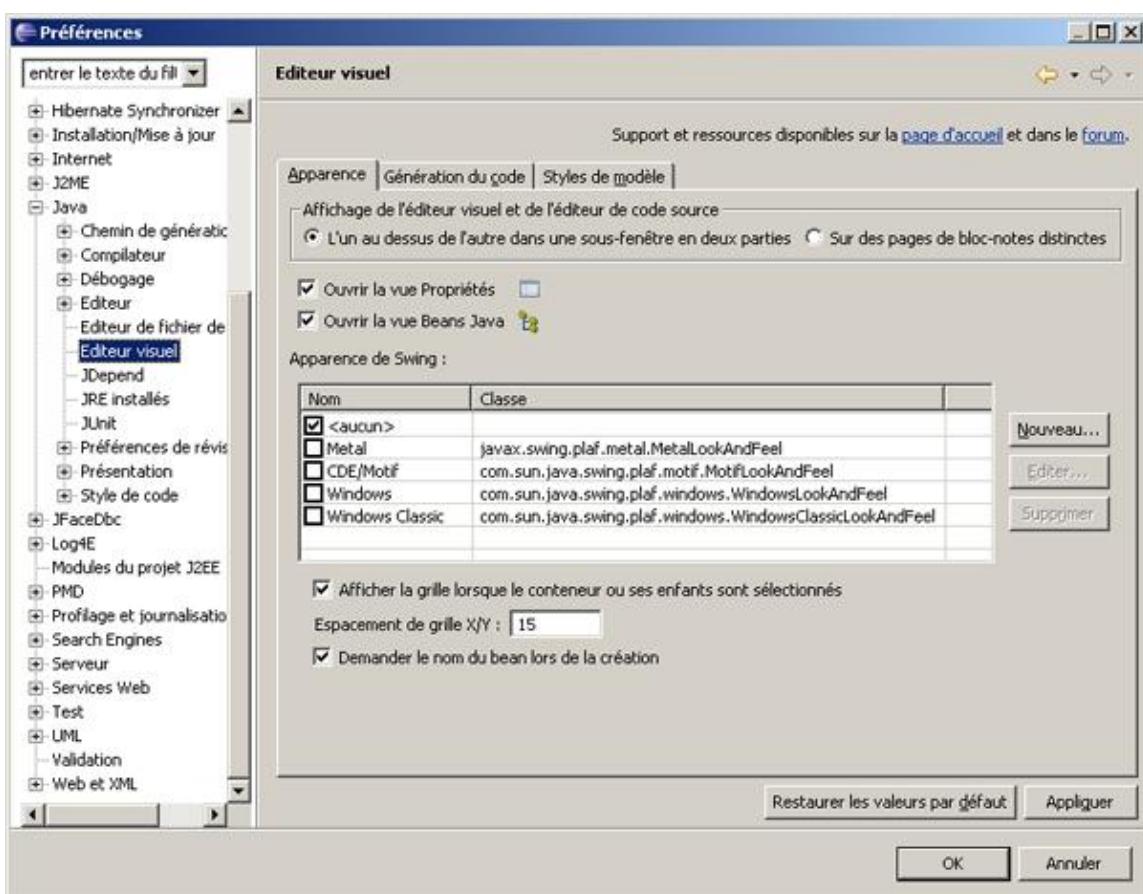
```
public class TestVeRun {  
    public static void main(String[] args) {  
        TestVE tv = new TestVE();  
        tv.show();  
    }  
}
```



Un clic sur le bouton « Quitter » ferme l'application.

Il est aussi possible de demander l'exécution sous le contrôle du débogueur comme pour une toute autre application.

Dans les Préférences, il est possible de modifier certains paramètres du plug-in en sélectionnant dans l'arborescence "Java/Editeur Visuel".



## **18. Le plug-in TPTP (Test & Performance Tools Platform)**

# **Chapitre 18**

Le projet TPTP (Test & Performance Tools Platform) a pour but de proposer un framework permettant de fournir des services de mesures de performance et d'automatisation de tests.

Historiquement, le projet TPTP était diffusé sous le nom Hyades.

Il se compose actuellement de 4 sous projets :

- TPTP Platform : il se compose du Coeur du projet TPTP et propose des services de bases communs aux autres sous projets
- Monitoring Tools : il propose des services pour collecter et analyser des informations concernant les ressources d'une application et du système
- Testing Tools : il propose des services pour faciliter les tests
- Tracing and profiling tools : il propose des services pour collecter et analyser des informations concernant les performances d'une application

La page du projet TPTP est à l'url : <http://www.eclipse.org/tptp/>

La version mise en oeuvre dans ce chapitre est la 4.1

### **18.1. Installation**

Il est possible d'installer le plug-in par l'outil de mise à jour de logiciels ou par un téléchargement et une décompression des fichiers requis. Dans ce dernier cas, il faut télécharger sur le site du projet les fichiers :

- tptp.runtime-TPTP-4.0.1-200510031151.zip : runtime du projet TPTP
- NLpack1-tptp.runtime-TPTP-4.0.1-200510031151.zip : traductions du runtime du projet TPTP
- tptpdcc.win\_ia32-TPTP-4.0.1-200510031151.zip : agent controller pour le système Windows
- rac-nl1-tptp.sdk-TPTP-4.0.1-200509281716.zip : traductions de l'agent controller

L'installation de la partie runtime du plug-in se fait comme pour tout autre plug-in en décompressant les archives.

L'agent controller est un processus dépendant du système sur lequel il s'exécute pour assurer le lien entre la JVM et le plug in TPTP.

Il faut décompresser le fichier tptpdcc.win\_ia32-TPTP-4.0.1-200510031151.zip et rac-nl1-tptp.sdk-TPTP-4.0.1-200509281716.zip dans le répertoire d'installation Eclipse (il est recommandé de réaliser cette décompression dans un répertoire dédié hors du répertoire d'installation d'Eclipse).

Il faut ensuite exécuter le fichier SetConfig.bat contenu dans le répertoire bin décompressé de l'agent controller.

Il suffit de valider chacune des réponses par un appui sur la touche Entrée sauf si la valeur proposée ne convient pas.

Il faut ajouter le chemin du répertoire bin dans la variable PATH du système



Il faut créer une nouvelle variable nommée RASERVER\_HOME ayant pour valeur le chemin d'installation de l'agent controller



## 18.2. Profiler une application

La classe ci-dessous est utilisée dans cette section.

Exemple :

```
public class TestTP {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        StringBuffer b = new StringBuffer();  
        for(int i=0; i<1000; i++) {  
            b.append("texte ");  
        }  
    }  
}
```

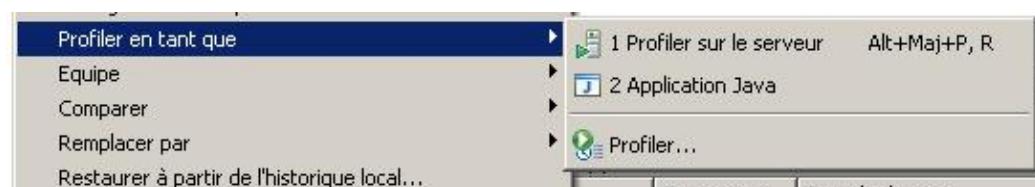
Pour utiliser le plug-in TPTP, il est nécessaire de lancer l'agent controller. Le plus simple est d'utiliser l'option « Exécuter... » du menu démarrer et de demander l'exécution de la commande raserver.



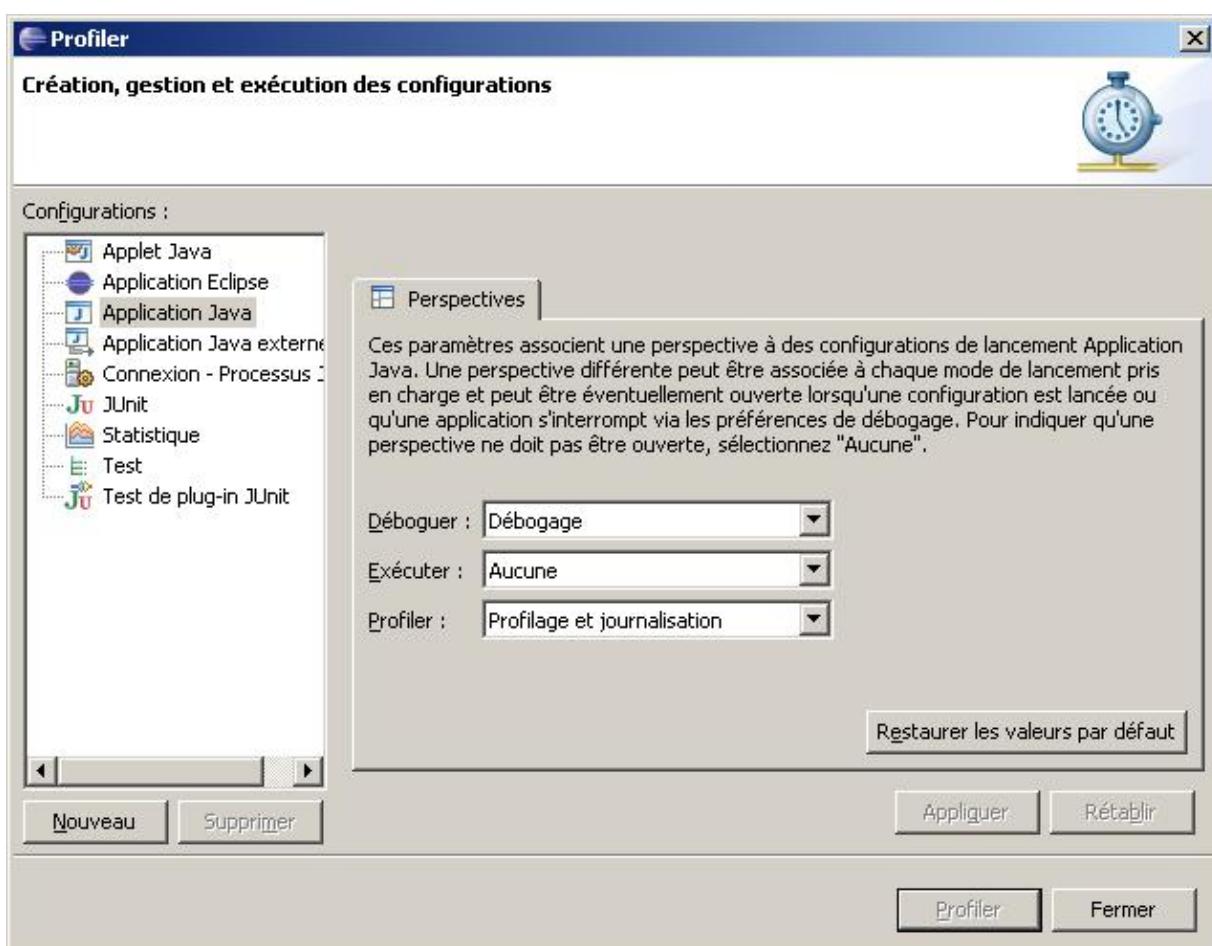
Cliquer sur le bouton « OK » pour lancer l'agent controller.



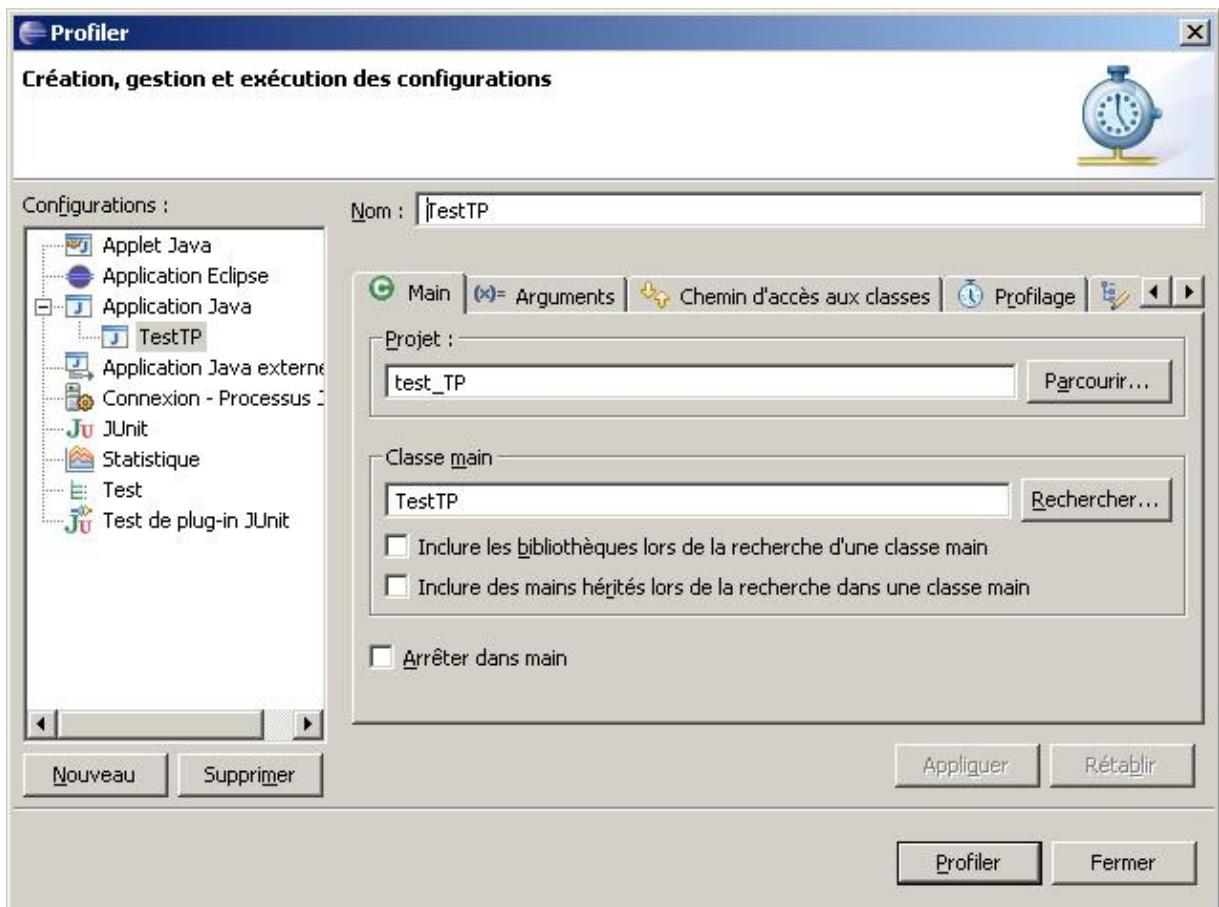
Pour créer une nouvelle configuration, sélectionner l'option « Profiler en tant que / Profiler ... » du menu contextuel de la classe dans l'explorateur de packages



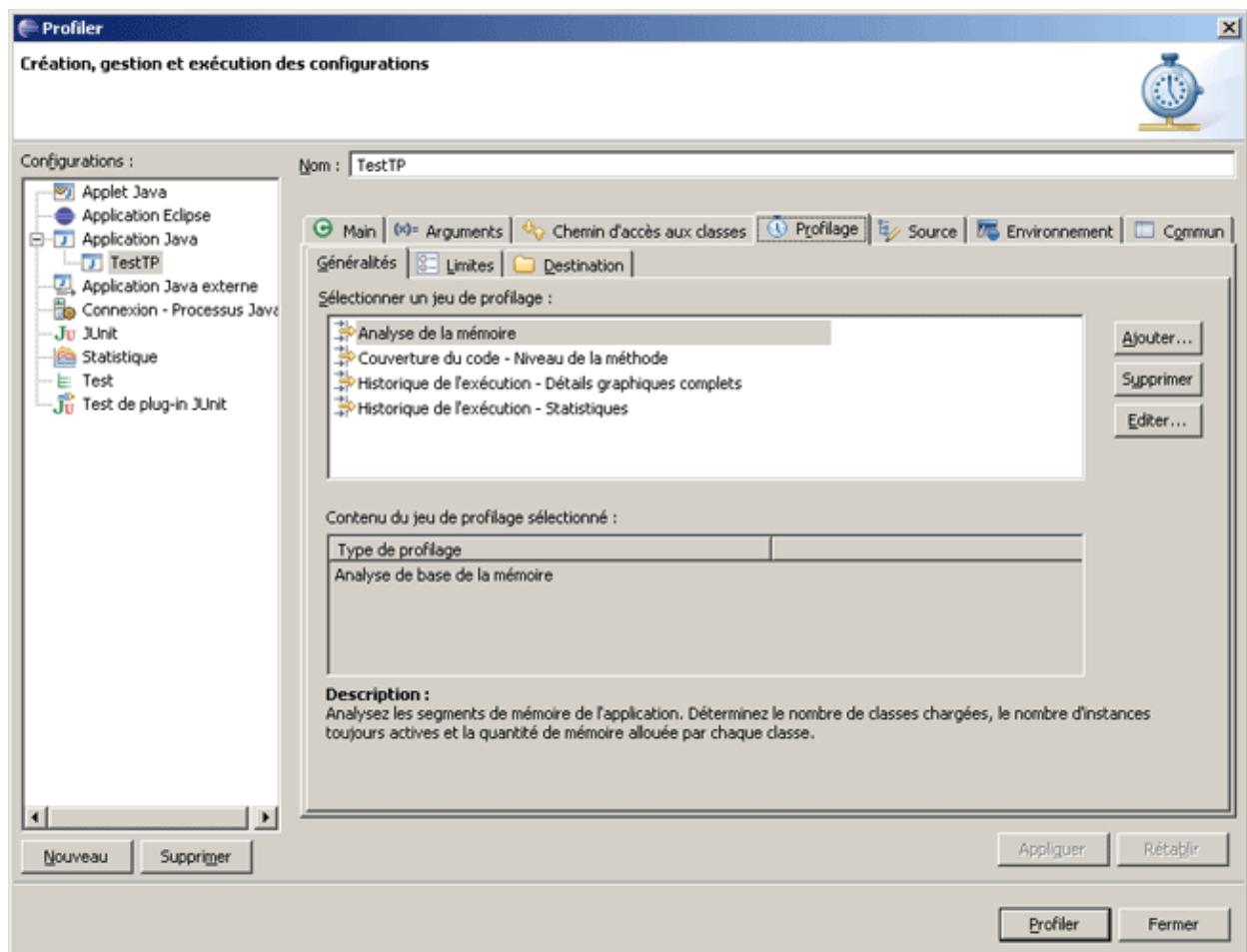
La boîte de dialogue « Profiler » s'ouvre



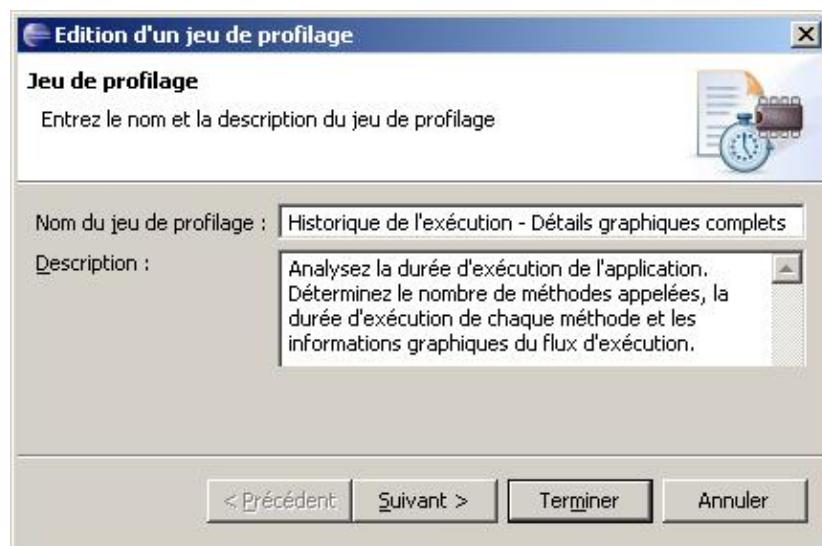
Sélectionner « Application Java » et cliquer sur le bouton « Nouveau »



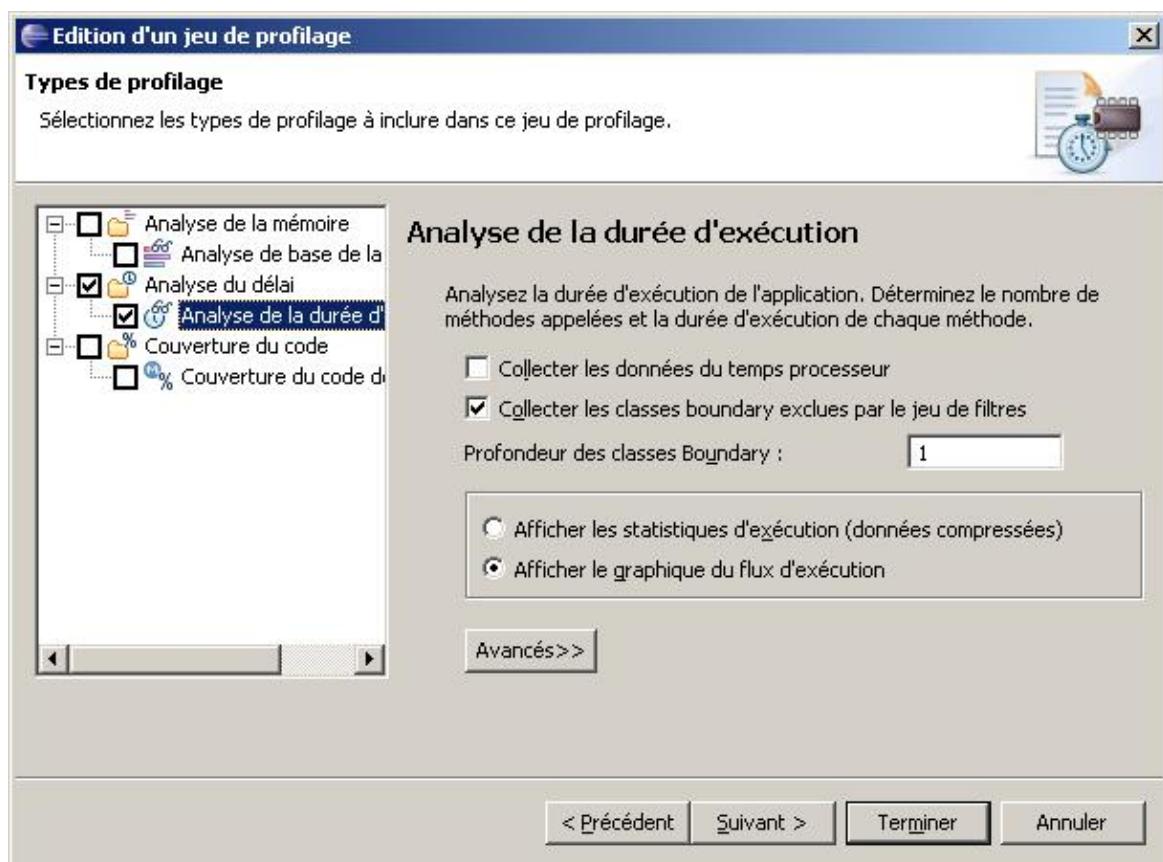
Pour définir un profil, il faut cliquer sur l'onglet « Profilage »



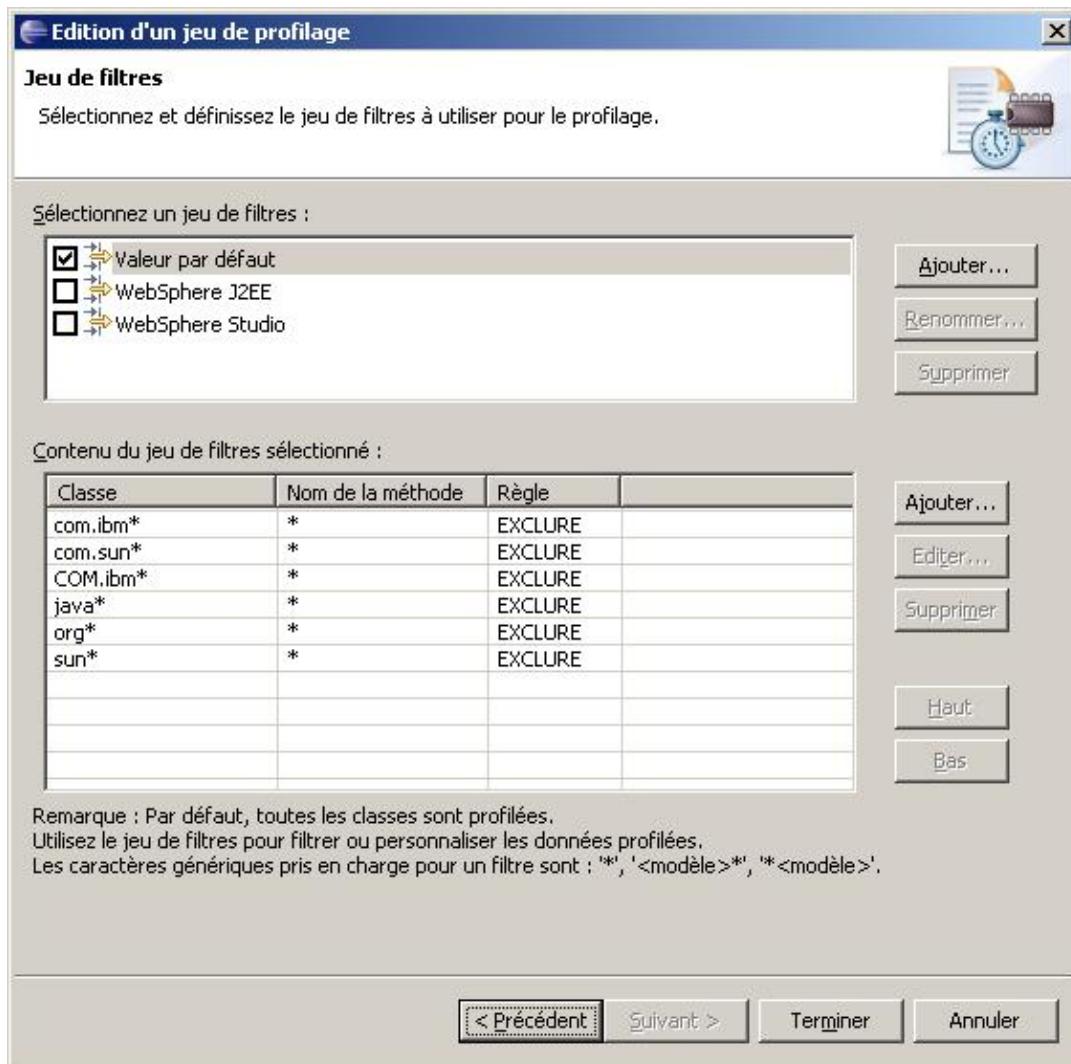
Sélectionner le jeu de profilage « Historique de l'exécution - Détails graphiques complets » et cliquer sur le bouton « Editer ... »



Cliquer sur le bouton « Suivant »

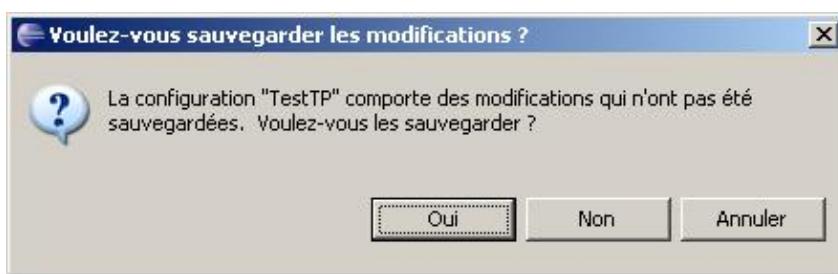


Cliquer sur le bouton « Suivant »



Cliquer sur le bouton « Terminer » pour revenir à la boîte de dialogue « Profiler ».

Si des modifications sont apportées lors d'un clic sur le bouton Fermer, une boîte de dialogue permet de demander la sauvegarde des modifications :



Cliquer sur le bouton « Profiler ».



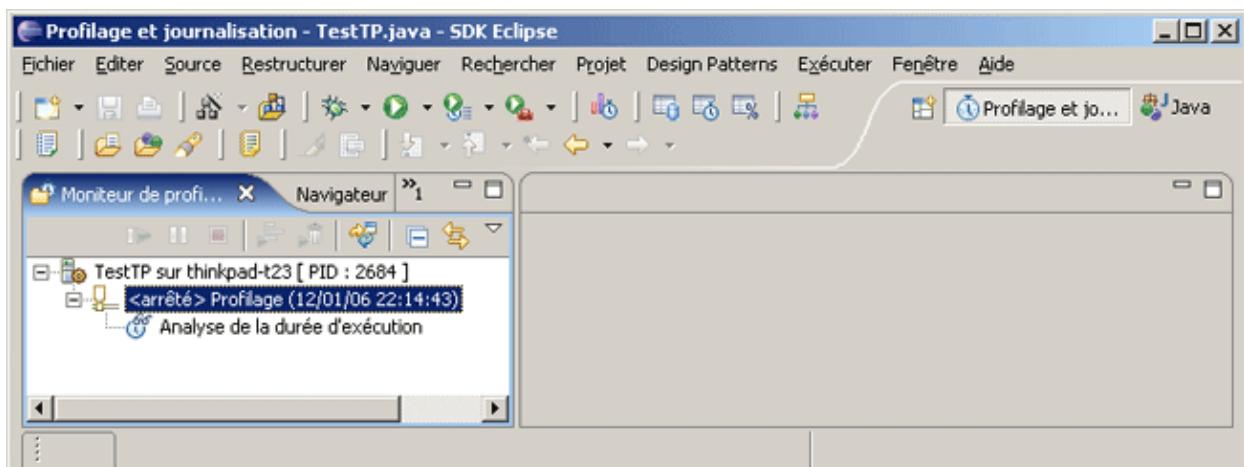
Cliquer sur « Oui »

Le message ci-dessous est affiché si la communication avec l'agent Controller n'est pas possible.

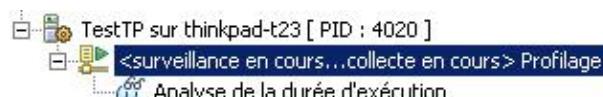


Dans ce cas, il est nécessaire de lancer l'Agent Controller.

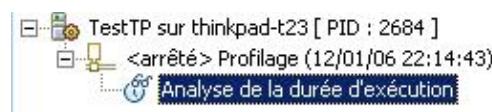
La perspective « Profilage et journalisation » s'ouvre.



Les traitements sont exécutés pour permettre la collecte des données :



Une fois que toutes les données sont collectées, la tâche « Profilage » est arrêtée :



Pour afficher les résultats, il suffit de double cliquer sur « Analyse de la durée d'exécution ». Par défaut, ce sont les informations au niveau package qui sont affichées.

Statistiques d'exécution - TestTP sur thinkpad-t23 [ PID : 2684 ] (Filtre : Aucun filtre )

>Package		Temps de base ...	Temps de base ...	Temps cumulé (...)	Appels
(package par défaut)	◆	0,086682	0,086682	0,098565	1
TestTP	◆	0,086682	0,086682	0,098565	1
main(java.lang.String[]) v	◆	0,086682	0,086682	0,098565	1
java.lang	◆	0,011883	0,000012	0,011883	1003
Class	◆	0,000000	0,000000	0,000000	0
ClassLoader	◆	0,000140	0,000070	0,000140	2
checkPackageAccess(java.lang.String)	◆	0,000026	0,000026	0,000026	1
loadClassInternal(java.lang.String)	◆	0,000113	0,000113	0,000113	1
StringBuffer	◆	0,011744	0,000012	0,011744	1001
append(java.lang.String)	◆	0,011734	0,000012	0,011734	1000
StringBuffer()	◆	0,000010	0,000010	0,000010	1

L'intérêt de ces données de permettre de détecter des points faibles du code. Par exemple, voici les résultats du code équivalent ci-dessous :

Exemple :

```
public class TestTP {
    /**
     * @param args
     */
    public static void main(String[] args) {
        String s = new String("");
        for(int i=0; i<1000; i++) {
            s = s + "texte ";
        }
    }
}
```

Statistiques d'exécution - TestTP sur thinkpad-t23 [ PID : 4020 ] (Filtre : Aucun filtre )

Package		Temps de base ...	Temps de base ...	Temps cumulé (...)	Appels
(package par défaut)	◆	0,111375	0,111375	0,218083	1
TestTP	◆	0,111375	0,111375	0,218083	1
main(java.lang.String[]) v	◆	0,111375	0,111375	0,218083	1
java.lang	◆	0,106708	0,000027	0,106708	4003
Class	◆	0,000000	0,000000	0,000000	0
String	◆	0,004190	0,000004	0,004190	1001
String(java.lang.String)	◆	0,000003	0,000003	0,000003	1
valueOf(java.lang.Object)	◆	0,004187	0,000004	0,004187	1000
ClassLoader	◆	0,000141	0,000071	0,000141	2
loadClassInternal(java.lang.String)	◆	0,000114	0,000114	0,000114	1
checkPackageAccess(java.lang.String)	◆	0,000027	0,000027	0,000027	1
StringBuffer	◆	0,102377	0,000034	0,102377	3000
StringBuffer(java.lang.String)	◆	0,054381	0,000054	0,054381	1000
append(java.lang.String)	◆	0,012878	0,000013	0,012878	1000
toString() java.lang.String	◆	0,035118	0,000035	0,035118	1000

L'utilisation de l'objet StringBuffer est ainsi justifiée : pour s'en convaincre il suffit de comparer le temps cumulé d'exécution des deux versions.

Il faut cliquer sur le bouton  pour que se soient les informations au niveau classe qui soient affichées

Statistiques d'exécution - TestTP sur thinkpad-t23 [ PID : 2684 ] (Filtre : Aucun filtre )

Classe	>Package	Temps de base ...	Temps de base ...	Temps cumulé (...)	Appels
TestTP	(package par ...)	0,086682	0,086682	0,098565	1
Class	java.lang	0,000000	0,000000	0,000000	0
ClassLoader	java.lang	0,000140	0,000070	0,000140	2
loadClassInternal(java.lang.String)	java.lang	0,000113	0,000113	0,000113	1
checkPackageAccess(java.lang.Class)	java.lang	0,000026	0,000026	0,000026	1
StringBuffer	java.lang	0,011744	0,000012	0,011744	1001
StringBuffer()	java.lang	0,000010	0,000010	0,000010	1
append(java.lang.String)	java.lang	0,011734	0,000012	0,011734	1000

Il faut cliquer sur le bouton pour que se soient les informations au niveau méthode qui soient affichées.

Statistiques d'exécution - TestTP sur thinkpad-t23 [ PID : 2684 ] (Filtre : Aucun filtre )

Méthode	Classe	>Package	Temps de base ...	Temps de base ...	Temps cumulé (...)	Appels
main(java.lang.String[])	void	TestTP	(package par ...)	0,086682	0,086682	0,098565
loadClassInternal(java.lang.String)	ClassLoader	java.lang	0,000113	0,000113	0,000113	1
checkPackageAccess(java.lang.Class)	ClassLoader	java.lang	0,000026	0,000026	0,000026	1
StringBuffer()	StringBuffer	java.lang	0,000010	0,000010	0,000010	1
append(java.lang.String)	StringBuffer	java.lang	0,011734	0,000012	0,011734	1000

Il faut cliquer sur le bouton pour que se soient les informations au niveau instance qui soient affichées.

Statistiques d'exécution - TestTP sur thinkpad-t23 [ PID : 2684 ] (Filtre : Aucun filtre )

Classe	>Package	Temps de base ...	Temps cumulé (...)	Appels
TestTP	(package par ...)	0,086682	0,098565	1
Class	java.lang	0,000000	0,000000	0
TestTP\$3045	(package par ...)	0,086682	0,098565	1
ClassLoader\$511	java.lang	0,000140	0,000140	2
StringBuffer\$1598	java.lang	0,011744	0,011744	1001
ClassLoader	java.lang	0,000140	0,000140	2
StringBuffer	java.lang	0,011744	0,011744	1001

Le bouton permet de basculer sur la perspective « Java » pour afficher la source.

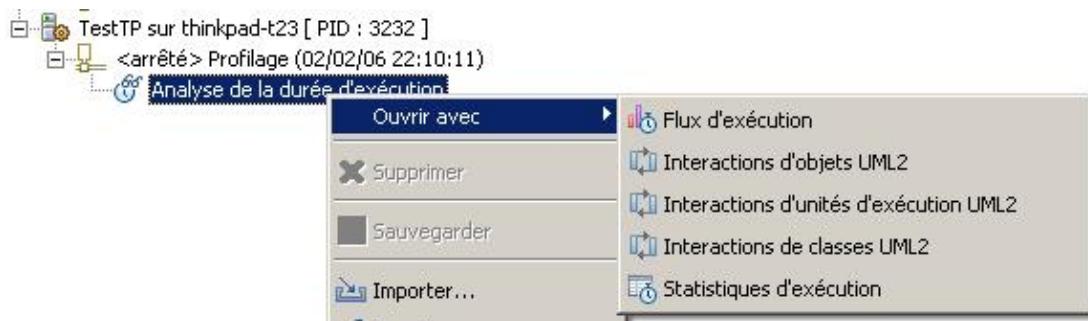
Le bouton permet d'afficher les résultats sous forme de pourcentage.

Le bouton permet de sélectionner les informations qui sont affichées.

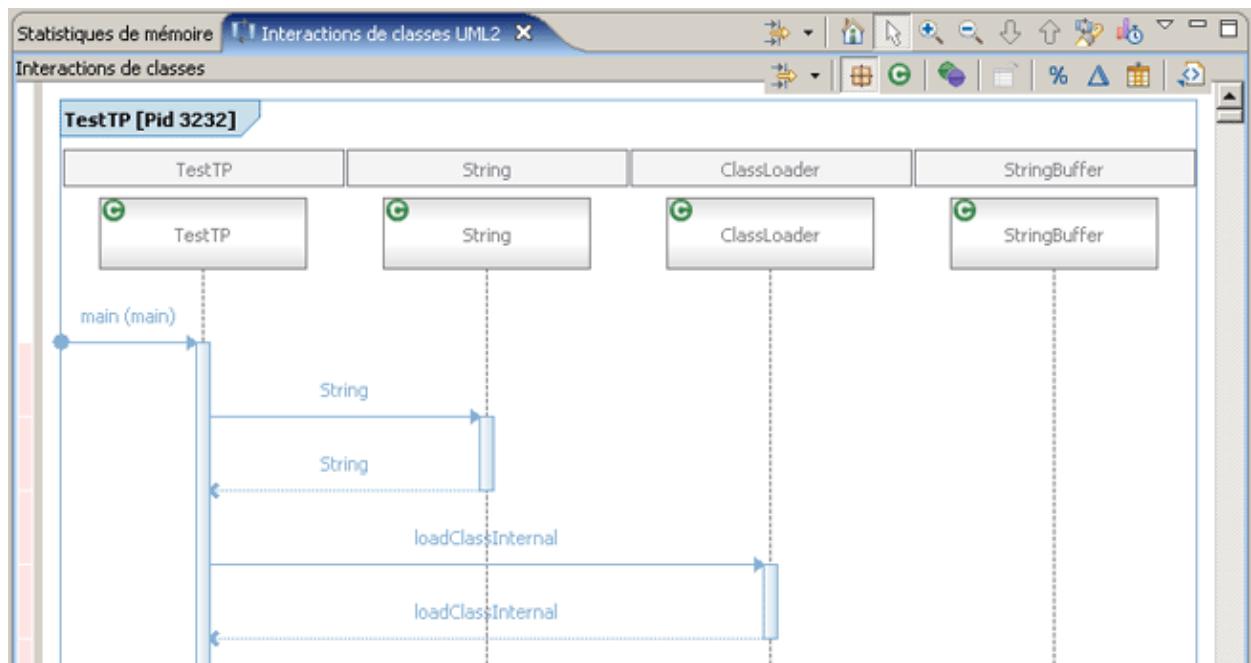


Le bouton permet d'exporter les résultats dans un fichier HTML.

Il est possible d'obtenir d'autres vues en utilisant l'option « Ouvrir avec »



L'option « Interactions de classes UML2 » affiche sous la forme d'un diagramme de séquences les échanges des traitements profilés.



En fonction de la configuration de profilage utilisée, plusieurs vues sont proposées.



## 18.3. Profiler une application Web



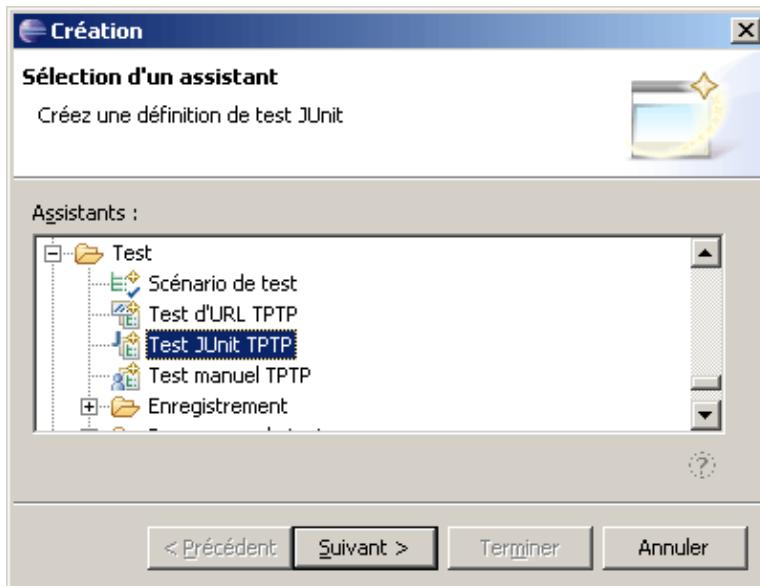
Cette section sera développée dans une version future de ce document

## 18.4. Les outils de tests

Le plug-in TPTP propose une série d'outils permettant de réaliser des tests d'une application : ces tests peuvent utiliser JUnit, concerné les performances d'une application Web ou être manuel.

### 18.4.1. Les tests avec JUnit

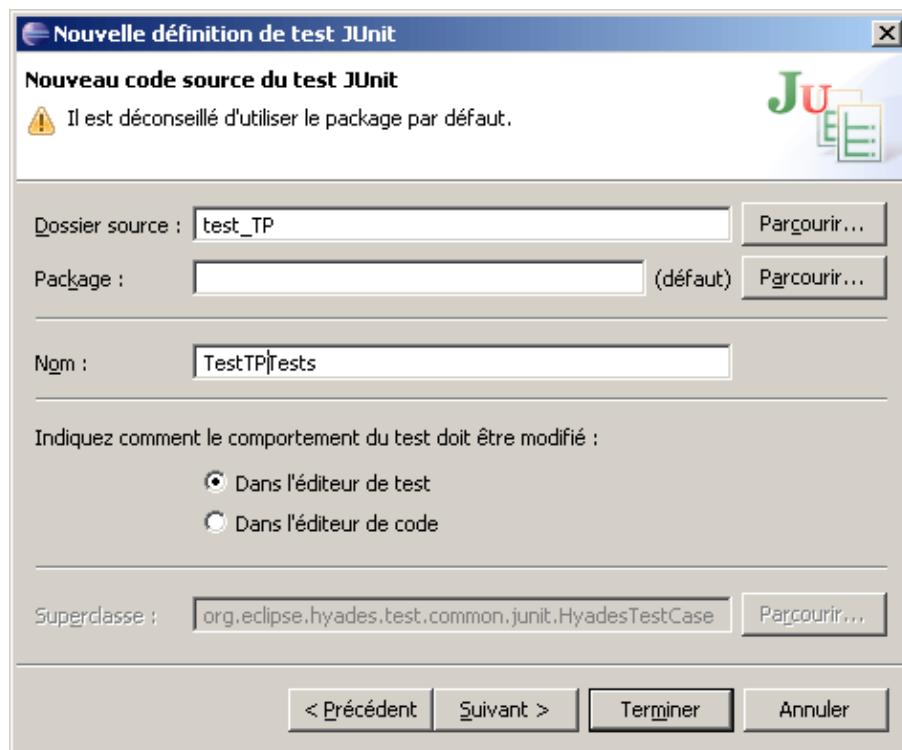
Dans l'explorateur de package, sélectionner un package et utiliser l'option « Nouveau / Autre ... » du menu contextuel



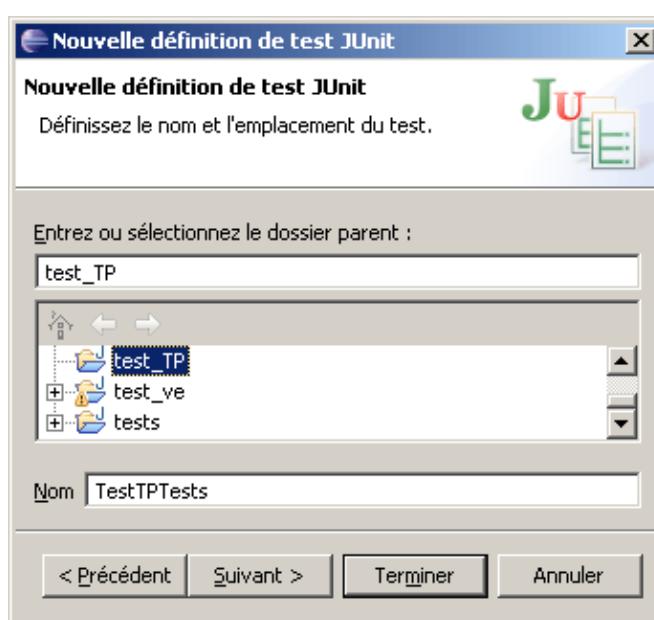
Il faut sélectionner « Test / Test JUnit TPTP » et cliquer sur le bouton « Suivant ».



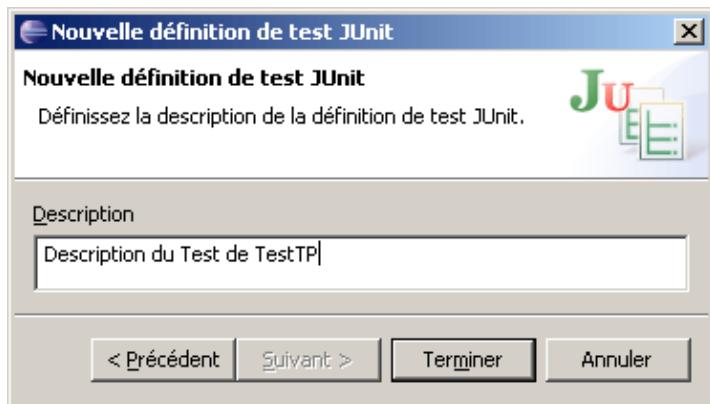
Cliquez sur le bouton « Suivant ».



Il faut saisir le nom du test et cliquer et cliquer sur le bouton « Suivant ».



Cliquez sur le bouton « suivant »

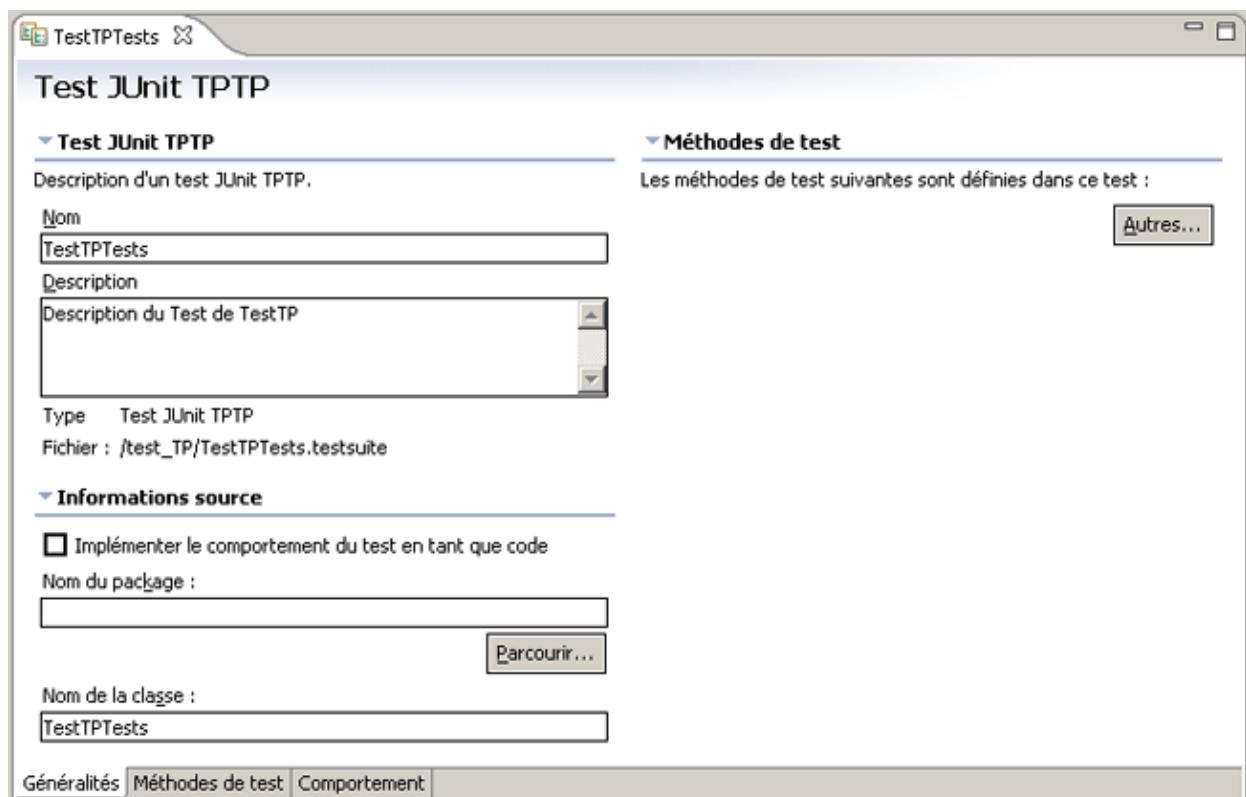


Saisissez la description du test et cliquez sur le bouton « Terminer ».

Une entité avec l'extension .testsuite est créée avec le nom fourni dans l'assistant.



La vue « Test JUnit TPTP » s'ouvre.



L'onglet « Généralités » affiche des informations générales sur le test.



Cette section sera développée dans une version future de ce document

## 19. Hibernate et Eclipse

# Chapitre 19

Hibernate est un projet open source visant à proposer un outil de mapping entre les objets et des données stockées dans une base de données relationnelle. Ce projet ne repose sur aucun standard mais il est très populaire notamment à cause de ses bonnes performances et de son ouverture avec de nombreuses bases de données.

Le site officiel <http://www.hibernate.org> contient beaucoup d'informations sur l'outil et propose de le télécharger ainsi que sa documentation.

### 19.1. Le plug-in Hibernate Synchronizer



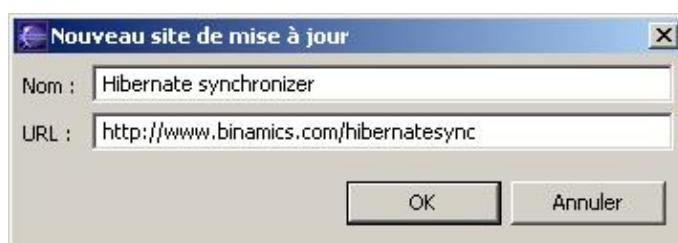
Le plug-in Hibernate Synchronizer permet la génération de code utilisant le framework Hibernate. Il permet aussi de re-générer ce code lorsqu'un fichier de mapping est modifié.

Le site du plug-in est à l'url : <http://hibernatesynch.sourceforge.net/>

	Version utilisée dans cette section
Eclipse	3.0.1
J2SE	1.4.2_03
Hibernate Synchronizer	2.3.1.

#### 19.1.1. Installation

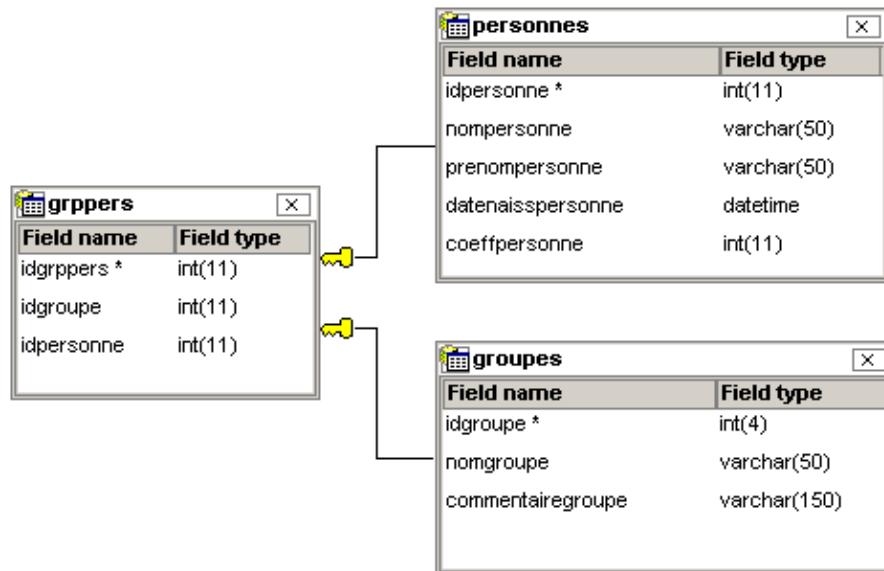
Le plus simple est d'utiliser la fonctionnalité de mise à jour proposée par l'option "Rechercher et installer" du menu "Aide". Cliquez sur le bouton "Rechercher les nouveaux dispositifs à installer" sur le bouton "Nouveau site distant", saisissez les informations ci dessous et suivez les instructions pour réaliser le téléchargement et l'installation.



## 19.1.2. La base de données utilisée

La base de données utilisées dans cette section contient trois tables :

- une table qui contient une liste de personnes
- une table qui contient une liste de groupes
- une table qui associe une personne à un groupe



### Exemple : le DDL de la base de données

```
drop table `grppers`;
drop table `groupes`;
drop table `personnes`;

#
# Structure for the `groupes` table :
#
CREATE TABLE `groupes` (
  `idgroupe` int(4) NOT NULL auto_increment,
  `nomgroupe` varchar(50) default NULL,
  `commentaireregroupe` varchar(150) default NULL,
  PRIMARY KEY  (`idgroupe`),
  UNIQUE KEY `idgroupe` (`idgroupe`)
) TYPE=InnoDB;

#
# Structure for the `personnes` table :
#
CREATE TABLE `personnes` (
  `idpersonne` int(11) NOT NULL auto_increment,
  `nompersonne` varchar(50) default NULL,
  `prenompersonne` varchar(50) default NULL,
  `datenaisspersonne` datetime default NULL,
  `coeffpersonne` int(11) default NULL,
  PRIMARY KEY  (`idpersonne`),
  UNIQUE KEY `idpersonne` (`idpersonne`)
) TYPE=InnoDB;

#
# Structure for the `grppers` table :
#
CREATE TABLE `grppers` (
  `idgrppers` int(11) NOT NULL auto_increment,
```

```

`idgroupe` int(11) default NULL,
`idpersonne` int(11) default NULL,
PRIMARY KEY (`idgrppers`),
UNIQUE KEY `idgrppers`(`idgrppers`),
KEY `idgroupe`(`idgroupe`),
KEY `idpersonne`(`idpersonne`),
CONSTRAINT `O_48` FOREIGN KEY (`idpersonne`) REFERENCES `personnes`(`idpersonne`),
CONSTRAINT `O_45` FOREIGN KEY (`idgroupe`) REFERENCES `groupes`(`idgroupe`)
) TYPE=InnoDB;

INSERT INTO `groupes`(`idgroupe`, `nomgroupe`, `commentairegroupe`) VALUES
(1,'groupe 1',NULL),
(2,'groupe 2',NULL);

INSERT INTO `grppers`(`idgrppers`, `idgroupe`, `idpersonne`) VALUES
(1,1,1),
(2,2,2),
(3,2,3),
(4,1,4),
(5,1,5);

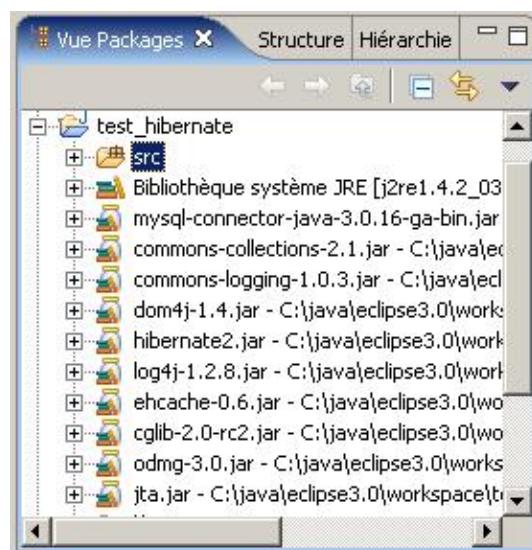
INSERT INTO `personnes`(`idpersonne`, `nompersonne`, `prenompersonne`,
`datenaisspersonne`, `coeffpersonne`) VALUES
(1,'nom1','prenom1','1967-01-06',123),
(2,'nom2','prenom2','1973-08-11',34),
(3,'nom3','prenom3','1956-04-28',145),
(4,'nom4','prenom4','1980-12-02',23),
(5,'nom5','prenom5','1966-10-13',119);

```

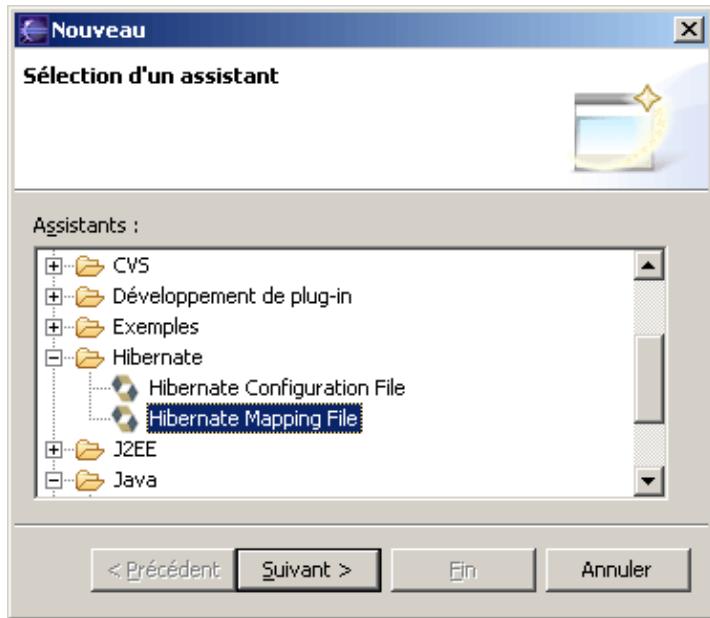
### 19.1.3. Création des fichiers de mapping

Créer un nouveau projet dont les sources et les binaires sont séparés et ajouter les fichiers mm.mysql-2.0.14-bin.jar et hibernate2.jar dans les bibliothèques.

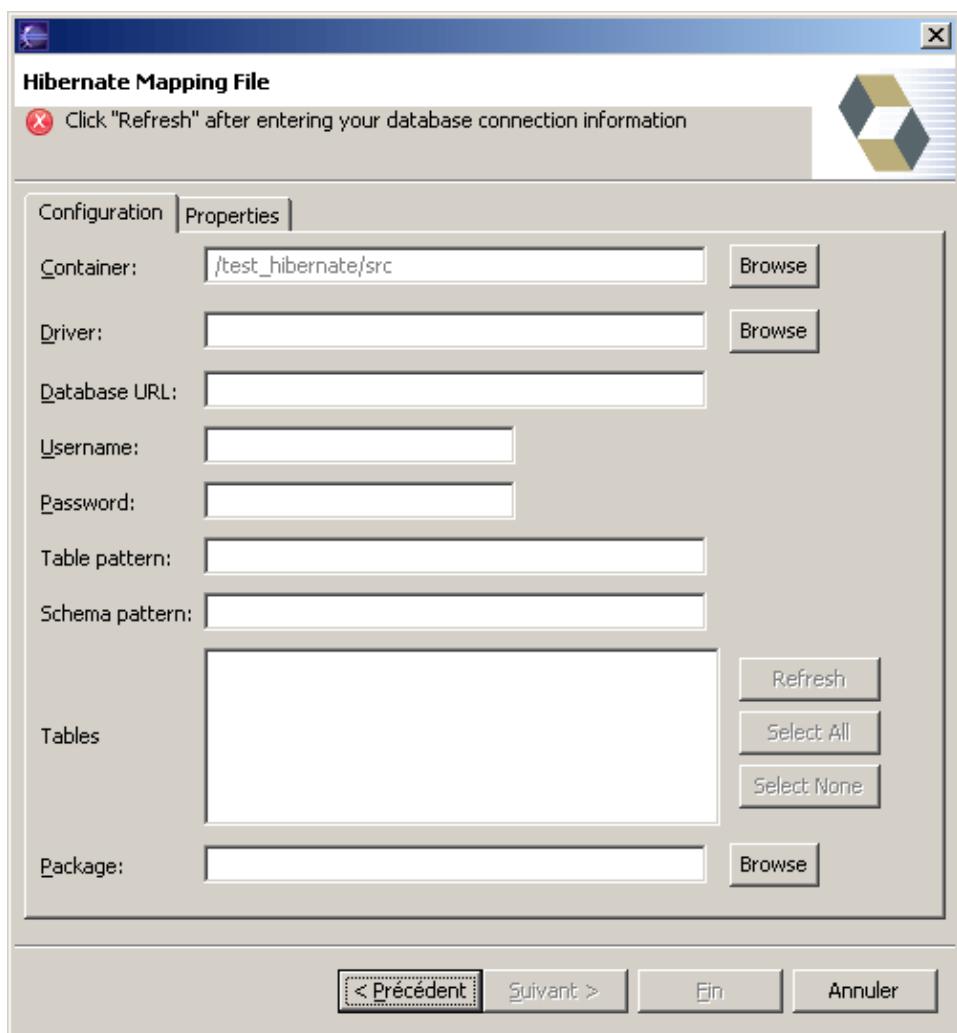
Il est aussi obligatoire d'ajouter toutes les bibliothèques nécessaires à Hibernate lors de son exécution.



Il faut ensuite créer une entité de type « Hibernate / Hibernate Mapping File » dans le répertoire src du projet.



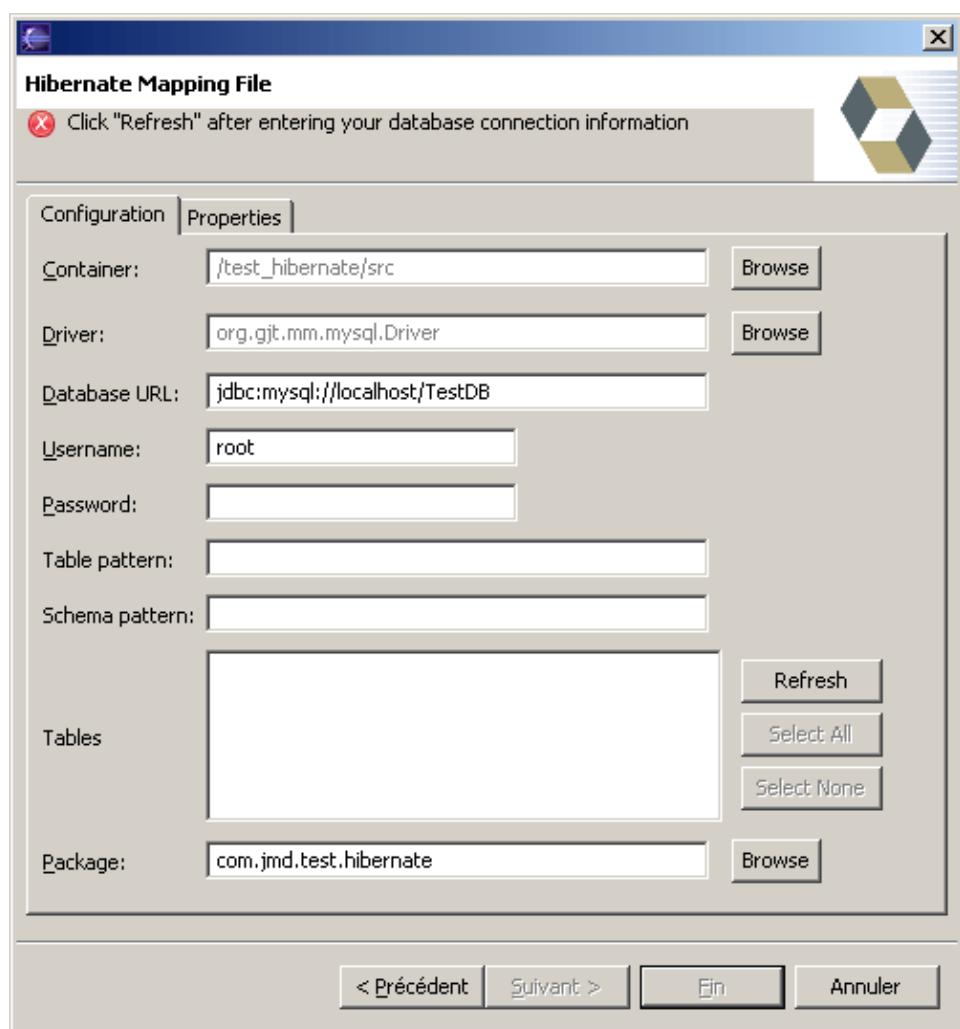
Cliquez sur le bouton « Suivant ». La page suivante permet de saisir les informations sur la base de données et sa connexion.



Cliquez sur le bouton « Browse » en face de « Driver ».



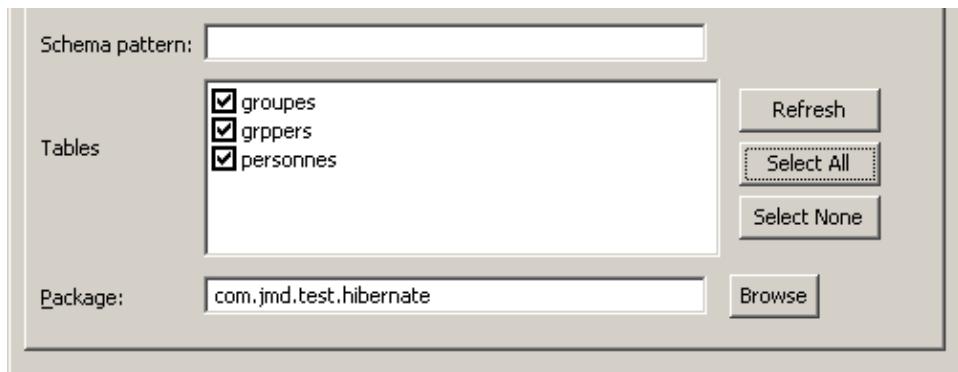
Sélectionnez la classe org.gjt.mm.mysql.Driver et cliquez sur le bouton « OK ».



Il faut ensuite saisir les informations concernant la connexion à la base de données et cliquer sur le bouton « Refresh ».

Si les paramètres concernant la connexion sont corrects alors la liste des tables est affichée. Il suffit alors de sélectionner la ou les tables désirées.

Enfin, il faut saisir le nom du package qui va contenir les fichiers générés.



Cliquez sur le bouton « Fin ». Trois fichiers sont générés : Groupes.hbm, Grppers.hbm et Personnes.hbm

#### Exemple : le fichier Personnes.hbm

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >
<hibernate-mapping package="com.jmd.test.hibernate">
    <class name="Personnes" table="personnes">

        <id
            column="idpersonne"
            name="Idpersonne"
            type="integer"
        >
            <generator class="vm" />
        </id>

        <property
            column="datenaisspersonne"
            length="19"
            name="Datenaissspersonne"
            not-null="false"
            type="timestamp"
        />

        <property
            column="prenompersonne"
            length="50"
            name="Prenompersonne"
            not-null="false"
            type="string"
        />

        <property
            column="coeffpersonne"
            length="11"
            name="Coeffpersonne"
            not-null="false"
            type="integer"
        />

        <property
            column="nompersonne"
            length="50"
            name="Nompersonne"
            not-null="false"
            type="string"
        />
    </class>
</hibernate-mapping>
```

### Exemple : le fichier Groupes.hbm

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >
<hibernate-mapping package="com.jmd.test.hibernate">
<class name="Groupes" table="groupes">
  <id
    column="idgroupe"
    name="Idgroupe"
    type="integer"
  >

    <generator class="vm" />
  </id>

  <property
    column="nomgroupe"
    length="50"
    name="Nomgroupe"
    not-null="false"
    type="string"
  />

  <property
    column="commentairegroupe"
    length="150"
    name="Commentairegroupe"
    not-null="false"
    type="string"
  />
</class>
</hibernate-mapping>
```

### Exemple : le fichier Grppers.hbm

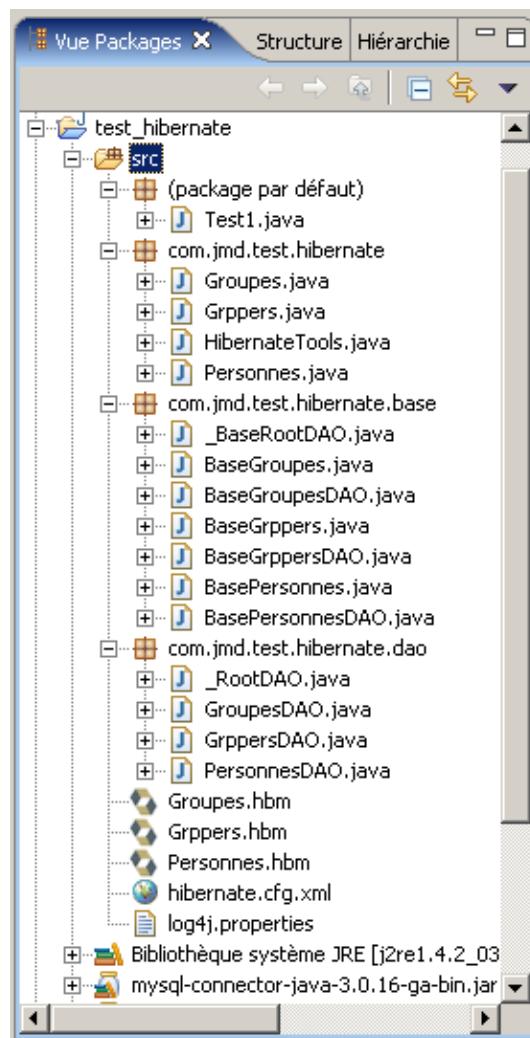
```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >
<hibernate-mapping package="com.jmd.test.hibernate">
<class name="Grppers" table="grppers">
  <id
    column="idgrppers"
    name="Idgrppers"
    type="integer"
  >
    <generator class="vm" />
  </id>
  <many-to-one
    class="Groupes"
    name="Idgroupe"
    not-null="true"
  >
    <column name="idgroupe" />
  </many-to-one>
  <many-to-one
    class="Personnes"
    name="Idpersonne"
    not-null="true"
  >
    <column name="idpersonne" />
  </many-to-one>
</class>
</hibernate-mapping>
```

#### 19.1.4. Génération des classes Java

La génération des classes correspondantes pour chaque fichier .hbm peut être demandée de deux façons :

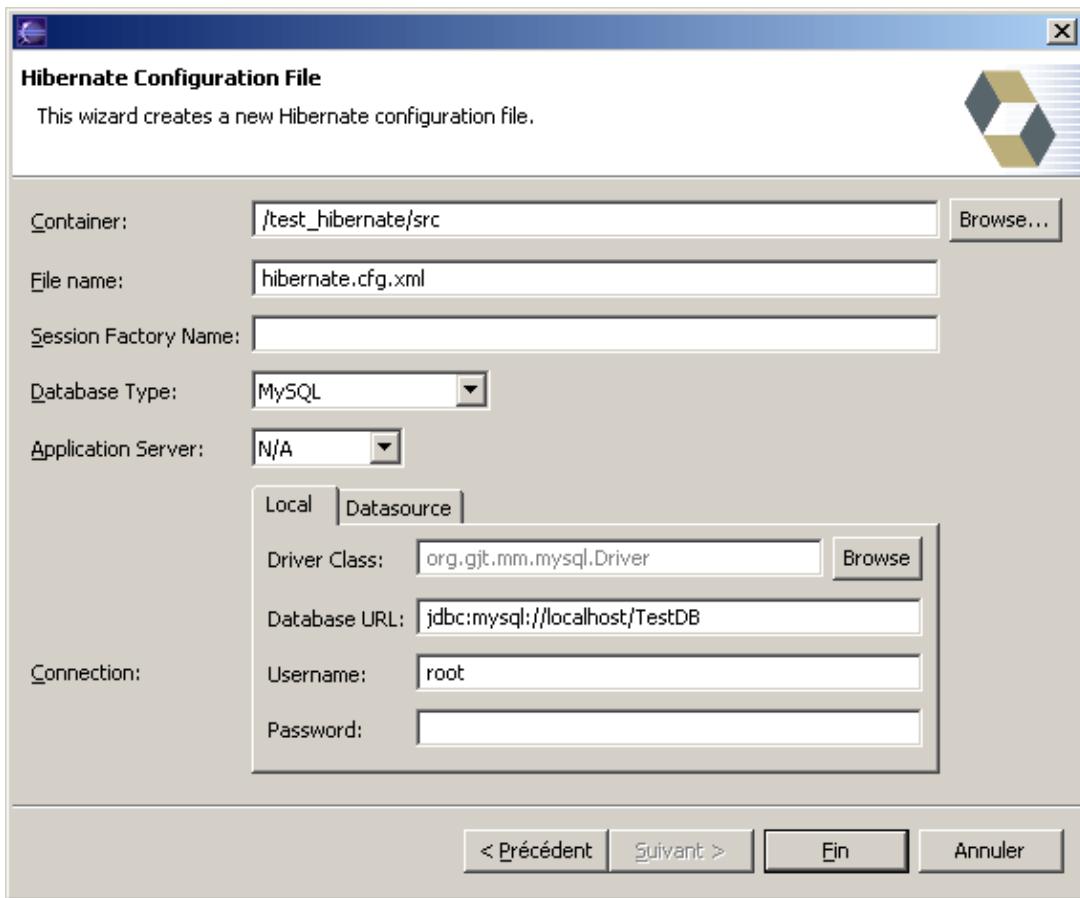
- en sauvegardant un fichier .hbm modifié dans l'éditeur
- en utilisant l'option « Hibernate Synchronizer / Synchronize Files » du menu contextuel d'un fichier .hbm dans la vue « Vue packages »

Cette génération va créer pour chaque fichier .hbm plusieurs classes.



#### 19.1.5. Création du fichier de configuration

Il faut créer une nouvelle entité de type « Hibernate / Hibernate Configuration File » dans le répertoire src. Une boîte de dialogue permet de saisir les informations qui seront insérées dans le fichier de configuration d'Hibernate.



Une fois les informations saisies, cliquez sur le bouton « Fin »

Il faut rajouter dans ce fichier tous les fichiers de mapping qui seront utilisés.

#### Exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration
PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">
<hibernate-configuration>
<session-factory >
  <!-- local connection properties -->
  <property name="hibernate.connection.url">jdbc:mysql://localhost/TestDB</property>
  <property name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver
  </property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password"></property>
  <!-- property name="hibernate.connection.pool_size"></property -->

  <!-- dialect for MySQL -->
  <property name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.show_sql">false</property>
  <property name="hibernate.use_outer_join">true</property>

  <mapping resource="Personnes.hbm" />
  <mapping resource="Grppers.hbm" />
  <mapping resource="Groupes.hbm" />
</session-factory>
</hibernate-configuration>
```

## 19.1.6. La mise en oeuvre des classes générées

Les différentes classes qui vont mettre en oeuvre les classes générées vont utiliser une classe utilitaire proposée dans la documentation de Hibernate pour configurer et obtenir une session.

Exemple :

```
package com.jmd.test.hibernate;

import net.sf.hibernate.*;
import net.sf.hibernate.cfg.*;

public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new Configuration().configure()
                .buildSessionFactory();
        } catch (HibernateException ex) {
            throw new RuntimeException("Exception building SessionFactory: "
                + ex.getMessage(), ex);
        }
    }

    public static final ThreadLocal session = new ThreadLocal();

    public static Session currentSession() throws HibernateException {
        Session s = (Session) session.get();
        // Open a new Session, if this Thread has none yet
        if (s == null) {
            s = sessionFactory.openSession();
            session.set(s);
        }
        return s;
    }

    public static void closeSession() throws HibernateException {
        Session s = (Session) session.get();
        session.set(null);
        if (s != null)
            s.close();
    }
}
```

Le premier exemple va simplement afficher le nom de toutes les personnes.

Exemple :

```
import java.util.*;
import net.sf.hibernate.*;
import com.jmd.test.hibernate.*;

public class Test1 {

    public static void main(String[] args) {
        try {
            Session session = HibernateUtil.currentSession();

            List list = session.find("from Personnes ");
            Iterator it = list.iterator();

            while(it.hasNext())
            {
```

```
        Personnes personne = (Personnes)it.next();
        System.out.println(personne.getNompersonne());
    }

    HibernateUtil.closeSession();
} catch (HibernateException e) {
    e.printStackTrace();
}
}
```

Exemple :

Résultat :  
nom1  
nom2  
nom3  
nom4  
nom5

Voici le même exemple utilisant les classes générées mettant en oeuvre le motif de conception DAO.

**Exemple :**

```
import java.util.Iterator;
import java.util.List;
import net.sf.hibernate.HibernateException;
import com.jmd.test.hibernate.Personnes;
import com.jmd.test.hibernate.dao.PersonnesDAO;
import com.jmd.test.hibernate.dao._RootDAO;

public class Test1DAO {
    public static void main(String[] args) {
        try {
            _RootDAO.initialize();
            PersonnesDAO dao = new PersonnesDAO();

            List liste = dao.findAll();
            Iterator it = liste.iterator();

            while (it.hasNext()) {
                Personnes personne = (Personnes) it.next();
                System.out.println(personne.getNompersonne());
            }
        } catch (HibernateException e) {
            e.printStackTrace();
        }
    }
}
```

Le second exemple va retrouver un groupe, créer une nouvelle personne et l'ajouter au groupe trouvé.

Exemple :

```
import java.util.*;
import net.sf.hibernate.*;
import com.jmd.test.hibernate.*;

public class Test2 {
    public static void main(String[] args) {
        try {
            Session session = HibernateUtil.currentSession();
            Transaction tx = session.beginTransaction();
```

```

Personnes personne = new Personnes();
personne.setNompersonne("nom6");
personne.setPrenompersonne("prenom6");
personne.setCoeffpersonne(new Integer(46));
personne.setDateaisspersonne(new Date());
session.save(personne);

Groupes groupe = (Groupes) session.load(Groupes.class, new Integer(1));
Grppers grppres = new Grppers();
grppres.setIdpersonne(personne);
grppres.setIdgroupe(groupe);
session.save(grppres);

tx.commit();
HibernateUtil.closeSession();
} catch (HibernateException e) {
e.printStackTrace();
}
}
}
}

```

Voici le même exemple utilisant les classes générées mettant en oeuvre le motif de conception DAO.

#### Exemple :

```

import java.util.*;
import net.sf.hibernate.*;
import com.jmd.test.hibernate.*;
import com.jmd.test.hibernate.dao.*;

public class Test2DAO {
    public static void main(String[] args) {
        try {
            _RootDAO.initialize();
            Session session = _RootDAO.createSession();
            Transaction tx = session.beginTransaction();

            Personnes personne = new Personnes();
            personne.setNompersonne("nom7");
            personne.setPrenompersonne("prenom7");
            personne.setCoeffpersonne(new Integer(46));
            personne.setDateaisspersonne(new Date());

            PersonnesDAO personnesDAO = new PersonnesDAO();
            personnesDAO.save(personne, session);

            GroupesDAO groupesDAO = new GroupesDAO();
            Groupes groupe = groupesDAO.load(new Integer(1), session);
            Grppers grppres = new Grppers();
            grppres.setIdpersonne(personne);
            grppres.setIdgroupe(groupe);

            GrppersDAO grppresDAO = new GrppersDAO();
            grppresDAO.save(grppres, session);

            tx.commit();
        } catch (HibernateException e) {
            e.printStackTrace();
        }
    }
}

```

Si une des tables à traiter ne contient que des données de références (un identifiant et une valeur), alors Hibernate Synchronizer propose de créer une classe particulière qui va encapsuler les données non pas de façon dynamique via un accès à la table mais de façon statique.

Dans ce cas, une boîte de dialogue demande si la classe générée doit l'être de façon statique (création comme une énumération)

Exemple : la table groupes ne possède que deux champs (idgroupe et nomgroupe)



En cliquant sur le bouton « Oui » la classe suivante est générée :

**Exemple :**

```
package com.jmd.test.hibernate;
import java.io.Serializable;
import net.sf.hibernate.PersistentEnum;

/**
 * This class has been automatically generated by Hibernate Synchronizer.
 * For more information or documentation, visit The Hibernate Synchronizer page
 * at http://www.binamics.com/hibernatesync or contact Joe Hudson at joe@binamics.com.
 */
public class Groupes implements Serializable, PersistentEnum {
    public static final Groupes GROUPE_2 = new Groupes(2);
    public static final Groupes GROUPE_1 = new Groupes(1);
    private final int code;

    protected Groupes(int code) {
        this.code = code;
    }

    public int toInt() { return code; }

    public static Groupes fromInt(int code) {
        switch (code) {
            case 2: return GROUPE_2;
            case 1: return GROUPE_1;
            default: throw new RuntimeException("Unknown value: " + code);
        }
    }

    public String toString () {
        switch (code) {
            case 2: return "groupe 2";
            case 1: return "groupe 1";
            default: return "Unknown value";
        }
    }
}
```

## **Partie 5 : le développement d'applications d'entreprise**

# **Partie 5 : le développement d'applications d'entreprise**

Cette partie concerne l'utilisation d'Eclipse pour développer des applications pour entreprise avec la version Enterprise Edition de Java.

Elle comporte les chapitres suivants :

- ◆ Le développement avec J2EE : Présente des plug-ins pour faciliter le développement avec J2EE.
- ◆ XML et Eclipse : détaille l'utilisation de plug-ins facilitant la mise en oeuvre de XML avec Eclipse.
- ◆ Le développement d'applications web : Propose de mettre en oeuvre le développement d'applications web grâce à des plug-ins d'Eclipse.
- ◆ Struts et Eclipse : détaille au travers d'un exemple l'utilisation du plug-in Easy Struts pour développement des applications web utilisant Struts.
- ◆ Java Server Faces et Eclipse : détaille au travers d'un exemple l'utilisation d'Eclipse pour développer une application web utilisant JSF.
- ◆ EJB et Eclipse : Propose de développer des EJB en utilisant des plug-ins d'Eclipse.
- ◆ Les services web et Eclipse : Propose d'utiliser Eclipse pour développer des services web manuellement ou avec le plug-in WTP.
- ◆ JPA et Eclipse : Propose de mettre en oeuvre le plug-in Dali pour faciliter la mise en oeuvre de l'API Java Persistence API.

## 20. Le développement avec J2EE

# Chapitre 20

Eclipse ne propose pas par défaut de plug-in permettant de faciliter le développement d'applications J2EE. Il existe cependant plusieurs plug-ins qui simplifient la mise en oeuvre de certaines tâches : Struts, Tomcat, EJB, ... ou des plug-ins plus complets permettant la mise en oeuvre des principales API de J2EE.

Le sous projet "Eclipse Web Tools Platform" a pour but de développer des plug-ins pour le développement d'applications Web et J2EE.

Le plug-in open source, nommé Lomboz, propose des fonctionnalités pour faciliter le développement d'applications web (servlets et JSP), d'EJB et de services web.

Ce chapitre va présenter les plug-ins suivants :

- le plug-in Tomcat de Sysdeo pour la mise en oeuvre de Tomcat pour des applications web
- le plug-in WTP développé en tant que sous projet d'Eclipse.
- le plug-in Lomboz

Ces deux derniers plug-in proposent de nombreuses fonctionnalités pour développer des applications J2EE avec utilisation possible de plusieurs serveurs d'applications.

### 20.1. Le plug-in Tomcat de Sysdeo

La société Sysdeo propose un plug-in pour faciliter le développement d'applications utilisant Tomcat notamment en permettant le démarrage et l'arrêt de Tomcat, le packaging d'une application sous la forme d'un fichier .war et son déploiement dans Tomcat.

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Plug-in Tomcat de Sysdeo	2.2.1 et 3.0

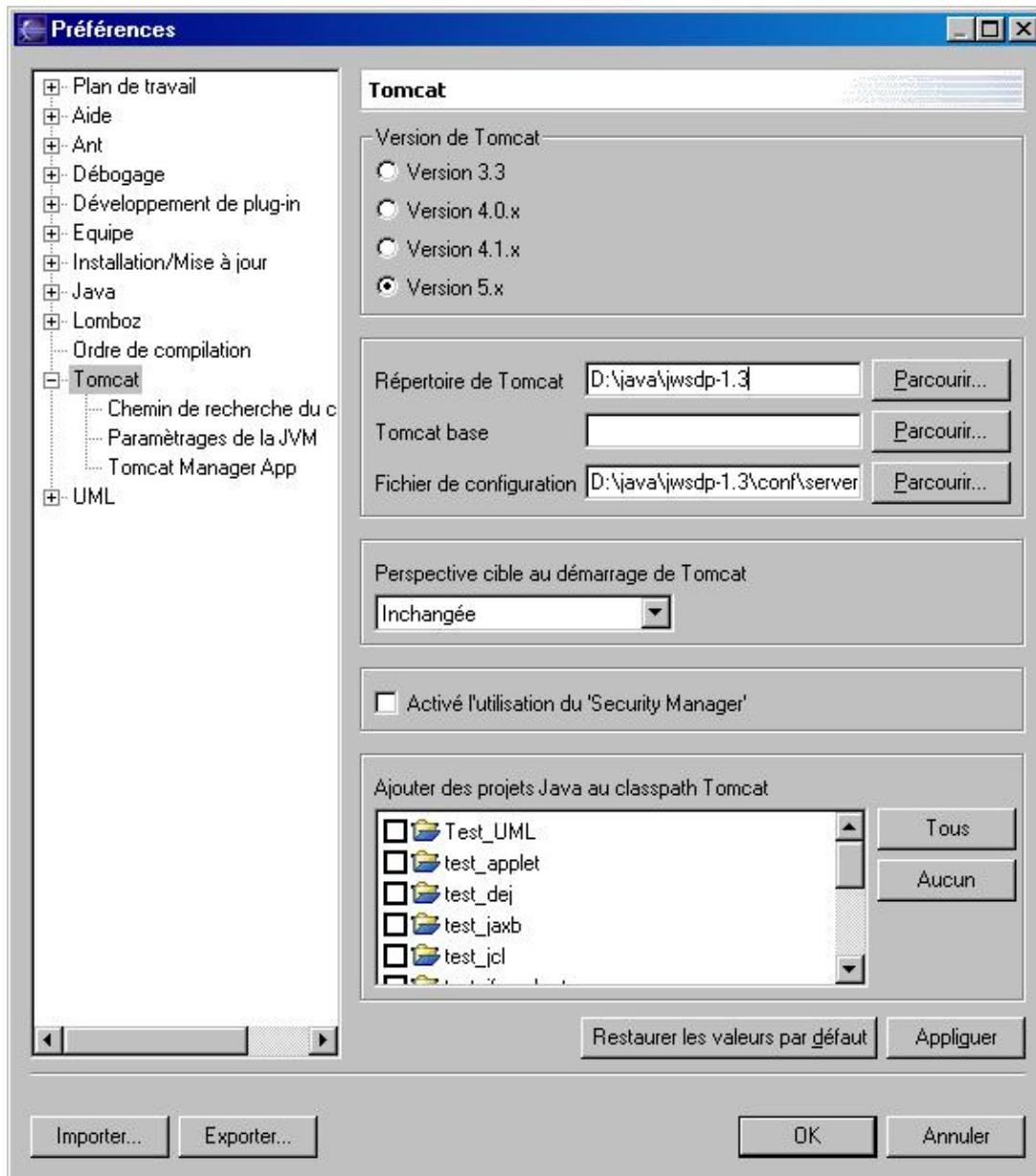
La page web du plug-in est consultable à l'URL : <http://www.sysdeo.com/eclipse/tomcatPluginFR.html>.

#### 20.1.1. Installation et paramétrage de la version 2.2.1

Il faut télécharger le fichier tomcatPluginV221.zip et décompresser son contenu dans le répertoire plugins du répertoire d'installation d'Eclipse.

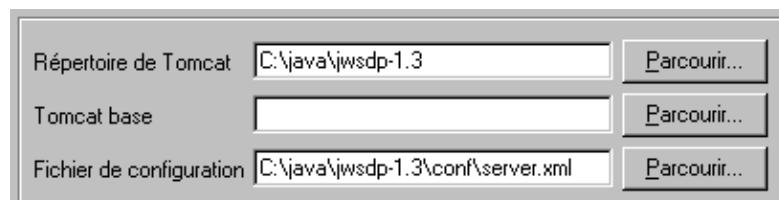
Il est possible d'ajouter un menu nommé « Tomcat » et trois boutons dans la barre d'outils pour démarrer, arrêter et redémarrer Tomcat : . Pour cela, il faut utiliser l'option « Personnaliser la perspective ... » du menu « Fenetre » et cocher l'option « Autre / Tomcat » dans les éléments disponibles.

Pour pouvoir utiliser les fonctionnalités du plug-in, il faut le configurer notamment pour lui préciser où se situe Tomcat. Cette configuration utilise plusieurs pages dans les préférences pour saisir les informations utiles.

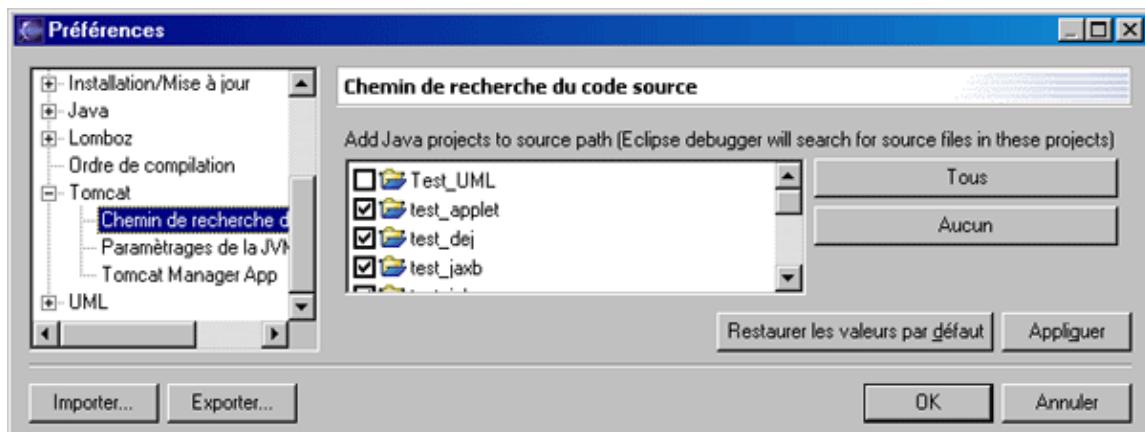
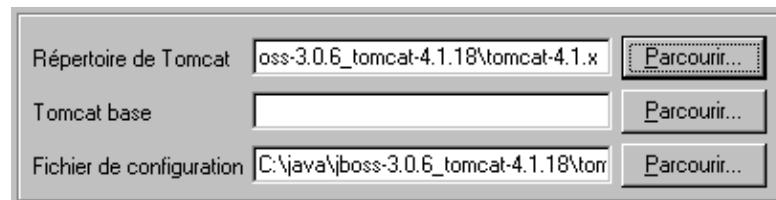


La première page permet de saisir les informations principales notamment la version de Tomcat utilisée et le répertoire de base de Tomcat.

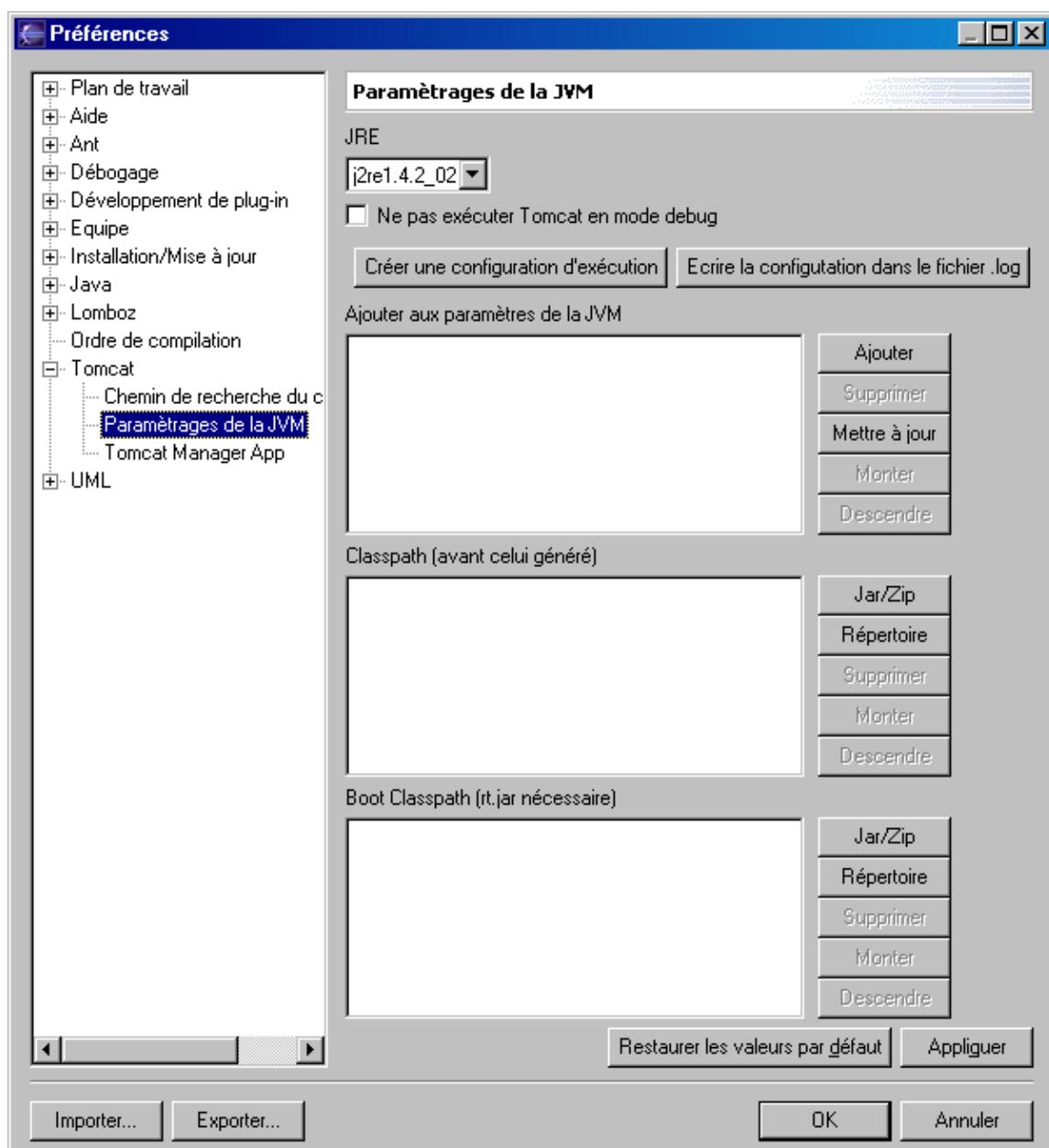
Exemple avec Tomcat fourni avec le JWSDP 1.3



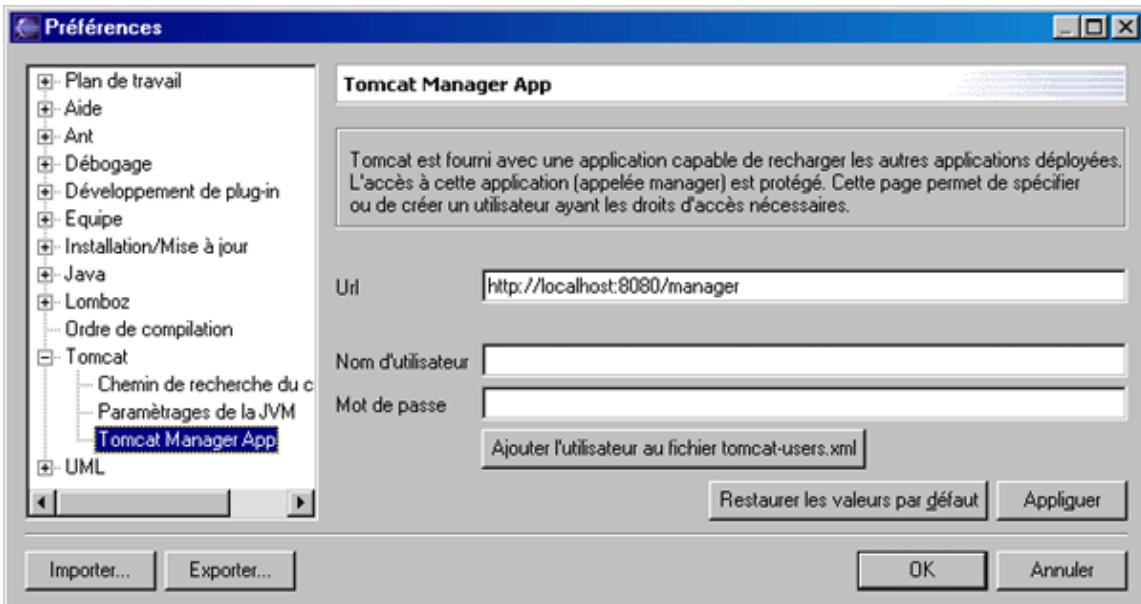
Exemple avec Tomcat fourni avec Jboss 3.0.6



La seconde vue permet de préciser des paramètres pour la JVM dans laquelle va s'exécuter Tomcat.



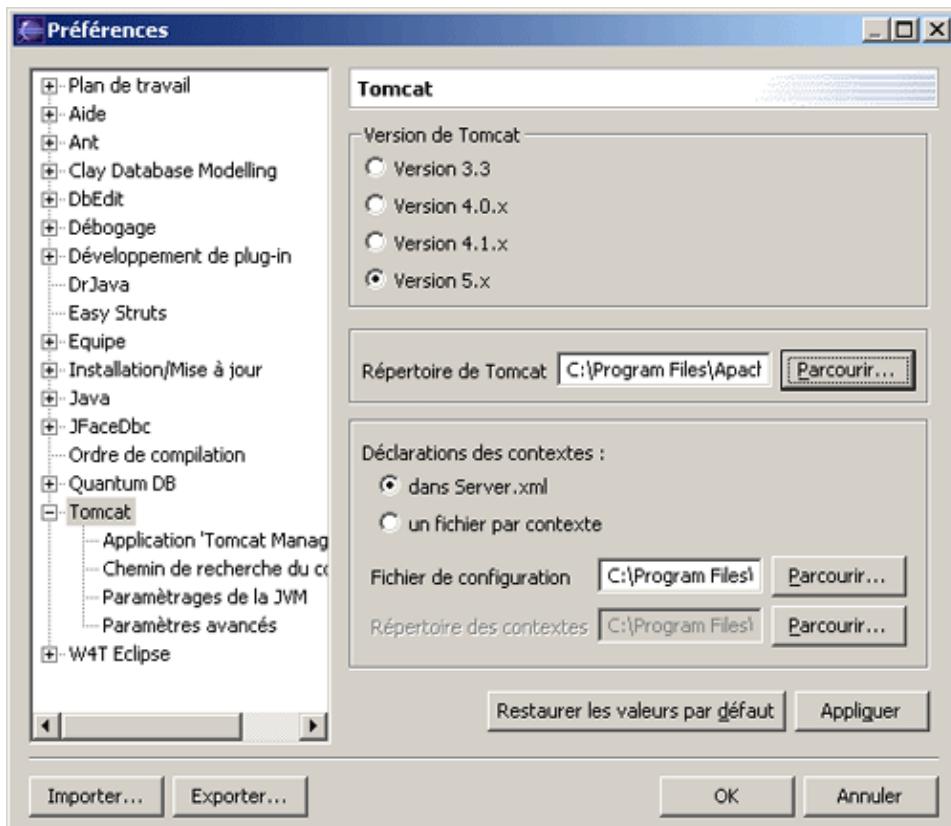
La troisième page permet de saisir les informations concernant le démarrage de l'application de gestion de Tomcat.



## 20.1.2. Installation et paramétrage de la version 3.0

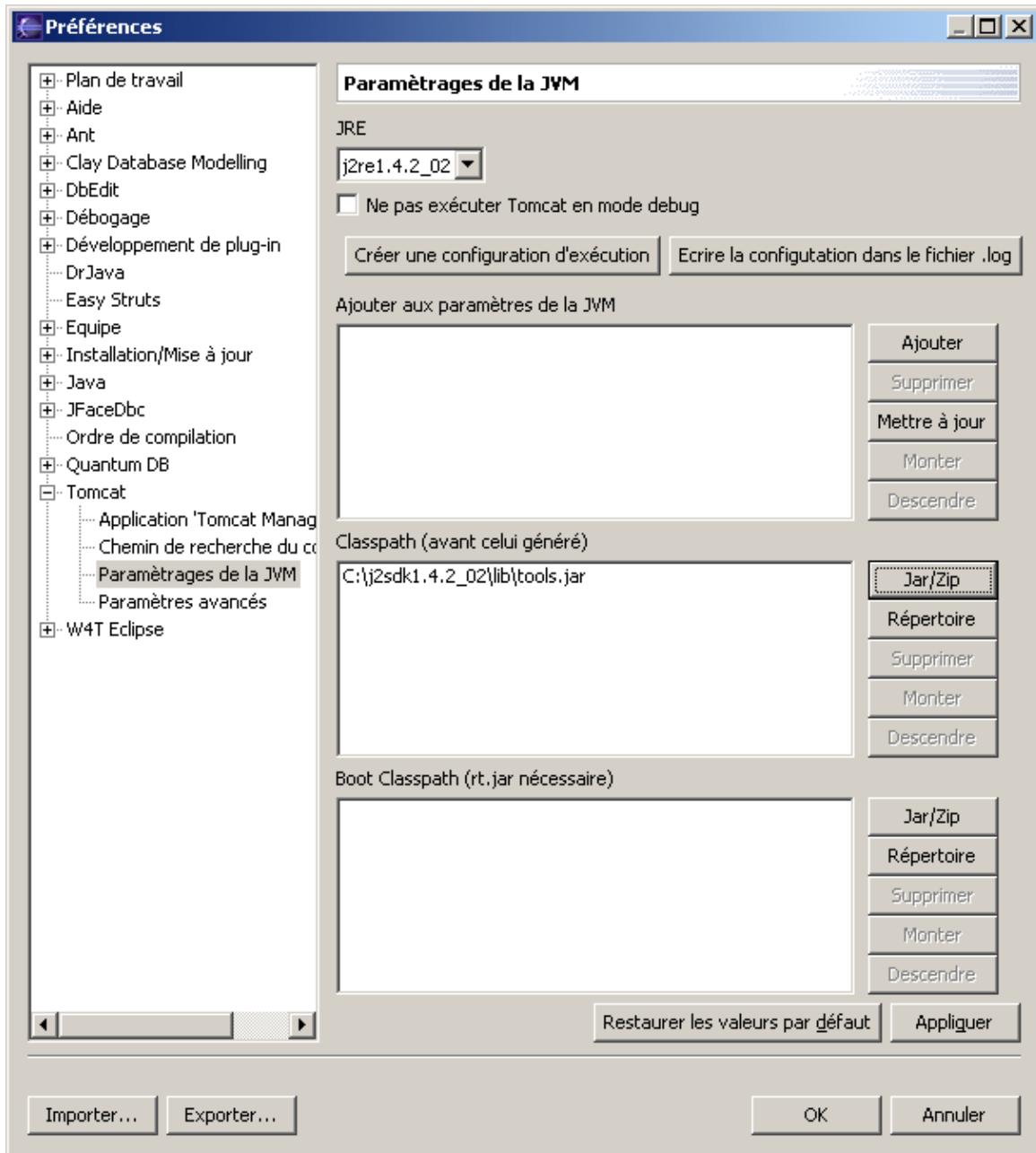
Il faut télécharger le fichier tomcatPluginV3.zip sur le site de Sysdeo : <http://www.sysdeo.com/eclipse/tomcatPluginFR.html> et décompresser son contenu dans le répertoire "plugins" du répertoire d'installation d'Eclipse.

Il faut ensuite lancer Eclipse et configurer le plug-in : sélectionnez le menu « Fenêtre/Préférences », puis l'option Tomcat dans l'arborescence.



Sur cette page, sélectionnez la version de Tomcat utilisée (version 5.x dans cette section) et sélectionnez le répertoire de Tomcat.

Sélectionnez l'option « Tomcat/Paramètres de la JVM » et ajoutez dans la liste "Classpath" le fichier tools.jar qui se situe dans le répertoire lib du JDK.



Cliquez sur le bouton « OK »

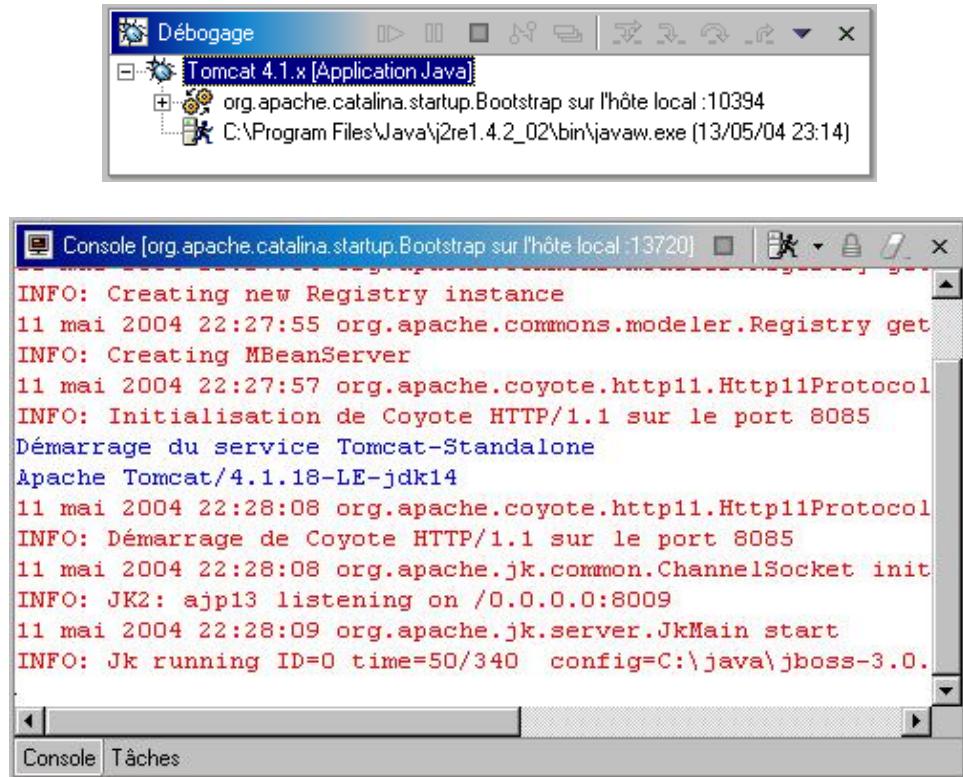
Sous la perspective « Java », sélectionnez l'option « Fenêtre / Personnaliser la perspective » puis dans l'arborescence, sélectionnez « Autres » et cochez la case « Tomcat ». Cliquez sur le bouton « OK » : les trois boutons permettant respectivement de démarrer, arrêter et redémarrer Tomcat sont ajoutés dans la barre d'outils comme avec la version précédente du plug-in.

Cliquez sur le bouton de démarrage de Tomcat : la console affiche les messages de démarrage de l'application. Pour vérifier le bon fonctionnement, il suffit d'ouvrir un navigateur à l'url : <http://localhost:8080>

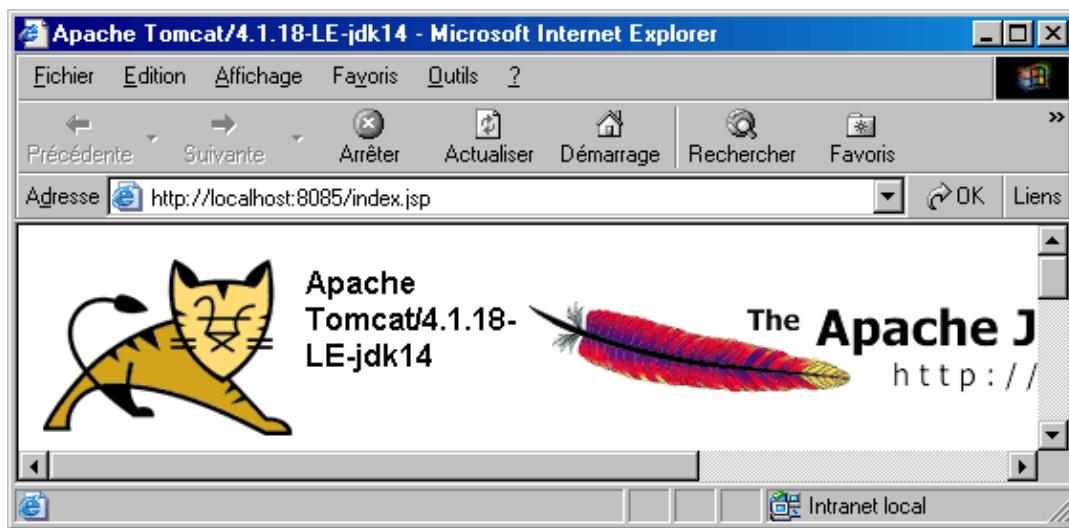
### 20.1.3. Lancement et arrêt de Tomcat

Pour lancer Tomcat, il suffit de cliquer sur le bouton  de la barre d'outils ou d'utiliser l'option « Démarrer Tomcat » du menu « Tomcat ».

La perspective « Debug » s'affiche et les informations relatives au lancement de Tomcat s'affichent dans la vue « Console ».



Pour vérifier la bonne exécution de Tomcat, il suffit d'ouvrir un navigateur et de saisir l'url <http://localhost> avec le port précisé dans la configuration dans Tomcat.



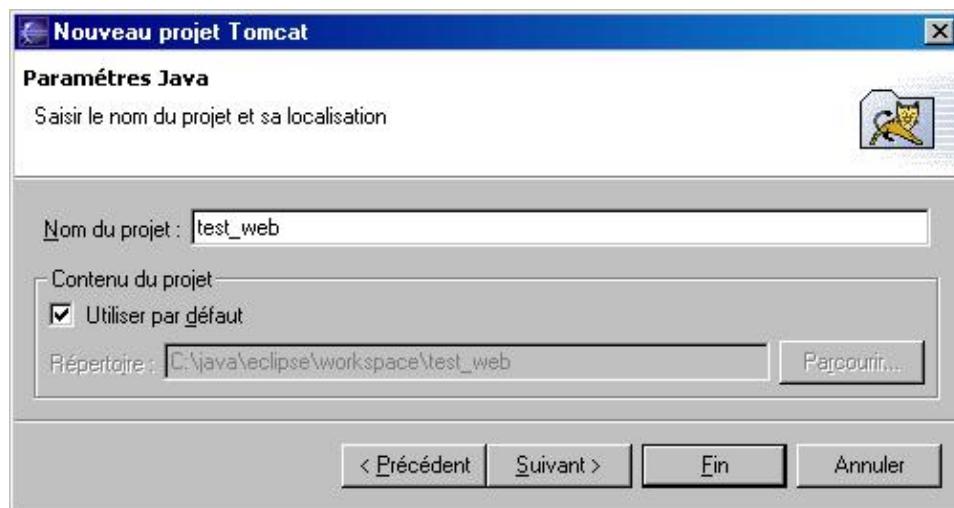
Pour arrêter Tomcat, il suffit de cliquer sur l'icône  ou de sélectionner l'option « Arrêter Tomcat » du menu « Tomcat ».

## 20.1.4. La création d'un projet Tomcat

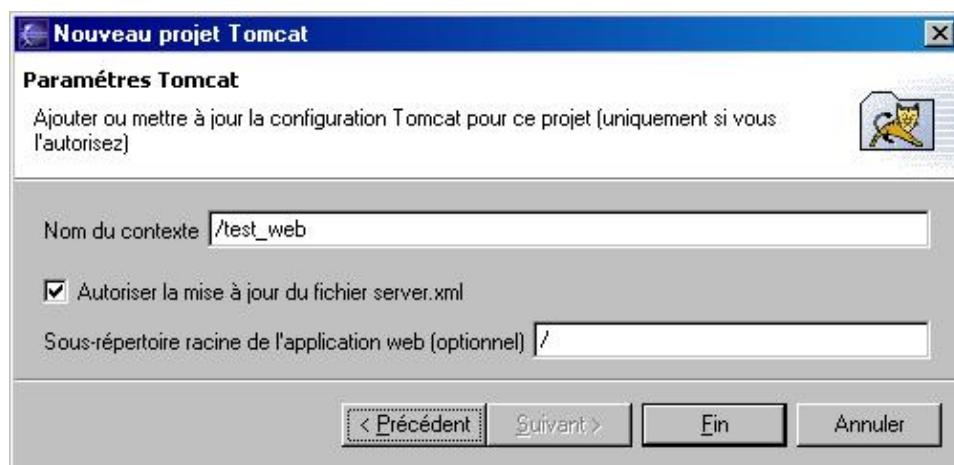
Le plug-in permet la création d'un projet particulier pour développer des applications web.



Cliquez sur le bouton « Suivant »

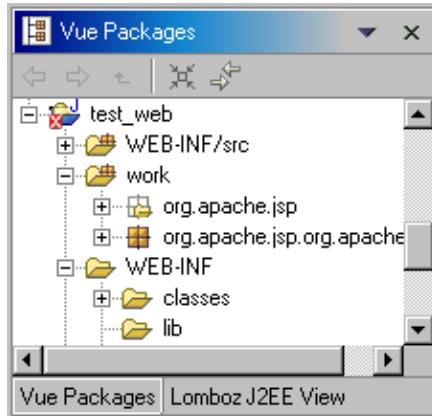


Saisissez le nom du projet et cliquez sur le bouton « Suivant ».



La page suivante de l'assistant permet de préciser certains paramètres : le nom du contexte de l'application web (par défaut, celui du projet), le répertoire racine de l'application et permettre au plug-in de modifier ou non le fichier server.xml de Tomcat.

En cliquant sur le bouton « Fin », le projet est créé.



La structure du projet reprend celle d'une application web incluse dans un fichier .war. Le fait d'autoriser le plug-in à modifier le fichier server.xml provoque sa mise à jour pour ajouter un nouveau contexte :

Exemple :

```
<Context path="/test_web" reloadable="true" docBase="D:\java\eclipse\workspace\test_web"
workDir="D:\java\eclipse\workspace\test_web\work\org\apache\jsp" />
```

Si Tomcat était déjà en cours d'exécution, il faut le relancer pour que les modifications soient prises en compte.

## 20.2. Le plug in WTP (Eclipse Web Tools Platform)

Le projet Eclipse Web Tools Platform (WTP) a pour but d'enrichir la plate-forme Eclipse avec des outils pour développer des applications web en utilisant J2EE. Il inclut :

- Des éditeurs pour les langages de pages web (HTML, CSS, Javascript)
- Des éditeurs pour des documents XML et associés (XML, DTD, XSD et WSDL)
- Le support de projet J2EE via des assistants
- Le support des services web via des assistants
- Le support des bases de données via SQL

Le projet WTP est composé de plusieurs sous projets :

- **Web Standard Tools (WST)**

Ce sous projet a pour but de développer un socle pour le développement d'applications web sous Eclipse.

- **J2EE Standard Tools (JST)**

Ce sous projet a pour but de développer des plug-ins pour faciliter le développement d'applications respectant la norme J2EE 1.4.

- **Java Server Faces (JSF)**

Ce sous projet a pour but de développer des plug-ins pour faciliter le développement d'application web utilisant les JSF

- **Dali**

Ce sous projet a pour but de développer des plug-ins pour faciliter le mapping O/R avec l'API JPA

- **Ajax Toolkit Framework (ATF)**

Le site officiel du projet est à l'url <http://www.eclipse.org/webtools/>

Le plug-in supporte plusieurs conteneurs et serveurs d'applications : Tomcat (version 3.2 à 5.5), JBoss 3.2.3, JOnAS 4.1.4 et Weblogic 8.1.

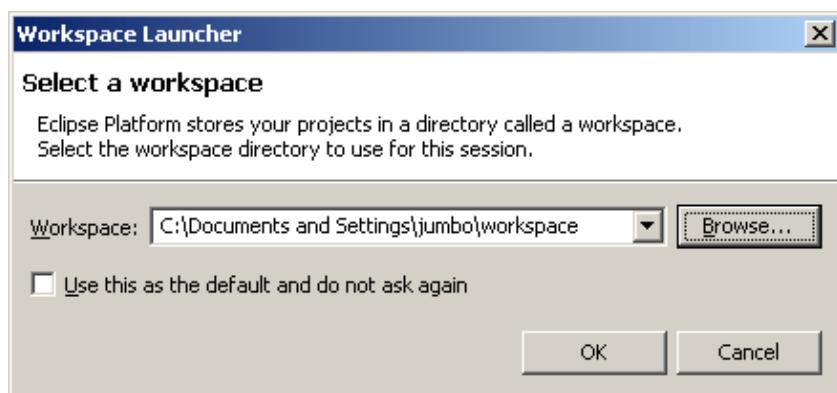
## 20.2.1. Installation de la version 1.0M3

Il faut télécharger le fichier wtp-eclipse-prereqs-sdk-1.0M3-win32.zip et le décompresser dans un répertoire du système en suivant les liens sur la page <http://www.eclipse.org/webtools/index.html>

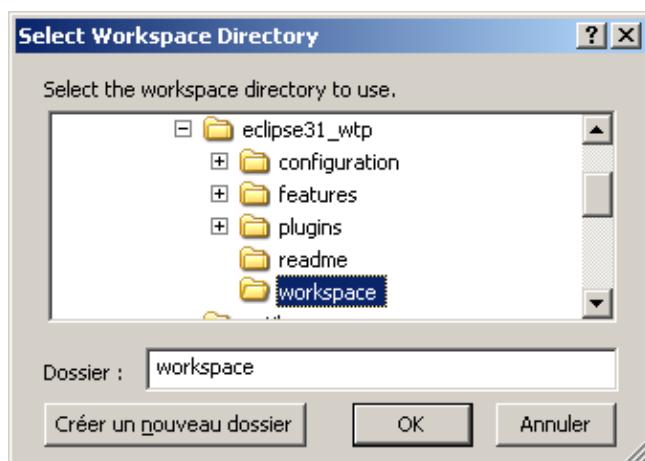
Attention, ce fichier possède une taille de 151Mo.

Le répertoire Eclipse résultant de la décompression est renommé en eclipse31\_wtp dans les exemples de ce chapitre et un sous répertoire workspace y est créé pour stocker les différents projets.

Lancez Eclipse à partir du répertoire décompressé.

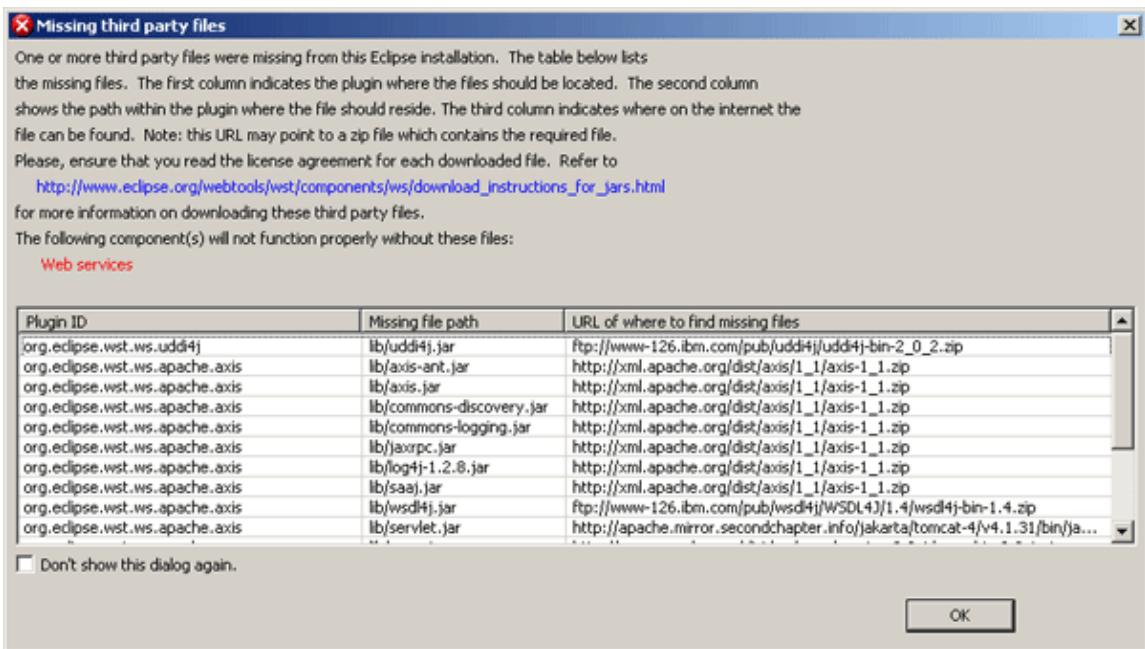


Cliquez sur le bouton « Browse »



Sélectionnez le répertoire workspace créé et cliquez sur le bouton « OK », puis une nouvelle fois sur le bouton « OK » pour valider le workspace sélectionné.

Au lancement d'Eclipse, un message informe du manque de certains plug-ins qui empêche l'utilisation des fonctionnalités concernant les « Web services »



Cliquez sur le bouton « OK »

Il est donc nécessaire de télécharger les plug-ins manquants qui ne sont pas fournis dans le bundle.

Créez un répertoire temp dans le répertoire eclipse31\_wtp et téléchargez dedans les fichiers dont l'url est indiquée dans le message.

Fichier	Taille
axis-1_1.zip	10.2 Mo
jakarta-tomcat-4.1.31.zip	9.7 Mo
soap-bin-2.3.1.zip	1,6 Mo
uddi4j-bin-2_0_2.zip	1 Mo
wsil4j.jar	78 Ko
jaf-1_0_2-upd.zip	349 Ko
javamail-1_3_2.zip	2.3 Mo
wsdl4j-bin-1.4.zip	891 Ko

Il faut décompresser ces fichiers dans leur répertoire approprié dans le sous répertoire plugins. Pour cela, le plus simple est d'utiliser un script Ant fourni par le projet (Attention, Ant doit être installé et configuré sur le poste pour pouvoir utiliser ce script).

Il faut télécharger le fichier à l'url

[http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/org.eclipse.wtp.releng/fetchVendorContent.xml?content-type=text/xml&cvsroot=WebTools\\_Project](http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/org.eclipse.wtp.releng/fetchVendorContent.xml?content-type=text/xml&cvsroot=WebTools_Project),

l'enregistrer dans le répertoire temp et l'exécuter en lui fournissant les paramètres indiqué dans l'exemple ci-dessous :

Exemple :

```
C:\java\eclipse31_wtp\temp>ant -DlocalDownloads=C:\java\eclipse31_wtp\temp -Dbui
ldDirectory=C:\java\eclipse31_wtp\plugins -f fetchVendorContent.xml
Buildfile: fetchVendorContent.xml
run:
extractAllBinary:
```

```

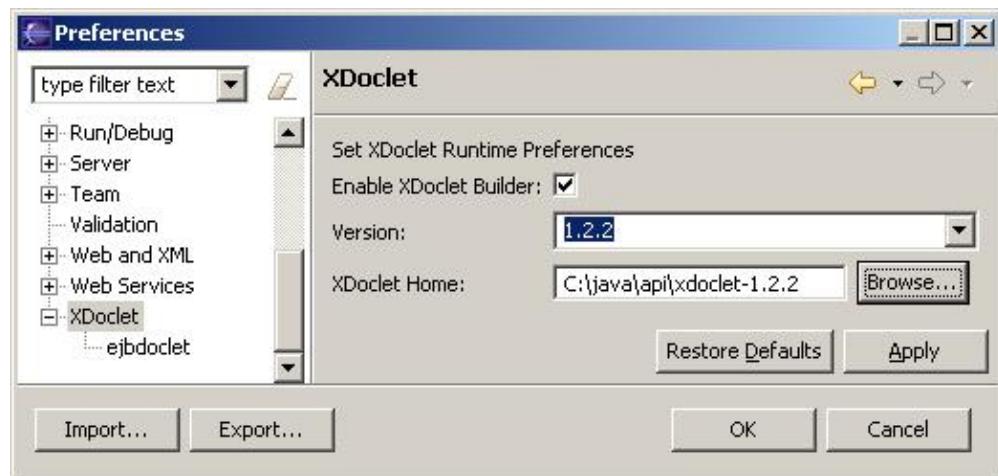
extractAll:
[unzip] Expanding: C:\java\eclipse31_wtp\temp\axis-1_1.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\jakarta-tomcat-4.1.31.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\jaf-1_0_2-upd.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\javamail-1_3_2.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\soap-bin-2.3.1.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\uddi4j-bin-2_0_2.zip into C:\java\eclipse31_wtp\temp
[unzip] Expanding: C:\java\eclipse31_wtp\temp\wsdl4j-bin-1.4.zip into C:\java\eclipse31_wtp\temp
[move] Moving 8 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.axis_1.0.0
[move] Moving 1 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.axis_1.0.0
[move] Moving 1 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.soap_1.0.0
[move] Moving 1 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.soap_1.0.0
[move] Moving 1 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.soap_1.0.0
[copy] Copying 1 file to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.apache.wsil_1.0.0
[move] Moving 1 files to C:\java\eclipse31_wtp\temp\org.eclipse.wst.ws.uddi4j_1.0.0
grabjarsxml:
copyJars:
copyJars2:
copyJars3:
[copy] Copying 9 files to C:\java\eclipse31_wtp\plugins\org.eclipse.wst.ws.apache.axis_1.0.0\lib
copyJars:
copyJars2:
copyJars3:
[copy] Copying 3 files to C:\java\eclipse31_wtp\plugins\org.eclipse.wst.ws.apache.soap_1.0.0\lib
copyJars:
copyJars2:
copyJars3:
[copy] Copying 1 file to C:\java\eclipse31_wtp\plugins\org.eclipse.wst.ws.apache.wsil_1.0.0\lib
copyJars:
copyJars2:
copyJars3:
[copy] Copying 1 file to C:\java\eclipse31_wtp\plugins\org.eclipse.wst.ws.uddi4j_1.0.0\lib
[copy] Copying 2 files to C:\java\eclipse31_wtp\plugins\org.eclipse.wst.wsdl_1.0.0\lib
extractAllSDK:
BUILD SUCCESSFUL
Total time: 17 seconds
C:\java\eclipse31_wtp\temp>cd ..
C:\java\eclipse31_wtp>eclipse -clean

```

Il faut relancer Eclipser avec l'option -clean dans une boîte de commande Dos

## 20.2.2. Configuration de XDoclet

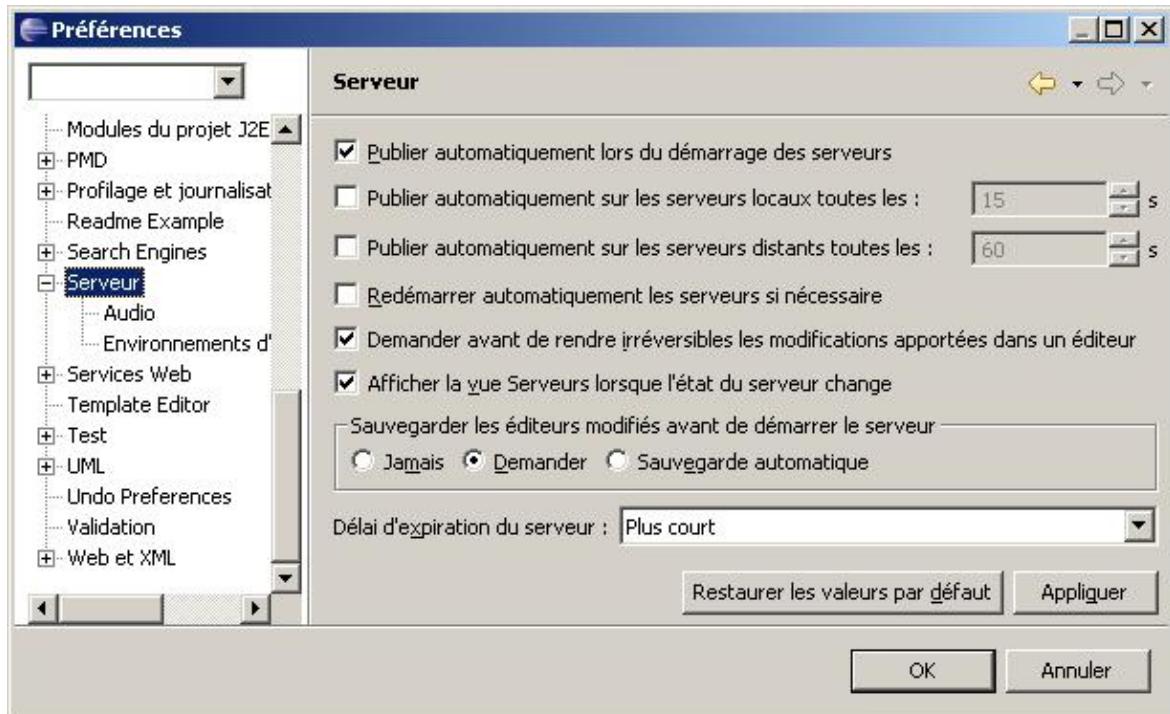
XDoclet doit être installé sur le système. Afin de permettre une utilisation de XDoclet par le plug-in, il faut le configurer dans les préférences.



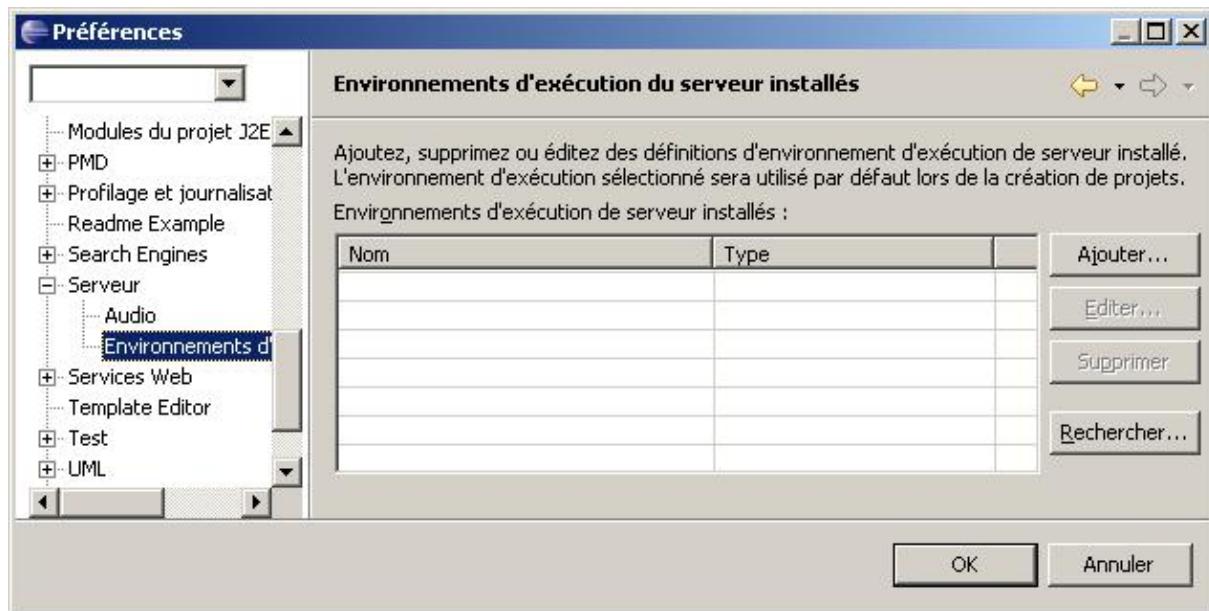
Il faut absolument cocher la case « Enable XDoclet Builder », sélectionner la version à utiliser et sélectionner le répertoire qui contient XDoclet sur le système.

## 20.2.3. Configuration d'un serveur

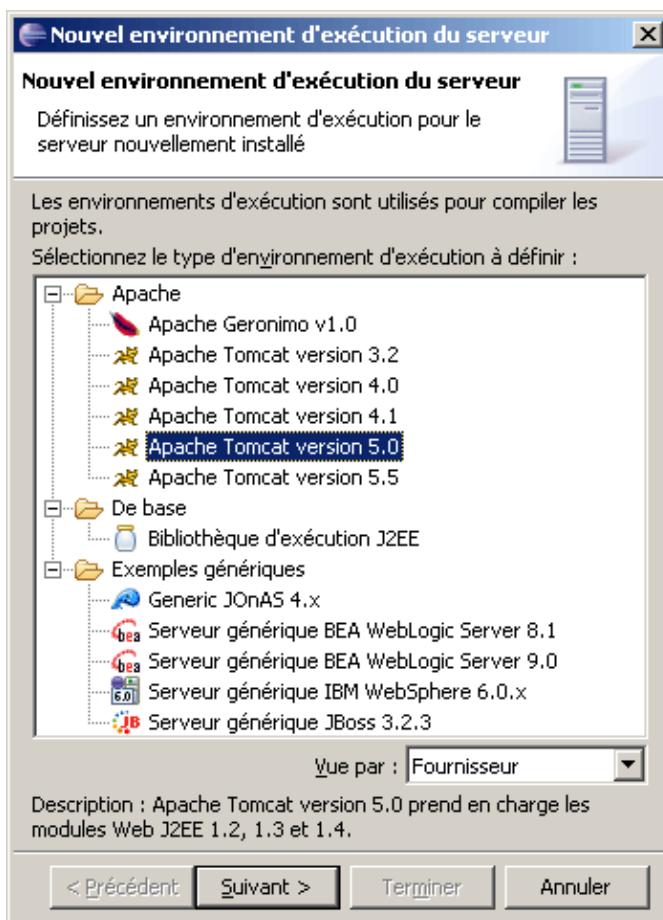
WTP supporte plusieurs conteneurs et serveurs d'applications : Tomcat, JBoss, JOnAS, ... Il est nécessaire de configurer un ou plusieurs serveurs dans les préférences du WTP. La page principale du noeud Server des préférences permet de régler des paramètres généraux concernant les serveurs.



La page « Environnements d'exécution installés » permet de configurer un ou plusieurs serveurs.



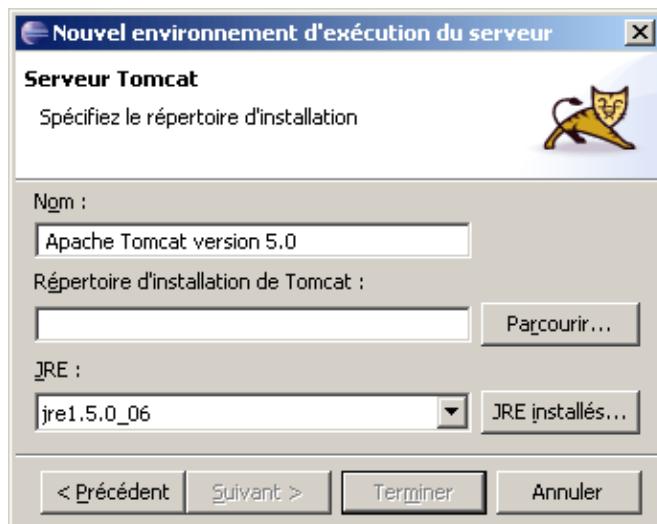
Pour ajouter un nouveau serveur, il faut cliquer sur le bouton « Ajouter ».



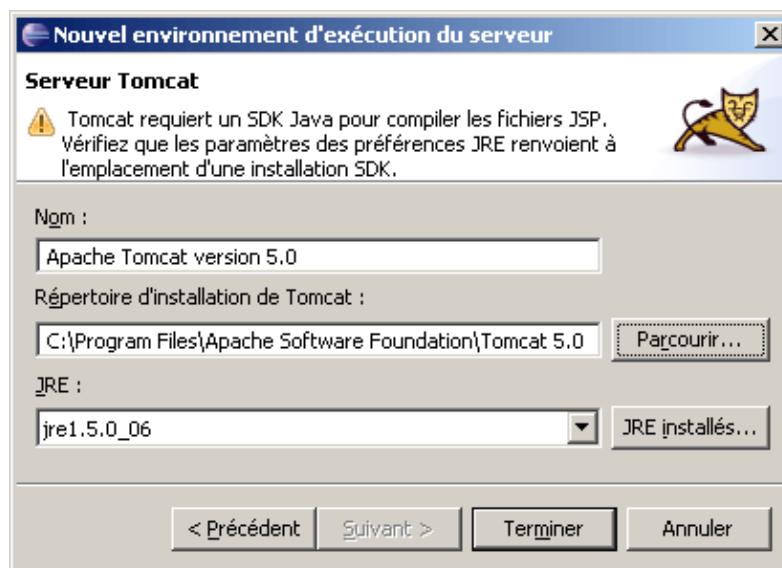
Il existe trois grands types de serveurs :

- Apache : permet de sélectionner le conteneur Tomcat ainsi que sa version dans lequel il est possible de modifier dynamiquement des éléments de l'application sans redéployer toute l'application packagée
- Exemples génériques (Generic Server runtime) : permet de sélectionner JBoss, JOnAS, Websphere ou Weblogic dans lesquels l'application doit être déployée sous la forme packagée avant d'être exécutée
- De base / Bibliothèque d'exécution J2EE (Basic/J2EE Runtime Library) ne désigne pas un conteneur ou un serveur d'application mais simplement un ensemble de bibliothèques

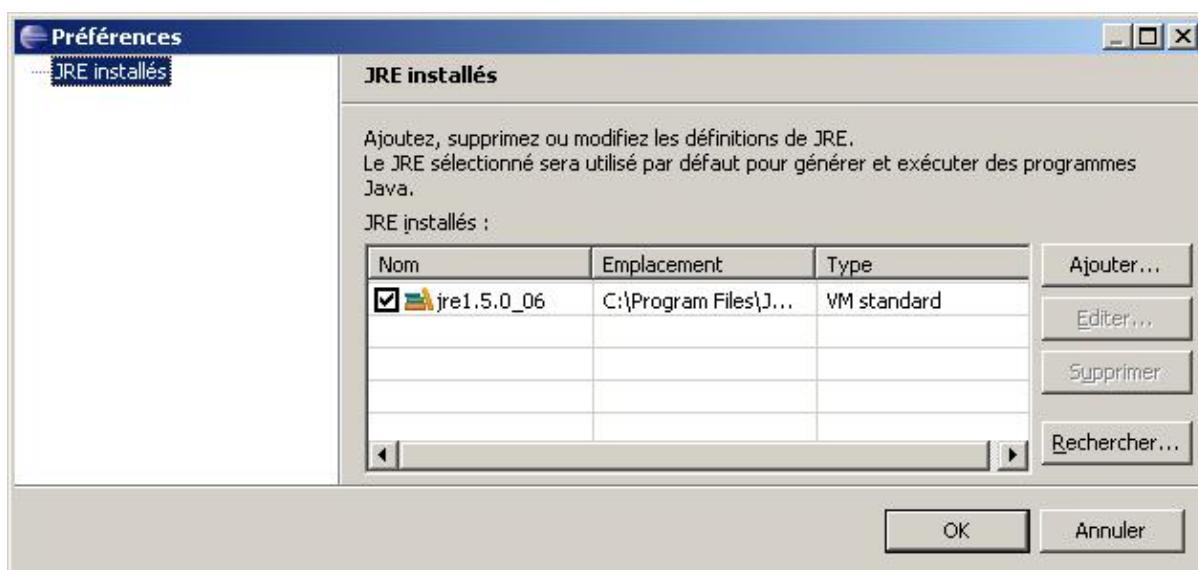
Pour Tomcat 5.0 par exemple, il faut sélectionner « Apache/Apache Tomcat v5.0 » et cliquer sur le bouton « Suivant ».



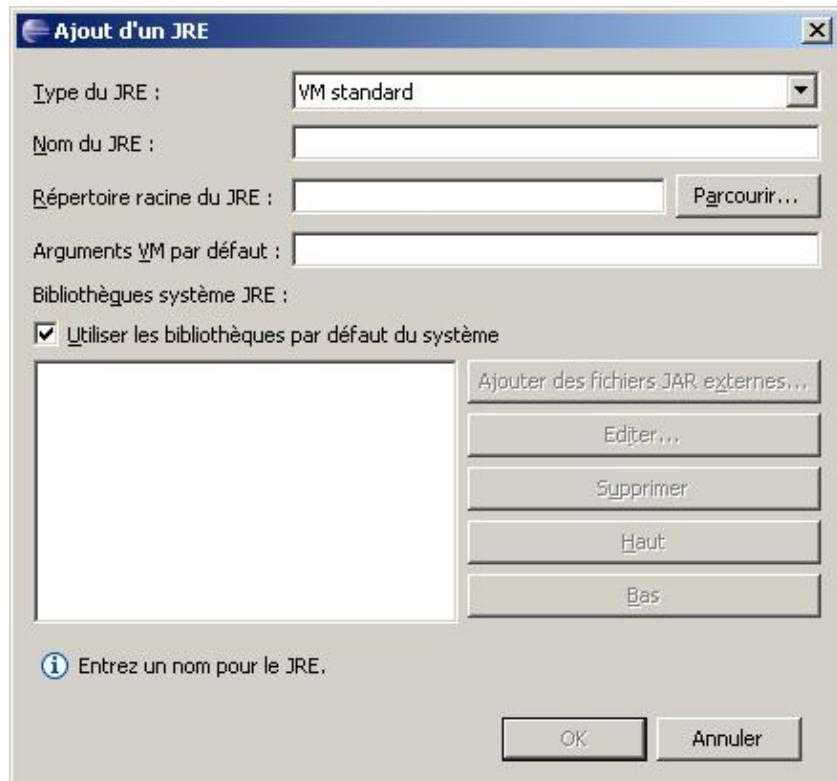
Cliquez sur le bouton « Parcourir » et sélectionnez le répertoire qui contient Tomcat



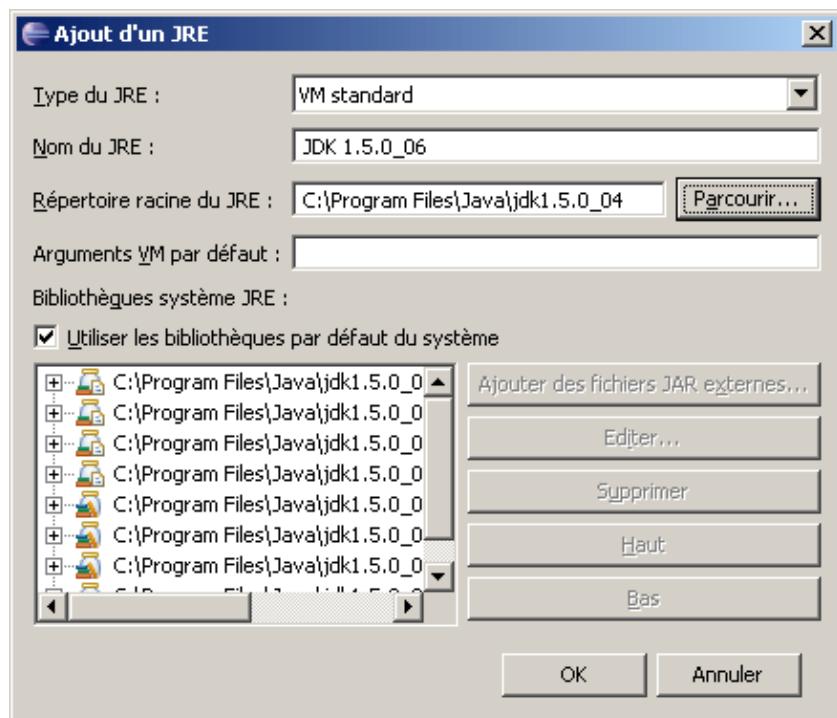
Pour utiliser Tomcat, un JDK est requis notamment pour permettre la compilation des JSP. Si aucun JDK n'est configuré, il faut cliquer sur le bouton « JRE installés »



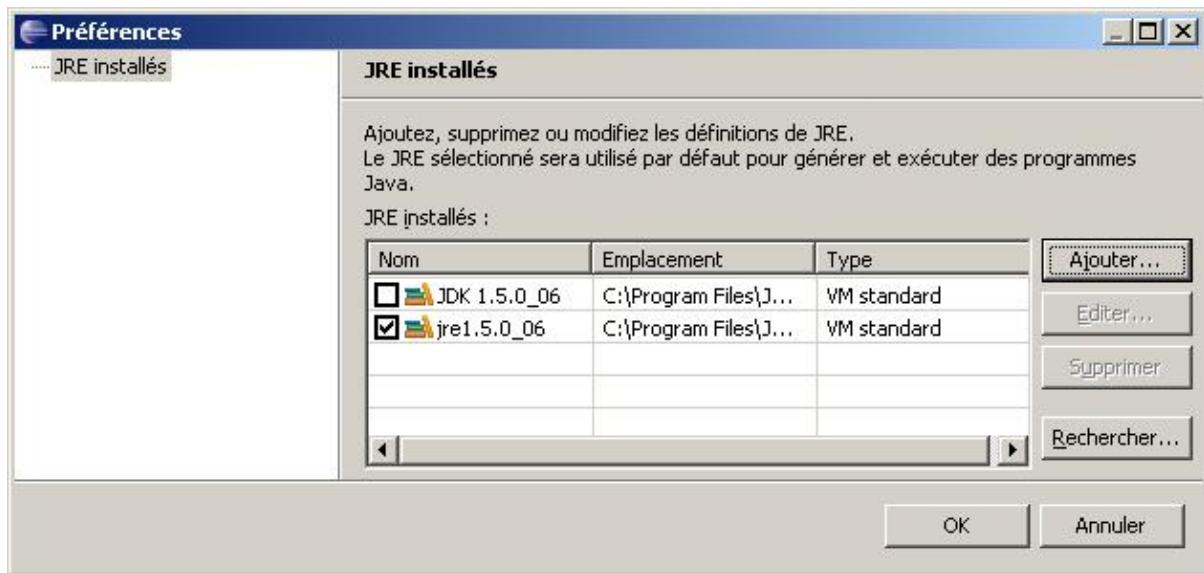
Cliquez sur le bouton « Ajouter ... »



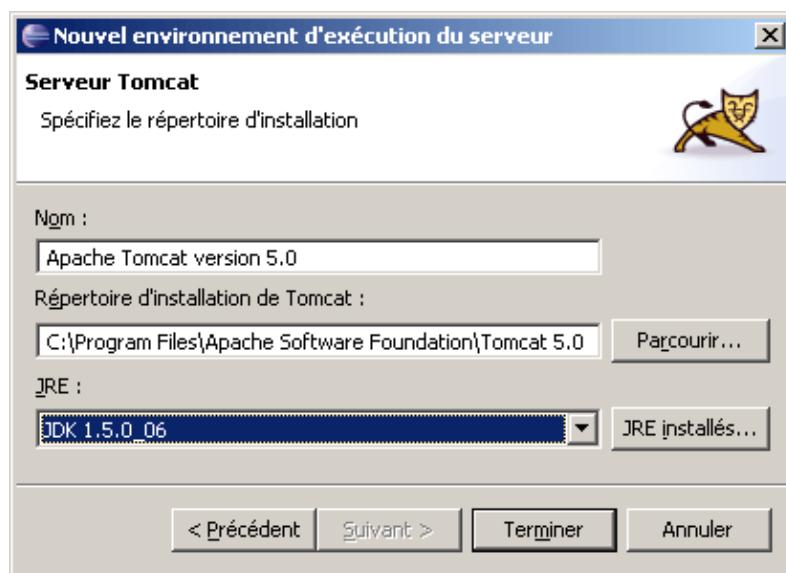
Saisissez le nom du JRE et sélectionnez le répertoire. Eclipse renseigne automatiquement les autres éléments requis mais il est possible de les modifier.



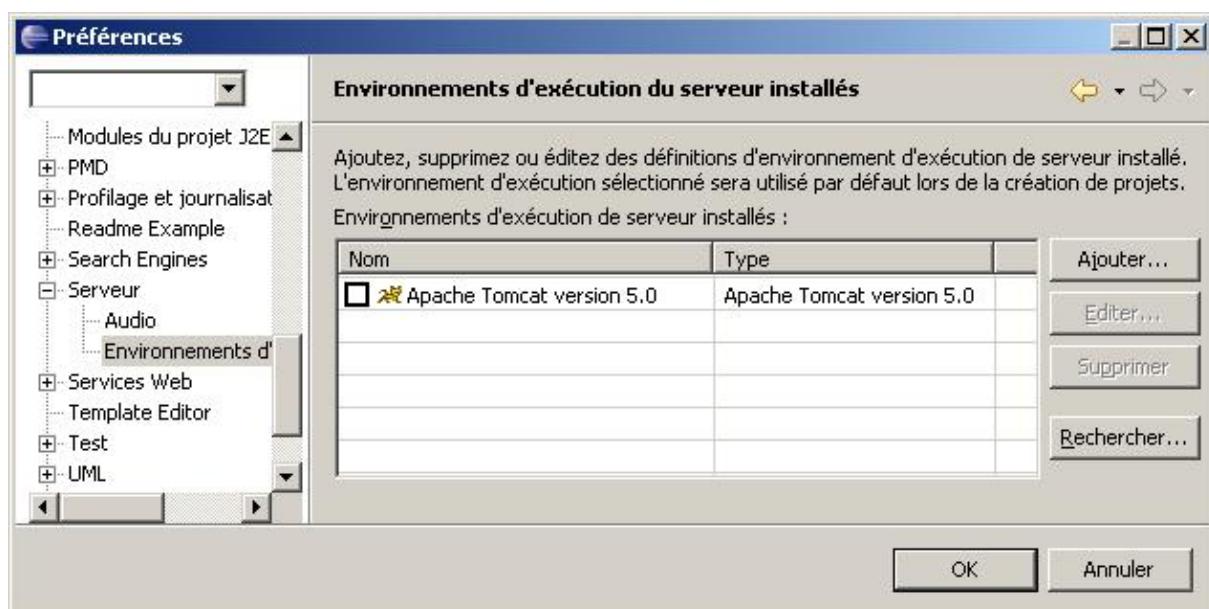
Cliquez sur le bouton « OK » pour ajouter le nouveau JRE.



Cliquez sur le bouton « OK »

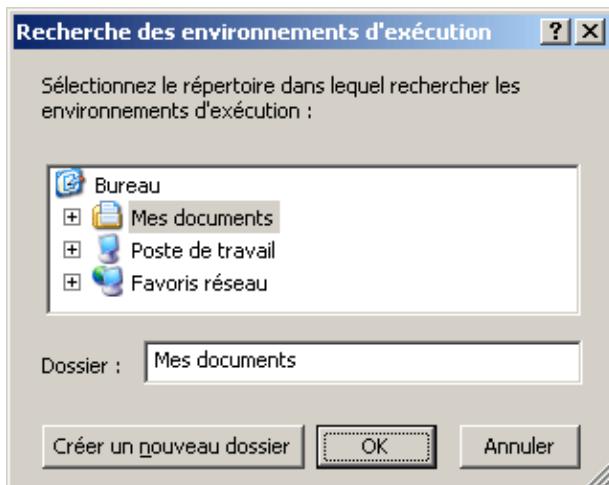


Il suffit alors de sélectionner le nouveau JRE défini et de cliquer sur le bouton « Terminer ».



Le nouveau serveur est ajouté. Il est possible de le définir comme serveur par défaut en cochant sa case à cocher et en cliquant sur le bouton « OK ».

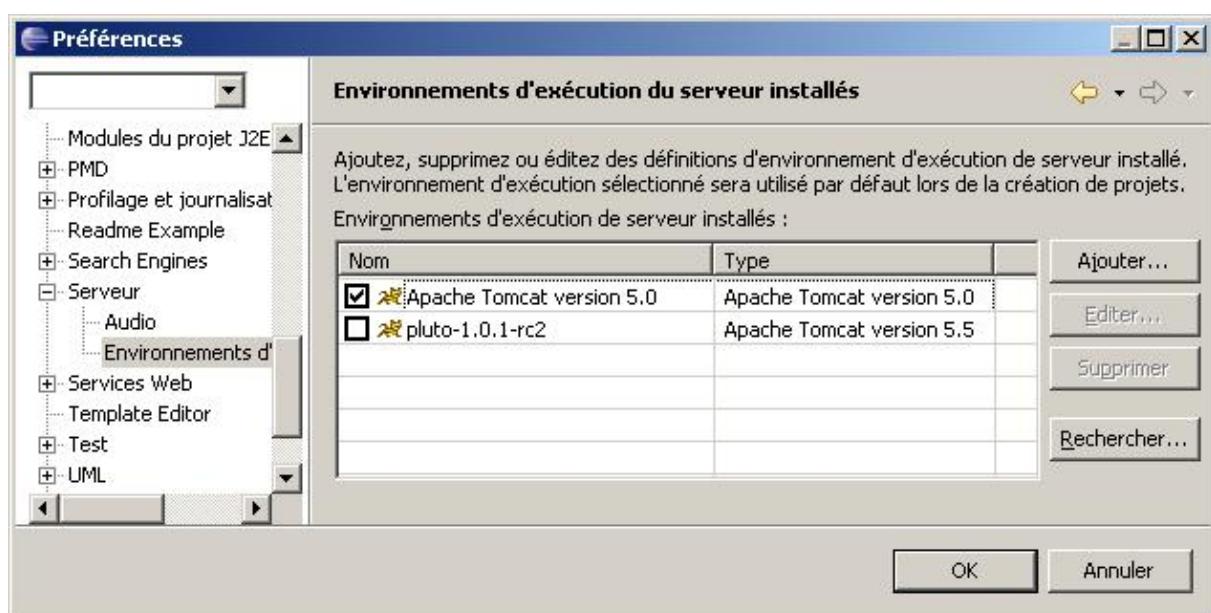
Cliquez sur le bouton « Rechercher ... » pour demander la recherche des serveurs connus par le plug-in et installés sur la machine.



Sélectionnez le répertoire de départ de la recherche et cliquez sur le bouton « OK »



Les différents serveurs trouvés sont automatiquement ajoutés



## 20.3. Lomboz

Lomboz est un plug-in dont le but est de faciliter le développement d'applications J2EE 1.3. Initialement développé par la société ObjectLearn (<http://www.objectlearn.com/index.jsp>), son statut est passé à open-source depuis l'intégration du projet au consortium ObjectWeb, début 2004 (<http://forge.objectweb.org/projects/lomboz>) .

Ce plug-in utilise plusieurs outils open source pour mener à bien différentes tâches : Ant, Xdoclet, Axis, ... ce qui lui permet de couvrir le cycle de développement des applications J2EE : rédaction et génération du code, déploiement et débogage.

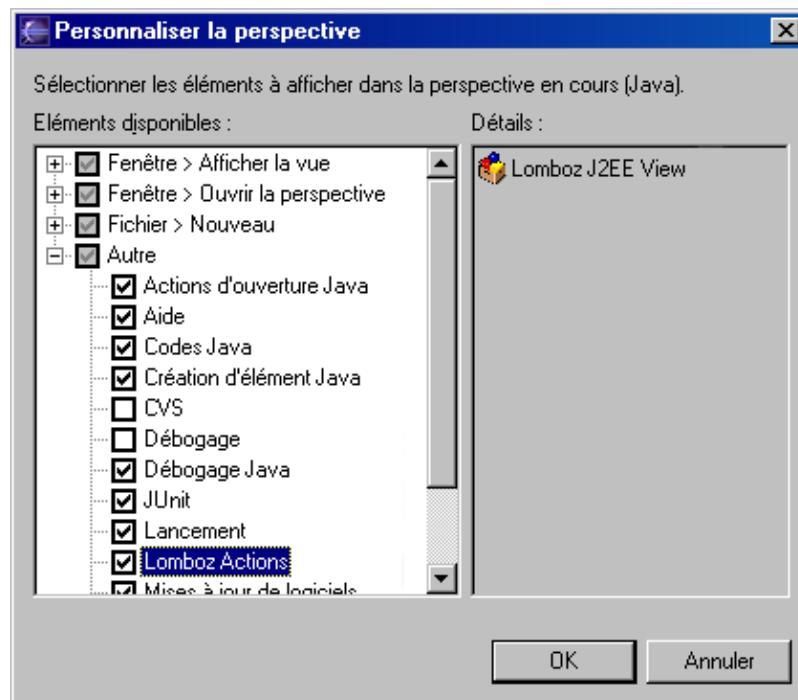
Cette section va utiliser Lomboz avec JBoss et Tomcat.

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Lomboz	2.1.2
JBoss	3.0.6
Tomcat	4.1.18

### 20.3.1. Installation et configuration

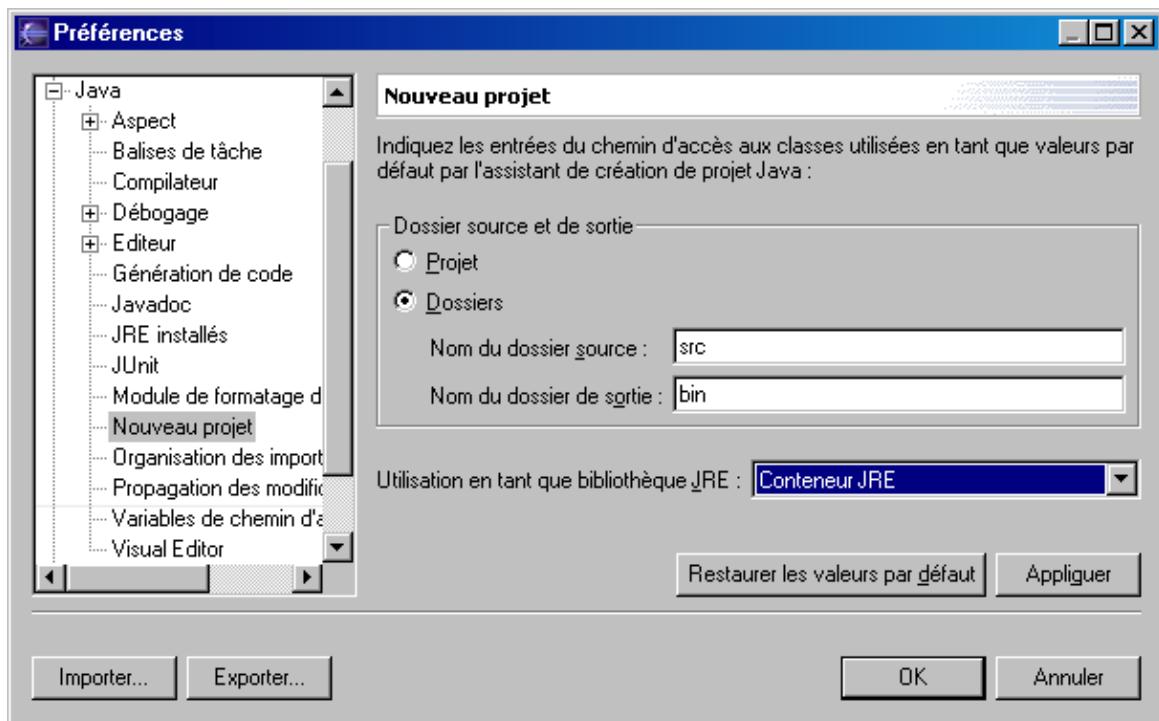
Téléchargez le fichier lomboz.21\_02[1].zip et le décompresser dans le répertoire d'installation d'Eclipse. L'installation de Lomboz se fait comme pour les autres plug-ins en décompressant l'archive dans le répertoire d'Eclipse.

Pour configurer Lomboz, il faut utiliser l'option « Fenêtre / Personnaliser la perspective ... »

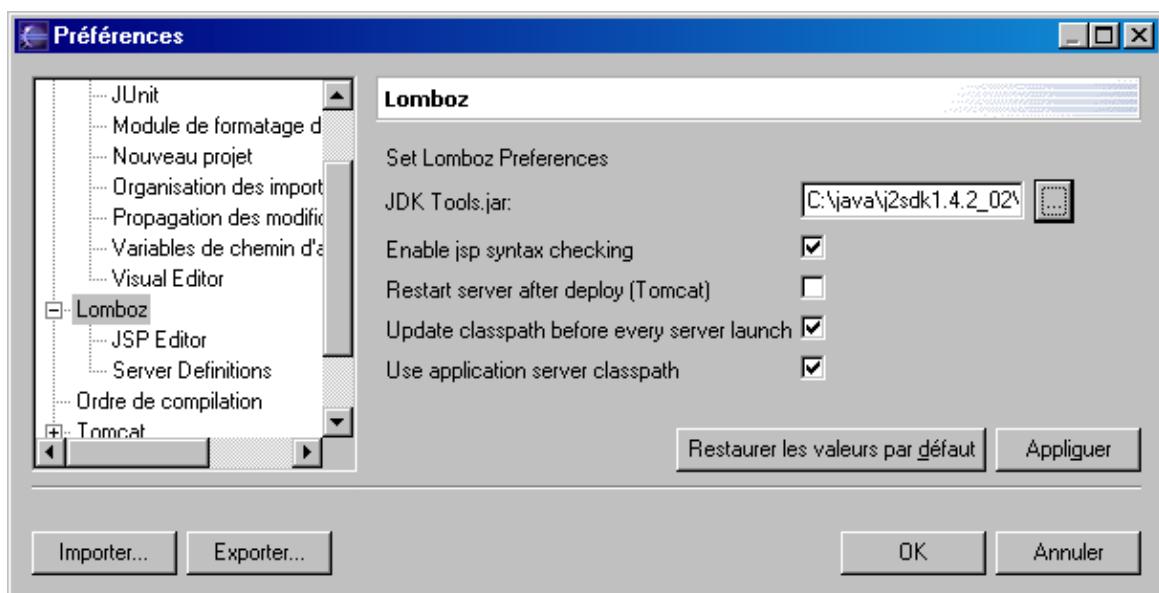


Cliquez sur la case à cocher « Autre / Lomboz Actions » puis sur le bouton « OK ». Un bouton supplémentaire apparaît dans la barre d'outils : . Il permet d'afficher la vue « Lomboz ».

Dans les préférences, sélectionnez « Java / Nouveau projet », puis cliquez sur le bouton radio « Dossiers » en conservant les noms des dossiers « src » et « bin ».

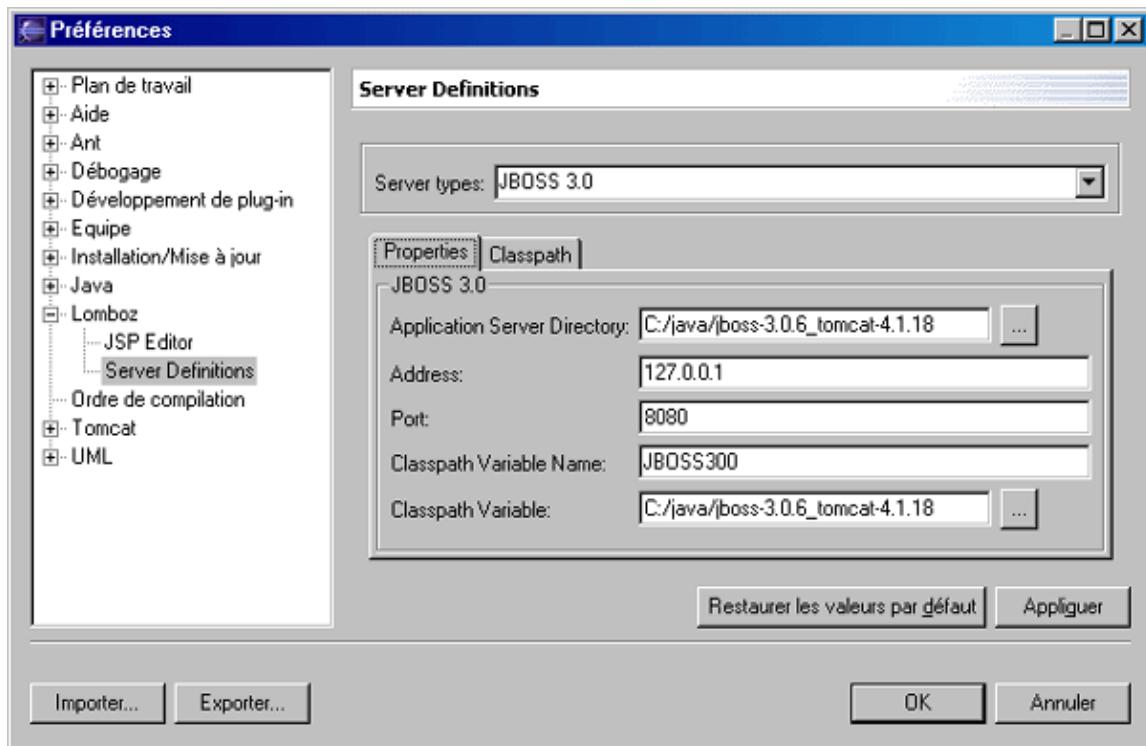


Toujours dans les préférences, il faut sélectionner « Lomboz » et vérifier que le chemin désignant le fichier tools.jar pointe bien vers le fichier \$JAVA\_HOME/lib/tools.jar.



Sélectionnez « Lomboz / Server Definitions » puis sélectionnez le ou les types de serveur utilisés : dans l'exemple de cette section, « JBOSS 3.0 » et « Apache Tomcat v4.1.0 ».

Il faut donc sélectionner le type « JBOSS 3.0 » et modifier les propriétés pour refléter celle du système utilisé notamment « Application server directory » et « Classpath variable » qui doivent pointer vers le répertoire d'installation de JBoss. Il suffit ensuite de cliquer sur le bouton « Appliquer ».



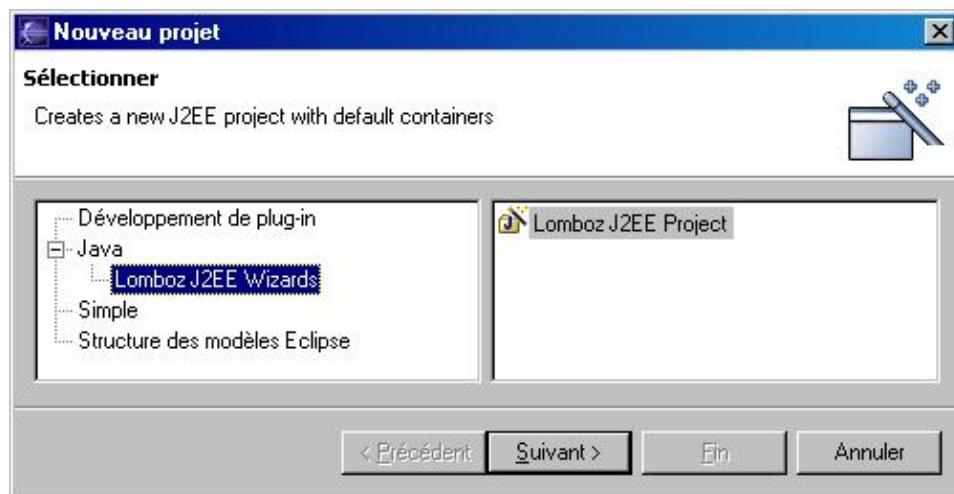
Pour s'assurer de la bonne configuration, il suffit de cliquer sur l'onglet de « Classpath » et de vérifier qu'aucune bibliothèque n'est erronée.

Il faut sélectionner le type de serveur « Apache Tomcat v4.1.0 » et faire de même.

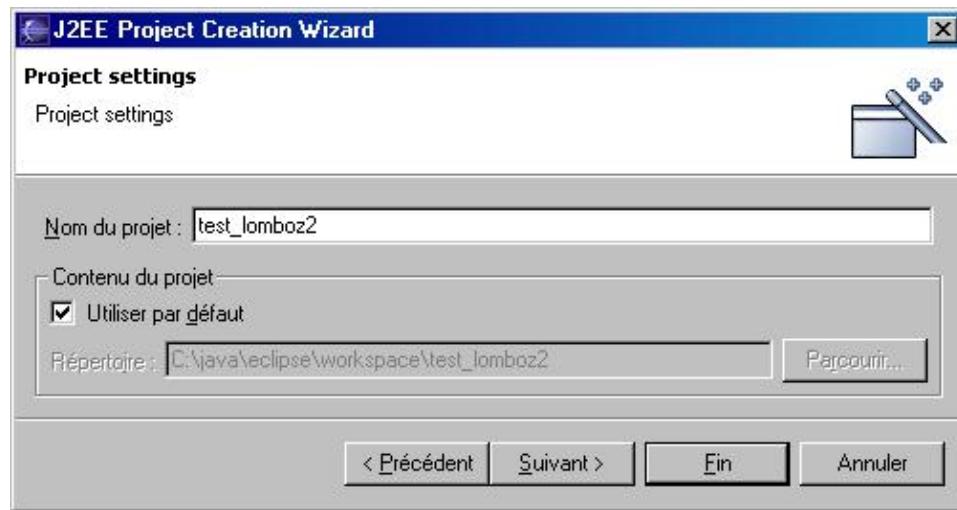
Pour valider les modifications, il suffit de cliquer sur le bouton « OK »

### 20.3.2. Création d'un nouveau projet

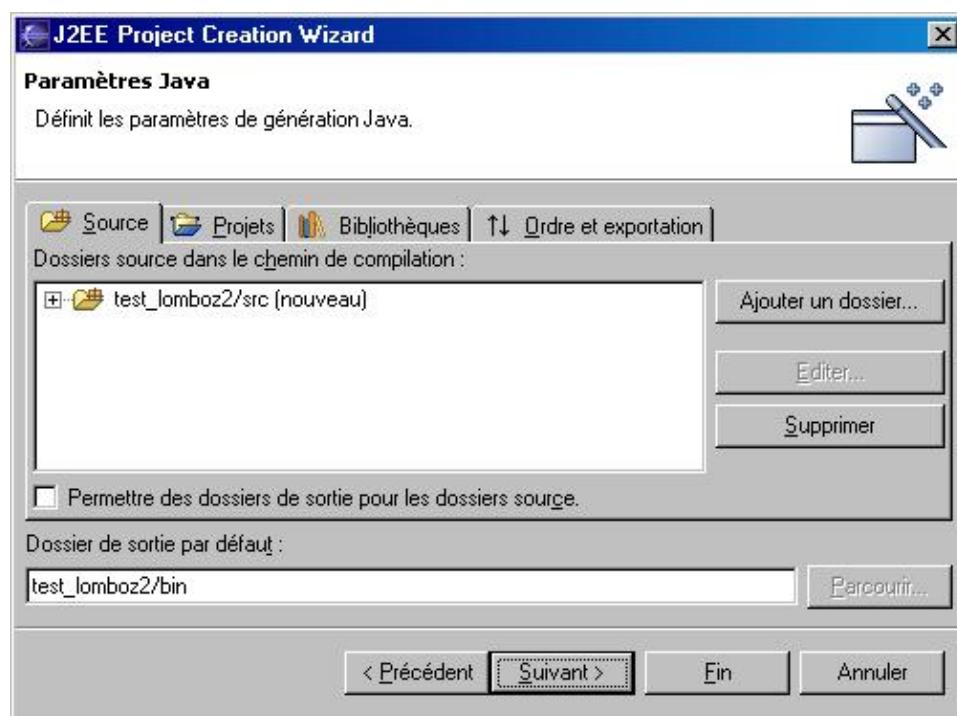
Pour créer un nouveau projet « Lomboz », il suffit de créer un nouveau projet de type « Java / Lomboz J2EE Wizard / Lomboz J2EE Project »



Cliquez sur le bouton « Suivant » et saisissez le nom du projet



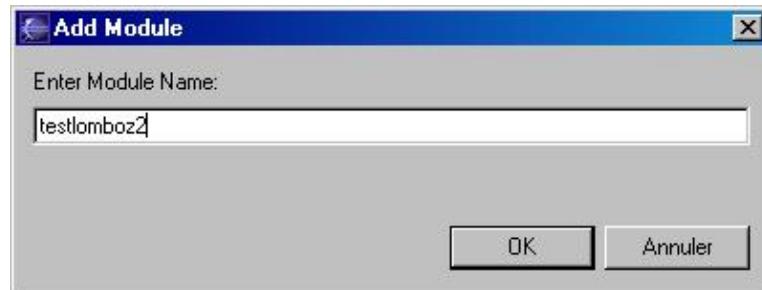
Cliquez sur le bouton « Suivant » et s'assurer que les répertoires sources et de sortie sont désignés respectivement par les répertoires "src" et "bin".



Cliquez sur le bouton « Suivant ». Sur l'onglet « Web Modules », il faut ajouter un nouveau module.



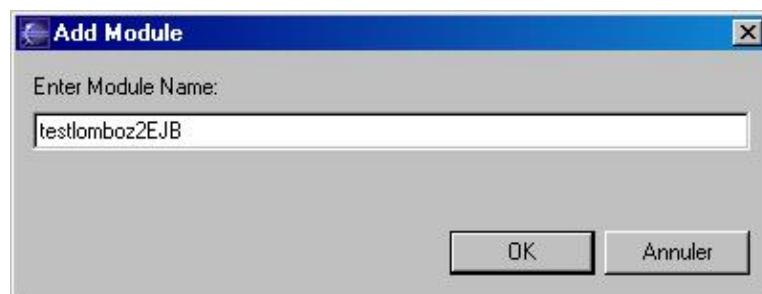
Cliquez sur le bouton « Add ... ».



Saisissez le nom et cliquez sur le bouton « OK ».

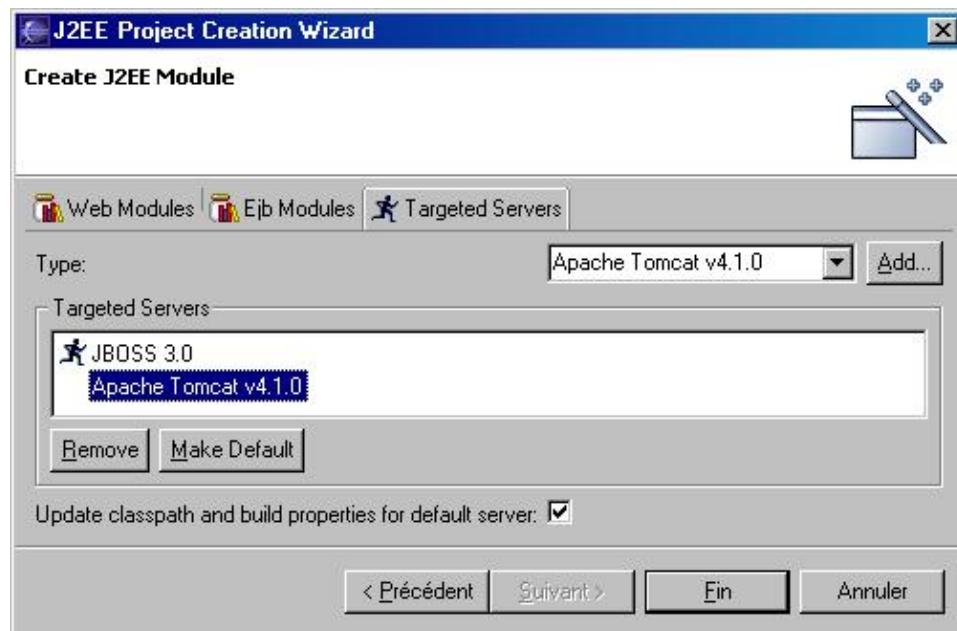
Sur l'onglet « EJB Modules », il est possible d'ajouter au projet un ou plusieurs modules qui vont contenir des EJB.

Le principe pour ajouter un module EJB est identique à celui utilisé pour un module web : cliquer sur le bouton « Add », saisir le nom du module dans la boîte de dialogue et cliquer sur le bouton « OK ».



L'ajout d'autres modules web ou EJB est toujours possible après la création du projet en utilisant l'assistant de création de nouvelles entités.

Sur l'onglet « Targeted Servers », il faut sélectionner le type de serveur à utiliser et cliquer sur le bouton Add . Dans l'exemple de cette section, il faut ajouter les serveurs « JBOSS 3.0 » et « Apache Tomcat v4.1.0 ».



Pour créer le projet, il suffit de cliquer sur le bouton « Fin ».

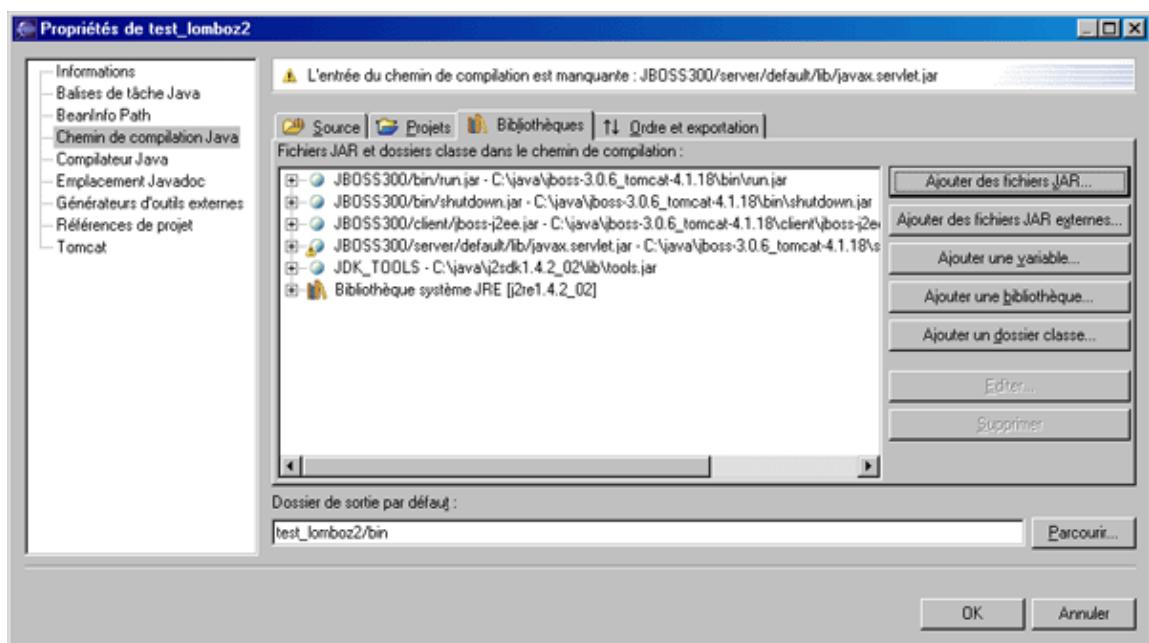


Lors de l'affichage de ce message, il suffit de cliquer sur le bouton « Oui » pour ouvrir la perspective « Java ».

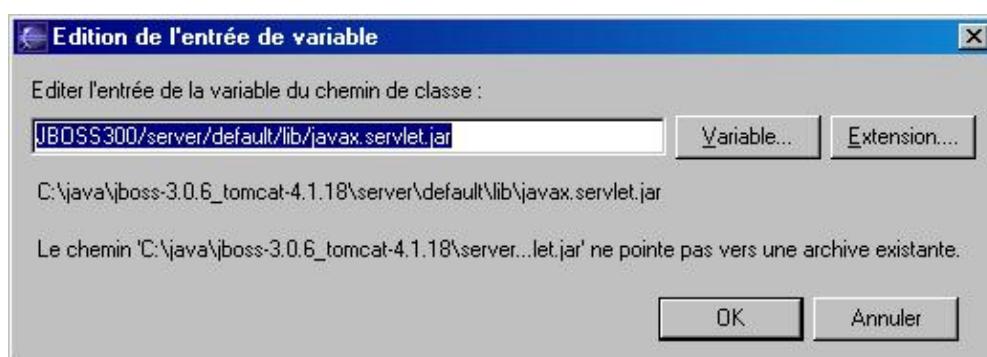
Des erreurs seront signalées dans la vue « Taches » car le chemin désigné pour le fichier servlet.jar est erroné.

	Description
✗	Le projet n'a pas été compilé en raison d'erreurs dans le chemin de classe (incomplet ou impliqué dans un cycle).
✗	Bibliothèque requise manquante : 'C:\java\jboss-3.0.6_tomcat-4.1.18\server\all\lib\javax.servlet.jar'.

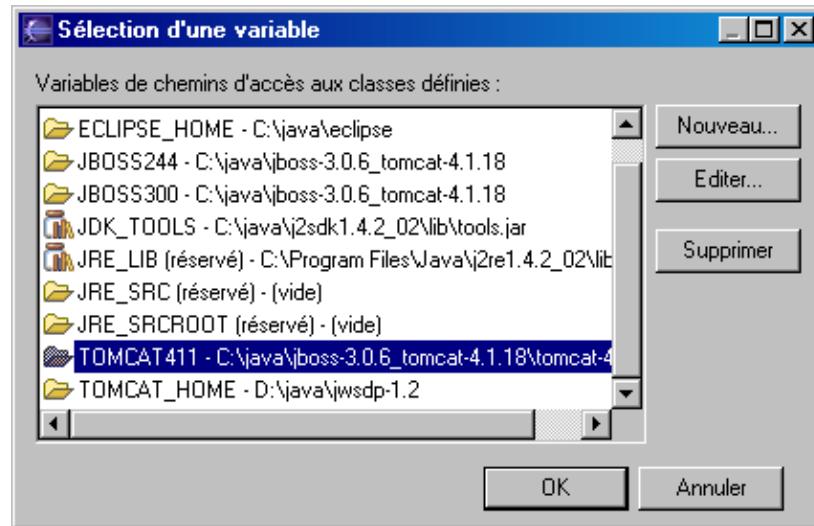
Pour corriger le problème, il faut modifier le chemin de la bibliothèque dans les propriétés du projet.



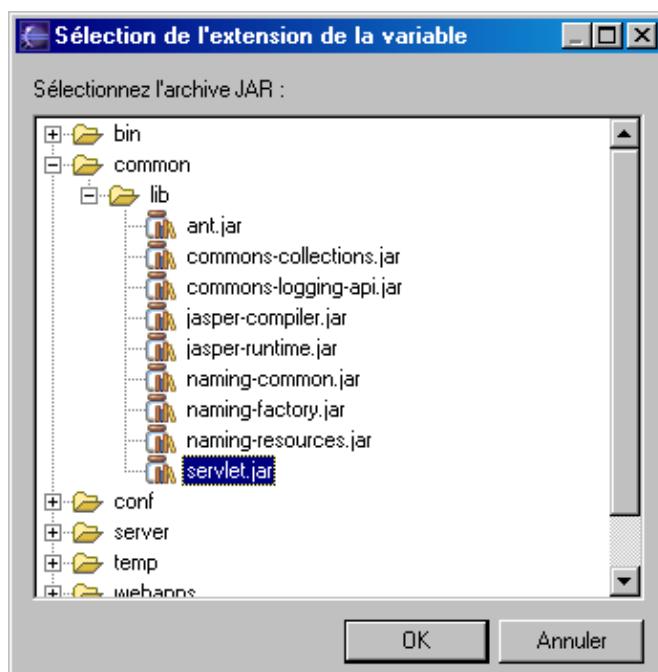
Il faut sélectionner la ligne correspondant au fichier servlet.jar (celle contenant une petite icône attention jaune) et cliquer sur le bouton « Editer ... ».



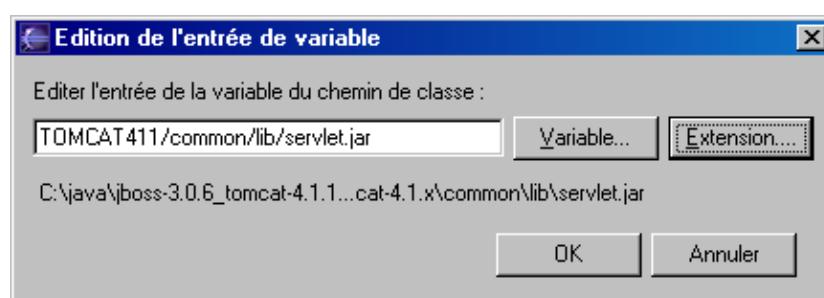
Pour faciliter la saisie du chemin il est possible de cliquer sur le bouton « Variable » pour sélectionner « TOMCAT411 » puis de cliquer sur le bouton « OK »



Pour faciliter la saisie du reste du chemin, il faut cliquer sur le bouton « Extension », sélectionner le chemin du fichier servlet.jar et cliquer sur le bouton « OK ».



La valeur associée à la variable doit avoir une valeur semblable à celle ci dessous :



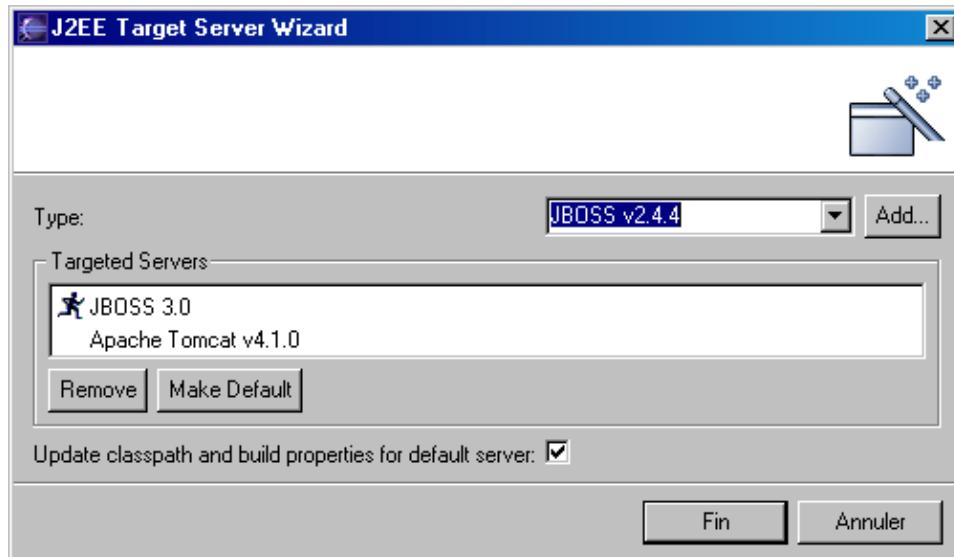
Il suffit de cliquer sur le bouton « OK » pour valider les modifications.

Cette correction n'est que temporaire : pour corriger le problème de façon définitive, il faut modifier le fichier .server correspondant à celui du serveur utilisé dans le répertoire :

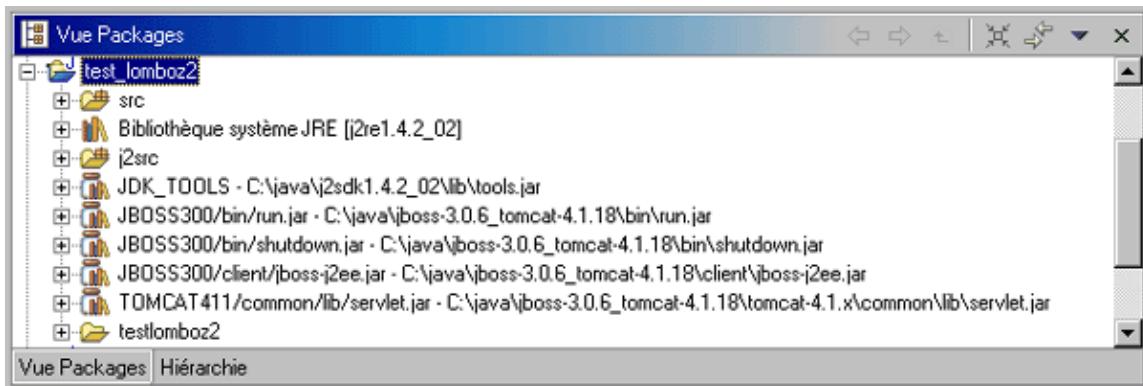
C:\java\eclipse\plugins\com.objectlearn.jdt.j2ee\servers

La ligne à modifier est celle permettant l'ajout du fichier servlet.jar. Une fois le fichier enregistré, il faut relancer Eclipse.

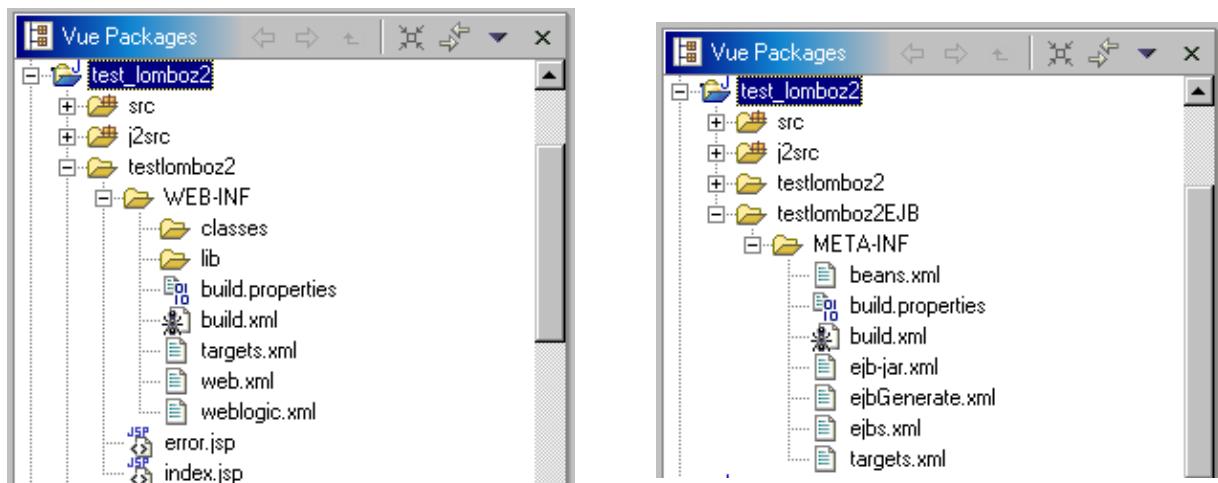
Dans la vue package, sur le module EJB, sélectionnez l'option « Lomboz J2EE/ Change default server » pour prendre en compte la modification puis cliquez sur le bouton "Fin".



La structure du projet est la suivante dans la vue package :



Les modules web et EJB contiennent déjà quelques fichiers de configuration.



# Chapitre 21

L'utilisation de XML est omniprésente dans la conception, le développement et la mise en œuvre d'applications que ce soit pour configurer ou déployer une application, échanger ou stocker des données.

Ce chapitre contient plusieurs sections :

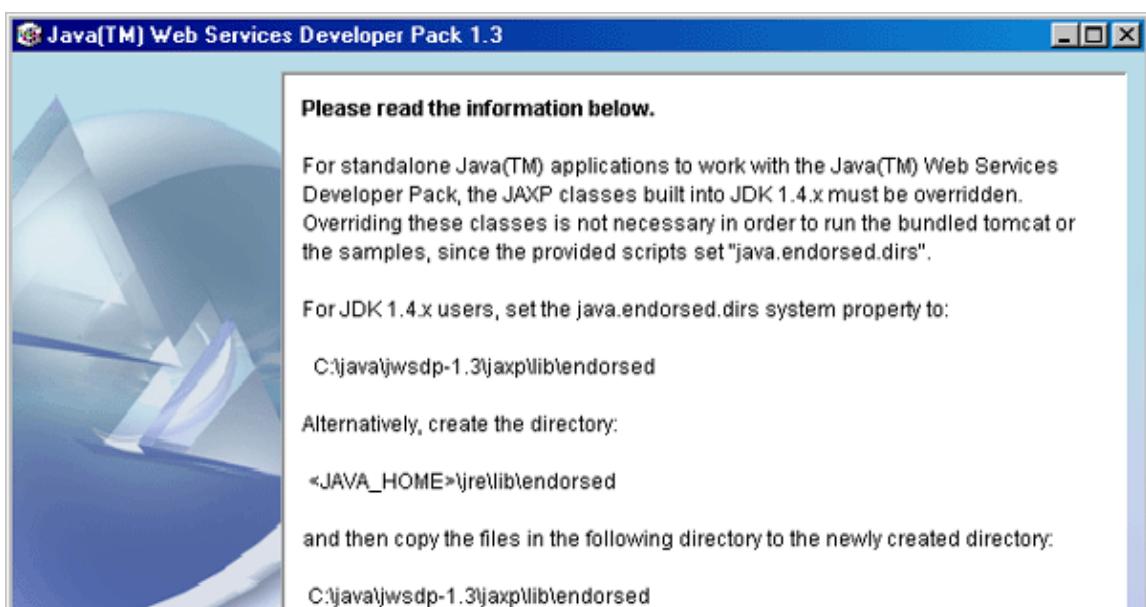
- ◆ [JAXB et Eclipse](#)
- ◆ [Le plug-in WTP pour utiliser XML](#)

### 21.1. JAXB et Eclipse

JAXB est l'acronyme de Java Architecture for XML Binding. Le but de l'API et des spécifications JAXB est de faciliter la manipulation d'un document XML en générant un ensemble de classes qui fournissent un niveau d'abstraction plus élevé que l'utilisation de JAXP (SAX ou DOM). Avec ces deux API, toute la logique de traitements des données contenues dans le document est à écrire.

JAXB au contraire fournit un outil qui analyse un schéma XML et génère à partir de ce dernier un ensemble de classes qui vont encapsuler les traitements de manipulation du document.

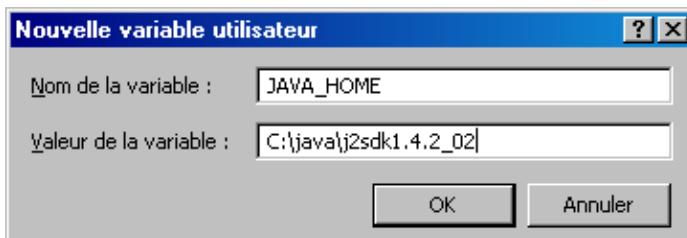
Pour pouvoir utiliser JAXB, il est nécessaire d'installer le Java Web Services Developer Pack 1.3. Dans le cas de l'utilisation d'un JDK 1.4, attention de bien suivre les instructions fournies à la fin de l'installation.



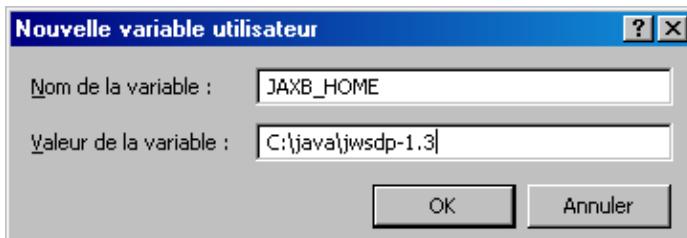
Il est aussi nécessaire de définir deux variables d'environnement système (sous Windows 2000 : menu "Paramètres / Panneau de configuration / Système", sélectionner l'onglet "avancé", puis cliquer sur le bouton

"variables d'environnement" ) :

JAVA\_HOME dont la valeur doit correspondre au répertoire d'installation du JDK :



JAXB\_HOME dont la valeur doit correspondre au répertoire dans lequel le JWSDP est installé :



Il existe deux moyens d'utiliser JAXB avec Eclipse :

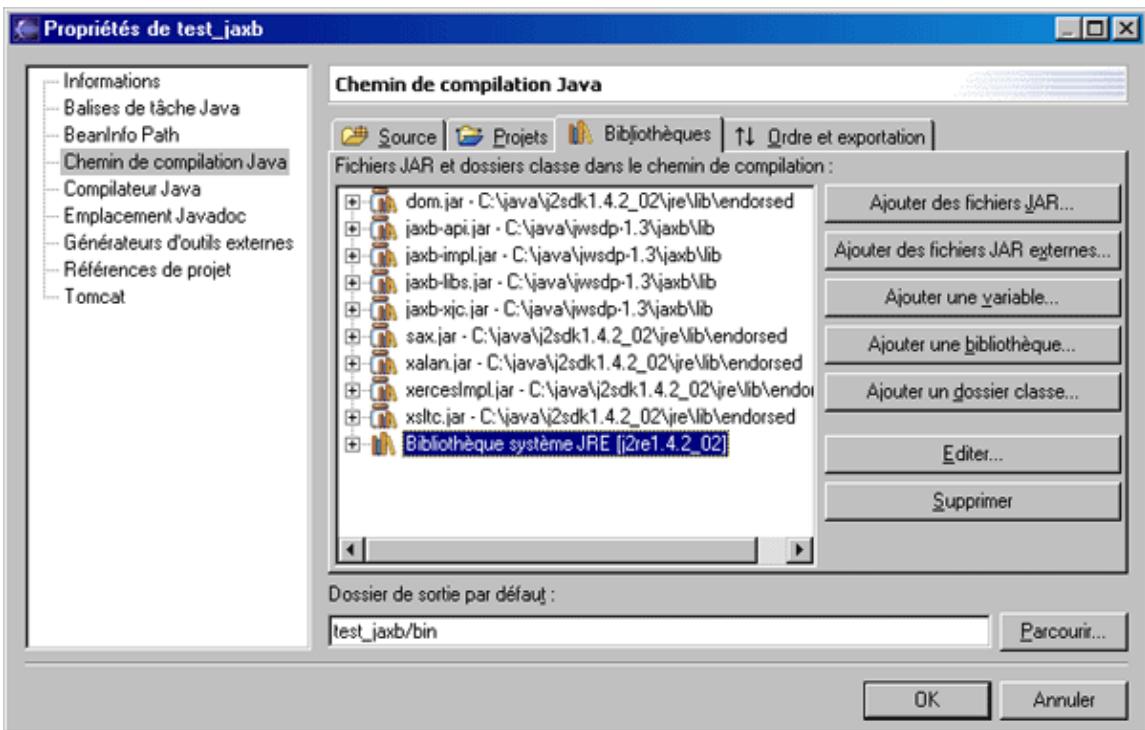
- Créer et configurer une tâche d'exécution qui va lancer la génération des classes Java via l'outil de JAXB
- Exécuter Ant fourni avec le JWSDP en tant qu'outil externe dans Eclipse

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
JAXB	1.3

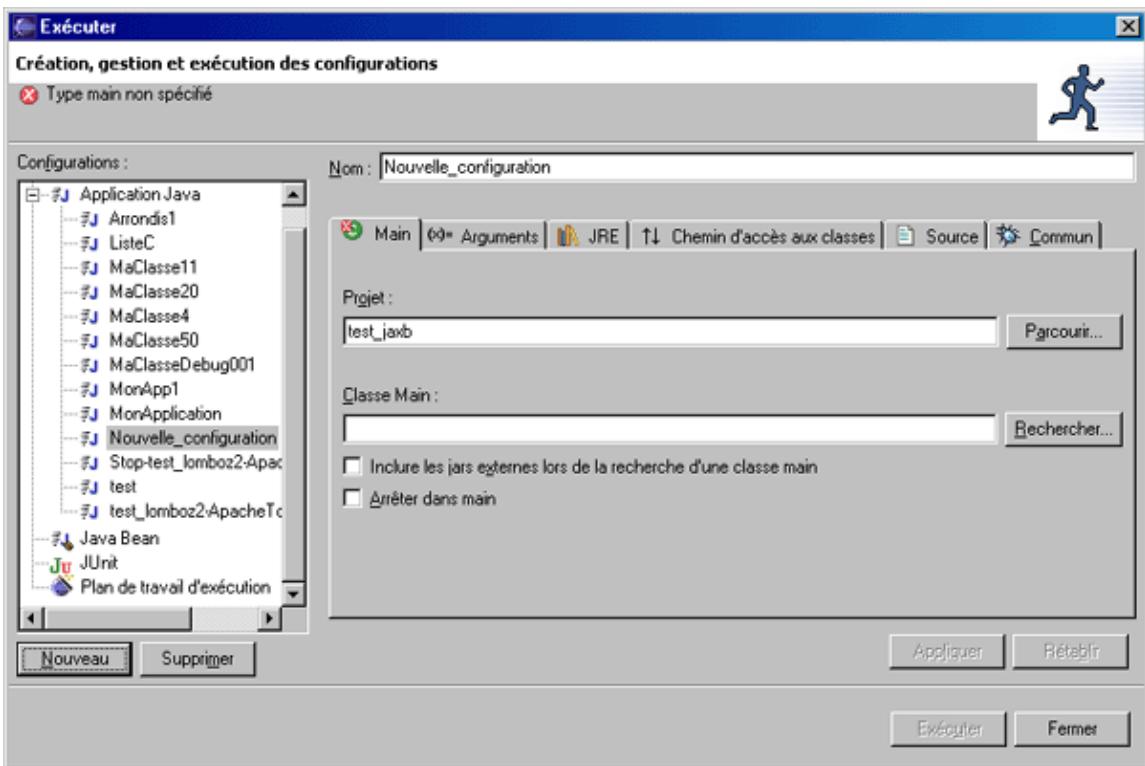
### 21.1.1. Crée et configurer une tâche d'exécution pour JAXB

Il faut créer un nouveau projet, par exemple nommé test\_jaxb et modifier son chemin de compilation (dans les propriétés du projet) pour ajouter plusieurs fichiers .jar :

- Les fichiers jar ajoutés dans le répertoire JAVA\_HOME\jre\lib\endorsed lors de l'installation du JWSDP
- Les fichiers jar contenus dans le répertoire JAXB\_HOME\jaxb\lib



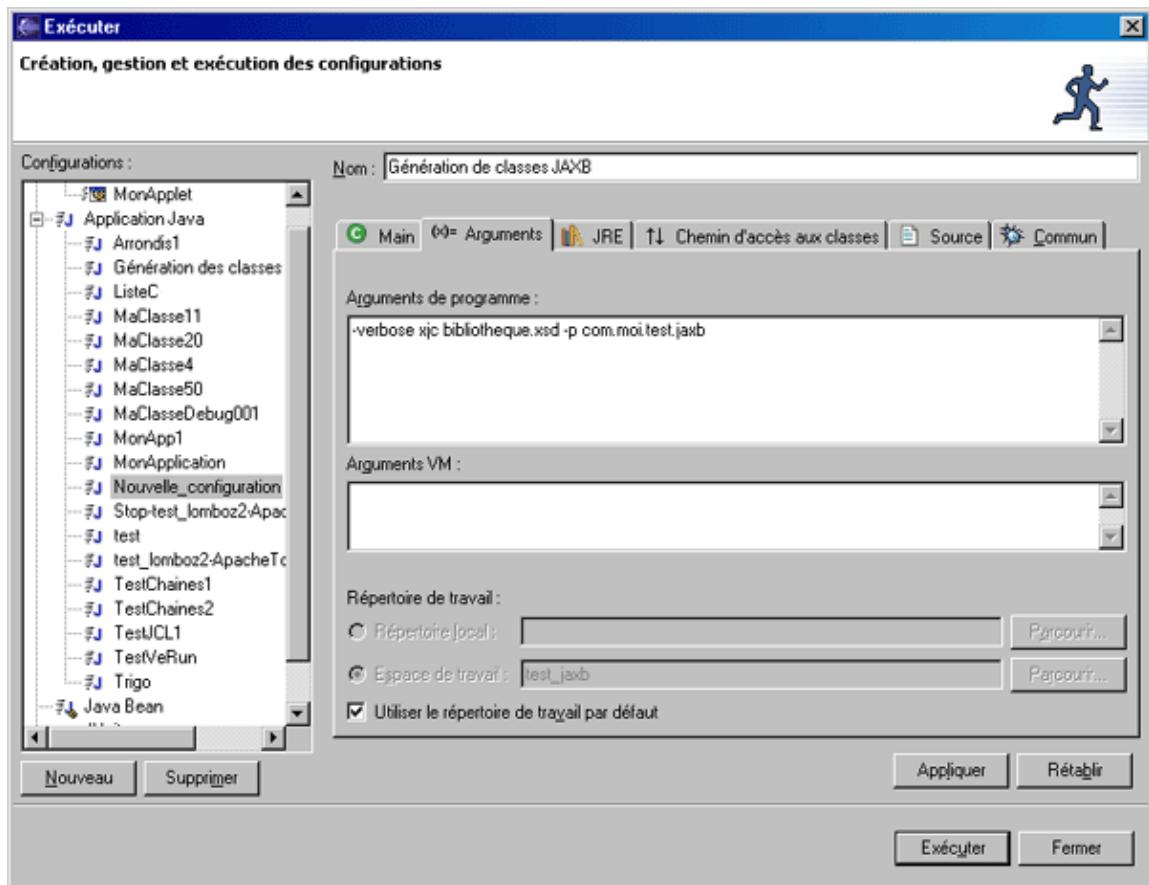
Il faut ensuite créer une nouvelle tâche d'exécution en utilisant l'option du menu « Exécuter/Exécuter ... » et en cliquant sur le bouton Nouveau .



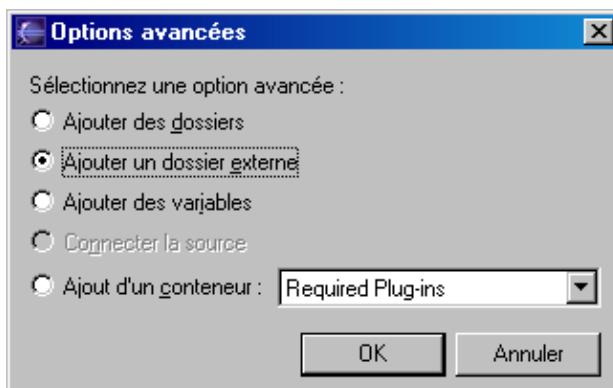
Il faut changer le nom, par exemple « Génération des classes JAXB » et saisir dans le champ Classe Main la valeur « LauncherBootstrap ».

Sur l'onglet « Arguments », dans la zone de saisie "Arguments de programme" saisir (bibliotheque.xsd est le nom du fichier qui contient le schéma du document XML) :

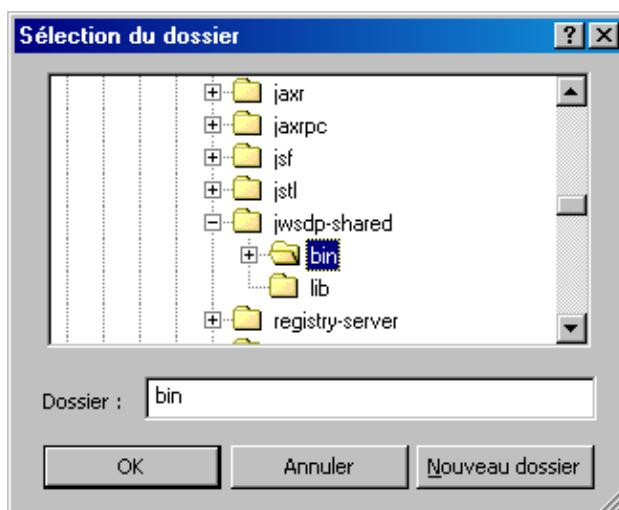
```
-verbose xjc bibliotheque.xsd -p com.moi.test.jaxb
```



Sur l'onglet « Chemin d'accès aux classes », décochez « Utiliser le chemin d'accès aux classes par défaut » puis cliquez sur le bouton « Avancées »



Cliquez sur le bouton « Ajouter un dossier externe » puis sur le bouton "Ok"



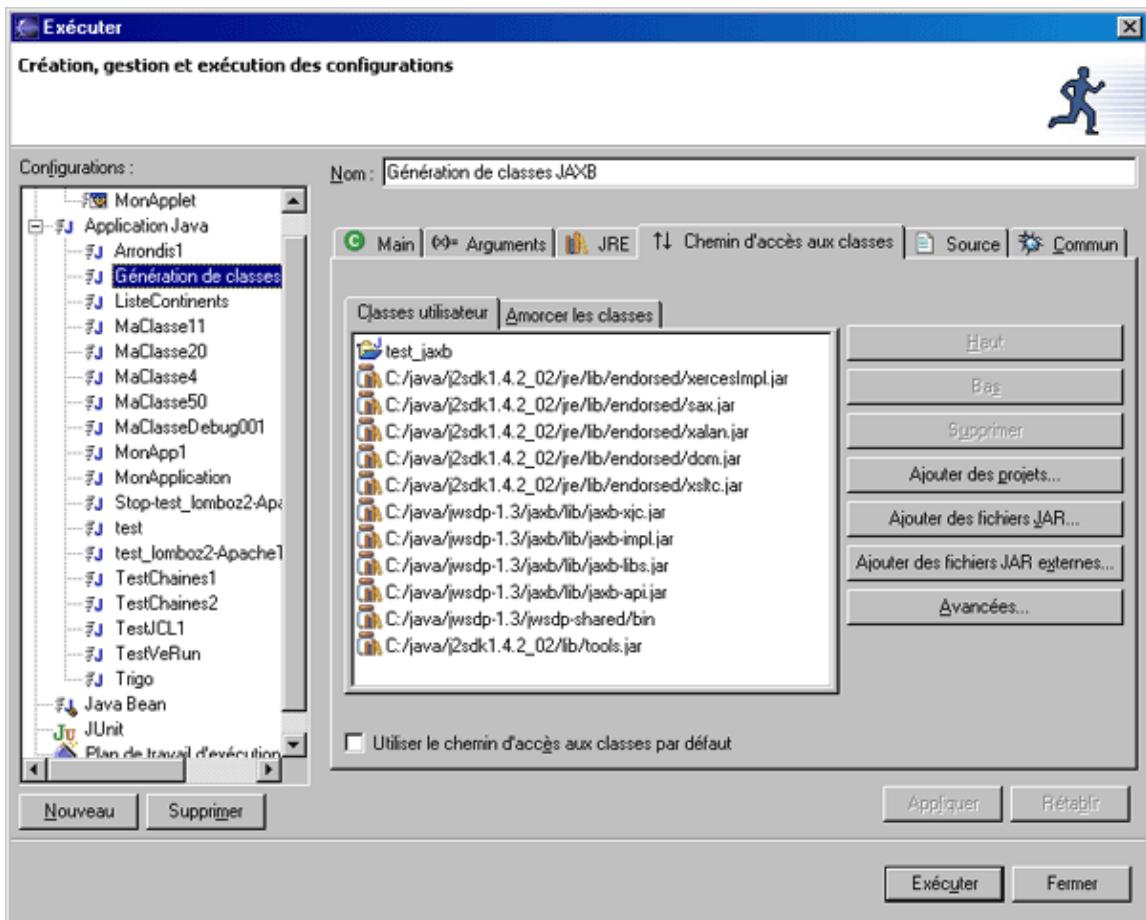
Selectionnez le répertoire %JWSDP%/jwsdp-shared/bin et cliquez sur le bouton « OK »

Cliquez sur le bouton « Fermer »



Cliquez sur le bouton « Oui »

Il faut aussi ajouter le fichier tools.jar présent dans le répertoire lib du JDK :

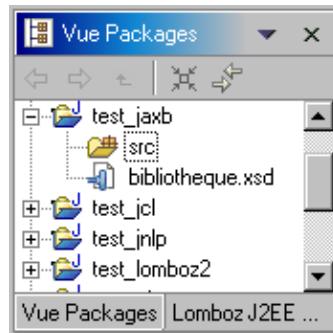


Si ce fichier n'est pas ajouté, une exception est levée lors de l'exécution :

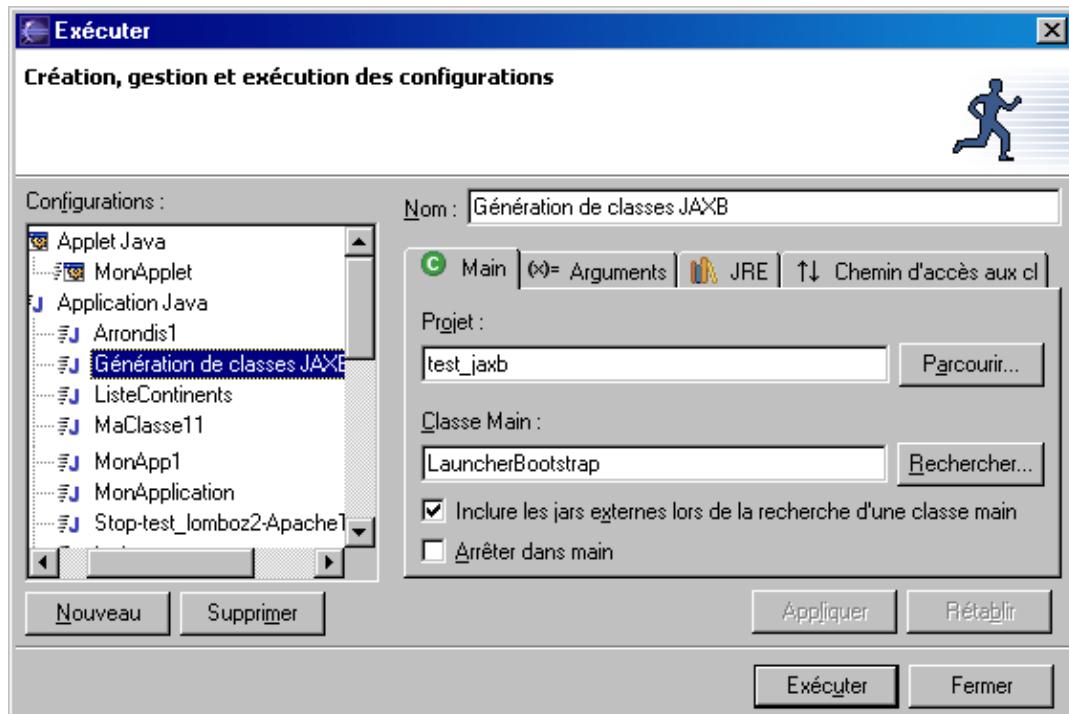
exemple :

```
file:C:/java/jwsdp-1.3/jwsdp-shared/bin/launcher.xml:329: java.io.IOException: Could not find Java(TM) 2 SDK
This application cannot run using the Java(TM) 2 JRE. It requires the full SDK.
: org.apache.commons.launcher.LaunchTask.execute(LaunchTask.java:728)
: org.apache.tools.ant.Task.perform(Task.java:341)
: org.apache.tools.ant.Target.execute(Target.java:309)
```

La configuration est maintenant terminée, il faut un schéma qui décrit le document XML directement à la racine du projet, par exemple bibliothèque.xsd.

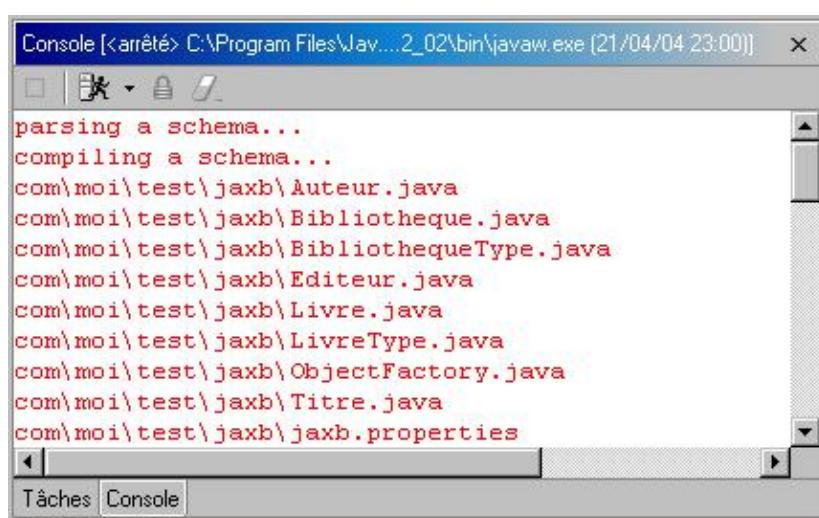


Pour lancer la génération des classes Java par JAXB, il faut utiliser l'option du menu « Exécuter/Exécuter ... » et sélectionner « Application Java/Generation de classe JAXB »

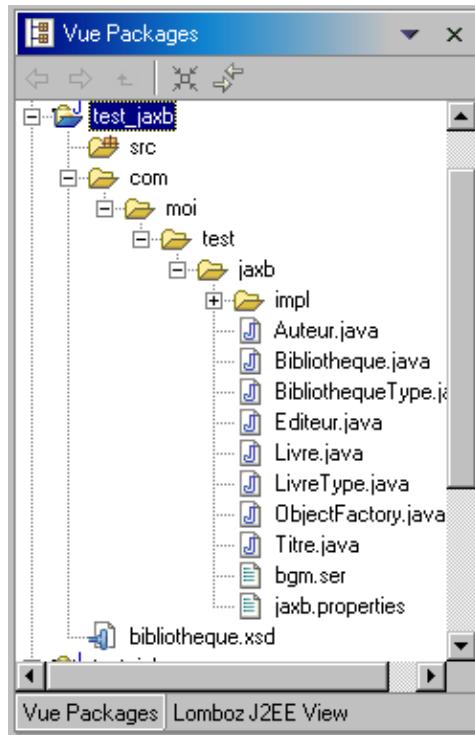


La boîte de dialogue réaffiche les paramètres précédemment enregistrés : il suffit de cliquer sur le bouton « Exécuter ».

Les messages issus des traitements sont affichés dans la console.

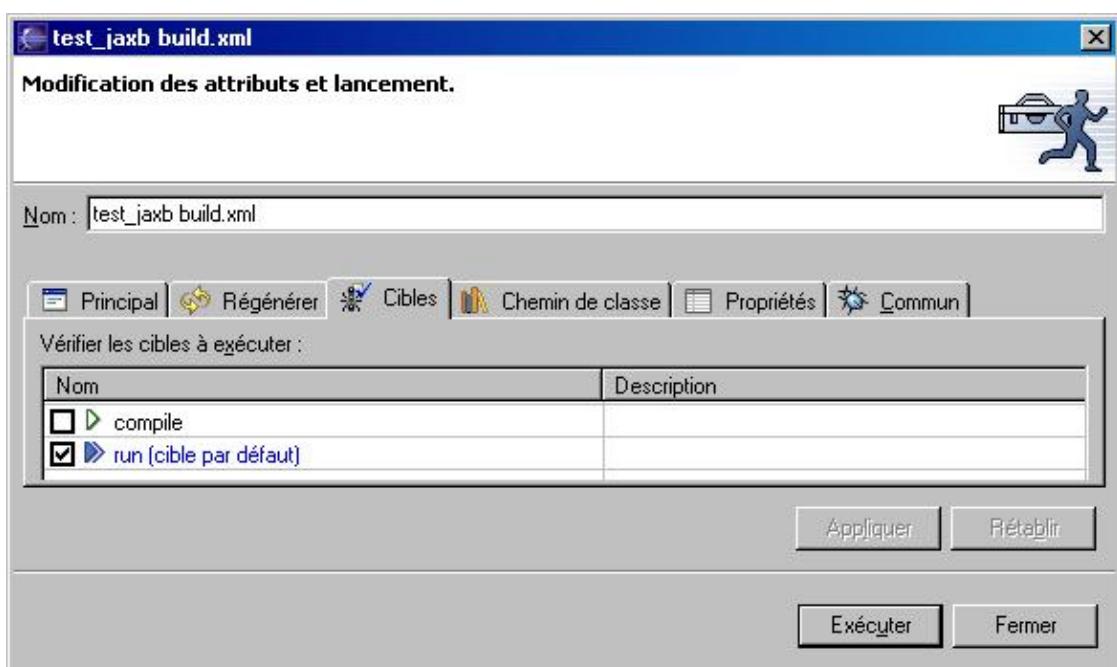


Dans la vue « Packages », sélectionner le projet test\_jaxb et utiliser l'option « Régénérer » du menu contextuel pour rafraîchir le contenu du projet et voir les fichiers générés lors des traitements.



### 21.1.2. Exécuter Ant en tant qu'outil externe

L'exécution de JAXB se fait normalement en utilisant une tâche Ant dédiée.



Malheureusement, il y a une incompatibilité entre la version de Ant fournie avec Eclipse et celle requise par JAXB. L'exécution d'un build.xml en utilisant Ant fourni avec Eclipse échoue systématiquement :

```
Résultat de l'exécution :

Buildfile: C:\java\eclipse\workspace\test_jaxb\build.xml
compile:
[echo] Compiling the schema external binding file...
[xjc] Compiling file:/C:/java/eclipse/workspace/test_jaxb/bibliotheque.xsd
[xjc] [WARNING] Unable to validate your schema. Most likely, the JVM has
loaded an incompatible XML parser implementation. You should fix this
```

```

before relying on the generated code. Please see the release notes
for details.
[xjc] unknown location
[xjc]
[xjc] BUILD FAILED: file:C:/java/eclipse/workspace/test_jaxb/build.xml:31:
      unable to parse the schema. Error messages should have been provided
Total time: 5 seconds

```

Pour résoudre ce problème et utiliser Ant, il faut l'exécuter en tant qu'outil externe.

Il faut définir le fichier build.xml qui va contenir les différents traitements à exécuter par Ant.

#### Résultat de l'exécution :

```

<?xml version="1.0"?>
<project basedir=". " default="compile">
  <property name="jwsdp.home" value="C:\java\jwsdp-1.3" />
  <property name="java.home" value="C:\java\j2sdk1.4.2_02" />

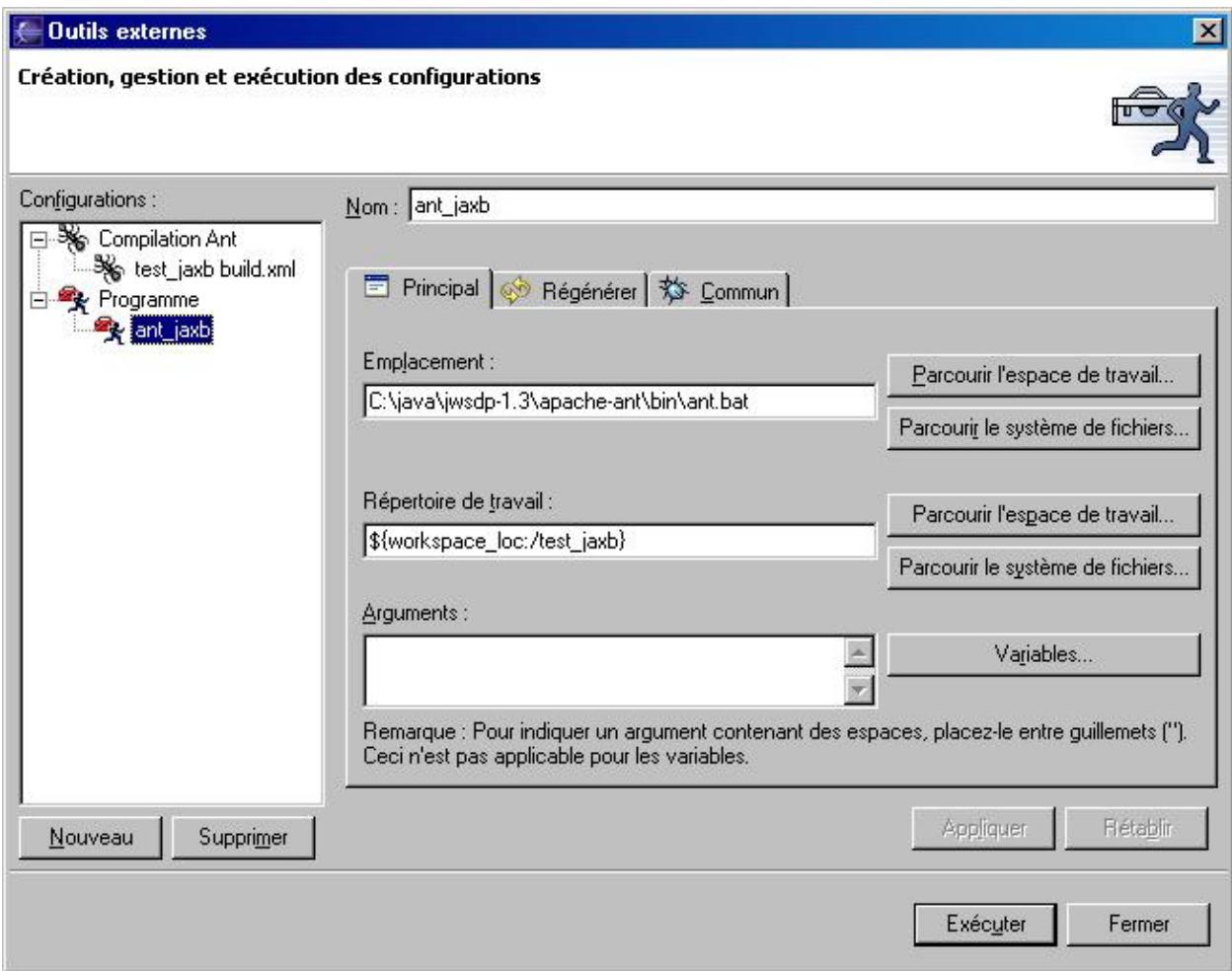
  <path id="classpath">
    <pathelement path=". " />
    <fileset dir="${java.home}" includes="jre/lib/endorsed/*.jar" />
    <fileset dir="${jwsdp.home}" includes="jaxb/lib/*.jar" />
    <fileset dir="${jwsdp.home}" includes="jwsdp-shared/lib/*.jar" />
    <fileset dir="${jwsdp.home}" includes="jaxp/lib/**/*.jar" />
  </path>

  <taskdef name="xjc" classname="com.sun.tools.xjc.XJCTask">
    <classpath refid="classpath" />
  </taskdef>

  <target name="compile">
    <echo message="Generation des classes Java a partir du schema ..."/>
    <xjc schema="bibliotheque.xsd" target=".src" package="com.moi.test.jaxb"/>
    <echo message="Compilation des sources ..."/>
    <javac srcdir=".src" destdir=". " debug="on">
      <classpath refid="classpath" />
    </javac>
  </target>
</project>

```

Il faut utiliser l'option du menu « Exécuter/ Outils externes/Outils externes ... » et cliquer sur le bouton « Nouveau ».



Sur l'onglet « Principal », il suffit de sélectionner l'emplacement du fichier ant.bat fourni avec le JWSDP, puis de cliquer sur le bouton « Appliquer », puis sur le bouton « Exécuter ».

Les informations générées par Ant au cours de l'exécution sont affichées dans la console.

```
Résultat de l'exécution :  
Buildfile: build.xml  
  
compile:  
  [echo] Generation des classes Java a partir du schema ...  
  [xjc] Compiling file:/C:/java/eclipse/workspace/test_jaxb/bibliotheque.xsd  
  [xjc] Writing output to C:\java\eclipse\workspace\test_jaxb\src  
  [echo] Compilation des sources ...  
  [javac] Compiling 44 source files to C:\java\eclipse\workspace\test_jaxb  
  
BUILD SUCCESSFUL  
Total time: 1 minute 4 seconds
```

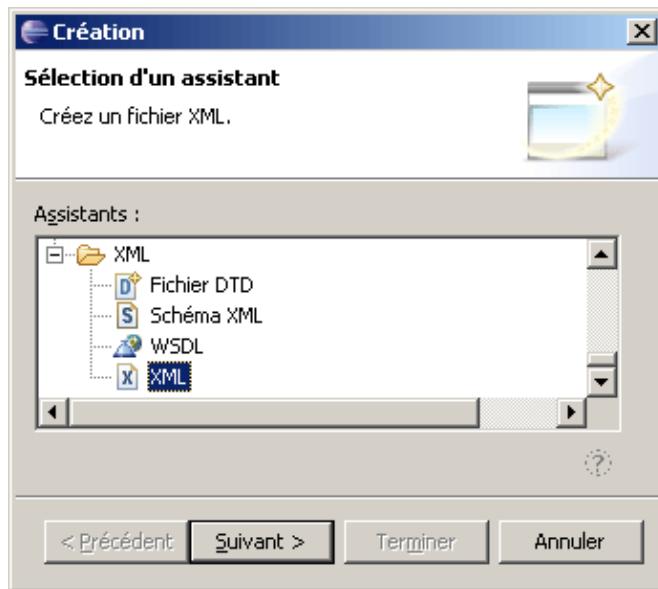
Pour voir les fichiers générés par JAXB lors de ces traitements, il suffit d'utiliser l'option « Régénérer » du menu contextuel du projet dans la vue « Packages ».

## 21.2. Le plug-in WTP pour utiliser XML

Le plug-in WTP facilite la manipulation de documents XML.

## 21.2.1. Créer un nouveau document XML

Il faut créer une nouvelle entité de type XML/XML



Cliquez sur le bouton « Suivant »

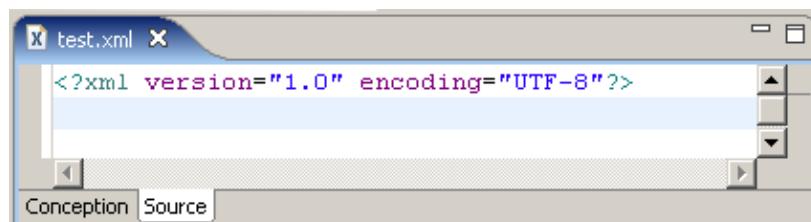


Sélectionnez « Crée un fichier XML à partir de rien » puis cliquez sur le bouton « Suivant »



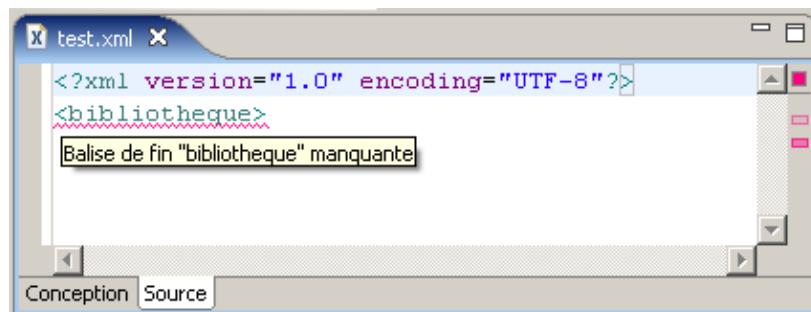
Sélectionnez le répertoire qui va contenir le fichier, saisissez son nom et cliquez sur le bouton « Terminer »

L'éditeur de documents XML s'ouvre avec le nouveau document créé.

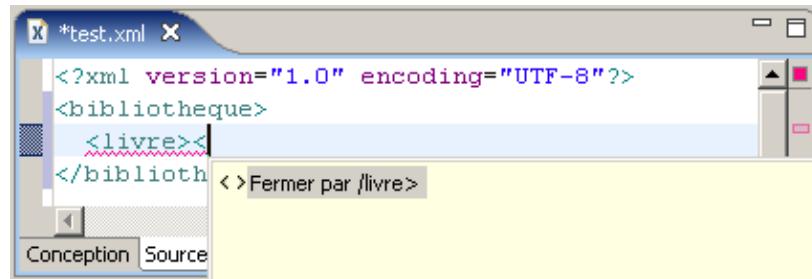


L'éditeur propose une coloration syntaxique du code.

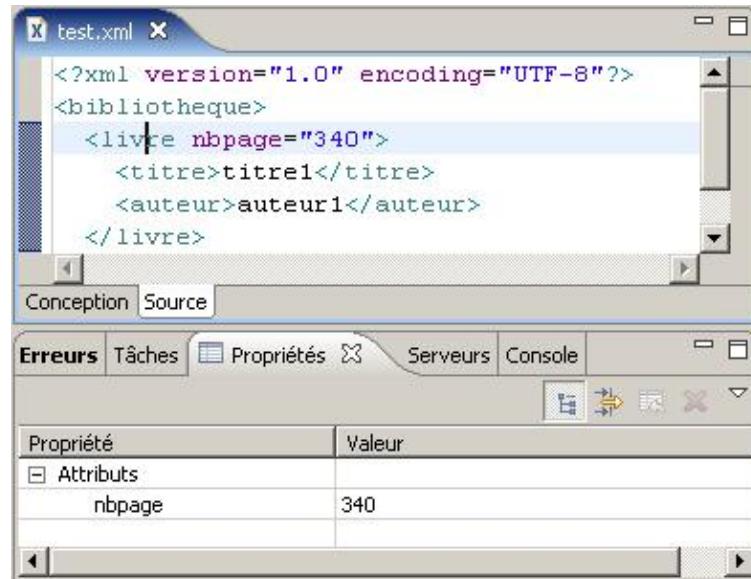
La syntaxe du code XML est vérifiée au fur et à mesure de la saisie



L'assistant de code permet de faciliter la saisie du code du document.

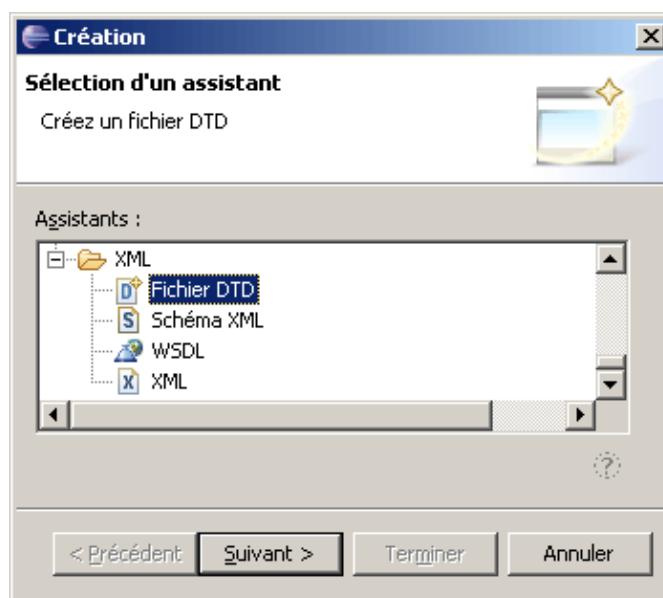


La vue « Properties » permet d'afficher les attributs du tag sur lequel est positionné le curseur dans l'éditeur de code



### 21.2.2. La création d'une DTD

Il faut créer une nouvelle entité du type « XML/Fichier DTD ».



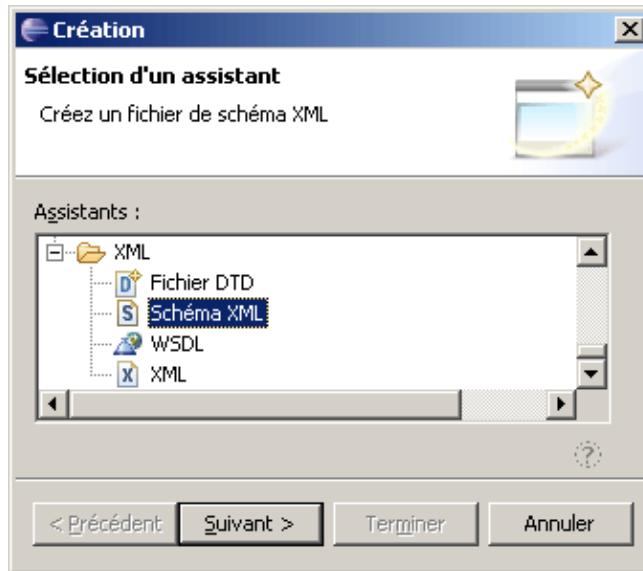
Cliquez sur le bouton "Suivant" puis sélectionnez le répertoire qui va contenir le fichier et saisissez son nom



Cliquez sur le bouton « Terminer » pour générer le fichier et l'ouvrir dans l'éditeur.

### 21.2.3. La création d'un schéma

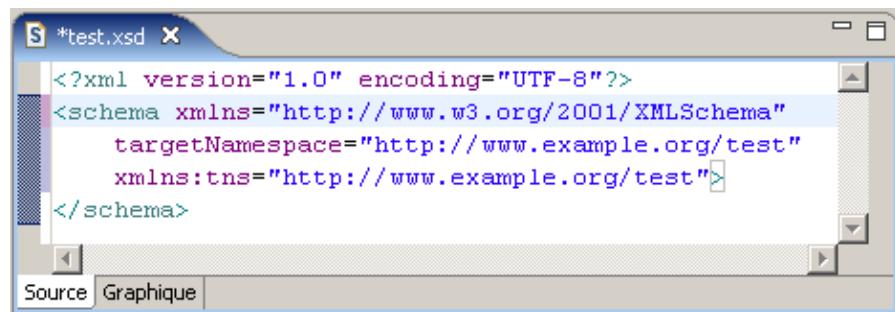
Il faut créer une nouvelle entité du type « XML/Schéma XML ».



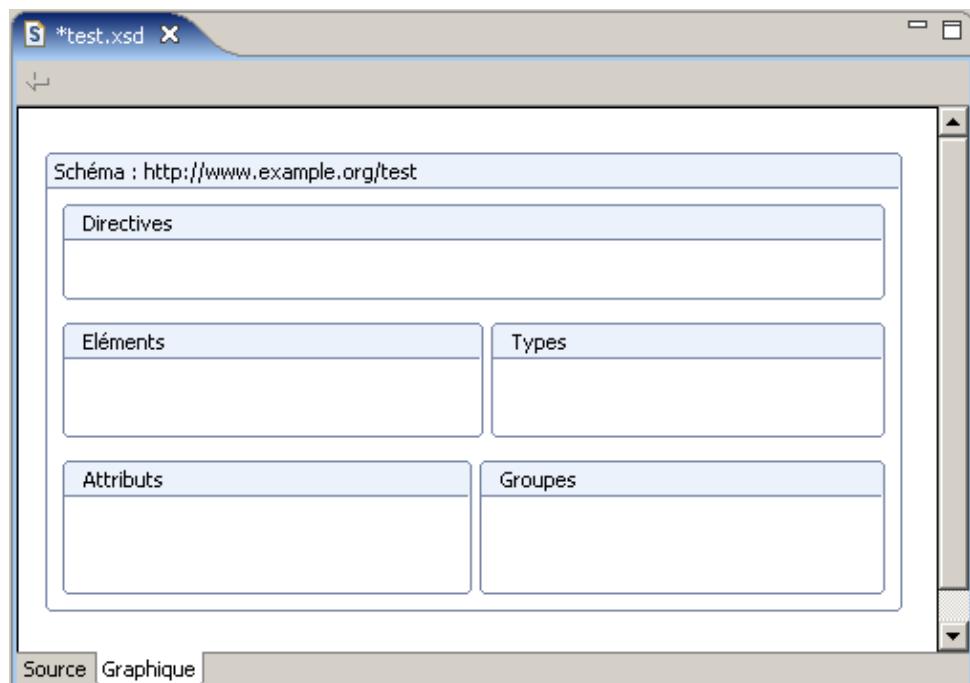
Cliquez sur le bouton "Suivant".



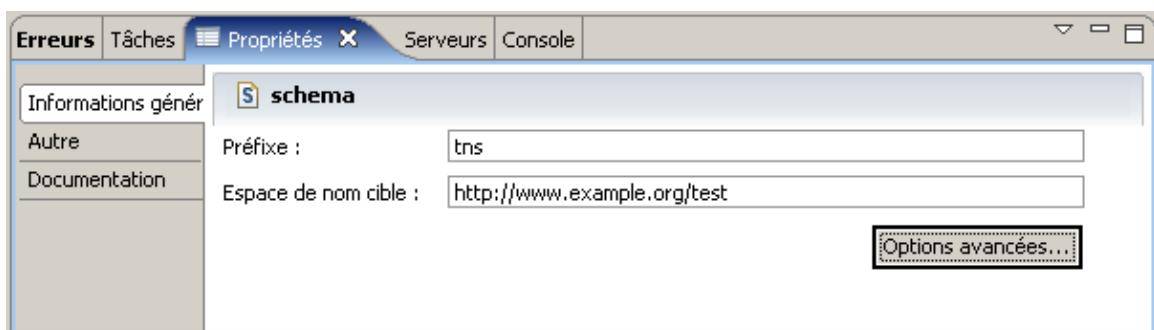
Saisissez le nom du fichier .xsd et cliquez sur le bouton "Terminer".



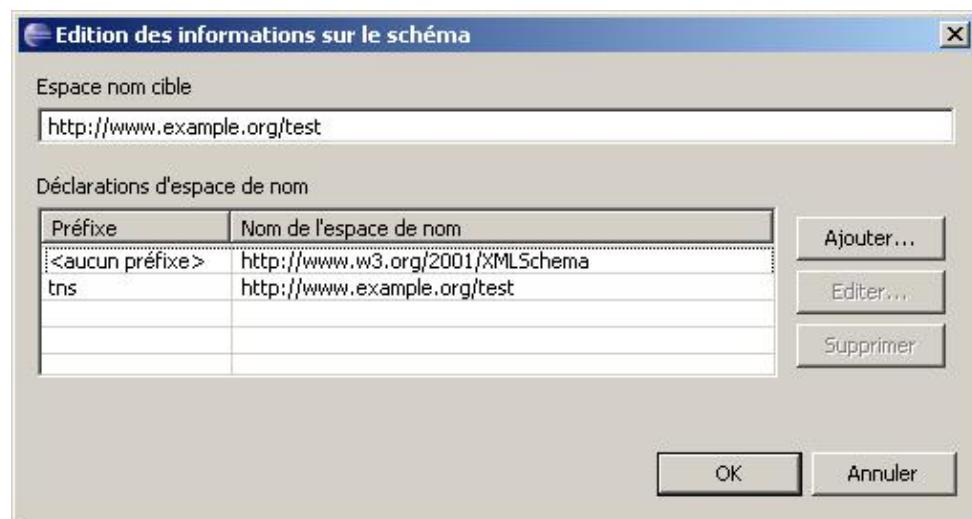
L'onglet "Graphique" permet d'offrir une vision graphique du contenu du document.



La vue propriétés permet d'afficher des informations sur le schéma.

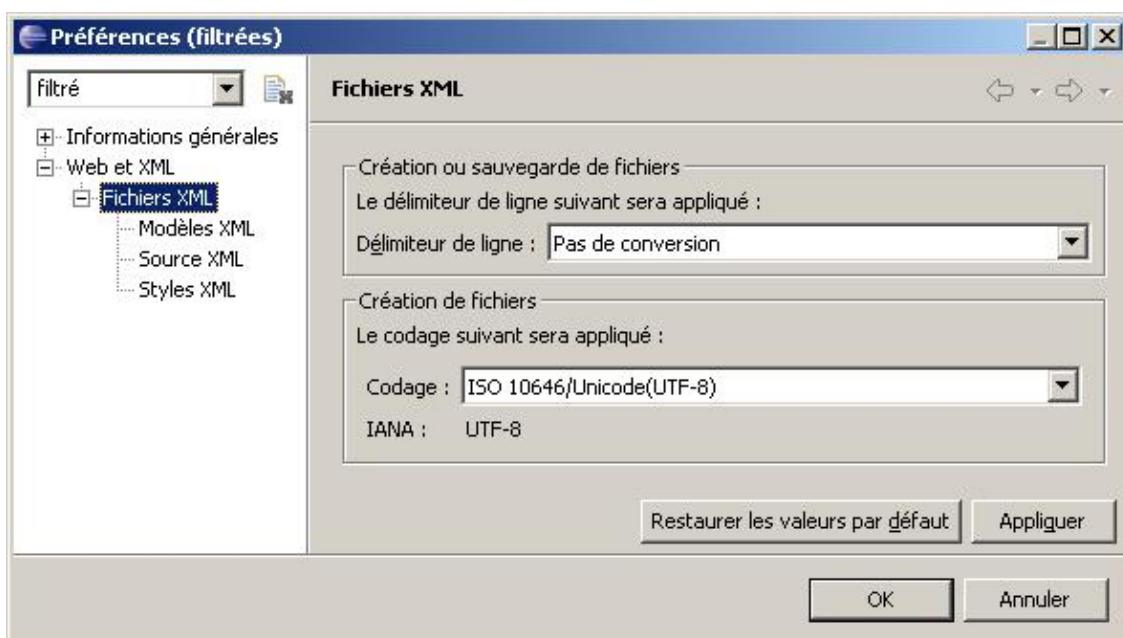


En cliquant sur le bouton "Option avancées", une boîte de dialogue s'ouvre en affichant des informations sur le schéma.

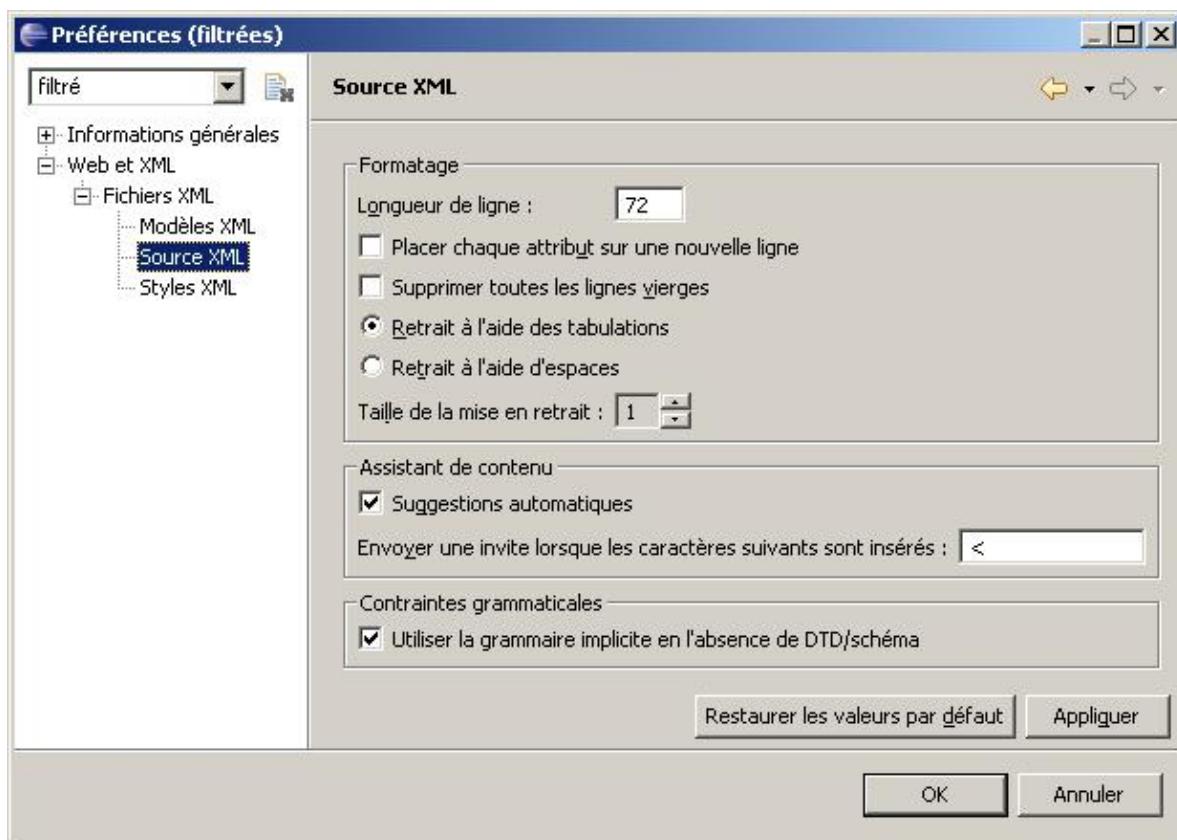


#### 21.2.4. Les préférences

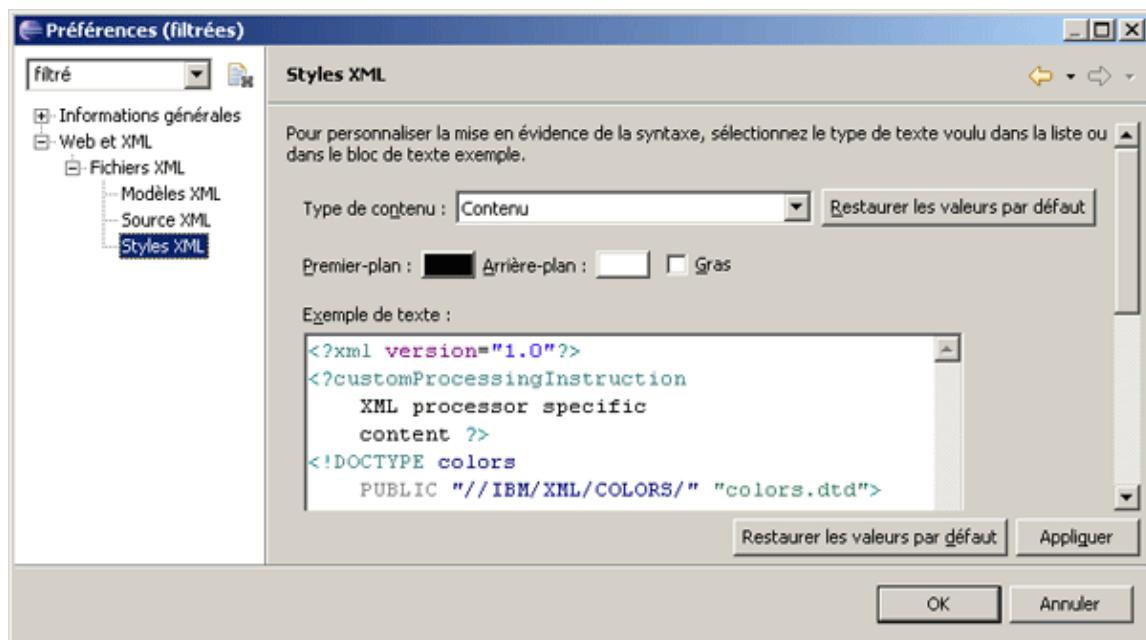
Pour accéder aux préférences concernant les fichiers XML, il est possible d'utiliser l'option « Préférences ... » du menu « Fenêtre » ou d'utiliser l'option « Préférence » du menu contextuel de l'éditeur de code XML.



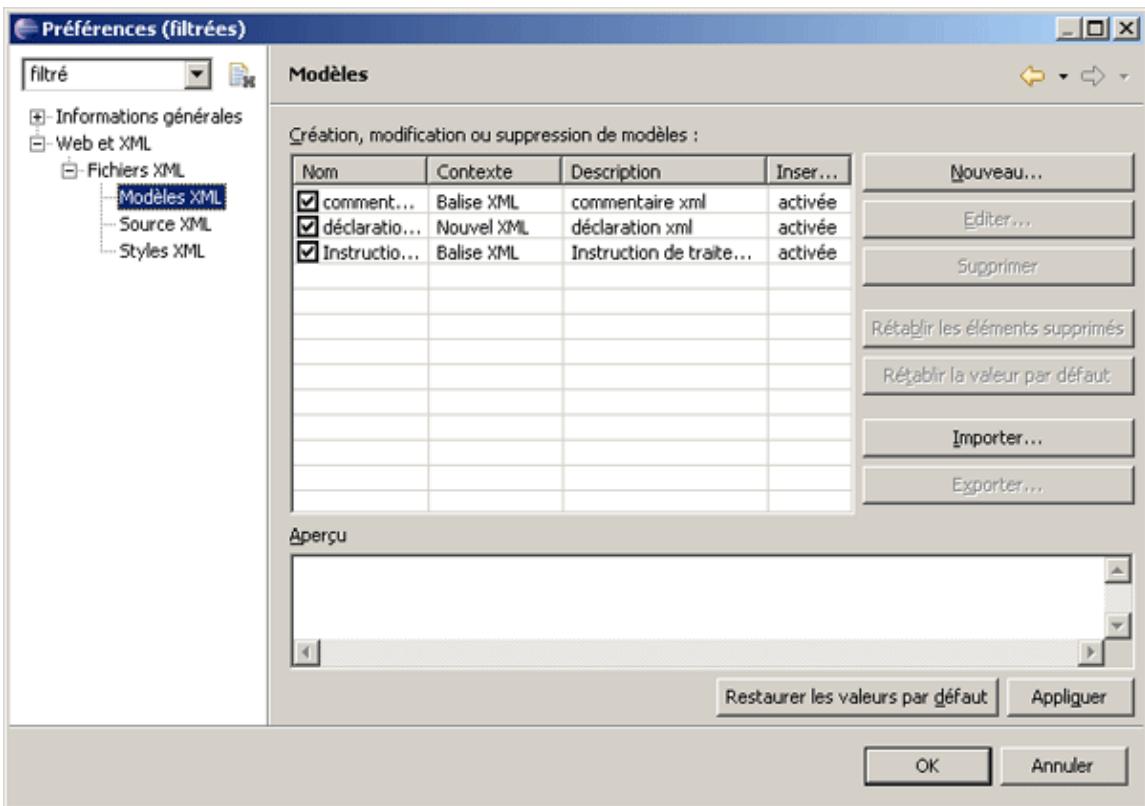
Cette page permet de sélectionner le caractère de fin de ligne des documents et la norme d'encodage par défaut des documents.



Cette page permet de configurer certaines règles pour le formatage du code source des documents XML.



Cette page permet de préciser les couleurs de chaque éléments qui peuvent composer un document XML.



Cette page permet de gérer des modèles réutilisables lors de l'édition de documents XML.

### 21.2.5. Création d'un document XML à partir d'une DTD

L'assistant de création d'un document XML propose de le faire à partir d'une DTD afin de faciliter sa rédaction

Le fichier test.dtd ci-dessous est utilisé dans les exemples de cette section.

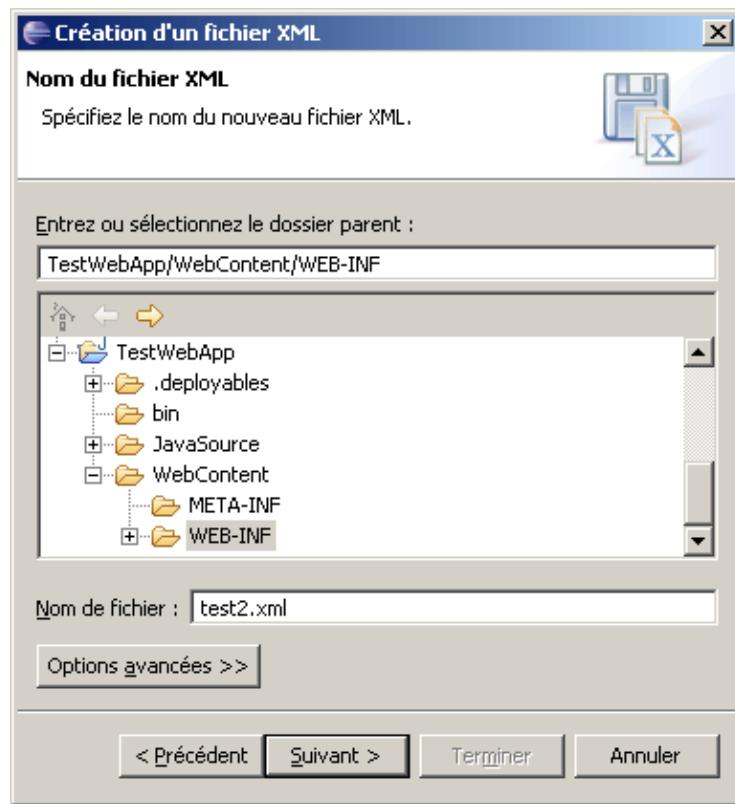
#### Exemple :

```
<!ELEMENT bibliothèque (livre+)>
<!ELEMENT livre (titre,auteur,parution?)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT parution (#PCDATA)>
```

Il faut créer une nouvelle entité de type XML/XML



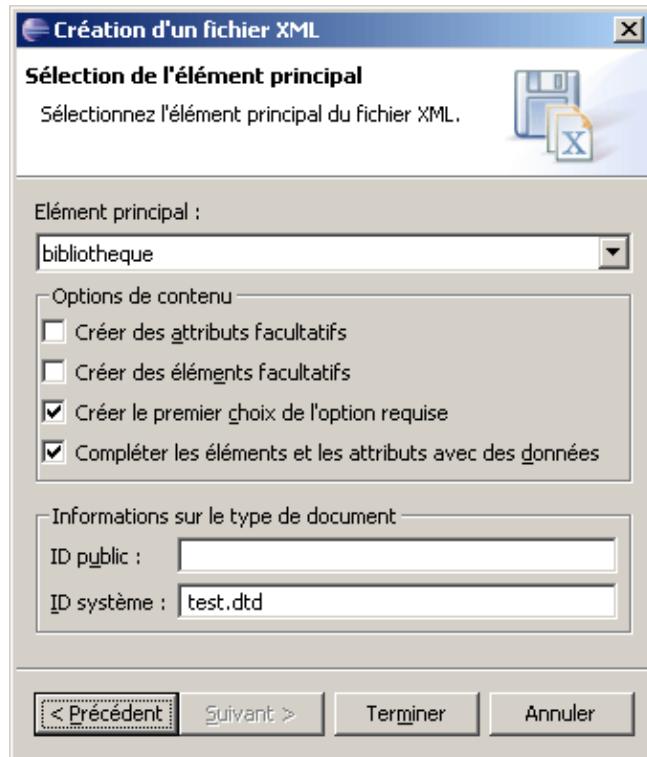
Sélectionnez « Créer un fichier XML à partir d'un fichier DTD » et cliquez sur le bouton « Suivant ».



Sélectionnez le répertoire qui va contenir le fichier, saisissez son nom et cliquez sur le bouton « Suivant »



Sélectionnez la dtd à utiliser et cliquez sur le bouton « Suivant »

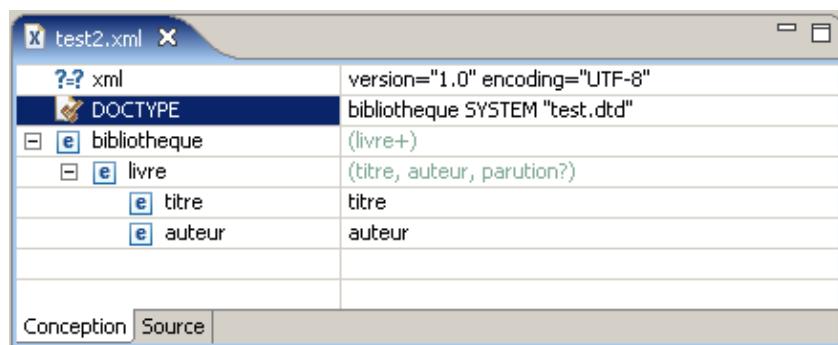


Sélectionnez l'élément racine et les options désirées puis cliquez sur le bouton « Terminer ».

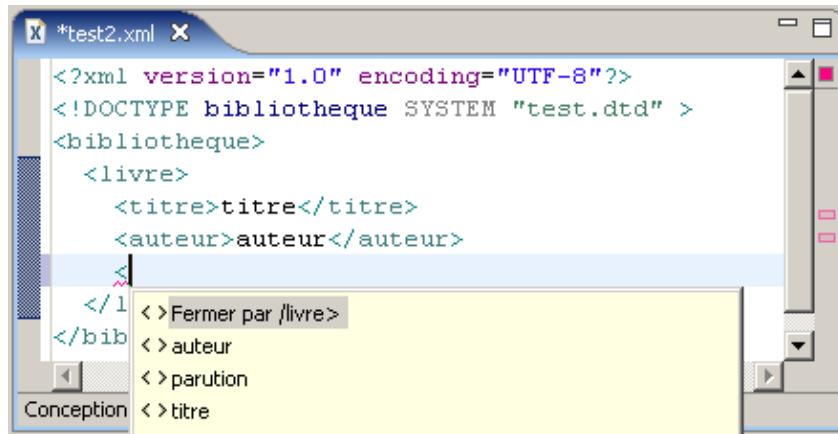
L'éditeur de code s'ouvre avec le document XML généré

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bibliotheque SYSTEM "test.dtd" >
<bibliotheque>
  <livre>
    <titre>titre</titre>
    <auteur>auteur</auteur>
  </livre>
</bibliotheque>
```

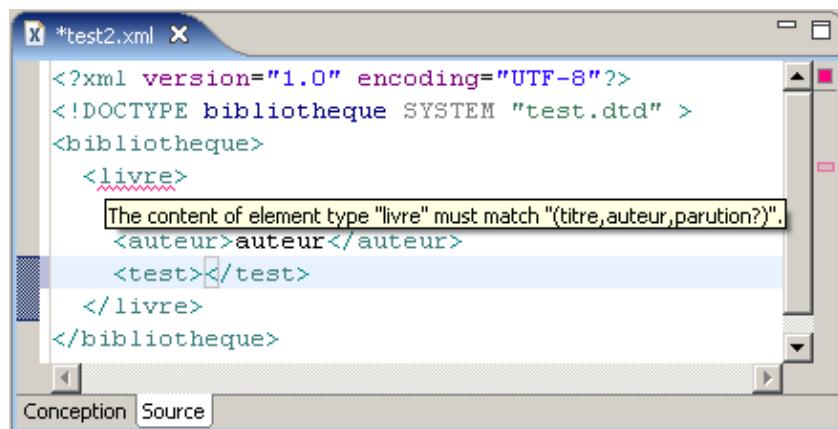
L'onglet concepteur permet de fournir une vue arborescente du contenu du document.



L'assistant de code propose uniquement les tags définis dans la dtd.



Le document est dynamiquement vérifié au fur et à mesure de la saisie du code du document



## 21.2.6. La validation des documents

Le plug in WTP propose des fonctionnalités pour valider les documents de type XML, DTD et XML Schema.

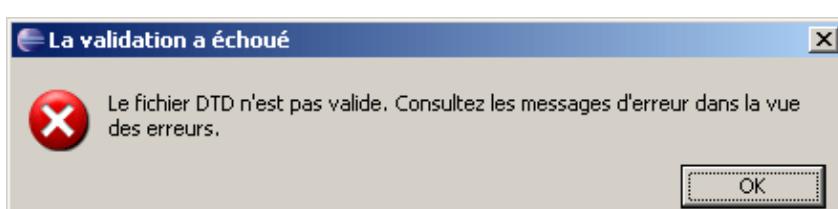
Cette validation peut être implicite (lors de la sauvegarde d'un document) ou explicite (à la demande de l'utilisateur d'Eclipse).

La validation explicite peut être demandée en utilisant l'option « Valider le fichier XXX » du menu contextuel associé à un document de type XML, DTD et XML Schema.

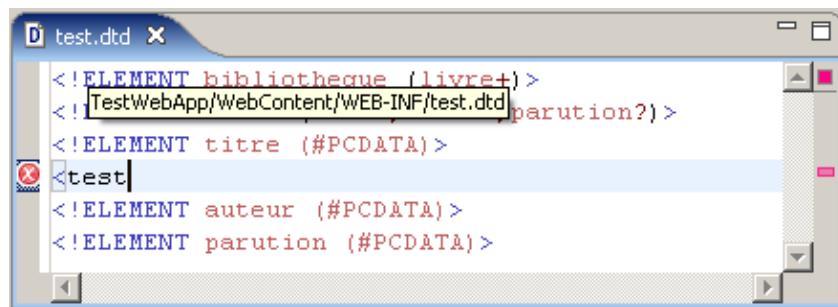
Si le document est valide, un message en informe l'utilisateur.



Si le document est invalide, un message d'erreur en informe aussi l'utilisateur.



Les erreurs des documents non valides sont signalées dans l'éditeur de code

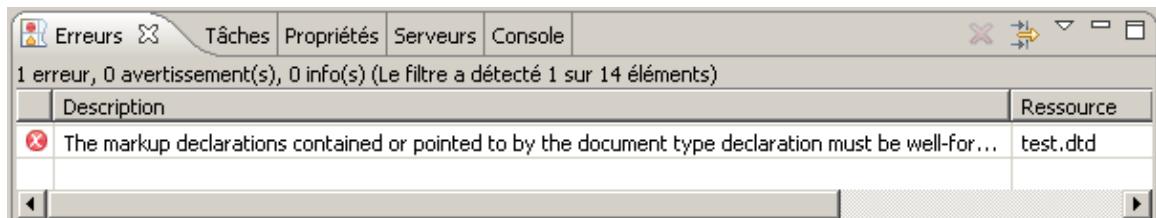


The screenshot shows the Eclipse XML editor window titled "test.dtd". It displays the following DTD code:

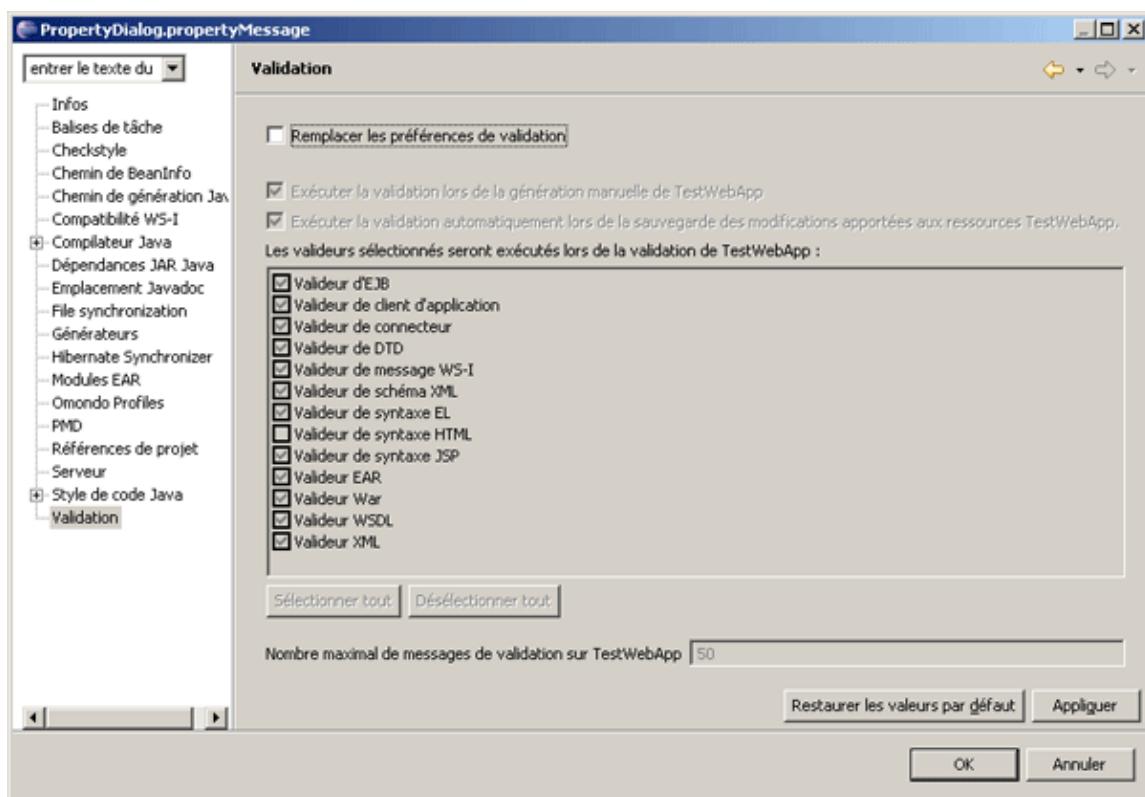
```
<!ELEMENT bibliothèque (#livre+)>
<!ELEMENT livre (#parution*)>
<!ELEMENT titre (#PCDATA)>
<?test?
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT parution (#PCDATA)>
```

The line "TestWebApp/WebContent/WEB-INF/test.dtd" is underlined in red, indicating a validation error. A red circle with a cross icon is positioned next to the opening tag "<?test?>".

Ces erreurs sont aussi incluses dans la vue « Erreurs » au même titre que les erreurs dans du code Java.



Il est possible de modifier le mode de fonctionnement de la validation automatique lors de l'enregistrement des documents d'un projet en utilisant les préférences liées au projet.



Il suffit de cocher la case « Remplacer les préférences de validation », de modifier les options voulues et de cliquez sur le bouton « OK » pour valider les options.

## 22. Le développement d'applications web

# Chapitre 22

Le développement d'applications web est particulièrement adapté pour être réalisé avec Java notamment grâce à plusieurs API de Java EE et de nombreux frameworks open source comme Struts.

Plusieurs plug-ins d'Eclipse facilitent le développement de ce type d'applications.

Ce chapitre contient plusieurs sections :

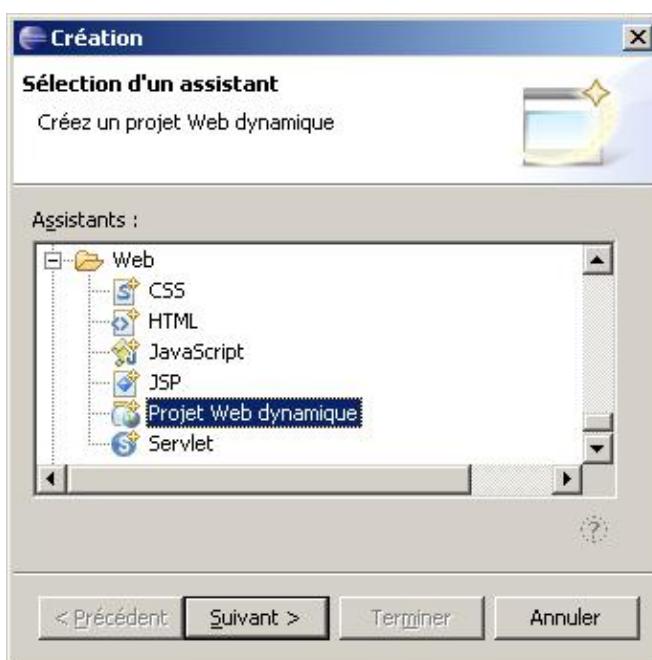
- ◆ [Le développement d'applications web avec WTP 1.0](#)
- ◆ [Le développement d'applications web avec le plug-in Lomboz 2.1](#)

### 22.1. Le développement d'applications web avec WTP 1.0

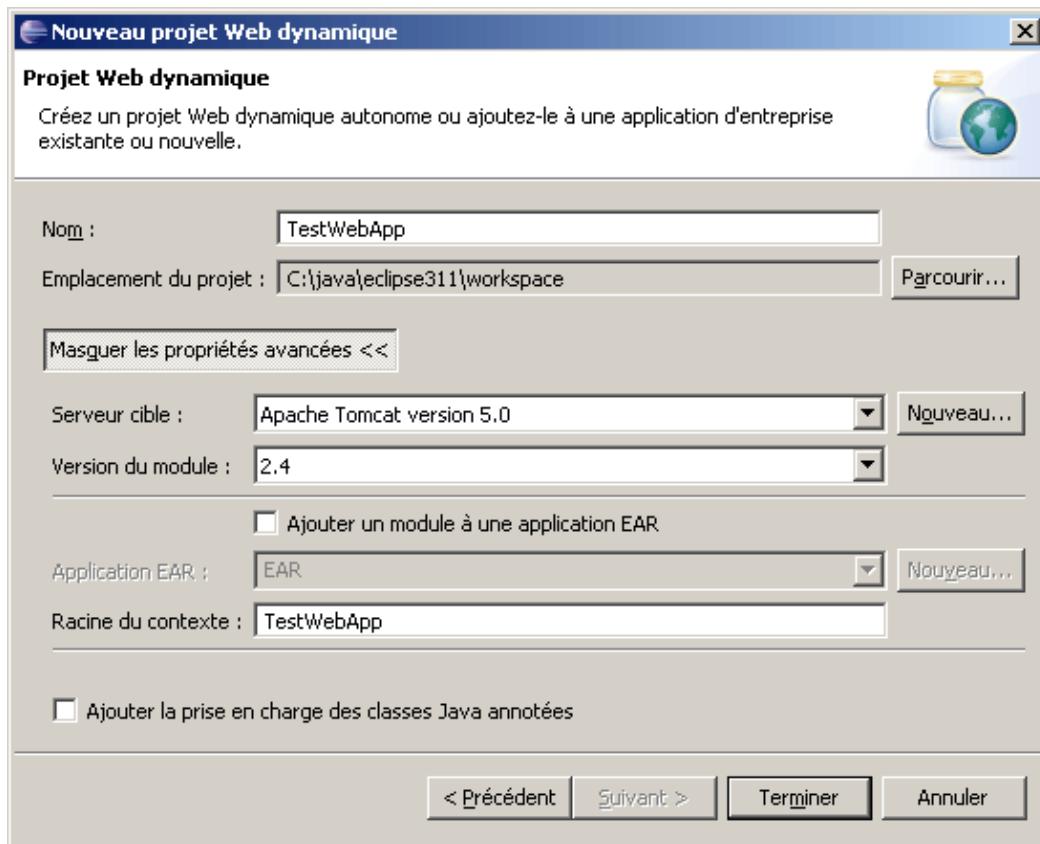
Le plug-in WTP propose des fonctionnalités pour faciliter le développement d'applications Web.

#### 22.1.1. La création d'un nouveau projet

Pour développer une application web, il est nécessaire de créer un projet de type « Web/Projet Web dynamique ». Ce type de projet est un projet Java qui va contenir une application serveur de type web.



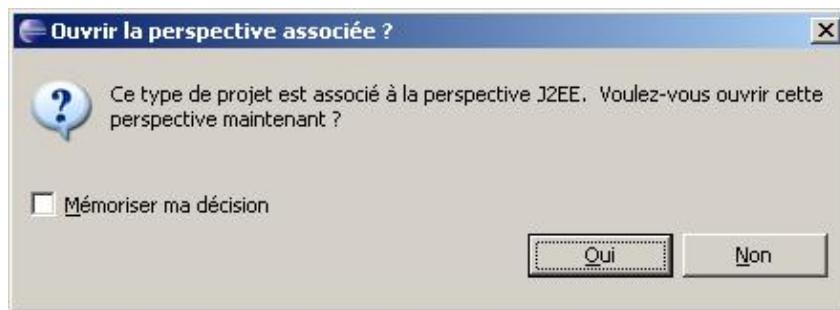
Cliquez sur le bouton « Suivant »



Il faut saisir le nom du projet et le contexte de l'application. Il est possible de sélectionner la version de l'API servlet à utiliser dans le projet et modifier le serveur cible si plusieurs sont définis.

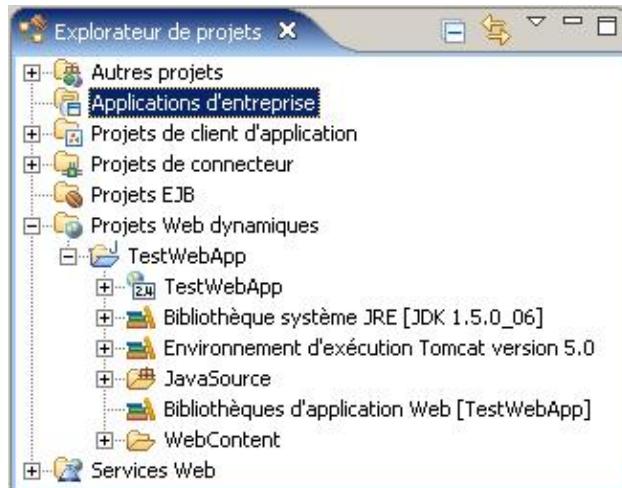
Si l'application doit être intégrée dans une archive de type EAR, il faut cocher la case demandant l'ajout de la webapp dans une telle archive.

Cliquez sur le bouton « Terminer ».

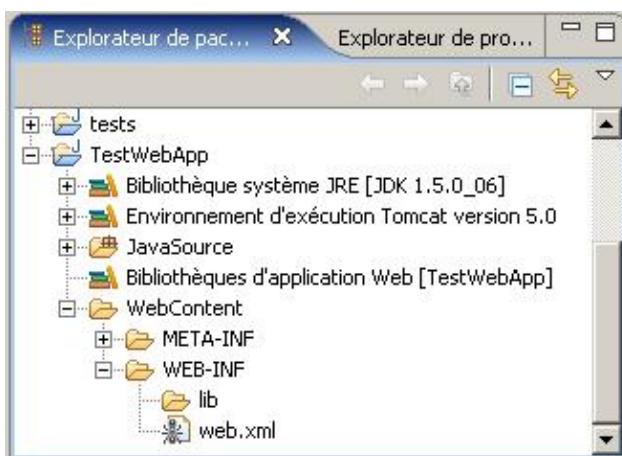


Le plus simple est de cocher « Mémoriser ma décision » et de cliquer sur le bouton « Oui » pour ouvrir la perspective J2EE.

Le nouveau projet est créé dans l'arborescence « Project Web dynamiques » de la vue « Explorateur de projets ». Cette vue permet d'avoir une présentation des projets selon les grandes familles de projets.

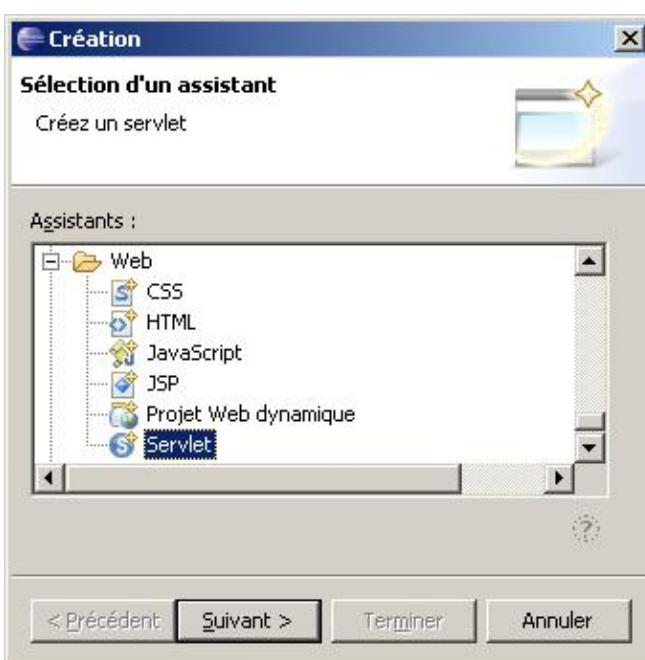


La vue « Explorateur de packages » permet d'avoir une vue plus concise du projet.

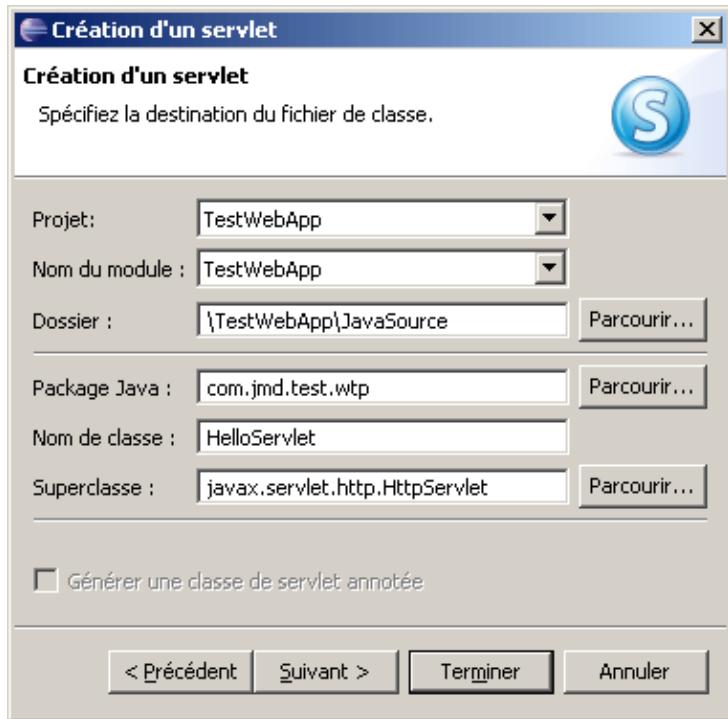


### 22.1.2. La création d'une servlet

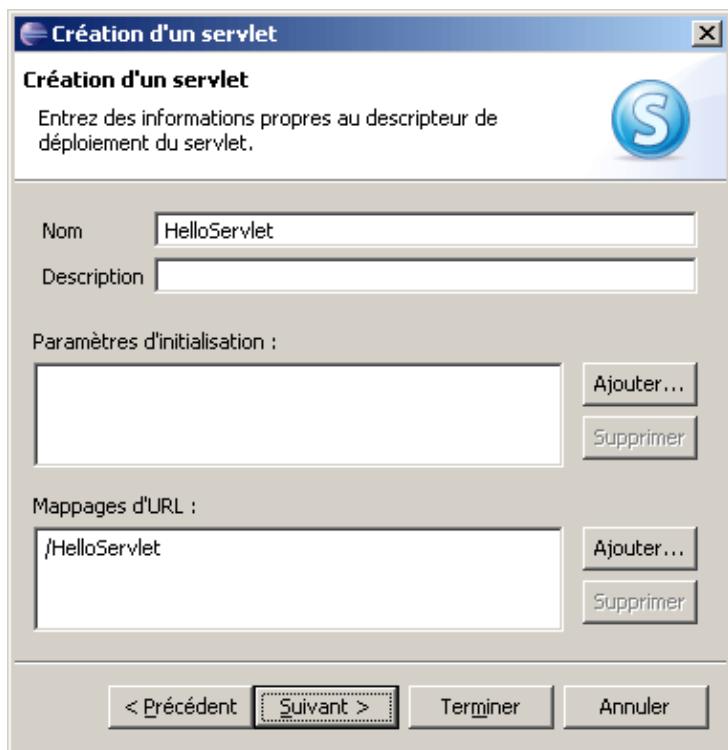
Pour ajouter une servlet au projet, il faut créer une nouvelle entité du type « Web/Servlet »



Cliquez sur le bouton « Suivant »



Il faut saisir le nom du package, de la servlet et les informations concernant la classe de la servlet : il suffit simplement de saisir le package qui va contenir la servlet et de cliquer sur le bouton « Suivant »

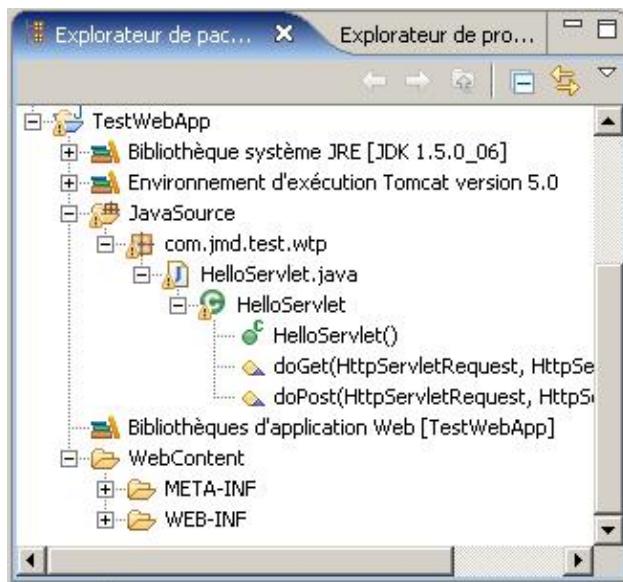


La page suivante de l'assistant permet de saisir sa description, ses paramètres et son url de mapping si celle proposée par défaut ne convient pas. Cliquez sur le bouton « Suivant ».



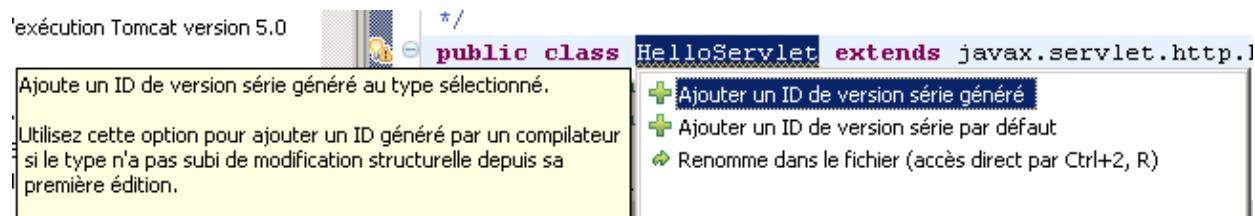
La page suivante de l'assistant permet de préciser les membres qui seront générés dans la servlet.

Pour générer la servlet, il suffit de cliquer sur le bouton « Terminer ».

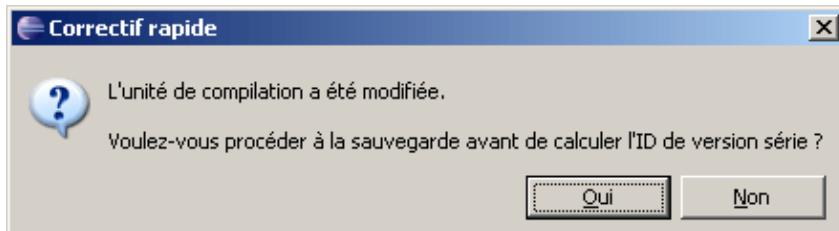


Il suffit alors de saisir le code de la servlet.

Pour éliminer l'avertissement, il suffit de demander la génération d'un identifiant en cliquant sur la petite ampoule jaune.



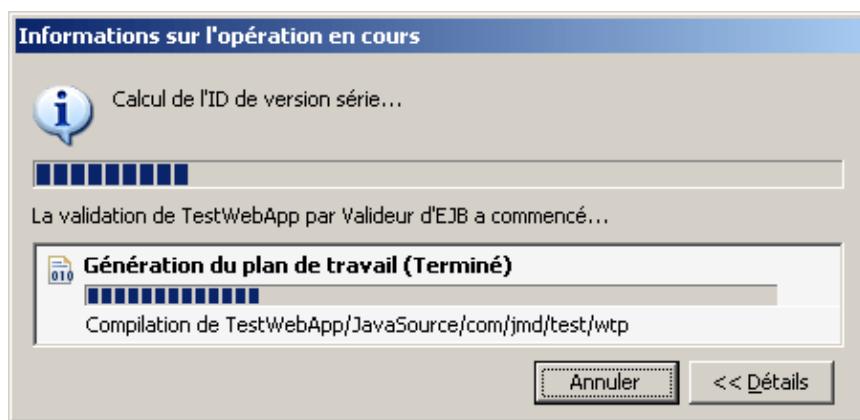
Si la source n'est pas sauvegardée, un message de confirmation de la sauvegarde est proposé à l'utilisateur.



Cliquez sur le bouton "Oui".



Ce message d'erreur apparaît si le répertoire cible de la compilation n'existe pas.



L'identifiant est généré et inséré dans le code de la servlet. Il ne reste plus alors qu'à écrire le code des traitements de la servlet.

#### Exemple :

```
package com.jmd.test.wtp;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class for Servlet: HelloServlet
 */
public class HelloServlet extends javax.servlet.http.HttpServlet
    implements javax.servlet.Servlet {

    /**
     *
     */
private static final long serialVersionUID = 341173098392449924L;

    /* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#HttpServlet()
     */
public HelloServlet() {
    super();
}
```

```

/* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>Bonjour</TITLE> ");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println("<H1>Bonjour</H1> ");
    out.println("</BODY> ");
    out.println("</HTML> ");
}

/* (non-Java-doc)
     * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
// TODO Auto-generated method stub
}
}

```

Lors de la sauvegarde de la servlet, le descripteur de déploiement est enrichi avec la déclaration de la servlet.

#### Exemple :

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>TestWebApp</display-name>
  <servlet>
    <description>
    </description>
    <display-name>HelloServlet</display-name>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>com.jmd.test.wtp.HelloServlet</servlet-class>
  </servlet>

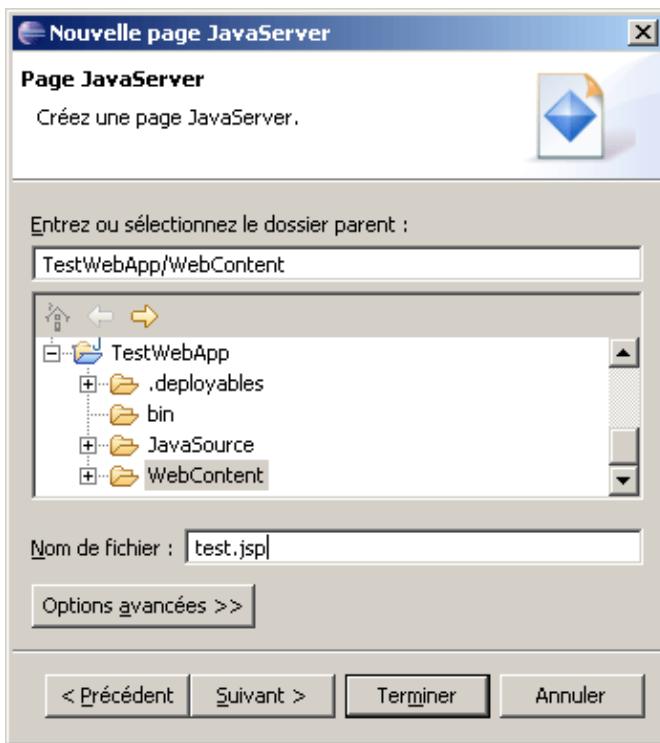
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/HelloServlet</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>

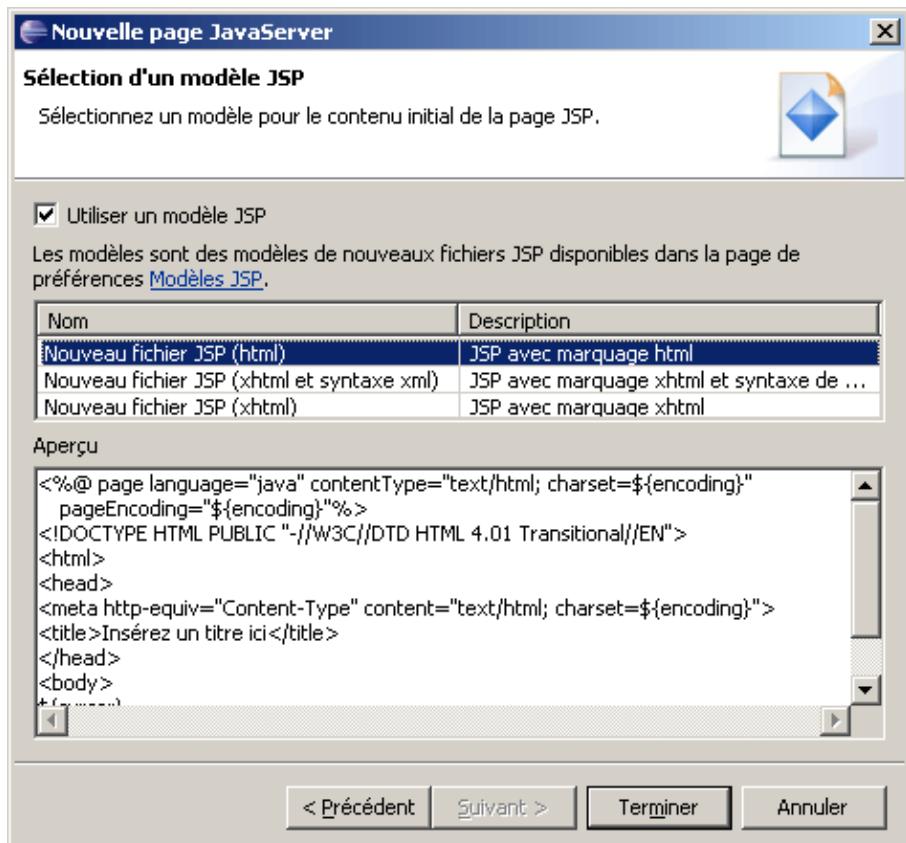
```

### 22.1.3. La création d'une JSP

Pour ajouter une JSP, il faut sélectionner l'élément « Projet Web Dynamic/WebContent » et créer une nouvelle entité du type " Web/JSP ".



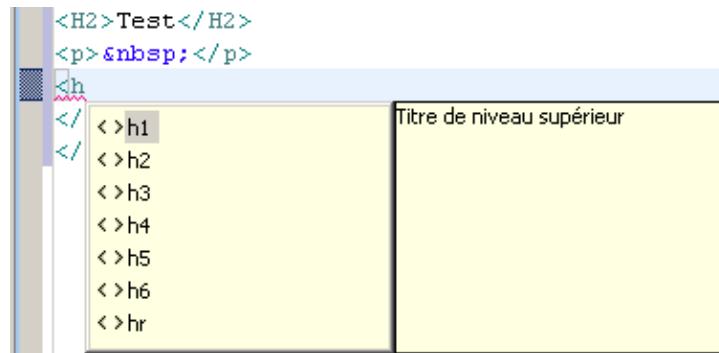
Il suffit alors de saisir le nom du fichier qui va contenir la JSP et de cliquer sur le bouton « Suivant ».



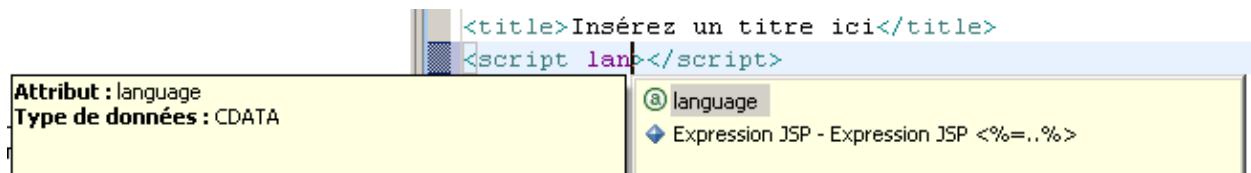
Sélectionnez le modèle à utiliser et cliquez sur le bouton "Terminer" pour créer le fichier.

L'éditeur de code s'ouvre avec le fichier créé. Cet éditeur propose plusieurs assistants pour la rédaction du code de la JSP

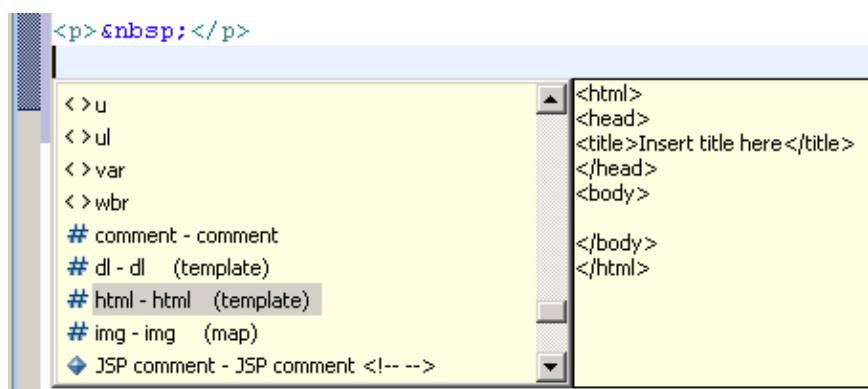
Par exemple pour un tag HTML, il suffit de saisir le début du tag et d'appuyer sur la combinaison de touches Ctrl+espace



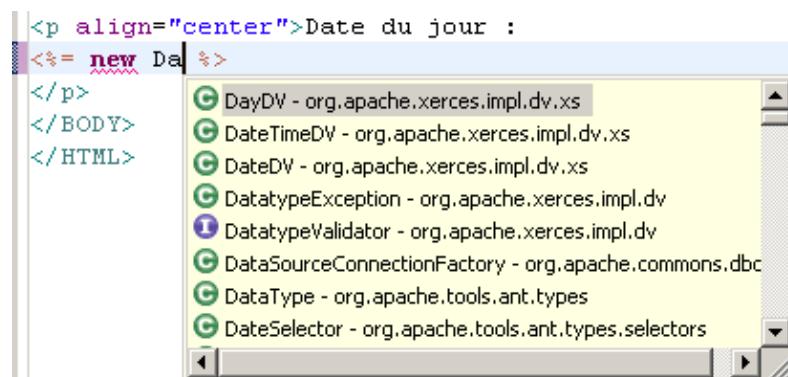
L'assistant permet aussi de saisir les attributs d'un tag.



L'assistant propose aussi des modèles

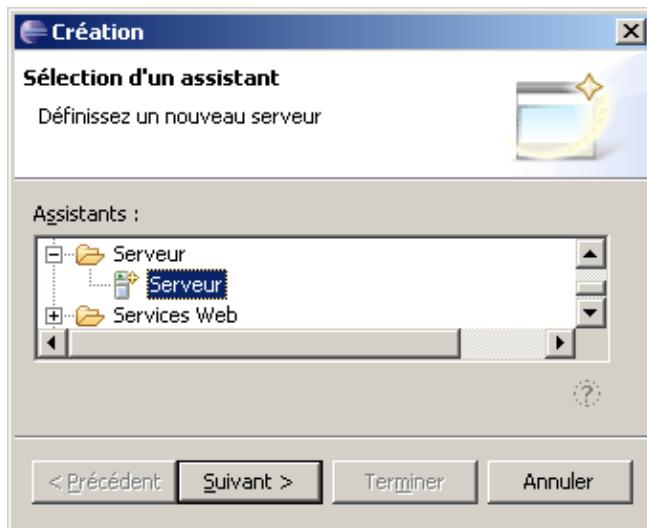


L'assistant propose la complétude de code Java dans les scriptlets.



## 22.1.4. L'exécution de l'application

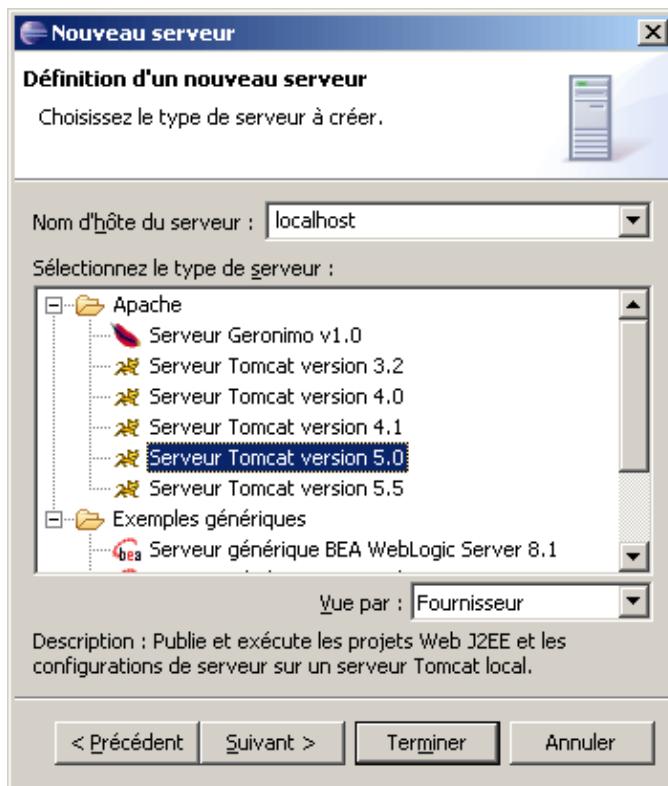
Il est nécessaire de définir un serveur en utilisant l'assistant de création d'entité de type « Serveur/Serveur ».



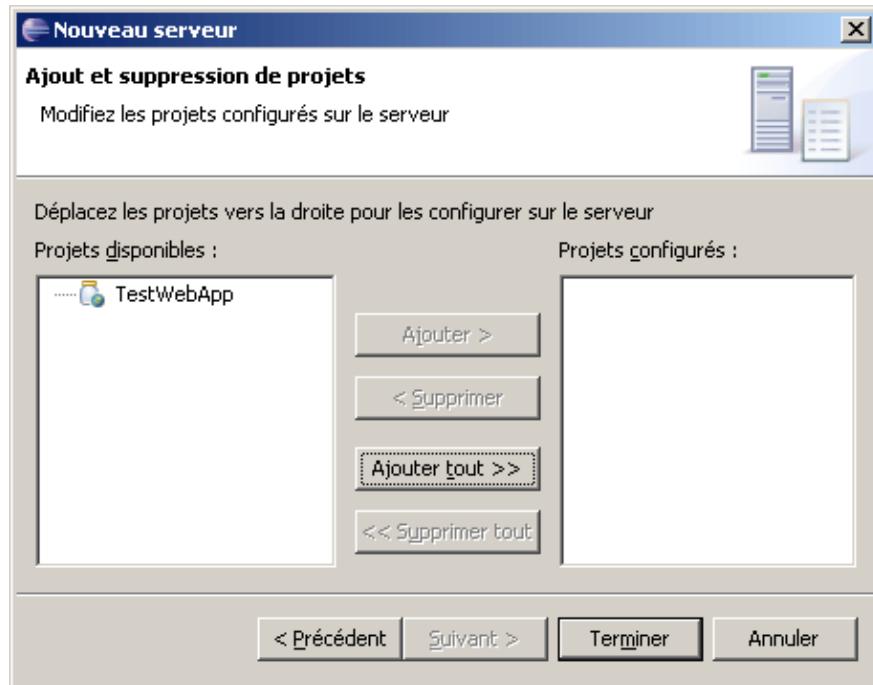
Il est aussi possible de créer un nouveau serveur en utilisant l'option « Nouveau/Serveur » du menu contextuel dans la vue « Serveurs »



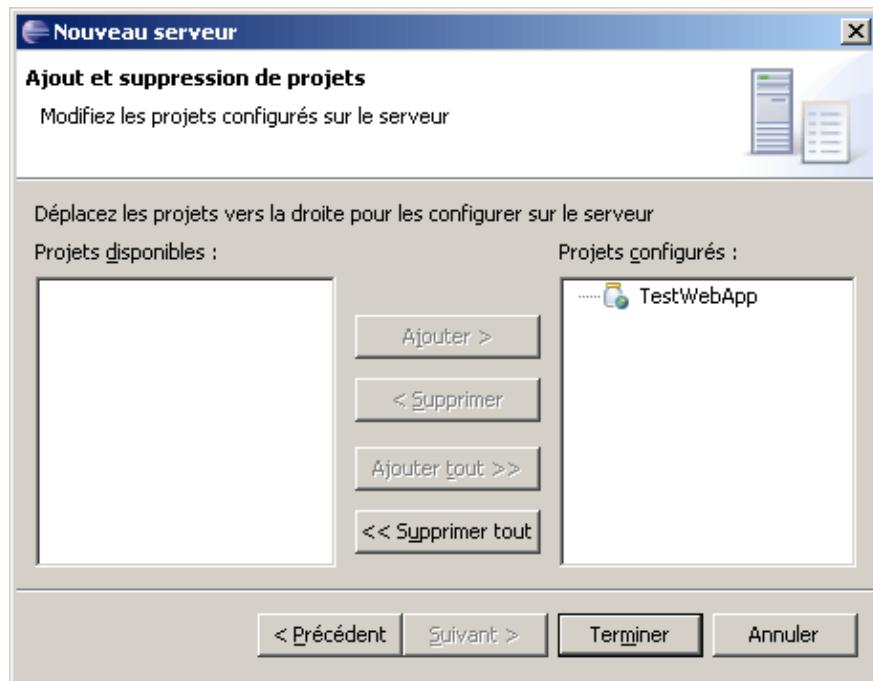
Un assistant permet de saisir les informations concernant le nouveau serveur.



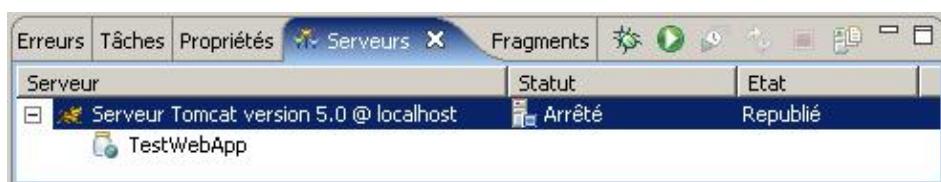
Par exemple pour Tomcat, sélectionnez la version de Tomcat installée et cliquez sur le bouton « Suivant »



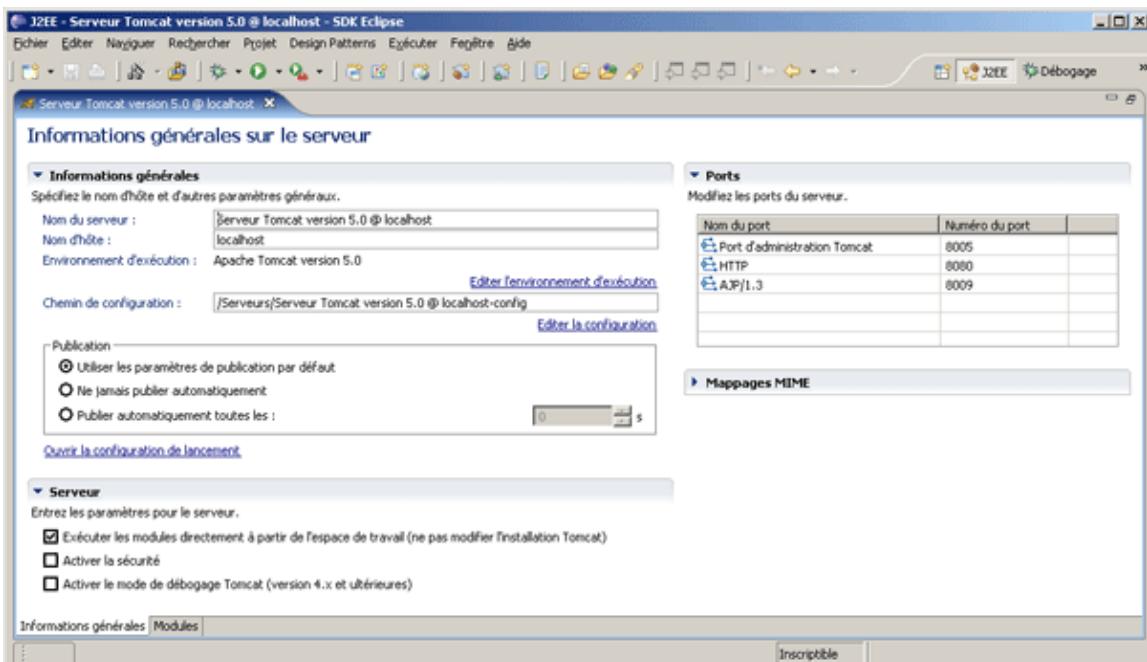
Sélectionnez le projet puis cliquez sur le bouton « Ajouter ».



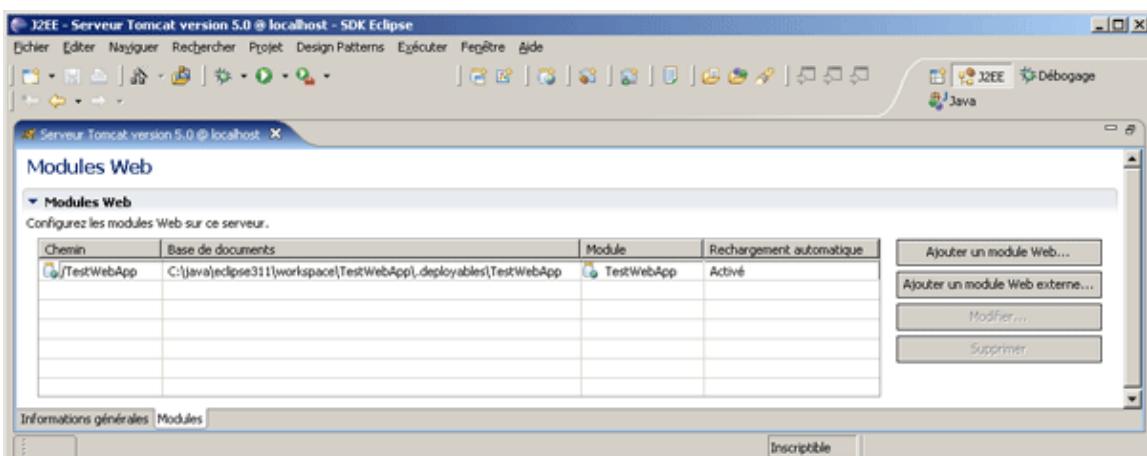
Cliquez sur le bouton « Terminer »



L'option « Ouverture » du menu contextuel sur le serveur permet d'obtenir des informations sur ce dernier.



L'onglet « Modules » permet d'obtenir la liste des applications installées sur le serveur.



Dans la vue "Serveurs", il est possible de supprimer un serveur en le sélectionnant dans la vue « Serveurs » et en utilisant l'option « Supprimer » du menu contextuel.



Pour lancer un serveur, il y a plusieurs solutions après avoir sélectionné ce serveur :

- Cliquez sur le bouton
- Utilisez l'option « Démarrer » du menu contextuel

Le serveur est lancé en arrière plan

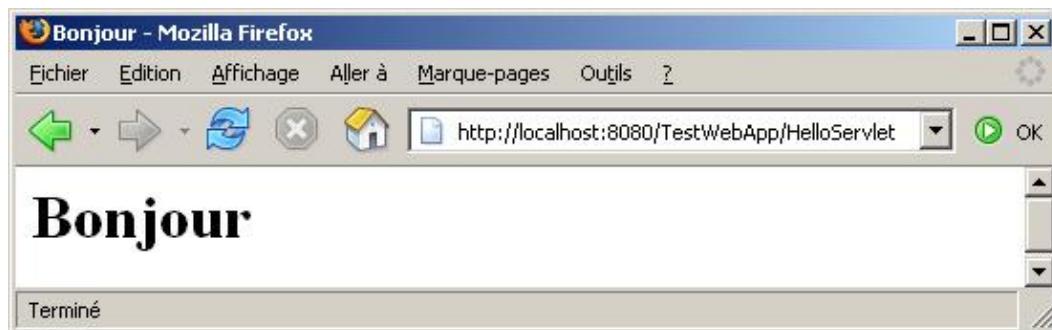


Les informations de démarrage du serveur sont affichées dans la vue « Console »

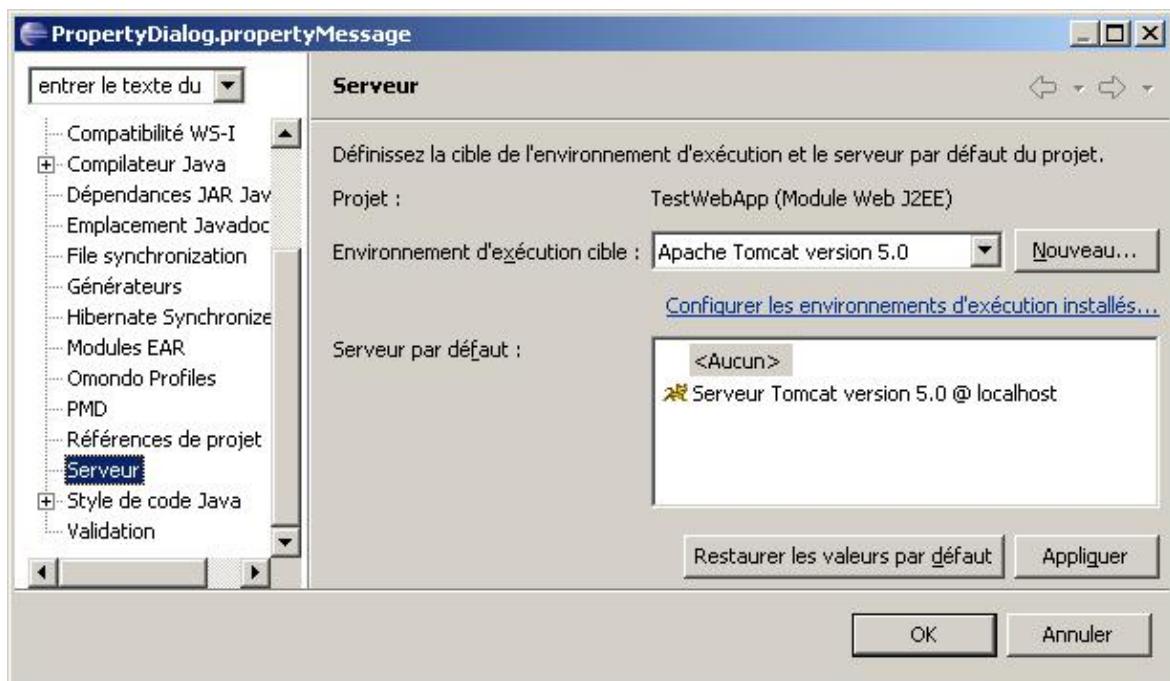
The screenshot shows the Eclipse IDE's Console view. The title bar includes tabs for Erreurs, Tâches, Propriétés, Serveurs, Fragments, Progression, and Console. The main area displays the following log output from Tomcat 5.0:

```
Serveur Tomcat version 5.0 _localhost [Apache Tomcat] C:\Program Files\Java\jdk1.5.0_04\bin\javaw.exe (13 févr. 2006 22:25:49)
13 févr. 2006 22:26:00 org.apache.coyote.http11.Http11Protocol start
INFO: Démarrage de Coyote HTTP/1.1 sur http-8080
13 févr. 2006 22:26:00 org.apache.jk.common.ChannelSocket init
INFO: JK2: ajp13 listening on /0.0.0.0:8009
13 févr. 2006 22:26:00 org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/170 config=null
13 févr. 2006 22:26:00 org.apache.catalina.startup.Catalina start
INFO: Server startup in 2744 ms
```

Il suffit alors d'ouvrir un navigateur et de saisir l'url de la servlet pour permettre son affichage



Il est possible de définir un serveur par défaut pour le projet dans les propriétés de ce dernier.



Il suffit alors de sélectionner le serveur précédemment défini, de cliquer sur le bouton « Appliquer » puis sur le bouton « OK ».

Si des modifications sont apportées à l'application alors que le serveur la concernant est lancé, le contexte est automatiquement recharge si l'option de la webapp prévoit un recharge automatique .

The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The log output from the Tomcat server is displayed:

```

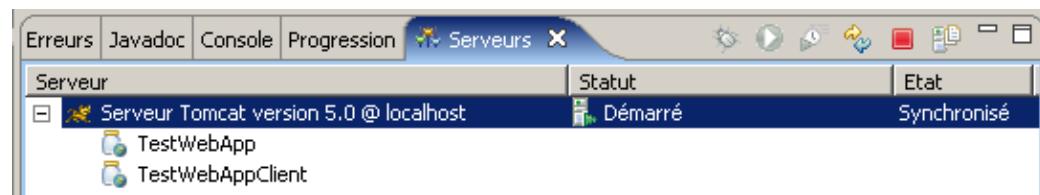
INFO: JK2: ajp13 listening on /0.0.0.0:8009
13 févr. 2006 22:32:28 org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/80 config=null
13 févr. 2006 22:32:28 org.apache.catalina.startup.Catalina start
INFO: Server startup in 1933 ms
13 févr. 2006 22:33:48 org.apache.catalina.core.StandardContext reload
INFO: Le rechargement de ce contexte a démarré

```

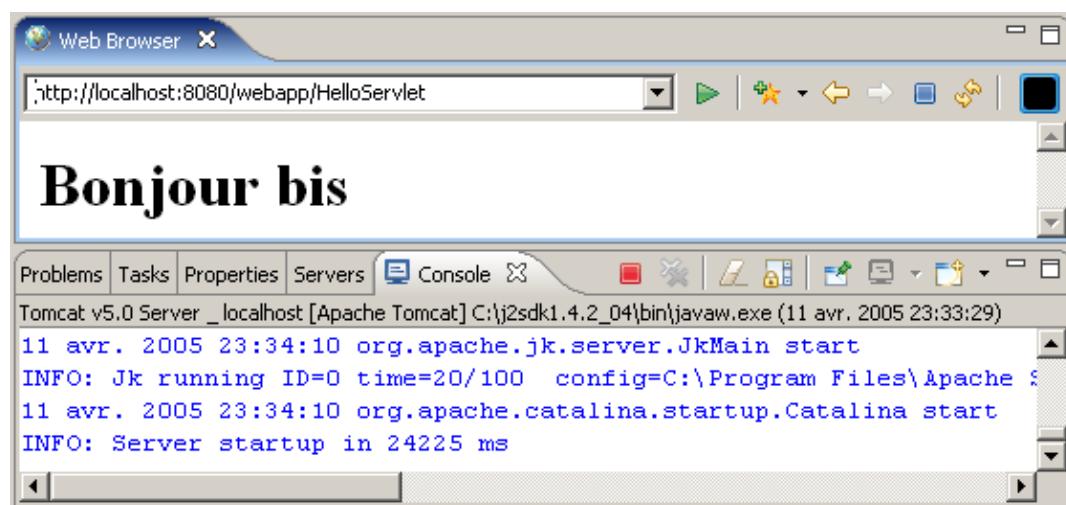
Sélectionnez l'option "Publier sur le serveur" du menu contextuel du serveur.



L'état passe alors à "Synchronisé"



Pour les exécutions suivantes, il est possible d'utiliser l'option « Executer en tant que/Exécution sur le serveur » pour lancer le serveur et ouvrir le navigateur intégré d'Eclipse avec l'url de l'application



## 22.2. Le développement d'applications web avec le plug-in Lomboz 2.1

Le plug-in Lomboz propose des fonctionnalités pour faciliter le développement d'applications de type web dans un projet de type « Java / Lomboz J2EE Wizard / Lomboz J2EE Project ». La configuration de Lomboz et la création d'un tel projet est détaillé dans le chapitre "".

Cette section va utiliser Lomboz avec JBoss et Tomcat.

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Lomboz	2.1.2
JBoss	3.0.6
Tomcat	4.1.18

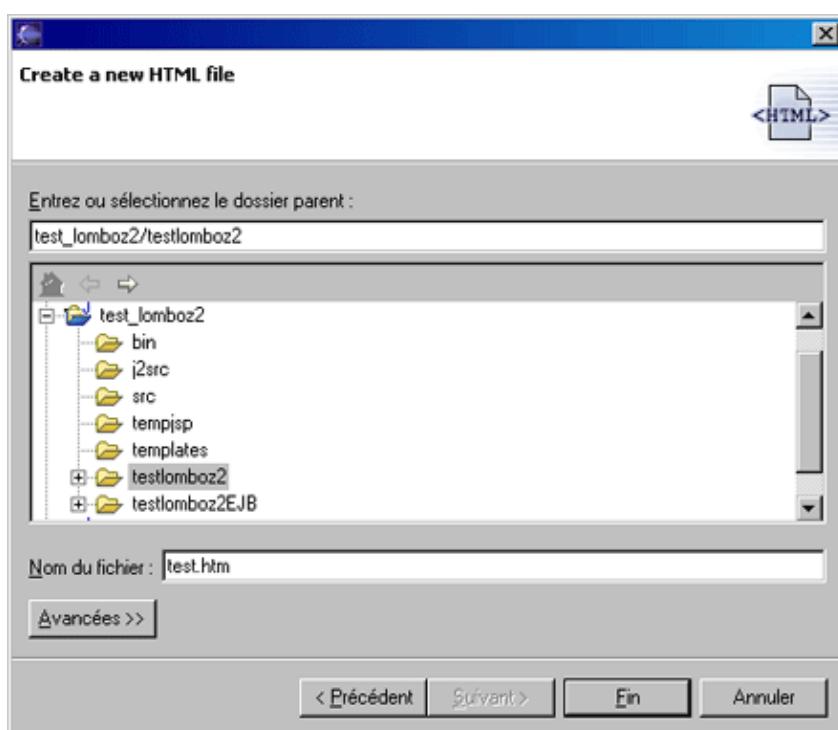
### 22.2.1. Création d'une webapp

Une webapp est contenue dans un module web. Si un tel module n'a pas été ajouté lors de la création du projet ou si il est nécessaire d'en ajouter un nouveau, il faut créer une nouvelle entité de type "Autre / Java / Lomboz J2EE Wizards / Lomboz J2EE Module".

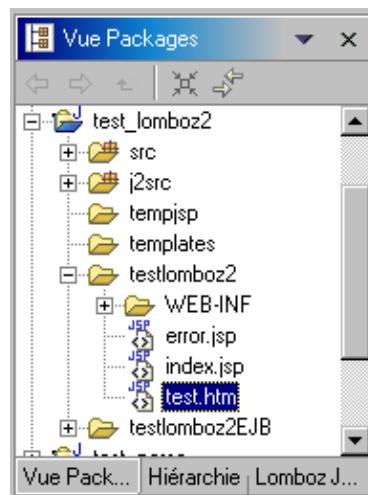
Sur l'onglet « Web module » de la page de l'assistant, il suffit de suivre les indications précisées dans la section de création d'un nouveau projet pour ajouter un nouveau module web.

### 22.2.2. Ajouter un fichier HTML à une webapp

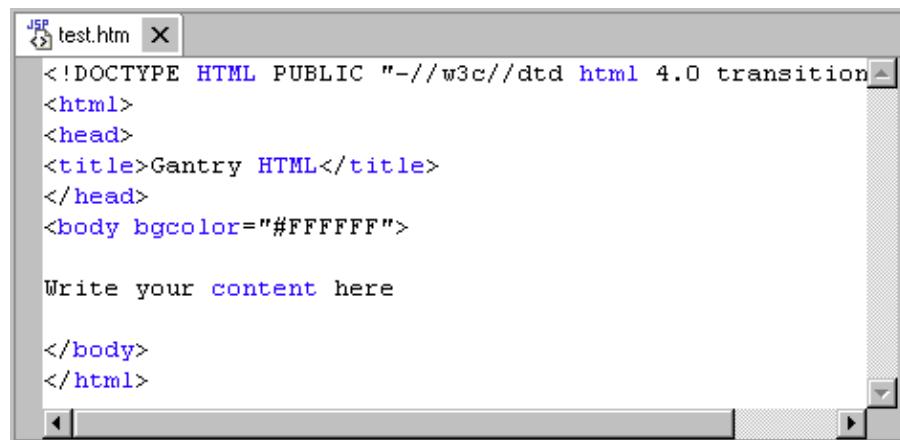
Il faut créer une nouvelle entité du type « Autres / Java / Lomboz J2EE Wizards / Lomboz HTML Wizard ».



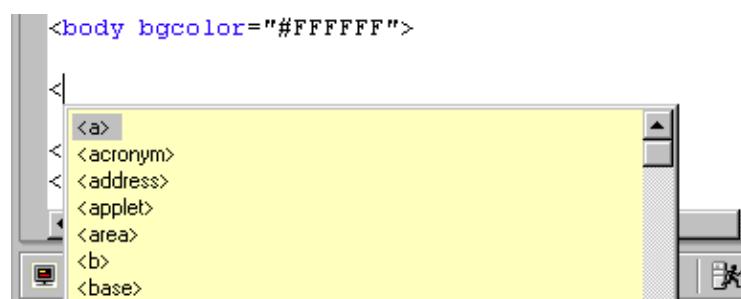
Il faut sélectionner le répertoire où sera stocké le fichier, saisir le nom du fichier et cliquer sur le bouton « Fin ».



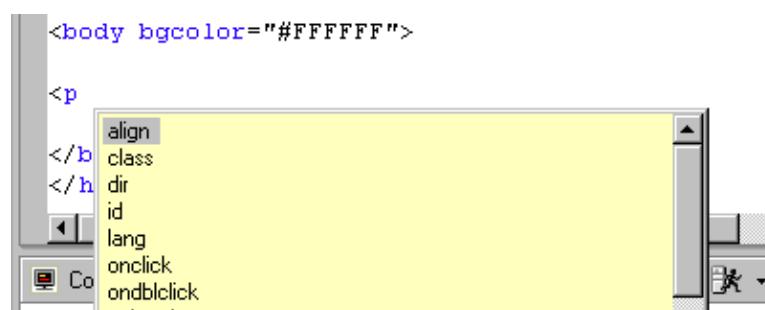
Le fichier créé est ouvert dans un éditeur dédié.



L'éditeur propose une assistance à la rédaction du code pour les tags en appuyant sur les touches "Ctrl+Espace".

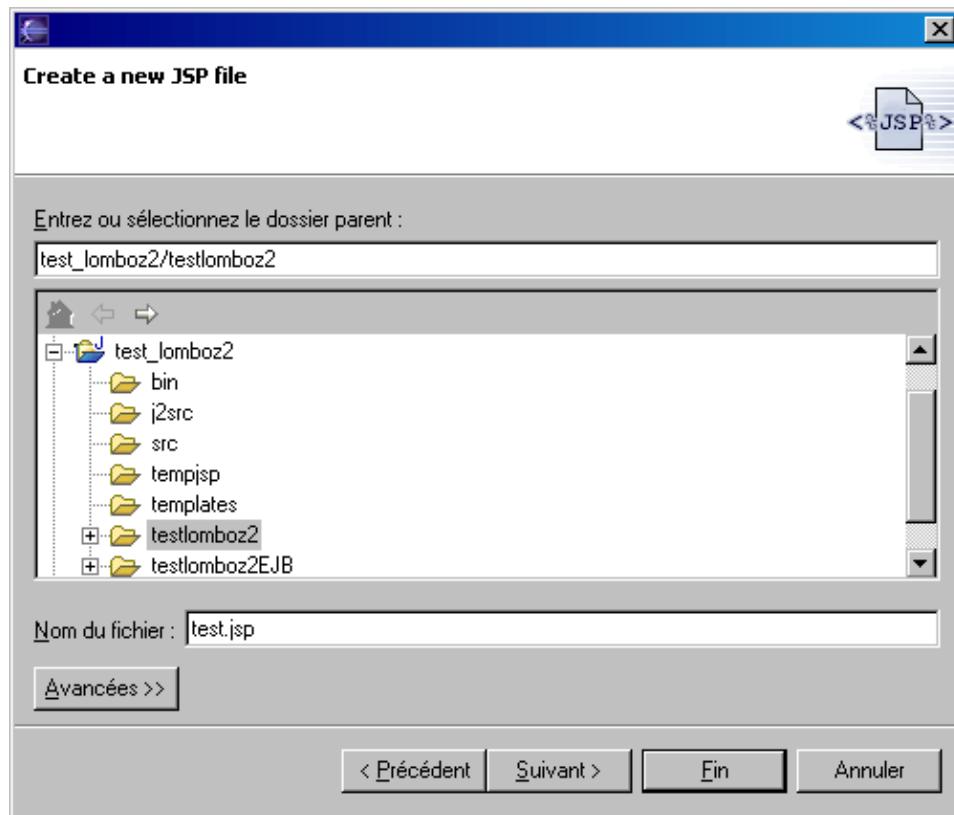


Il propose aussi une assistance pour la saisie des attributs d'un tag.

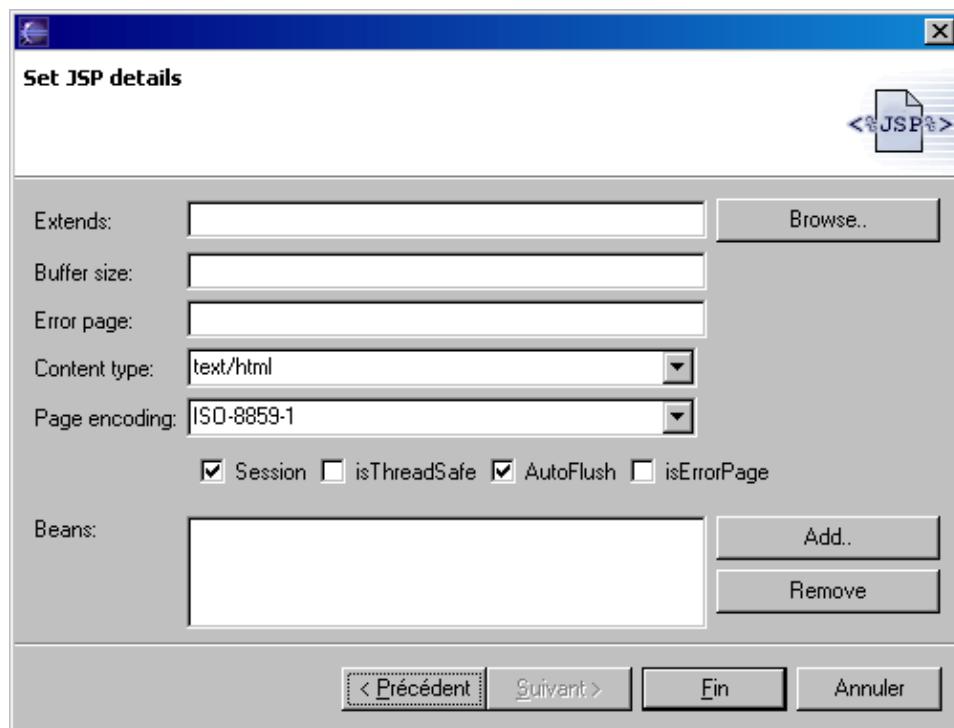


### 22.2.3. Ajouter une JSP à une webapp

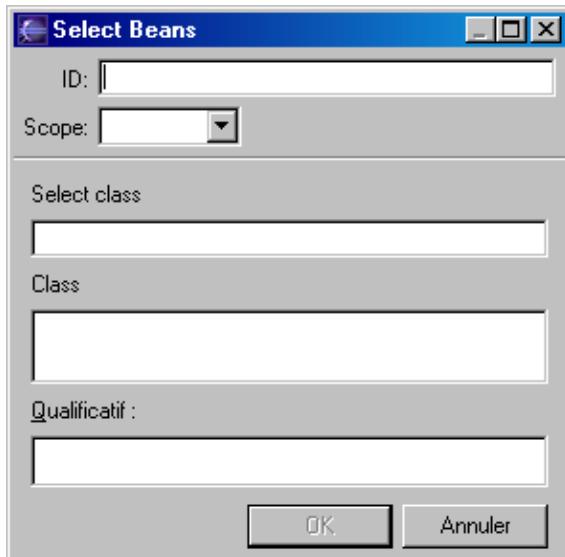
Il faut créer une nouvelle entité du type « Autres / Java / Lomboz J2EE Wizards / Lomboz JSP Wizard ».



Il faut sélectionner le répertoire, saisir le nom de la JSP et cliquer sur le bouton « Suivant ».



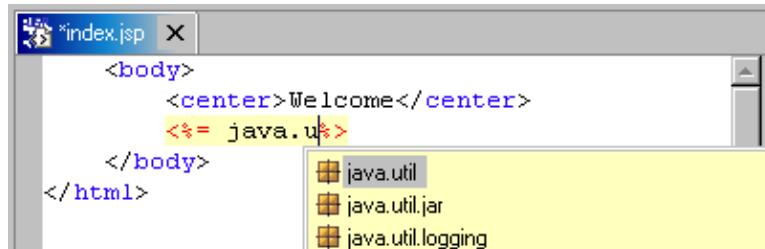
La page suivante de l'assistant permet de saisir des informations concernant la nouvelle JSP. Un clic sur le bouton « Add » permet d'ouvrir une boîte de dialogue demandant les paramètres du bean à ajouter :



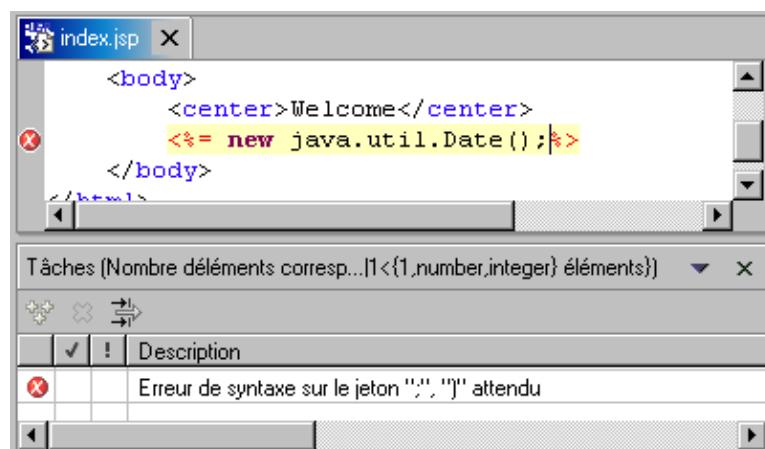
Il suffit de cliquer sur le bouton « Fin » pour générer la fichier et l'ouvrir dans l'éditeur.



L'éditeur propose une coloration syntaxique des éléments de la JSP et une assistance à la rédaction du code en appuyant sur les touches "Ctrl+Espace".



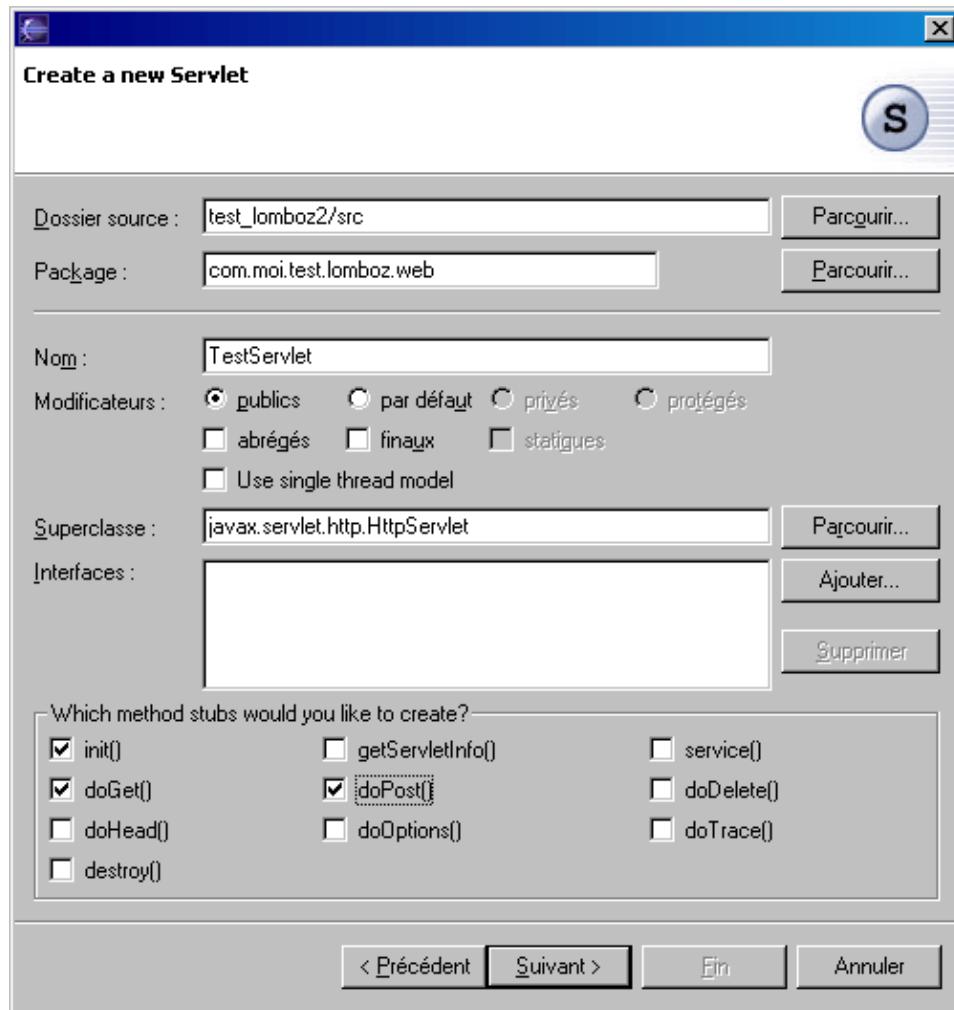
Lors de la sauvegarde du fichier, la JSP est compilée et les éventuelles erreurs sont signalées :



Il suffit alors de corriger les erreurs signalées et de sauvegarder le fichier.

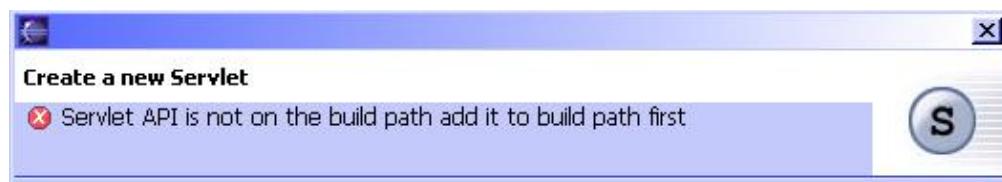
#### 22.2.4. Ajouter une servlet à une webapp

Il faut créer une nouvelle entité du type « Autres / Java / Lomboz J2EE Wizards / Lomboz Servlet Wizard ».

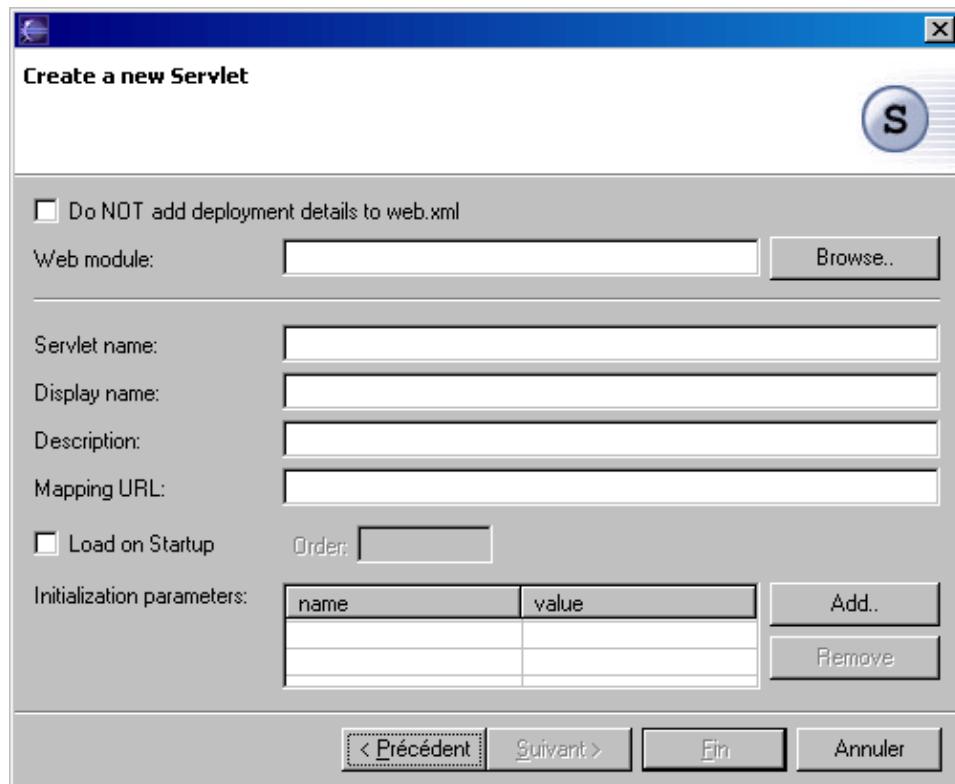


La première page de l'assistant permet de saisir les informations générales sur la nouvelle servlet notamment son package, son nom, ces modificateurs et les méthodes qui doivent être générées.

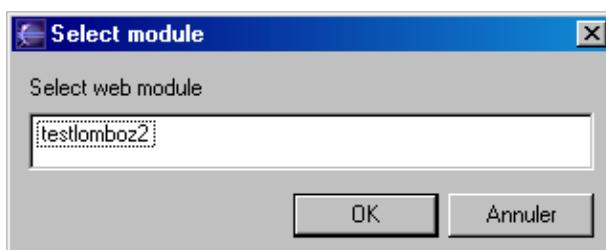
Pour créer une servlet, il faut absolument que le fichier servlet.jar soit dans le classpath du projet, sinon un message d'erreur est affiché.



Une fois toutes les informations saisies, il suffit de cliquer sur le bouton « Suivant » pour accéder à la seconde page de l'assistant.



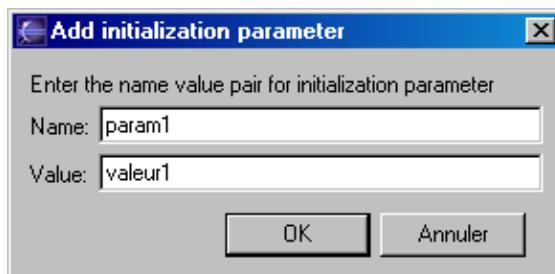
Cette page permet de saisir les informations de la servlet pour enrichir le fichier web.xml. Pour sélectionner le module web qui va contenir la servlet, il faut cliquer sur le bouton « Browse » :

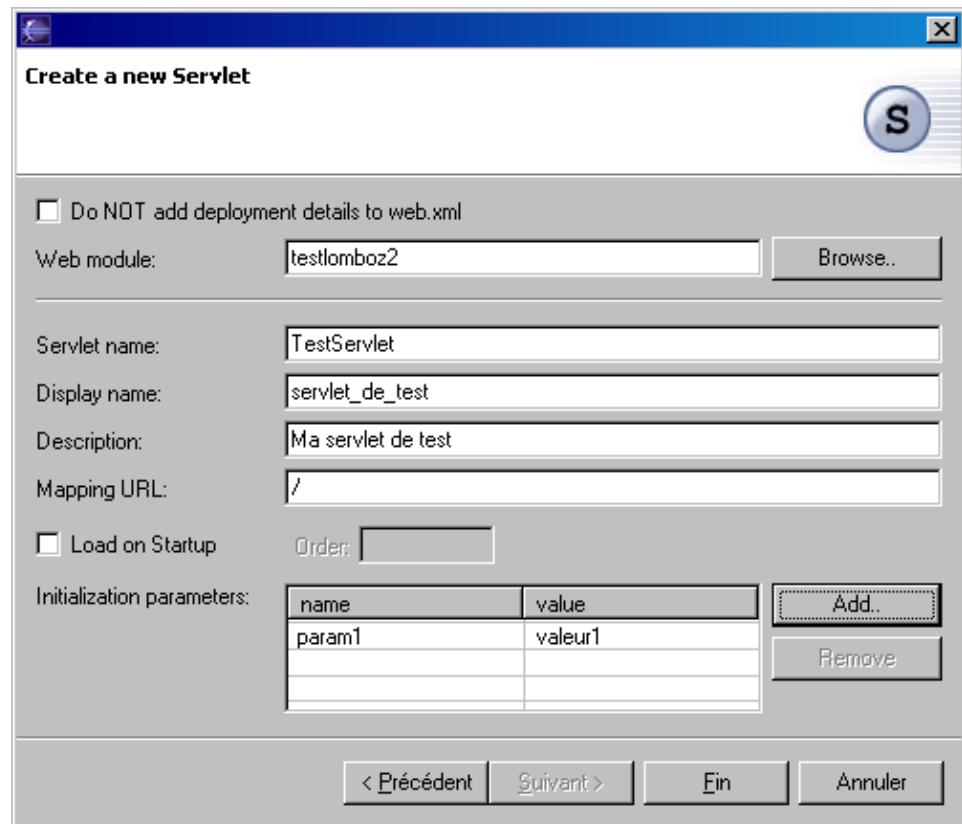


Il suffit alors de sélectionner le module web et de cliquer sur le bouton « Ok ».

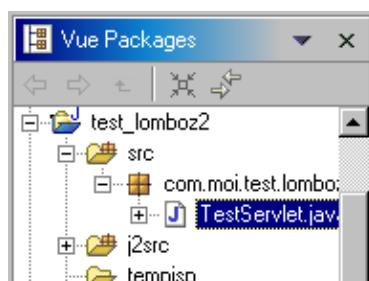
La saisie du nom de la servlet et du mapping URL est obligatoire.

Pour ajouter un paramètre à la servlet, il suffit de cliquer sur le bouton « Add » et de renseigner le nom et la valeur du paramètre dans la boîte de dialogue qui s'ouvre puis de cliquer sur le bouton « OK ».

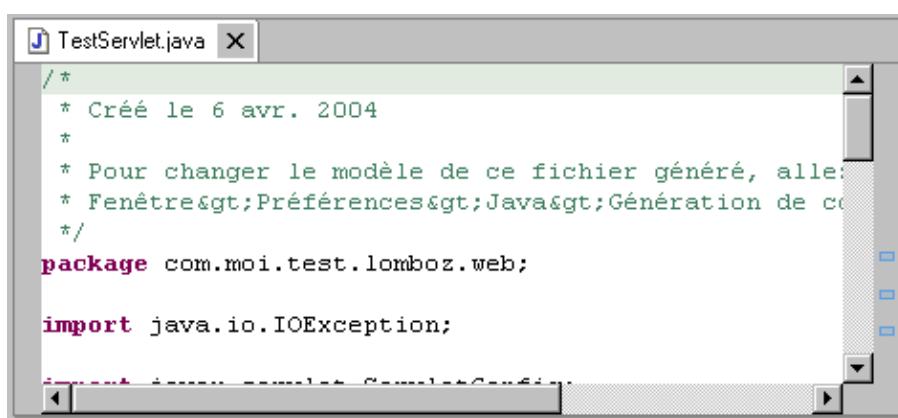




Pour créer la servlet, il suffit enfin de cliquer sur le bouton « Fin ».



L'éditeur de code s'ouvre avec le source de la servlet créée.



Le fichier WEB-INF/web.xml est automatiquement modifié par Lomboz pour tenir compte des informations concernant la nouvelle servlet :

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">
```

```

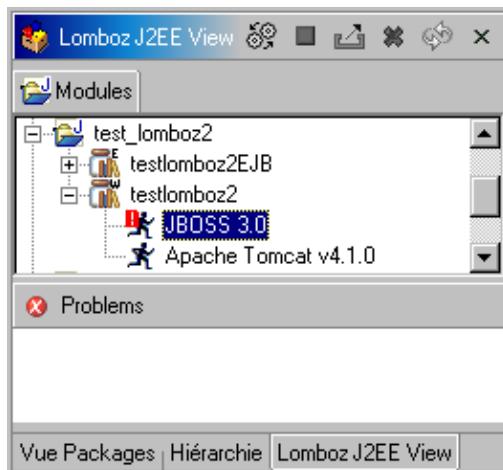
<!-- Copyright (c) 2002 by ObjectLearn. All Rights Reserved. -->
<web-app>
    <servlet>
        <servlet-name>TestServlet</servlet-name>
        <servlet-class>com.moi.test.lomboz.web.TestServlet</servlet-class>
        <display-name>servelet_de_test</display-name>
        <description>Ma servlet de test</description>
        <init-param>
            <param-name>param1</param-name>
            <param-value>valeur1</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>TestServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <error-page>
        <error-code>404</error-code>
        <location>/error.jsp</location>
    </error-page>
</web-app>

```

## 22.2.5. Tester une webapp

Pour tester une webapp, il faut démarrer le serveur d'application et déployer l'application sur le serveur.

Pour cela, il faut utiliser la vue « Lomboz » en appuyant sur le bouton  dans la barre d'outils.



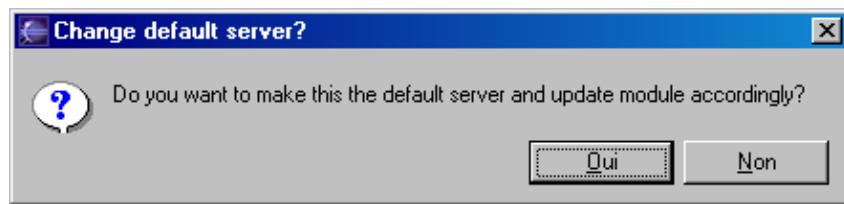
Cette vue possède plusieurs boutons pour réaliser certaines tâches :

-  Lancer le serveur
-  Arrêter le serveur
-  Déployer le module dans le serveur
-  Supprimer le module du serveur
-  Permet de rafraîchir la vue

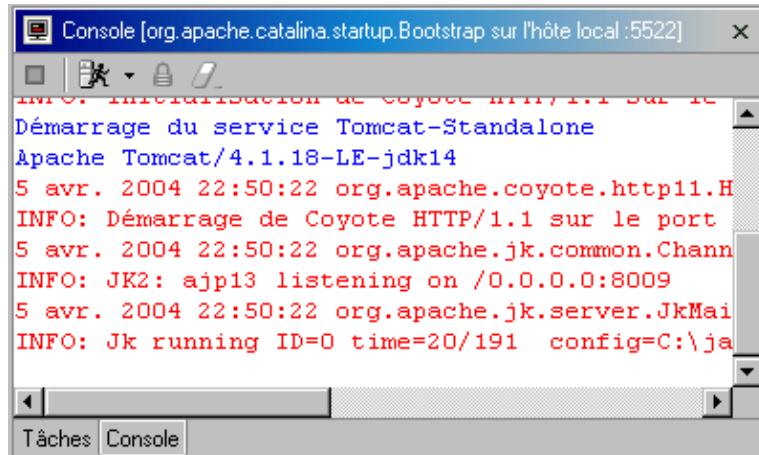
Pour lancer un serveur, il suffit de le sélectionner dans l'arborescence et de cliquer sur le bouton

correspondant ou d'utiliser l'option « Debug server » ou « Run server » du menu contextuel.

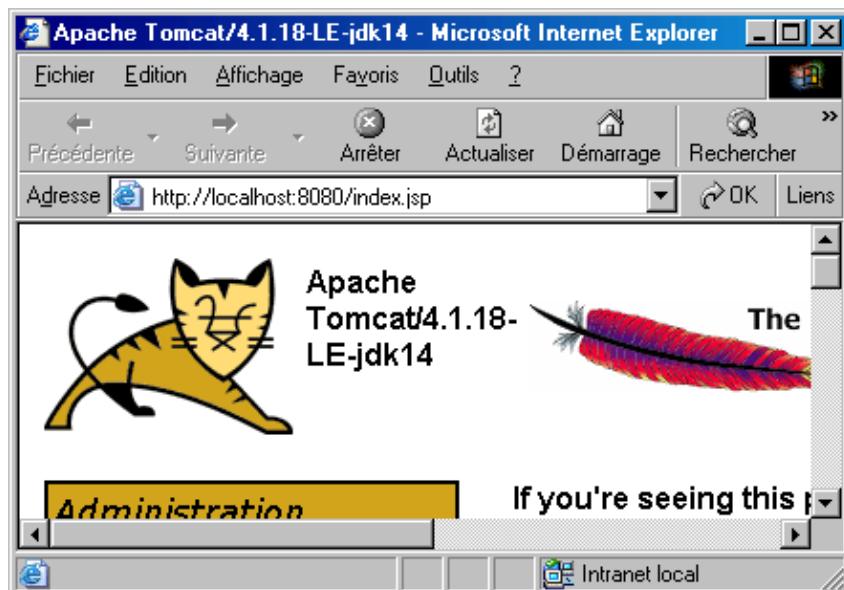
Lors du premier lancement, un boîte de dialogue demande si le serveur doit devenir le serveur par défaut du module.



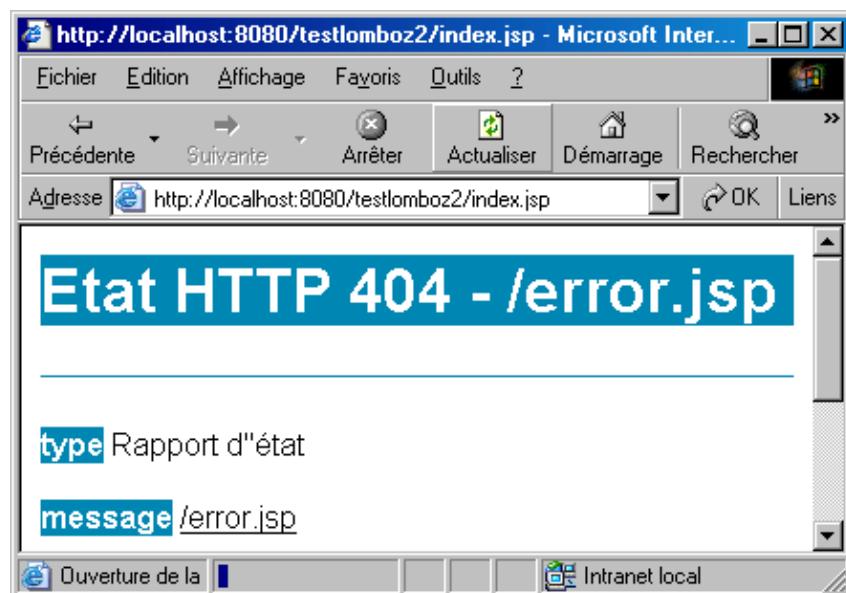
La progression du démarrage du serveur web Tomcat est affichée dans la vue « Console ».



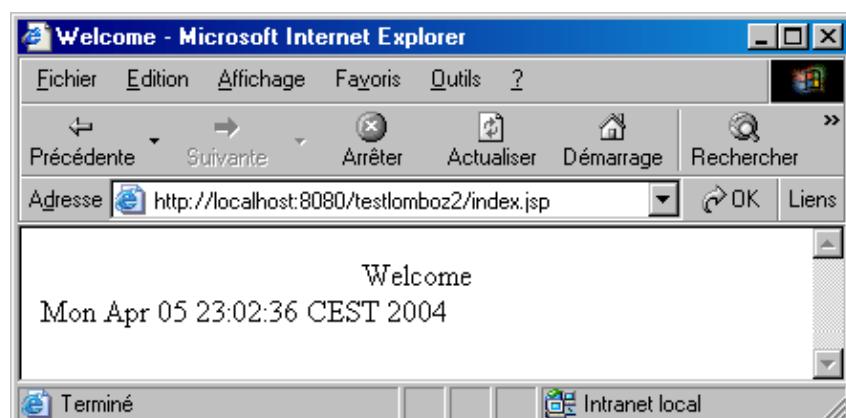
Pour vérifier que Tomcat est lancé, il suffit d'ouvrir un navigateur et de saisir l'url <http://localhost:8080>



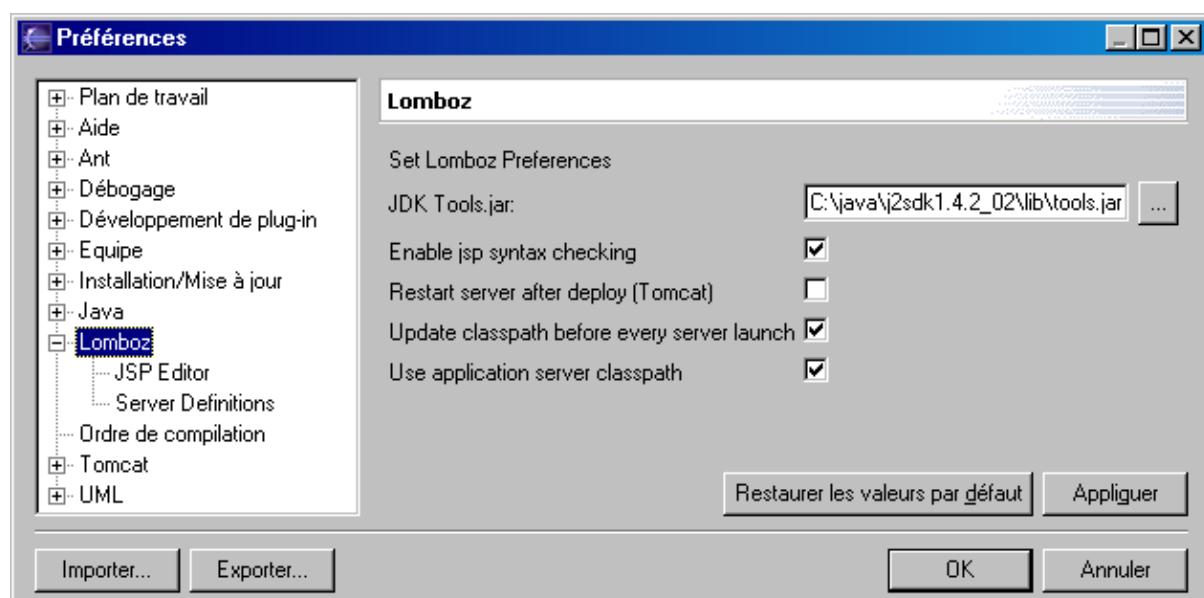
Avec Tomcat, il faut le redémarrer après le déploiement d'une webapp sinon un message d'erreur est signalé lors de l'appel de la webapp dans le navigateur.



Pour résoudre le problème manuellement, il faut arrêter le serveur et le lancer de nouveau.



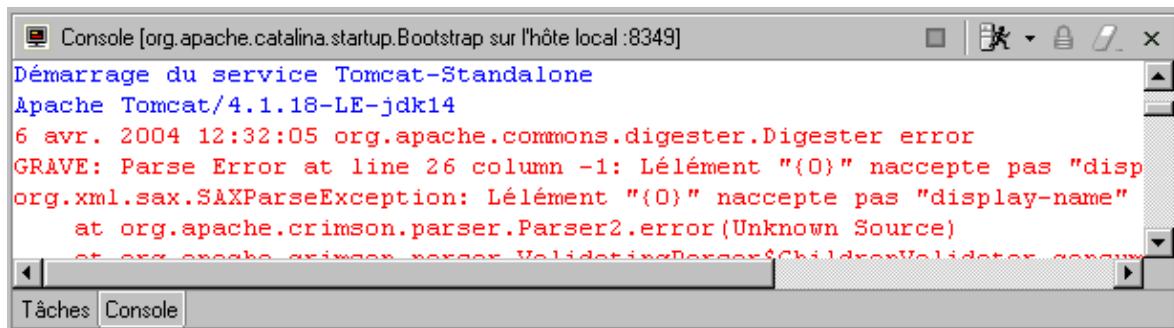
Pour résoudre automatiquement ce problème, il suffit de changer les paramètres de lomboz



Il faut cocher la case à cocher "Restart server after deploy (Tomcat)" et cliquer sur le bouton "OK".

Il est possible que certaines erreurs empêchent le démarrage de la webapp.

Exemple :



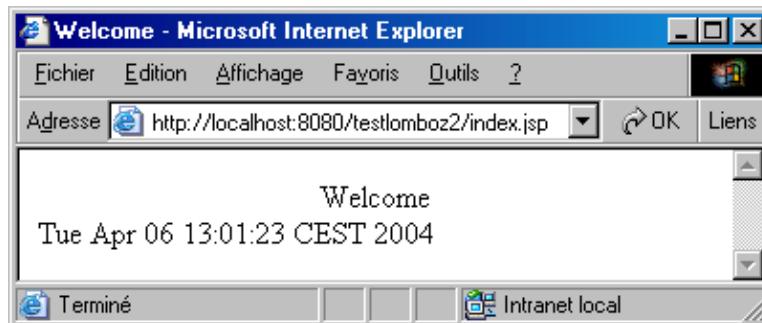
The screenshot shows the Eclipse IDE's Console view with the title "Console [org.apache.catalina.startup.Bootstrap sur l'hôte local :8349]". It displays the output of a Tomcat startup. The text is mostly blue, indicating standard output, except for several error messages in red:

```
Démarrage du service Tomcat-Standalone
Apache Tomcat/4.1.18-LE-jdk14
6 avr. 2004 12:32:05 org.apache.commons.digester.Digester error
GRAVE: Parse Error at line 26 column -1: L'élément "(0)" n'accepte pas "disp"
org.xml.sax.SAXParseException: L'élément "(0)" n'accepte pas "display-name"
    at org.apache.crimson.parser.Parser2.error(Unknown Source)
    at org.apache.crimson.parser.ValidatingParser$ChildValidationHandler.error(ValidatingParser.java:111)
```

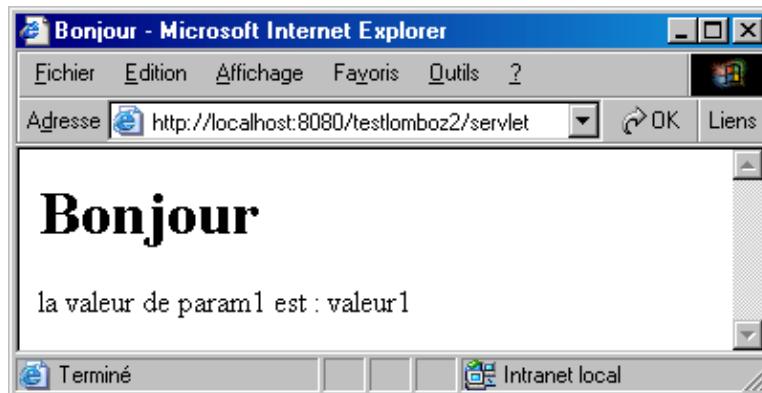
Pour résoudre ce problème, il faut modifier dans le fichier web.xml l'ordre du tag web-app/servlet/servlet-class pour le mettre après le tag web-app/servlet/description

Pour accéder aux entités qui composent la webapp, il suffit de saisir l'url correspondante dans le navigateur.

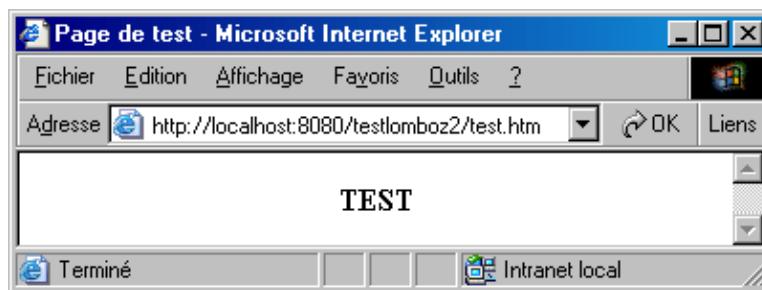
Exemple avec la JSP d'accueil de la webapp



Exemple avec la servlet (attention, le mapping url a été modifié en /servlet dans le fichier web.xml)



Exemple avec le fichier HTML



# Chapitre 23

Struts est un framework pour applications web développé par le projet Jakarta de la fondation Apache. C'est la plus populaire des frameworks pour le développement d'applications web avec Java .

Struts met en oeuvre le modèle MVC 2 (Modèle / Vue / Contôleur) basé sur une seule servlet et des JSP pour chaque application. L'application de ce modèle permet une séparation en trois parties distinctes de l'interface, des traitements et des données de l'application.

Struts se concentre sur la vue et le contrôleur. L'implémentation du modèle est laissée libre aux développeurs : ils ont le choix d'utiliser des java beans, un outil de mapping objet/relationnel ou des EJB.

Pour le contrôleur, Struts propose une unique servlet par application qui lit la configuration de l'application dans un fichier au format XML. Cette servlet reçoit toutes les requêtes de l'utilisateur concernant l'application. En fonction du paramétrage, il instancie un objet de type Action qui contient les traitements et renvoie une valeur particulière à la servlet. Ceci lui permet de déterminer la JSP qui affichera le résultat à l'utilisateur.

Il existe un projet open source nommé Easy Struts qui est un plug-in dont le but est de faciliter la mise en oeuvre de Struts avec Eclipse. Ce plug-in très intéressant et pratique ne semble malheureusement plus évoluer : il ne fonctionne d'ailleurs pas avec les versions 3.x de d'Eclipse.

Cette section va mettre en oeuvre les outils suivants sous Windows :

Outil	Version	Rôle
JDK	1.4.2_03	
Eclipse	2.1	IDE
Tomcat	5.0.28	conteneur web
plug-in Tomcat de Sysdeo	3	arrêt et démarrage de Tomcat
plug-in Easy Struts	0.6.4	faciliter l'utilisation de Struts

### 23.1. Le plug-in Easy Struts

Le site officiel du plug-in Easy Struts est à l'url : <http://easystruts.sourceforge.net/>

Ce plug-in permet notamment de faciliter la réalisation de certaines tâches et la création des éléments suivants :

- "Add Easy Struts support" : permet d'ajouter les éléments nécessaires à l'utilisation de Struts dans l'application (fichier .jar, .tld, ...) et créer les fichiers de configuration
- "Easy Form" : créer une JSP avec une classe de type ActionForm associée et ajoute la définition de ce bean dans le fichier de configuration

- "Easy Action" : créer une classe de type Action et ajoute la définition du mapping de cette classe dans le fichier de configuration
- "Easy Action associated with a form" : créer une JSP avec une classe de type ActionForm associée, ajoute la définition de ce bean dans le fichier de configuration et crée une classe de type Action et ajoute la définition du mapping de cette classe dans le fichier de configuration
- "Easy Forward" : créer des renvois dédiés à une Action ou globaux
- "Easy Exception" : créer des handler pour les exceptions.
- "Easy Message resources" : créer des fichiers de ressources pour localiser l'application
- Easy Plug-in :
- "Easy Datasource" : permet de définir une source de données qui sera utilisée dans l'application
- Easy Module :

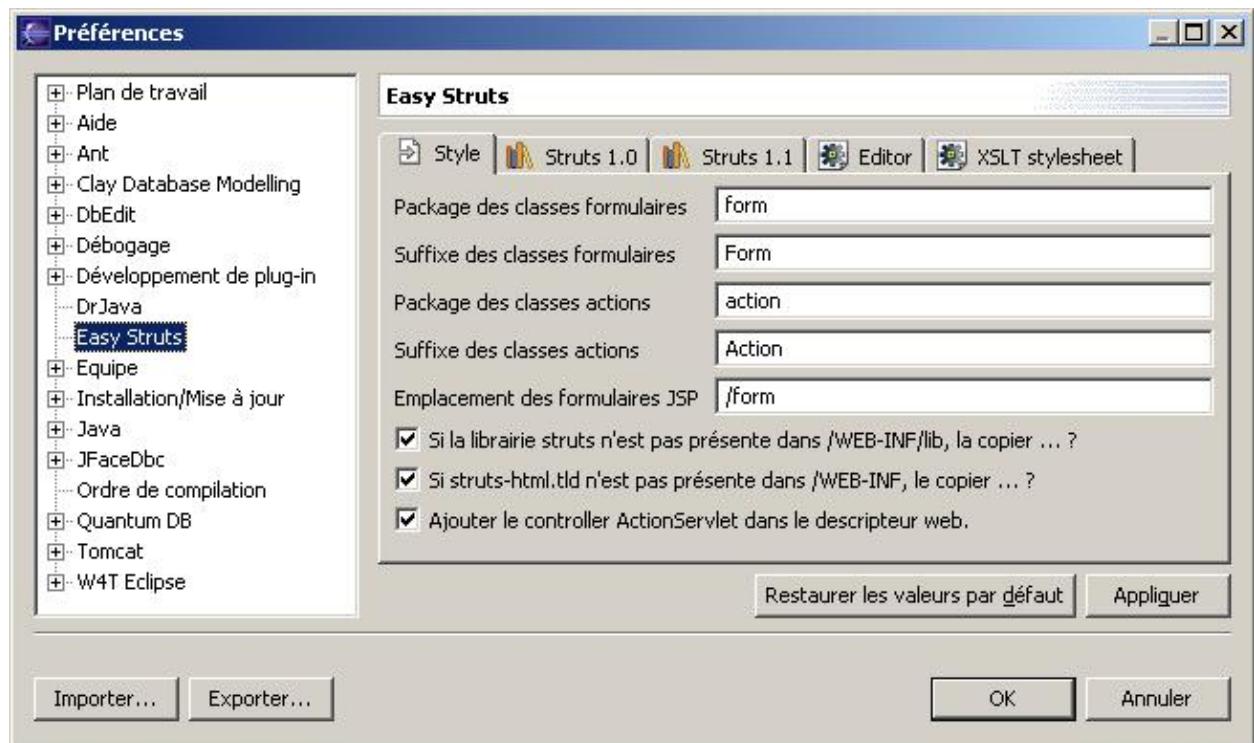
### 23.1.1. Installation et configuration d'Easy Struts

Il y a deux façons pour installer Easy Struts :

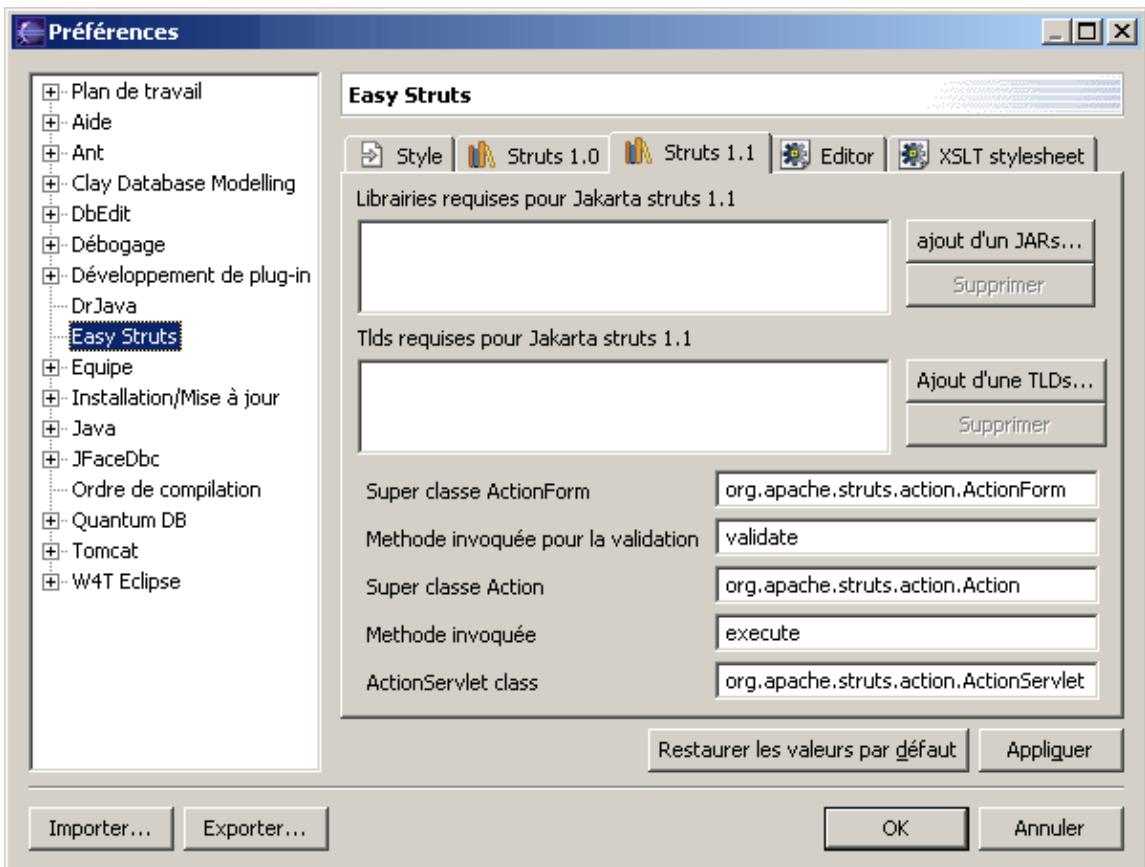
- téléchargez et décompressez le fichier org.easystruts.eclipse\_0.6.4.zip dans le répertoire « plugins » d'Eclipse puis lancer Eclipse.
- utilisez le gestionnaire de mises à jour avec l'url  
<http://easystruts.sourceforge.net/eclipse/updates/site.xml>

Editez les préférences (menu Fenêtre/Préférences) et sélectionnez Easy Struts.

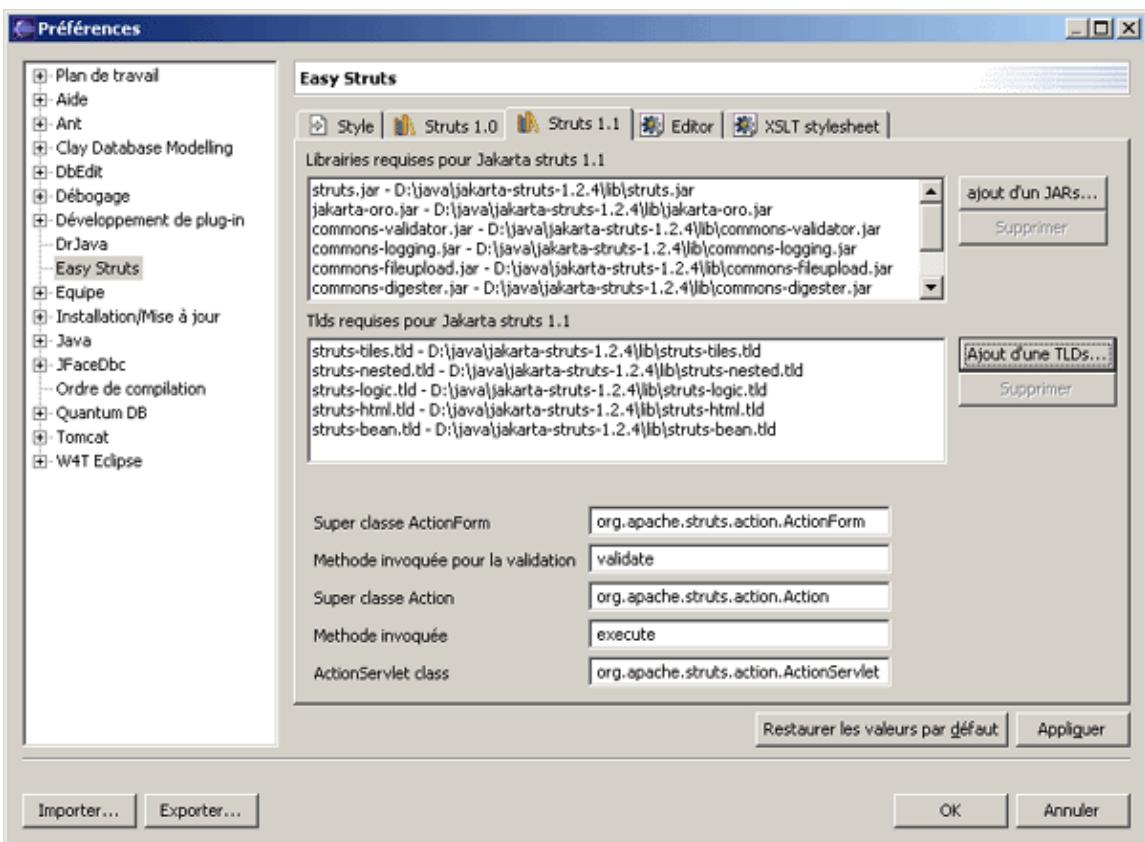
L'onglet « Style » permet de définir des comportements par défaut.



L'onglet « Struts 1.0 » et « Struts 1.1 » permet de fournir l'emplacement requis par la version de Struts désignée par l'onglet.

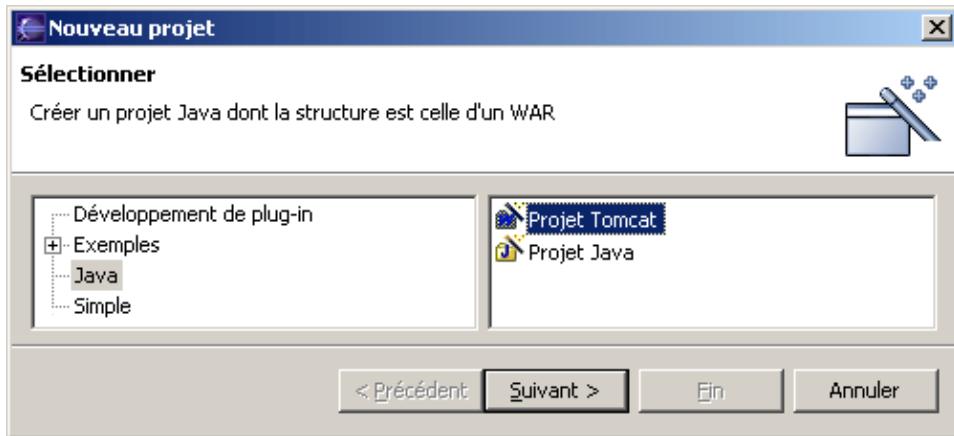


Il faut impérativement ajouter les fichiers .jar requis par Struts ainsi que les fichiers .tld des bibliothèques personnalisées de Struts.

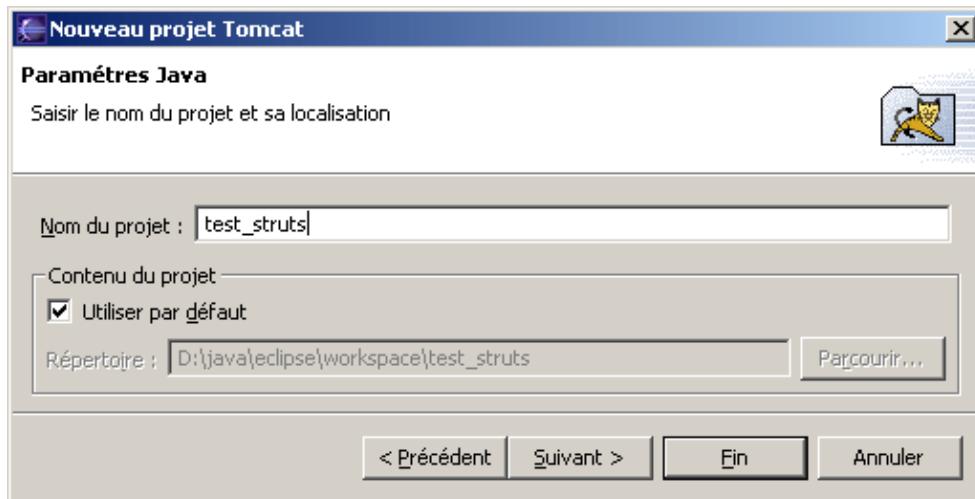


Une fois tous les paramètres renseignés, cliquez sur le bouton « Appliquer » puis sur le bouton « OK ».

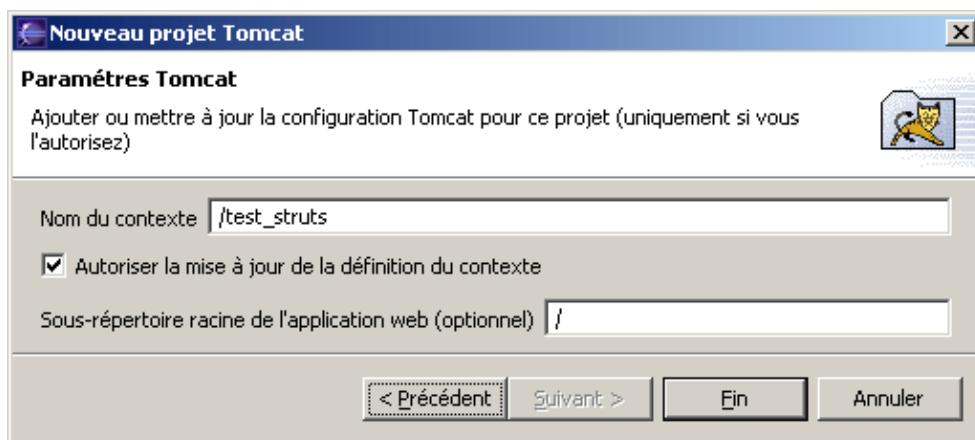
Il faut créer un projet de type « Java/Projet Tomcat » qui va contenir l'application web.



Cliquez sur le bouton « Suivant »

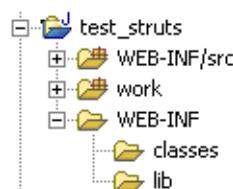


Saisissez le nom du projet, par exemple « test\_struts » et cliquez sur le bouton « Suivant ».

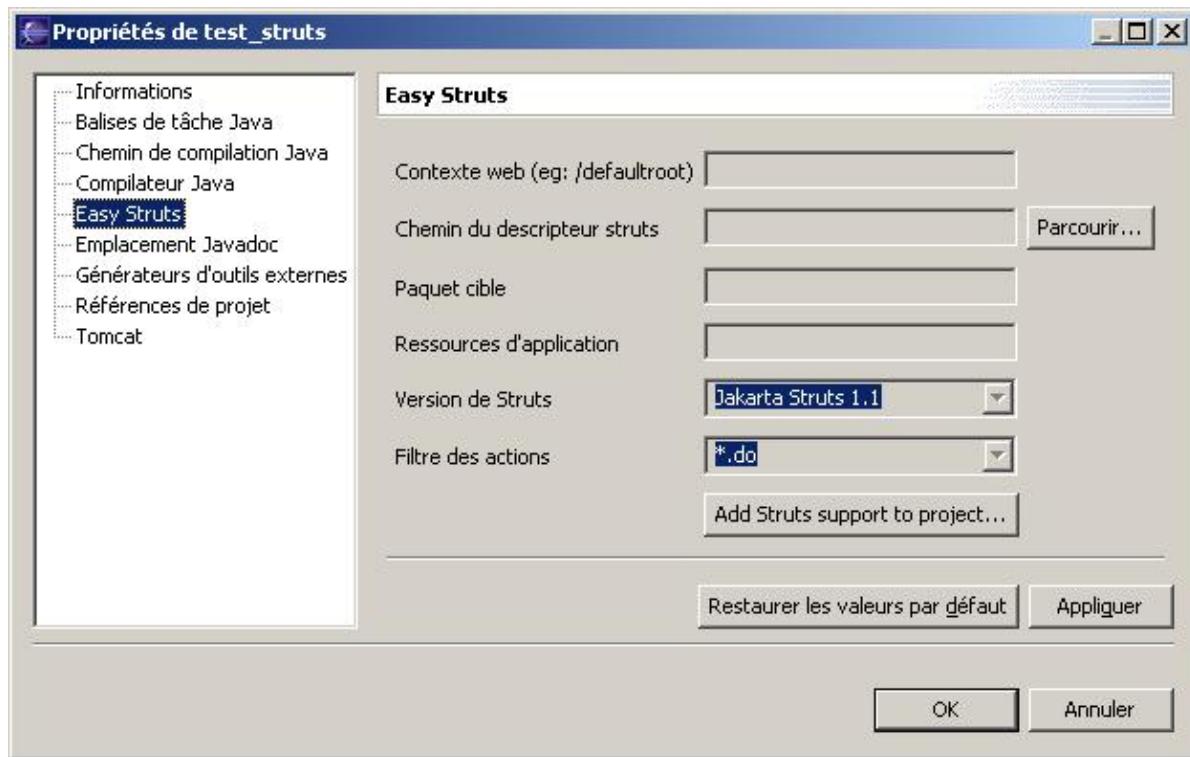


Cliquez sur le bouton « Fin »

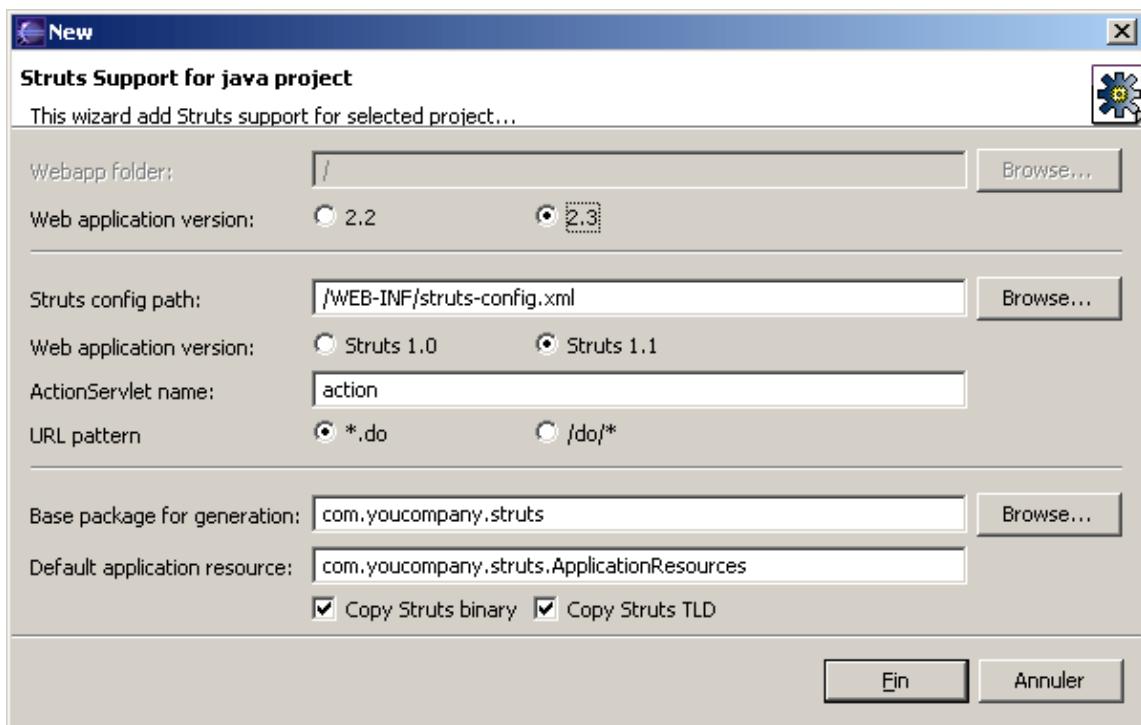
La structure du projet est la suivante :



Il faut sélectionner les propriétés du projet. Sélectionnez l'option « Easy Struts »



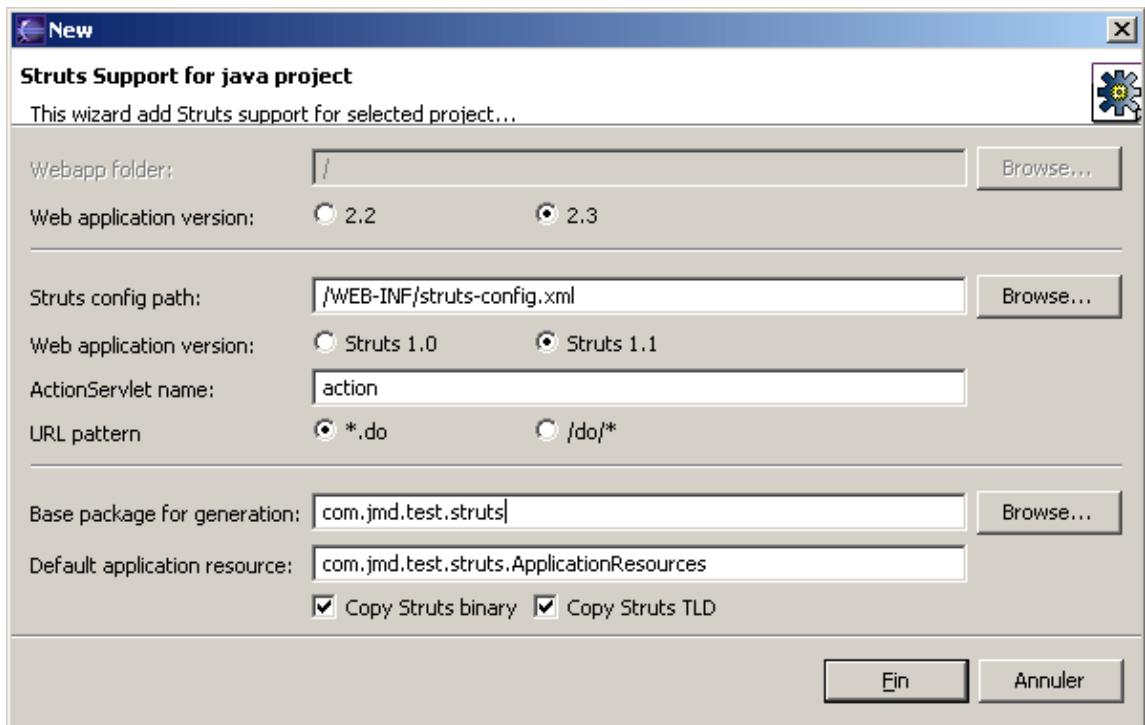
Cliquez sur le bouton « Add Struts support to project ... ». Une boîte de dialogue s'ouvre pour permettre la sélection des options



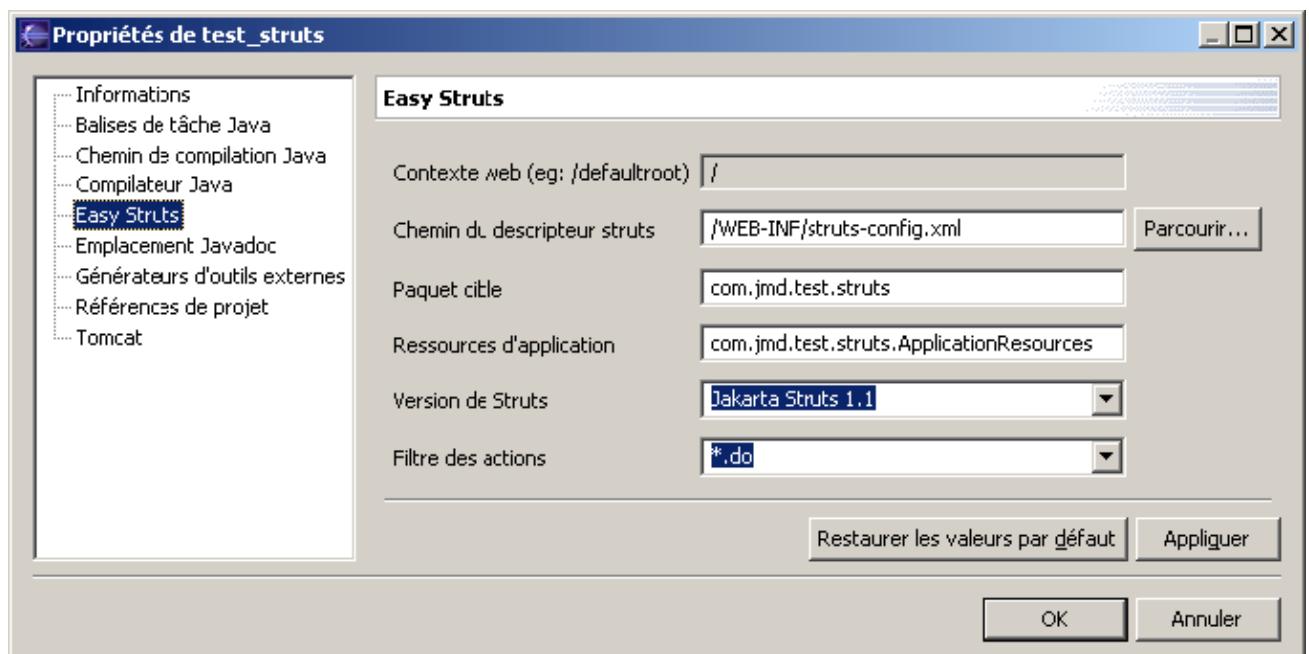
Si les paramètres ne sont pas correctement renseignés, un message d'erreur est affiché



Il faut saisir les informations nécessaires à la configuration de l'application.



Renseignez les informations nécessaires et cliquez sur le bouton « Fin ». Les fichiers sont copiés et les propriétés sont mises à jour.



Cliquez sur le bouton « OK ».

Le fichier web.xml généré est le suivant :

#### Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">
<web-app>
    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
```

```

        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
        <param-name>debug</param-name>
        <param-value>3</param-value>
    </init-param>
    <init-param>
        <param-name>detail</param-name>
        <param-value>3</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
</web-app>

```

Le fichier struts-config.xml généré est le suivant :

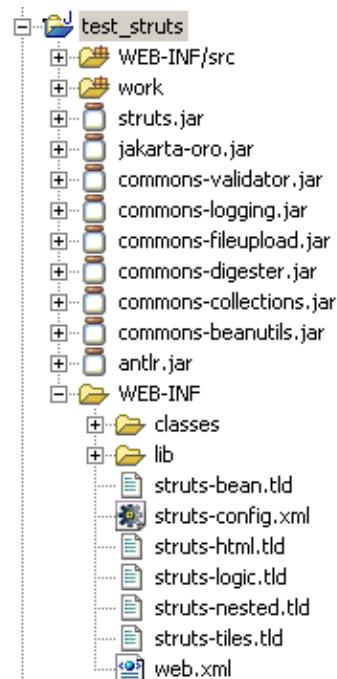
Exemple :

```

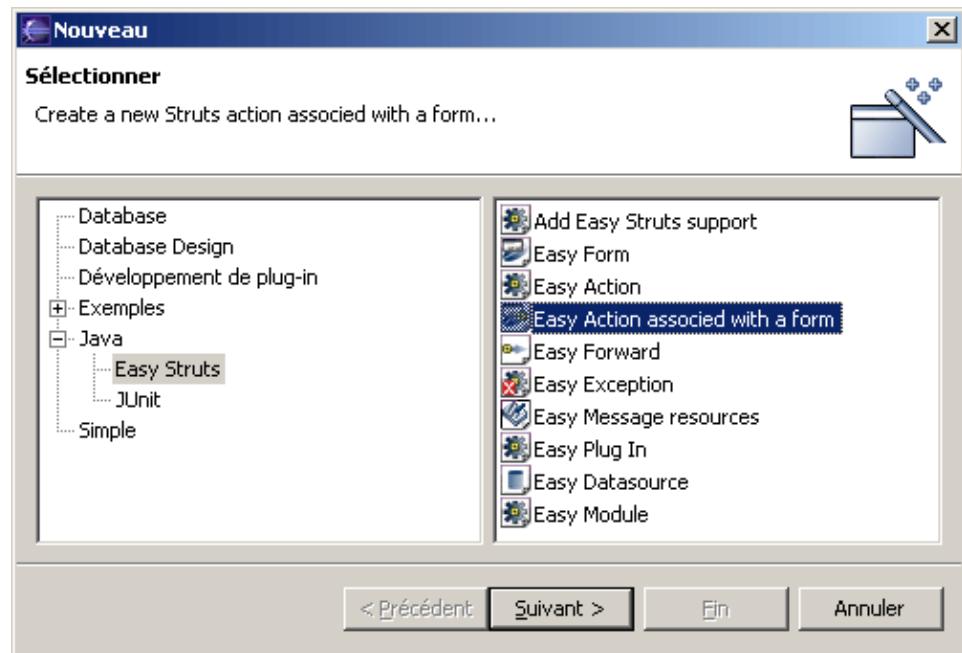
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <data-sources />
    <form-beans />
    <global-exceptions />
    <global-forwards />
    <action-mappings />
    <controller />
    <message-resources parameter="com.jmd.test.struts.ApplicationResources" />
</struts-config>

```

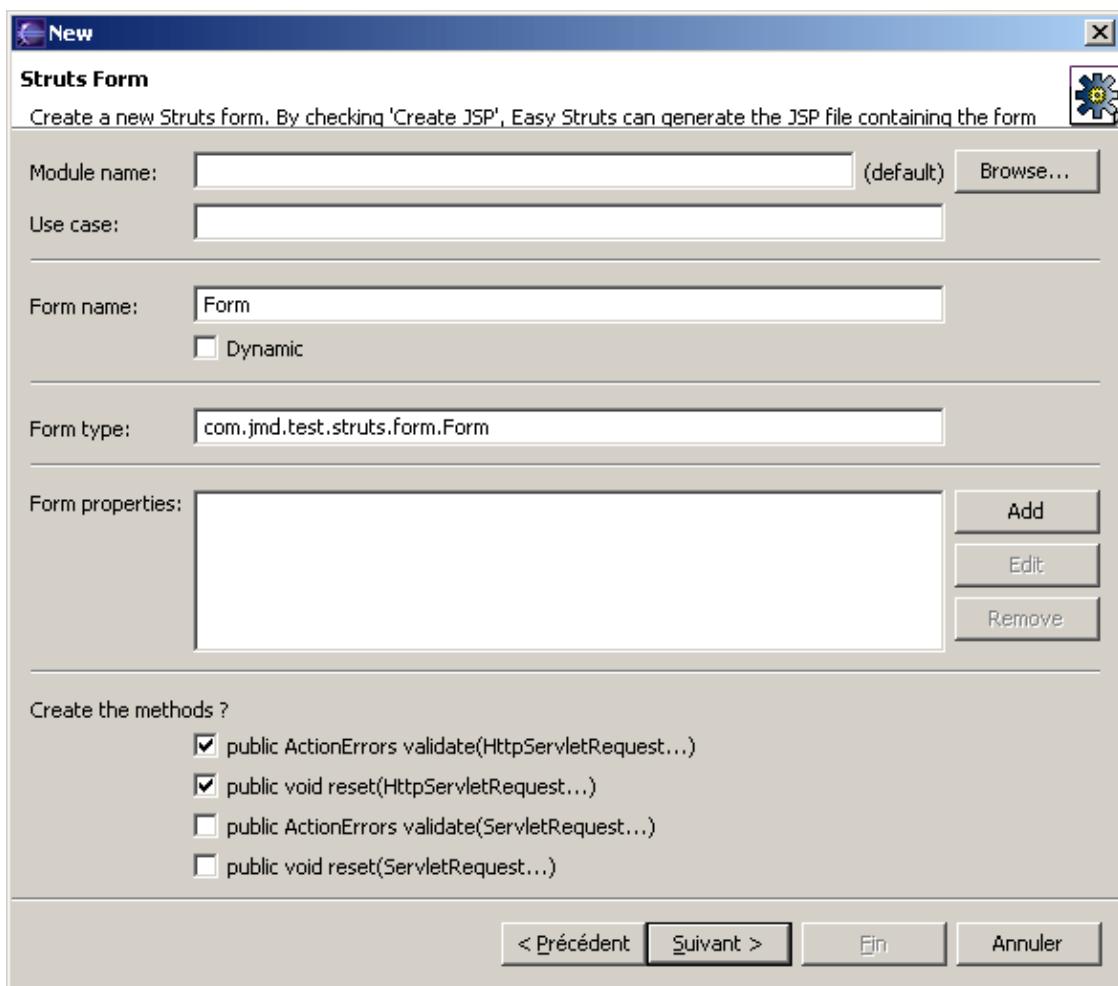
La structure du projet est alors la suivante :



Il faut ensuite créer une nouvelle entité de type « Java / Easy Struts / Easy Action associated with form ».



Cliquez sur le bouton « Suivant ». La page suivante permet de renseigner les caractéristiques de la page.



Renseignez le « Use case » avec le nom logique de la page, par exemple « Login ». Automatiquement les champs Form name et Form Type sont pré-renseignés en fonction du « Use Case » saisi et des propriétés saisies pour Struts.

Pour ajouter des propriétés, il suffit de cliquer sur le bouton « Add »

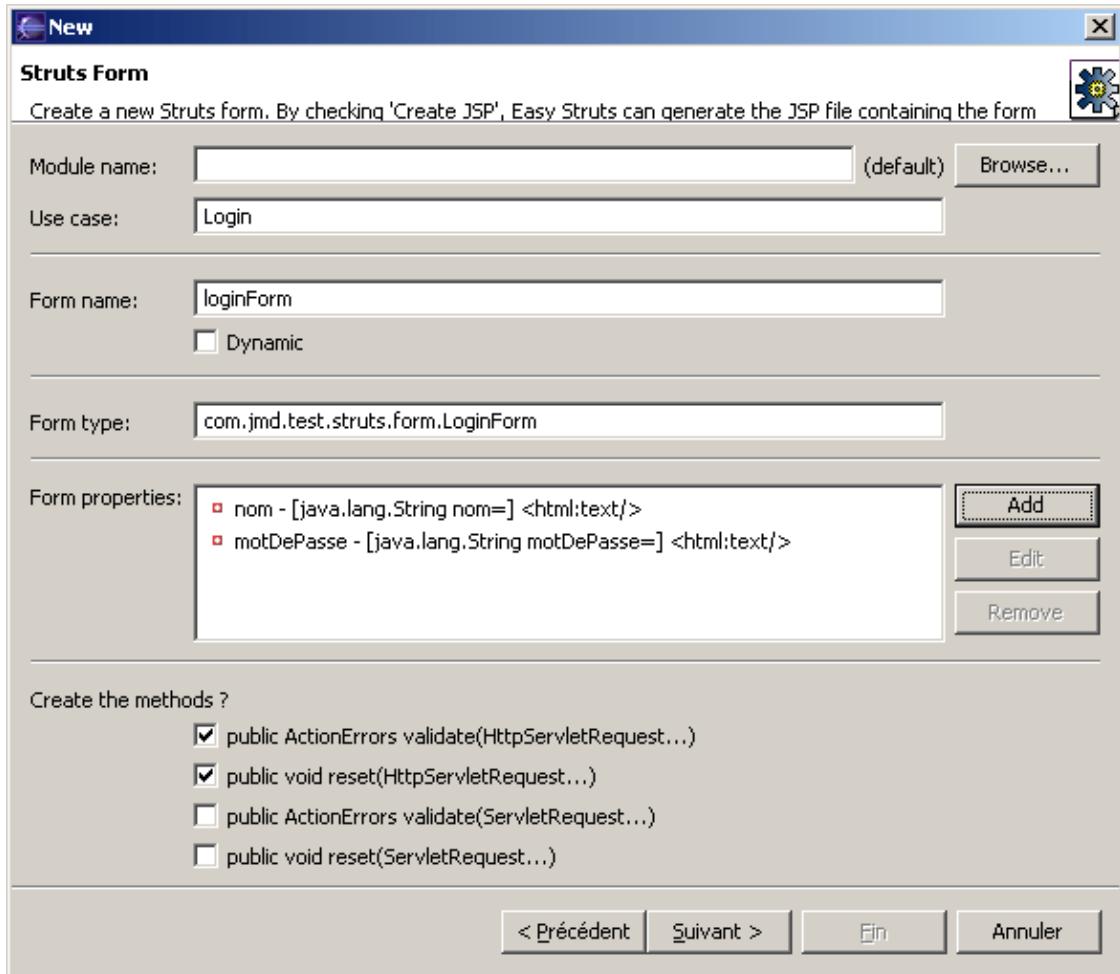


Il suffit alors de renseigner les caractéristiques de la donnée : le nom, le type, la valeur initiale et le mode de saisie dans la JSP.

Pour valider chaque donnée, il faut cliquer sur le bouton « OK ». Une fois la saisie terminée, il faut cliquer sur le bouton « Fin ».

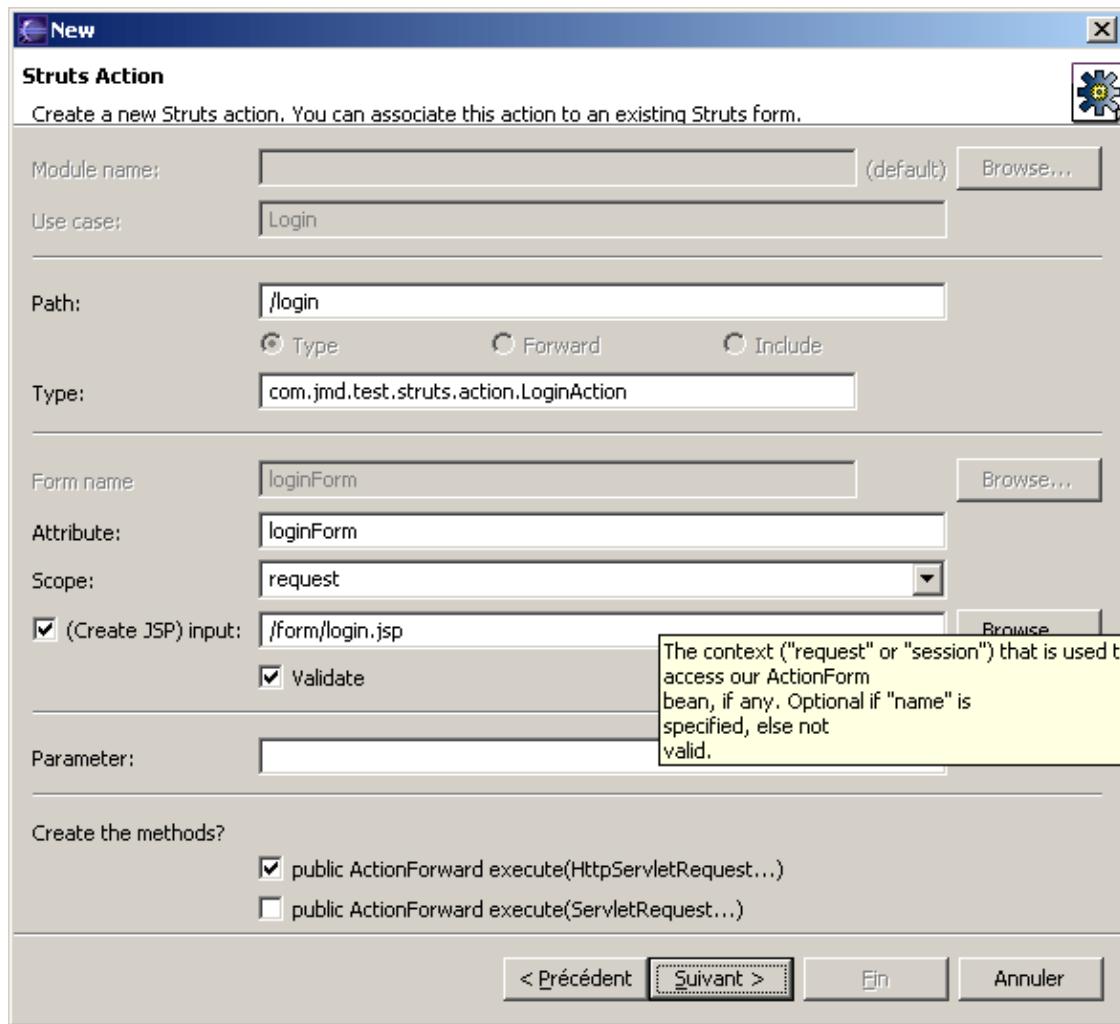
Dans l'exemple de cette section, deux données sont ajoutées :

- nom de type String
- motDePasse de type String



Cliquez sur le bouton « suivant »

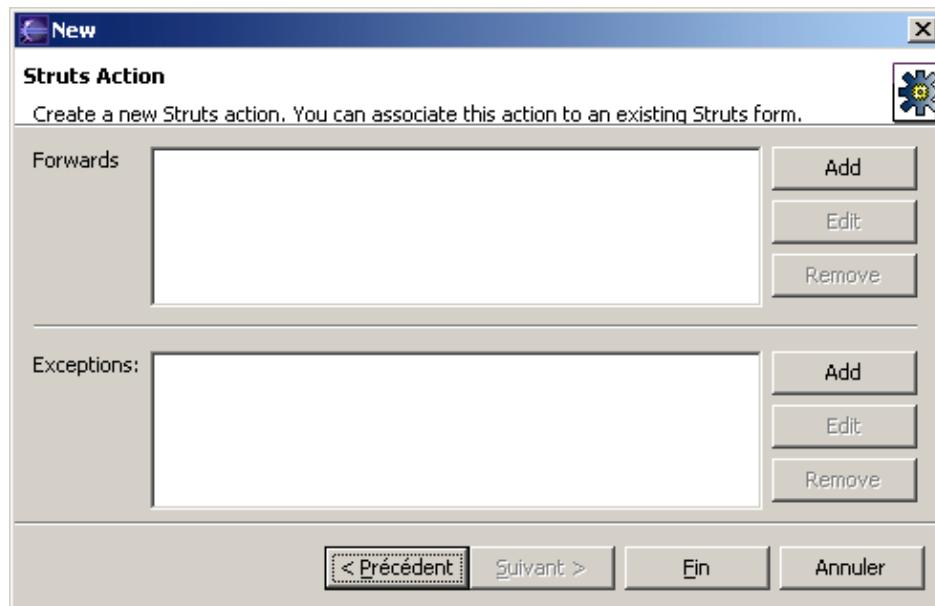
La page suivante permet de saisir les renseignements sur la classe Action qui sera générée.



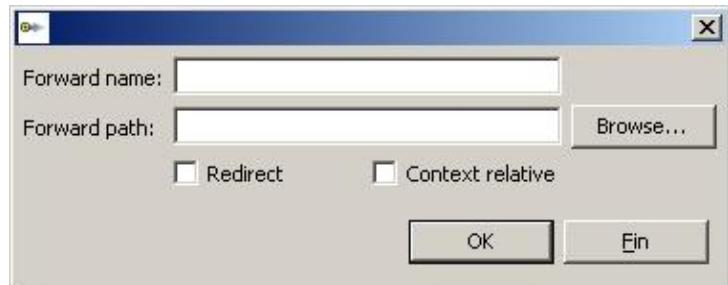
Les divers renseignements saisis vont permettre de créer la JSP et de modifier le fichier struts-config.xml

Une fois les données saisies, cliquez sur le bouton « Suivant ».

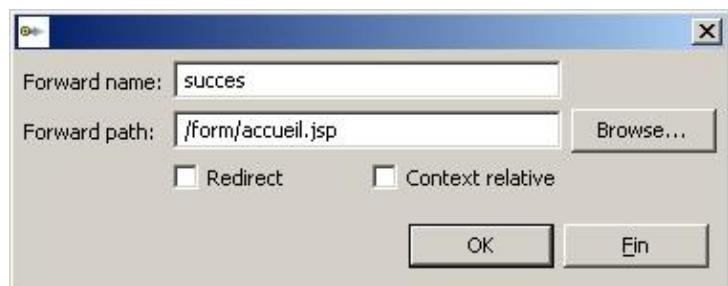
La page suivante permet de préciser les renvois et les captures d'exceptions associés pour l>Action.



Pour ajouter un renvoi (forward), il faut cliquer sur le bouton « Add » correspondant.

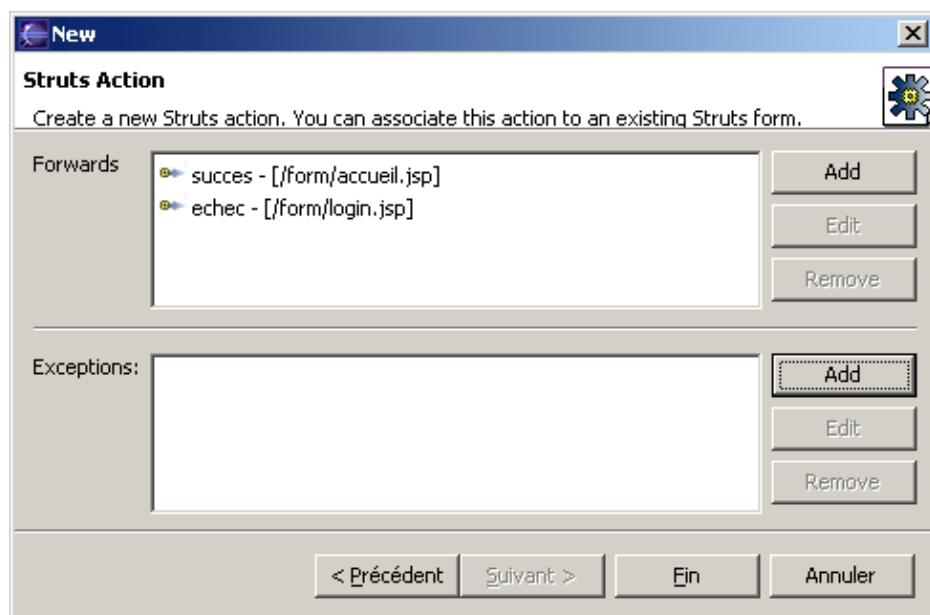


Il faut alors saisir le nom et le chemin du renvoi.



Une fois les renvois ajoutés, cliquez sur le bouton « Fin ».

Si nécessaire, procédez de la même façon avec les captures d'exceptions.



Cliquez sur le bouton « Fin ».

Le fichier struts-config.xml est modifié pour tenir compte des éléments paramétrés dans l'assistant.

#### Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <!-- ===== Data Source Configuration ===== -->
    <data-sources />
    <!-- ===== Form Bean Definitions ===== -->
    <form-beans>
        <form-bean name="loginForm" type="com.jmd.test.struts.form.LoginForm">
            <form-property name="motDePasse" type="java.lang.String" />
    </form-beans>
</struts-config>
```

```

        <form-property name="nom" type="java.lang.String" />
    </form-bean>
</form-beans>
<!-- ===== Global Exception Definitions ===== -->
<global-exceptions />
<!-- ===== Global Forward Definitions ===== -->
<global-forwards />
<!-- ===== Action Mapping Definitions ===== -->
<action-mappings>
    <action
        attribute="loginForm"
        input="/form/login.jsp"
        name="loginForm"
        path="/login"
        type="com.jmd.test.struts.action.LoginAction">
        <forward name="succes" path="/form/accueil.jsp" />
        <forward name="echec" path="/form/login.jsp" />
    </action>
</action-mappings>
<!-- ===== Controller Configuration ===== -->
<controller />
<!-- ===== Message Resources Definitions ===== -->
<message-resources parameter="com.jmd.test.struts.ApplicationResources" />
<!-- ===== Plug Ins Configuration ===== -->
</struts-config>

```

Le fichier login.jsp est créé dans le répertoire form

#### Exemple :

```

<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<html>
    <head>
        <meta name="Generator" content="Easy Struts Xslt generator for Eclipse.">
        <title>Struts Form for loginForm</title>
    </head>
    <body>
        <html:form action="/login">
            motDePasse : <html:text property="motDePasse"/>
            <html:errors property="motDePasse"/><br>
            nom : <html:text property="nom"/><html:errors property="nom"/><br>
            <html:submit/><html:cancel/>
        </html:form>
    </body>
</html>

```

Il faut modifier le fichier LoginForm.java pour mettre le code à exécuter lors de la validation des données dans la méthode validate() à la place de la levée de l'exception de type UnsupportedOperationException

#### Exemple :

```

public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
    ActionErrors erreurs = new ActionErrors();
    if (nom == null || nom.equals("")) {
        erreurs.add("nom", new ActionError("error.login.nommanquant"));
    }
    return erreurs;
}

```

Il faut ajouter une clause d'importation sur la classe org.apache.struts.action.ActionError

Il faut modifier le fichier LoginAction.java pour mettre le code à exécuter lors des traitements dans la méthode execute() à la place de la levée de l'exception de type UnsupportedOperationException

Exemple :

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) throws Exception {
    ActionForward resultat = null;
    LoginForm loginForm = (LoginForm) form;
    String nom = loginForm.getNom();
    String mdp = loginForm.getMotDePasse();
    request.setAttribute("nom", nom);
    if (nom.equals("test") && mdp.equals("test")) {
        resultat = (mapping.findForward("succes"));
    } else {
        resultat = (mapping.findForward("echec"));
    }
    return resultat;
}
```

Il faut modifier le fichier ApplicationResources.properties se trouvant dans le répertoire WEB-INF/classes/com/jmd/test/struts

Exemple :

```
# Resources for parameter 'com.jmd.test.struts.ApplicationResources'
# Project P/test_struts
errors.header=<ul>
errors.footer=</ul>
error.login.nommanquant=<li>La saisie du nom de l'utilisateur est obligatoire</li>
```

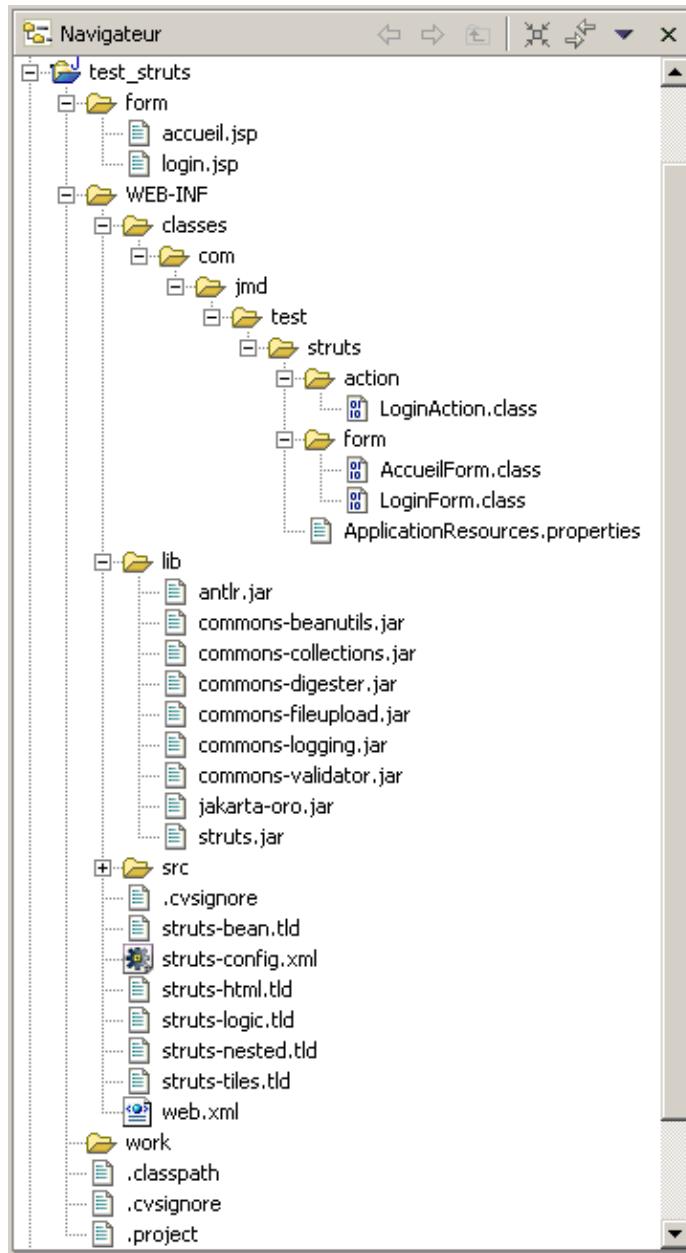
Il ensuite créer une nouvelle entité de type « Java / Easy Struts / Easy Form » dont le « Use Case » sera « Accueil ».

Il faut modifier le fichier form/accueil.jsp généré.

Exemple :

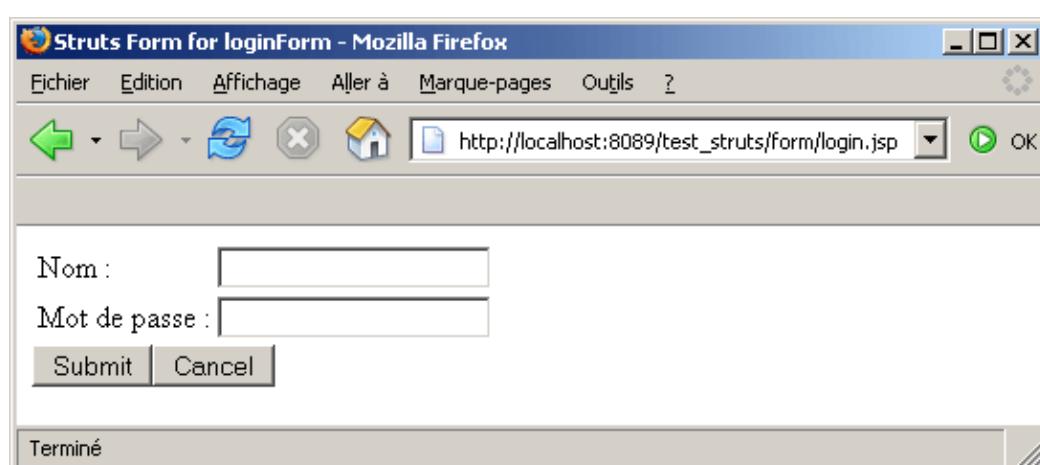
```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<html>
    <head>
        <meta name = "Generator"
              content = "Easy Struts Xslt generator for Eclipse.">
        <title>Struts Form for accueilForm</title>
    </head>
    <body>
        <H1>Bienvenue<logic:present name="nom" scope="request">
            <bean:write name="nom" scope="request"/>
        </logic:present></H1>
    </body>
</html>
```

Les fichiers composants l'application sont les suivants :

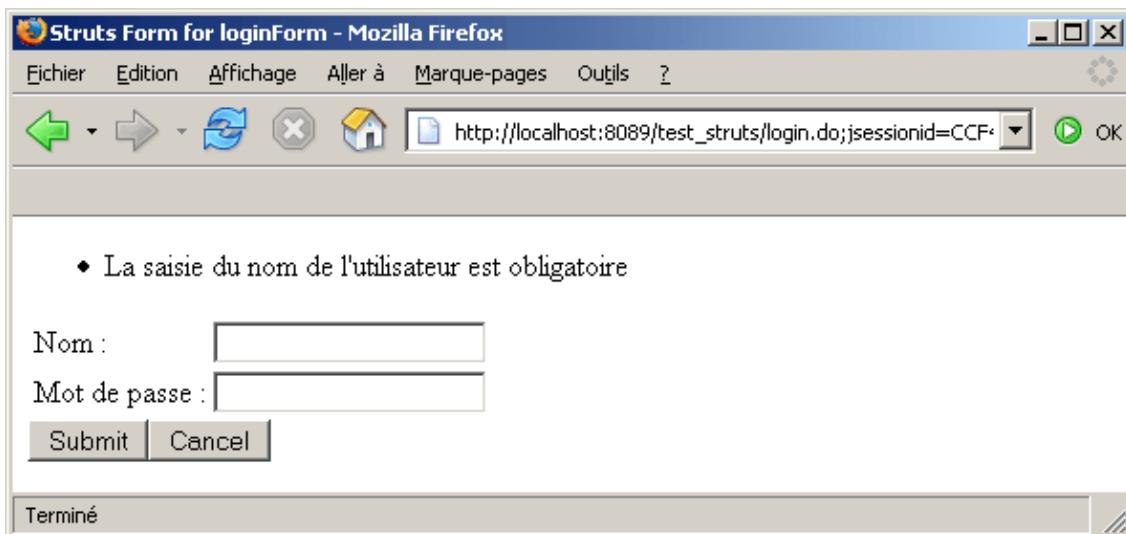


### 23.1.2. L'exécution de l'application

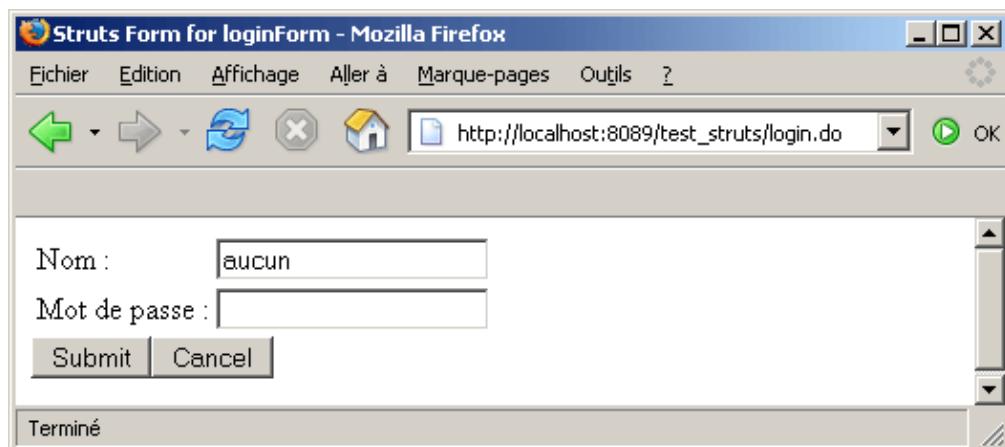
Il faut lancer Tomcat en cliquant sur le bouton



Un clic sur le bouton « Submit » sans saisir de nom affiche le message d'erreur



La validation avec un nom différent de test réaffiche la page

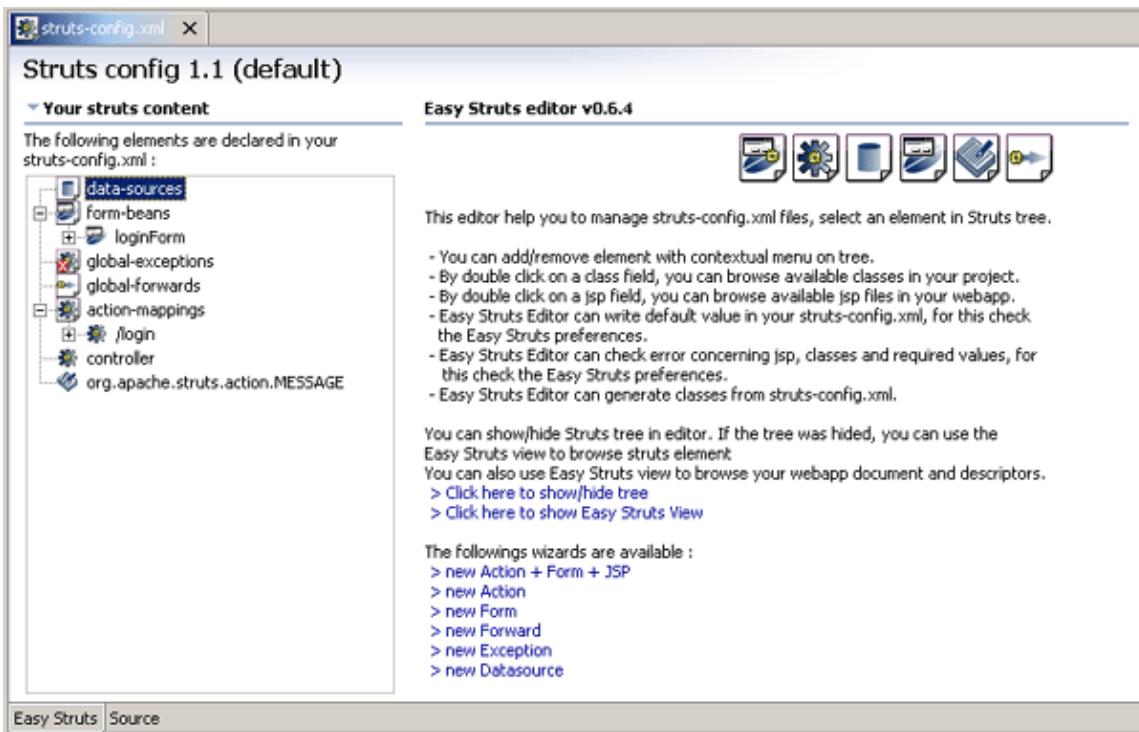


La validation avec le nom « test » et le mot de passe « test » affiche la page d'accueil



### 23.1.3. La modification du fichier struts-config.xml

EasyStruts propose un éditeur dédié à la mise à jour du fichier de configuration de Struts

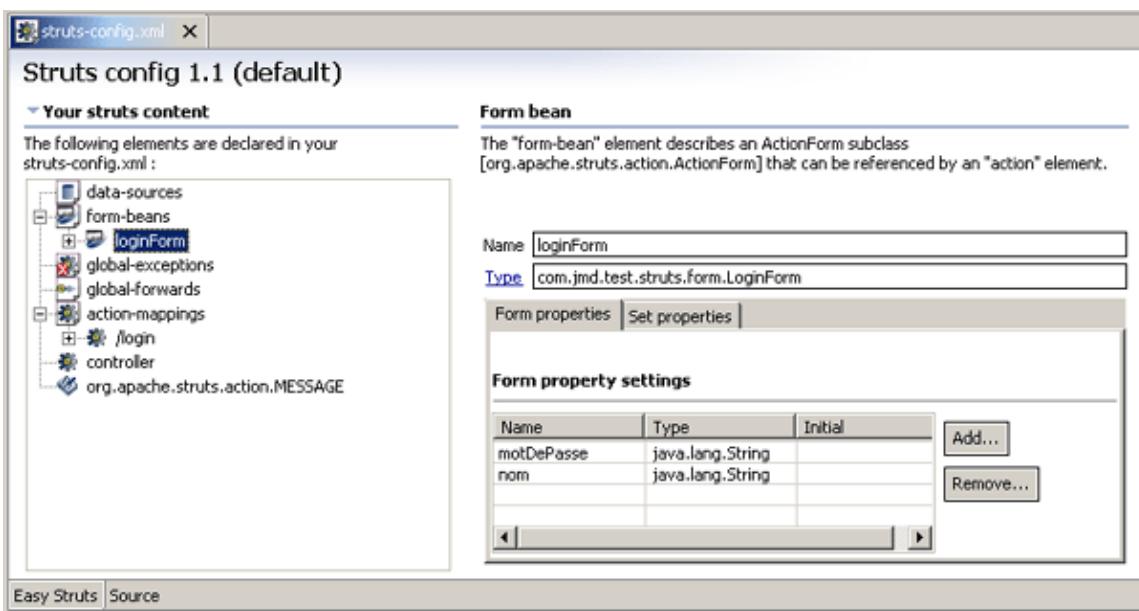


Cet éditeur propose deux onglets : « Easy Struts » pour modifier les informations avec une interface et « Source » qui permet de visualiser et de modifier directement le fichier xml.

L'onglet « Easy Struts » de compose de deux parties :

La partie de gauche permet une navigation dans les éléments qui composent le fichier xml et la sélection d'un élément.

La partie de droite permet de modifier l'élément sélectionné.



# Chapitre 24

Java Server Faces (JSF) est une technologie dont le but est de proposer un framework qui facilite et standardise le développement d'applications web avec Java. Son développement a tenu compte des différentes expériences acquises lors de l'utilisation des technologies standards pour le développement d'applications web (servlet, JSP, JSTL) et de différents frameworks (Struts, ...).

Le grand intérêt de JSF est de proposer un framework qui puisse être mis en oeuvre par des outils pour permettre un développement de type RAD pour les applications web et ainsi faciliter le développement des applications de ce type.

Ce chapitre contient la section :

- ♦ Utilisation de JSF sans plug-in dédié

### 24.1. Utilisation de JSF sans plug-in dédié

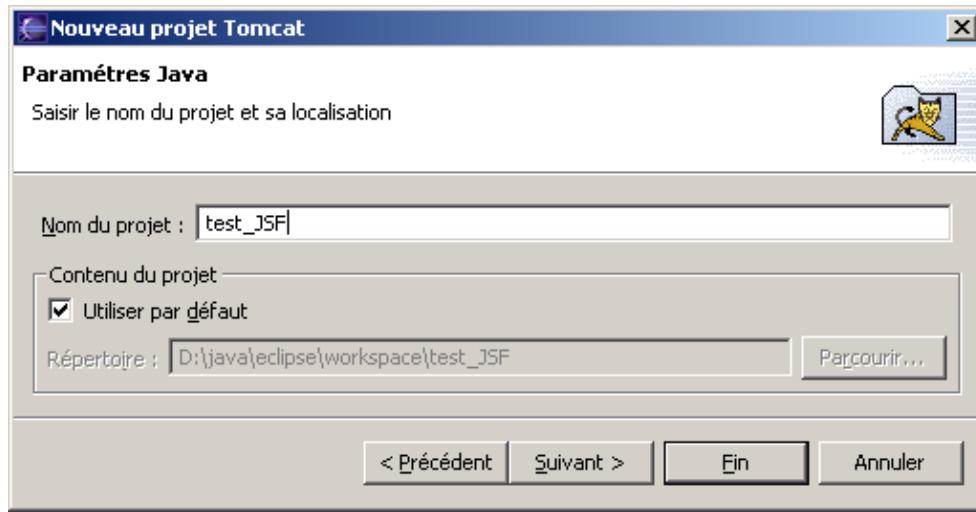
En attendant un plug-in fonctionnel dédié à l'utilisation de JSF dans Eclipse, il est possible de développer une application Web utilisant JSF avec Eclipse. Cependant, toutes les opérations doivent être réalisées "à la main".

Cette section va développer une petite application constituée de deux pages. La première va demander le nom de l'utilisateur et la seconde afficher un message de bienvenue en mettant en oeuvre les outils suivants sous Windows :

Outil	Version	Rôle
JDK	1.4.2_03	
Eclipse	2.1.3	IDE
Tomcat	5.0.28	conteneur web
plug-in Tomcat de Sysdeo	3	Arrêt et démarrage de Tomcat
JSF implémentation de référence de Sun	1.1_01	

#### 24.1.1. Création du projet

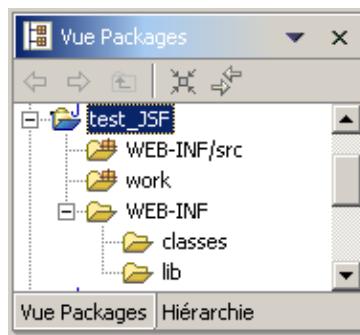
Il faut créer un nouveau projet de type « Java / Projet Tomcat ».



Saisissez le nom du projet, par exemple test\_JSF et cliquez sur le bouton « Suivant ».



Cliquez sur le bouton « Fin ». L'assistant va créer un nouveau projet possédant une structure de répertoire typique pour une application Web.



Il faut ensuite copier les fichiers nécessaires à une utilisation de JSF dans l'application web.

Il suffit à partir d'un explorateur de fichier de faire un cliquer/glisser des fichiers \*.jar du répertoire lib de l'implémentation de référence vers le répertoire WEB-INF/lib du projet. Attention, il est aussi nécessaire d'ajouter les fichier jstl.jar et standard.jar qui contiennent l'implémentation de la JSTL.

#### 24.1.2. Création des éléments qui composent l'application

Il faut créer une nouvelle entité de type « Simple / Fichier » à la racine du projet et la nommer index.htm

### Exemple :

```
<html>
<head>
    <meta http-equiv="Refresh" content= "0; URL=login.faces" />
    <title>Demarrage de l'application</title>
</head>
<body>
    <p>Démarrage de l'application ...</p>
</body>
</html>
```

Il faut créer une nouvelle entité de type « Simple / Fichier » à la racine du projet et la nommer login.jsp

### Exemple :

```
<html>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<head>
    <title>Application de tests avec JSF</title>
</head>
<body>
    <h:form>
        <h3>Identification</h3>
        <table>
            <tr>
                <td>Nom : </td>
                <td><h:inputText value="#{login.nom}" /></td>
            </tr>
            <tr>
                <td>Mot de passe :</td>
                <td><h:inputSecret value="#{login.mdp}" /></td>
            </tr>
            <tr>
                <td colspan="2"><h:commandButton value="Login" action="login"/></td>
            </tr>
        </table>
    </h:form>
</body>
</f:view>
</html>
```

Il faut créer une nouvelle classe nommée com.jmd.test.jsf.LoginBean

### Exemple :

```
package com.jmd.test.jsf;

public class LoginBean {
    private String nom;
    private String mdp;

    public String getMdp() {
        return mdp;
    }

    public String getNom() {
        return nom;
    }

    public void setMdp(String string) {
        mdp = string;
    }
}
```

```

    }

    public void setNom(String string) {
        nom = string;
    }
}

```

Il faut créer une nouvelle entité de type « Simple / Fichier » à la racine du projet et la nommer accueil.jsp

**Exemple :**

```

<html>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<head>
    <title>Page d'accueil de l'application</title>
</head>
<body>
    <h:form>
        <h3>Bienvenue <h:outputText value="#{login.nom}" />, </h3>
    </h:form>
</body>
</f:view>
</html>

```

Il faut créer une nouvelle entité de type « Simple / Fichier » dans le répertoire /WEB-INF et la nommer faces-config.xml

**Exemple :**

```

<?xml version="1.0"?>
<!DOCTYPE faces-config PUBLIC
"-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
"http://java.sun.com/dtd/web-facesconfig_1_0.dtd">
<faces-config>
    <navigation-rule>
        <from-view-id>/login.jsp</from-view-id>
        <navigation-case>
            <from-outcome>login</from-outcome>
            <to-view-id>/accueil.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <managed-bean>
        <managed-bean-name>login</managed-bean-name>
        <managed-bean-class>com.jmd.test.jsf.LoginBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>
</faces-config>

```

Il faut créer une nouvelle entité de type « Simple / Fichier » dans le répertoire /WEB-INF et la nommer web.xml

**Exemple :**

```

<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet>

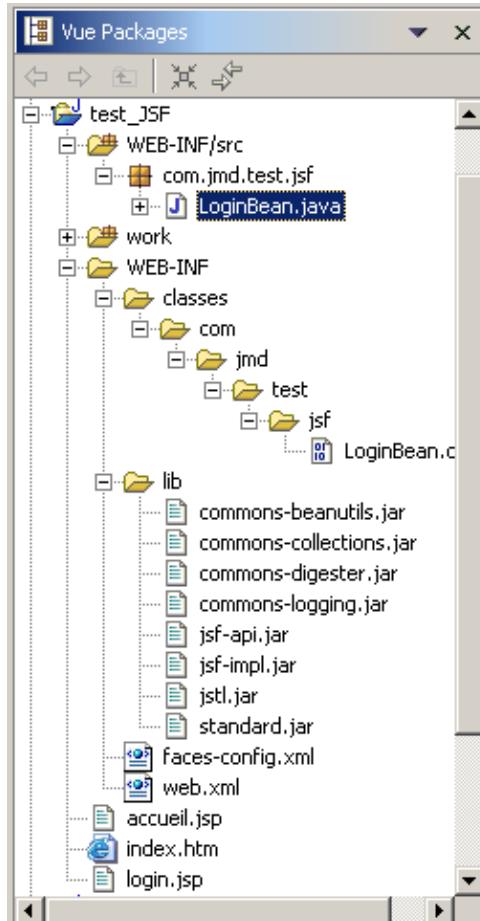
```

```

<servlet-name>Faces Servlet</servlet-name>
<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.faces</url-pattern>
</servlet-mapping>
<welcome-file-list>
  <welcome-file>index.htm</welcome-file>
</welcome-file-list>
</web-app>

```

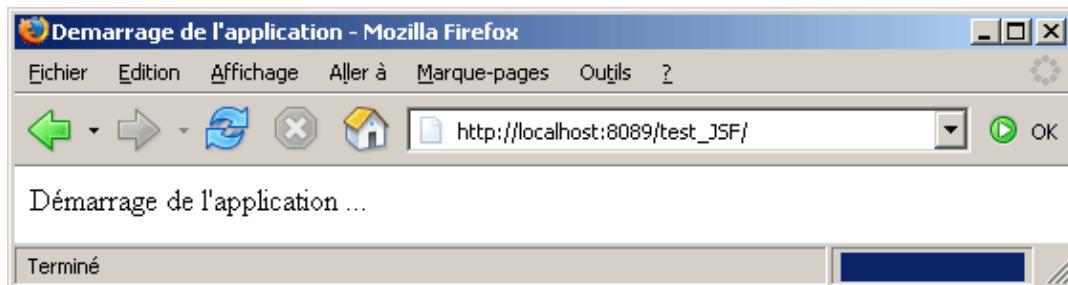
Les éléments qui composent le projet sont donc les suivants :



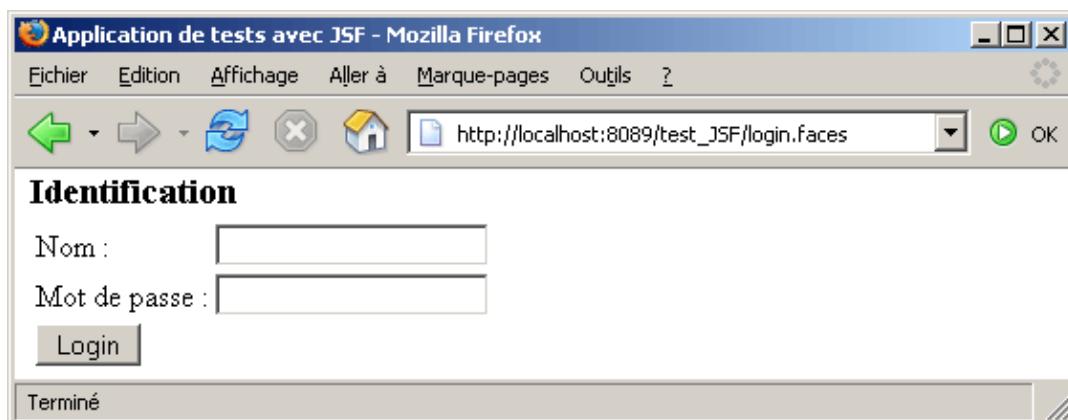
### 24.1.3. Exécution de l'application

Il suffit alors de démarrer Tomcat en cliquant sur le bouton .

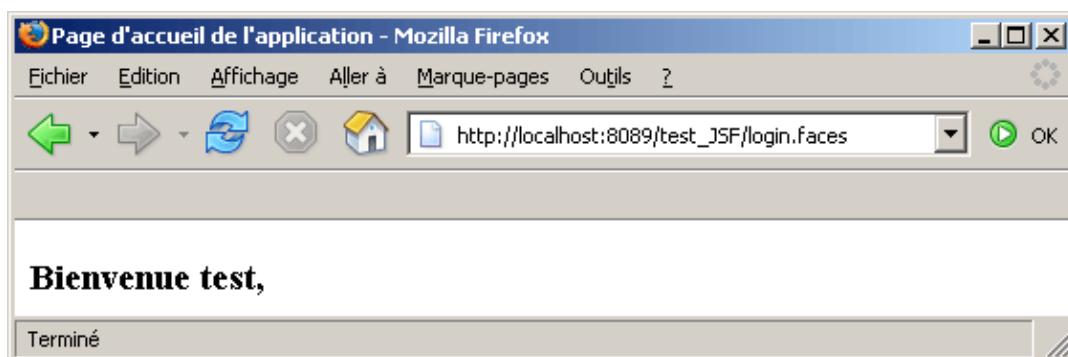
Une fois Tomcat démarré, ouvrir un navigateur et taper l'url [http://localhost:8089/test\\_JSF/](http://localhost:8089/test_JSF/) (en remplaçant le port 8089 par celui défini dans Tomcat).



Une fois l'application démarrée, la page de login s'affiche



Il faut saisir un nom par exemple test et cliquer sur le bouton « Login ».



Cet exemple ne met pas en valeur la puissance de Java Server Faces mais propose simplement de mettre en oeuvre le minimum pour développer des applications utilisant JSF avec Eclipse.

## 25. EJB et Eclipse

# Chapitre 25

Eclipse en standard ne propose aucune fonctionnalité particulière pour développer des EJB. Il est cependant possible d'utiliser un plug-in pour faciliter le développement des EJB. C'est notamment le cas du plug-in WTP.

Ce chapitre contient plusieurs sections :

- ◆ [Le développement d'EJB avec le plug-in WTP 1.5](#)
- ◆ [Le développement d'EJB avec le plug-in Lomboz 2.1](#)

### 25.1. Le développement d'EJB avec le plug-in WTP 1.5

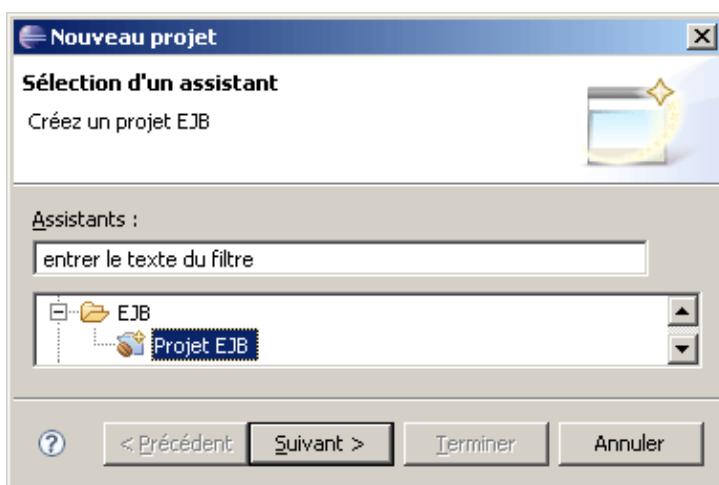
Le plug-in WTP 1.5 propose des assistants et l'utilisation de XDoclet pour développer des EJB en version 2.1.

Cette section va mettre en oeuvre les outils suivants sous Windows :

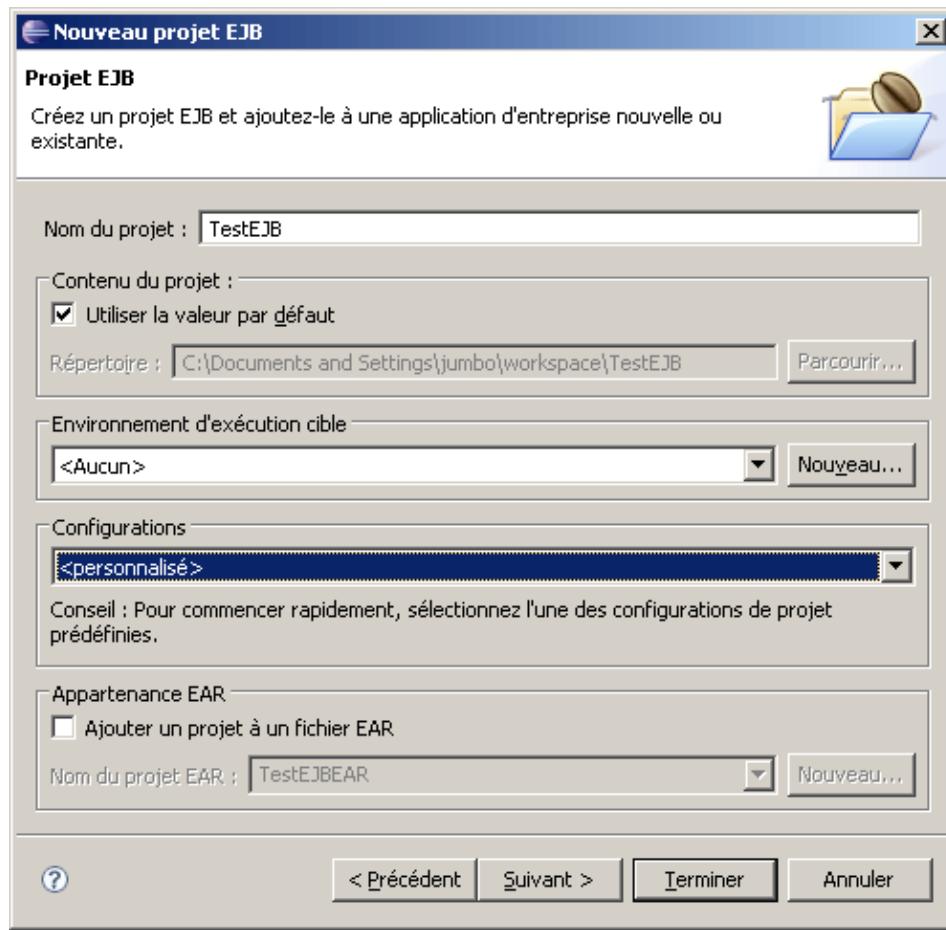
Outil	Version	Rôle
JDK	1.5.0_7	
Eclipse	3.2	IDE
JBoss	4.0.4.GA	conteneur EJB

#### 25.1.1. La création d'un projet de type EJB

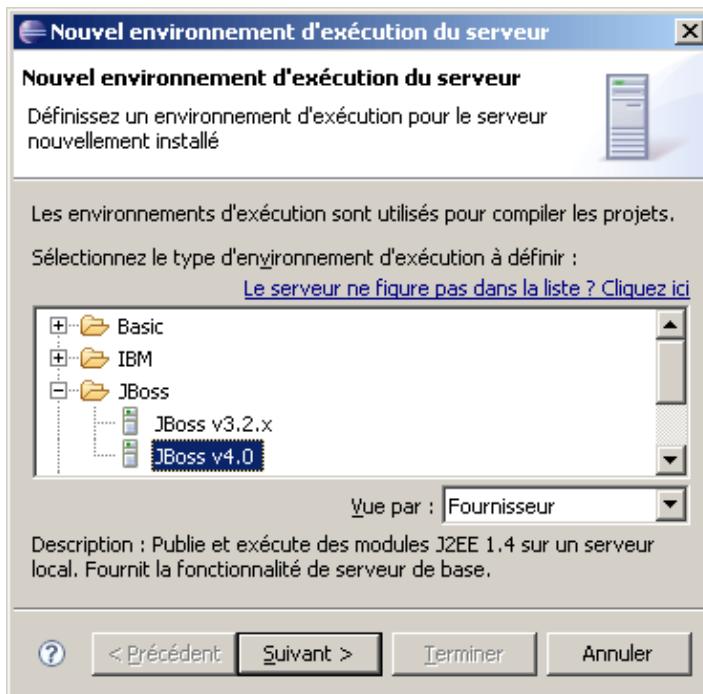
Créez un nouveau projet de type « EJB / Projet EJB »



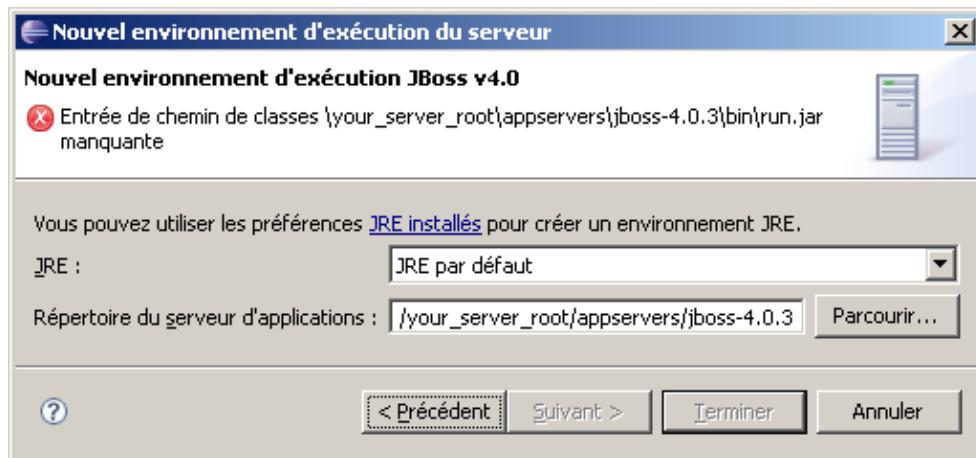
Cliquez sur le bouton « Suivant »



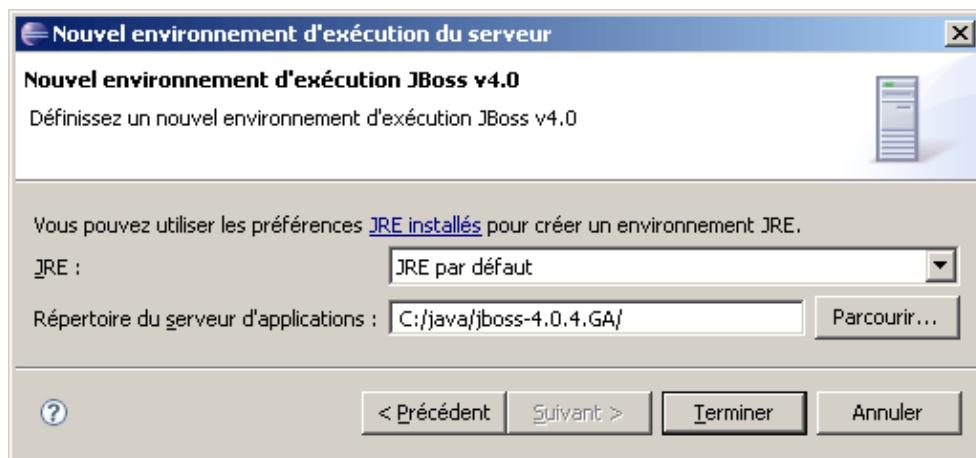
Saisissez le nom du projet, cochez la case « Ajouter à un fichier EAR » et sélectionnez l'environnement d'exécution. Si aucun n'est défini cliquez sur le bouton « Nouveau ... ».



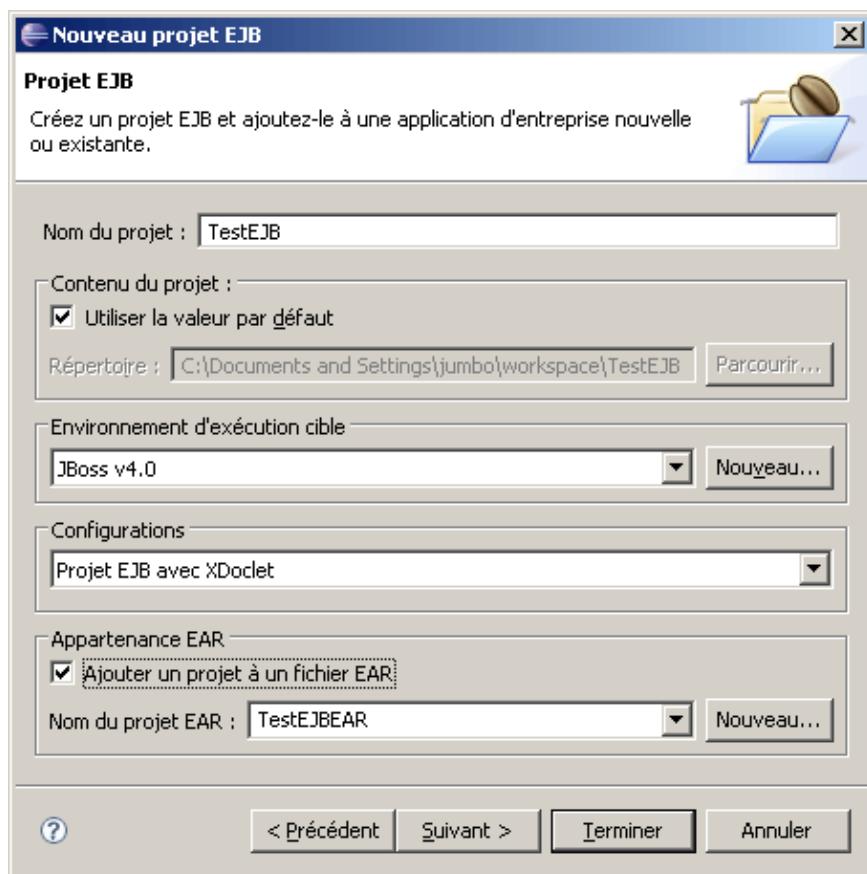
Sélectionnez le type de serveur « JBoss / JBoss v4.0 » dans l'exemple de cette section et cliquez sur le bouton « Suivant ».



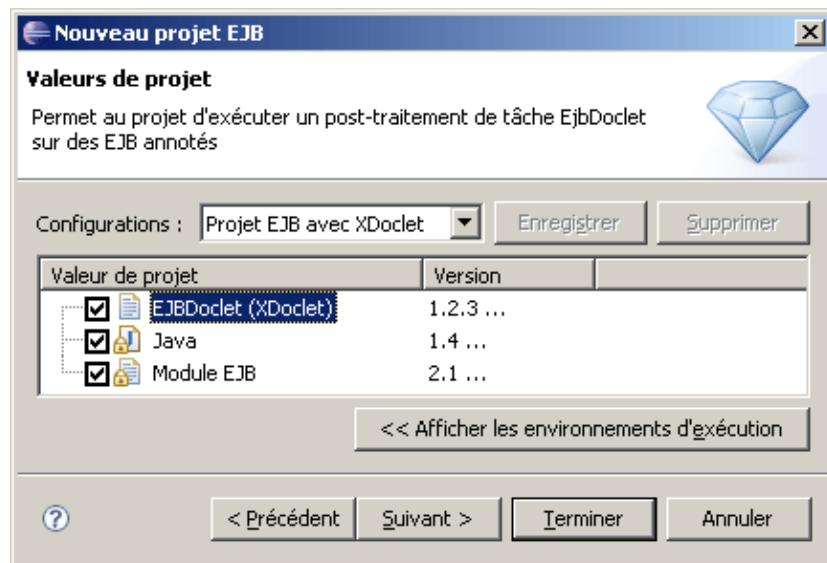
Sélectionnez le répertoire contenant le JBoss en cliquant sur le bouton « Parcourir ».



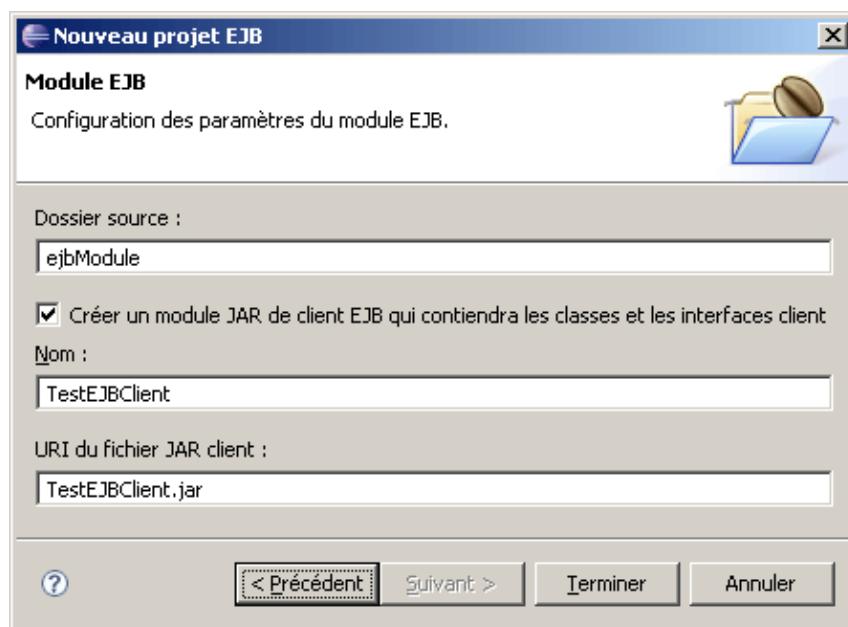
cliquez sur le bouton « Terminer »



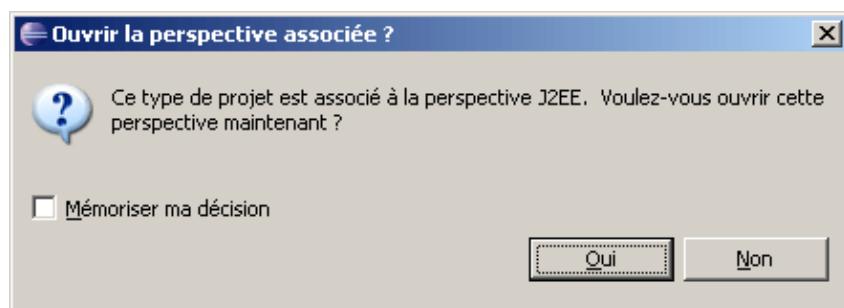
Cliquez sur le bouton « Suivant »



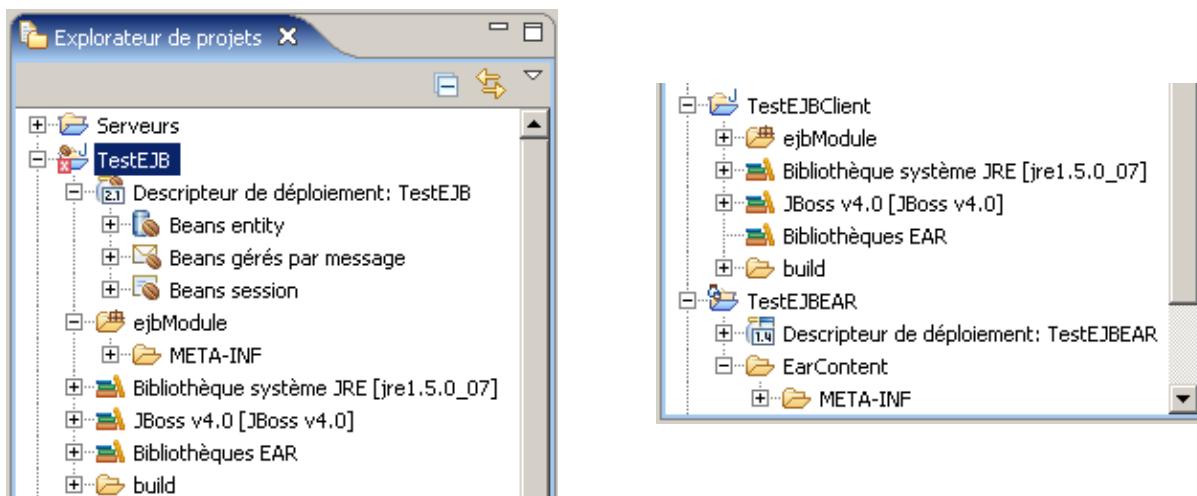
Cliquez sur le bouton « Suivant »



Cliquez sur le bouton « Terminer »

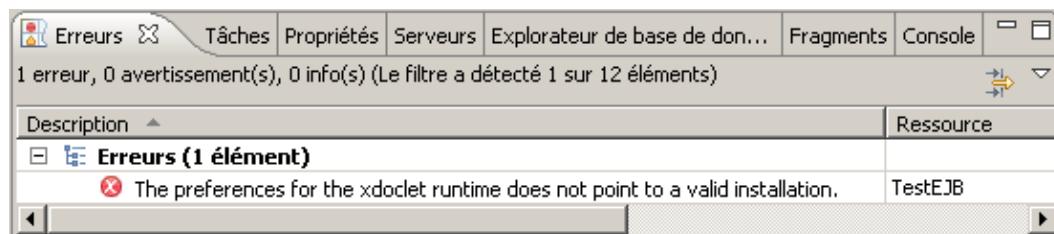


Cliquez sur « Oui » pour créer les projets

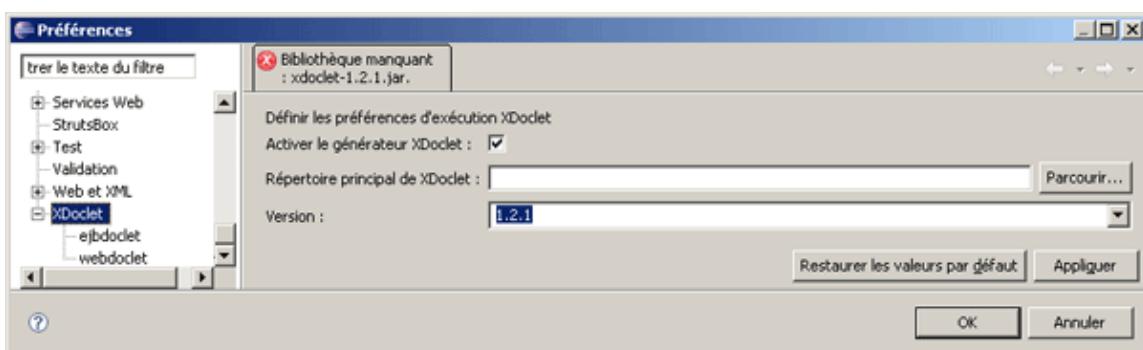


Trois projets sont créés

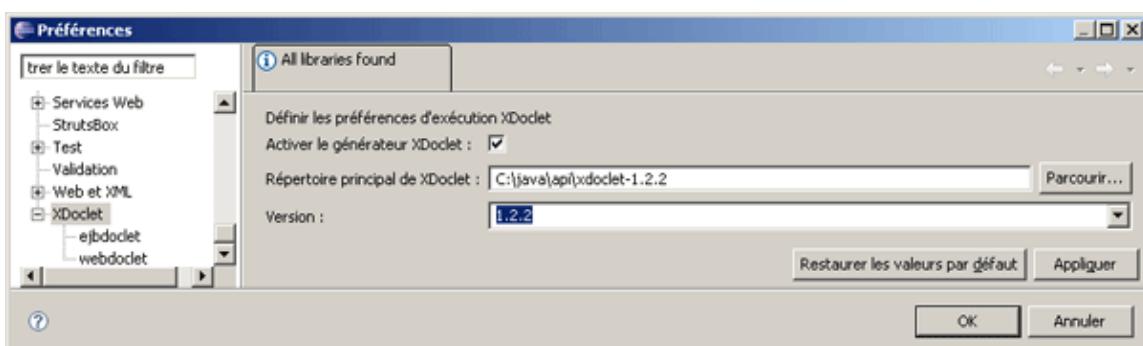
- TestEJB
- TestEJBClient
- TestEJBEAR



Une erreur est signalée sur le projet TestEJB si XDoclet n'est pas correctement configuré. Pour la corriger, il faut ouvrir les Préférences d'Eclipse.



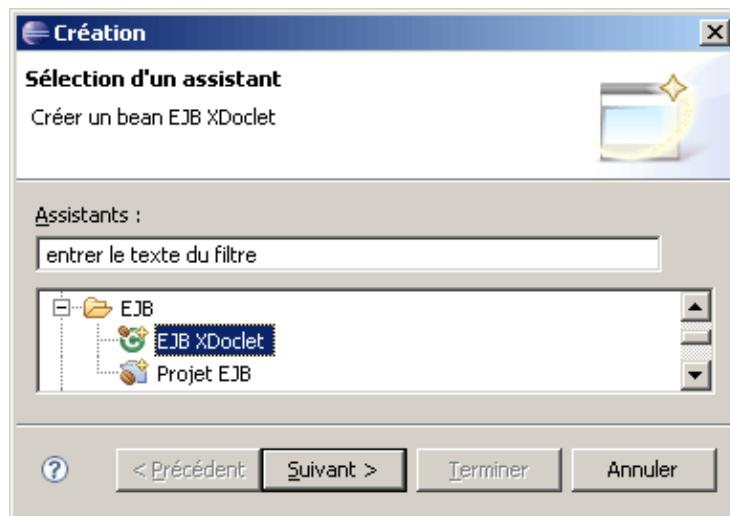
Selectionnez le répertoire d'installation de XDoclet et sa version



Cliquez sur le bouton « OK »

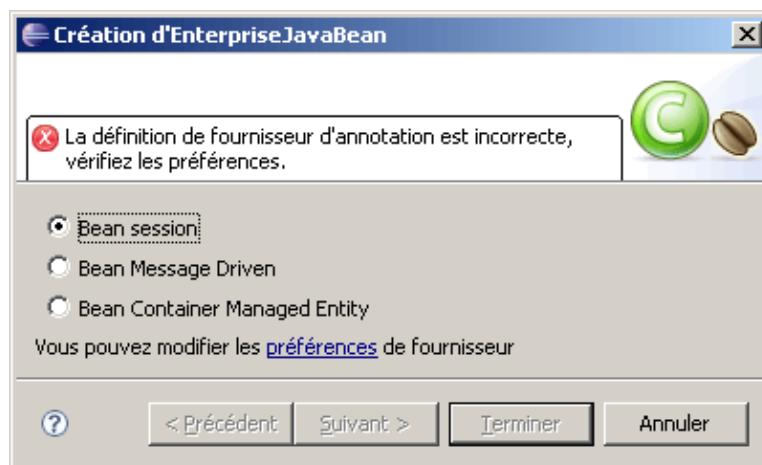
## 25.1.2. La création d'un EJB de type SessionBean stateless

Créez une nouvelle entité de type EJB / EJB XDoclet



Cliquez sur le bouton « Suivant »

Si XDoclet n'est pas correctement configuré, une erreur empêche la poursuite de l'assistant. Pour réaliser cette configuration, il suffit de cliquer sur le lien « Préférences »



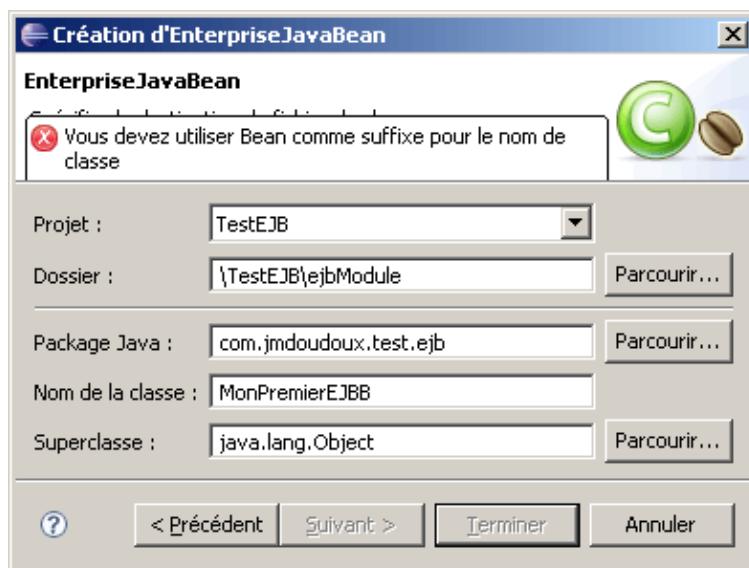
L'assistant permet de sélectionner le type d'EJB à créer.



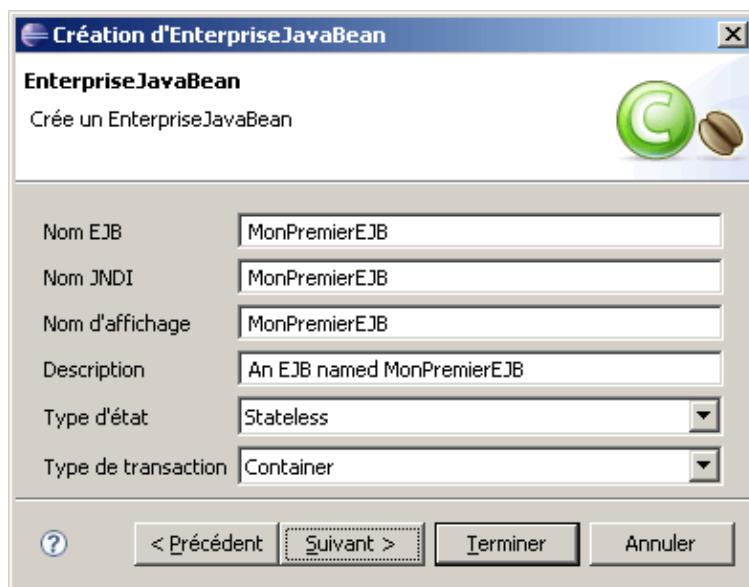
Sélectionnez le type et cliquez sur le bouton « Suivant »



Saisissez le package et le nom de la classe : il est recommandé de suffixer le nom de la classe par Bean.



Cliquez sur le bouton « Suivant »



La page suivante de l'assistant permet de préciser des informations sur l'EJB, cliquez sur le bouton « Suivant ».



Cliquez sur le bouton « Terminer ».

Les fichiers de base sont générés puis les autres fichiers sont générés grâce à un script Ant qui exécute XDoclet

```
Erreurs Tâches Propriétés Serveurs Explorateur de base de données Fragments Console
<arrété> C:\Program Files\Java\jre1.5.0_07\bin\javaw.exe (19 déc. 06 19:46:56)

Buildfile: C:\Documents and Settings\jumbo\workspace\.metadata\.plugins\org.eclipse.jst\xdoclet\init.xml
ejbdoclet:
[XDocletMain.start] (XDocletMain.start) Running <deploymentdescriptor/>
[XDocletMain.start] Generating EJB deployment descriptor (ejb-jar.xml).
[XDocletMain.start] (XDocletMain.start) Running <remoteinterface/>
[XDocletMain.start] Generating Remote interface for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <homeinterface/>
[XDocletMain.start] Generating Home interface for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <localinterface/>
[XDocletMain.start] Generating Local interface for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <localhomeinterface/>
[XDocletMain.start] Generating Local Home interface for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <utilobject/>
[XDocletMain.start] Generating Util class for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <dataobject/>
[XDocletMain.start] (XDocletMain.start) Running <dao/>
[XDocletMain.start] (XDocletMain.start) Running <valueobject/>
[XDocletMain.start] (XDocletMain.start) Running <entitypk/>
[XDocletMain.start] (XDocletMain.start) Running <entitycmp/>
[XDocletMain.start] (XDocletMain.start) Running <entitybmp/>
[XDocletMain.start] (XDocletMain.start) Running <session/>
[XDocletMain.start] Generating Session class for 'com.jmdoudoux.test.ejb.MonPremierEJBBean'.
[XDocletMain.start] (XDocletMain.start) Running <mdb/>
client.jar:
[move] Moving 4 files to C:\Documents and Settings\jumbo\workspace\TestEJBClient\ejb-client.jar
BUILD SUCCESSFUL
Total time: 6 seconds
```

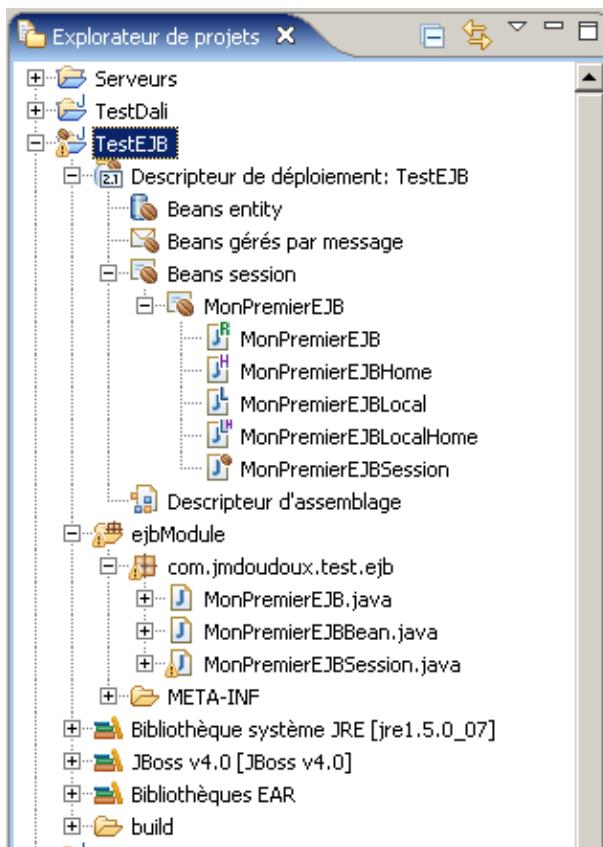
Ouvrez la classe MonPremierEJBBean générée

```
* //TODO: Must provide implementation for bean method stub
*/
public String foo(String param) {
    return null;
}
```

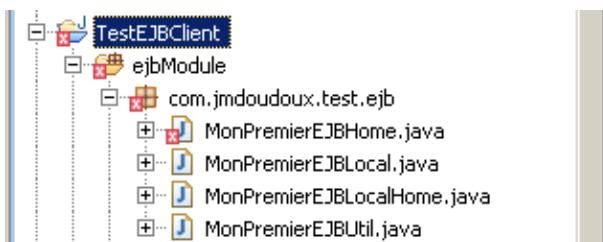
Remplacez la méthode foo par la méthode afficher en saisissant son code :

```
* //TODO: Must provide implementation for bean method stub
*/
public String afficher(String param) {
    return "Bonjour "+param;
}
```

Enregistrez le fichier modifié : le script est exécuté pour permettre à XDoclet de synchroniser les différents fichiers notamment l'interface MonPremierEJB. Le projet est ainsi composé des entités suivantes :

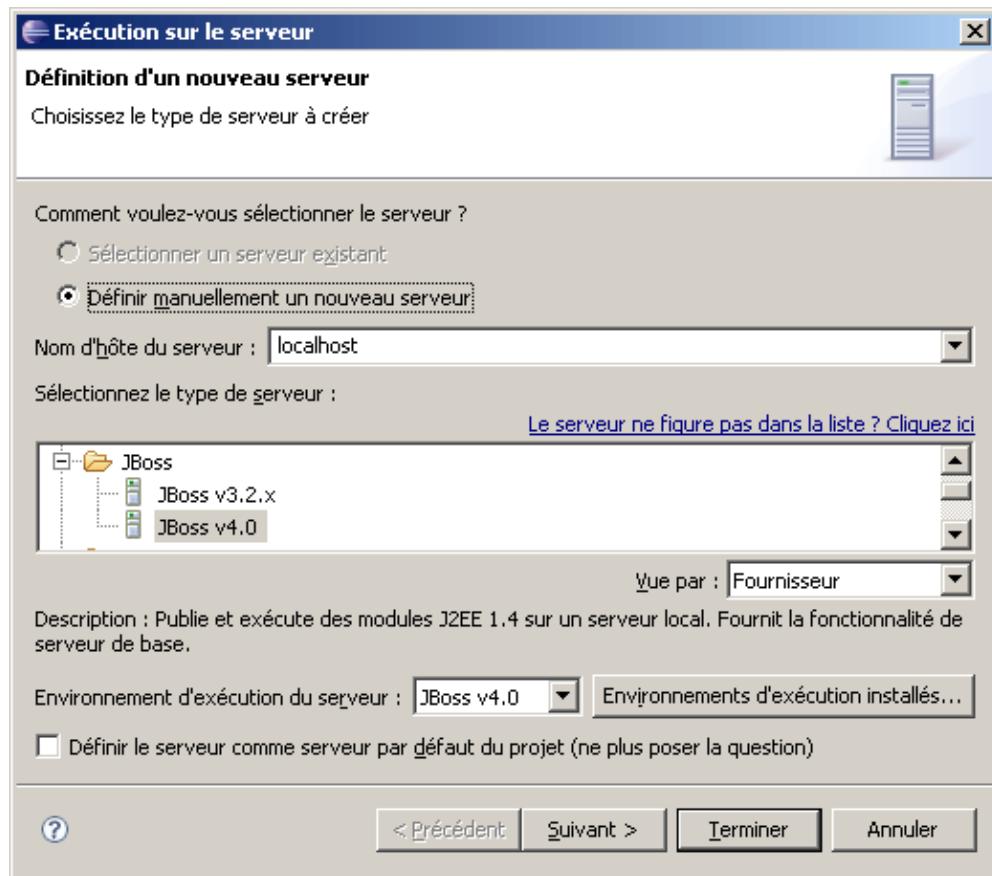


Le projet TestEJBClient comporte une erreur car la classe MonPremierEJB ne peut être résolue.

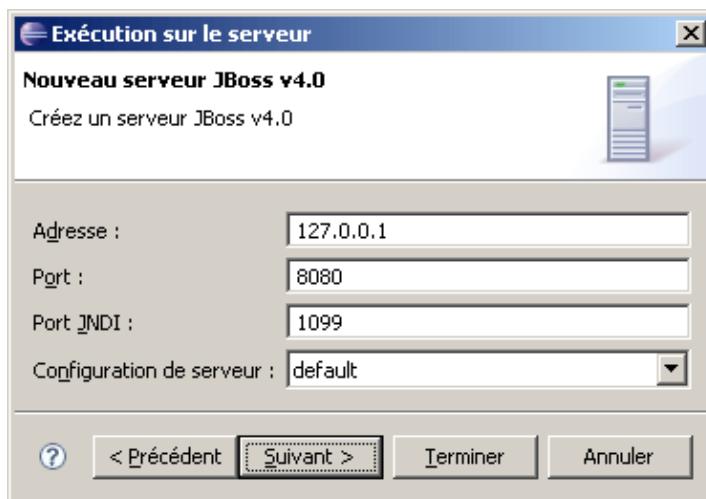


Il suffit dans ce cas de copier l'interface MonPremierEJB du projet TestEJB dans le projet TestClient.

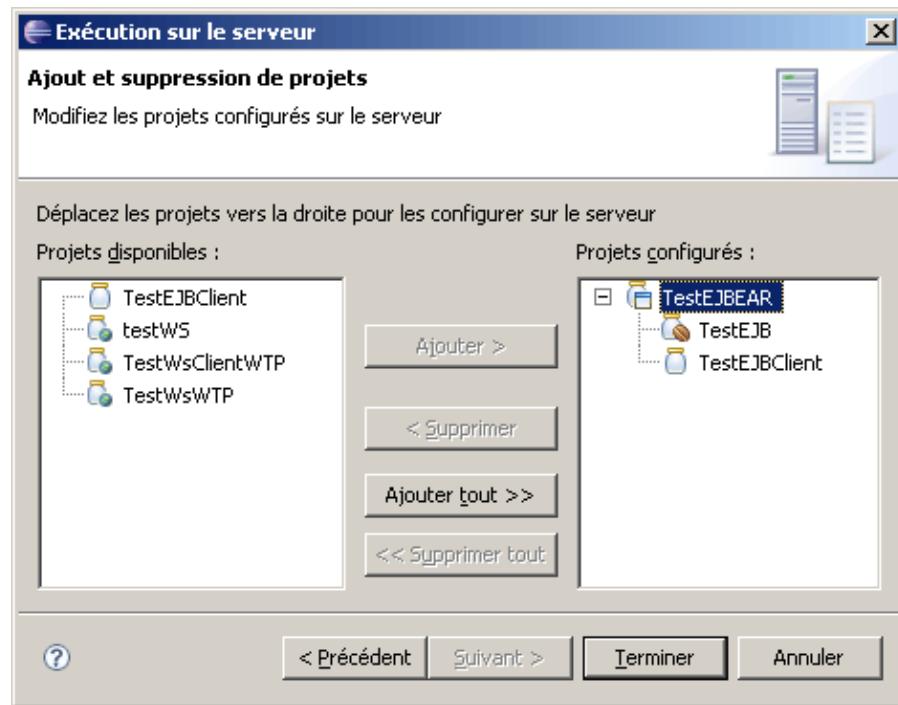
Selectionnez le projet TestEJBEAR et utilisez l'option « Exécuter en tant que / Exécuter sur le serveur »



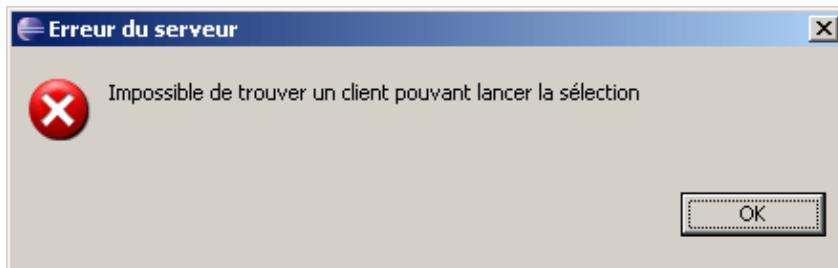
Cochez la case « Définir le serveur comme serveur par défaut du projet » et cliquez sur le bouton « Suivant »



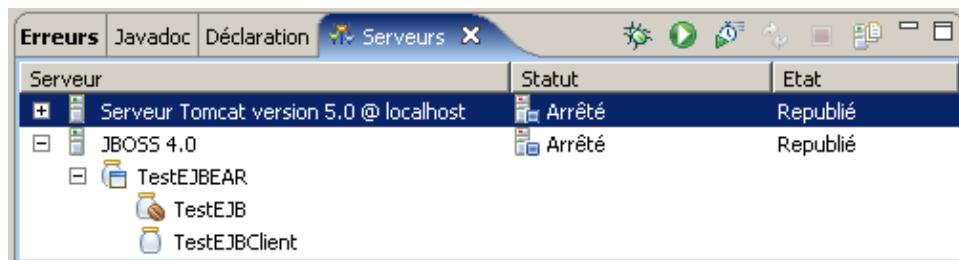
Cliquez sur le bouton « Suivant ».



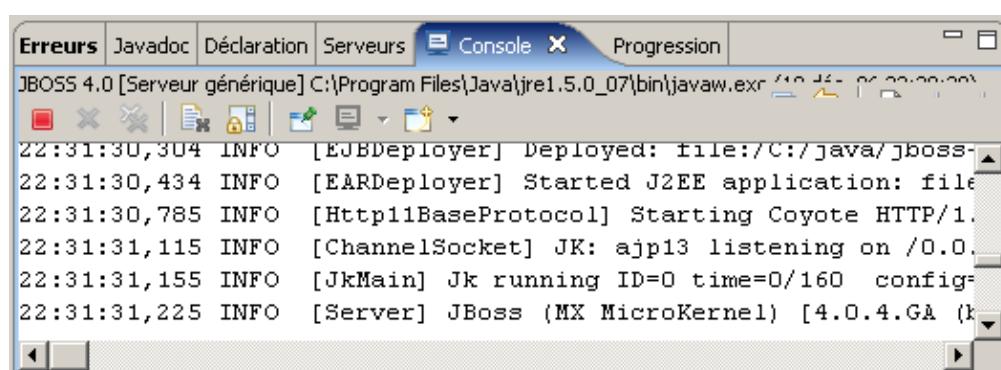
Cliquez sur le bouton « Terminer ».



Ouvrez la vue « Serveur »



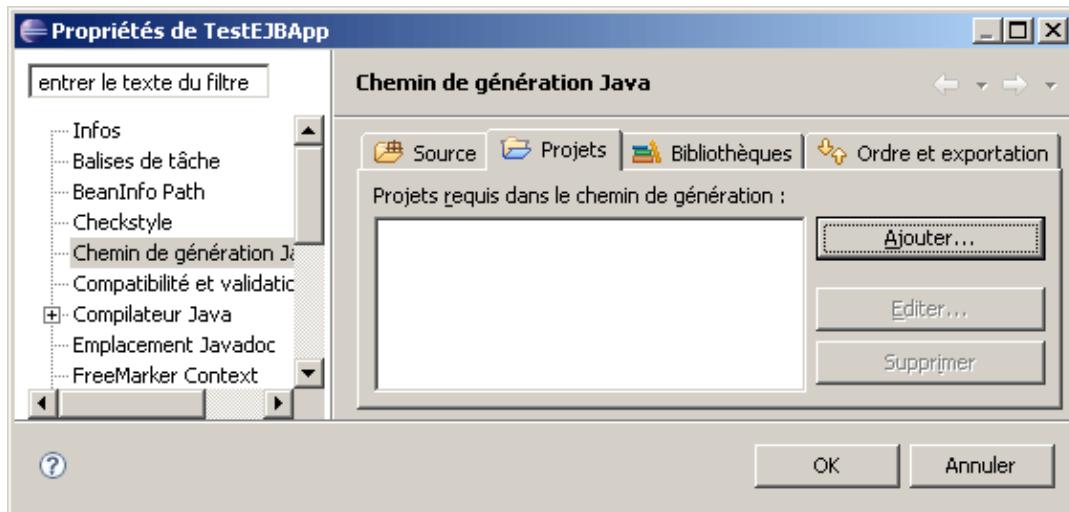
Sélectionnez le serveur JBoss et cliquez sur le bouton ou sélectionnez l'option « Démarrer » du menu contextuel



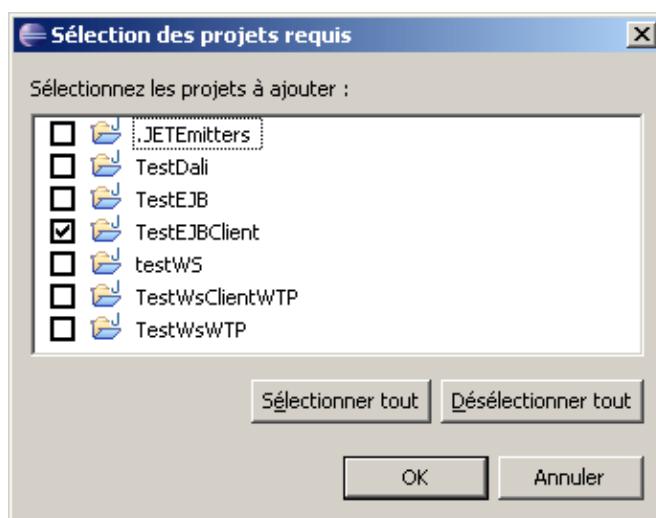
La vue console affiche les informations lors du démarrage du serveur.

### 25.1.3. La création et l'exécution d'un client de test

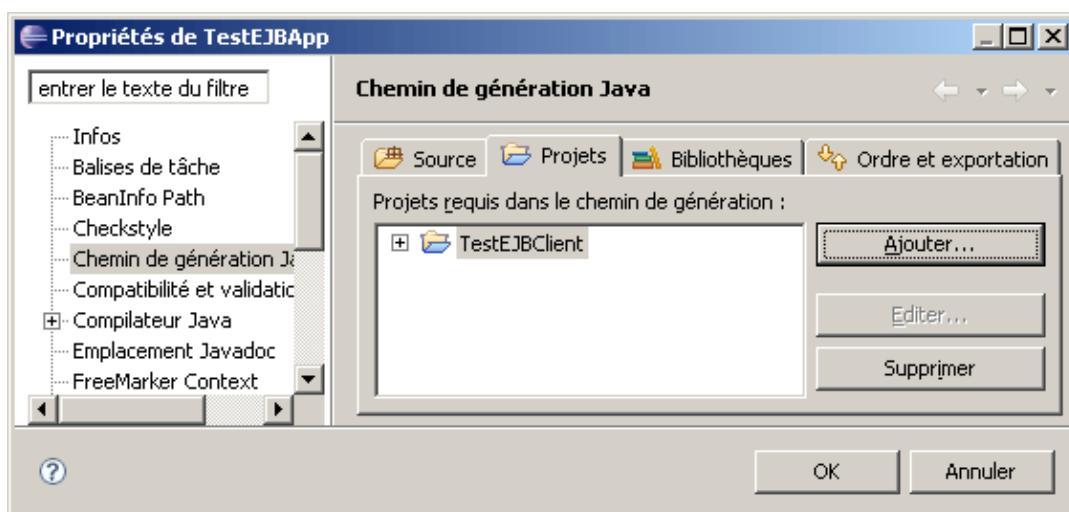
Créez un nouveau projet de type Java nommé TestEJBApp.



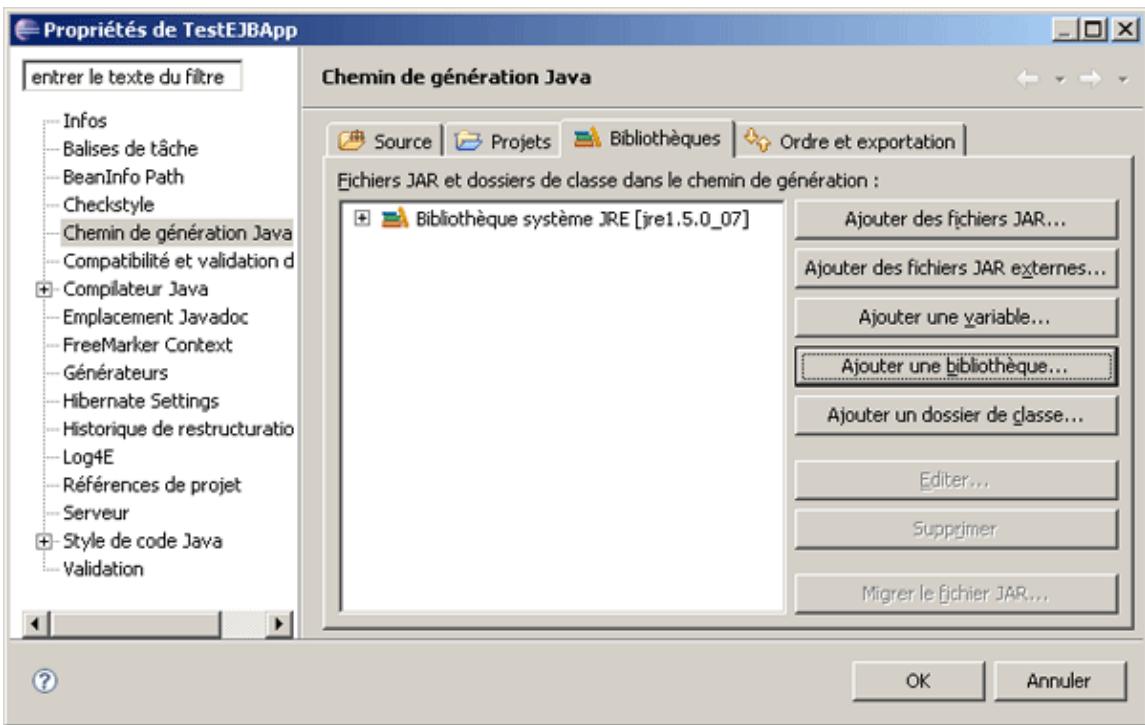
Dans les propriétés du projet, sur l'onglet « Projets », cliquez sur le bouton « Ajouter »



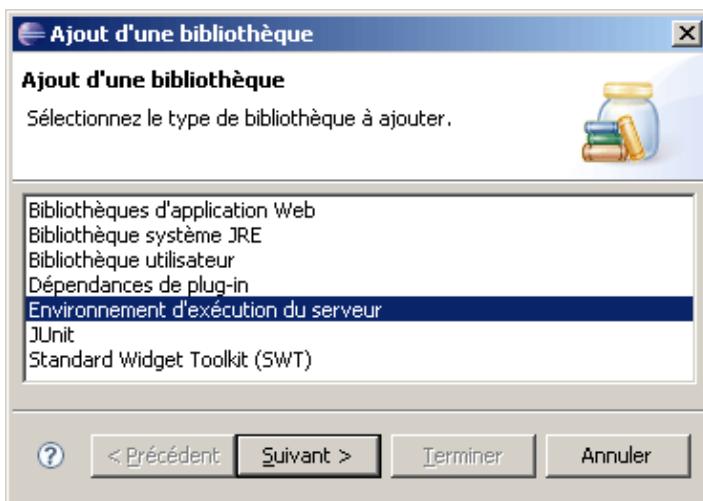
Sélectionnez le projet TestEJBClient et cliquez sur le bouton « OK ».



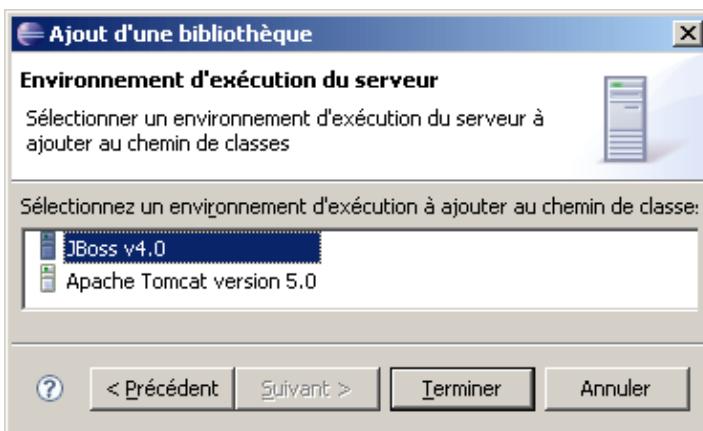
Cliquez sur l'onglet « Bibliothèques »



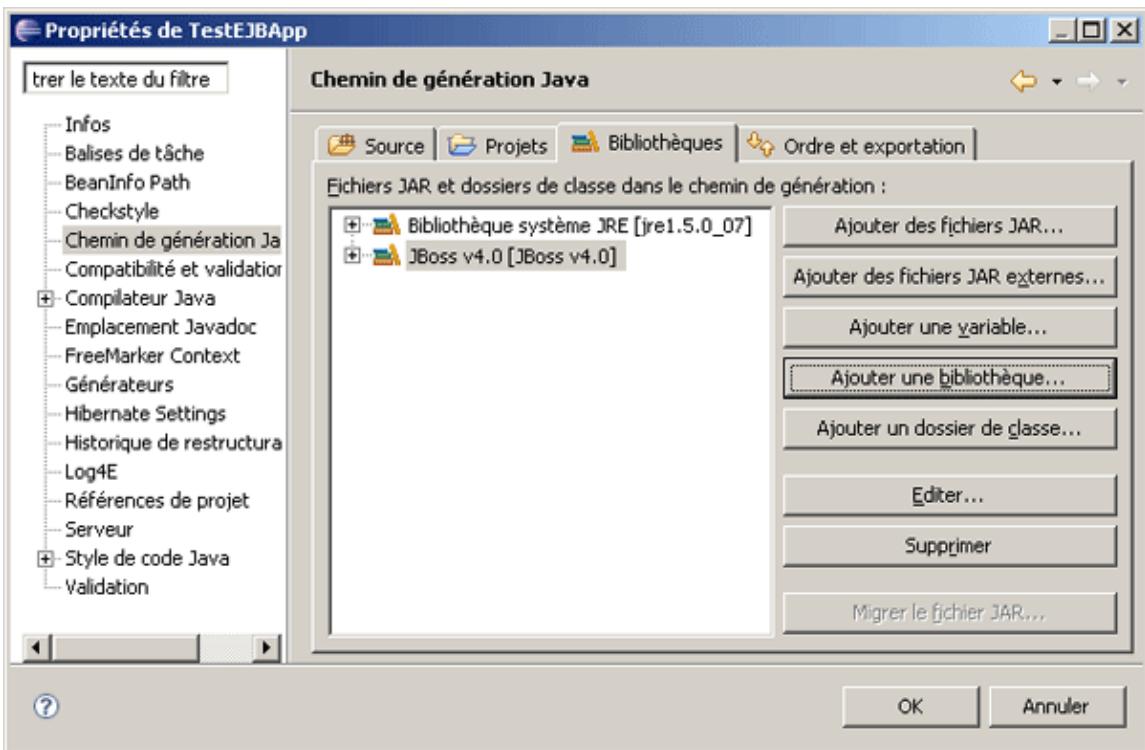
Cliquez sur le bouton « Ajouter une bibliothèque ... ».



Sélectionnez « Environnement d'exécution du serveur » et cliquez sur le bouton « Suivant ».



Sélectionnez JBoss v4.0 et cliquez sur le bouton « Terminer ».



Cliquez sur le bouton « OK ».

Créez une nouvelle classe Java nommée TestEJBApp et saisissez son code :

Exemple :

```
package com.jmdoudoux.test.ejb.app;

import java.util.Hashtable;
import javax.naming.Context;
import com.jmdoudoux.test.ejb.MonPremierEJB;
import com.jmdoudoux.test.ejb.MonPremierEJBHome;
import com.jmdoudoux.test.ejb.MonPremierEJBUtil;

public class TestEJBApp {

    public static void main(String[] args) {
        Hashtable environment = new Hashtable();
        environment.put(Context.INITIAL_CONTEXT_FACTORY,
                      "org.jnp.interfaces.NamingContextFactory");
        environment.put(Context.URL_PKG_PREFIXES,
                      "org.jboss.naming:org.jnp.interfaces");
        environment.put(Context.PROVIDER_URL, "jnp://127.0.0.1:1099");
        MonPremierEJB monEJB = null;

        try {
            MonPremierEJBHome home = MonPremierEJBUtil.getHome(environment);
            monEJB = home.create();
            System.out.println(monEJB.afficher("test"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Exécutez la classe en tant qu'application Java

Résultat :

## 25.2. Le développement d'EJB avec le plug-in Lomboz 2.1

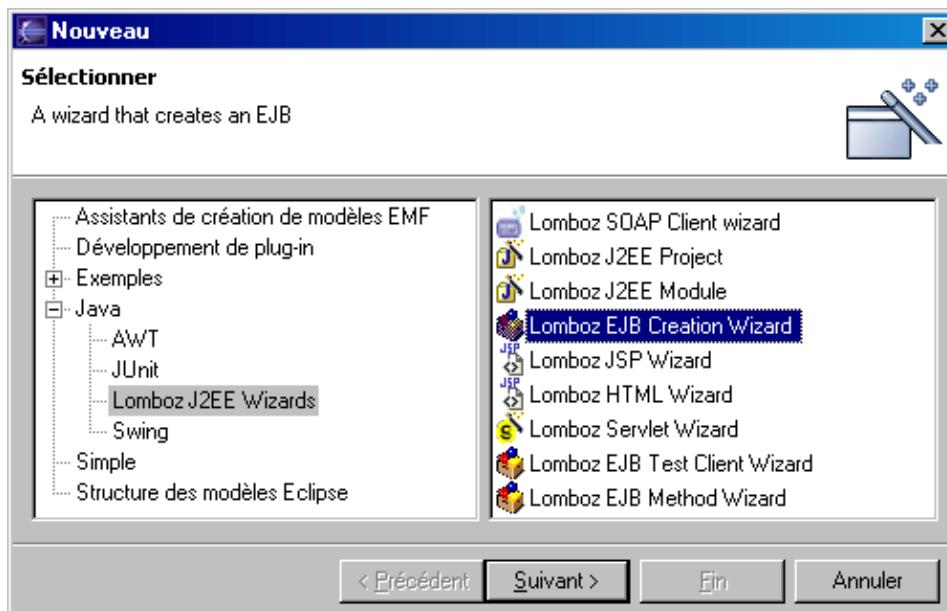
Le plug-in Lomboz propose des fonctionnalités pour faciliter le développement d'EJB dans un projet de type « Java / Lomboz J2EE Wizard / Lomboz J2EE Project ». La configuration de Lomboz et la création d'un tel projet est détaillé dans le chapitre "".

Cette section va utiliser Lomboz avec JBoss.

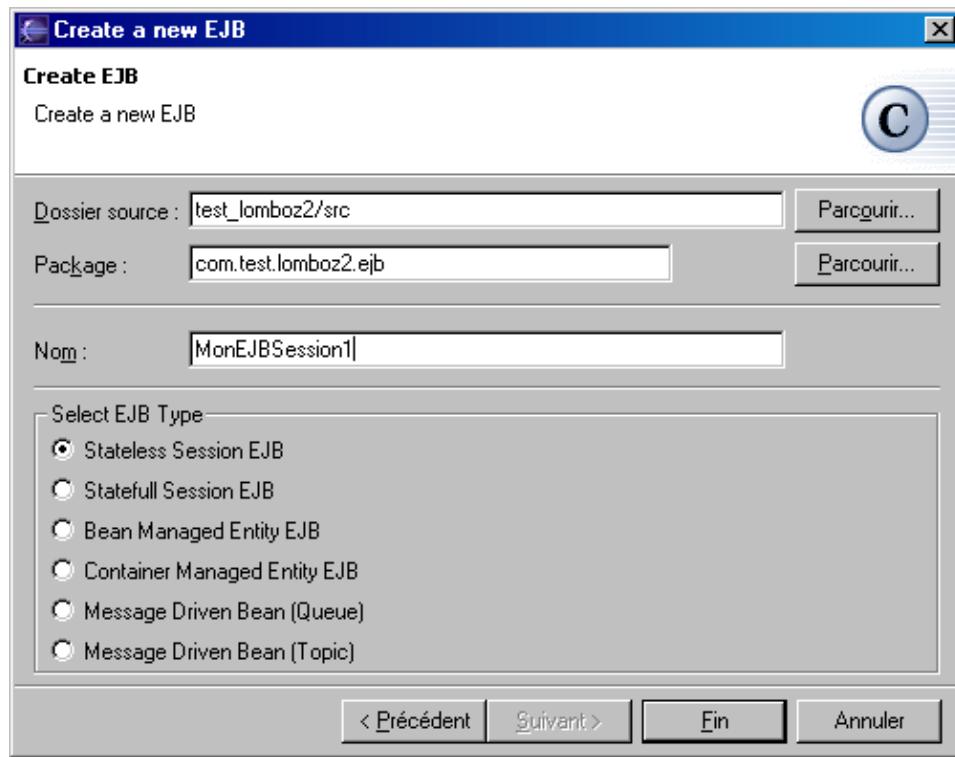
	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Lomboz	2.1.2
JBoss	3.0.6

### 25.2.1. La création un EJB de type Session

Il faut créer une nouvelle entité de type « Java / Lomboz J2EE Wizards / Lomboz EJB Creation Wizard ».



Cliquez sur le bouton « Suivant »



Sur la page suivante, il faut saisir le nom du package, le nom de l'EJB, sélectionner le type « Stateless Session EJB » ou « Statefull Session EJB » selon le type d'EJB session à créer et enfin cliquer sur le bouton « Fin ».

L'assistant génère un fichier source java contenant le squelette du code de l'implémentation de l'EJB. Le nom du fichier généré est constitué du nom de l'EJB et du suffixe Bean qui est automatiquement ajouté.

#### Exemple :

```
package com.test.lomboz2.ejb;

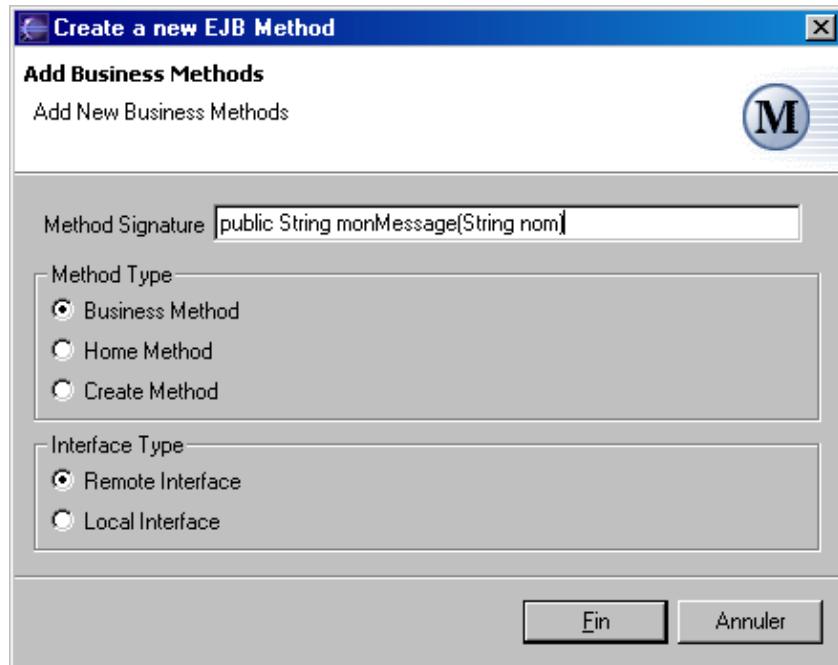
import javax.ejb.SessionBean;

/**
 * @ejb.bean name="MonEJBSession1"
 *           jndi-name="MonEJBSession1Bean"
 *           type="Stateless"
 */
public abstract class MonEJBSession1Bean implements SessionBean { }
```

Ce code contient un tag qui sera traiter par Xdoclet pour générer ou enrichir certains fichiers (l'interface Home du Bean, le fichier descripteur de déploiement et le fichier jboss.xml).

### 25.2.2. Ajouter une méthode à un EJB

Dans la vue « Packages », il faut sélectionner la classe d'implementation du Bean précédemment générée et utiliser l'option « Add EJB Method » du menu contextuel « Lomboz J2EE » ou utiliser l'option « Autre... » du menu contextuel « Nouveau ». Avec cette dernière possibilité, il faut sélectionner la création d'une entité de type « Lomboz EJB Method Wizard »



L'assistant permet de saisir la signature complète de la méthode, le type de méthode et dans quelle interface sera générée la méthode.

L'assistant génère le code suivant :

Exemple :

```
/***
 * @ejb.interface-method
 *   tview-type="remote"
***/

public String monMessage(String nom){
    return null;
}
```

Le tag dans les commentaires sera utilisé par Xdoclet pour la génération des fichiers nécessaire à l'EJB.

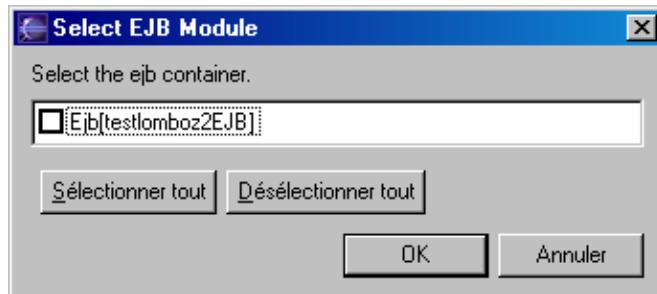
Il faut saisir le code des traitements de la nouvelle méthode.

Exemple :

```
/***
 * @ejb.interface-method
 *   tview-type="remote"
***/
public String monMessage(String nom) {
    return "Bonjour " + nom;
}
```

ajout de l'EJB à un module

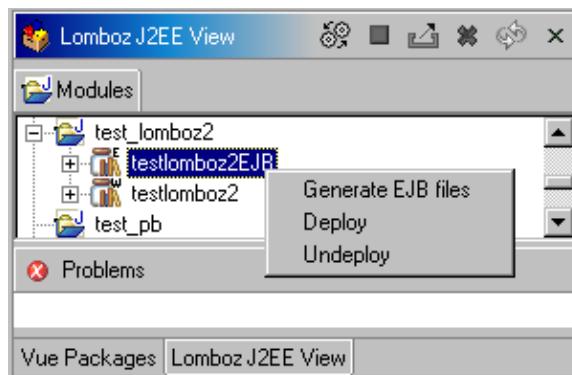
Dans la vue « Package », sélectionner la classe du bean et utiliser l'option « Lomboz J2EE / Add EJB to Module »



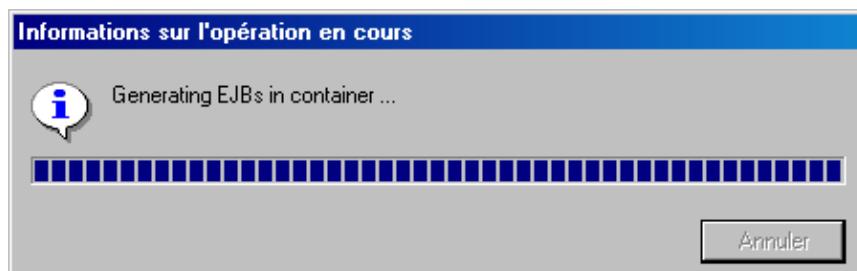
Sélectionner le module et cliquer sur « OK ».

### 25.2.3. La génération des fichiers des EJB

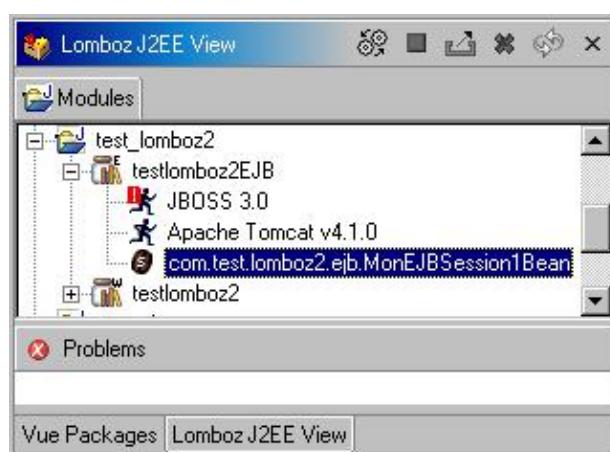
Avant de pouvoir déployer un ou plusieurs EJB, il faut demander à Lomboz de générer les fichiers nécessaires aux EJB grâce à Xdoclet et aux tags utiles insérer dans le code



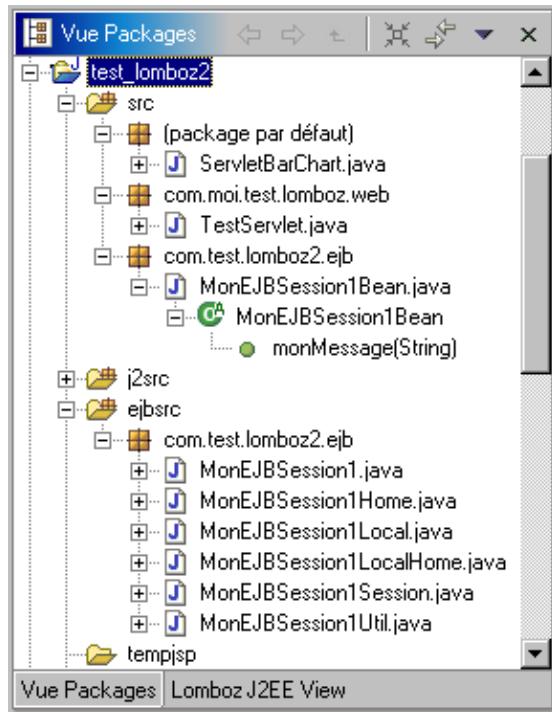
Dans la vue « Lomboz », il faut utiliser sur le module EJB l'option « Generate EJB files ».



L'EJB est affichée dans la vue Lomboz.

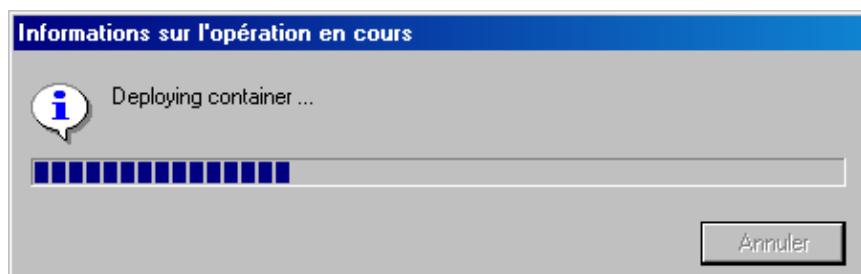


Lomboz a généré tous les fichiers nécessaires au bean



#### 25.2.4. Déploiement du module EJB

Dans la vue « Lomboz », sélectionner le module EJB concerné et utiliser l'option « Deploy ».



Le fichier testlomboz2EJB.jar est créé et enregistrer dans le répertoire server/default/deploy de JBOSS.

## 26. Les services web et Eclipse

# Chapitre 26

Ce chapitre propose le développement de services web avec Eclipse essentiellement avec Axis avec ou sans utilisation d'un plug-in dédié.

Ce chapitre contient plusieurs sections :

- ◆ [La mise en oeuvre manuelle d'Axis](#)
- ◆ [La consommation du service Web en .Net](#)
- ◆ [Le développement de services web avec WTP 1.0](#)
- ◆ [La mise en oeuvre d'Axis avec le WTP 1.5](#)

### 26.1. La mise en oeuvre manuelle d'Axis

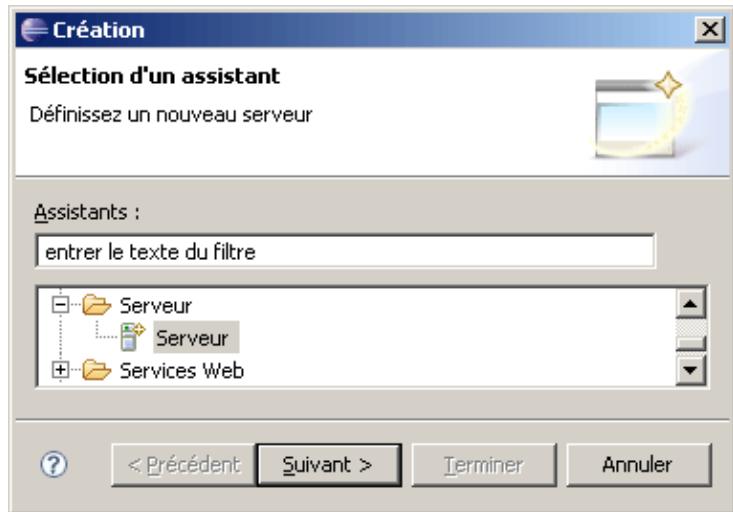
Cette section va mettre en oeuvre les outils suivants sous Windows :

	Version utilisée dans cette section
Eclipse	3.2.1
WTP	1.5.1
JDK	1.5_07
Axis	1.4
Tomcat	5.0

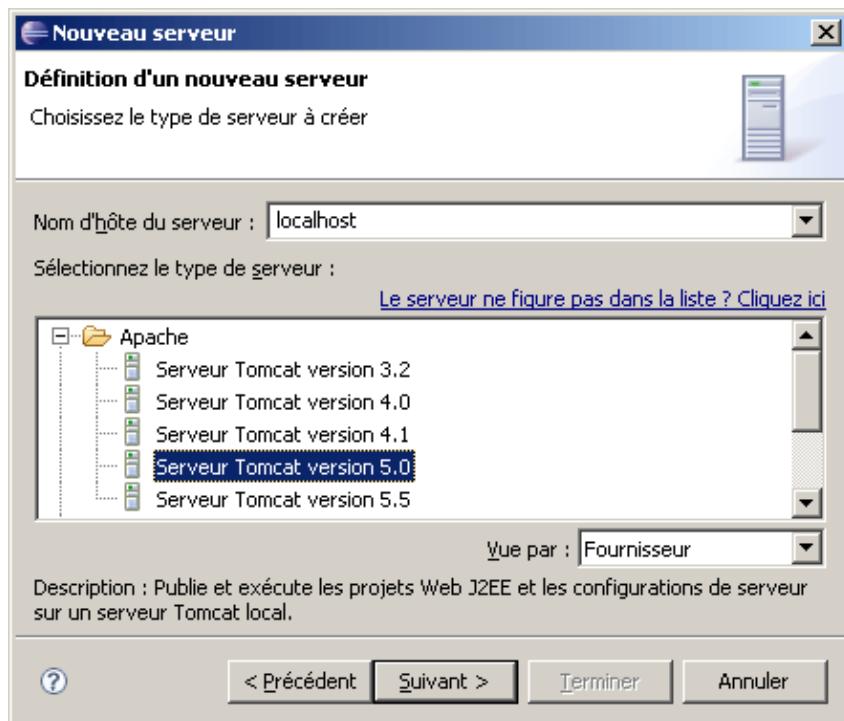
#### 26.1.1. La configuration de l'environnement

Téléchargez le fichier axis-bin-1\_4.zip et le décompresser dans un répertoire du système, par exemple c:\java.

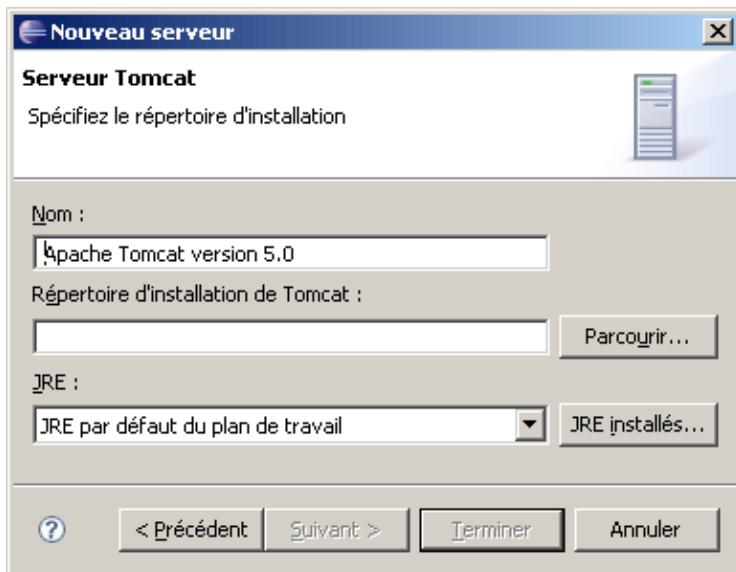
Créer une nouvelle entité de serveur si aucun serveur n'est déjà défini dans Eclipse. L'exemple de cette section va mettre en oeuvre Tomcat 5.0.



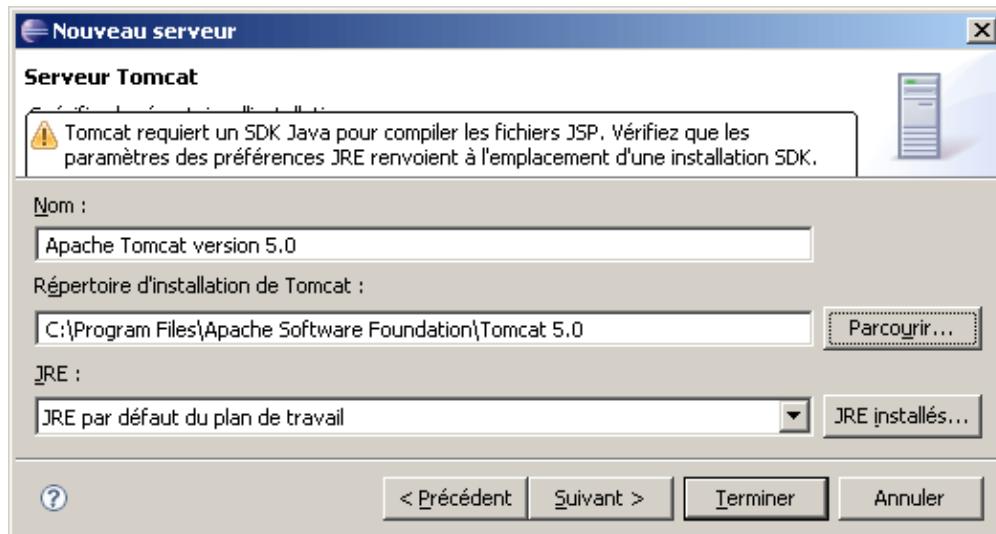
Créer une nouvelle entité de type « Serveur/Serveur » puis cliquez sur le bouton « Suivant ».



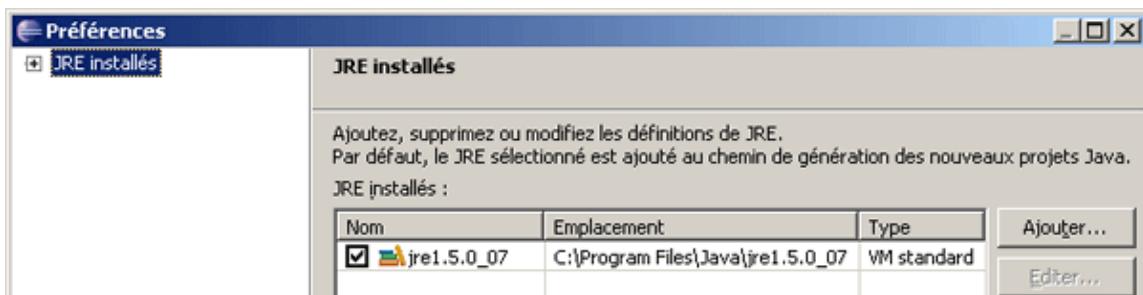
Sélectionnez le type de serveur parmi ceux proposés puis cliquez sur le bouton « Suivant ».



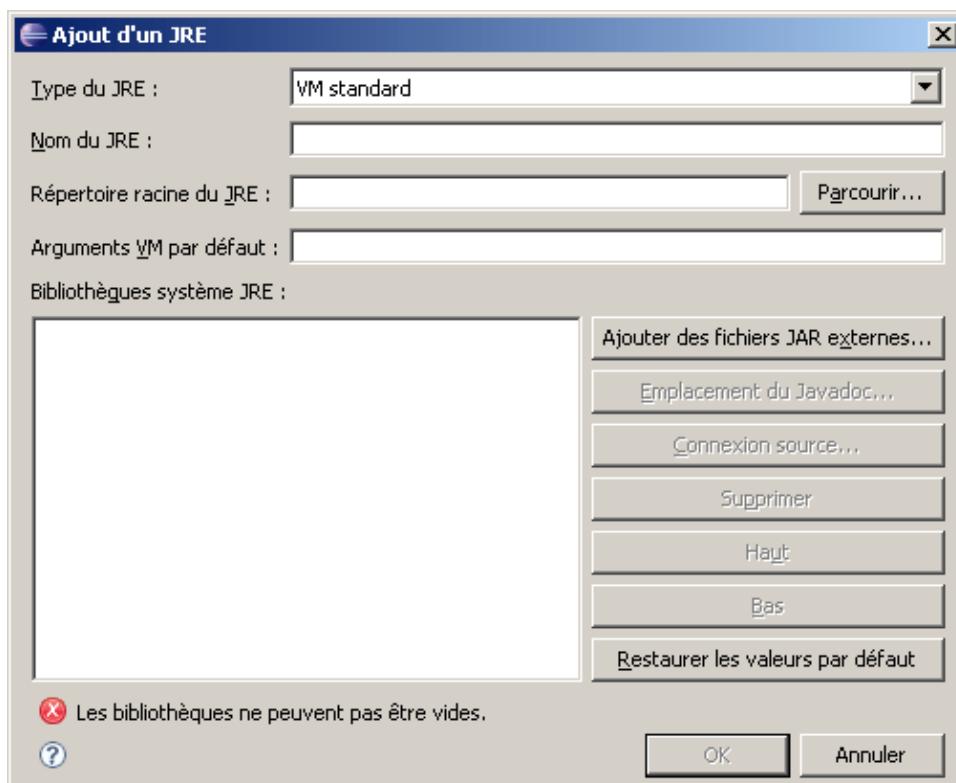
Cliquez sur « Parcourir » pour sélectionner le répertoire d'installation de Tomcat.



Pour utiliser Tomcat, il est nécessaire de sélectionner un JDK pour permettre à Tomcat d'utiliser certains outils. Si aucun JDK n'apparaît dans la liste déroulante, cliquez sur le bouton « JRE installés ».



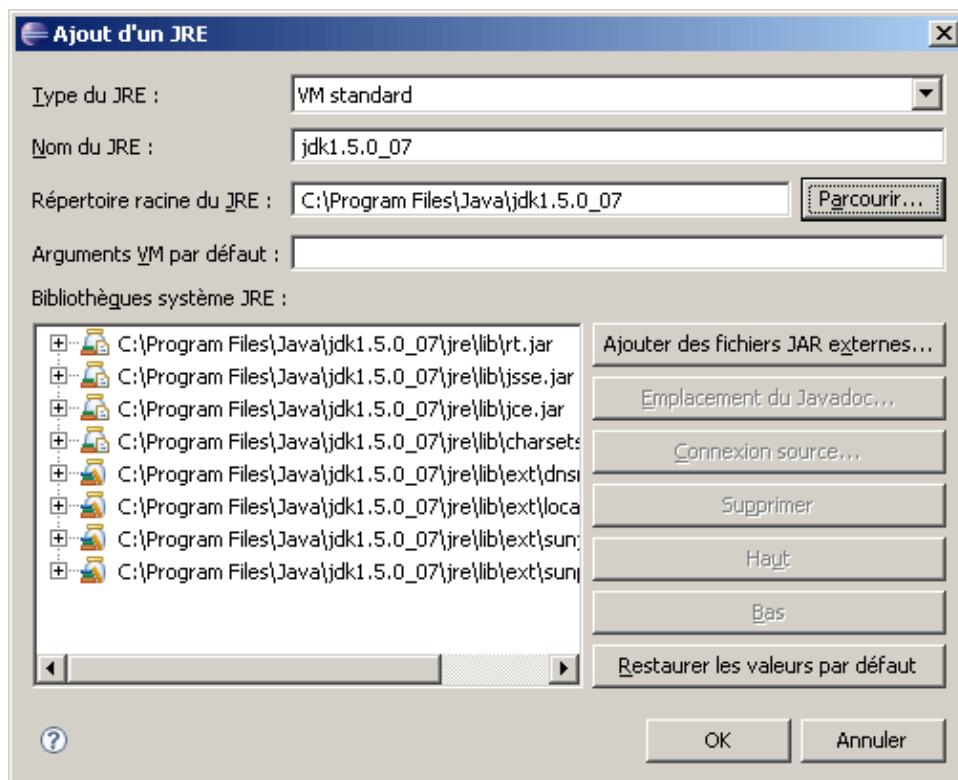
Cliquez sur le bouton « Ajouter »



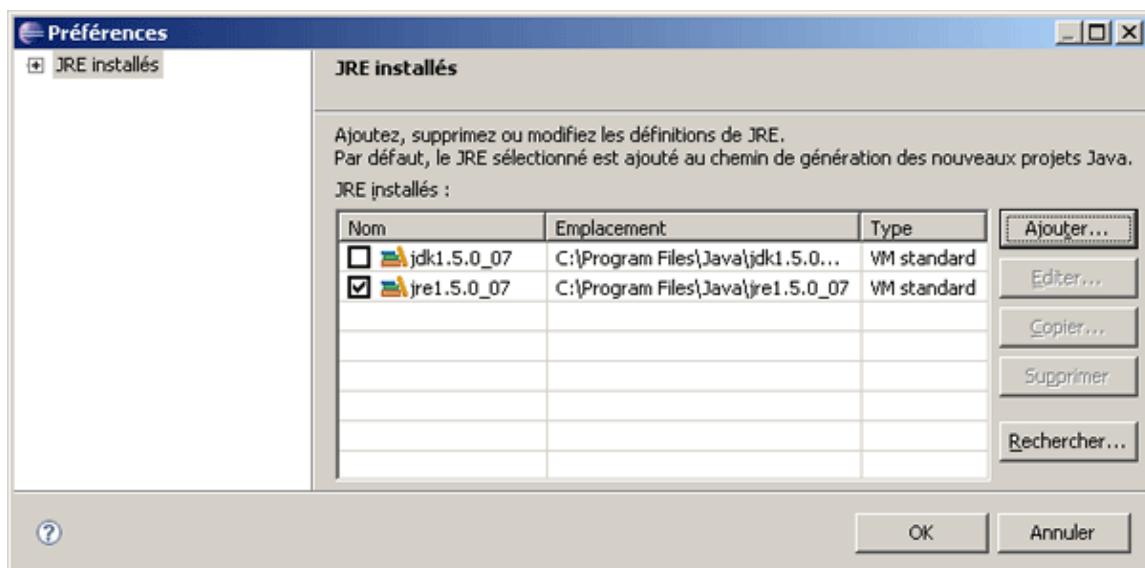
Cliquez sur le bouton « Parcourir »



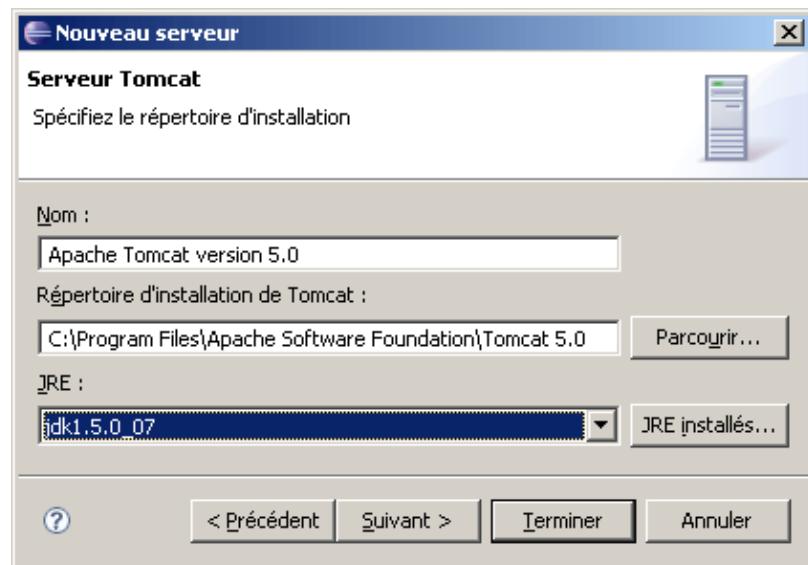
Sélectionnez le répertoire d'installation du JDK et cliquez sur le bouton « OK ».



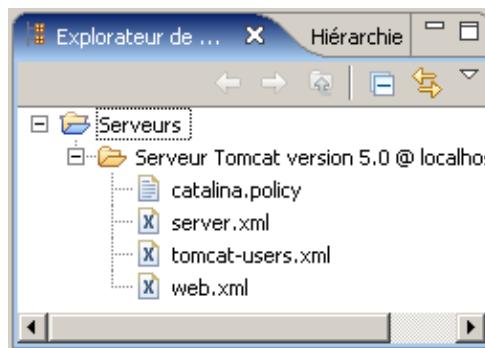
Eclipse renseigne automatiquement les informations extraites à partir du JDK. Cliquez sur le bouton « OK ».



Cliquez sur le bouton « OK » puis sélectionnez le JDK créé dans la liste déroulante



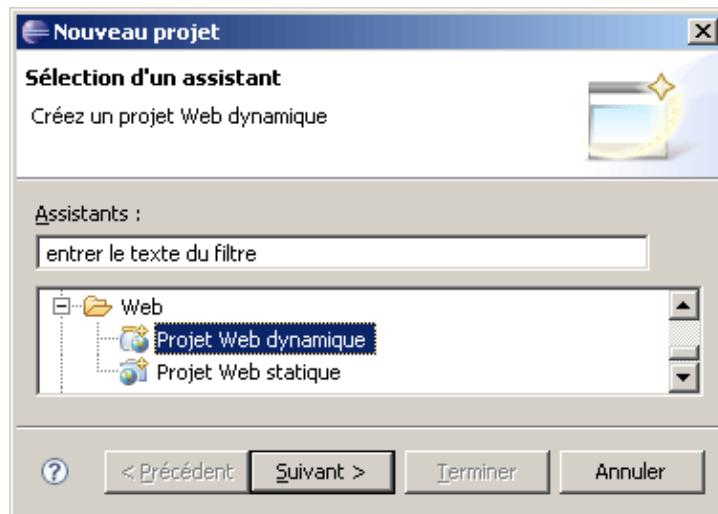
Cliquez sur le bouton « Terminer »



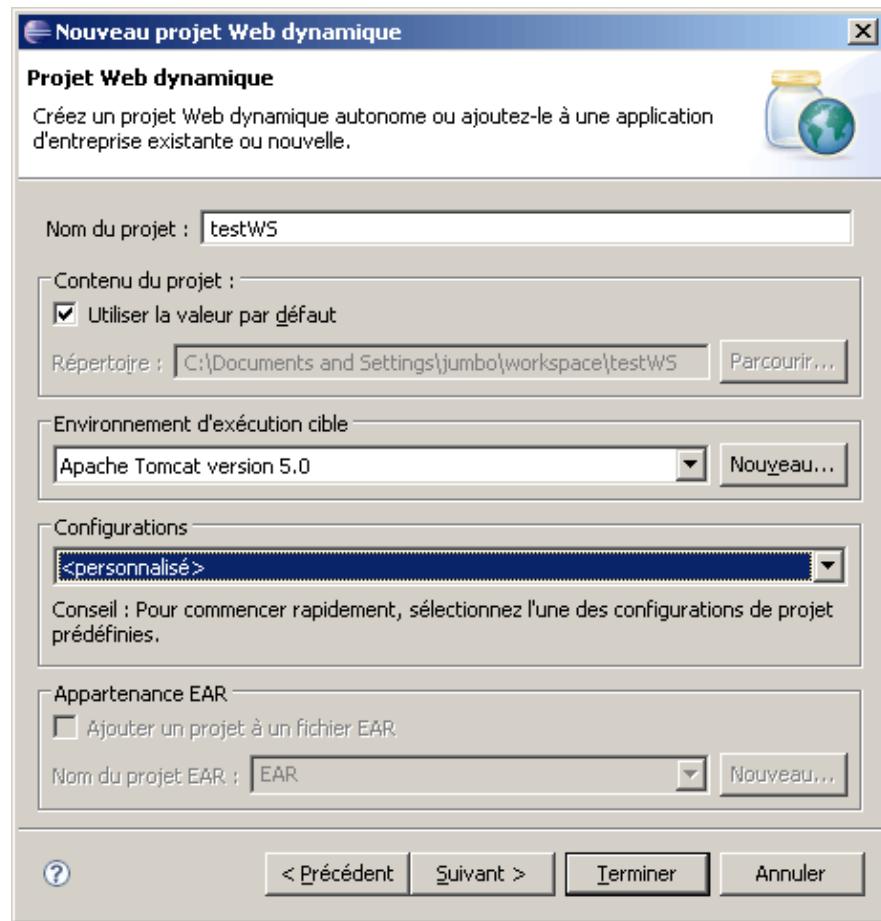
Le nouveau serveur apparaît dans la branche serveurs de la vue « Explorateur de package ».

### 26.1.2. La création d'un projet de type web

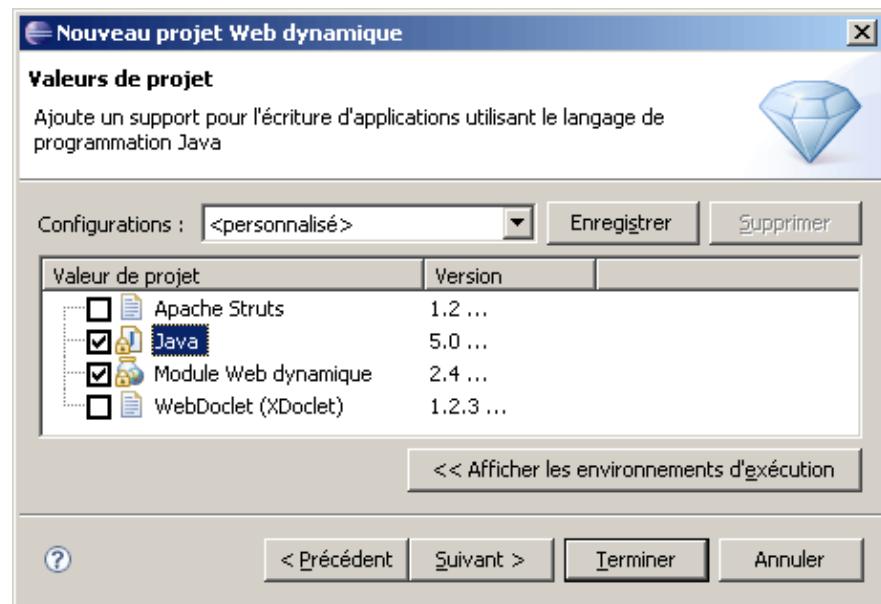
Créez une nouvelle entité de type « Web/Projet Web dynamique »



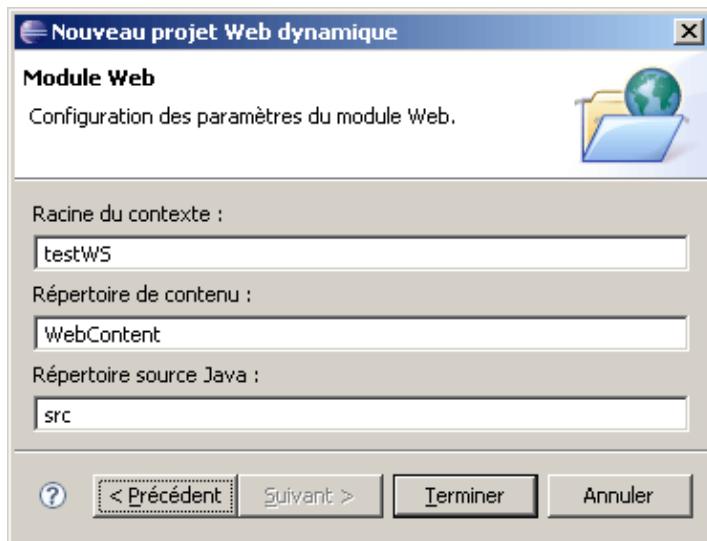
Cliquez sur le bouton « Suivant »



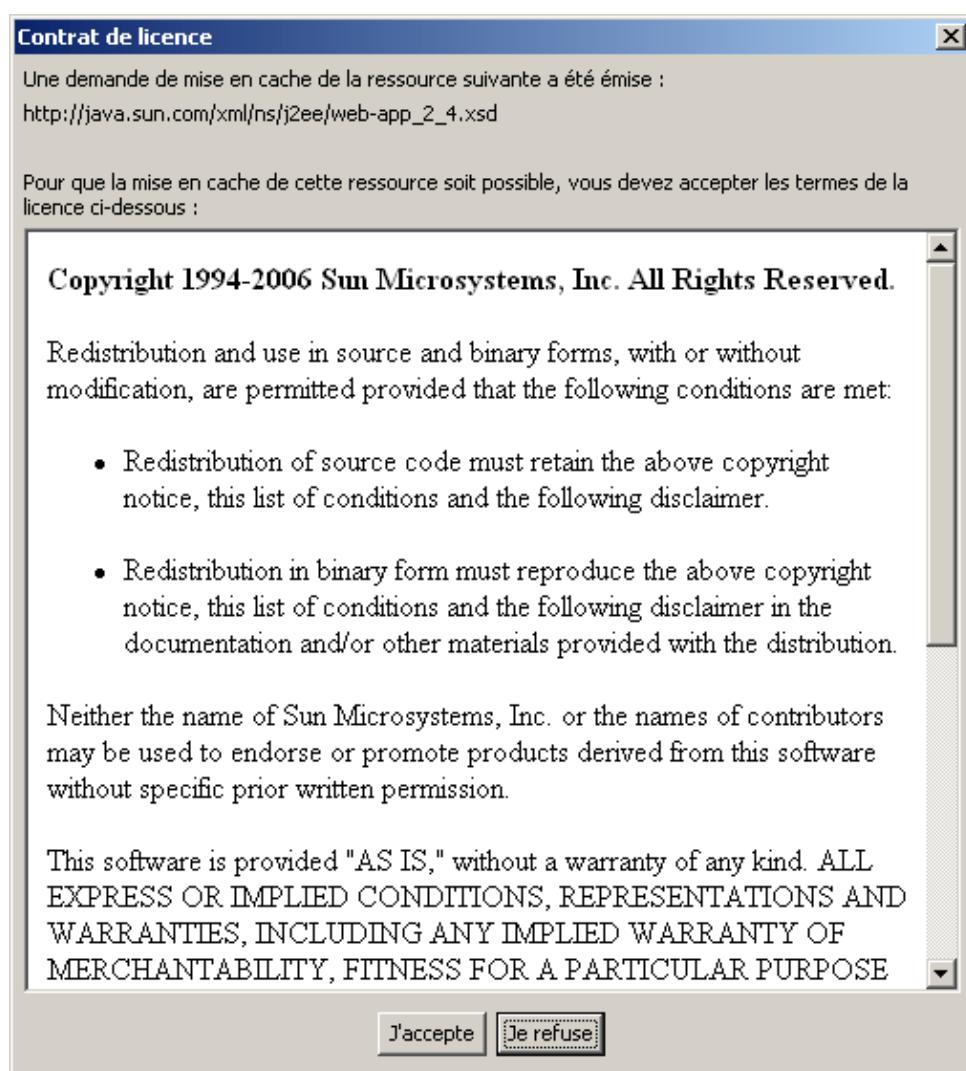
Saisissez le nom du projet puis cliquez sur le bouton « Suivant »



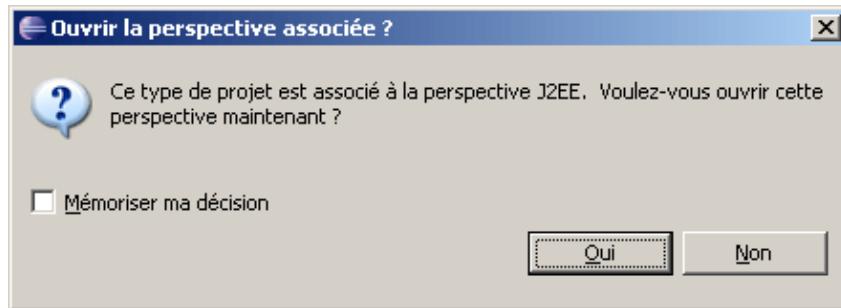
Cliquez sur le bouton « Suivant »



Cliquez sur le bouton « Terminer »

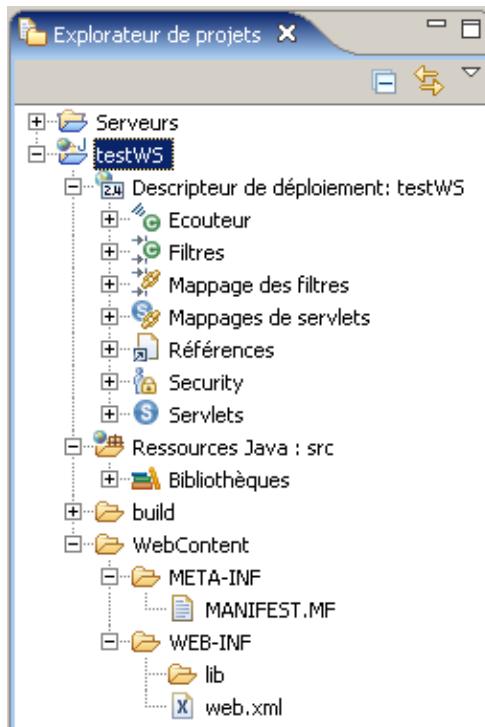


Lisez la licence et si vous l'acceptez, cliquez sur le bouton « J'accepte ».



Cliquez sur la case à cocher « Mémoriser ma décision » puis sur le bouton « Oui »

La perspective « J2EE » s'ouvre et le projet est affiché dans la vue « Explorateur de projets »



### 26.1.3. La configuration du projet

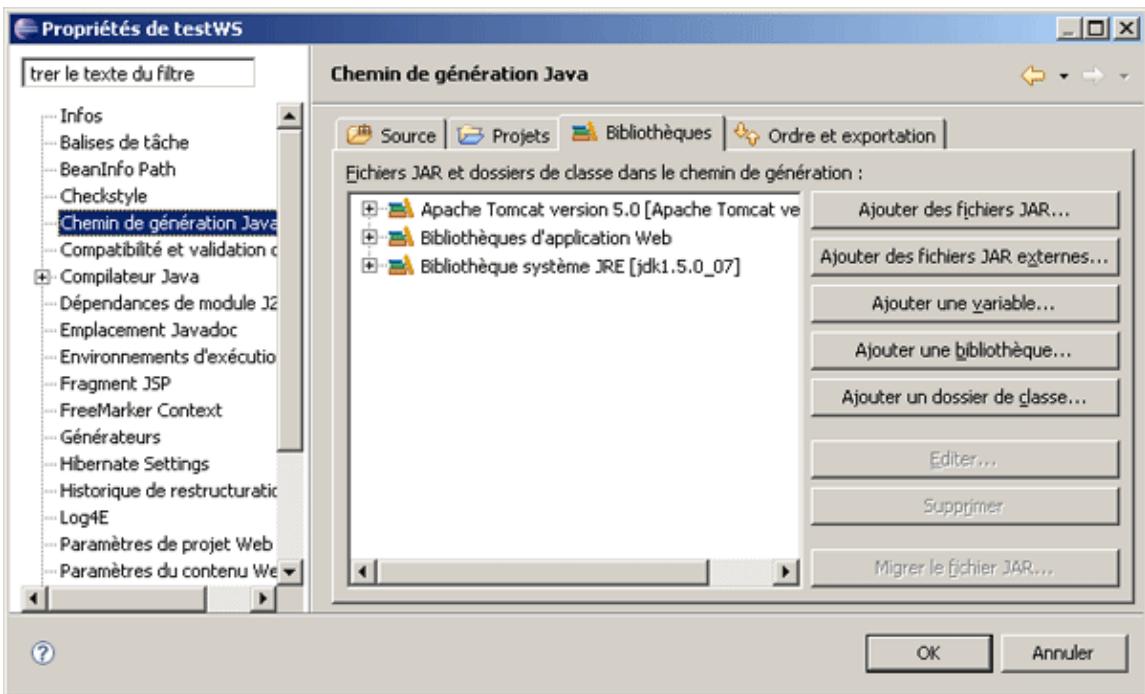
Créez un répertoire nommé « classes » dans le répertoire WEB-INF du projet.

Dans les propriétés du projet, sélectionnez « chemin de génération Java », cliquez sur l'onglet « Source », sélectionnez dans la zone de texte « Dossier de sortie par défaut » le chemin du répertoire « classes » puis cliquez sur le bouton « OK ».

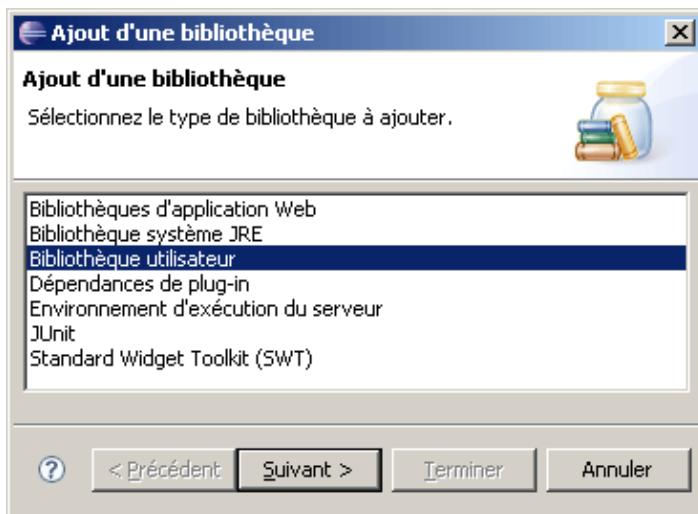
Remplacez le fichier WEBContent\WEB\_INF\web.xml par celui présent dans le sous répertoire webapps\axis\WEB-INF du répertoire d'installation d'axis.

Copiez les fichiers du sous répertoire webapps\axis\WEB-INF\lib du répertoire d'installation d'axis dans le répertoire WEBContent\WEB-INF\lib.

Dans les propriétés du projet, sélectionnez « Chemin de génération Java » puis cliquez sur l'onglet « Bibliothèques »



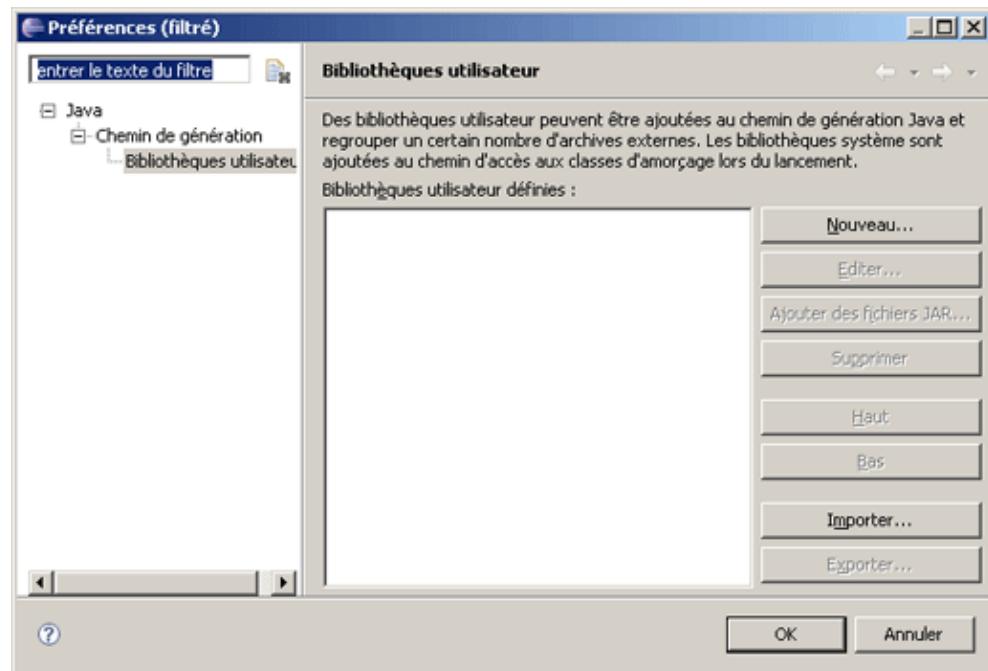
Cliquez sur le bouton « Ajouter une bibliothèque »



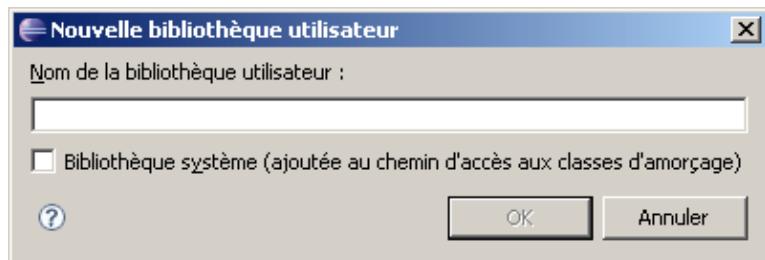
Sélectionnez « Bibliothèque utilisateur » puis cliquez sur le bouton « Suivant ».



Cliquez sur le bouton « Bibliothèque utilisateur ... »

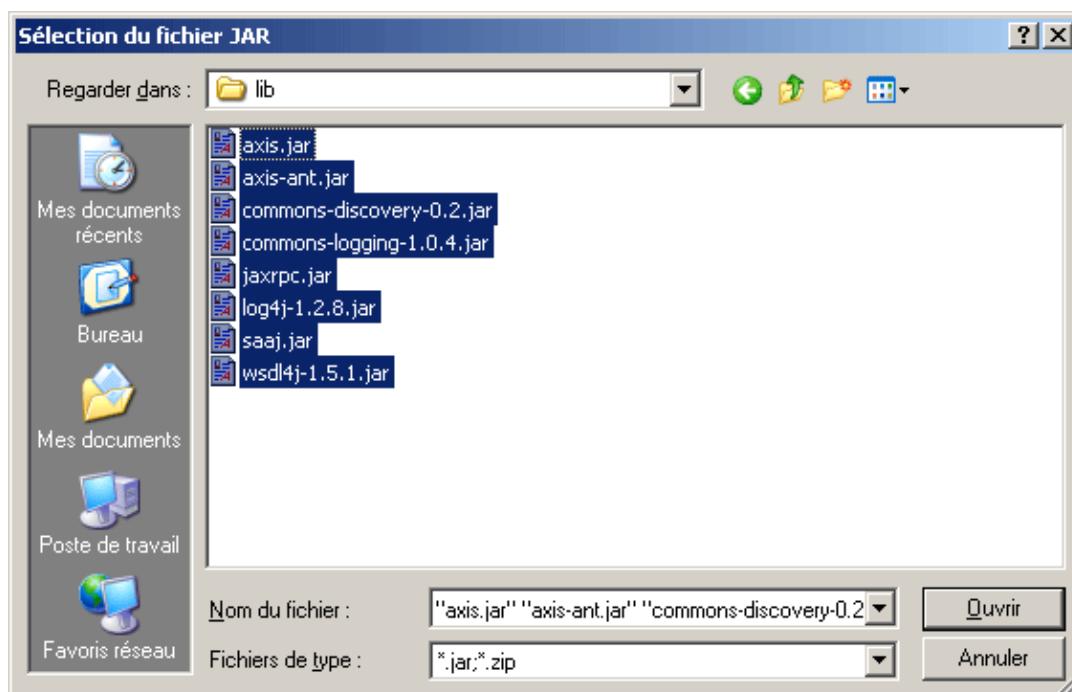


Cliquez sur le bouton « Nouveau ».

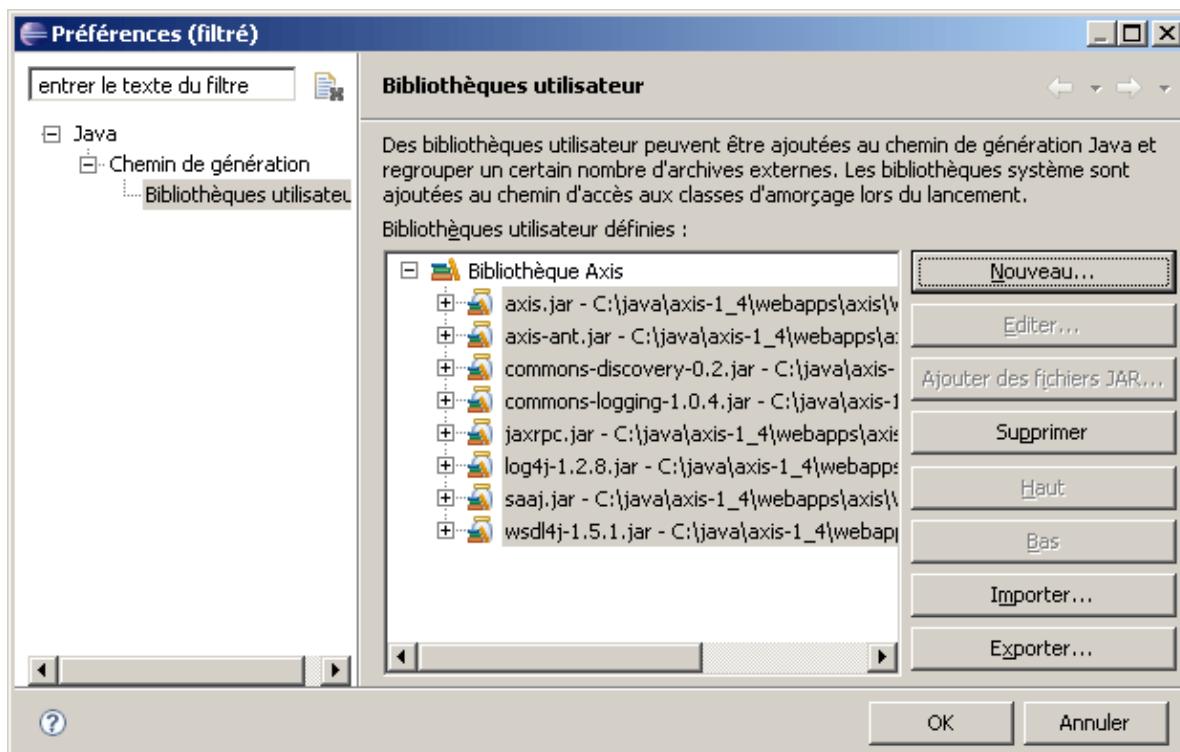


Saisissez le nom de la bibliothèque, par exemple « Bibliothèque Axis » et cliquez sur le bouton « OK ».

Cliquez sur le bouton « Ajouter des fichiers jar ... ».



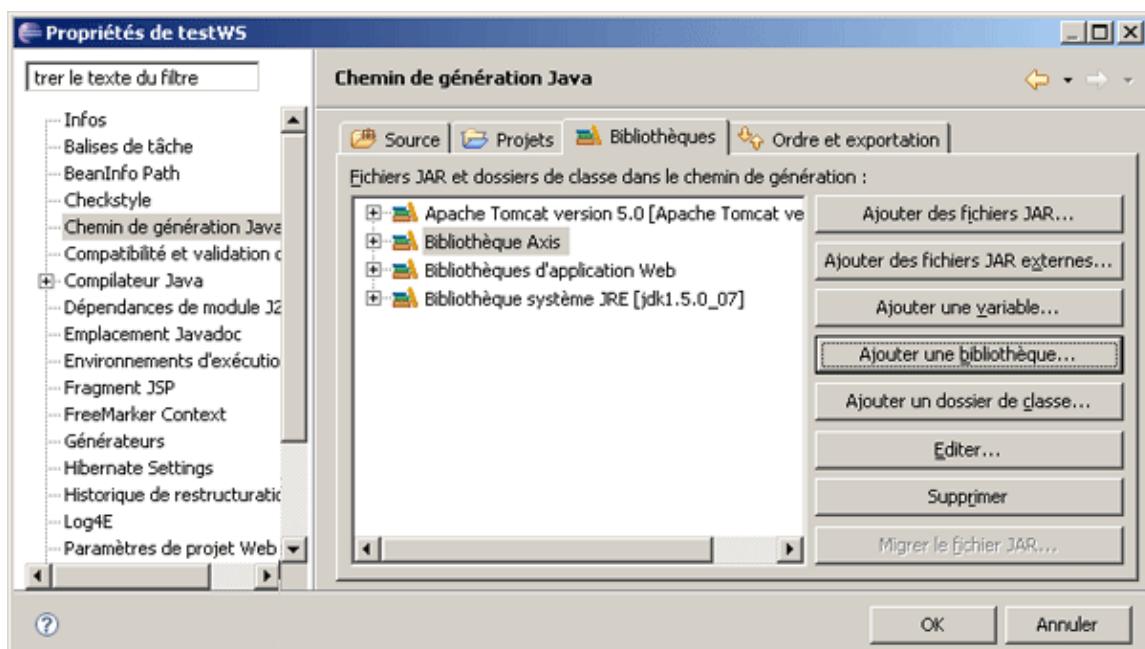
Selectionnez les fichiers .jar du répertoire lib et cliquez sur le bouton « Ouvrir »



Cliquez sur le bouton « OK ».



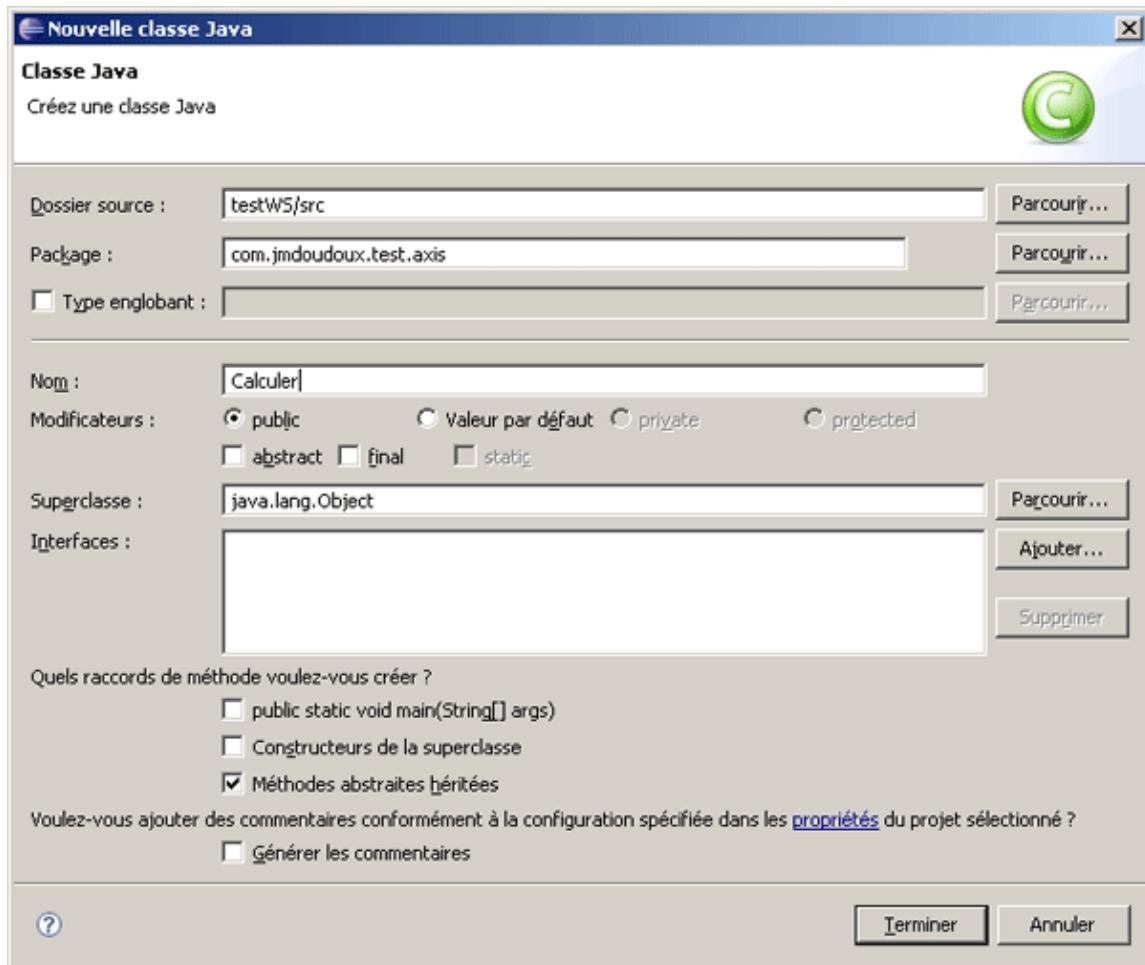
Cliquez sur le bouton « Terminer ».



Cliquez sur le bouton « OK ».

#### 26.1.4. La création d'une nouvelle classe

Créer une nouvelle entité de type « classe ».



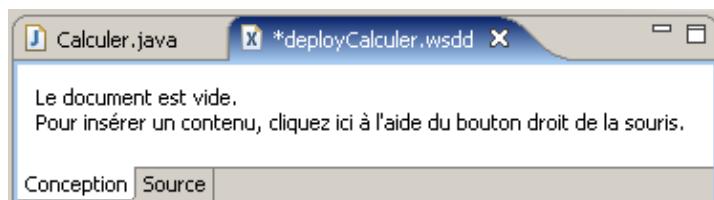
Saisissez le code d'une méthode nommée additionner() :

Exemple :

```
public int additionner(int valeur1, int valeur2) {  
    <b>return</b> valeur1 + valeur2;  
}
```

Dans le répertoire WEB-INF du projet, créer un répertoire deploy. La création de ce répertoire n'est pas obligatoire mais il permet de rassembler tous les fichiers de déploiement d'Axis.

Dans ce répertoire deploy, créez un fichier nommé Calculer.wsdd.



Saisissez le contenu du fichier :

Exemple :

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"  
           xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">  
  <service name="CalculerWS" provider="java:RPC" style="wrapped" use="literal">  
    <parameter name="className" value="com.jmdoudoux.test.axis.Calculer" />  
    <parameter name="allowedMethods" value="*" />  
    <parameter name="scope" value="Request" />  
  </service>  
</deployment>
```

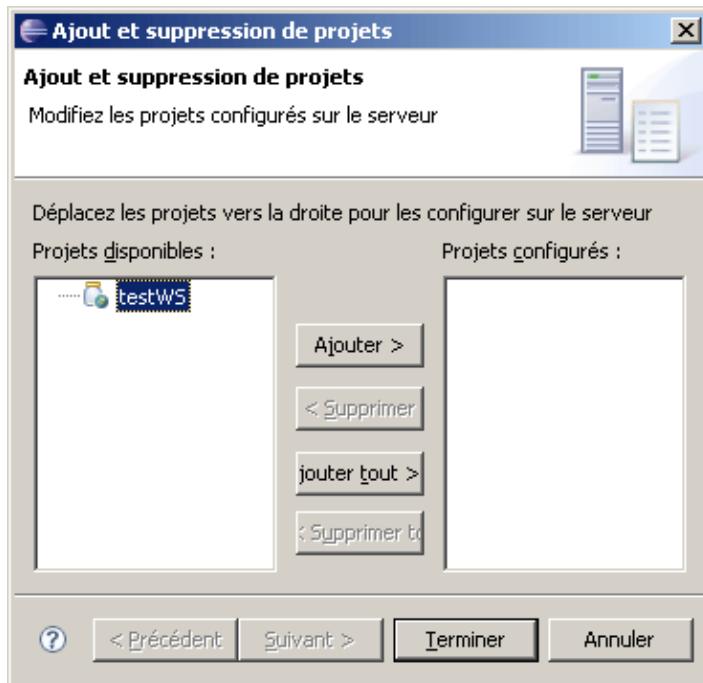
Créez un fichier undeployCalculator.wsdd dans le répertoire deploy et saisissez son contenu :

Exemple :

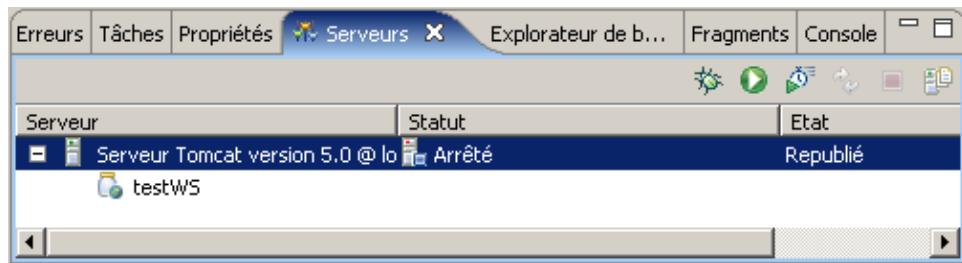
```
<undeployment xmlns="http://xml.apache.org/axis/wsdd/">  
  <service name="CalculerWS" />  
</undeployment>
```

### 26.1.5. Le lancement de l'application web

Dans la vue « Serveurs », sélectionnez le serveur Tomcat et utiliser l'option « Ajouter et supprimer des projets ... » du menu contextuel.



Cliquez sur le bouton « Ajouter > » pour faire basculer le projet testWS dans la liste des projets configurés, puis cliquez sur le bouton « Terminer ».



Cliquez sur le bouton pour démarrer le serveur.

### 26.1.6. Le déploiement du service dans Axis

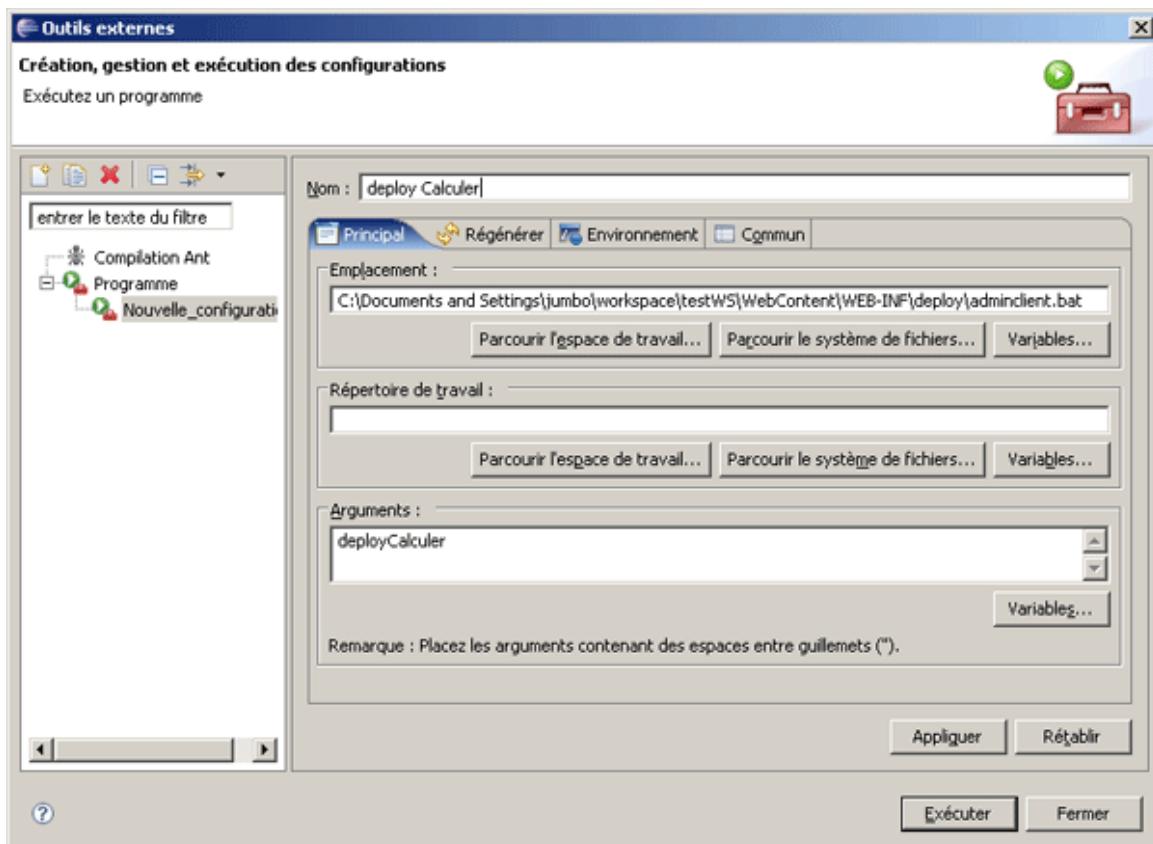
Le plus simple est de créer un script Dos qui va se charger du déploiement. Créez un fichier adminclient.bat dans le répertoire WEB-INF/deploy de la webapp et saisissez le code de ce script :

Exemple :

```
@echo off
cd "C:\Documents and Settings\jumbo\workspace\testWS\WebContent\WEB-INF\deploy"
set CLASSPATH=..\\lib\\axis.jar;..\\lib\\axis-ant.jar;..\\lib\\commons-discovery-0.2.jar;
..\\lib\\commons-logging-1.0.4.jar;..\\lib\\jaxrpc.jar;..\\lib\\log4j-1.2.8.jar;
..\\lib\\saaj.jar;..\\lib\\wsdl4j-1.5.1.jar
java org.apache.axis.client.AdminClient -s /testWS/services/AdminService %1%.wsdd
```

Adaptez simplement le chemin du répertoire deploy à votre contexte.

Pour exécuter ce script dans Eclipse, il faut utiliser l'option « Outils externes / Outils externes » du menu principal « Exécuter ».



Saisissez le nom de la configuration, par exemple « deploy Calculer »

Selectionnez l'emplacement du script et saisissez dans la zone de texte Arguments « deployCalculer ».

Cliquez sur le bouton « Exécuter ».

Créez de la même façon une configuration « undeploy Calculer » qui appelle le script adminclient.bat avec l'argument undeployCalculer.

La console affiche le résultat des traitements de déploiement

- Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.

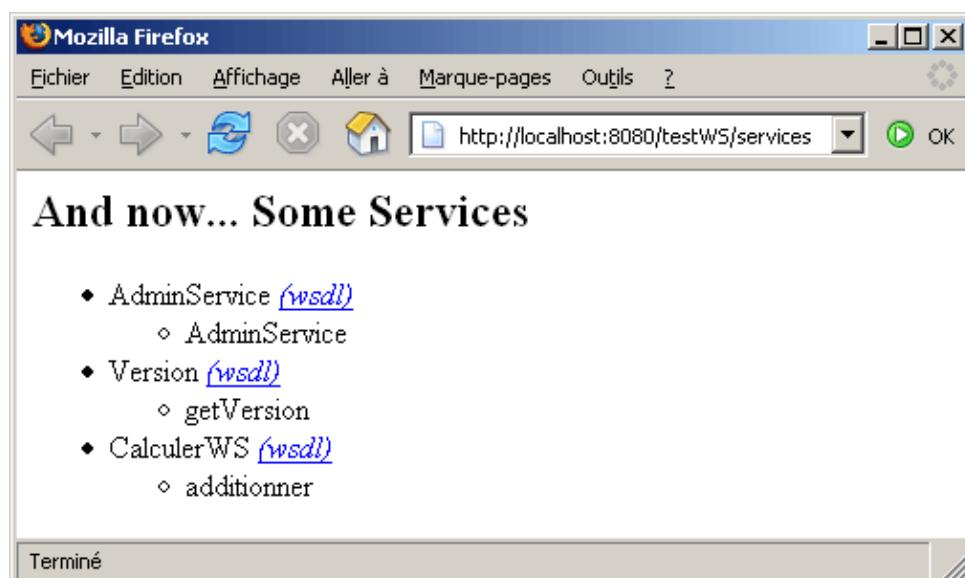
Processing file deployCalculer.wsdd

<Admin>Done processing</Admin>

Le message d'avertissement n'est pas important pour les besoins du service web.

### 26.1.7. La vérification du déploiement du service web

Ouvrez un navigateur et saisissez l'url <http://localhost:8080/testWS/services>



Le service web CalculerWS apparaît dans la liste des services.

Cliquez sur le lien wsdl du service pour afficher la description du service.

The screenshot shows the Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "Fichier", "Edition", "Affichage", "Aller à", "Marque-pages", "Outils", and "?". The toolbar includes back, forward, refresh, and home buttons. The address bar shows the URL "http://localhost:8080/testWS/services/CalculerWS?wsdl". The main content area displays the WSDL code:

```
<wsdl:definitions targetNamespace="http://localhost:8080/testWS/services/CalculerWS">
    <!--
        WSDL created by Apache Axis version: 1.4
        Built on Apr 22, 2006 (06:55:48 PDT)
    -->
    <wsdl:types>
        <schema elementFormDefault="qualified"
            targetNamespace="http://axis.test.jmdoudoux.com">
            <element name="additionner">
                <complexType>
                    <sequence>
                        <element name="valeur1" type="xsd:int"/>
                        <element name="valeur2" type="xsd:int"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </wsdl:types>
```

Saisissez l'url <http://localhost:8080/testWS/services/CalculerWS?method=additionner>

The screenshot shows the Microsoft Internet Explorer browser window. The title bar says "http://localhost:8080/testWS/services/CalculerWS?method=additionner - Microsoft Internet Explorer". The address bar shows the URL "http://localhost:8080/testWS/services/CalculerWS?method=additionner". The main content area displays the XML response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <soapenv:Fault>
            <faultcode>soapenv:Server.generalException</faultcode>
            <faultstring>Tried to invoke method public int
com.jmdoudoux.test.axis.Calculer.additionner(int,int) with arguments
null,null. The arguments do not match the signature.; nested
exception is: java.lang.IllegalArgumentException</faultstring>
            <detail>
                <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">thinkpad-
t23</ns1:hostname>
            </detail>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

La réponse contient une erreur puisque les paramètres de la méthode ne sont pas fournis.

Saisissez l'url :

<http://localhost:8080/testWS/services/CalculerWS?method=additionner&valeur1=10&valeur2=20>

```

<soapenv:Envelope>
  <soapenv:Body>
    <additionnerResponse>
      <nsl:additionnerReturn>30</nsl:additionnerReturn>
    </additionnerResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

La réponse SOAP contient le résultat de l'exécution du service.

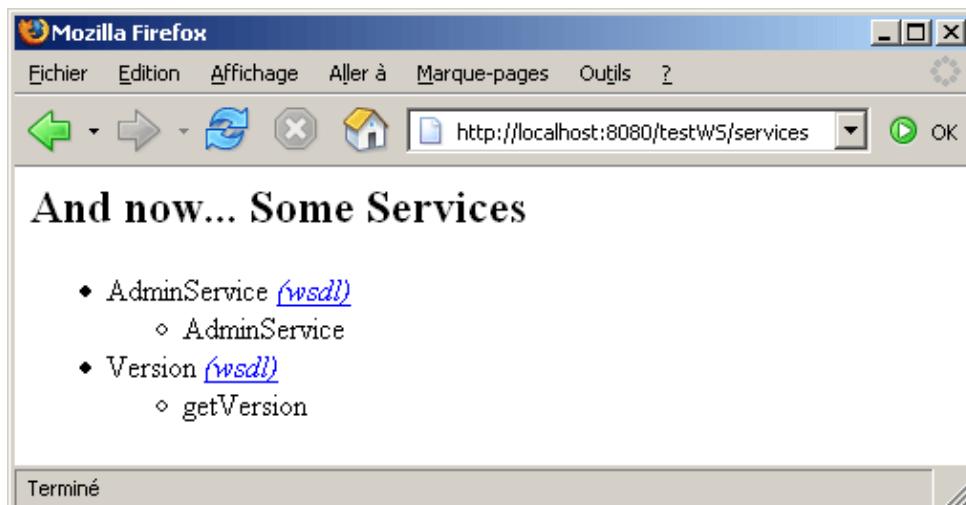
### 26.1.8. La modification du type du service web

Appelez la configuration externe « undeploy calculer » : la console affiche les traitements

- Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.

Processing file undeployCalculer.wsdd

<Admin>Done processing</Admin>



Modifiez le fichier Calculer.wsdd en remplaçant style= wrapped par style= document et enregistrez les modifications.

Utilisez la configuration externe « deploy Calculer » pour redéployer le service

Appelez l'url <http://localhost:8080/testWS/services/CalculerWS?wsdl>

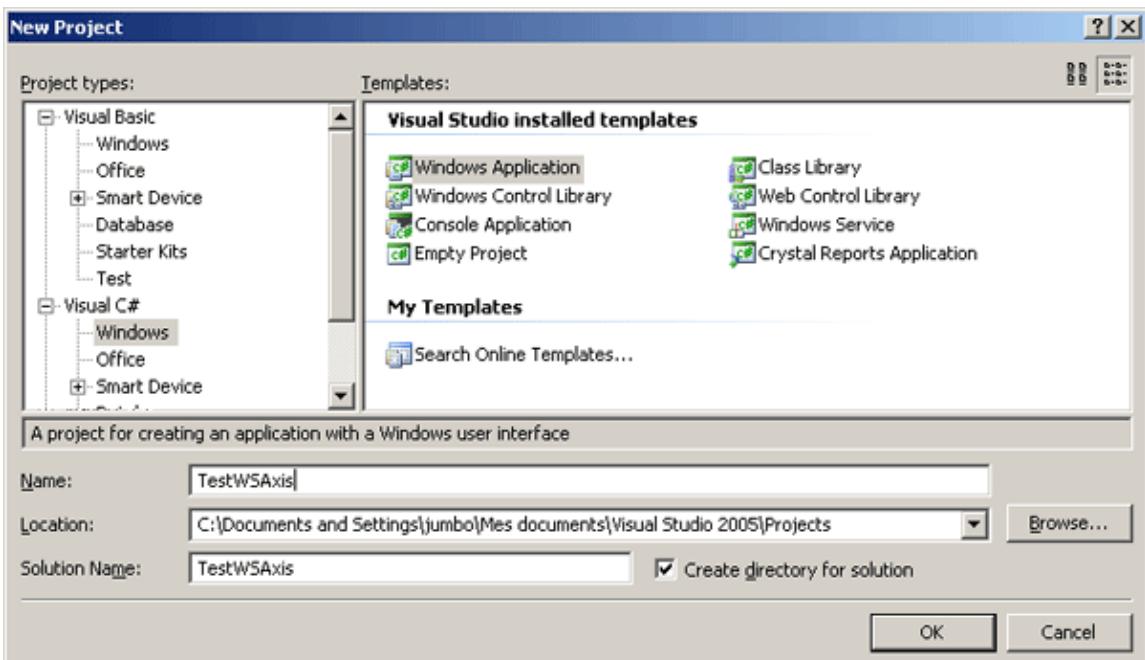
The screenshot shows the Mozilla Firefox browser window displaying the WSDL (Web Services Description Language) document for the 'CalculerWS' service. The URL in the address bar is `http://localhost:8080/testWS/services/CalculerWS?wsdl`. The content of the page is the XML structure of the WSDL file, which defines the service's operations, data types, and messages. The XML is color-coded for readability, with tags in purple and values in blue.

```
<wsdl:definitions targetNamespace="http://localhost:8080/testWS/services/CalculerWS">
  <!--
    WSDL created by Apache Axis version: 1.4
    Built on Apr 22, 2006 (06:55:48 PDT)
  -->
  <wsdl:types>
    <schema elementFormDefault="qualified"
      targetNamespace="http://axis.test.jmdoudoux.com">
      <element name="valeur1" type="xsd:int"/>
      <element name="valeur2" type="xsd:int"/gt;
    </schema>
    <schema elementFormDefault="qualified"
      targetNamespace="http://localhost:8080/testWS/services/CalculerWS">
      <element name="additionnerReturn" type="xsd:int"/gt;
    </schema>
  </wsdl:types>
  <wsdl:message name="additionnerResponse">
```

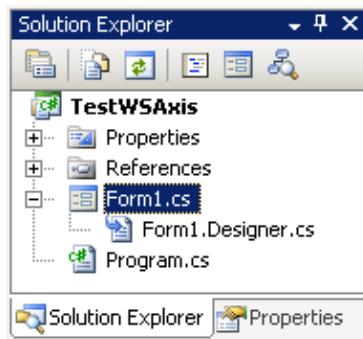
Le format du fichier wsdl est adapté par Axis en fonction du format précisé.

## 26.2. La consommation du service Web en .Net

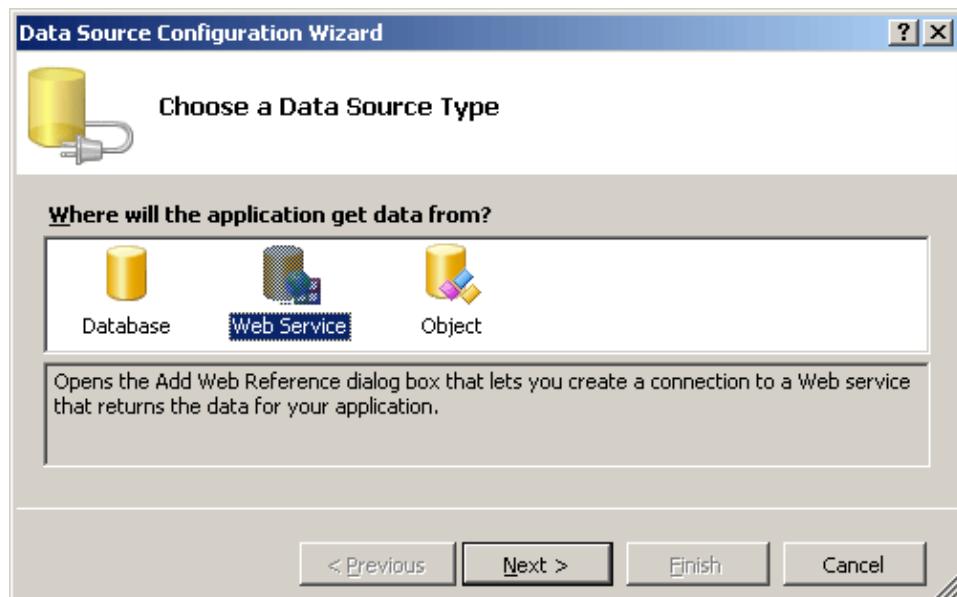
Sous Visual Studio 2005, créez un nouveau projet de type « Visual C# / Windows / Windows Application » nommé TestWSAxis :



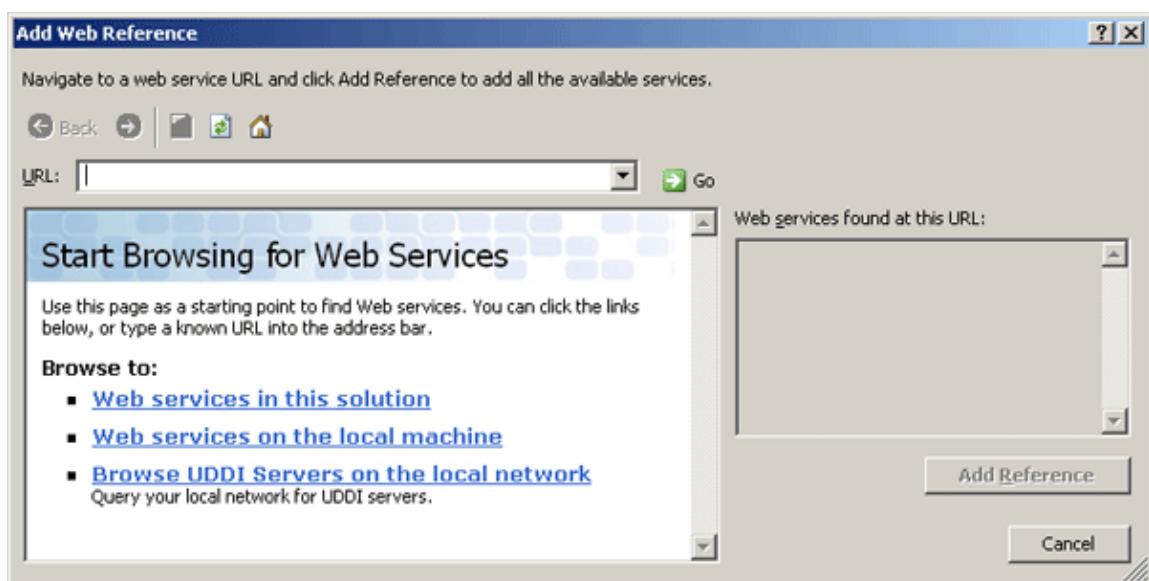
Le projet est créé avec une fenêtre principale vide



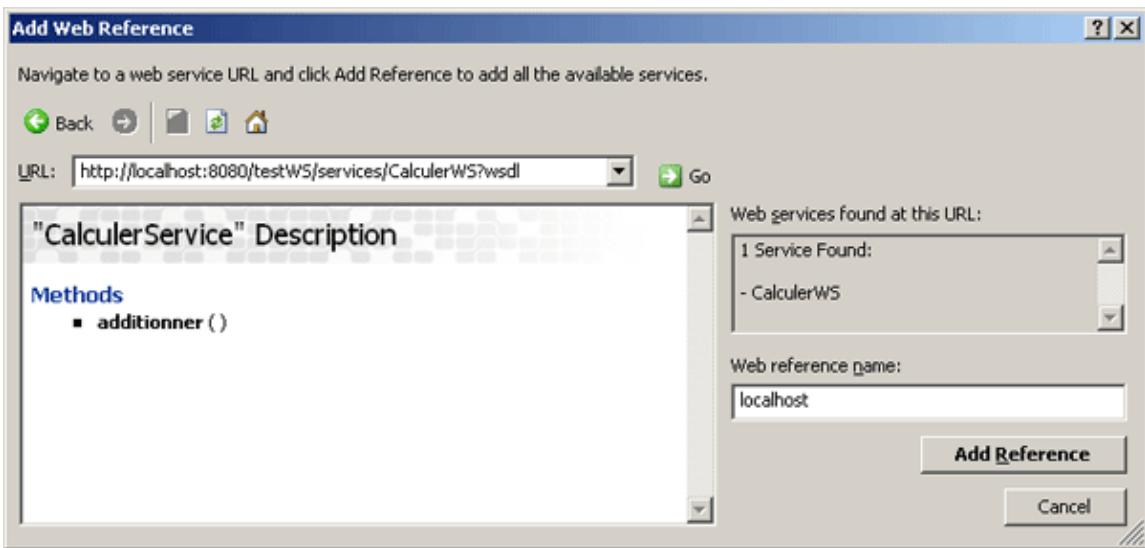
Sélectionnez l'option « Add new DataSource » du menu principal « Data »



Sélectionnez « Web Service » et cliquez sur bouton « Next ».

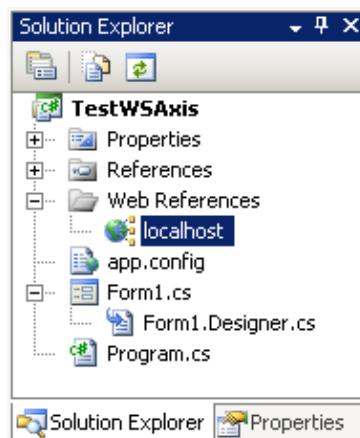


Saisissez l'url <http://localhost:8080/testWS/services/CalculerWS?wsdl> et cliquez sur le bouton « Go ».



Le document wsdl est analysé et les méthodes trouvées sont affichées.

Saisissez le nom CalculerWS et cliquez sur le bouton « Add Reference ».



Une « Web Reference » concernant le service web est ajoutée. Renommez la en CalculerWS.

Visual Studio a créé un fichier Reference.cs dans le sous répertoire Web References\CalculerWS du projet. Ce fichier contient une classe qui fait office de proxy vers le service web. Le contenu de ce fichier est automatiquement généré par Visual Studio à partir du wsdl du service web.

Dans la fenêtre du projet, ajouter un composant de type TextBox et un composant de type Button.



Ajouter un événement sur le clic du bouton en double cliquant dessus. L'éditeur s'ouvre sur la méthode générée.

Ajouter une clause using concernant la web reference générée.

```
using TestWSAxis.CalculerWS;
```

Dans le corps de la méthode, saisissez le code de l'appel du service web.

Exemple :

```
CalculerService cs = new CalculerService();
int resultat = cs.additionner(10, 20);
textBox1.Text = ""+resultat;
```

Remarque : ce code est spartiate puisqu'il ne contient aucune gestion des erreurs qui peuvent survenir dans ces traitements.

Appuyer sur la touche F5 pour compiler le projet et l'exécuter.

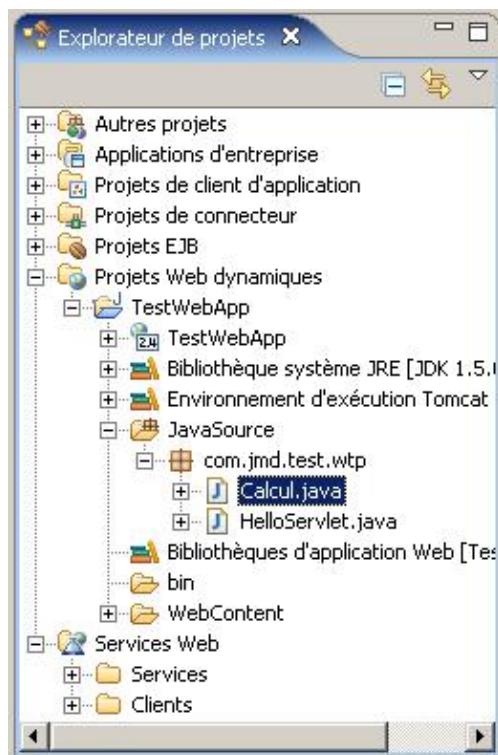


## 26.3. Le développement de services web avec WTP 1.0

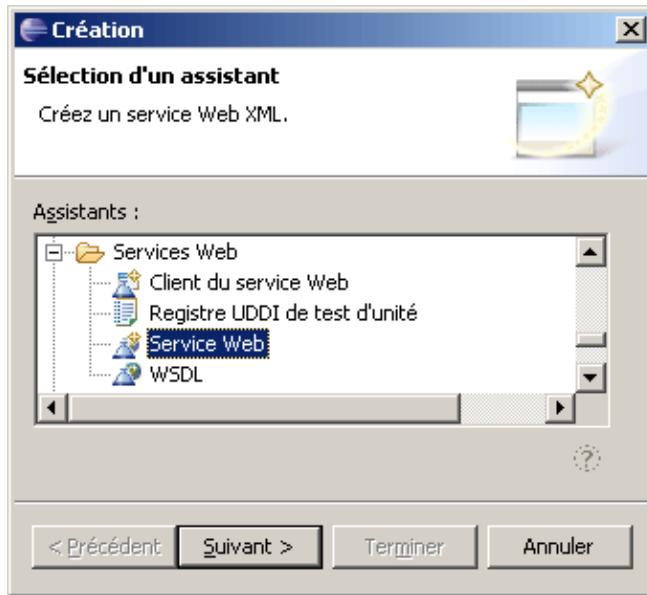
Le plug in WTP propose plusieurs fonctionnalités pour faciliter le développement de services web.

### 26.3.1. Convertir une classe en service web

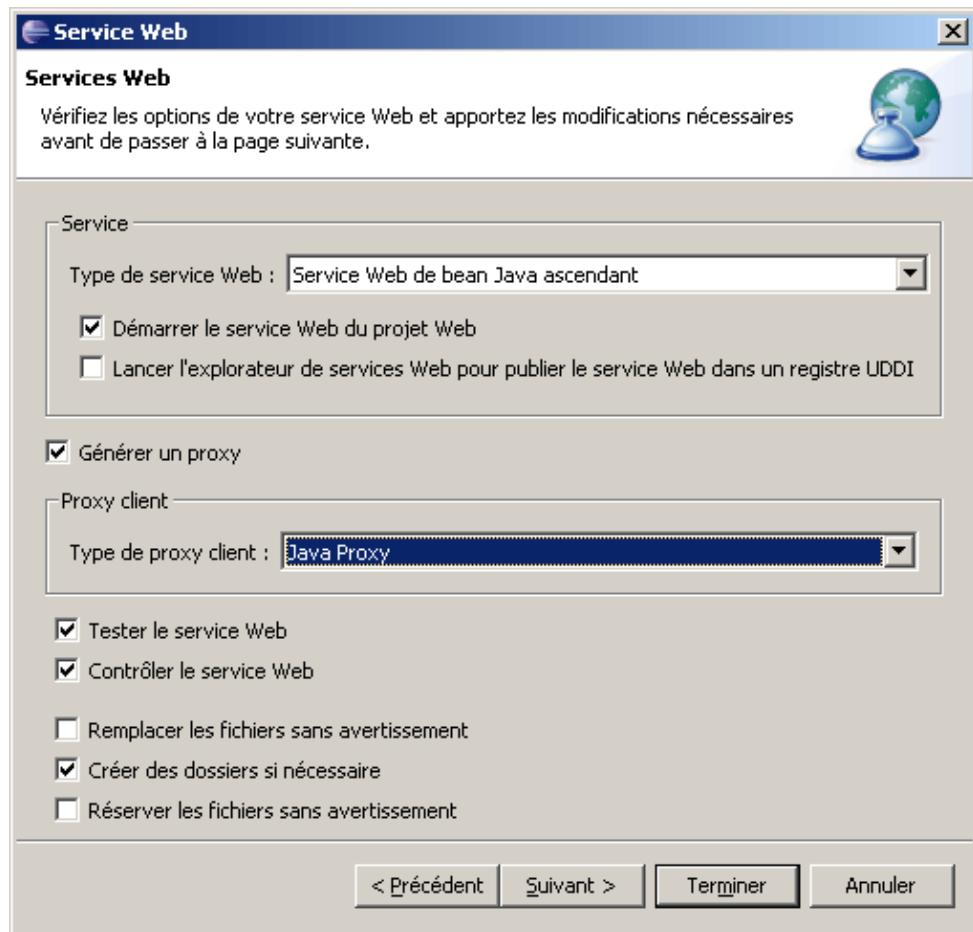
Le plug-in propose de convertir automatiquement une classe en un service web, reposant sur Axis du groupe Apache Jakarta.



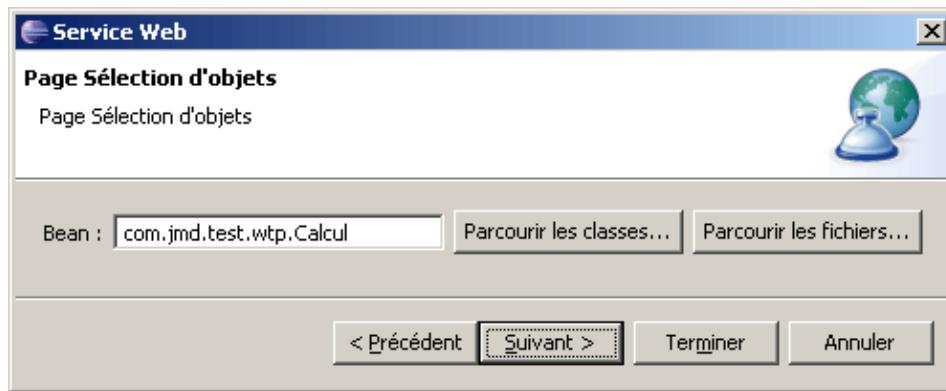
Il faut sélectionner la classe et utiliser l'option « Nouveau/Autre ... »



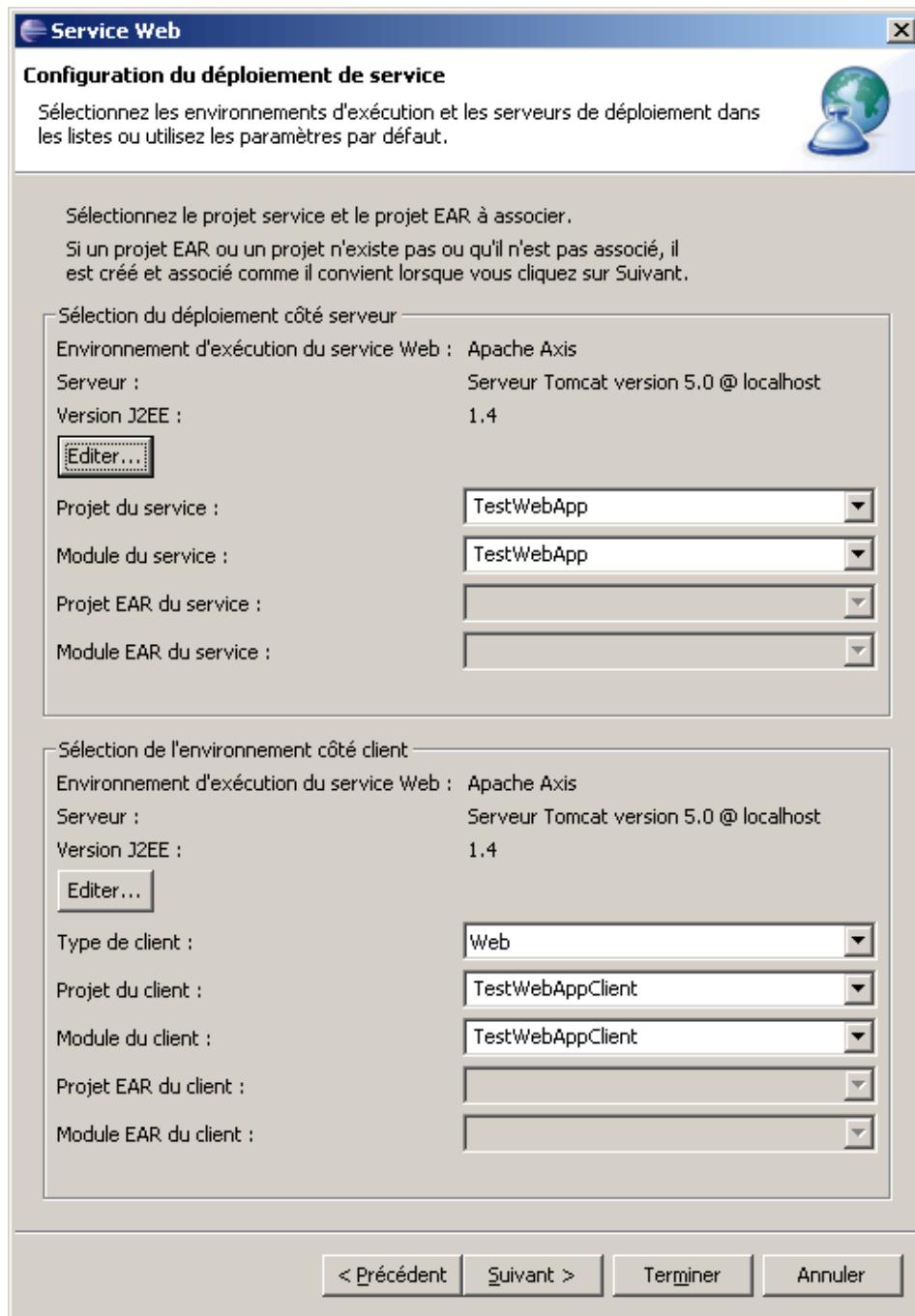
Sélectionnez « Services Web/Service Web » et cliquez sur le bouton « Suivant »



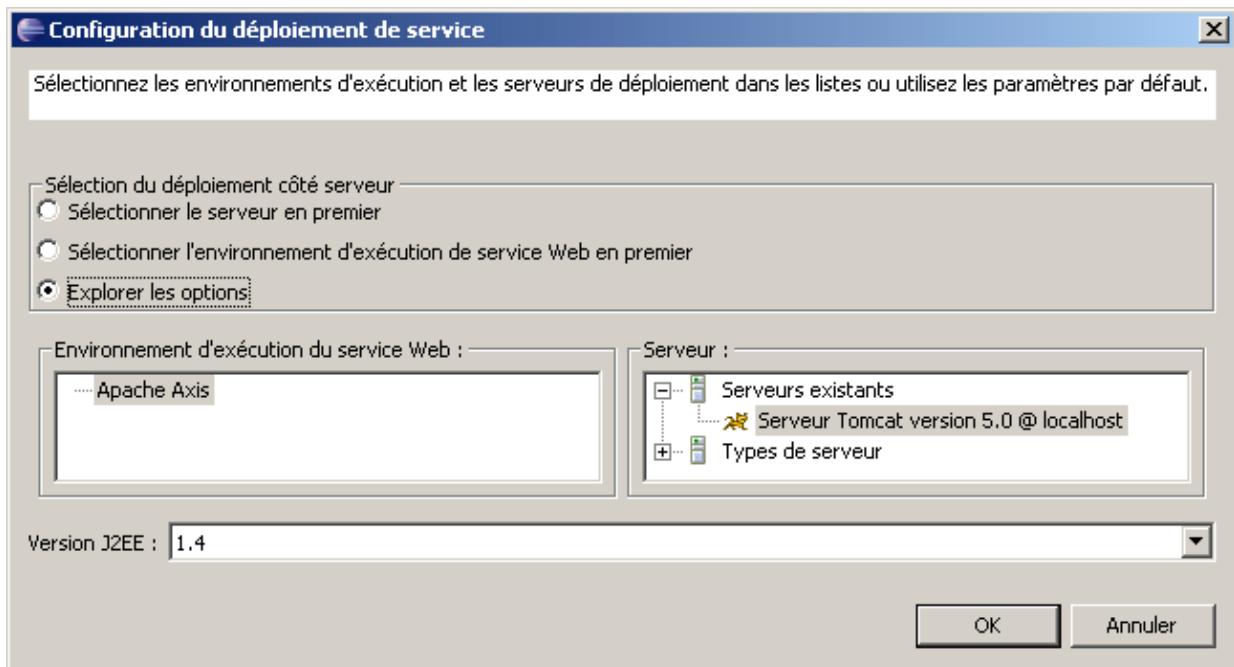
Cochez « Générer un proxy », « Tester le service web » et « Contrôler le service web » et cliquez sur le bouton « Suivant ».



Par défaut, la classe sélectionnée est utilisée. Cliquez sur le bouton « Suivant ».

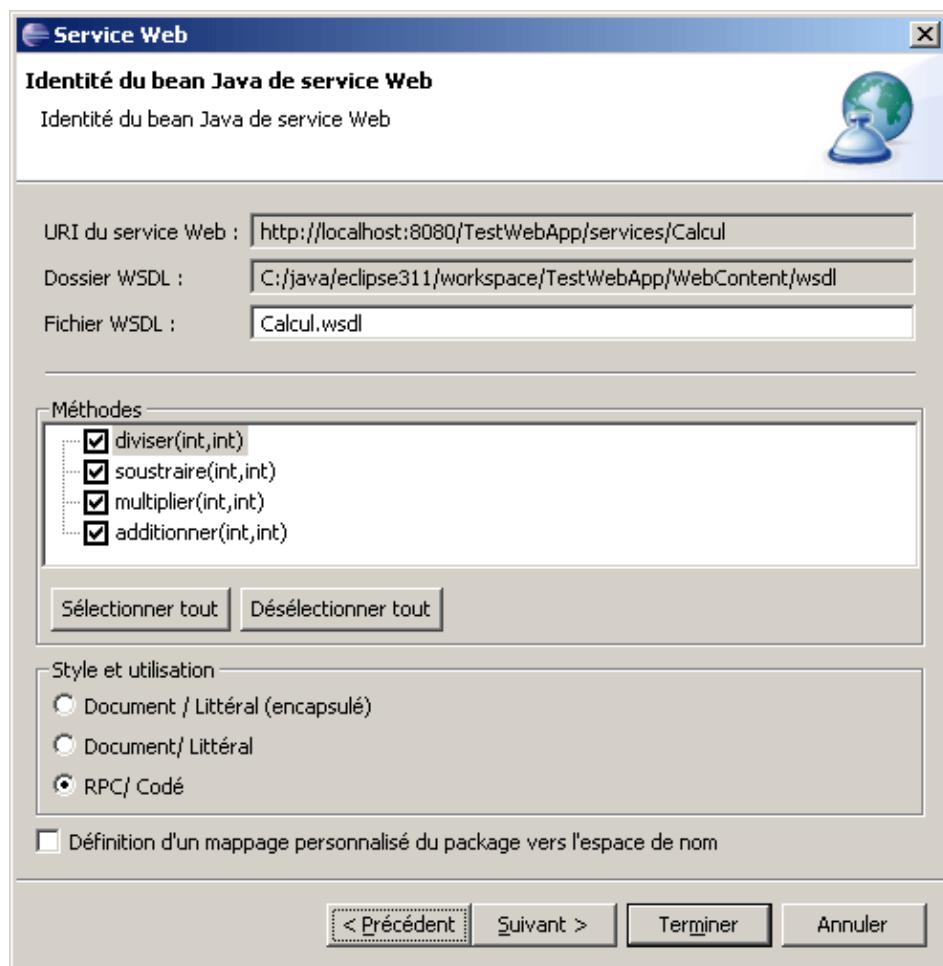


Le bouton « Editer » de la partie « Sélection du déploiement côté serveur » permet de sélectionner le serveur et l'environnement d'exécution.

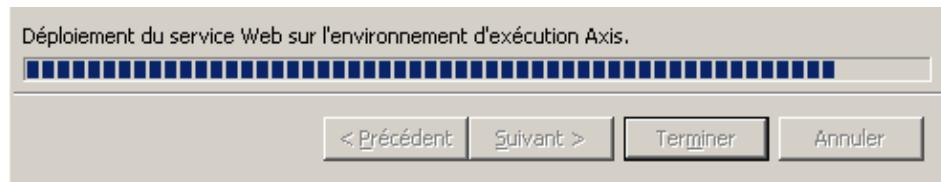


Une fois le serveur sélectionné, cliquez sur le bouton « OK ».

Cliquez sur le bouton « Suivant » pour afficher la page « Identité du bean Java de service web » de l'assistant

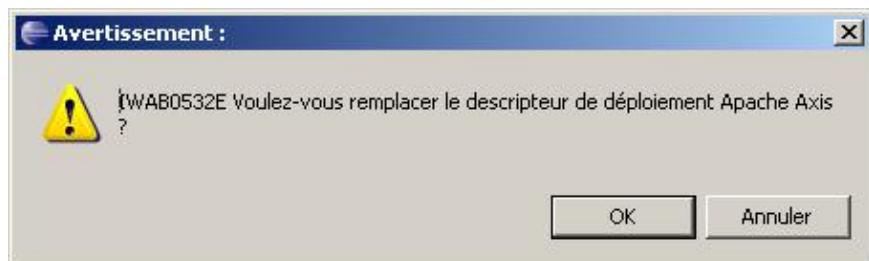


Cette page permet de sélectionner les informations qui composeront le fichier .wsdl. Cliquez sur le bouton « Suivant ».

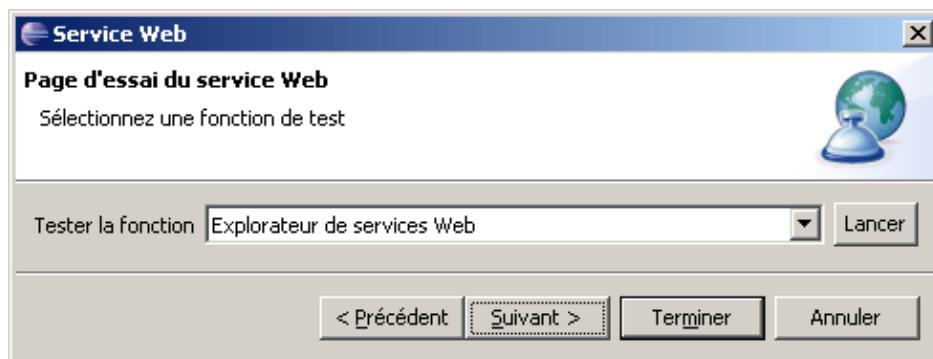


L'assistant génère les fichiers correspondant au service.

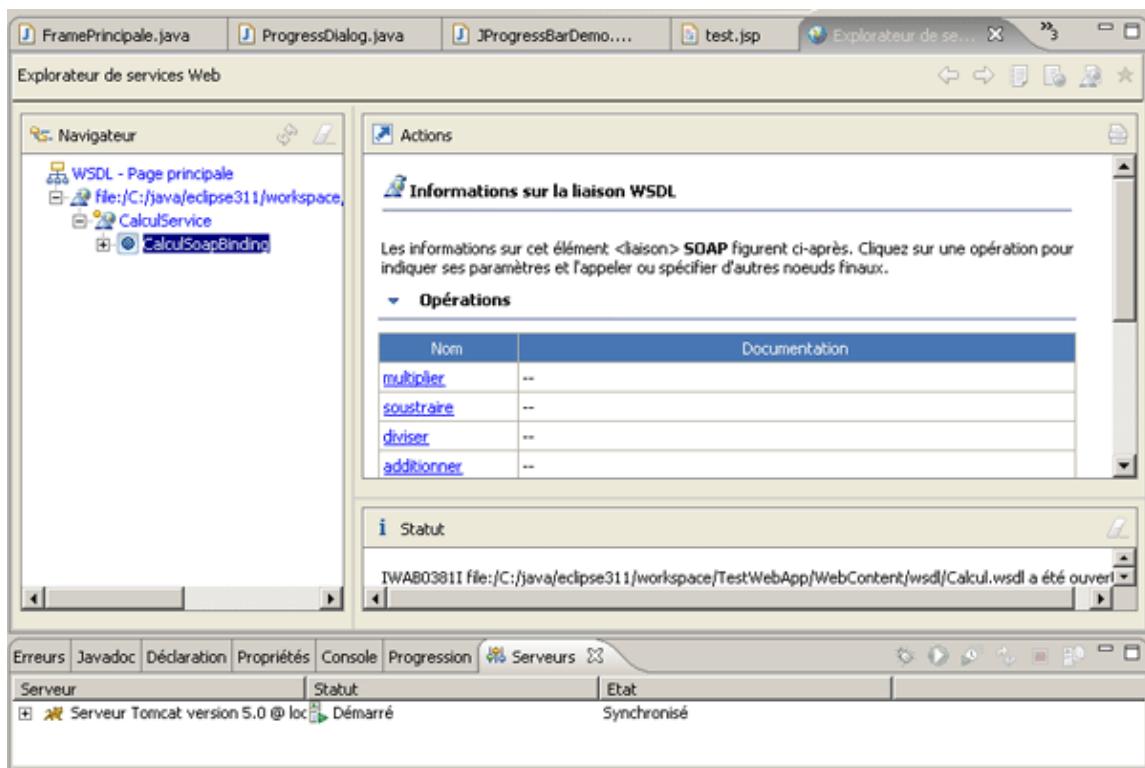
Une confirmation est demandée dans le cas d'un remplacement d'un descripteur existant.



La page suivante de l'assistant permet de demander le test du service Web.



Cliquez sur le bouton « Lancer » pour démarrer le service dans le serveur.



La vue Explorateur de services web s'ouvre

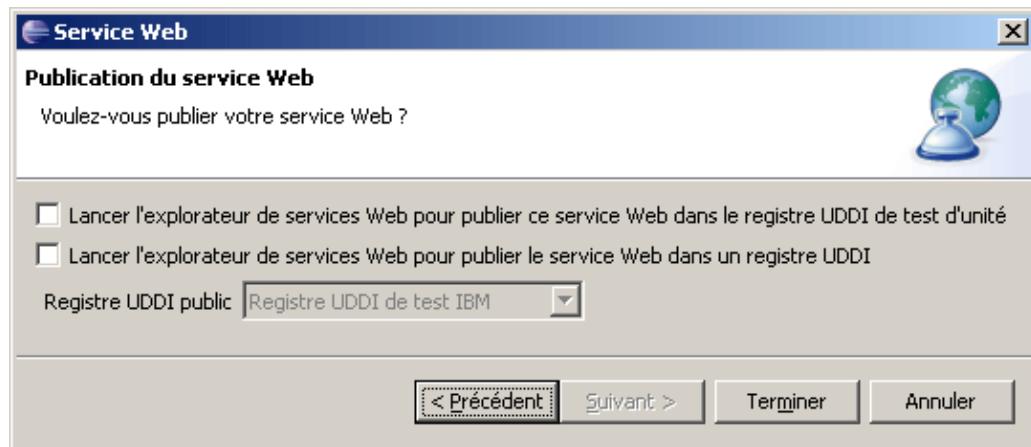
Cliquez sur le bouton « Suivant » de l'assistant.



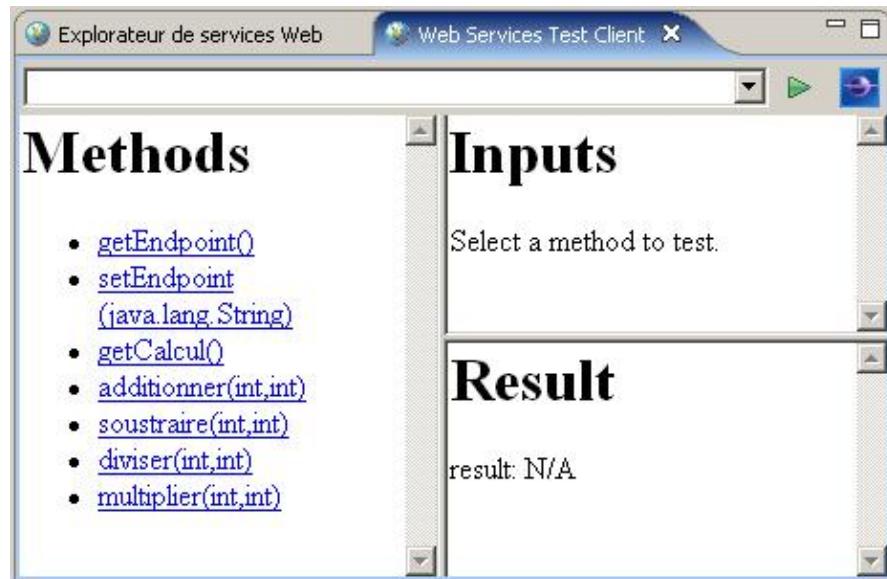
La page « Page Proxy de service web » permet fournir des précisions sur le proxy qui sera généré. Cliquez sur le bouton « Suivant »



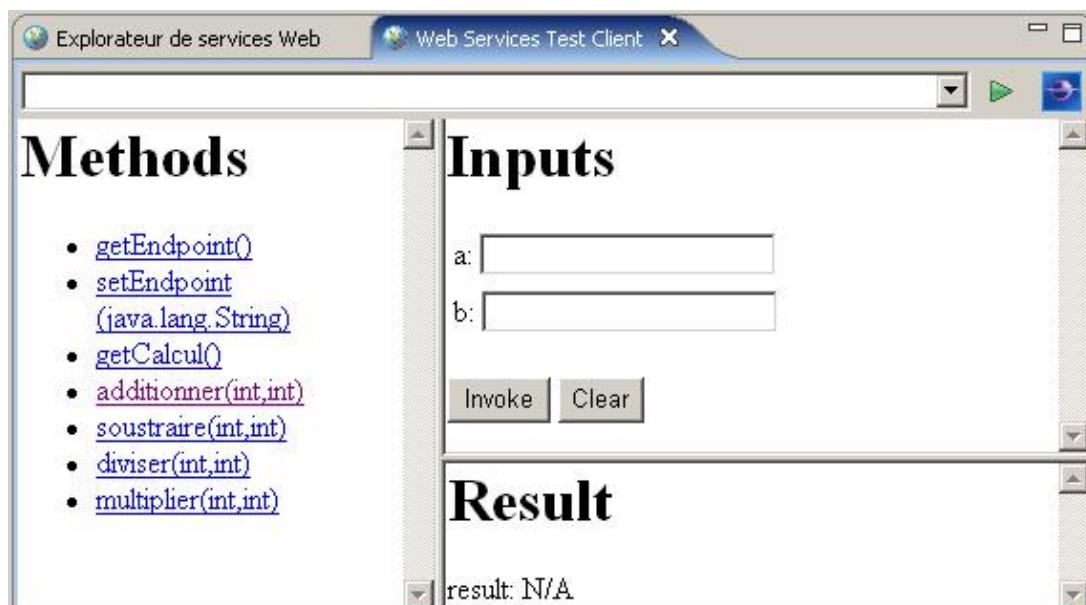
Cette page permet de tester le proxy généré. Cliquez sur le bouton « Suivant »



Cette page permet de demander la publication du service web. Cliquez sur le bouton « Terminer »

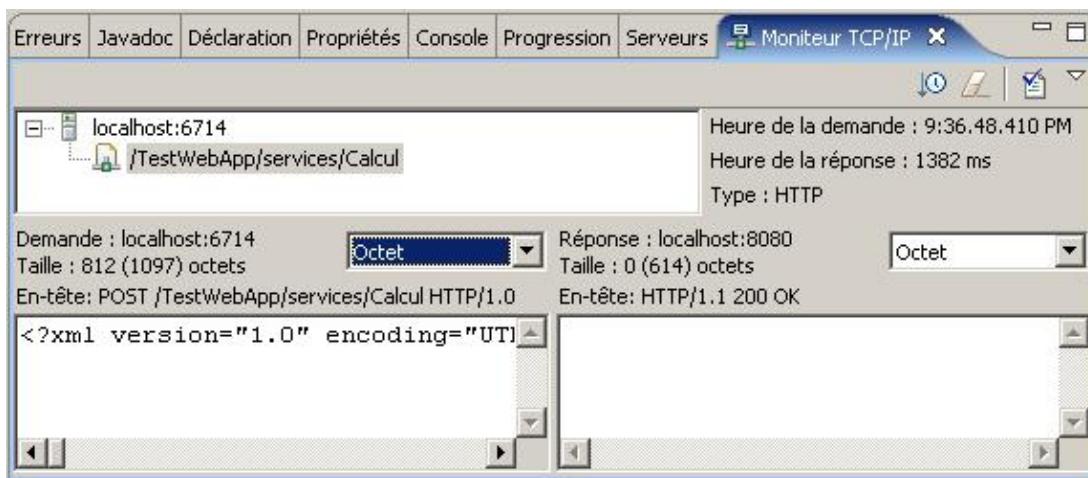


La vue « Web Services Test Client » s'ouvre. Il suffit de cliquer sur une méthode dans la partie « Methods », par exemple la méthode additionner().



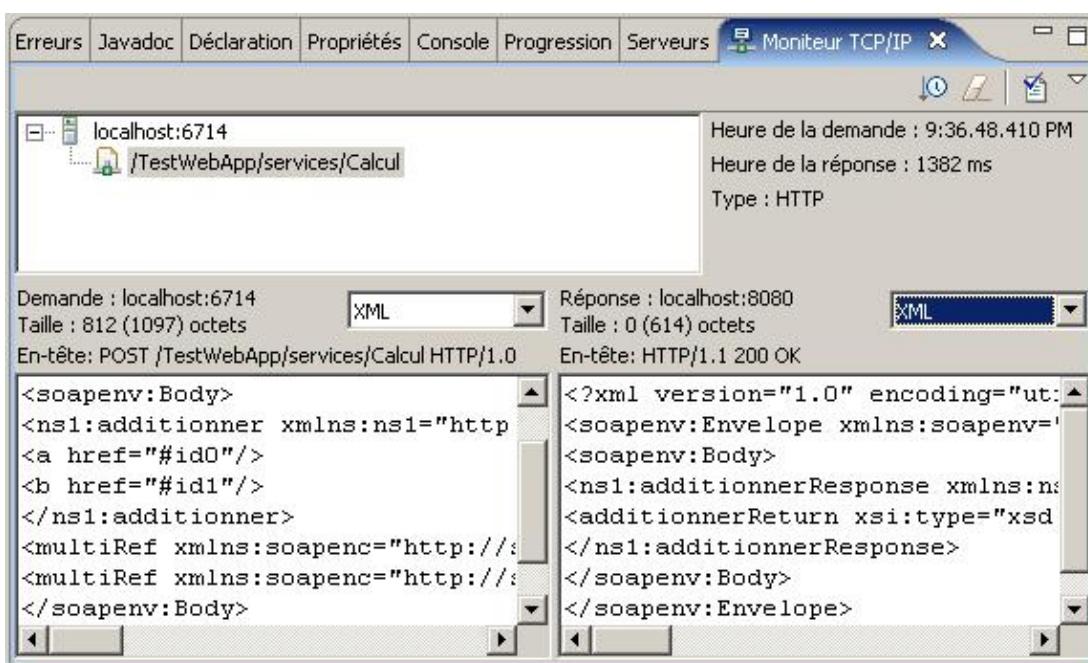
La partie « Inputs » permet de saisir les valeurs des deux paramètres de la méthode sélectionnée.

Il suffit de saisir les deux valeurs et de cliquer sur le bouton « Invoke »

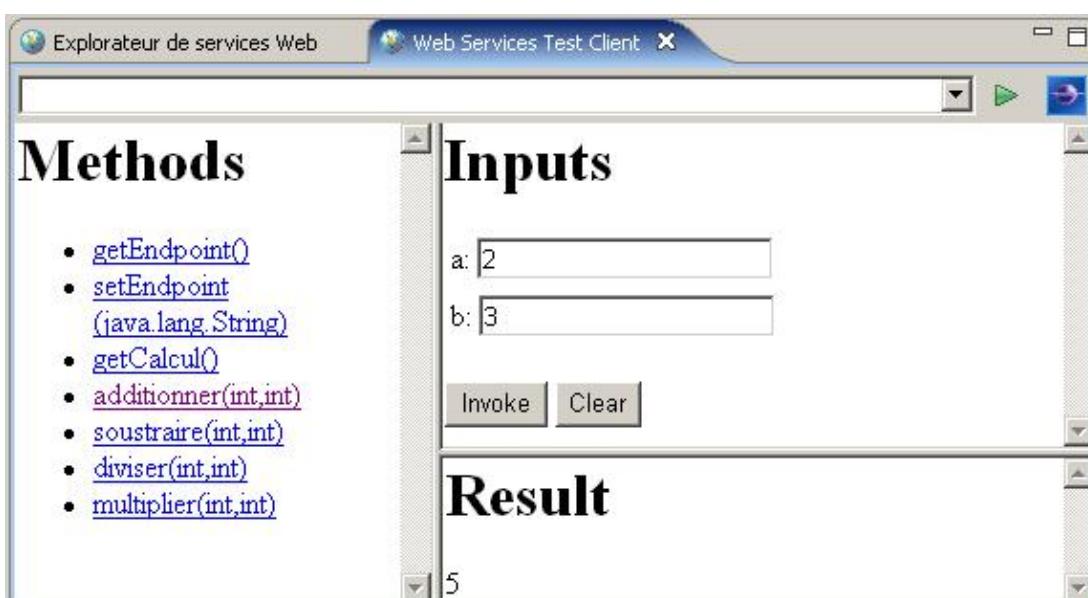


La vue « Moniteur TCP/IP » s'ouvre et affiche de détails de l'appel du service web.

Il est possible de modifier le format d'affichage de la requête et de la réponse en utilisant les listes déroulantes.

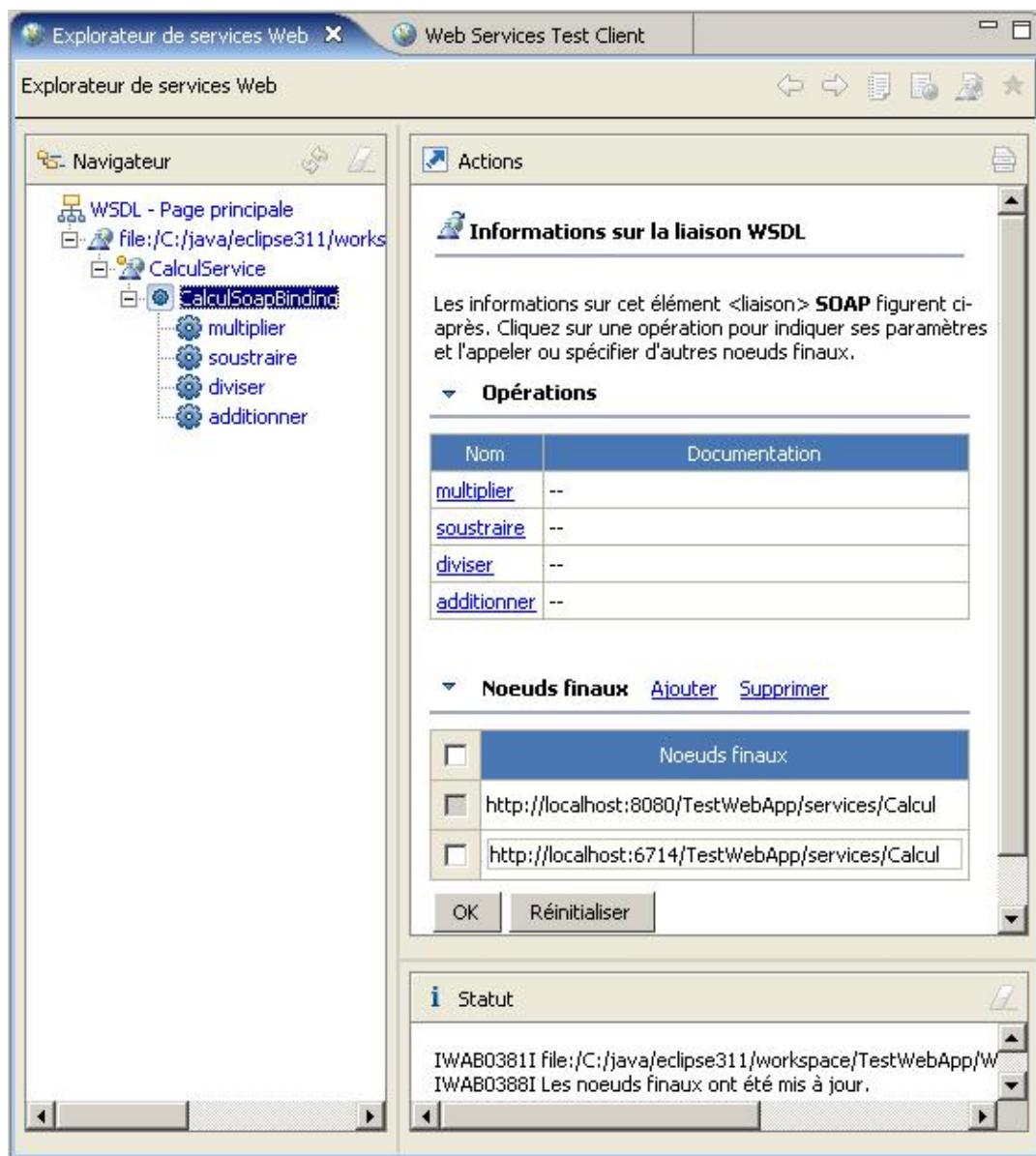


Le résultat de l'appel au service web s'affiche dans la partie « Result » de la vue « Web Services Test Client »

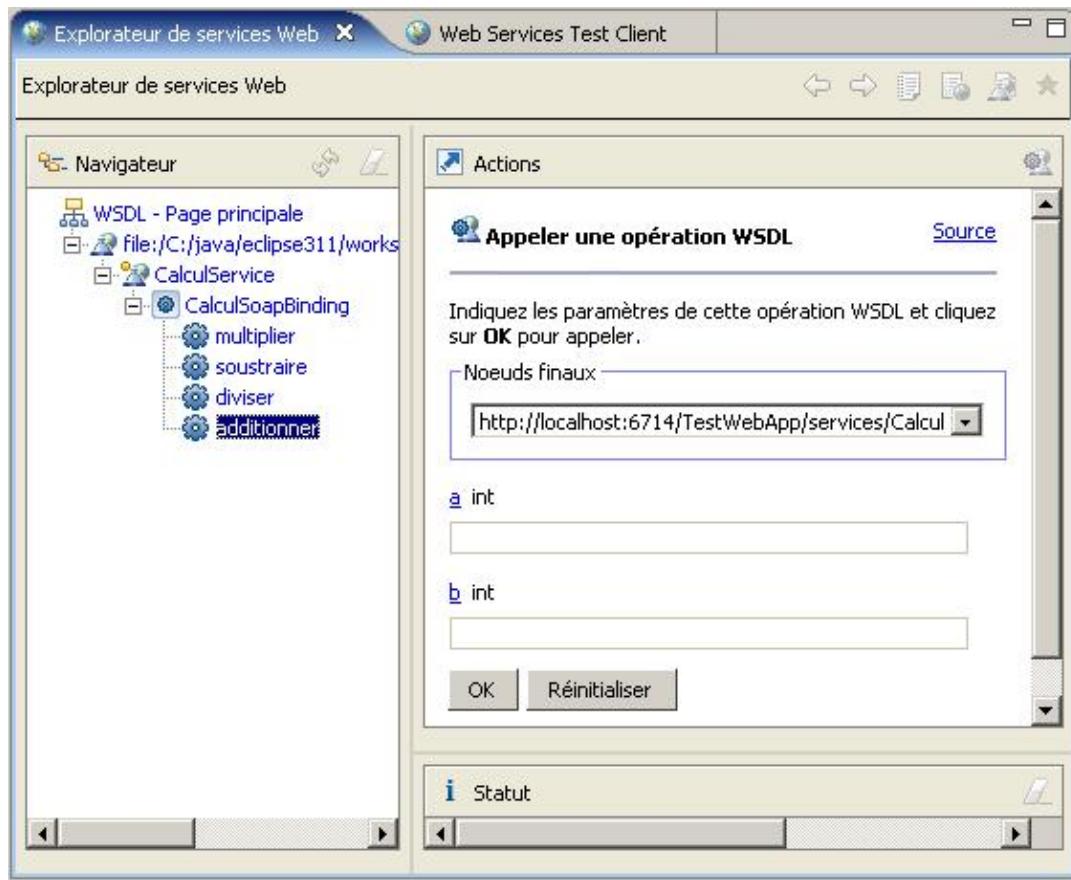


La vue « Explorateur de services web » permet d'obtenir des informations sur les services web.

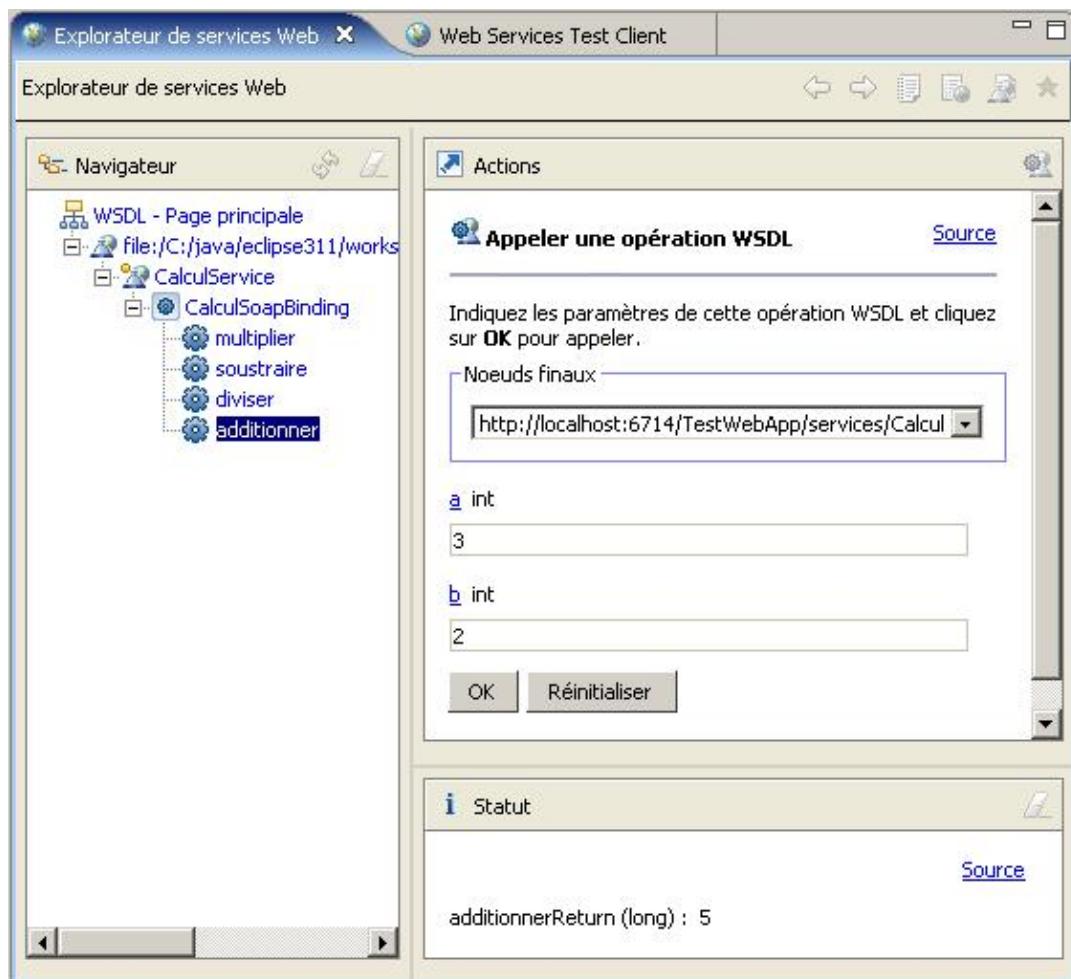
Pour afficher cette vue, il suffit de sélectionner le fichier .wsdl du service web dans le répertoire WebContent/wsdl et d'utiliser l'option « Services web/Tester avec un explorateur de services web ».



Il suffit de cliquer sur une méthode du service pour la tester.

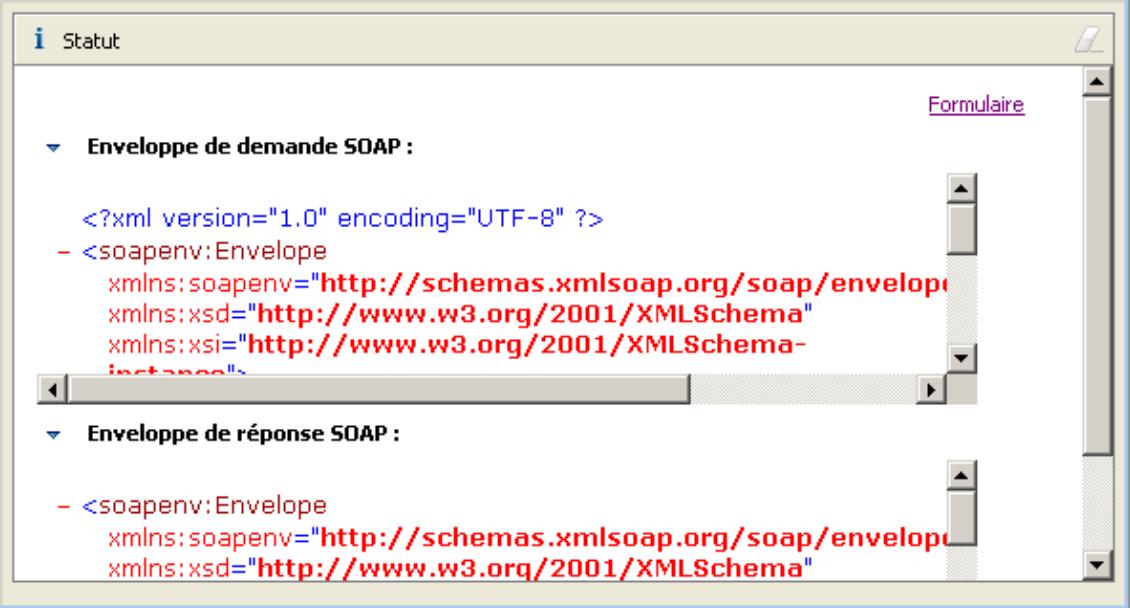


Il suffit de saisir les paramètres et de cliquer sur le bouton « OK ».



Le service est appelé avec les paramètres et le résultat est affiché dans la partie « Statut »

En cliquant sur le lien « Source » est possible d'afficher la requête et la réponse SOAP.



The screenshot shows the Eclipse IDE's XML Editor window. The title bar says "Statut". The main area contains two sections: "Enveloppe de demande SOAP :" and "Enveloppe de réponse SOAP :". The "Enveloppe de demande SOAP :" section displays the following XML code:

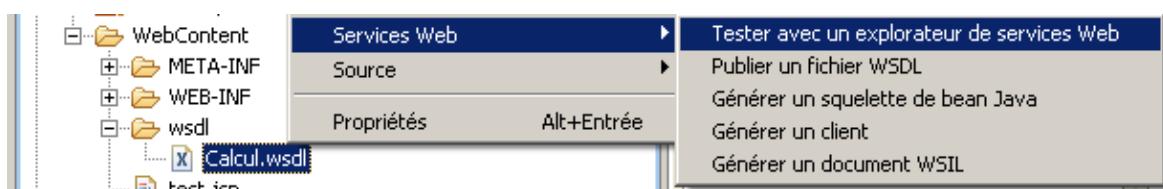
```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

The "Enveloppe de réponse SOAP :" section displays the following XML code:

```
- <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

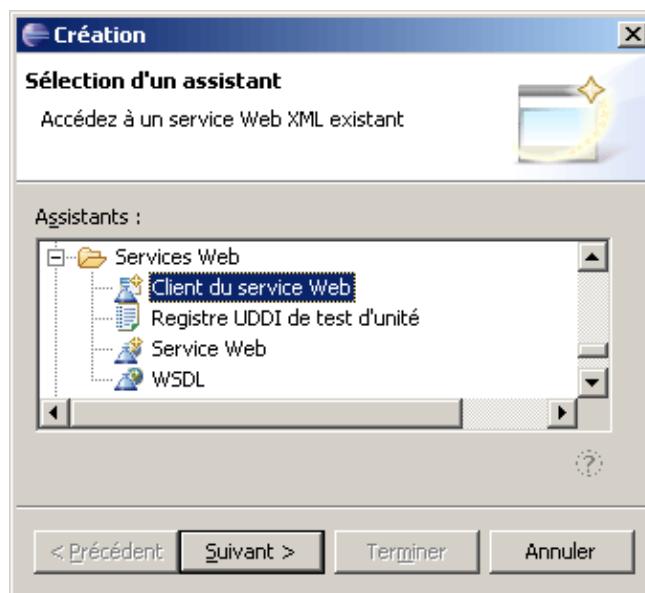
### 26.3.2. Les différentes fonctionnalités sur les services web

Le menu contextuel « Services web » du fichier .wsdl propose plusieurs fonctionnalités.

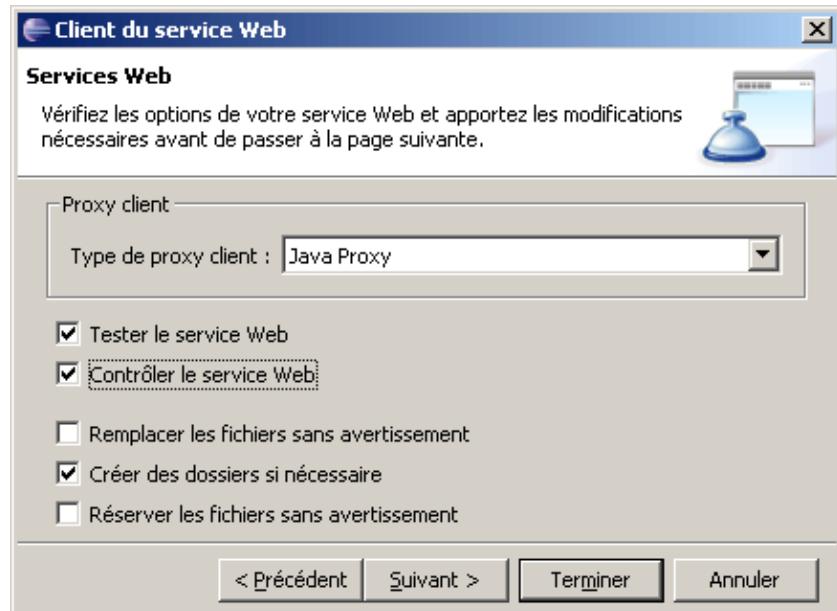


L'option « Services web/Publier un fichier WSDL » permet d'enregistrer le service web dans un annuaire UDDI

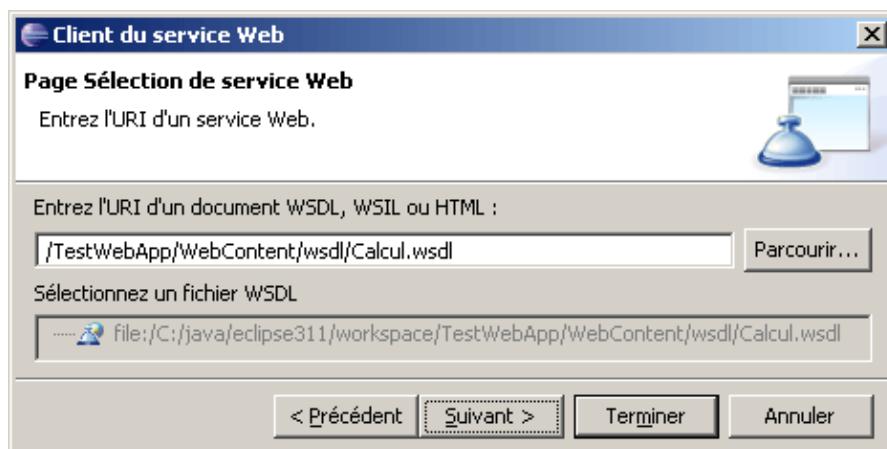
L'option « Services web/Générer un client » permet de générer une application web pour tester le service web. Il est aussi possible de sélectionner l'option « Nouveau/Autre » du menu contextuel du fichier .wsdl.



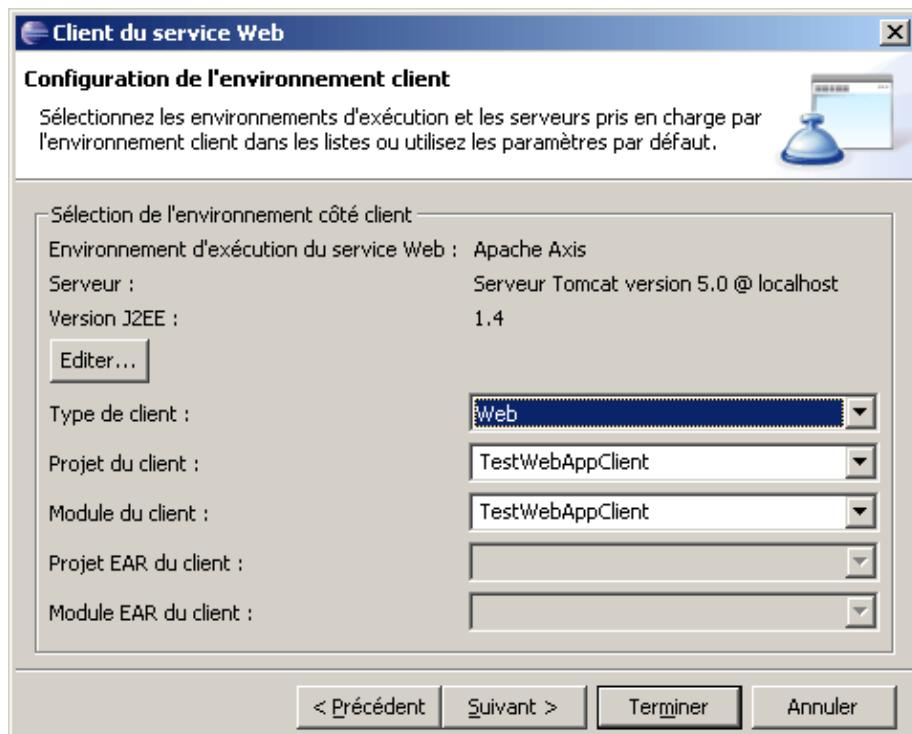
Un assistant permet de saisir les caractéristiques de l'application à générer.



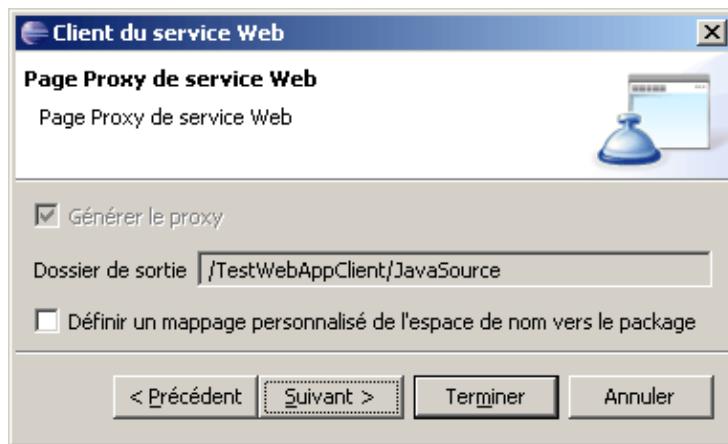
L'option « Contrôler le service Web » permet de lancer le Moniteur TCP/IP. Cliquez sur le bouton « Suivant »



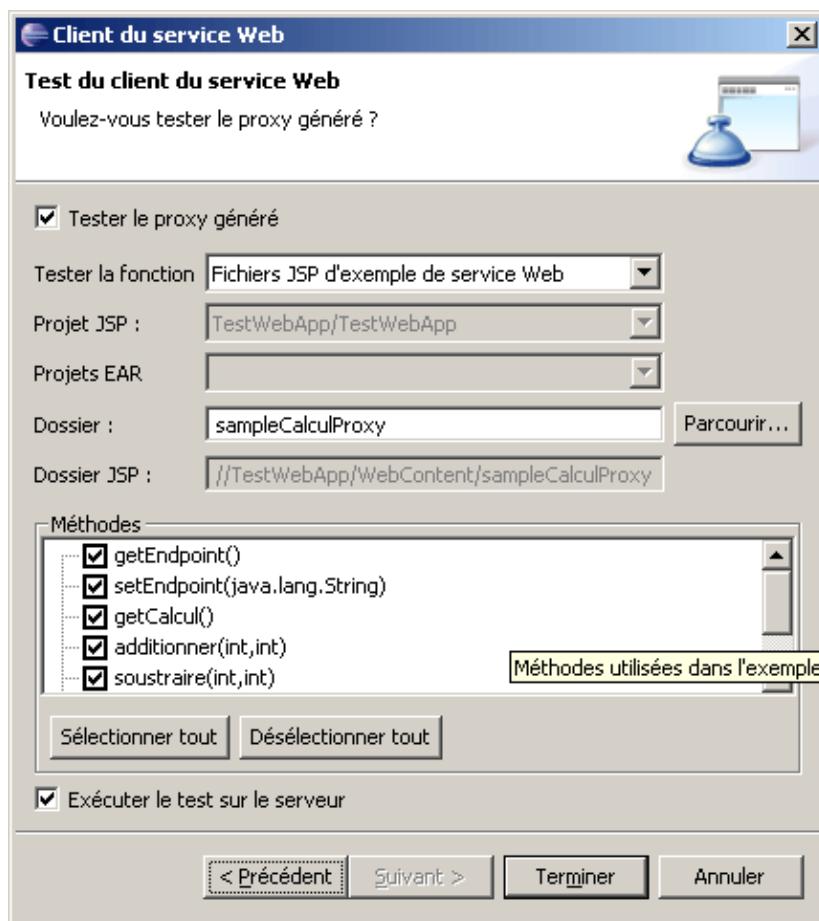
Cliquez sur le bouton « suivant »



Cliquez sur le bouton « Suivant »

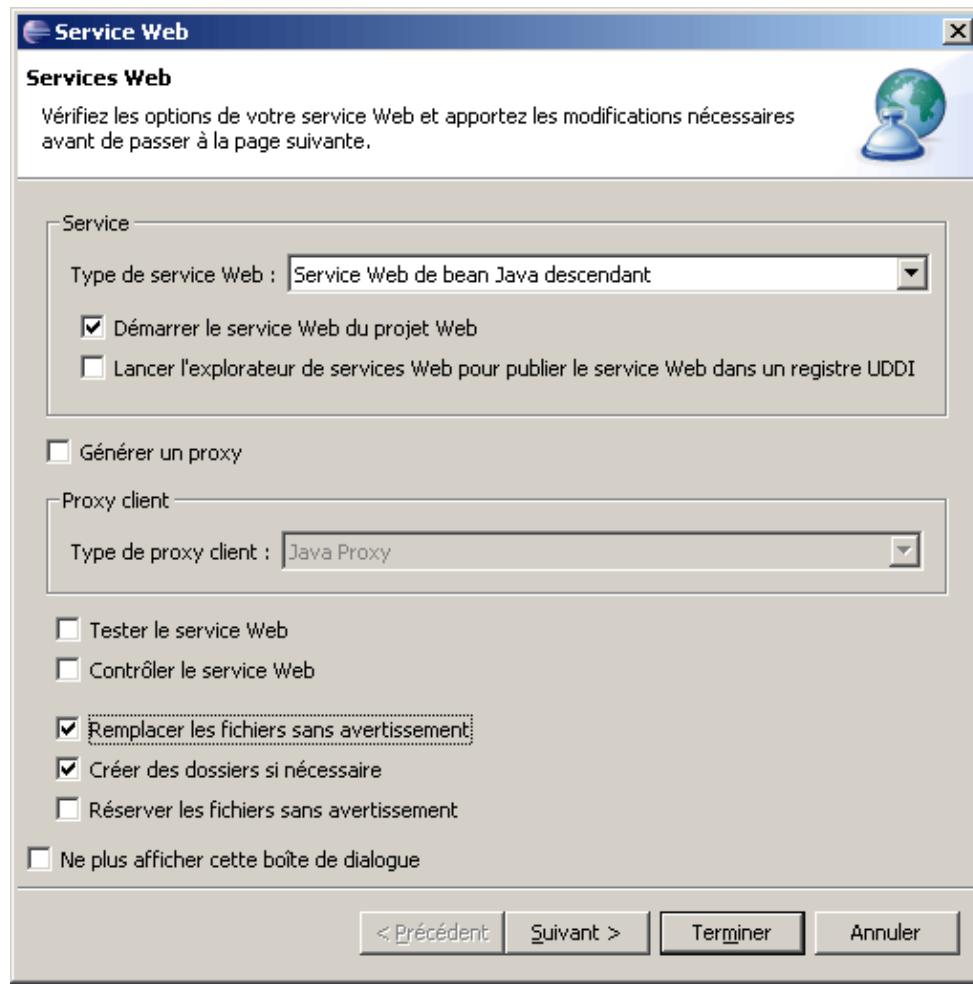


Cliquez sur le bouton « Suivant » pour permettre à l'assistant de générer les premières entités.

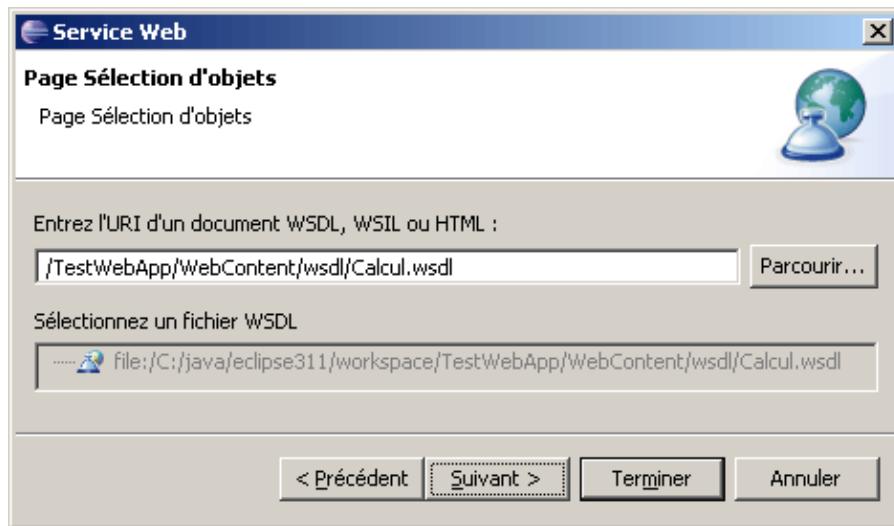


Cliquez sur le bouton « Terminer » pour générer les dernières entités, publier l'application sur le serveur et ouvrir la vue « Web Services Test Client »

L'option « Services web/Générer un squelette de bean Java » permet de générer une implémentation d'une description d'un service web à partir d'un fichier .wsdl



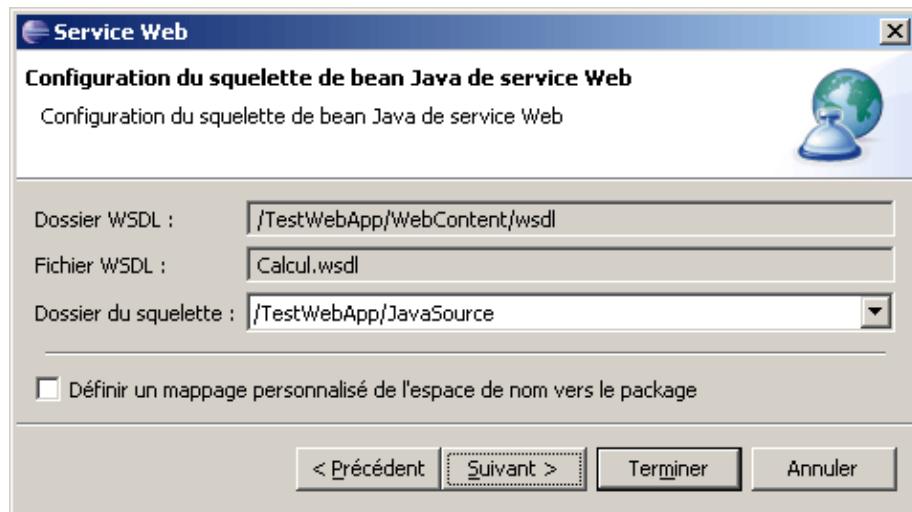
Cliquez sur le bouton « Suivant »



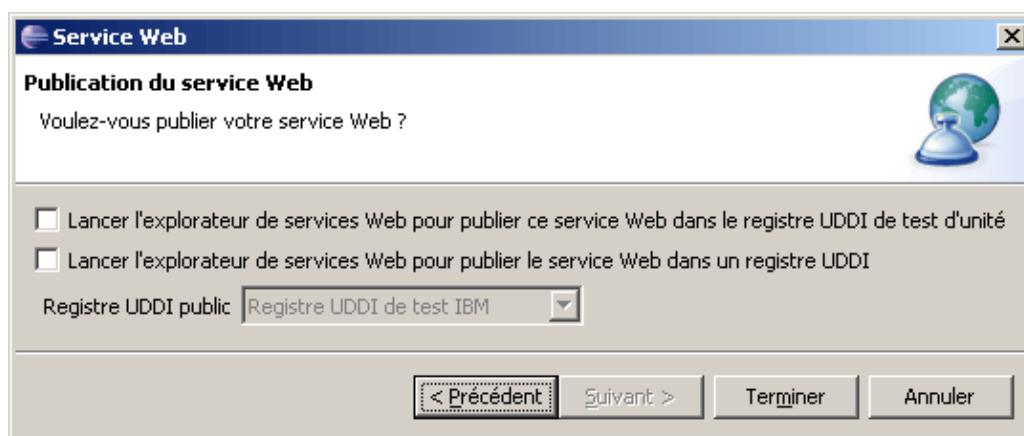
Cliquez sur le bouton « Suivant »



Cliquez sur le bouton « Suivant »



Cliquez sur le bouton « Suivant »



Cliquez sur le bouton « Terminer ». Le fichier CalculSoapBindingImpl.java est généré

Exemple :

```
/***
 * CalculSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis 1.2.1 Jun 14, 2005 (09:15:57 EDT) WSDL2Java emitter.
 */

package com.jmd.test.wtp;

public class CalculSoapBindingImpl implements com.jmd.test.wtp.Calcul{

    public long additionner(int a, int b) throws java.rmi.RemoteException {
        return -3;
    }

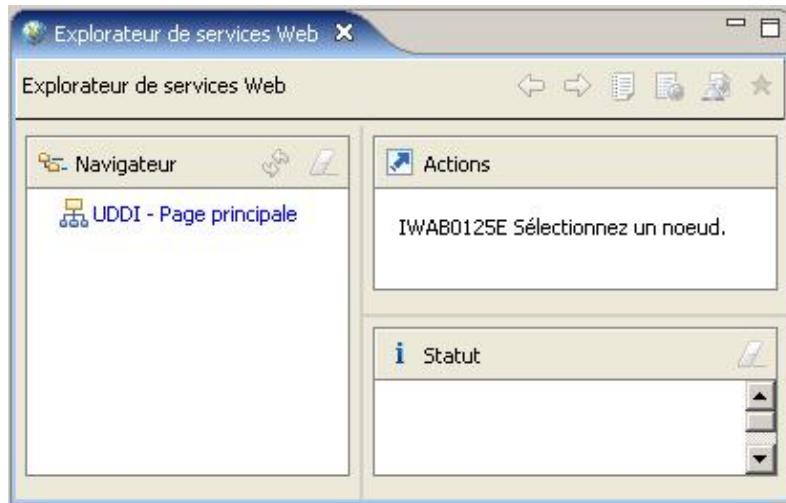
    public int soustraire(int a, int b) throws java.rmi.RemoteException {
        return -3;
    }

    public double diviser(int a, int b) throws java.rmi.RemoteException {
        return -3;
    }

    public long multiplier(int a, int b) throws java.rmi.RemoteException {
        return -3;
    }
}
```

### 26.3.3. L'explorateur de services web

Il est possible de lancer « l'Explorateur de services web » en utilisant l'option « Exécuter/Explorateur de services web » du menu principale

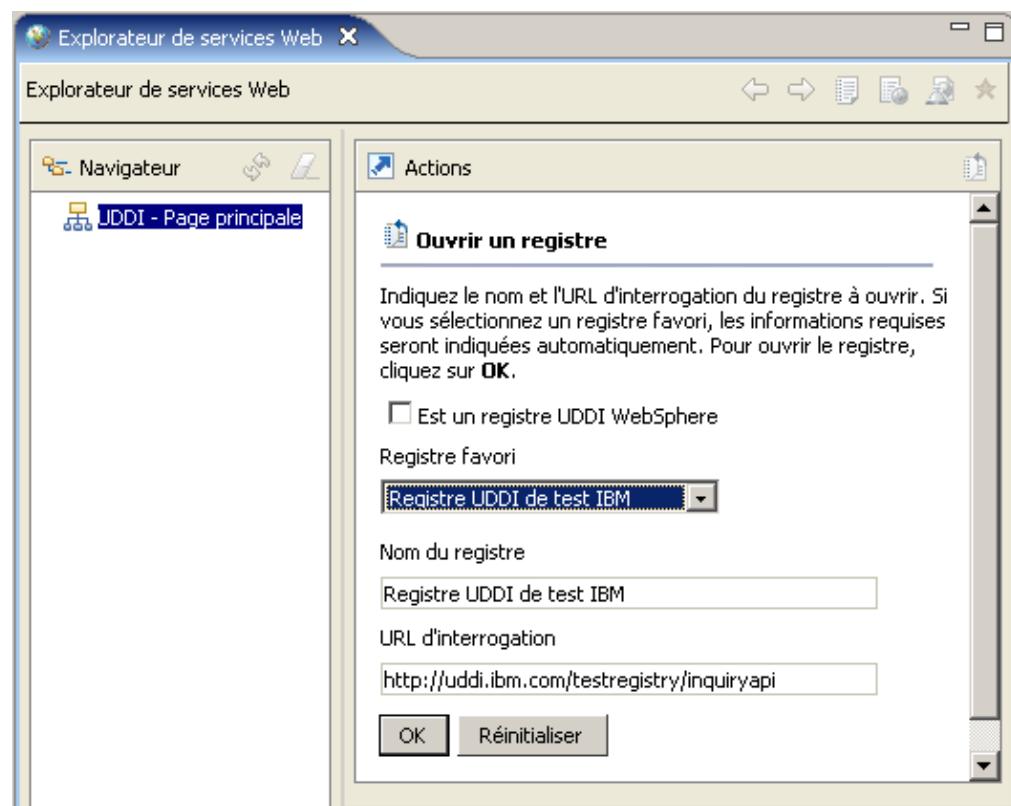


En cliquant sur « UDDI - Page principale », il est possible d'interroger un annuaireUDDI.

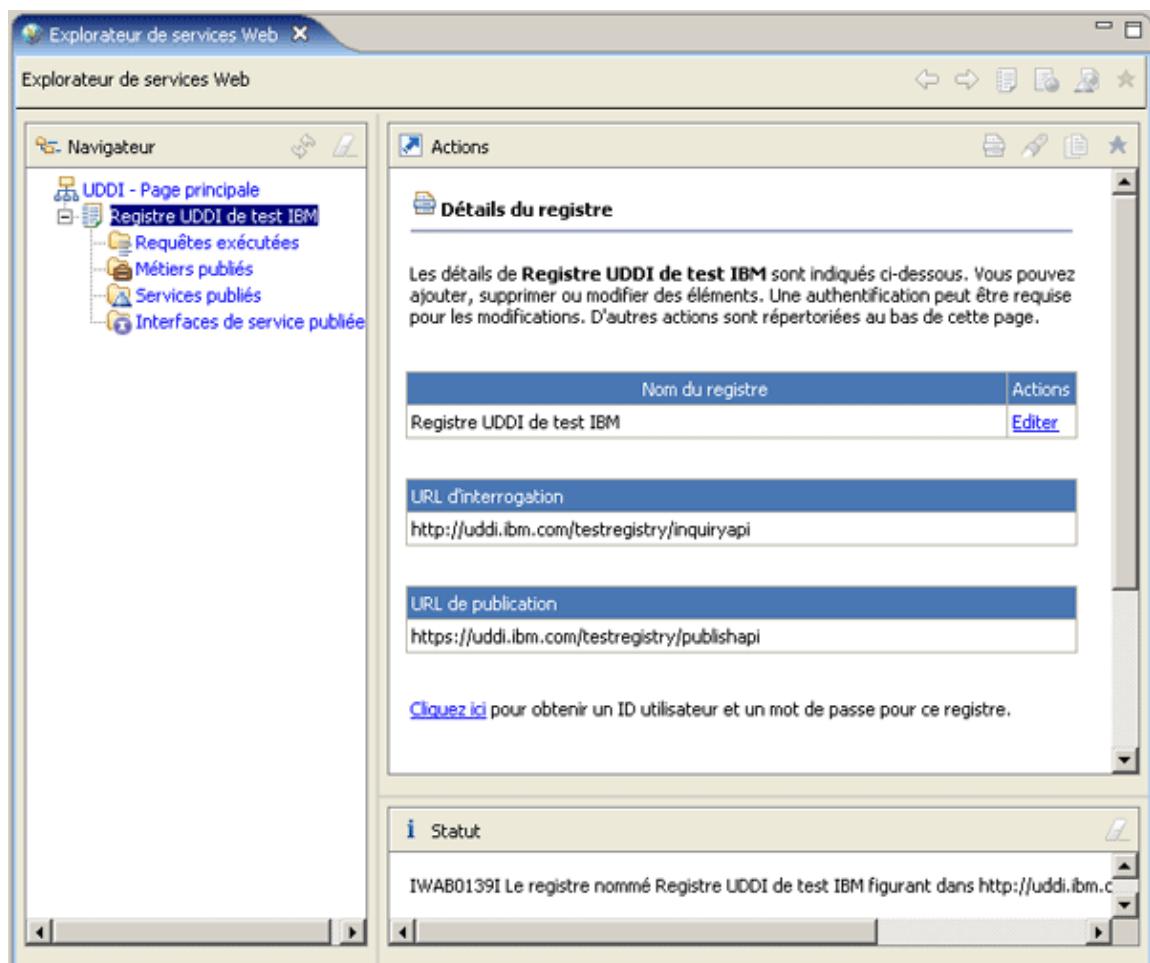
Cette vue permet d'explorer trois types d'entités liées au service web :

- un annuaire UDDI :
- un fichier .WSIL :

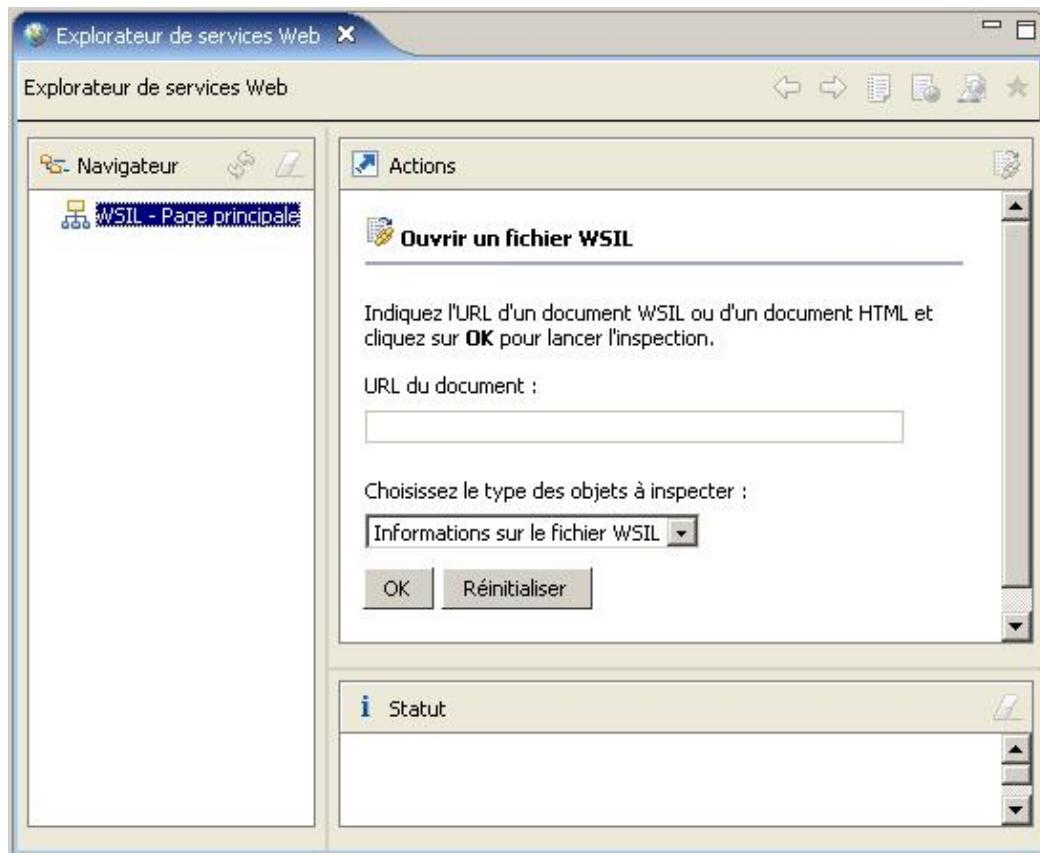
- un fichier .WSDL :



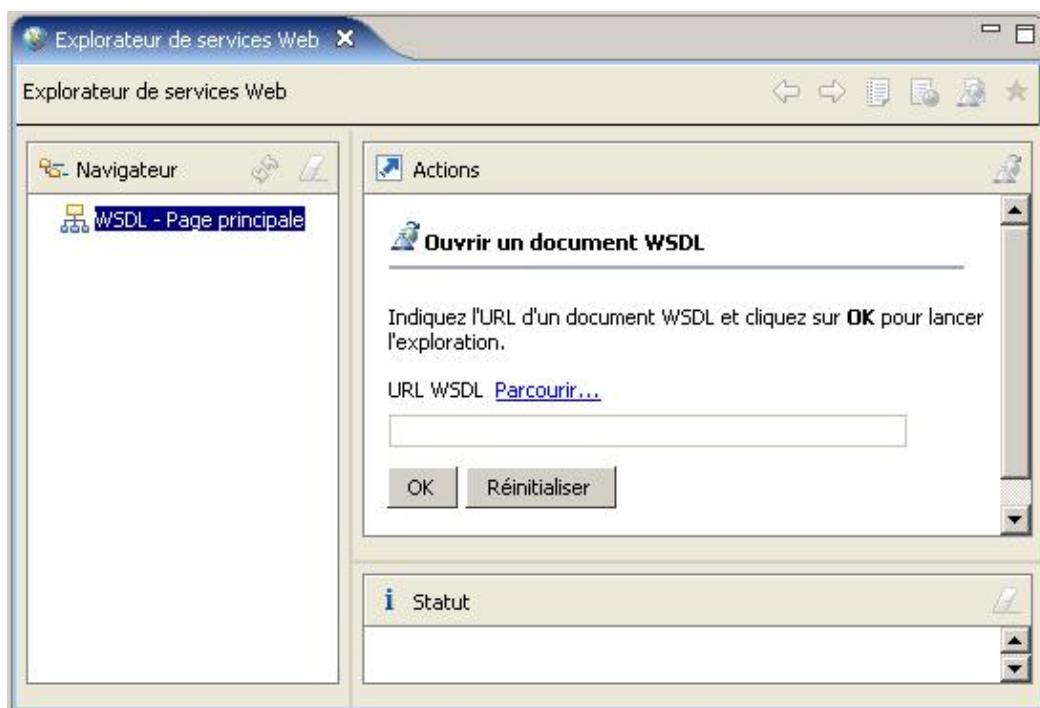
Sélectionner l'annuaire et cliquez sur le bouton « OK »



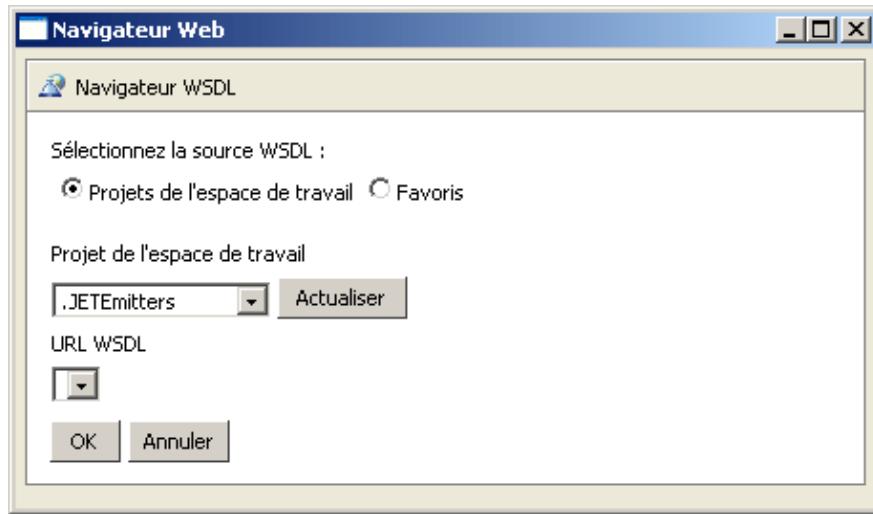
Cliquez sur « WSIL - Page principale »



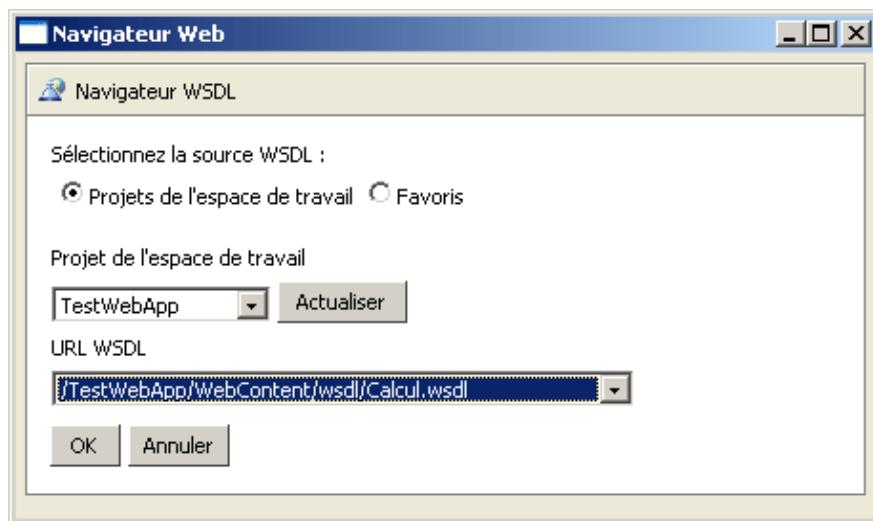
Cliquez sur « WSDL - Page principale »



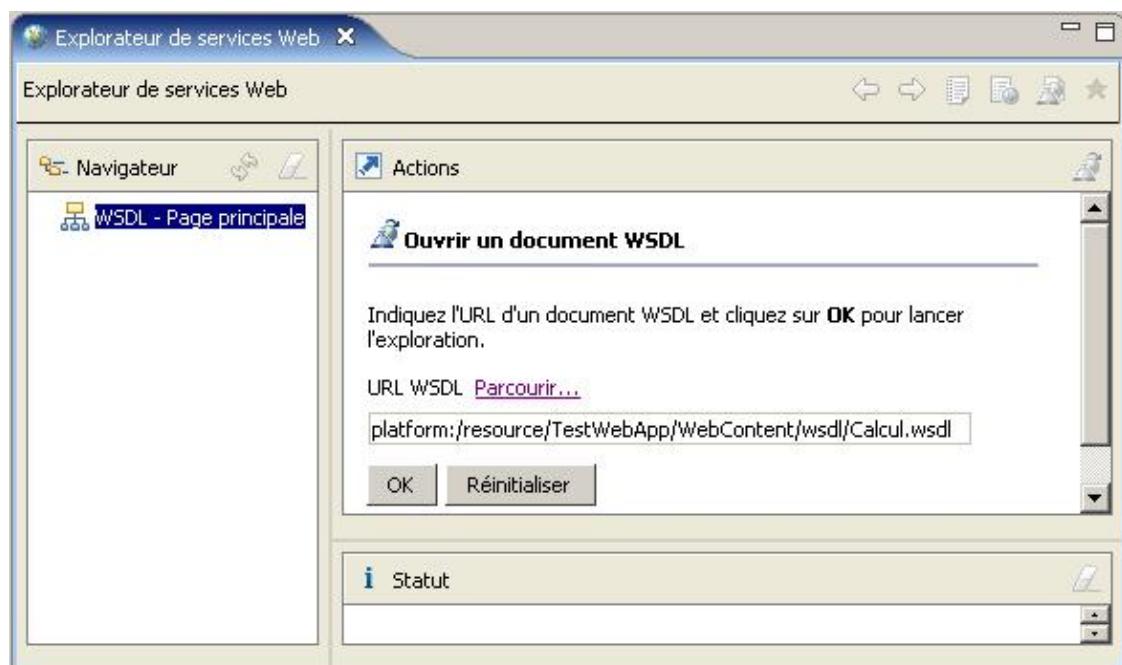
Il suffit de cliquer sur le lien « Parcourir » pour ouvrir une boîte de dialogue permettant de sélectionner le fichier .wsdl.



Il suffit de sélectionner le projet de l'espace de travail puis de sélectionner le fichier .wsdl de ce projet à utiliser



Cliquez sur le bouton « OK »



Cliquez sur le bouton « OK »

The screenshot shows the Eclipse Web Services Explorer interface. The left sidebar is titled "Navigateur" and lists "WSDL - Page principale" and "platform:/resource/TestWebApp". Under "TestWebApp", there is a "CalculService" entry with a "CalculSoapBinding" child. The main panel is titled "Actions" and contains a section "Informations sur la liaison WSDL". It states: "Les informations sur cet élément <liaison> **SOAP** figurent ci-après. Cliquez sur une opération pour indiquer ses paramètres et l'appeler ou spécifier d'autres noeuds finaux." Below this is a section "Opérations" with a table:

Nom	Documentation
<a href="#">multiplier</a>	--
<a href="#">soustraire</a>	--
<a href="#">diviser</a>	--
<a href="#">additionner</a>	--

At the bottom of the main panel is a "Statut" section with the message: "IWAB0381I platform:/resource/TestWebApp/WebContent/wsdl/Calcul.wsdl a été ouvert."

Il est aussi possible d'appeler un service web externe. Par exemple :  
<http://www.dataaccess.com/webservicesserver/conversions.wso?WSDL>

The screenshot shows the Eclipse Web Services Explorer interface with a dialog box open. The dialog is titled "Actions" and "Ouvrir un document WSDL". It instructs the user to "Indiquez l'URL d'un document WSDL et cliquez sur OK pour lancer l'exploration." Below this is a "URL WSDL" field containing the URL: "http://www.dataaccess.com/webservicesserver/conversions.wso?WSDL". At the bottom of the dialog are two buttons: "OK" and "Réinitialiser". The status bar at the bottom of the interface shows the message: "IWAB0381I platform:/resource/TestWebApp/WebContent/wsdl/Calcul.wsdl a été ouvert."

Saisissez l'url du fichier .wsdl et cliquez sur le bouton « OK »

The screenshot shows the Eclipse Web Services Explorer interface. The left pane, titled "Navigateur", lists the WSDL interface structure:

- WSDL - Page principale
- platform:/resource/TestWebApp
- http://www.dataaccess.com/w
- Conversions
  - ConversionsSoapBinding
    - NumberToDollars
    - TitleCaseWords
    - NumberToWords

The right pane, titled "Actions", displays "Informations sur la liaison WSDL". It contains a table of operations:

Nom	Documentation
<a href="#">NumberToDollars</a>	Returns the non-zero dollar amount of the passed number.
<a href="#">TitleCaseWords</a>	Returns the passed text title cased and with the passed token between each word.
<a href="#">NumberToWords</a>	Returns the word corresponding to the positive number passed as parameter. Limited to quadrillions.

A status bar at the bottom indicates: IWAB0381I http://www.dataaccess.com/webservicesserver/conversions.wsdl a été

Cliquez sur le lien " NumberToWord ".

The screenshot shows the Eclipse Web Services Explorer interface. The left pane, titled "Navigateur", lists the WSDL interface structure, identical to the previous screenshot.

The right pane, titled "Actions", displays "Appeler une opération WSDL". It shows the URL for the operation and a parameter entry field:

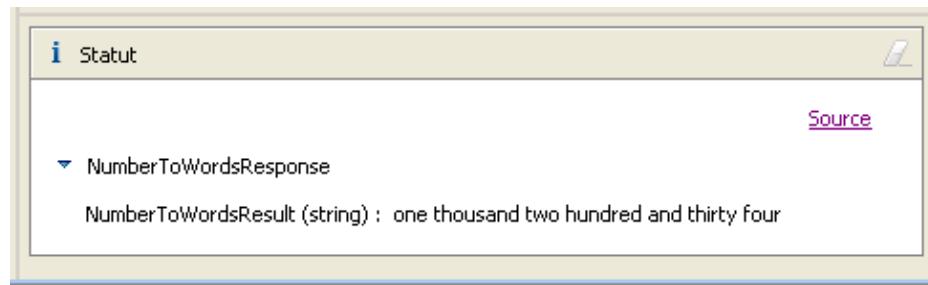
Noeuds finaux  
http://www.dataaccess.com/webservicesserver/conversions.wsdl

NumberToWords  
ubNum unsignedLong  
1234

OK Réinitialiser

A status bar at the bottom indicates: IWAB0381I http://www.dataaccess.com/webservicesserver/conversions.wsdl a été

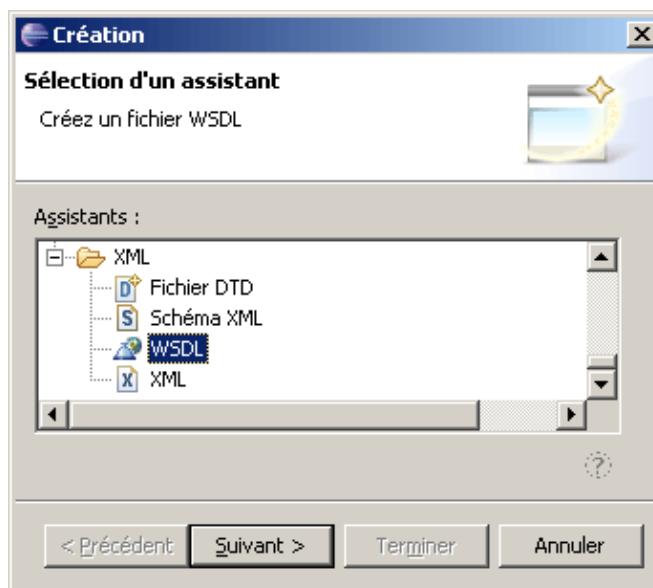
Saisissez un nombre et cliquez sur le bouton « OK »



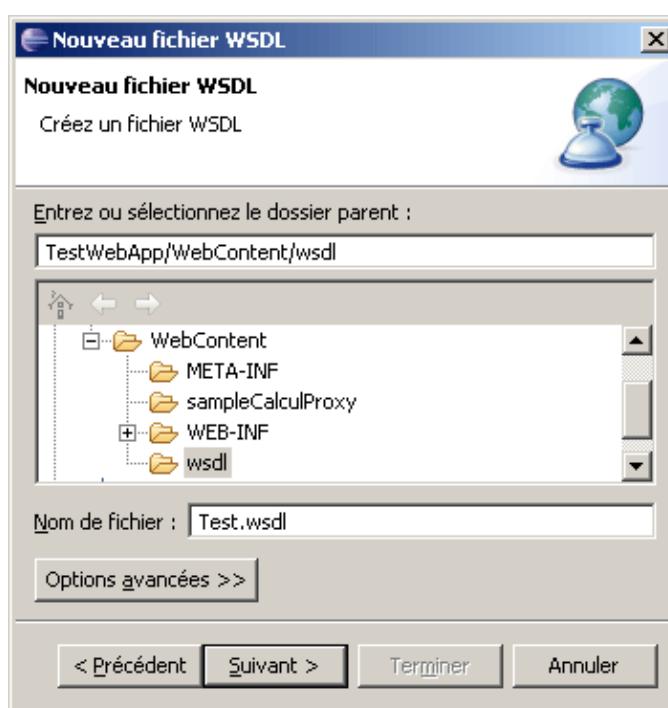
Le service web est appelé avec les paramètres fournis et le résultat est affiché dans la partie « Statut ».

#### 26.3.4. L'éditeur de fichier .wsdl

Il faut créer une nouvelle entité de type XML/WSDL



Cliquez sur le bouton « Suivant »



Saisissez le nom du fichier .wsdl et cliquez sur le bouton « Suivant »



## 26.4. La mise en oeuvre d'Axis avec le WTP 1.5

	Version utilisée dans cette section
Eclipse	3.2.1
WTP	1.5.1
JDK	1.5_07
Axis	1.4
Tomcat	5.0

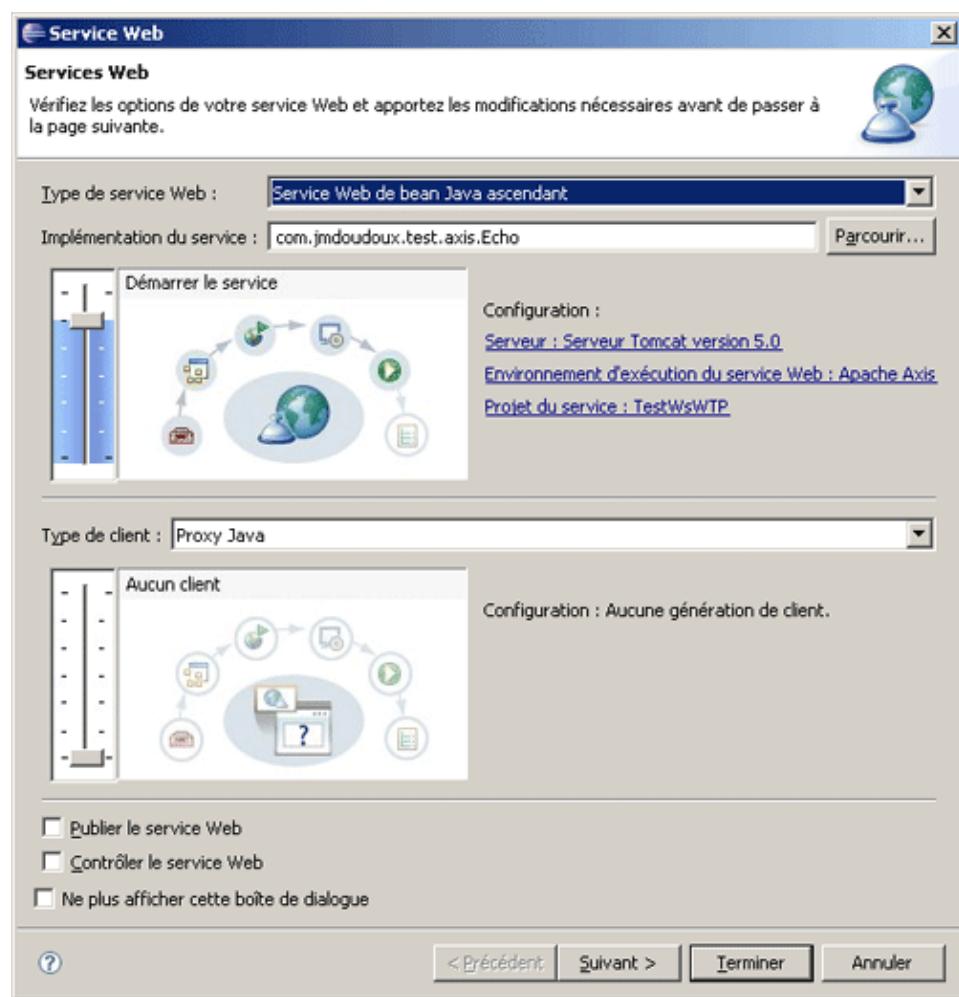
Cette section va utiliser un projet de type web dynamique nommé TestWsWTP dans lequel la cible de compilation a été modifiée pour pointer sur le répertoire WebContent/WEB-INF/classes créé pour l'occasion.

### 26.4.1. La création du service web

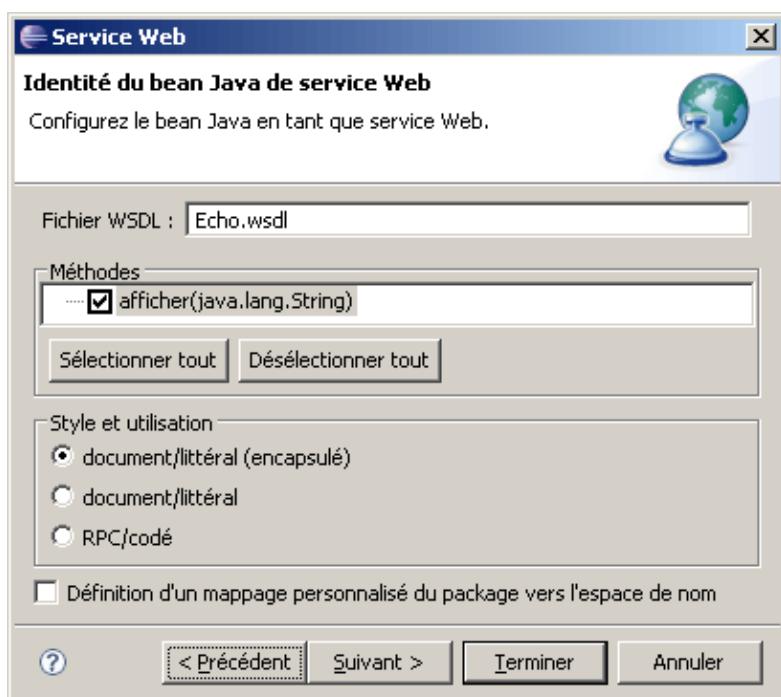
Créez une classe nommée Echo

Exemple :
<pre>package com.jmdoudoux.test.axis;  public class Echo {     public String afficher(String message) {         return message;     } }</pre>

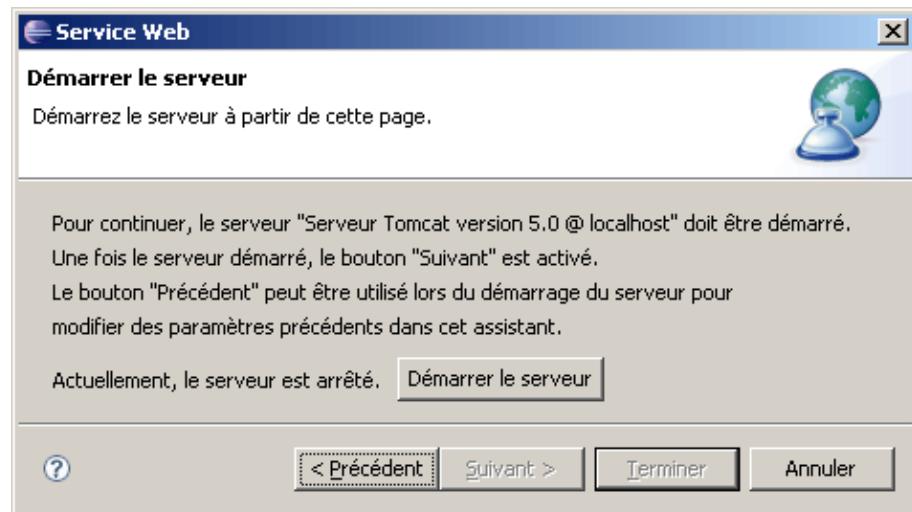
Dans la vue « Explorateur de projets », sélectionnez la classe Echo et utilisez l'option « Service web / Créer un service web » du menu contextuel.



Il est possible de préciser le niveau de fonctionnalités pris en compte par l'assistant grâce au curseur de gauche. Cliquez sur le bouton « Suivant ».



La boîte de dialogue suivante permet de préciser les options pour générer le fichier wsdl notamment en précisant son nom, les méthodes de la classe qui seront proposées dans le service web et le type de service web. Cliquez sur le bouton « Suivant »



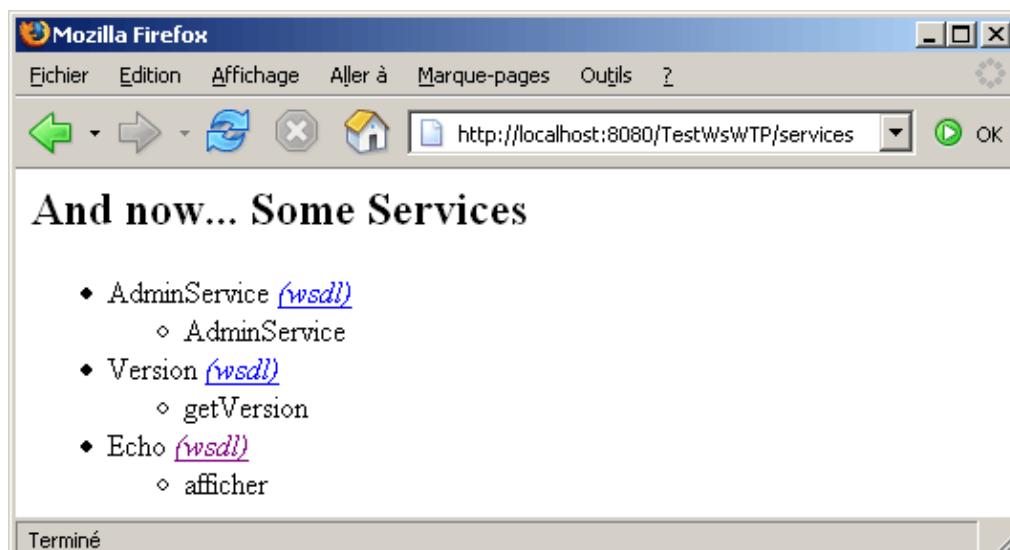
La page suivante permet de démarrer le serveur : cliquez sur le bouton « Démarrer le serveur ». Le serveur démarre : cliquez sur le bouton « Suivant ».



La page suivante permet de demander l'enregistrement du service dans un annuaire UDDI.

Cliquez sur le bouton « Terminer ».

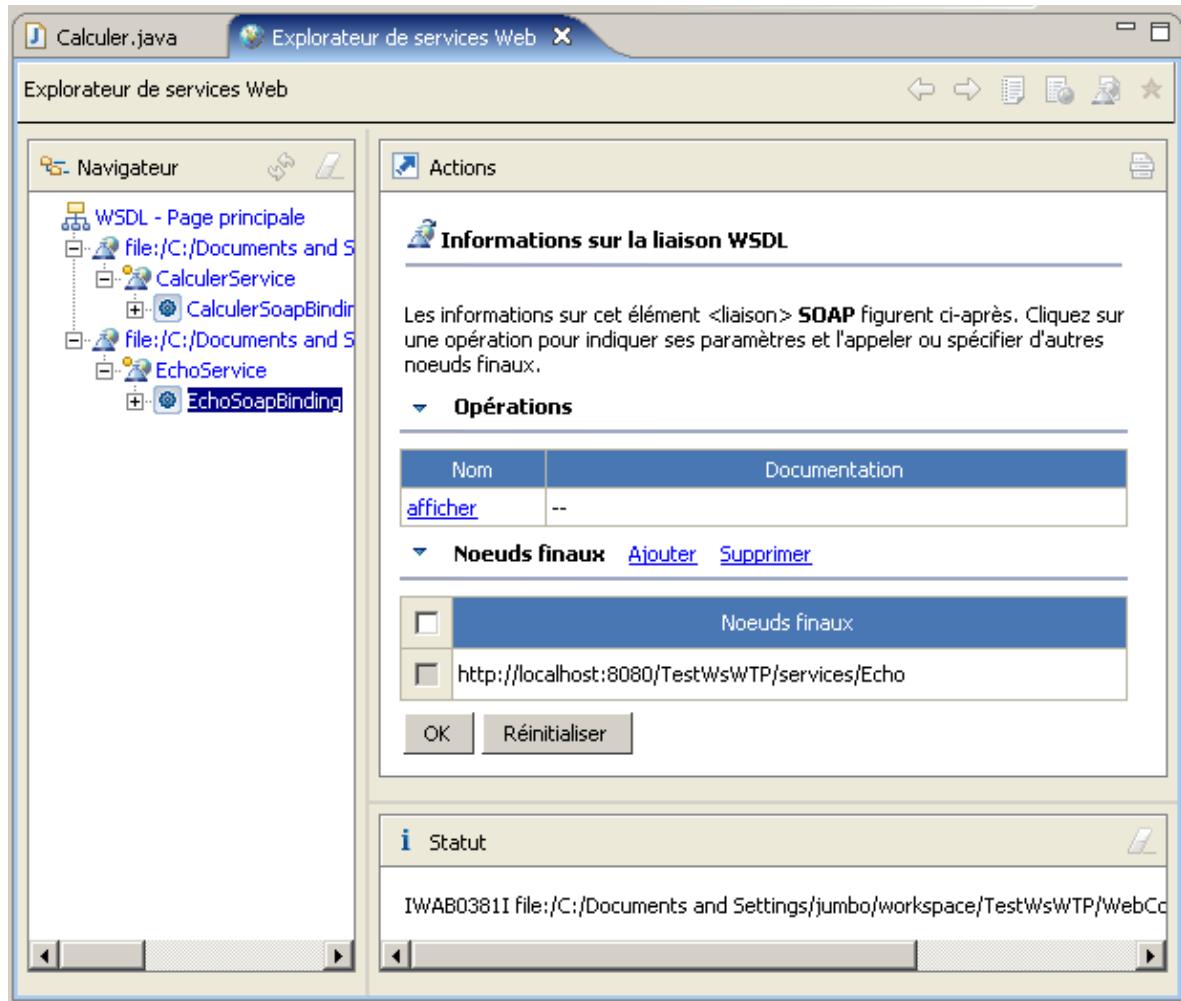
Ouvrez un navigateur et saisissez l'url <http://localhost:8080/TestWsWTP/services>



## 26.4.2. Le test du service web avec l'explorateur de services web

Le WTP propose un outil qui permet de tester un service web.

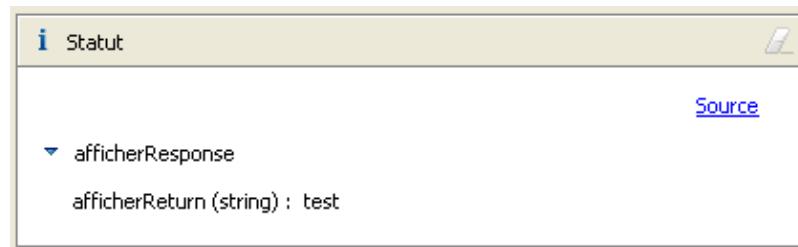
Dans l'explorateur de projets, sélectionner le fichier wsdl correspondant au service web puis utilisez l'option « Service Web / Tester avec un explorateur de services web ».



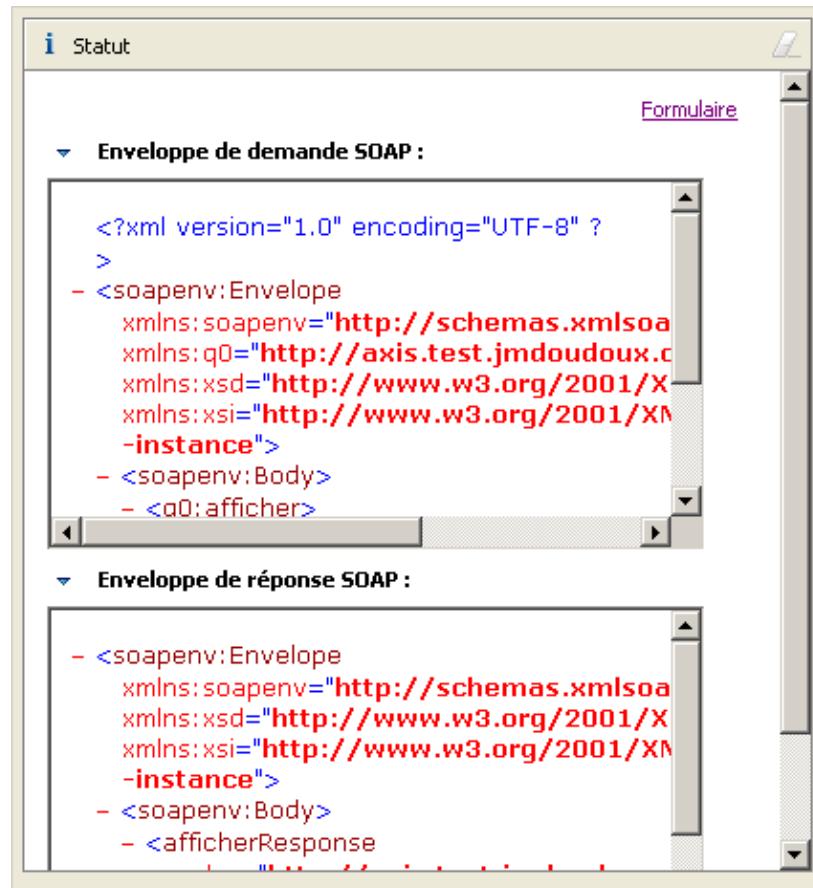
Cliquez sur le lien « afficher » dans la liste des opérations.



Saisissez test comme valeur du paramètre message de type string et cliquez sur le bouton « OK ».

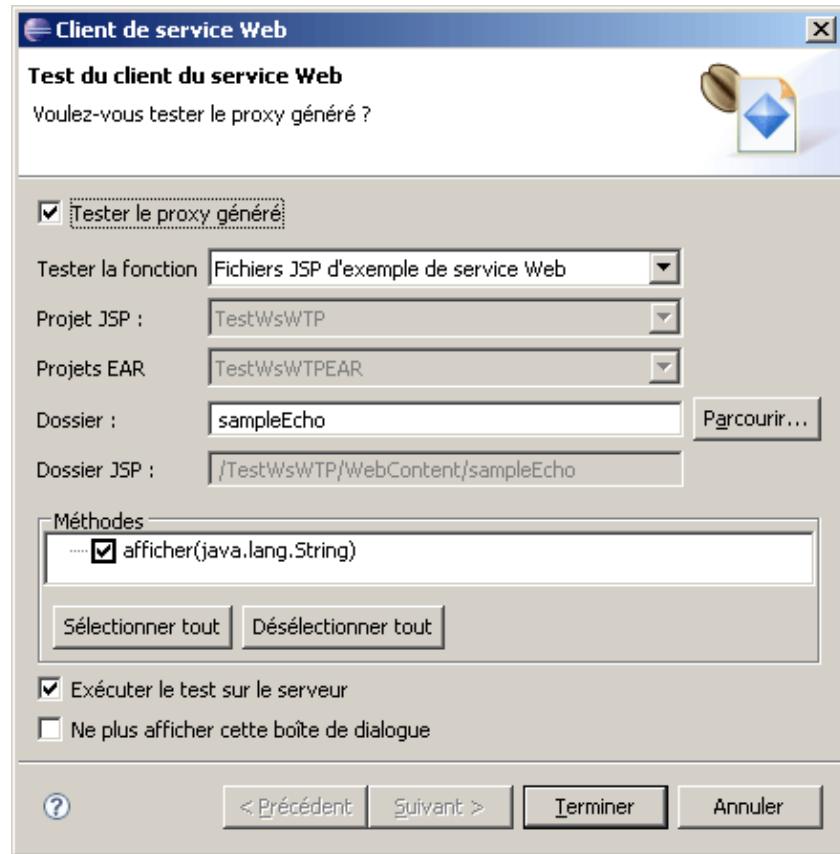


Cliquez sur le lien source pour voir la requête et la réponse SOAP

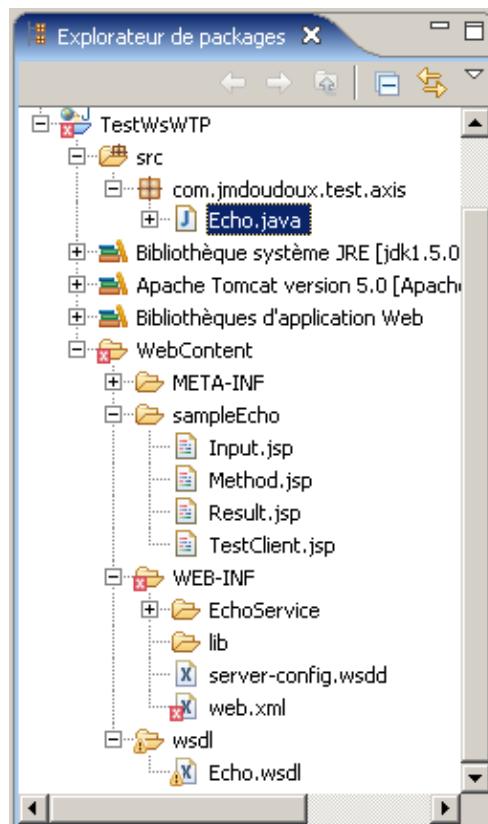


### 26.4.3. La création d'un client de test

Dans l'explorateur de packages, sélectionnez la classe qui implémente le service web et utilisez l'option « Services web / Générer des fichiers JSP d'exemples » du menu contextuel.

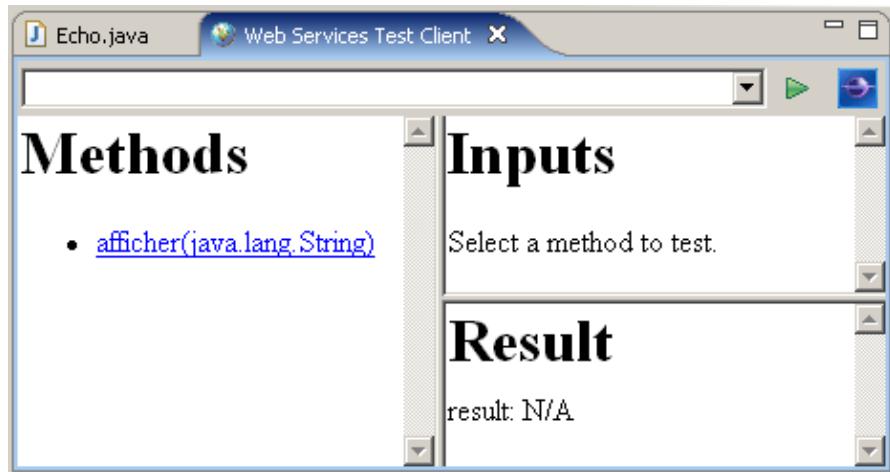


Cliquez sur le bouton « Terminer » pour générer les fichiers



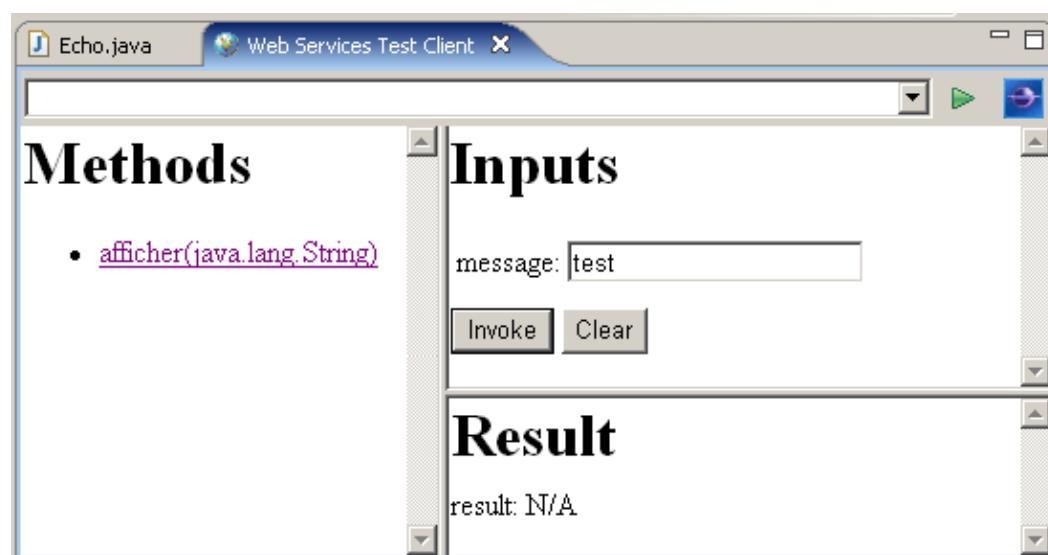
Les pages JSP de test sont générées dans un répertoire nommé « sample » suivi du nom de la classe

L'assistant lance le navigateur interne pour afficher la page principale générée.

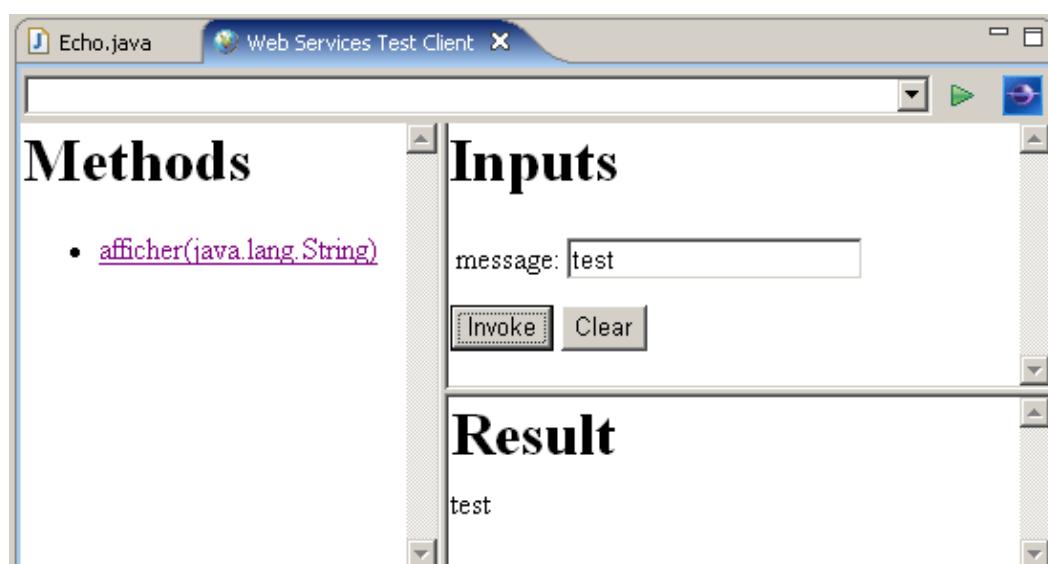


L'url de cette page est <http://localhost:8080/TestWsWTP/sampleEcho/TestClient.jsp>

Cliquez sur le lien de la méthode afficher() :



Saisissez une valeur pour la propriété message de la méthode et cliquez sur le bouton « Invoke »



La méthode du service web est invoquée et le résultat est affiché dans le cadre Result.



Cette section sera développée dans une version future de ce document

# Chapitre 27

### 27.1. Dali

Dali est un plug-in proposant de fournir des outils pour faciliter le mapping objet/relationnel.

Dali est un sous projet du projet WTP dont le but est de faciliter la mise en oeuvre de l'API JPA (Java Persistence API) définie par la JSR 220.

La page officiel du projet est l'url : <http://www.eclipse.org/dali>.

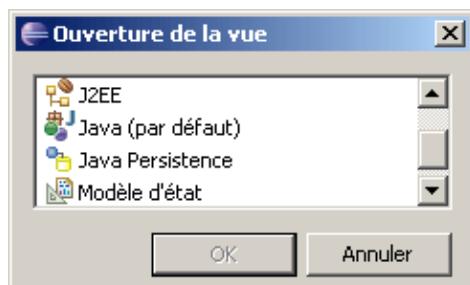
Les versions utilisées dans cette section sont :

Eclipse	3.2.1
WTP	1.5.1
Dali	0.5
MySQL	4.1
API JPA	Implémentation de référence (toplink-essentials.jar) fournie avec GlassFish

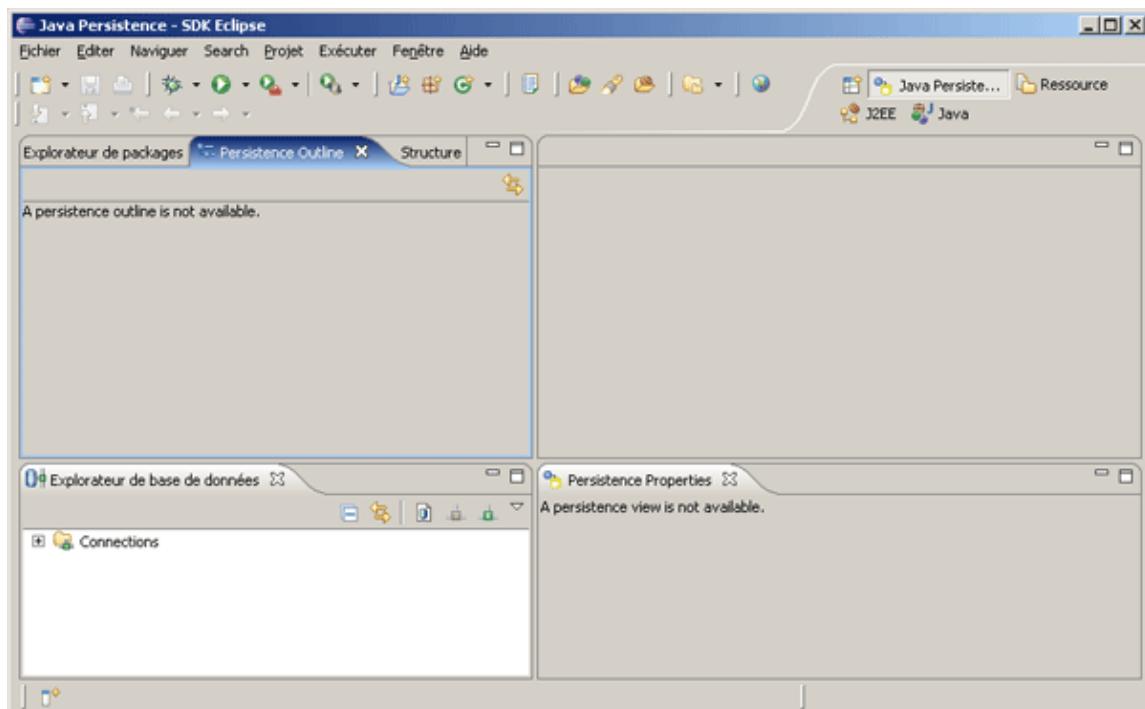
La version utilisée du plug-in dans cette section est en cours développement mais elle propose déjà des fonctionnalités intéressantes.

#### 27.1.1. La création et la configuration d'un nouveau projet

Il faut ouvrir la perspective Java Perspective : sélectionnez l'option « Fenêtre / ouverture la perspective » du menu principal.



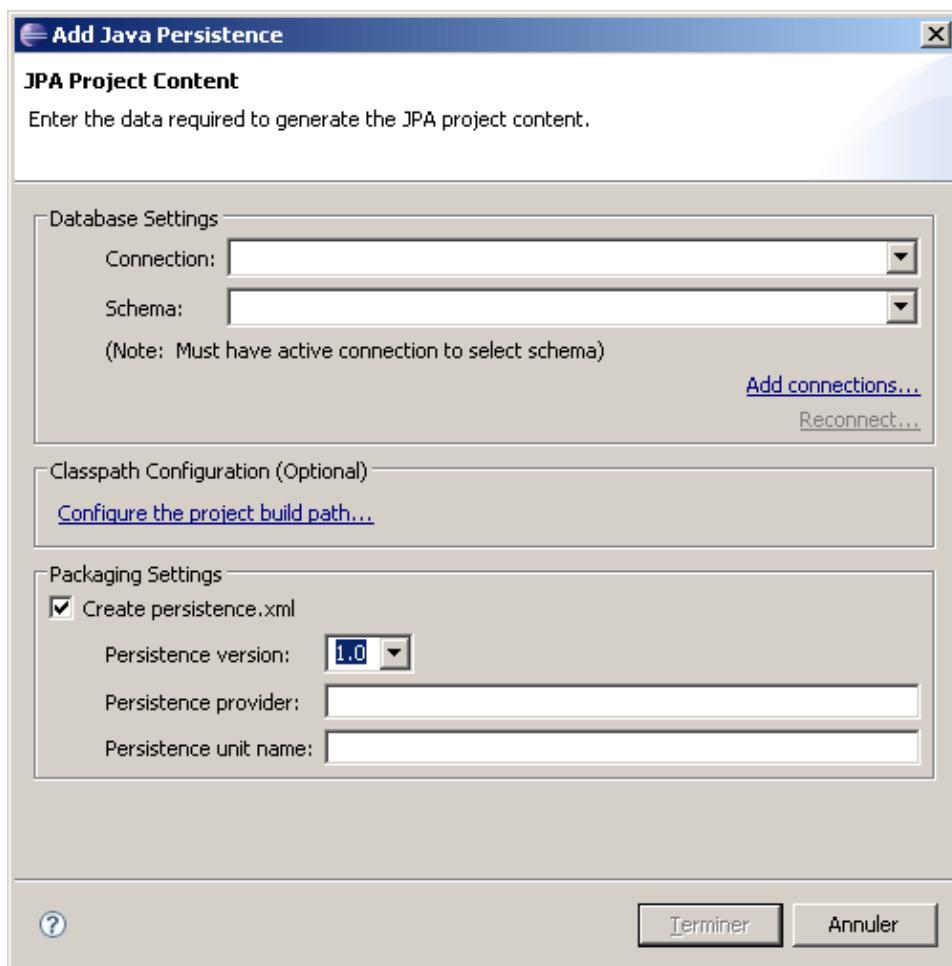
Sélectionnez « Java Persistence » et cliquez sur le bouton « OK » pour ouvrir la perspective.



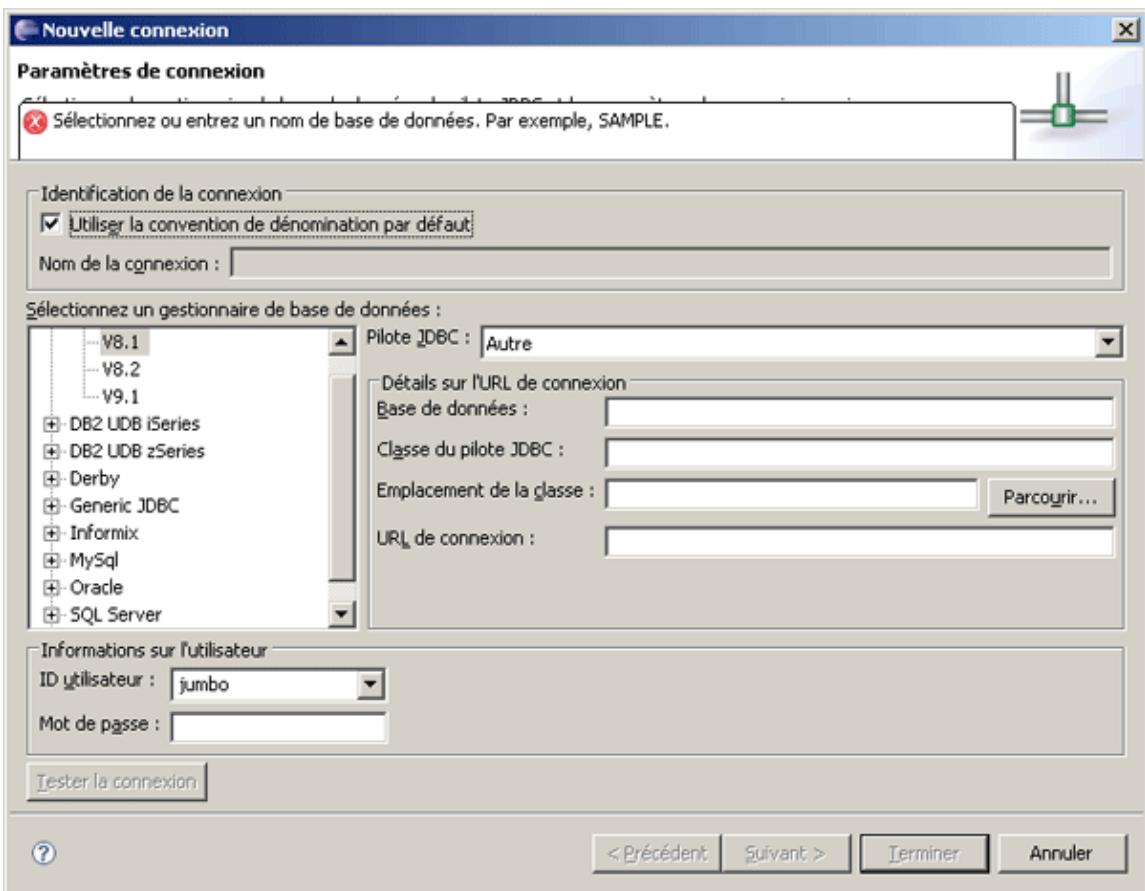
Créez un nouveau projet de type Java associé à un JRE 1.5 minimum.

Il faut ajouter le support de l'API Java Persistence au projet.

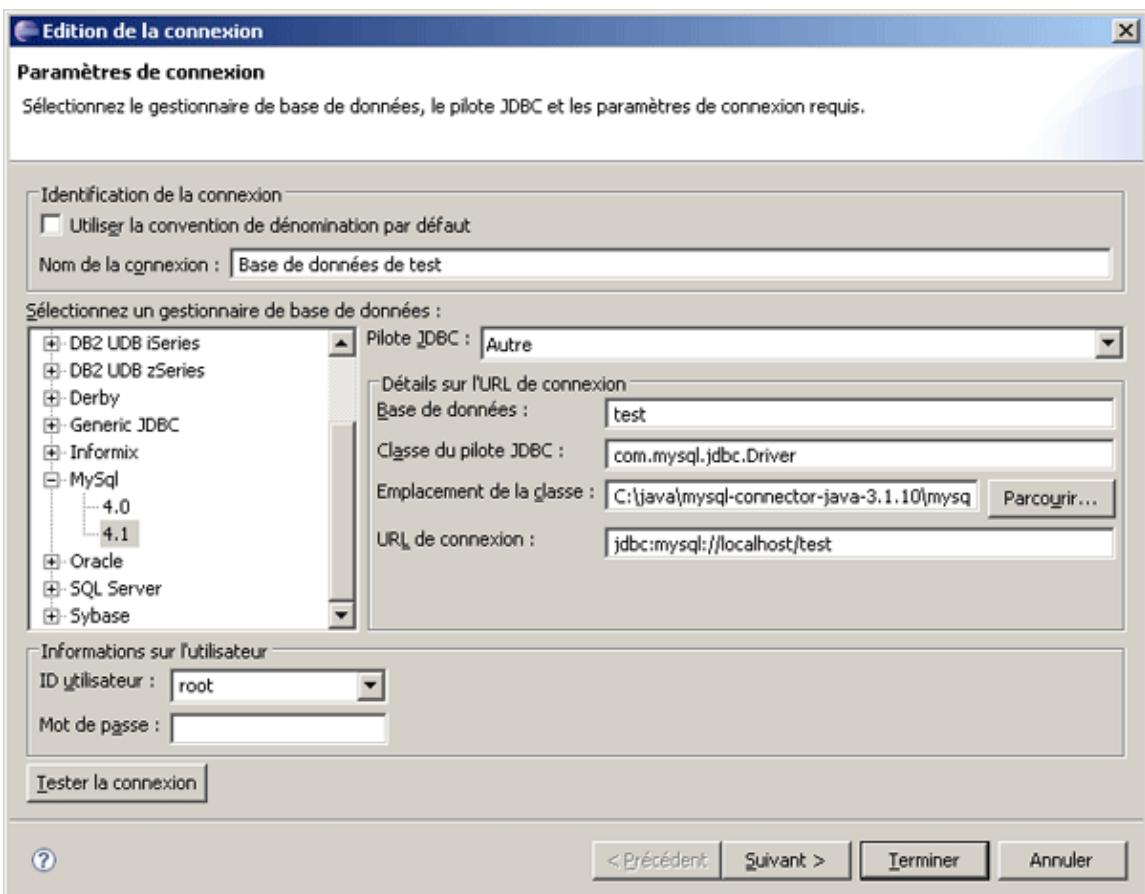
Sélectionnez le projet dans l'explorateur de projet et utiliser l'option Java Persistence / Add Java Persistence » : une boîte de dialogue permet de saisir les informations requises.



Cliquez sur le lien « Add connections »

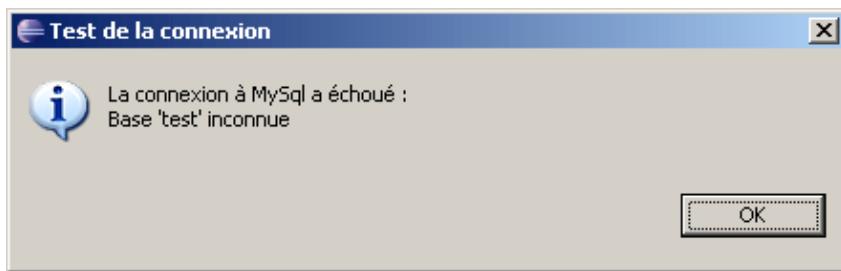


Saisissez les informations concernant la base de données et sa connexion.

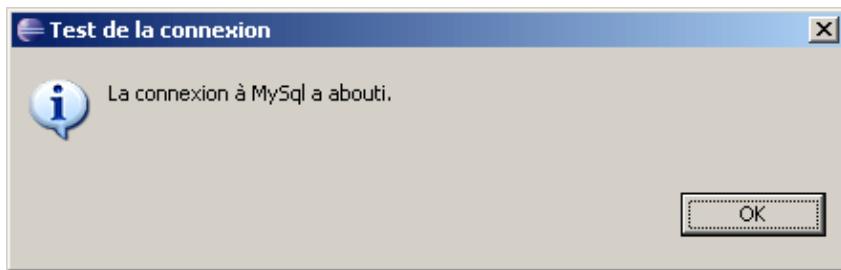


Cliquez sur le bouton « Tester la connexion »

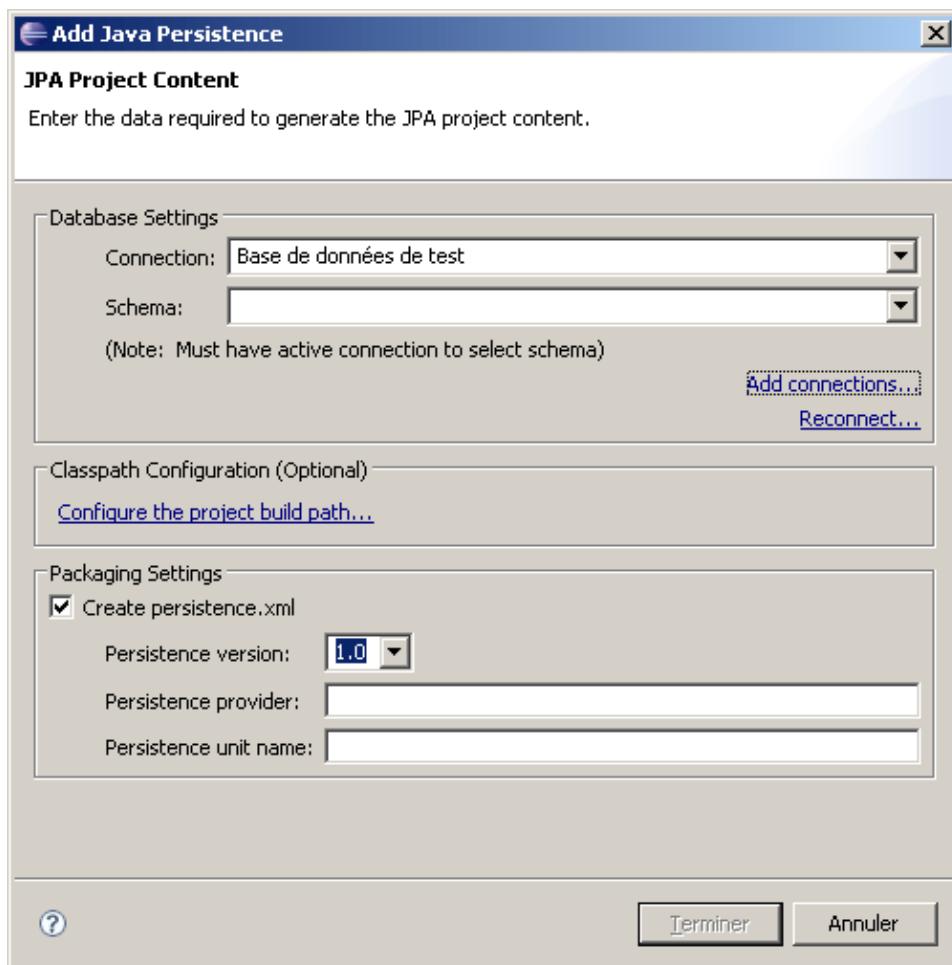
Si la base de données n'est pas trouvée, un message est affiché :



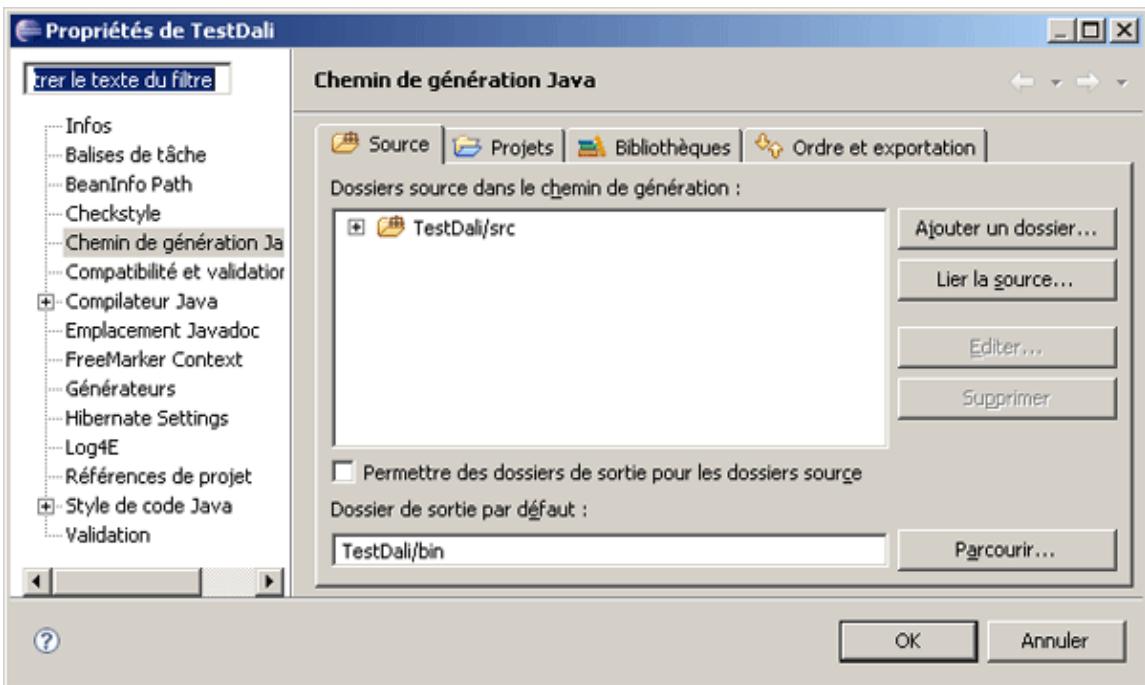
Si la base de données est trouvée, un message d'information est affiché



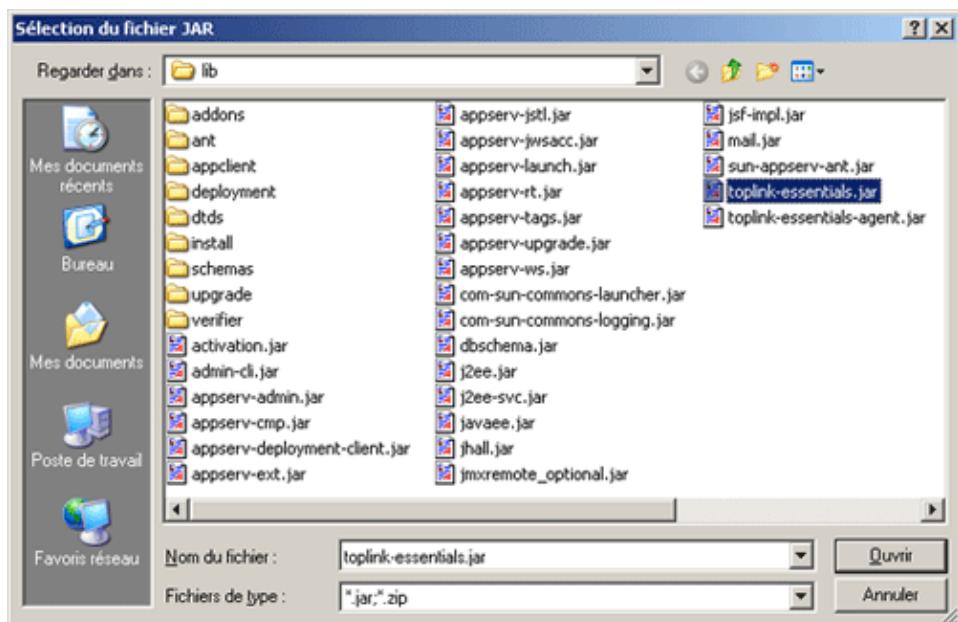
Cliquez sur le bouton « OK » puis sur le bouton « Terminer »



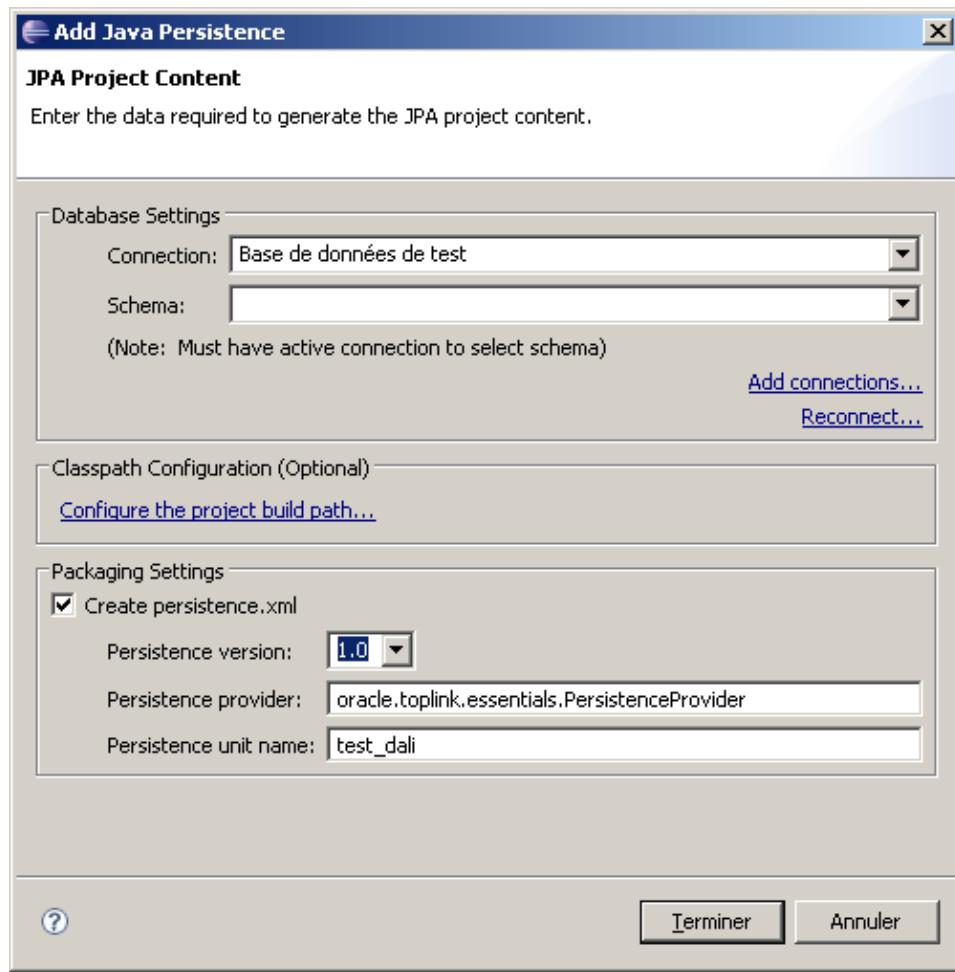
Cliquez sur le bouton « Configure the project build path »



Cliquez sur l'onglet « Bibliothèques », puis sur le bouton « Ajouter des fichiers jar externes »



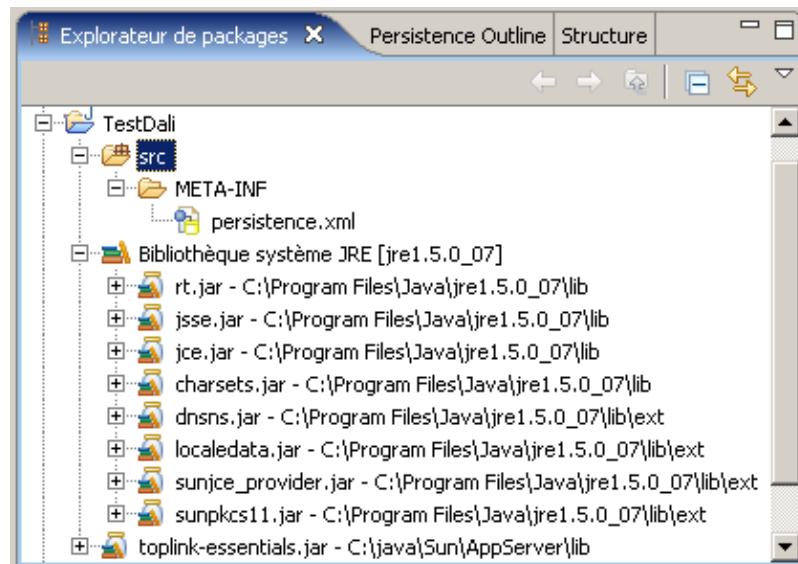
Sélectionnez le fichier toplink-essentials.jar, puis cliquez sur le bouton « Ouvrir » et sur le bouton « OK ».



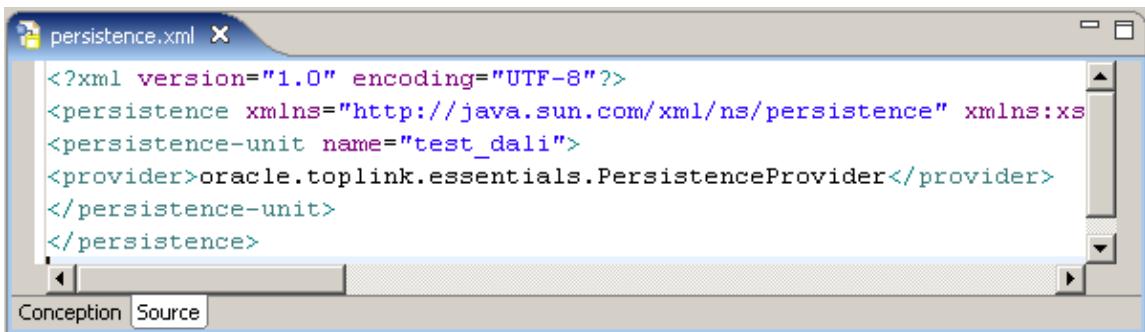
Sélectionnez le schéma de la base de données (dans le cas de MySQL, c'est la base de données elle-même)

Saisissez le nom de la classe du Provider de Persistence : oracle.toplink.essentials.PersistenceProvider et le nom de l'unité de persistence puis cliquez sur le bouton « Terminer ».

Le fichier persistence.xml est créé dans le répertoire META-INF



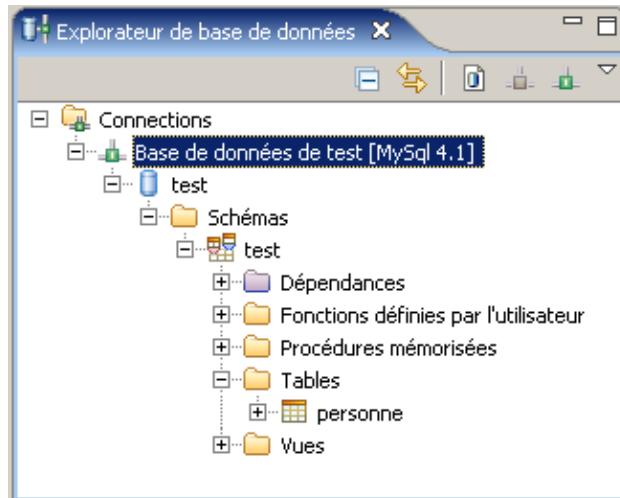
Ce fichier xml est le fichier de configuration de l'API JPA.



The screenshot shows the Eclipse IDE interface with a code editor window titled "persistence.xml". The XML content defines a persistence unit named "test\_dali" using the "oracle.toplink.essentials.PersistenceProvider". Below the editor are two tabs: "Conception" and "Source".

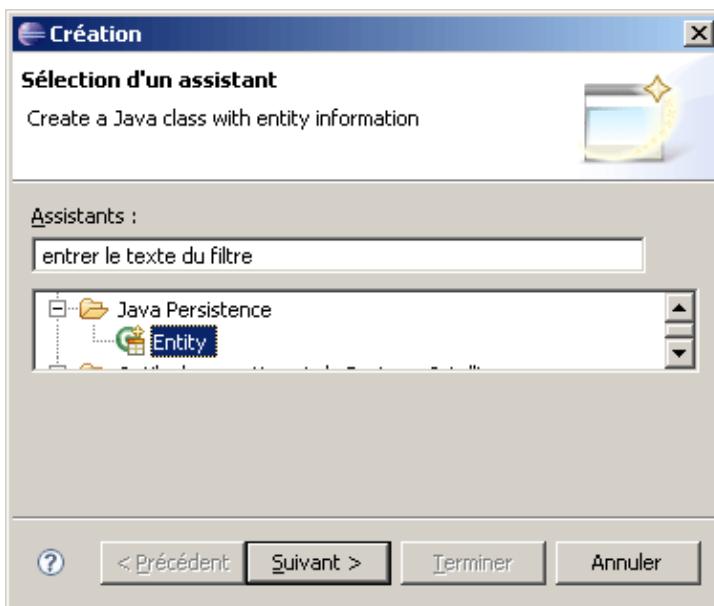
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xs
<persistence-unit name="test_dali">
<provider>oracle.toplink.essentials.PersistenceProvider</provider>
</persistence-unit>
</persistence>
```

La connexion à la base de données est affichée dans la vue « Explorateur de base de données ».

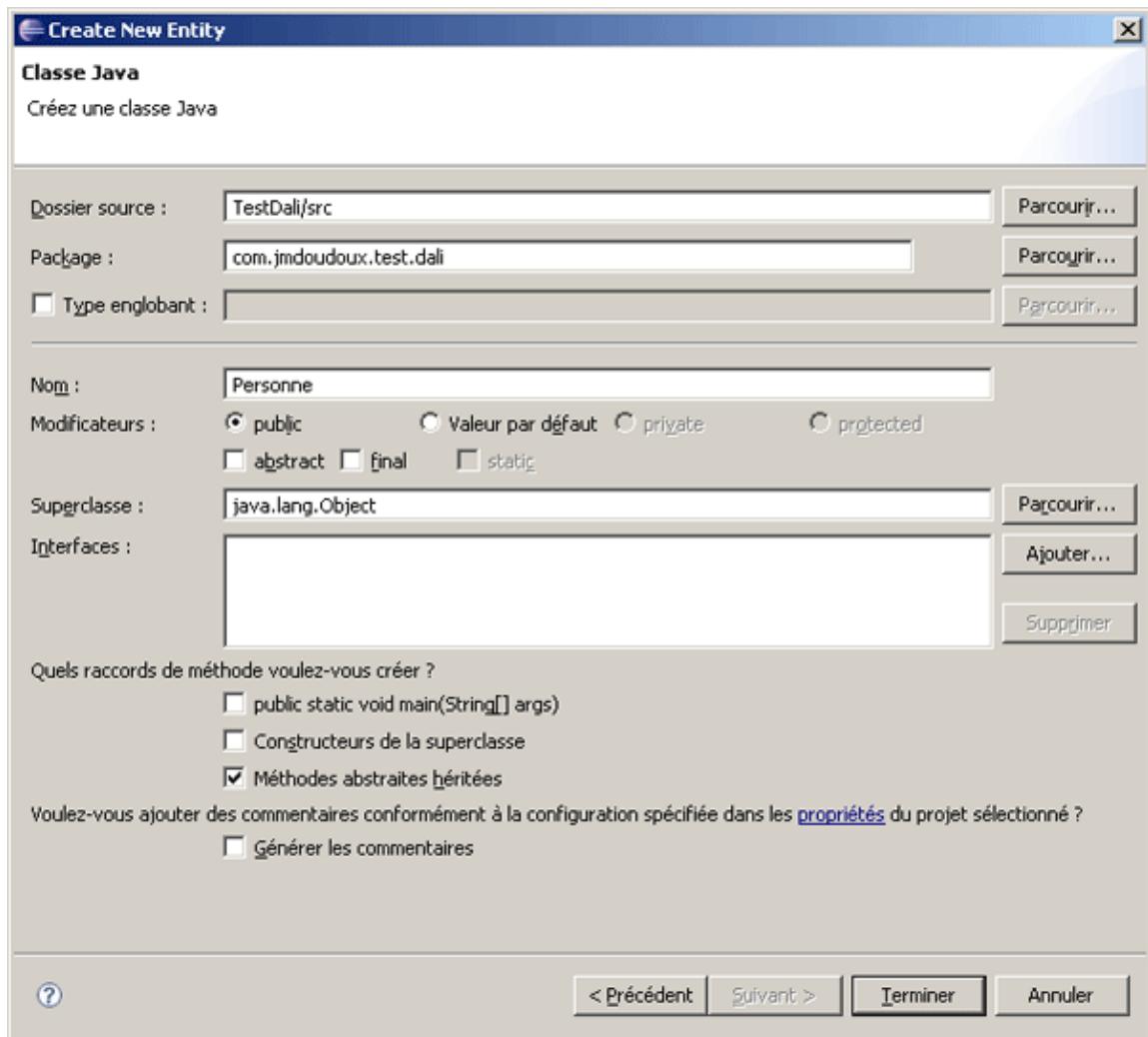


### 27.1.2. Ajouter une entité persistante

Créez une nouvelle entité de type « Java Persistence / Entity »



Cliquez sur le bouton « Suivant »



Saisissez le nom du package et le nom de la classe puis cliquez sur le bouton « Terminer ».

La classe est générée et son contenu est ouvert dans un éditeur.

```

package com.jmdoudoux.test.dali;

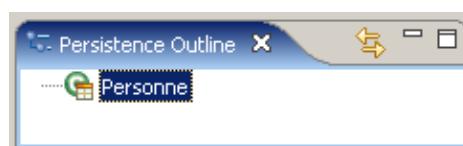
import javax.persistence.Entity;

@Entity
public class Personne {
}

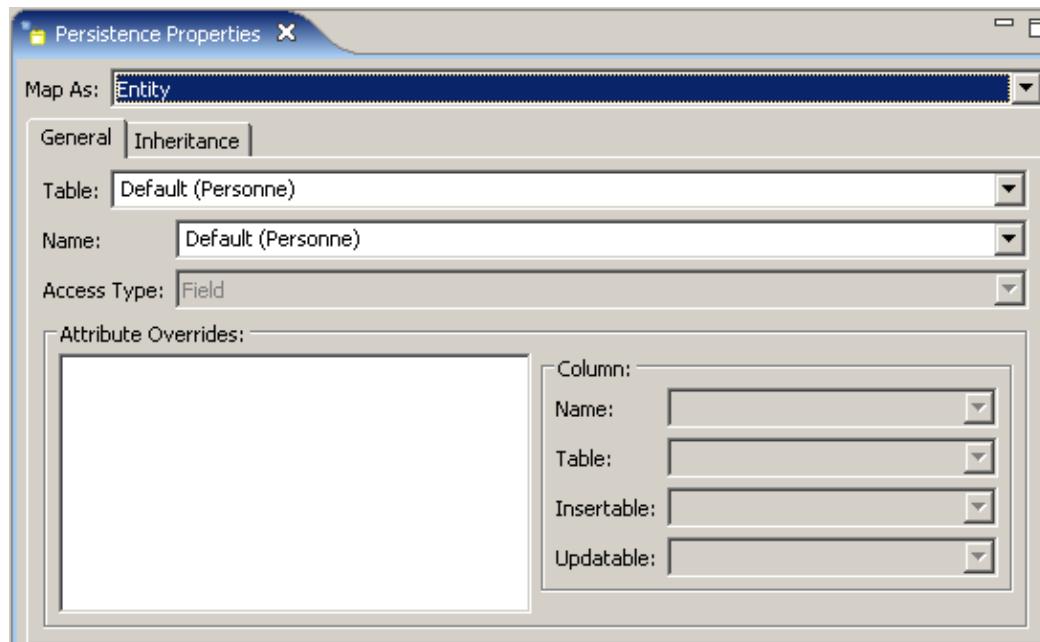
```

La classe est marquée avec l'annotation `@Entity` : elle est en erreur car aucun champ de la classe n'est marqué avec l'annotation `@Id` qui est obligatoire.

La vue « Persistence Outline » affiche la nouvelle entity créée.



La vue « Persistence properties » affiche les propriétés de cet élément.



Cette vue indique que l'entité est mappée sur la table Personne.

Il faut ajouter les champs dans l'entité :

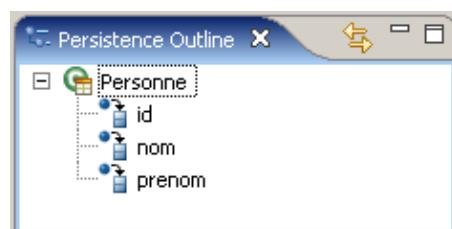
```
package com.jmdoudoux.test.dali;

import javax.persistence.Entity;

@Entity
public class Personne {

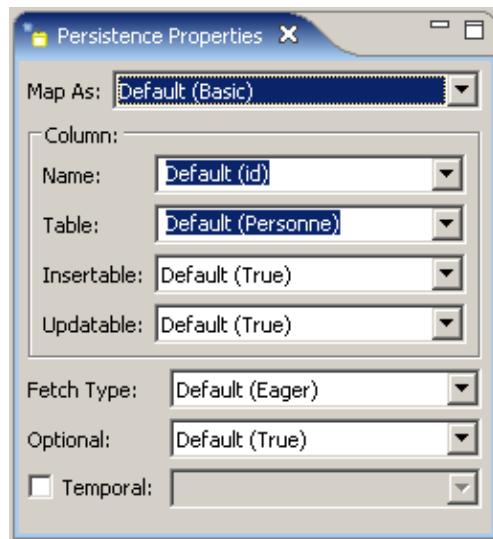
    private Long id;
    private String nom;
    private String prenom;
}
```

La vue « Persistence outline » affiche les champs ajoutés

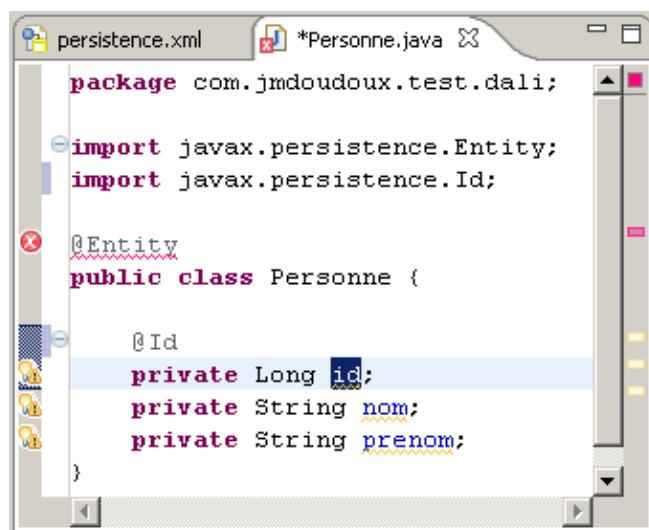


Il faut réaliser le mapping entre les champs de l'entité et les champs de la table

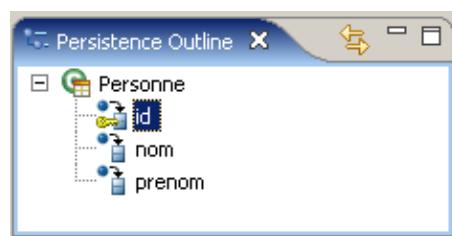
Dans la vue « Persistence outline », sélectionnez le champ id.



Modifiez le champ « Map As » en sélectionnant le type « Id » : le code de la classe est modifié pour ajouter l'annotation @Id sur le champ.



L'icône du champ est aussi modifié dans la vue « Persistence outline » pour indiquer que ce champ est la clé.



Sélectionnez le type « Basic » pour les deux autres champs. Comme le nom des champs correspond exactement dans la table et dans la classe, le mapping est automatique.

L'entité ainsi modifiée est la suivante

```

package com.jmdoudoux.test.dali;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Basic;
import javax.persistence.Table;

@Entity
public class Personne {

    @Id
    private Long id;
    @Basic
    private String nom;
    @Basic
    private String prenom;
}

```

L'éditeur affiche les erreurs de mapping si certaines sont détectées.

```

package com.jmdoudoux.test.dali;

import javax.persistence.Entity;

@Entity
public class Personne {

    @Id
    private String nom;
    @Basic
    private String prenom;
}

```

### 27.1.3. Exemple d'utilisation de l'API

Il faut rajouter dans le classpath de l'application la bibliothèque du pilote JDBC (dans cet exemple c'est la base de données MySQL qui est utilisée : le fichier mysql-connector-java-3.1.10-bin.jar est donc ajouté au classpath du projet).

Il faut enrichir le fichier persistence.xml avec les propriétés de connexion à la base de données.

Exemple :

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="test_dali">
    <provider>
      oracle.toplink.essentials.PersistenceProvider

```

```

</provider>
<class>com.jmdoudoux.test.dali.Personne</class>
<properties>
    <property name="toplink.jdbc.driver" value="com.mysql.jdbc.Driver" />
    <property name="toplink.jdbc.url" value="jdbc:mysql://localhost/test" />
    <property name="toplink.jdbc.user" value="root" />
    <property name="toplink.jdbc.password" value="" />
    <property name="toplink.logging.level" value="INFO" />
</properties>
</persistence-unit>
</persistence>

```

Il faut enfin écrire le code qui va utiliser l'API JPA et les différentes entités de mapping générées.

Exemple : obtenir la personne dont l'identifiant vaut 1

Exemple :

```

package com.jmdoudoux.test.dali;

import javax.persistence.Persistence;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityManager;

public class TestDali {

    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("test_dali");
        EntityManager em = emf.createEntityManager();
        Personne personne = em.find(Personne.class, 1L);
        System.out.println("Personne.nom=" + personne.getNom());
        em.close();
        emf.close();
    }
}

```

Résultat d'exécution :

```

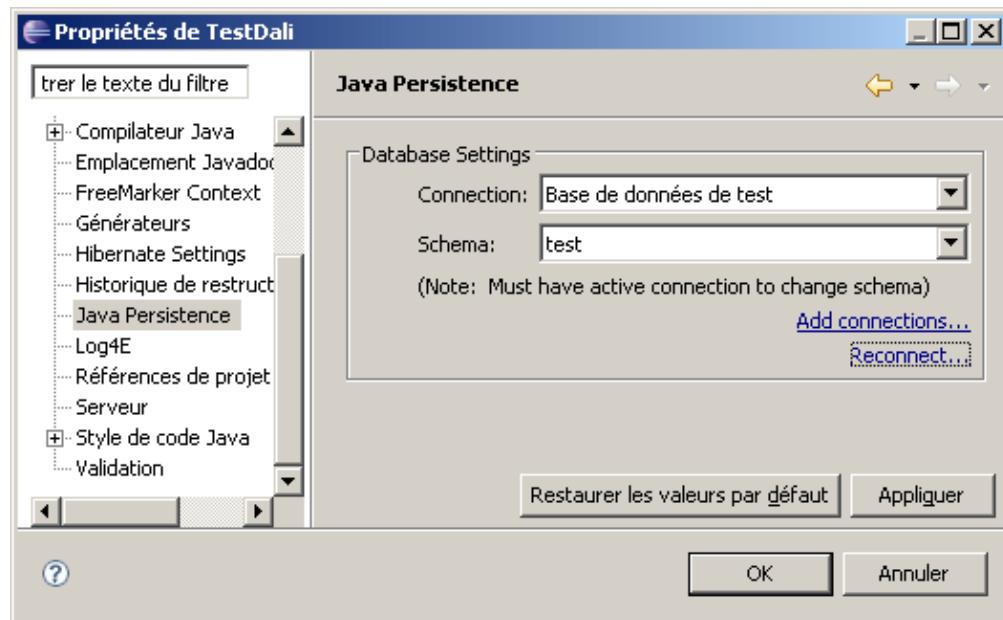
[TopLink Info]: 2006.11.06
06:01:18.134--ServerSession(25378506)--TopLink, version:
Oracle TopLink Essentials - 2006.4 (Build 060412)
[TopLink Info]: 2006.11.06
06:01:19.757--ServerSession(25378506)--file:/C:/Documents%20and%20Settings
/jumbo/workspace/TestDali/bin-test_dali
login successful
Personne.nom=nom1
[TopLink Info]: 2006.11.06
06:01:20.117--ServerSession(25378506)--file:/C:/Documents%20and%20Settings
/jumbo/workspace/TestDali/bin-test_dali
logout successful

```

## 27.1.4. Connexion à la base de données.

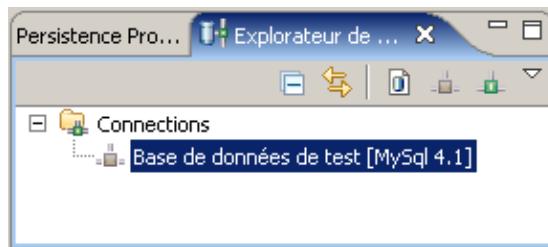
Avant de pouvoir utiliser les fonctionnalités du plug-in, il est nécessaire d'être connecté à la base de données afin de pouvoir contrôler les informations de mapping.

Il est possible de le faire dans les propriétés du projet.



Sélectionnez Java Persistence et cliquez sur le bouton Reconnect .

Il est aussi possible d'utiliser la vue « Explorateur de base de données ».

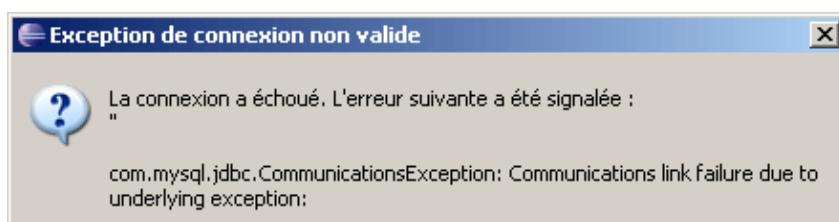


Sélectionnez la connexion et cliquez sur l'option "Reconnecter" du menu contextuel

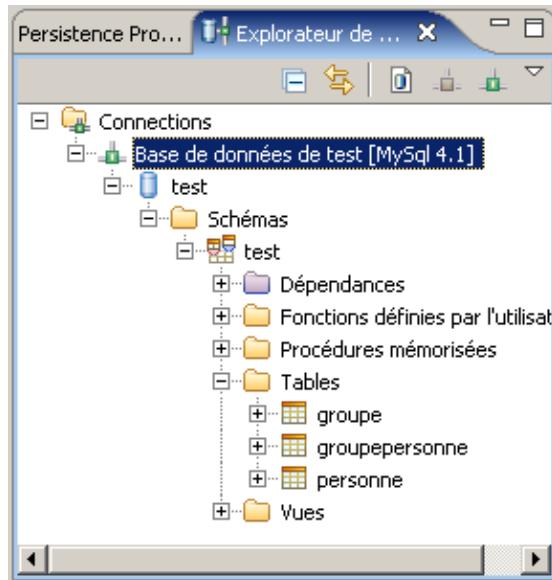


Une boîte de dialogue permet de saisir le login et le mot de passe de connexion.

Si la connexion échoue alors un message d'erreur affiche la pile d'appel des exceptions



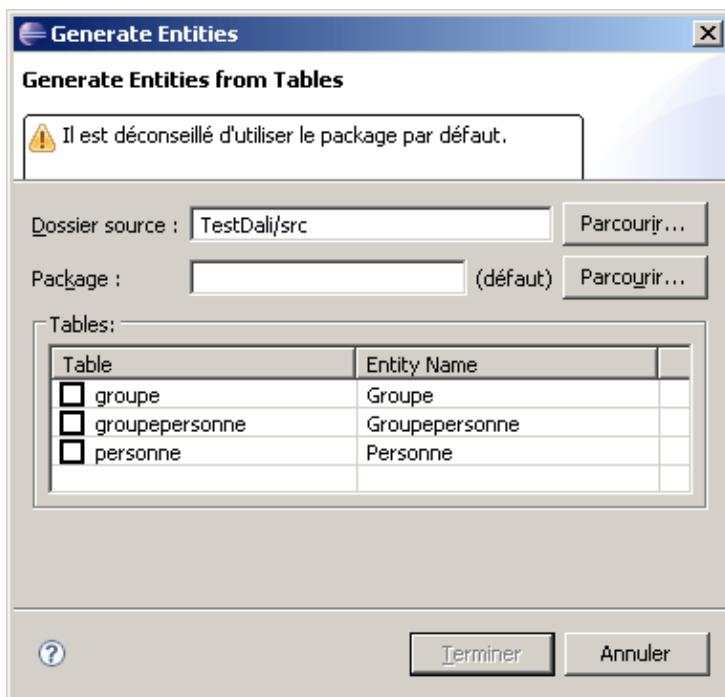
Si la connexion réussie, la base de données est affichée avec une petite icône verte et il est possible de parcourir l'arborescence des éléments qui la compose.



### 27.1.5. La génération des entités à partir de la base de données

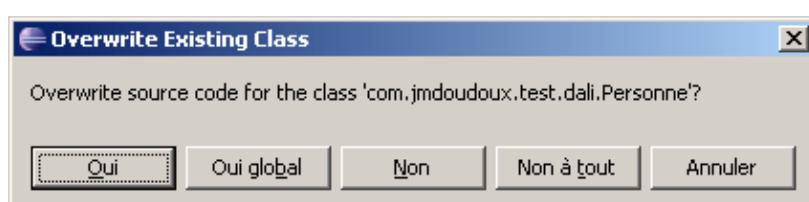
Le plug-in propose une fonctionnalité pour générer automatiquement les entités à partir des tables d'une base de données connectées.

Selectionnez le projet et utilisez l'option Java « Java Persistence/Generate entities »



Saisissez le nom du package, sélectionnez les tables concernées et cliquez sur le bouton « Terminer ».

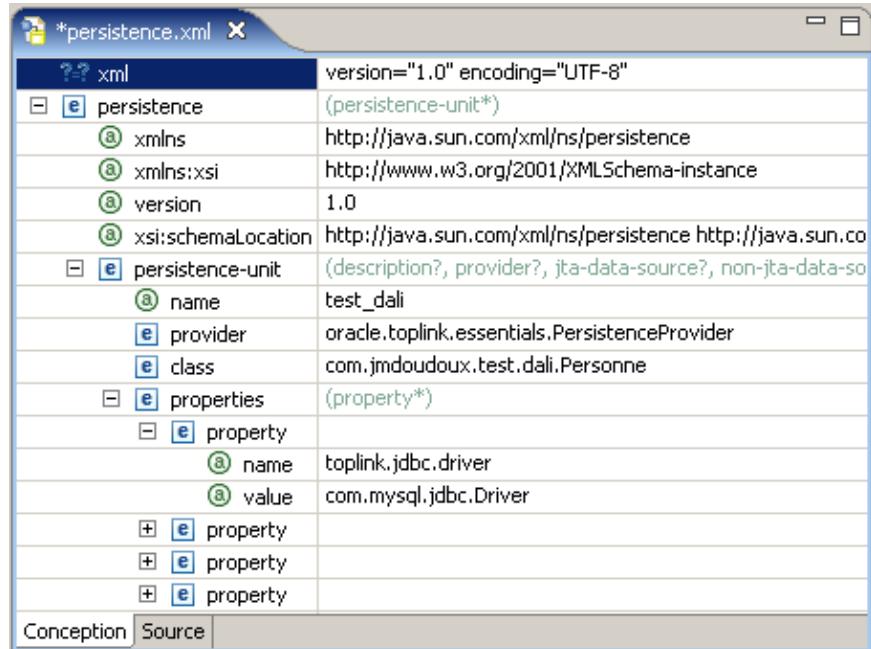
Si une classe correspondant à une entité existante, alors un message de confirmation est affiché :



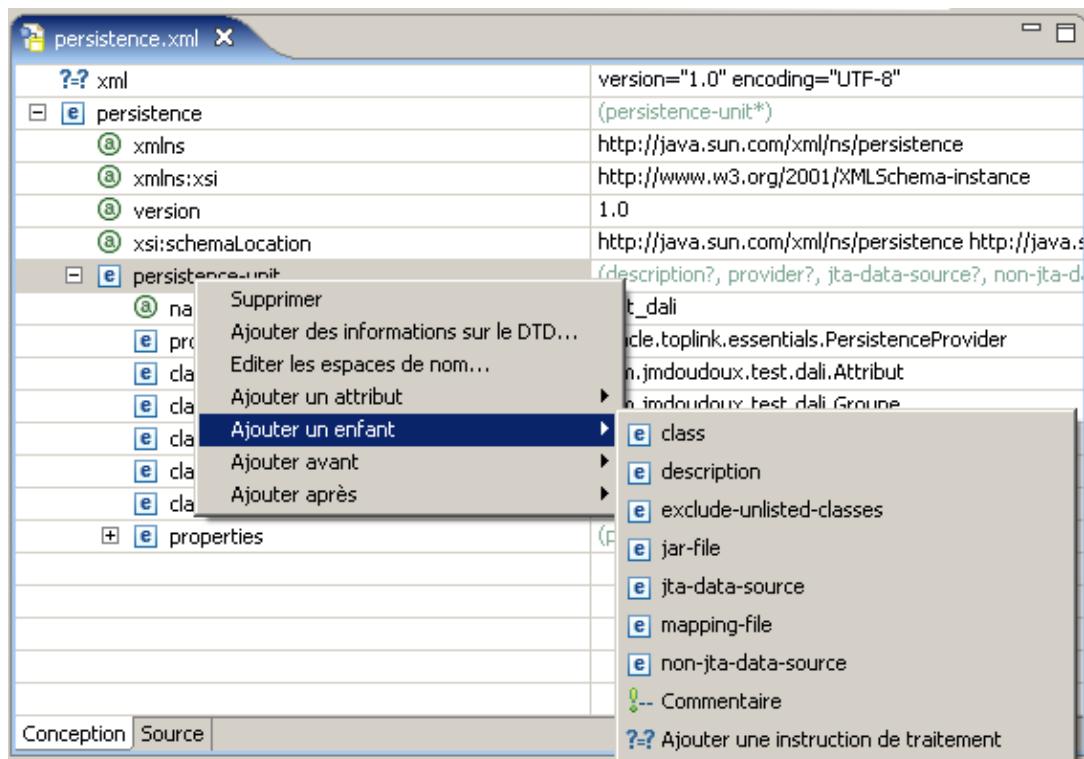
Les différentes classes des entités sont générées.

### 27.1.6. L'éditeur du fichier persistence.xml

Dali propose un éditeur dédié à la modification du fichier persistence.xml. C'est l'éditeur par défaut de ce fichier. Il propose deux onglets : Conception et Source



La modification du fichier est facilitée par l'éditeur qui intègre le schéma du document XML.



## 27.1.7. La synchronisation des classes

Dans l'explorateur de packages, il faut sélectionner le fichier persistence.xml et utiliser l'option « Persistence / Synchronize classes » du menu contextuel.

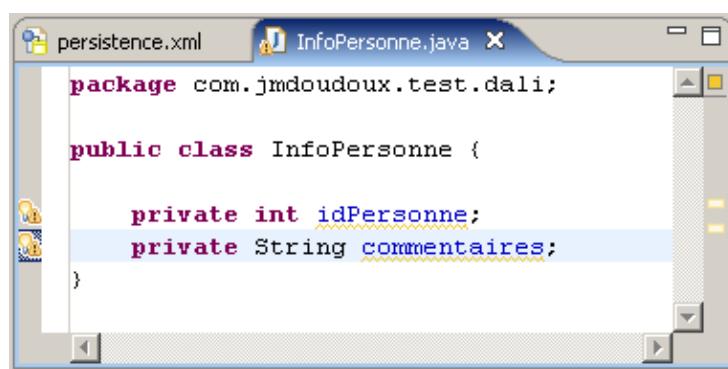
Chaque nouvelle entité définie dans les sources est automatiquement ajoutée dans un tag <class> du fichier persistence.xml.

Le travail de synchronisation est effectué en tâche de fond.

## 27.1.8. Transformer une classe existante en entité

Il est possible de transformer une classe existante en Entity.

Exemple :



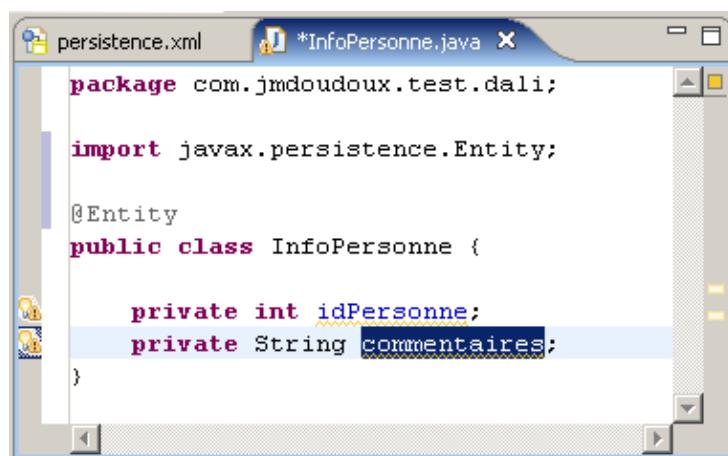
```
package com.jmdoudoux.test.dali;

public class InfoPersonne {

    private int idPersonne;
    private String commentaires;
}
```

Sélectionnez la classe (ne sélectionnez pas le fichier .java mais la classe directement) dans l'explorateur de packages et utiliser l'option « Persistence / Make Java Persistence Entity ».

La clause import nécessaire et l'annotation « Entity » sont ajoutées :



```
package com.jmdoudoux.test.dali;

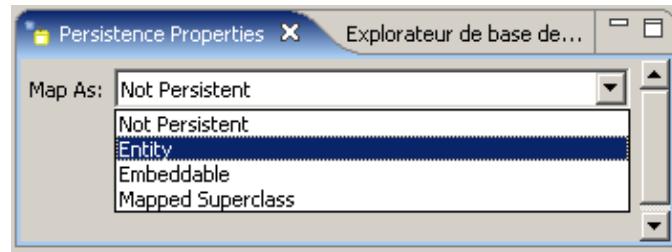
import javax.persistence.Entity;

@Entity
public class InfoPersonne {

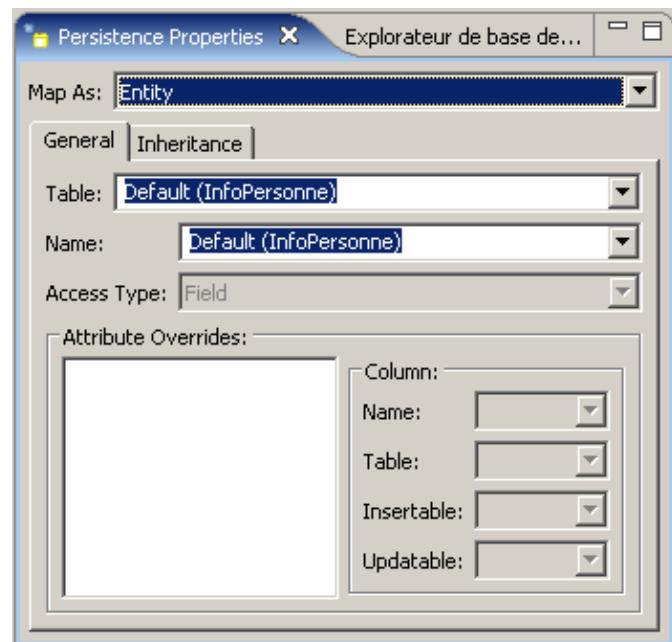
    private int idPersonne;
    private String commentaires;
}
```

Dans l'éditeur de code, sélectionnez la classe puis affichez la vue « Persistence Properties »

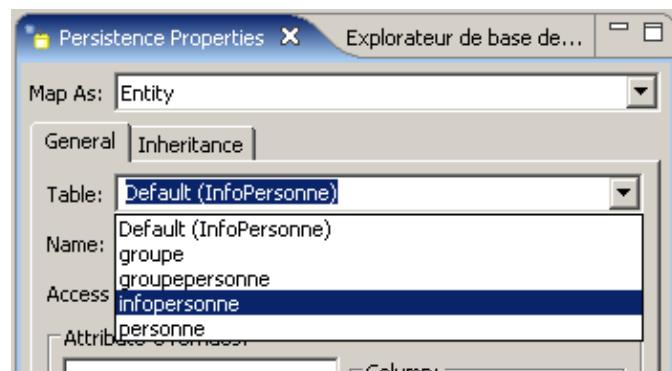
Dans la liste déroulante « Map As » sélectionnez « Entity »



La boîte de dialogue s'enrichie avec les propriétés relative au type Entity.



Dans la liste déroulante Table, sélectionnez la table infopersonne



Le code de l'entité est modifié

A screenshot of the Eclipse IDE showing two open files. On the left is "persistence.xml" containing XML configuration for persistence. On the right is "\*InfoPersonne.java" containing Java code for an entity class. The code includes imports for javax.persistence.Entity and javax.persistence.Table, and a class definition with annotations @Entity and @Table(name="infopersonne").

```
package com.jmdoudoux.test.dali;

import javax.persistence.Entity;
import javax.persistence.Table;

@Entity
@Table(name="infopersonne")
public class InfoPersonne {
```

## 27.1.9. La création d'une association 1-1

L'exemple de cette section va utiliser les tables ci dessous



La table « personne » contient les données suivantes :

idPersonne	nomPersonne	prenomPersonne	idInfoPersonne
1	nom1	prenom1	1
2	nom2	prenom2	Null
3	nom3	prenom3	Null
4	nom4	prenom4	Null
5	nom5	prenom5	Null

La table « infopersonne » contient les données suivantes :

idInfoPers...	commentaires
1	commentaires de test

Il faut mettre en place l'association en modifiant la classe Personne

Exemple :

```
package com.jmdoudoux.test.dali;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Basic;
import javax.persistence.Table;

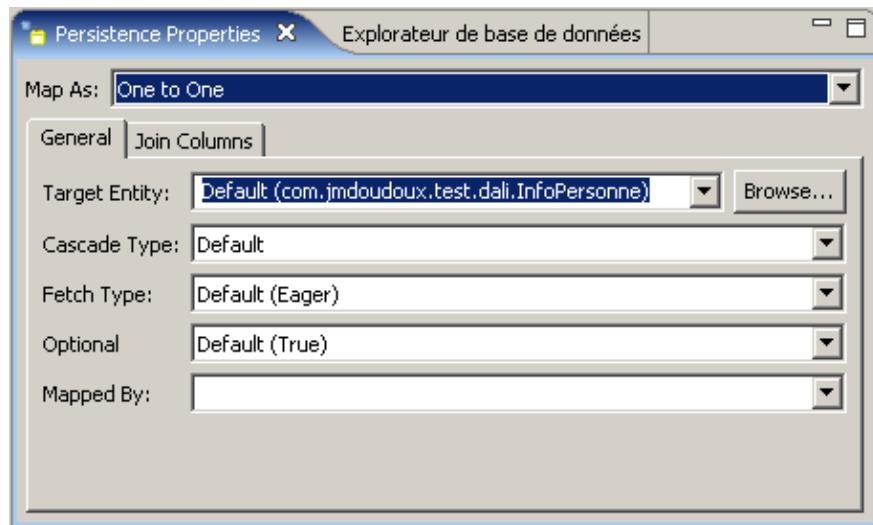
@Entity
@Table(name="personne")
public class Personne {

    @Id
    private Long idPersonne;

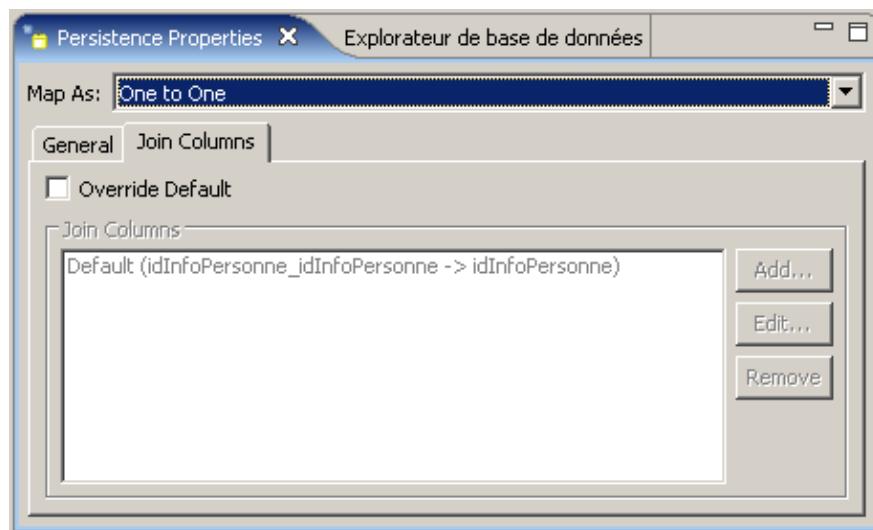
    @Basic
    private String nomPersonne;

    @Basic
    private String prenomPersonne;
    private InfoPersonne idInfoPersonne;
}
```

Dans l'éditeur de code, sélectionnez le champ idInfoPersonne

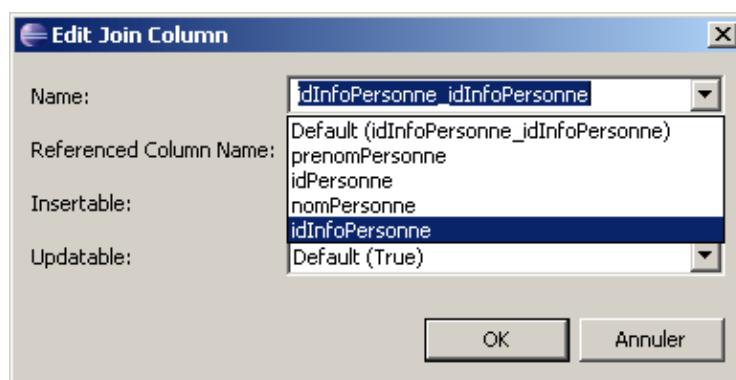


Dans la vue « Persistence Properties », l'onglet « Join columns » permet de modifier les options de la jointure



Cochez la case « Override Default »

Sélectionnez la ligne dans la liste et cliquez sur le bouton « Edit » pour ouvrir la boîte de dialogue « Edit Join Column »



Dans la liste déroulante Name, sélectionnez le champ idInfoPersonne

Cliquez sur le bouton « OK » pour modifier le code de l'entité

**Exemple :**

```
@OneToOne
```

```

@JoinColumn(name="idInfoPersonne", referencedColumnName = "idInfoPersonne")
private      InfoPersonne idInfoPersonne;

```

Sources complets de l'exemple :

#### Le fichier Persistence.xml :

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="test_dali">
    <provider>
      oracle.toplink.essentials.PersistenceProvider
    </provider>
    <class>com.jmdoudoux.test.dali.InfoPersonne</class>
    <class>com.jmdoudoux.test.dali.Personne</class>
    <properties>
      <property name="toplink.jdbc.driver"
        value="com.mysql.jdbc.Driver" />
      <property name="toplink.jdbc.url"
        value="jdbc:mysql://localhost/test" />
      <property name="toplink.jdbc.user" value="root" />
      <property name="toplink.jdbc.password" value="" />
      <property name="toplink.logging.level" value="INFO" />
    </properties>
  </persistence-unit>
</persistence>

```

#### Le fichier InfoPersonne.java :

```

package com.jmdoudoux.test.dali;

import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.Column;
import javax.persistence.Id;
import javax.persistence.Basic;

@Entity
@Table(name="infopersonne")
public class InfoPersonne {

    @Id
    @Column(name="idInfoPersonne")
    private long idInfoPersonne;

    @Basic
    private String commentaires;

    public String getCommentaires() {
        return commentaires;
    }

    public void setCommentaires(String commentaires) {
        this.commentaires = commentaires;
    }

    public long getIdInfoPersonne() {
        return idInfoPersonne;
    }

    public void setIdInfoPersonne(long idInfoPersonne) {
        this.idInfoPersonne = idInfoPersonne;
    }
}

```

```
}
```

### Le fichier Personne.java

```
package com.jmdoudoux.test.dali;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Basic;
import javax.persistence.Table;
import javax.persistence.OneToOne;
import javax.persistence.JoinColumn;

@Entity
@Table(name="personne")
public class Personne {

    @Id
    private Long idPersonne;

    @Basic
    private String nomPersonne;

    @Basic
    private String prenomPersonne;

    @OneToOne
    @JoinColumn(name="idInfoPersonne", referencedColumnName = "idInfoPersonne")
    private InfoPersonne idInfoPersonne;

    public Long getId() {
        return idPersonne;
    }

    public void setId(Long id) {
        this.idPersonne = id;
    }

    public String getNom() {
        return nomPersonne;
    }

    public void setNom(String nom) {
        this.nomPersonne = nom;
    }

    public String getPrenom() {
        return prenomPersonne;
    }

    public void setPrenom(String prenom) {
        this.prenomPersonne = prenom;
    }

    public InfoPersonne getIdInfoPersonne() {
        return idInfoPersonne;
    }

    public void setIdInfoPersonne(InfoPersonne idInfoPersonne) {
        this.idInfoPersonne = idInfoPersonne;
    }
}
```

### Le fichier TestDali.java

```
package com.jmdoudoux.test.dali;
```

```

import javax.persistence.Persistence;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityManager;

public class TestDali {

    public static void main(String[] argv) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("test_dali");
        EntityManager em = emf.createEntityManager();
        Personne personne = em.find(Personne.class, 11);
        System.out.println("Personne.nom=" + personne.getNom());
        System.out.println("commentaires = "
            + personne.getIdInfoPersonne().getCommentaires());
        em.close();
        emf.close();
    }
}

```

#### Résultat d'exécution :

```

[TopLink Info]: 2006.11.09
 07:06:29.267--ServerSession(31999426)--TopLink, version:
  Oracle TopLink Essentials - 2006.4 (Build 060412)
[TopLink Info]: 2006.11.09
 07:06:30.679--ServerSession(31999426)--file:/C:/Documents%20and%20Settings
 /jumbo/workspace/TestDali/bin-test_dali
  login successful
Personne.nom=nom1
commentaires = commentaires de test
[TopLink Info]: 2006.11.09 07:06:31.009--ServerSession(31999426)--file:/C:
 /Documents%20and%20Settings
 /jumbo/workspace/TestDali/bin-test_dali
  logout successful

```

L'exemple fonctionne correctement car la clé étrangère existe. Il est nécessaire de gérer le cas si la clé n'existe pas.

#### Exemple : demander les informations de la personne ayant pour id 2

```

...
    Personne personne = em.find(Personne.class, 21);
...

```

#### Résultat :

```

[TopLink Info]: 2006.11.10
 05:17:15.523--ServerSession(31999426)--TopLink, version:
  Oracle TopLink Essentials - 2006.4 (Build 060412)
[TopLink Info]: 2006.11.10
 05:17:16.895--ServerSession(31999426)--file:/C:/Documents%20and%20Settings
 /jumbo/workspace/TestDali/bin-test_dali
  login successful
Personne.nom=nom2
Exception in thread "main" java.lang.NullPointerException
      at com.jmdoudoux.test.dali.TestDali.main(TestDali.java:16)

```

Il suffit de tester si la clé étrangère est différente de null.

#### Exemple :

```

Personne personne = em.find(Personne.class, 21);

```

```
System.out.println("Personne.nom= " + personne.getNom());
if (personne.getIdInfoPersonne() != null) {
    System.out.println("commentaires = "
        + personne.getIdInfoPersonne().getCommentaires());
}
```

## **Partie 6 : le développement d'applications mobiles**

# **Partie 6 : le développement d'applications mobiles**

Cette partie concerne l'utilisation d'Eclipse pour développer des applications pour appareils mobiles avec la version Mobile Edition de Java.

Elle comporte les chapitres suivants :

- ◆ Le développement avec J2ME : Présente le plug-in EclipseME qui permet de faciliter le développement d'applications J2ME.

## 28. Le développement avec J2ME

# Chapitre 28

### 28.1. EclipseME

Le but de ce plug-in est de permettre le développement d'applications J2ME reposant sur MIDP en utilisant un Wireless Toolkit.

Les fonctionnalités proposées par ce plug-in sont :

- Le support de plusieurs Wireless Toolkit
- Un assistant de création de projets de type Midlet Suite
- Un assistant de création de Midlets
- Un éditeur pour les fichiers .jad
- Une compilation incrémentale avec pré-vérification
- Le débogage du code des Midlets
- L'exécution dans les emulateurs fournis avec le Wireless Toolkit
- La création d'un package pour les applications J2ME
- La création d'un package obscurci avec Proguard
- Le support du mode « Over The Air »

Le site officiel de ce plug-in est à l'url : <http://eclipseme.org/>

Un JDK 1.4, une version 3.0.x d'Eclipse et un Wireless Toolkit sont des pré-requis pour pouvoir utiliser EclipseME.

Cette section va mettre en oeuvre les outils suivants sous Windows :

Outil	Version	Rôle
JDK	1.4.2_04	
Eclipse	3.0.1	IDE
EclipseME	0.75	
Proguard	3.2	
J2ME Wireless Toolkit	2.1	Kit de développement J2ME

#### 28.1.1. Installation

Pour installer le plug-in, téléchargez le fichier eclipseme.feature\_0.7.5\_site.zip et enregistrez le dans un répertoire du système.

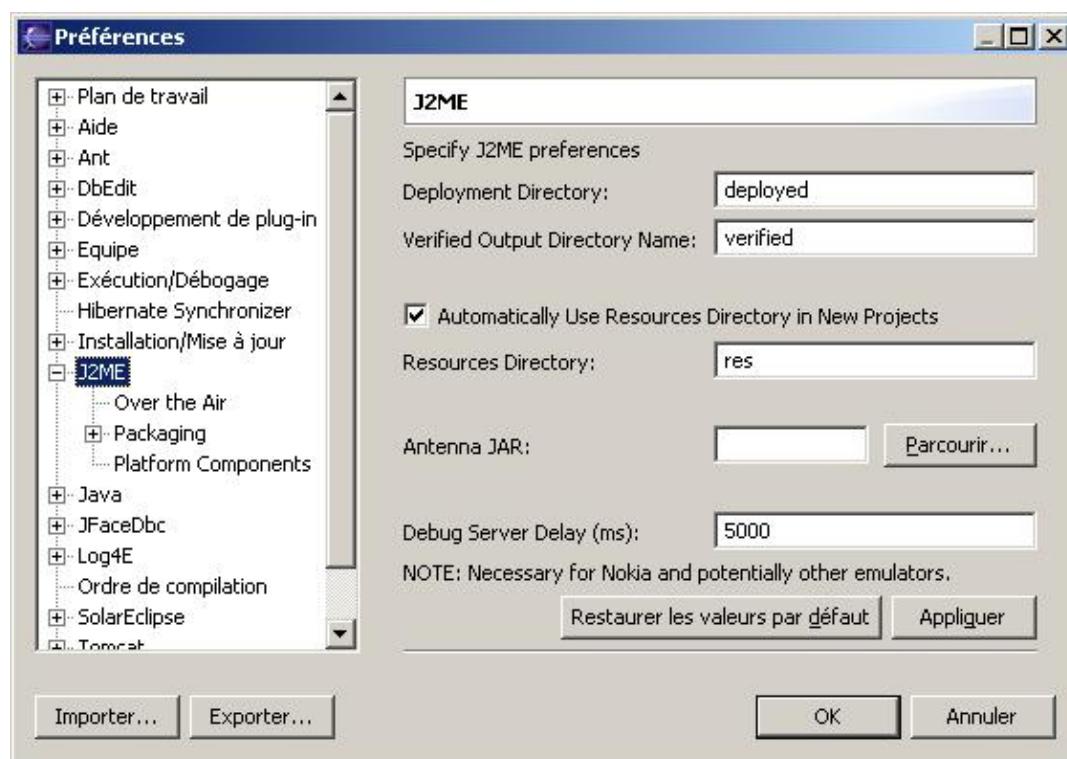
Il suffit alors de suivre les étapes suivantes :

- Utilisez l'option « Mise à jour de logiciels/Rechercher et installer » du menu Aide.
- Sélectionnez « Rechercher les nouveaux dispositifs à installer » et cliquez sur le bouton « Suivant »
- Cliquez sur le bouton « Nouveau site archivé »
- Sélectionnez le fichier et cliquez sur « Ouvrir »
- Dans l'arborescence des sites, sélectionnez eclipseme.feature\_0.7.5\_site.zip et cliquez sur le bouton « Suivant »
- Sélectionnez les dispositifs « EclipseMe » et « eclipseme.features.siemens »
- Lisez la licence et si vous l'acceptez cliquer sur « J'accepte les termes du contrat » et cliquez sur le bouton « Suivant »
- Cliquez sur le bouton « Fin »
- Lors de l'affichage de la boîte de dialogue « Vérification du dispositif », cliquez sur le bouton « Installer »
- Acceptez de relancer le plan de travail.

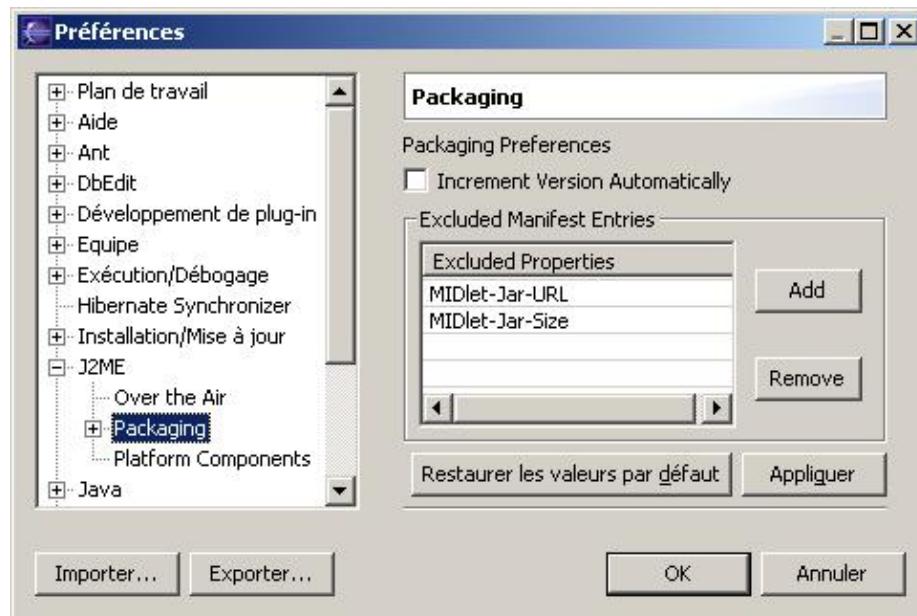
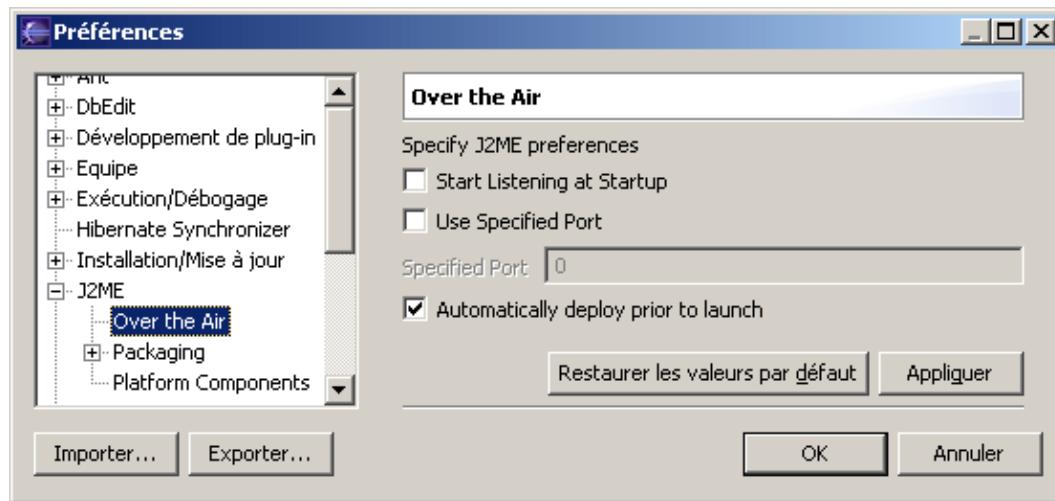
### 28.1.2. Les préférences du plug-in

Le plug-in propose plusieurs pages pour gérer les options du plug-in lors de son utilisation.

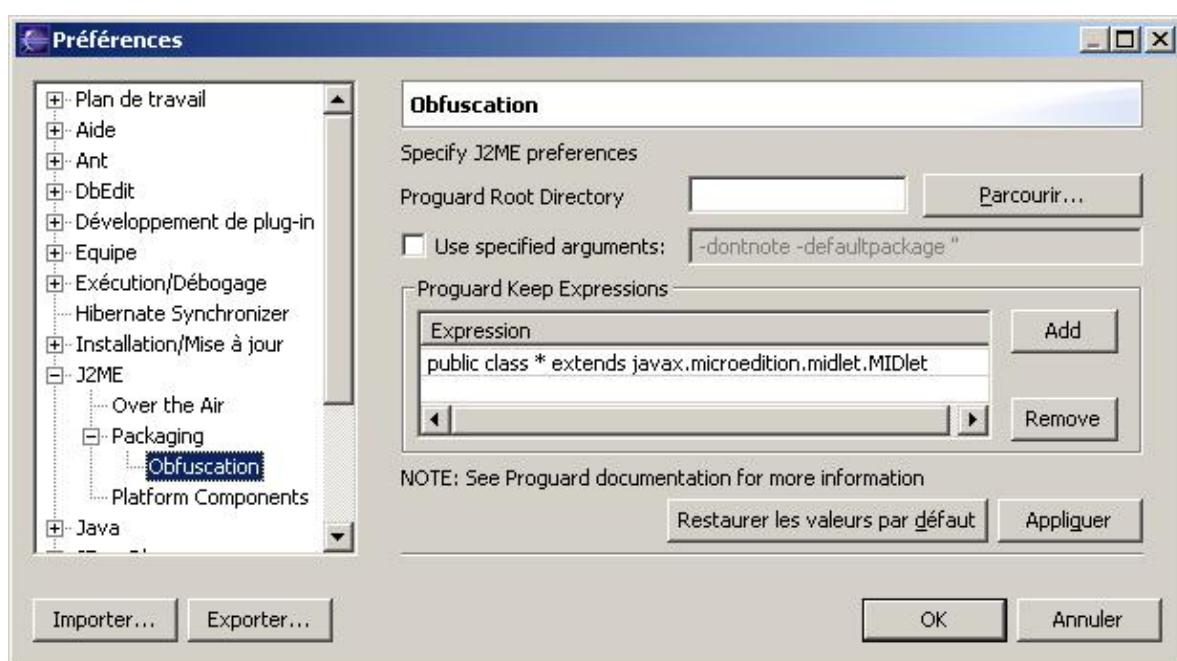
Ces pages se trouvent sous l'arborescence « J2ME ».



Il est possible de configurer le mode « Over the Air » en sélectionnant « J2ME/Over the Air » dans l'arborescence.

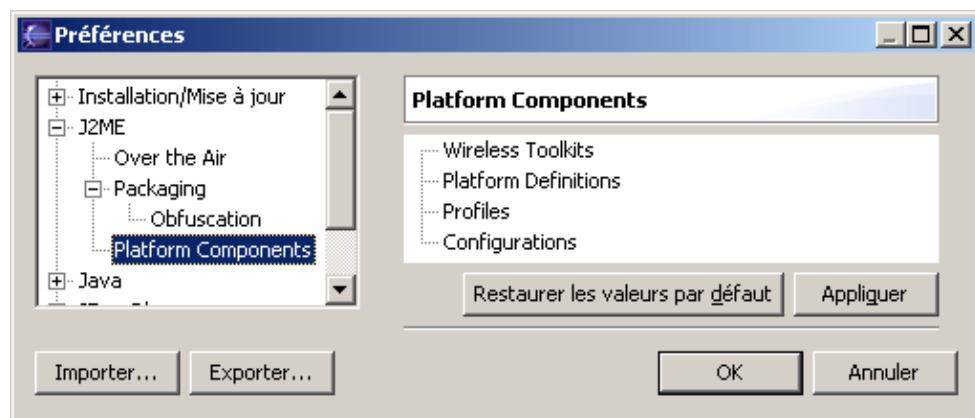
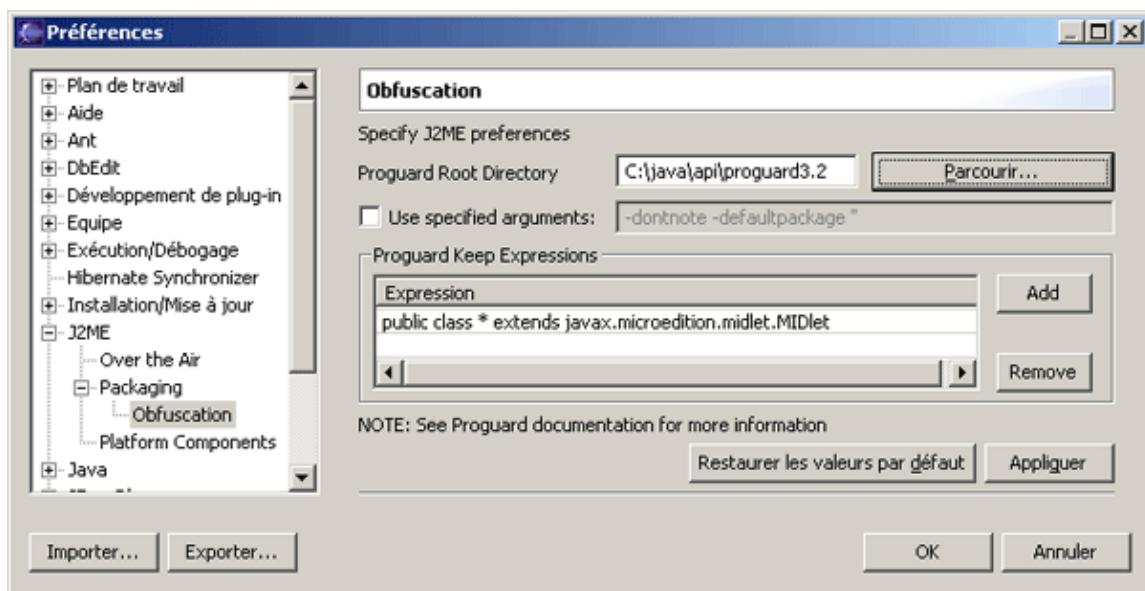


EclipseMe peut travailler avec Proguard, un projet open source qui a pour but de proposer un outil pour rendre le code obscurci. Pour cela, il faut sélectionner J2ME/Packaging/Obfuscation



Pour l'utiliser Proguard, il faut télécharger le fichier proguard3.2.zip sur le site <http://proguard.sourceforge.net/> et le décompresser dans un répertoire.

Il suffit alors de sélectionner le répertoire qui contient Proguard en cliquant sur le bouton « Parcourir ».

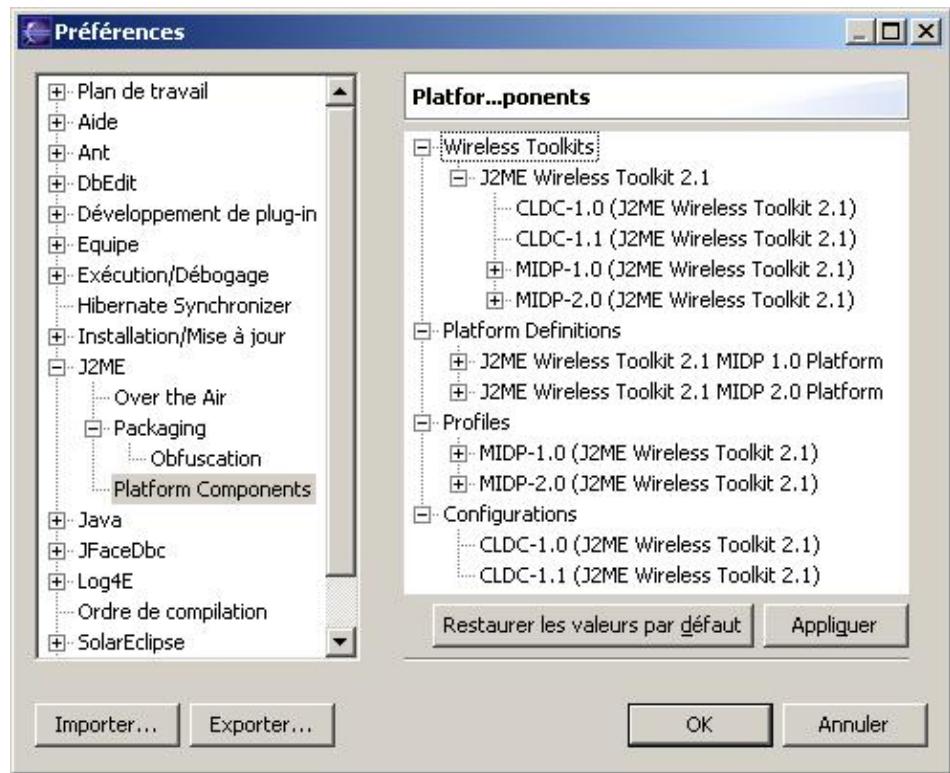


Cet écran permet de configurer le Wireless Toolkit qui sera utilisé avec le plug-in.

Par exemple, pour ajouter le J2ME Wireless Toolkit 2.1, sélectionnez l'option « Add Wireless Toolkit » du menu contextuel de « Wireless Toolkits »

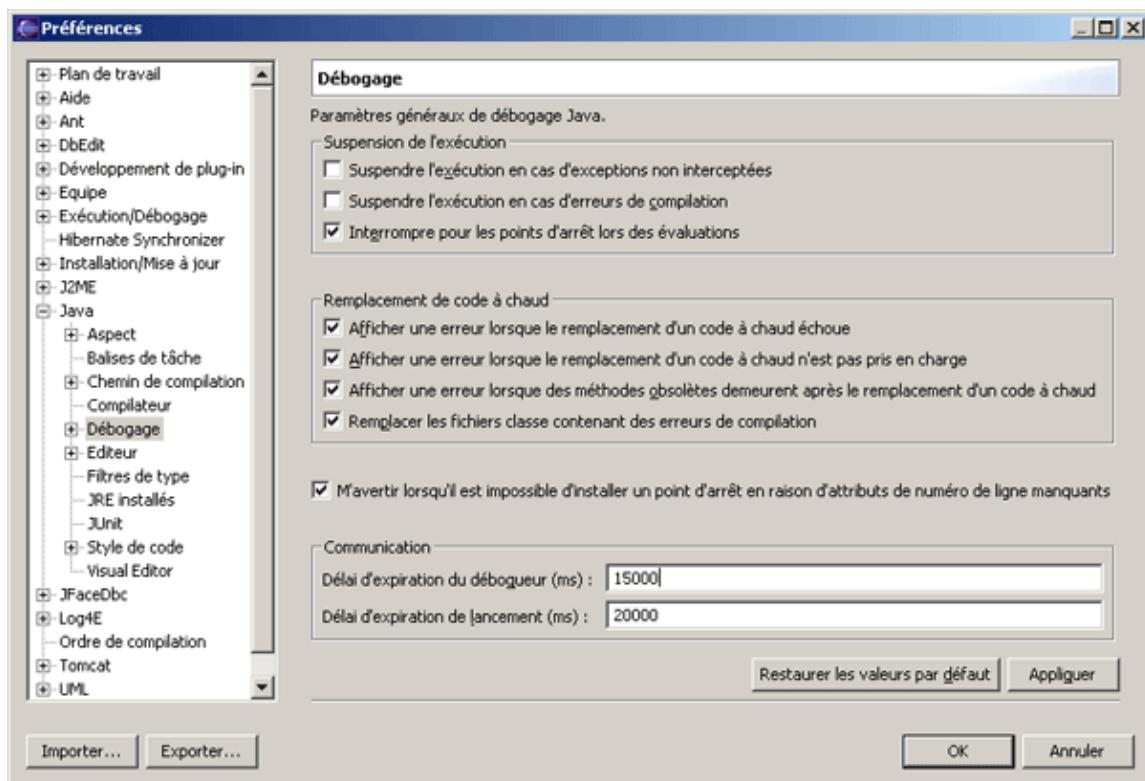


Cliquez sur le bouton « Browse » pour sélectionner le répertoire où le J2ME Wireless Toolkit 2.1 est installé et cliquez sur le bouton « Fin ».



Pour pouvoir utiliser le débogueur d'Eclipse avec un « Wireless Toolkit », il faut modifier les paramètres du débogueur.

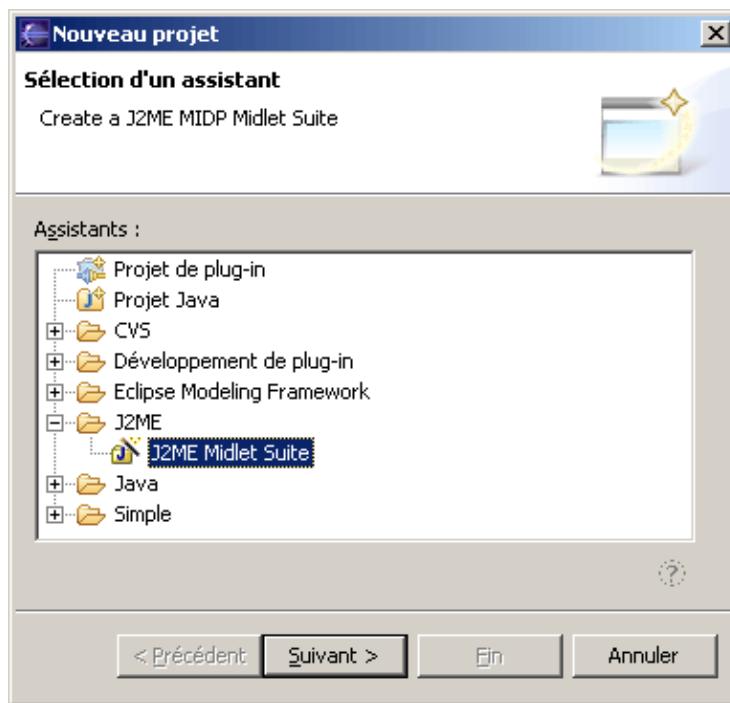
Dans les préférences, décochez les deux premières options et modifiez la valeur du délai d'expiration du débogueur pour mettre la valeur 15000.



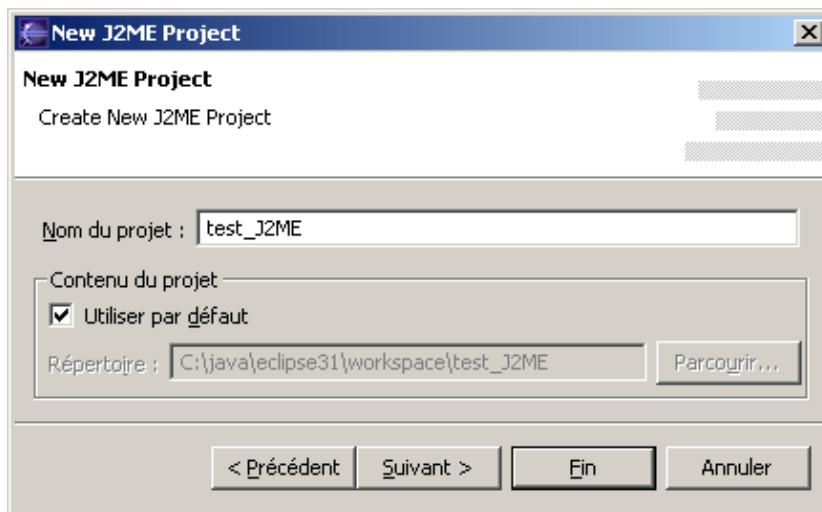
Cliquez sur le bouton « OK »

### 28.1.3. Création d'un premier exemple

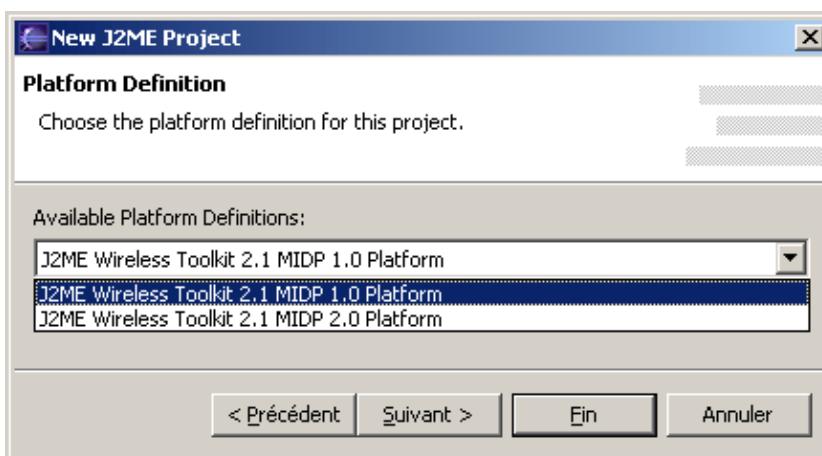
Il faut créer un nouveau projet de type « J2ME/J2ME Midlet Suite ».



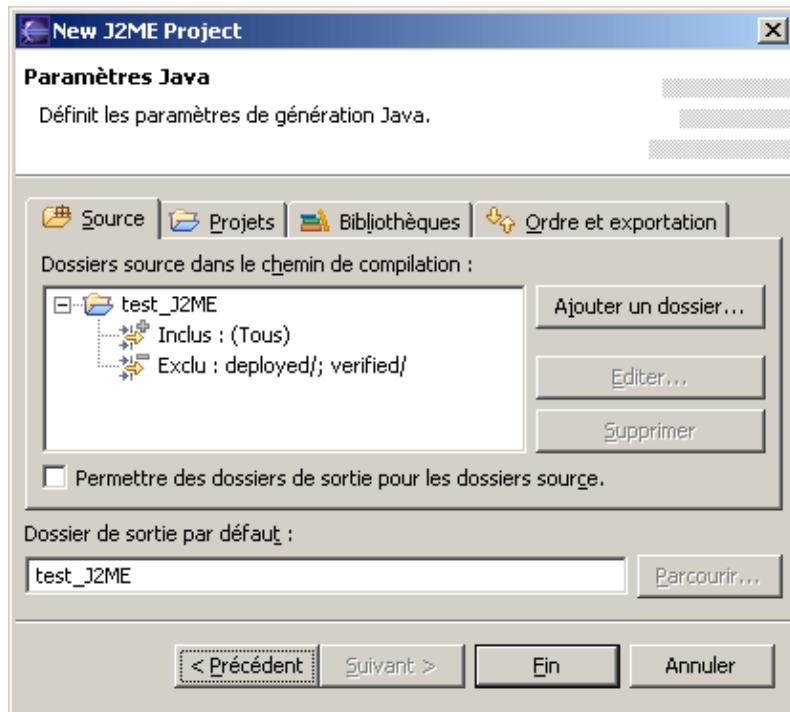
Cliquez sur le bouton « Suivant ».



Saisissez le nom du projet et cliquez sur le bouton « Suivant ».

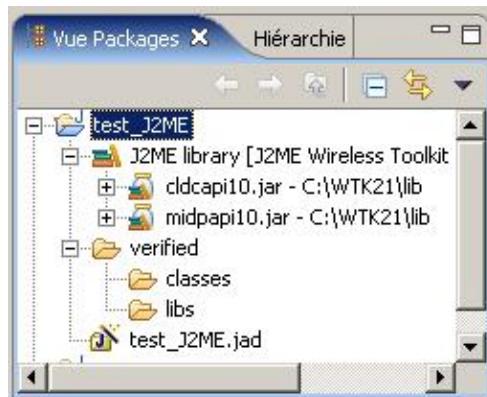


Sélectionnez la plate-forme et cliquez sur le bouton « Suivant ».

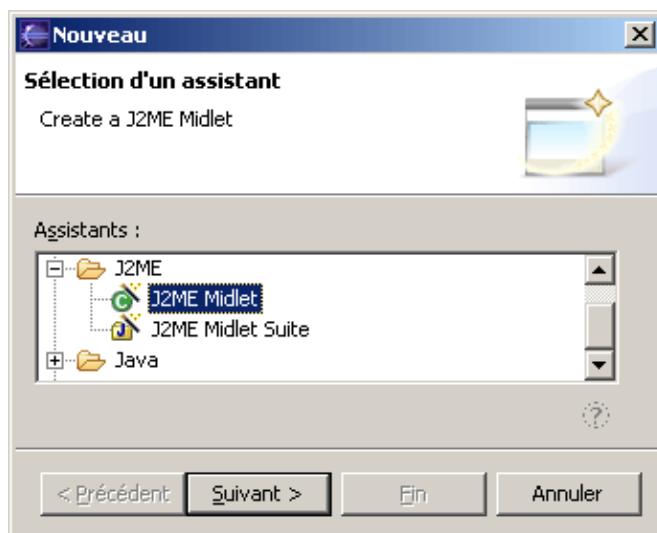


La dernière page de l'assistant permet de gérer les paramètres du projet tel qu'il est possible de le faire pour un projet Java classique.

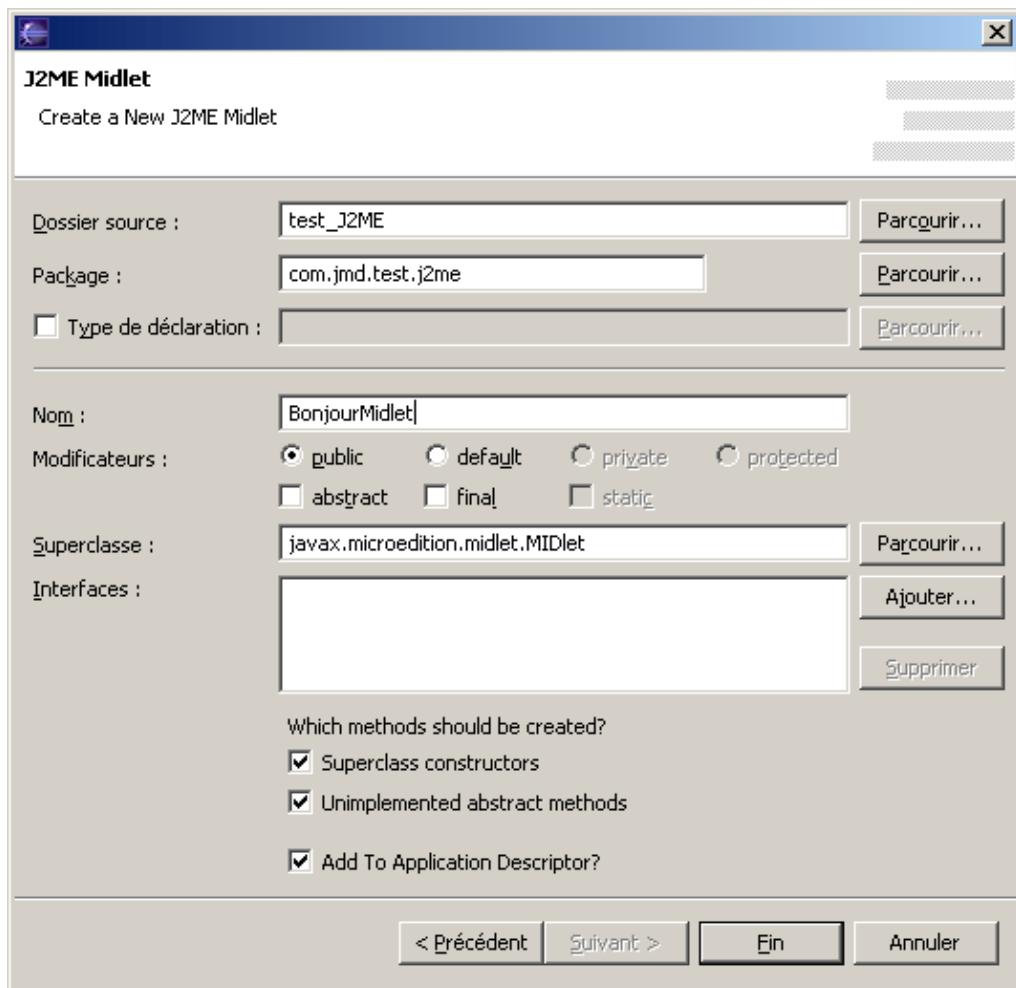
Cliquez sur le bouton « Fin » pour créer le projet.



Ajouter une midlet en créant une nouvelle entité de type « J2ME/J2ME Midlet ».



Cliquez sur le bouton « Suivant ».



La page suivante de l'assistant permet de préciser les caractéristiques de la midlet.

Saisissez le nom du package et le nom de la classe et cliquez sur le bouton « Fin ».

Il faut ensuite saisir le code de la midlet.

#### Exemple :

```
package com.jmd.test.j2me;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class BonjourMidlet extends MIDlet {
    private Display display;
    private TextBox textbox;
    public BonjourMidlet() {
        super();
        display = Display.getDisplay(this);
        textbox = new TextBox("", "Bonjour", 20, 0);
    }

    /* (non-Javadoc)
     * @see javax.microedition.midlet.MIDlet#startApp()
     */
    protected void startApp() throws MIDletStateChangeException {
        display.setCurrent(textbox);
    }

    /* (non-Javadoc)
```

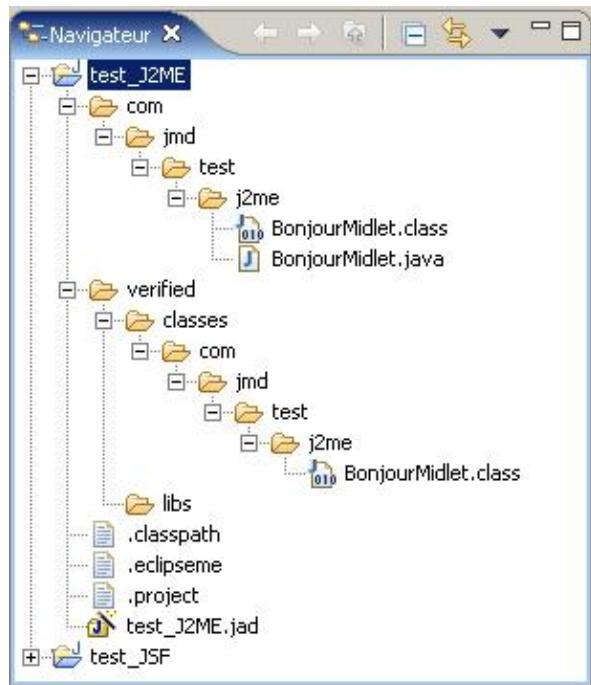
```

    * @see javax.microedition.midlet.MIDlet#pauseApp()
    */
protected void pauseApp() {
    // TODO Raccord de méthode auto-générée
}

/* (non-Javadoc)
 * @see javax.microedition.midlet.MIDlet#destroyApp(boolean)
 */
protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
    // TODO Raccord de méthode auto-générée
}
}

```

La structure du projet est la suivante :

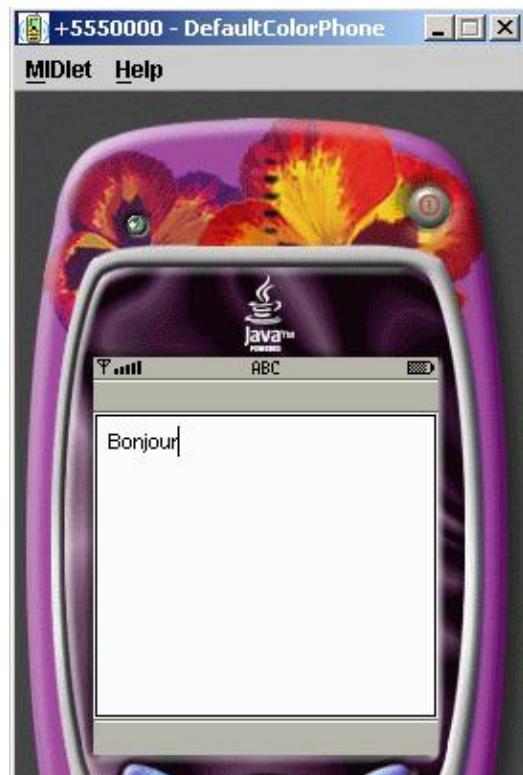


#### 28.1.4. Exécution de l'application

Pour exécuter la midlet, il faut cliquer sur la petite flèche de l'icône de la barre d'outils.

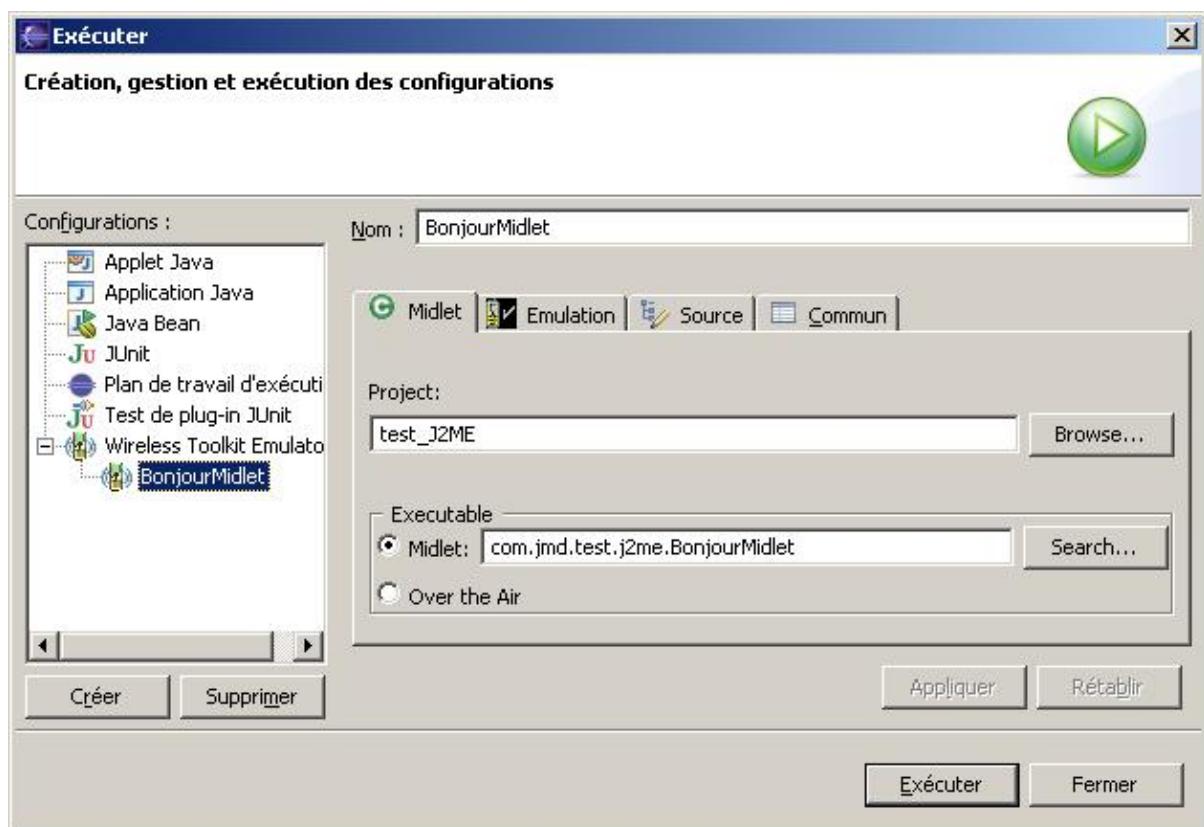


Il suffit alors de sélectionner « Emulated J2ME Midlet » pour lancer l'application dans l'émulateur.

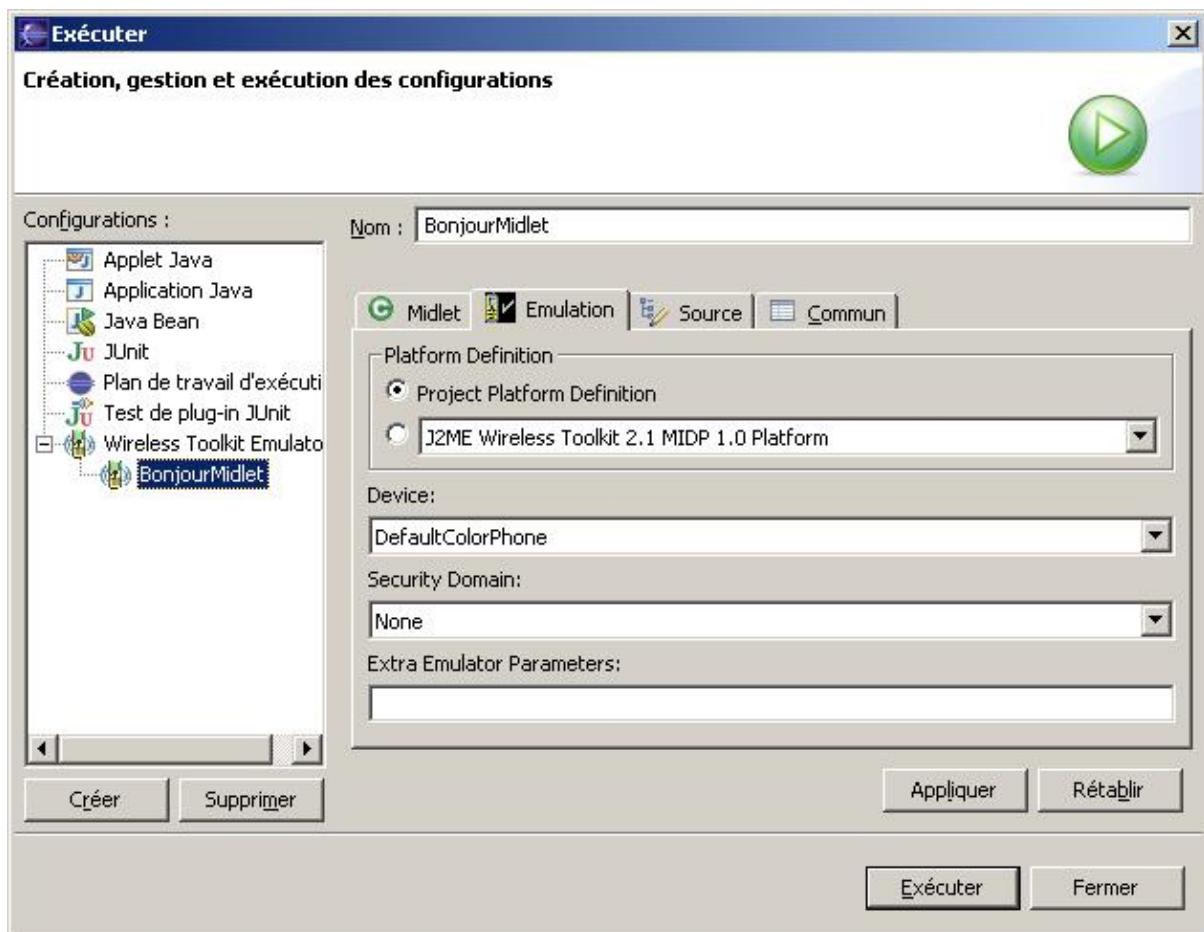


L'émulateur couleur par défaut est utilisé pour exécuter l'application.

Pour permettre de paramétriser le mode d'exécution de la midlet, il suffit d'utiliser l'option « Exécuter en tant que ... »



L'onglet Midlet permet notamment de préciser le mode d'exécution.



L'onglet Emulation permet de préciser la plate-forme et l'émulateur à utiliser.

### 28.1.5. Déboguer l'application

Pour déboguer l'application, il suffit de placer un point d'arrêt dans le code et de cliquer sur la petite flèche de l'icône  dans la barre de tache.



La perspective « Débogueur » s'ouvre lors de l'exécution de l'instruction contenant le point d'arrêt.

```

    *
    */
public BonjourMidlet() {
    super();
    display = Display.getDisplay(this);
    textbox = new TextBox("", "Bonjour", 20, 0);
}

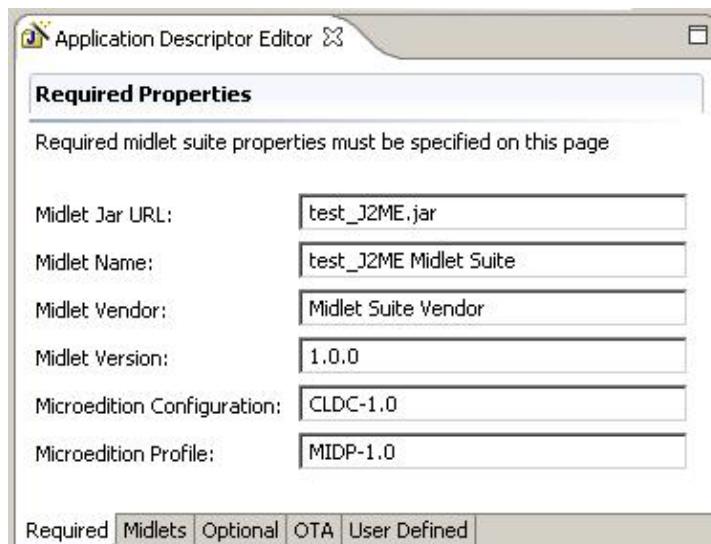
```

### 28.1.6. Modification des propriétés du descripteur d'application

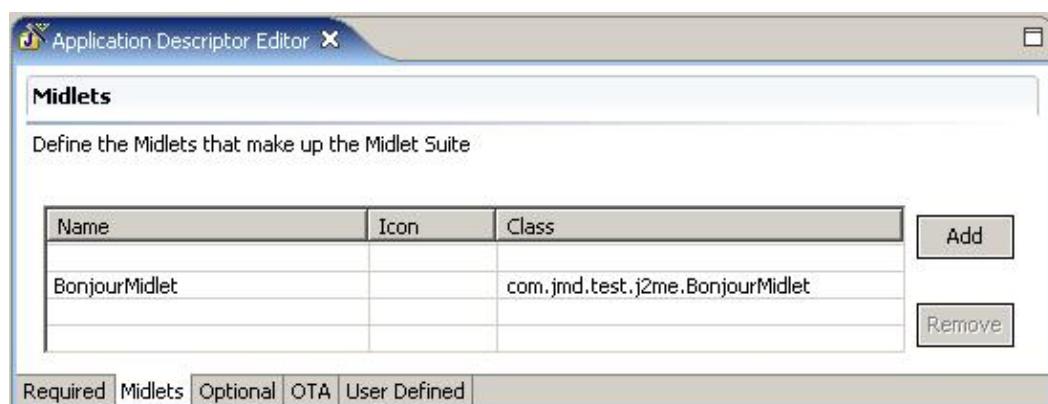
Une application J2ME est composée de deux fichiers :

- Un fichier .jar qui contient l'exécutable
- Un fichier .jad qui contient des informations que l'application

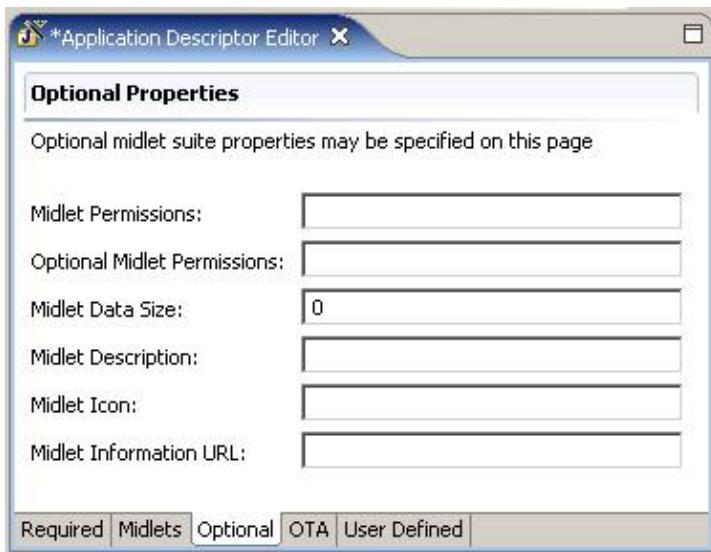
EclipseMe propose un éditeur dédié pour éditer le fichier.jad associé à l'application. Pour ouvrir cet éditeur, il suffit de double-cliquer sur le fichier .jad du projet.



Le premier onglet permet de modifier les propriétés obligatoires.



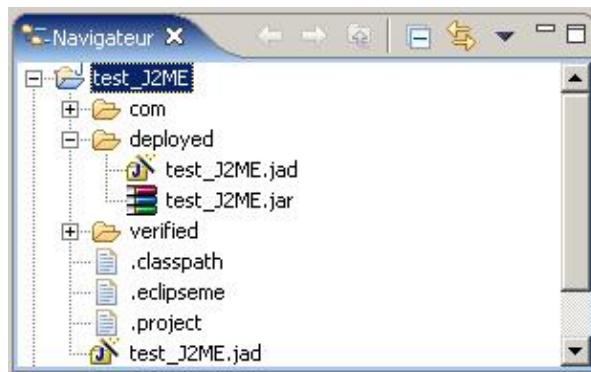
L'onglet Midlets permet de préciser la ou les midlets qui composent l'application.



L'onglet « Optional » permet de préciser des paramètres optionnels.

### 28.1.7. Packager l'application

Pour créer un package de l'application, il faut utiliser l'option « J2ME/Create package » du menu contextuel du projet dans le navigateur.



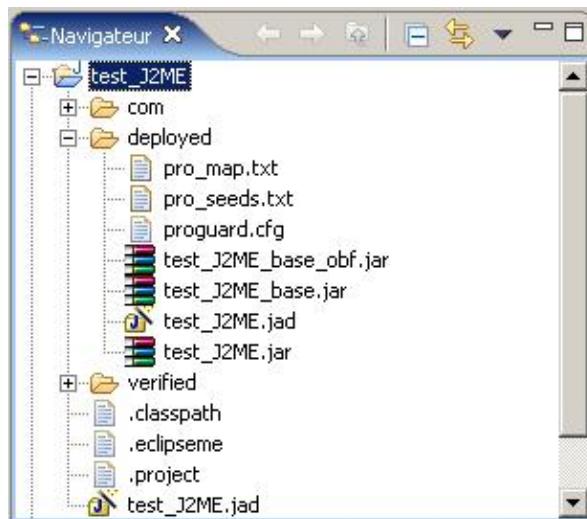
Les fichiers .jad et .jar sont créés dans le répertoire deployed. Le fichier .jar inclus les classes et les bibliothèques contenues dans le répertoire verified, les ressources (images, son, ...) contenues dans le répertoire res et le fichier manifest généré à partir du fichier .jad à la racine du projet.

#### Exemple de fichier META-INF/MANIFEST.MF :

```
Manifest-Version: 1.0
MIDlet-2: BonjourMidlet,,com.jmd.test.j2me.BonjourMidlet
MicroEdition-Configuration: CLDC-1.0
MIDlet-Name: test_J2ME Midlet Suite
MIDlet-Vendor: Midlet Suite Vendor
MIDlet-1: ,
MIDlet-Version: 1.0.0
MicroEdition-Profile: MIDP-1.0
```

Si Proguard est installé et configuré dans le plug-in, il est possible de créer un package obscurci de l'application, il faut utiliser l'option « J2ME/Create obfuscated package » du menu contextuel du projet dans le navigateur.

Plusieurs fichiers sont générés dans le répertoire deployed.

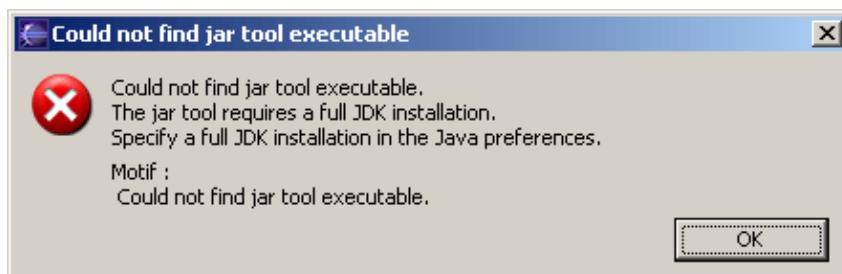


Le fichier `test_J2ME_base.jar` est le fichier `.jar` avant l'opération d'obscurcissement. Le fichier `test_J2ME_base_obf.jar` est le fichier `.jar` après l'opération d'obscurcissement. Le fichier `proguard.cfg` contient les options utilisées par Proguard pour réaliser le package.

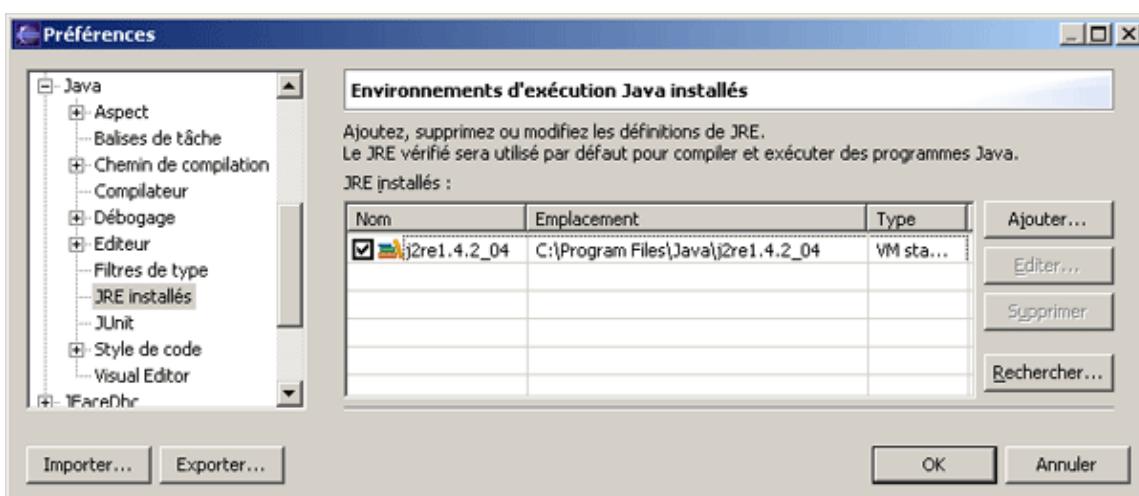
#### Exemple :

```
-libraryjars C:\WTK21\lib\cldcapi10.jar;C:\WTK21\lib\midpapi10.jar
-injars      C:\java\eclipse31\workspace\test_J2ME\deployed\test_J2ME_base.jar
-outjar     C:\java\eclipse31\workspace\test_J2ME\deployed\test_J2ME_base_obf.jar
-printseeds  C:\java\eclipse31\workspace\test_J2ME\deployed\pro_seeds.txt
-printmapping C:\java\eclipse31\workspace\test_J2ME\deployed\pro_map.txt
-dontnote -defaultpackage ''
-keep public class * extends javax.microedition.midlet.MIDlet
```

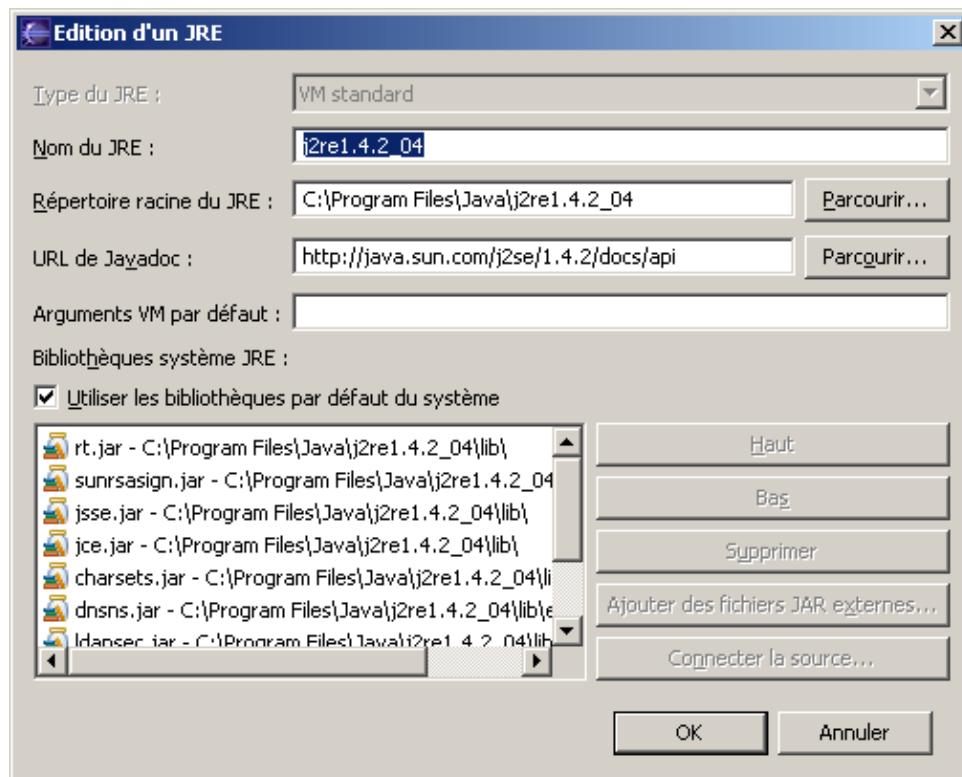
Lors de la réalisation du packaging, il est possible d'obtenir ce message d'erreur.



Dans ce cas, il faut modifier les paramètres du JRE définis dans les Préférences d'Eclipse : par défaut Eclipse utilise un JRE et Proguard nécessite un JDK lors de ces opérations.



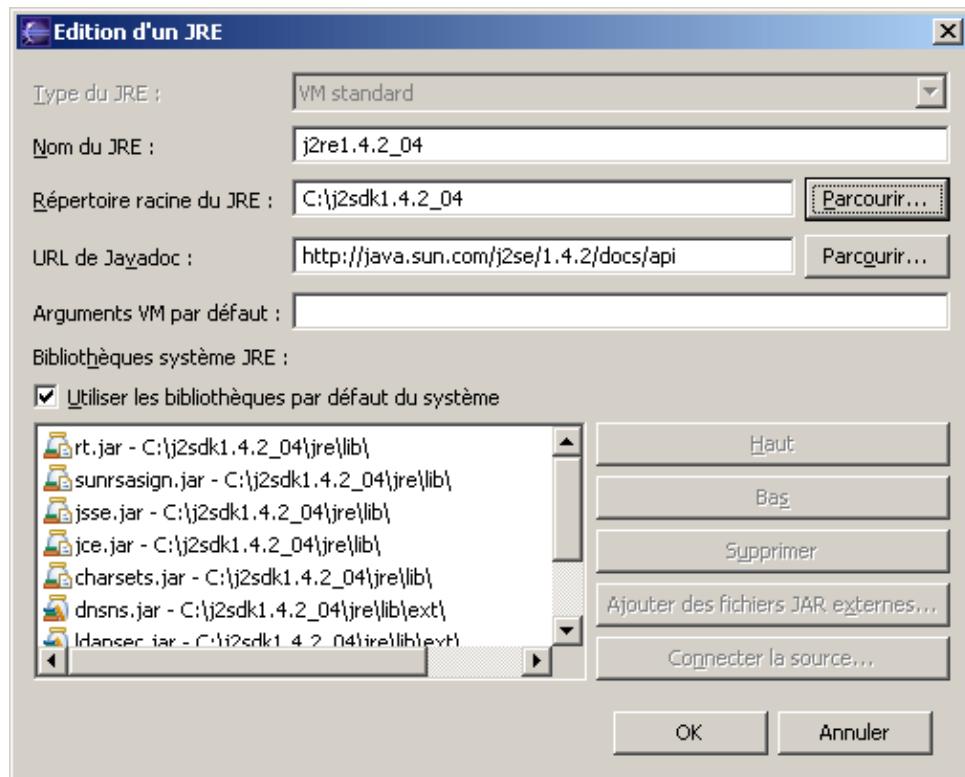
Sélectionnez le JRE et cliquez sur le bouton « Editer »



Cliquez sur le bouton « Parcourir » associé à « Répertoire racine du JRE »



Il suffit de sélectionner le répertoire qui contient le JDK et de cliquer sur le bouton « OK ».



Cliquez sur le bouton « OK ».

Cliquez encore sur le bouton « Ok » pour fermer la fenêtre des « Préférences ».

# Partie 7 : d'autres plug-ins

Cette partie présente des plug-ins tiers qui ne sont pas dédié directement aux développements avec Java.

Eclipse est un outil de développement, écrit en Java, dont un des buts initiaux est de pouvoir développer non seulement en Java mais aussi avec d'autres langages grâce à des plug-ins spécifiques. Il existe déjà plusieurs plug-ins plus ou moins avancés pour réaliser des développements en C/C++ et Cobol développés par le projet Eclipse mais aussi C# ou PHP développés par des tiers.

L'extensibilité d'Eclipse par plug-in lui permet aussi d'être pourvu de plug-in qui n'ai pas pour vocation de manipuler directement du code mais par exemple explorer une base de données ou rédiger des diagrammes UML.

Cette partie comporte les chapitres suivants :

- ◆ Le plug-in CDT pour le développement en C / C++ : présente le plug-in CDT qui est sous projet officiel du projet Eclipse dont le but est faciliter le développement avec les langages C et C++.
- ◆ Le plug-in EclipseUML : présente le plug-in EclipseUML de la société Omondo pour réaliser des diagrammes UML avec Eclipse.
- ◆ Les bases de données et Eclipse : présente plusieurs plug-ins pour réaliser des opérations sur des bases de données.

## 29. Le plug-in CDT pour le développement en C / C++

# Chapitre 29

CDT (C/C++ Development Tools) est un sous projet du projet "Eclipse Tools" dont le but est de fournir un environnement intégré de développement en C /C++ sous la forme de plug-ins pour Eclipse.

Le CDT ajoute une perspective nommée "C/C++" au workbench.

Le CDT ne fournit pas de compilateur : il est nécessaire d'utiliser un compilateur externe. Le seul compilateur actuellement supporté par le CDT est le célèbre compilateur GCC du projet GNU. D'autres outils du projet GNU sont aussi nécessaires tel que make ou GDB (GNU Debugger).

Sous linux, ces outils peuvent être facilement installés en utilisant les packages adéquats selon la distribution utilisée.

Sous Windows, l'utilisation de ces outils peut être mise en oeuvre grâce à l'installation d'un des deux outils suivants :

- Cygwin : c'est un projet de la société Red Hat (<http://www.redhat.com/software/cygwin/>)
- MinGW (Minimalist GNU for Windows) : c'est un projet open source qui a pour but de fournir un ensemble de fichier en-tête et de bibliothèques pour générer des exécutables natifs sous Windows en utilisant des outils du projet GNU. (<http://www.mingw.org/>)

Dans ce chapitre, l'installation et l'utilisation de MinGW avec le CDT sera détaillée pour une utilisation sur une plate-forme Windows.

Bien qu'écrit en Java, le CDT est dépendant de la plate-forme sur laquelle il s'exécute.

	Version utilisée dans cette section
Eclipse	2.1
CDT	1.1
MinGW	3.1

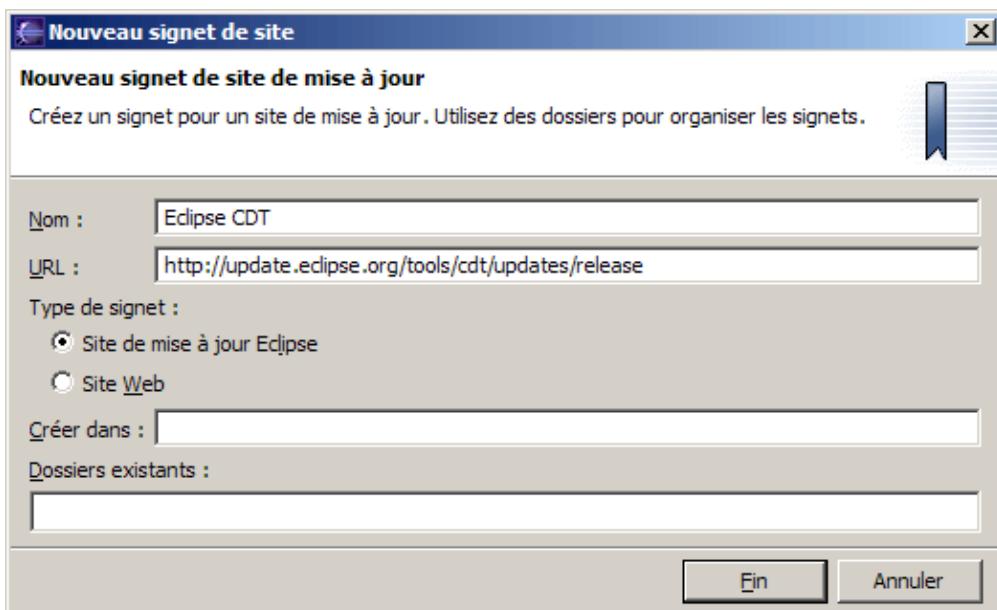
### 29.1. Installation du CDT

Pour installer le CDT, il faut utiliser le mécanisme de mise à jour en utilisant l'option "Mise à jour des logiciels / Gestionnaire des mises à jour" du menu "Aide".

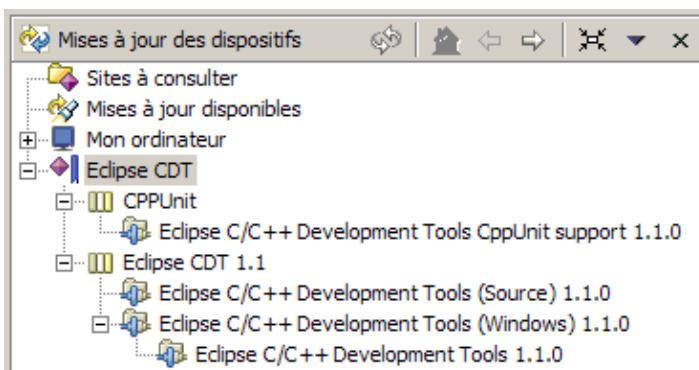


Dans la vue mise à jour des dispositifs, utiliser l'option « Nouveau / Signet du site » du menu contextuel.

Il faut saisir un nom par exemple "Eclipse CDT" et saisir l'url "http://update.eclipse.org/tools/cdt/updates/release".



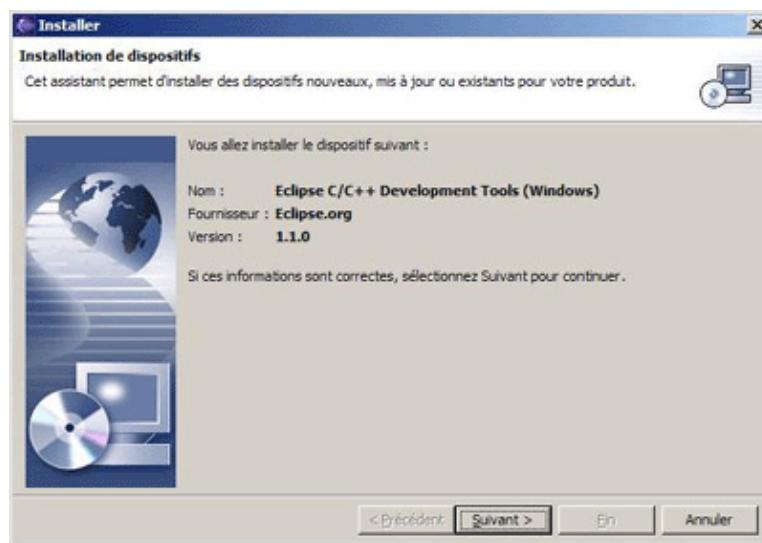
Cliquez sur le bouton "Fin" pour permettre à Eclipse de rechercher les mises à jour. La vue "Mise à jour des dispositifs" affiche celles disponibles.



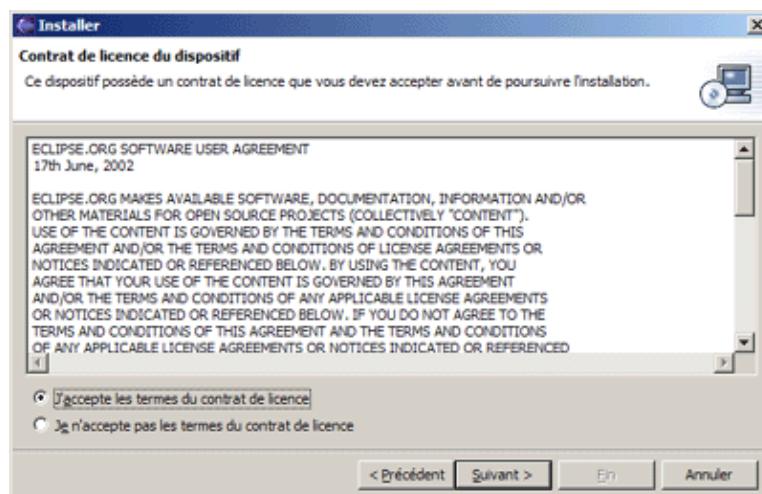
Sélectionnez la mise à jour « Eclipse C/C++ Development Tools (Windows). »



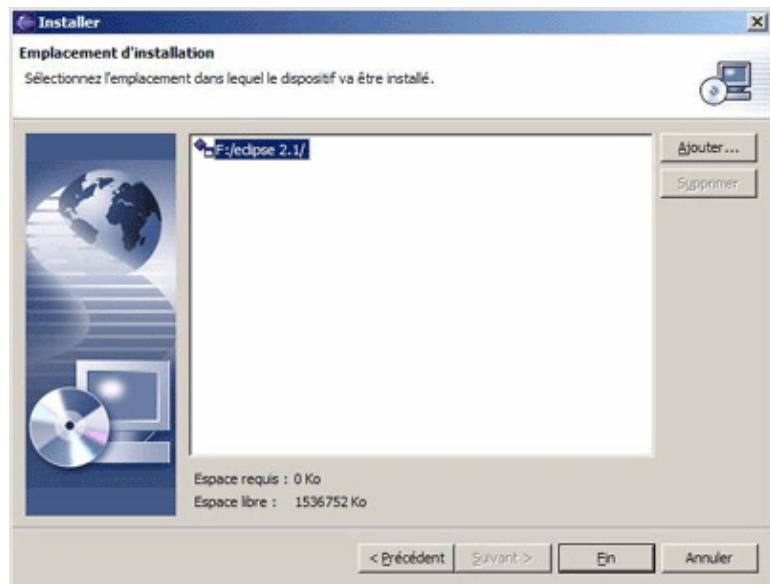
La vue "Aperçu" affiche des informations sur le plug-in. Cliquez sur le bouton « Installer Maintenant »



Cliquez sur le bouton « Suivant »



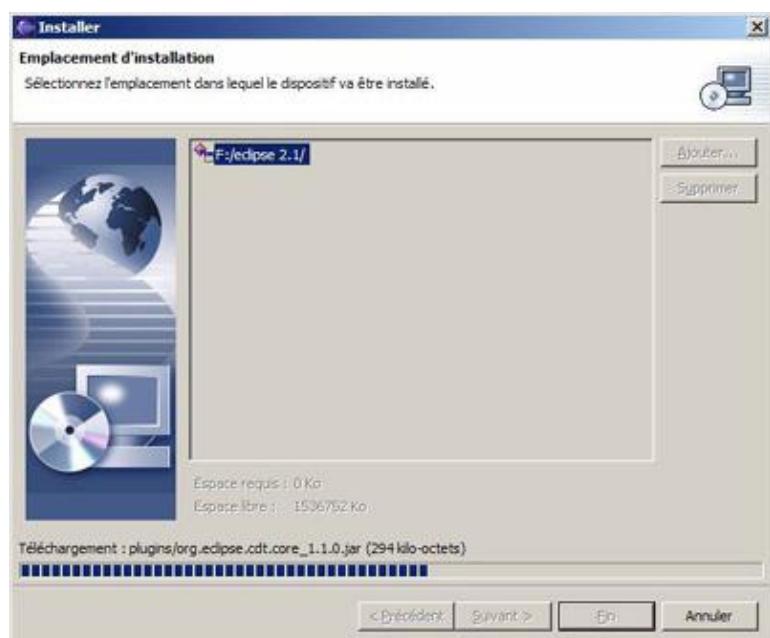
Lisez la licence et si vous acceptez les termes du contrat, cliquez sur le bouton « Suivant »



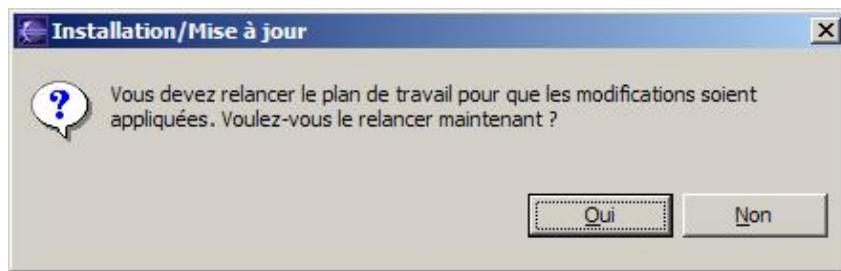
Cliquez sur le bouton « Fin »



Cliquer sur le bouton « Installer ».



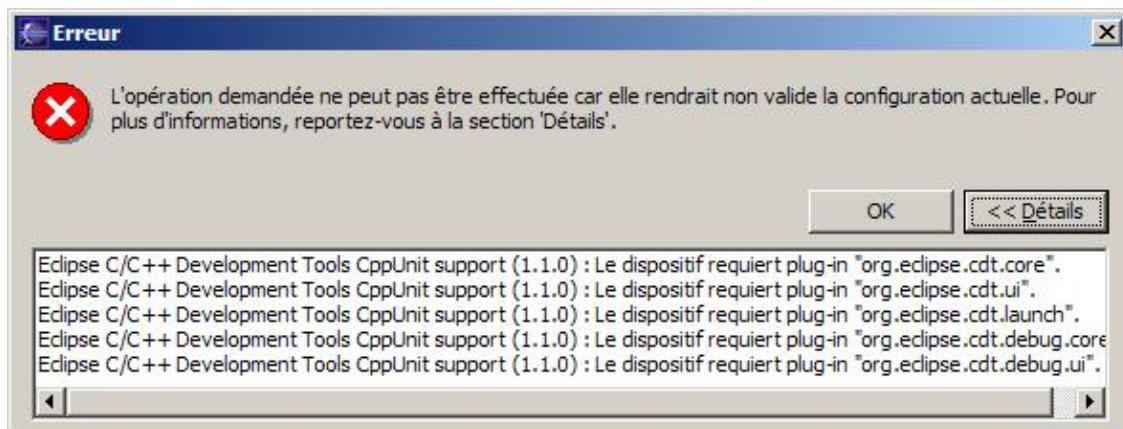
L'installation nécessite un redémarrage de l'application.



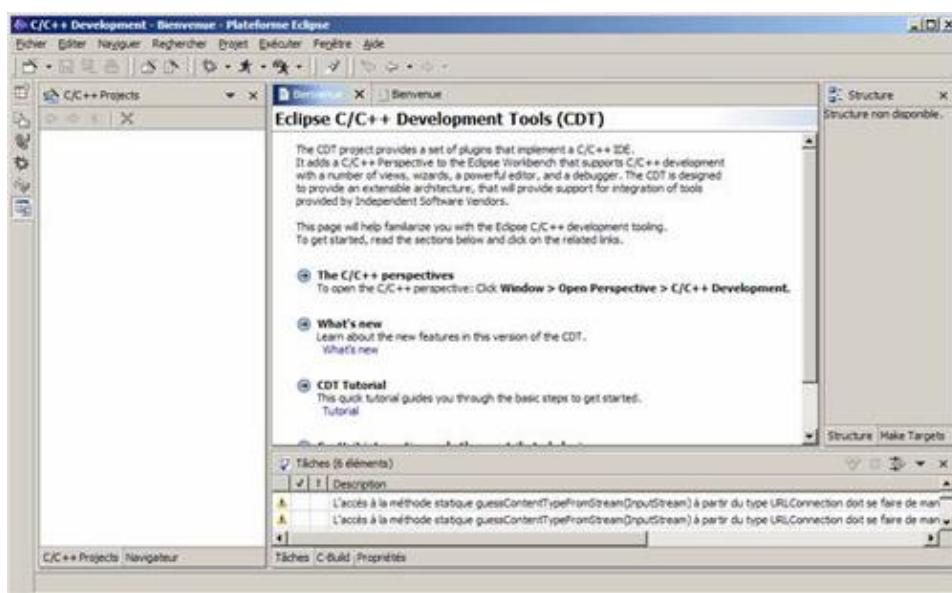
Cliquez sur le bouton « Oui ».

Faites la même opération avec la mise à jour « Eclipse C/C++ development tools CppUnit support 1.1.0. »

Si vous installez cette mise à jour avant la précédente, le message d'erreur suivant est affiché.

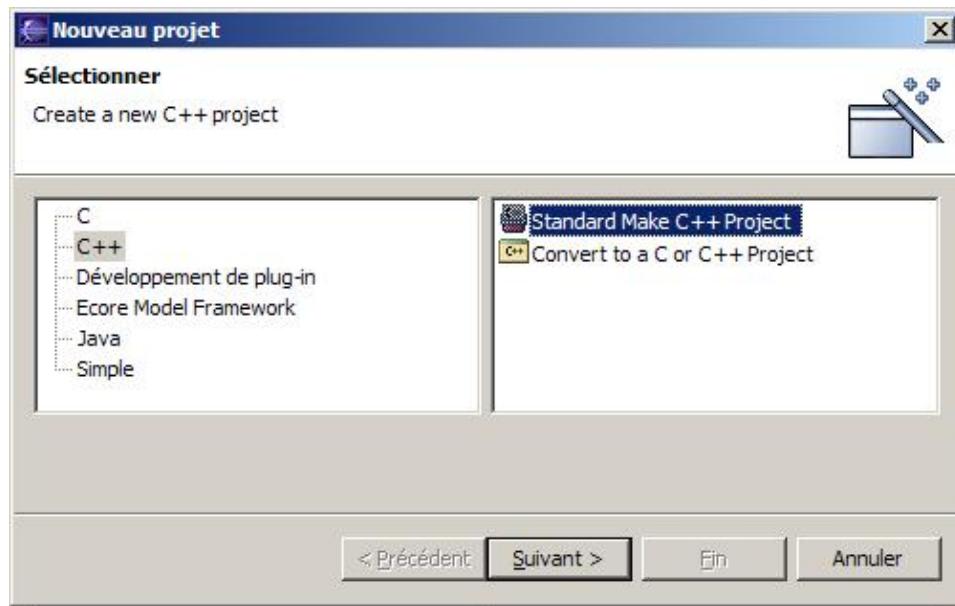


Une fois l'installation terminée, l'éditeur de la prospective ressource affiche des informations sur le CDT.

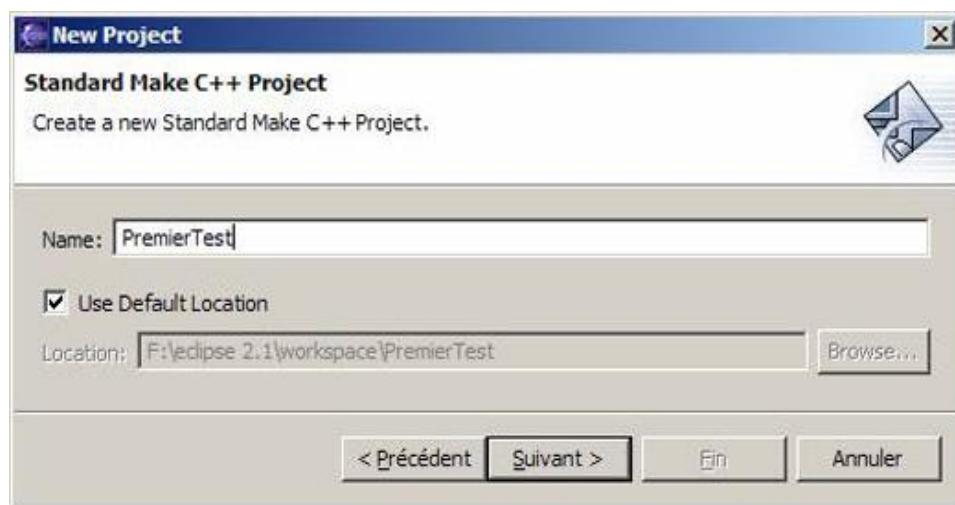


## 29.2. Création d'un premier projet

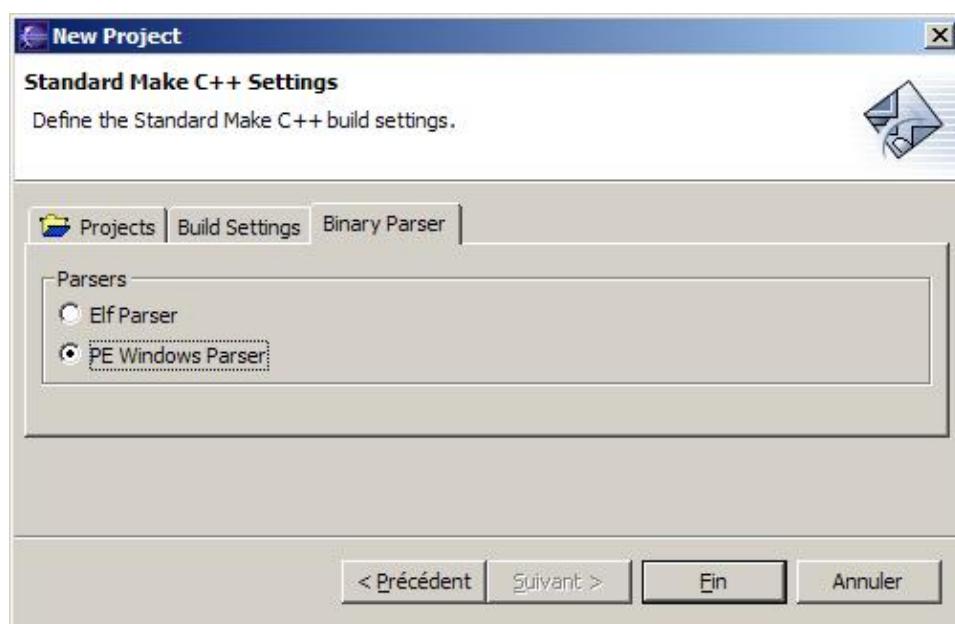
Pour pouvoir utiliser le CDT, il faut tout d'abord créer un nouveau projet.



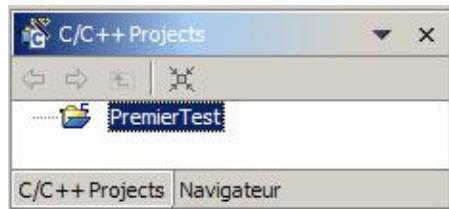
Pour un projet en C++, sélectionnez "C++" dans la liste de gauche puis "Standard Make C++ Project" dans la liste de droite et enfin cliquez sur le bouton "Suivant".



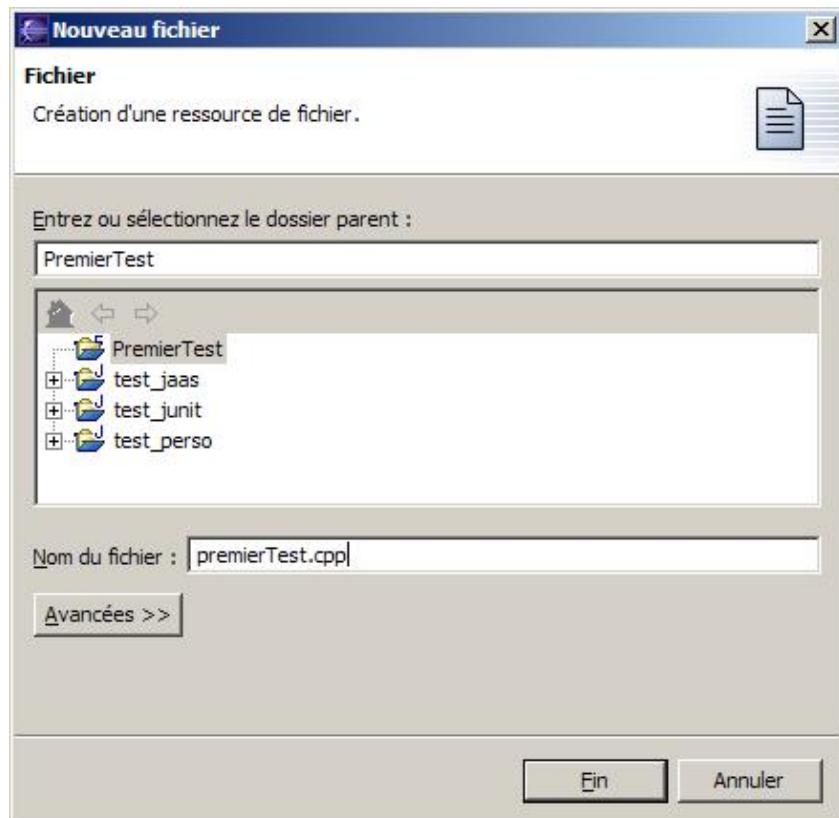
Saisissez le nom du projet et cliquez sur « Suivant ».



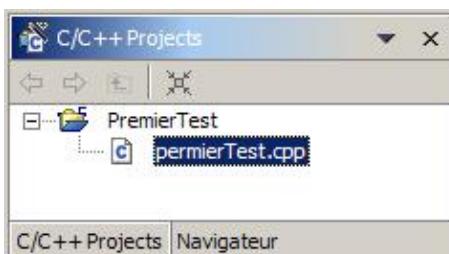
Sous Windows, dans l'onglet « Binary Parser », cliquez sur « PE Windows Parser » puis sur le bouton « Fin ».



Il faut ensuite créer un nouveau fichier dans le projet.



Saisissez le nom du fichier et cliquez sur le bouton « Fin ».

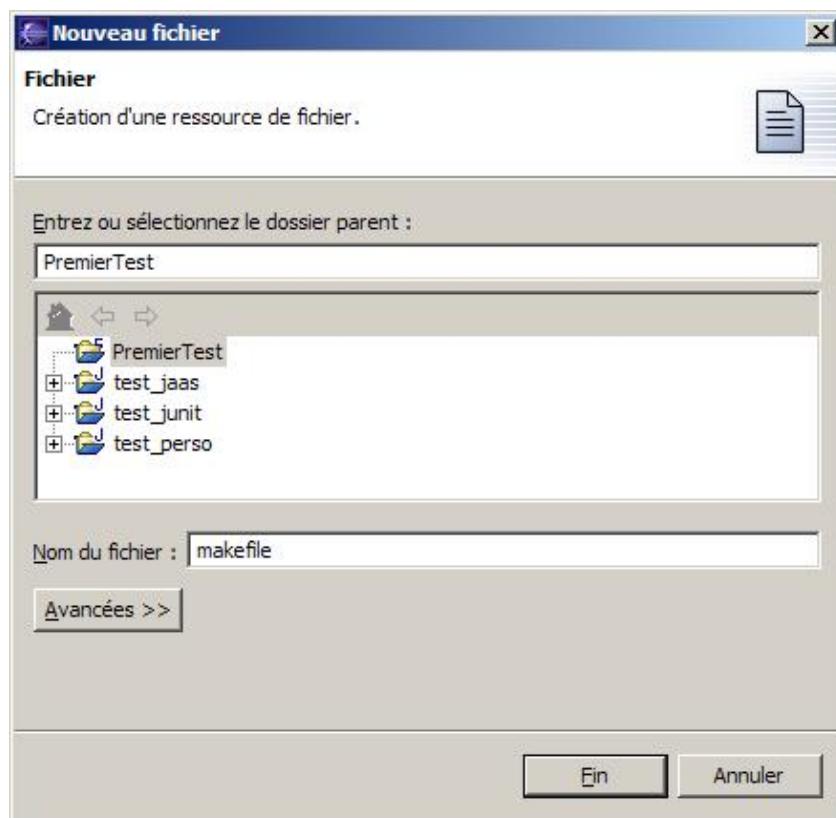


Il suffit de saisir le code de l'application dans l'éditeur :

**Exemple :**

```
#include <iostream>
using namespace std;
int main () {
    cout << "Bonjour" << endl;
    char input = ' ';
    cin >> input;
    exit(0);
}
```

Il faut ensuite créer un fichier nommé makefile pour automatiser la génération de l'exécutable grâce à l'outil make.



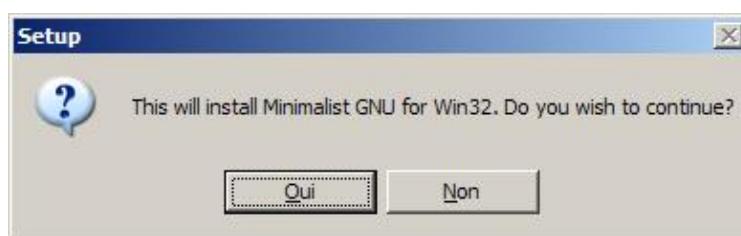
Cliquez sur le bouton « Suivant » puis sur le bouton « Fin ».

### 29.3. Installation de MinGW

Pour pouvoir compiler des fichiers sources avec le CDT, il est nécessaire de disposer d'un compilateur externe. Pour l'instant seul le compilateur GCC du projet GNU est supporté. N'étant pas directement fourni dans Windows, il faut l'installer grâce à un outil tiers. Cette section décrit la procédure d'installation avec l'outil open source Minimalist GNU for Win32 (MinGW).

Il faut downloader les fichiers MSYS-1.0.9.exe et MinGW-3.1.0-1.exe sur le site <http://www.mingw.org/>

Lancez l'exécution de MinGW-3.1.0-1.exe en premier.





L'assistant d'installation se compose de plusieurs pages :

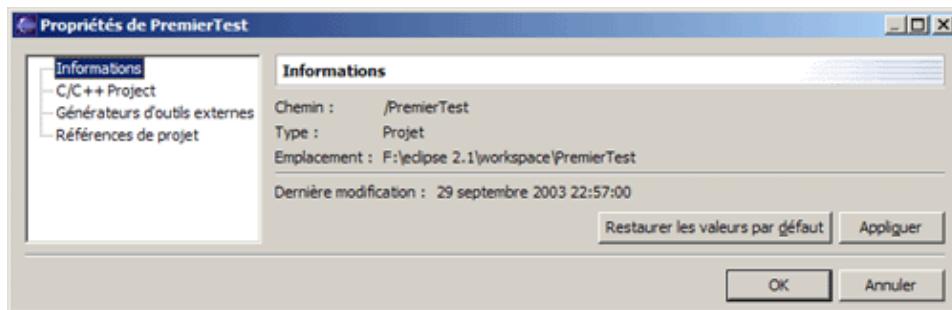
- Sur la page d'accueil, cliquez sur le bouton "Next".
- La page "License Agreement" apparait : lisez la licence et cliquez sur le bouton "Yes" pour accepter les termes de la licence et poursuivre l'installation.
- La page "Information" apparait : lisez le texte et cliquez sur le bouton "Next".
- La page "Select destination directory" apparait : sélectionnez le répertoire où sera installer MinGW et cliquez sur le bouton "Next".
- La page "Ready to install" apparait : cliquez sur le bouton "Install".

Lancez l'exécution de MSYS-1.0.9.exe. Un script permet de configurer l'environnement.

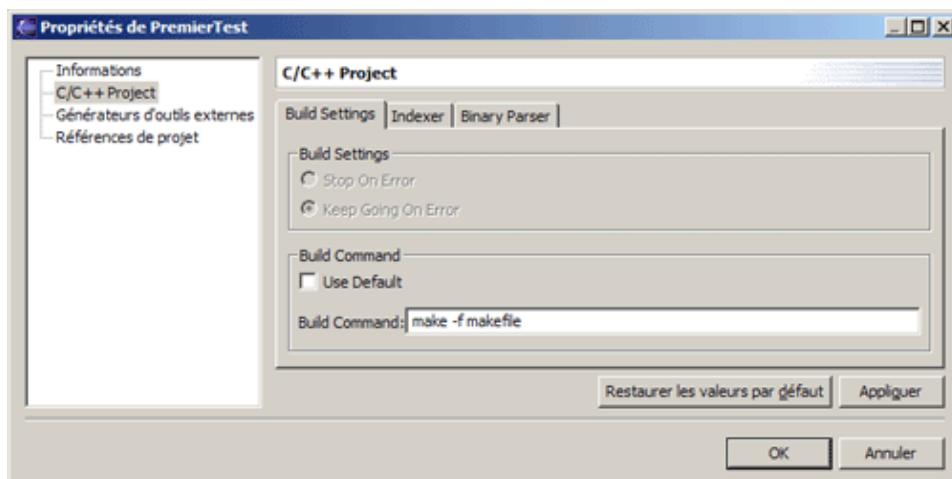
A screenshot of a Windows terminal window titled "C:\WINDOWS\System32\cmd.exe". The window displays a command-line interface. The text shows the execution of a script named "pi.sh" which performs a post-installation process. It includes steps to normalize between MinGW installations, ask if MinGW is installed, ask for the MinGW installation path, create fstab mount bindings, and list available scripts. It also mentions that "make.exe" is missing from the path. The script ends with a "pause" command.

## 29.4. Première configuration et exécution

Il faut modifier les propriétés du projet pour utiliser les options de la commande make fournie avec MSYS.



Il faut décocher l'option "use default" et saisir "make -f makefile" dans le champ "build command".



Il faut ajouter les répertoire suivants dans la variable d'environnement path : C:\msys\1.0\bin;C:\MinGW\bin;

Attention : après la modification, il faut relancer Eclipse si celui ci était en cours d'exécution.

Si les outils make et gcc ne sont pas trouvés, alors le message suivant est afficher dans la vue « C-Build »

Exemple :
Build Error (Exec error:Launching failed)

Attention : il est important que le bon programme makefile soit le premier trouvé dans le classpath. Il est par exemple possible d'avoir des problèmes avec l'outil Delphi de Borland qui ajoute dans le classpath un répertoire qui contient un programme Makefile.

A screenshot of the Eclipse C-Build terminal window titled 'C-Build [PremierTest]'. It displays the following text:

```
make -k clean all
MAKE Version 5.2 Copyright (c) 1987, 1998 Inprise Corp.
Incorrect command line argument: -k

Syntax: MAKE [options ...] target[s]
        -B                  Builds all targets regardless of dependency
        dates
        -Dsymbol[=string]   Defines symbol [equal to string]
        -Idirectory        Names an include directory
```

The window has tabs at the bottom: 'Tâches', 'C-Build' (which is selected), and 'Propriétés'.

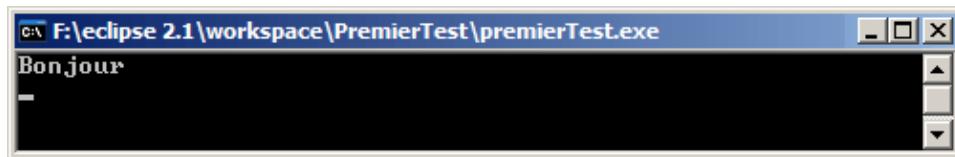
Dans ce cas, il suffit de déplacer ou de supprimer la référence sur le répertoire C:\Program Files\Borland\Delphi7\Bin; dans la variable classpath. (il faut le remettre en cas de suppression pour une utilisation correcte de Delphi).

Si la génération se passe bien, la vue C-Build affiche les étapes de la génération.

Exemple :

```
make -f makefile
gcc -c premierTest.cpp
gcc -o premierTest premierTest.o -L C:/MinGW/lib/gcc-lib/mingw32/3.2.3/ -lstdc++
```

Lancer l'exécution, il suffit de sélectionner le fichier premierTest.exe et d'utiliser l'option « Ouvrir » du menu contextuel.



## 29.5. Utilisation du CDT

L'environnement de développement proposé par le CDT étant un cours de développement, il n'est pas encore aussi complet que l'environnement proposé par le JDT pour Java.

La version 1.1. du CDT propose cependant les fonctionnalités suivantes :

- coloration syntaxique dans l'éditeur de code
- navigation dans un code source grâce à la vue outline
- l'éditeur possède un assistant de code utilisant des modèles (templates)
- utilisation de l'historique locale pour retrouver en local un certain nombre de versions locales du code
- ...

## 30. Le plug-in EclipseUML

# Chapitre 30

La société Omondo propose EclipseUML qui est un plug-in permettant de réaliser des diagrammes UML dans Eclipse.

EclipseUML propose une version gratuite avec le support des 11 diagrammes d'UML 2.0.

	Version utilisée dans cette section
Eclipse	3.0.1
J2SE	1.4.2_03
EclipseUML	2.0.0

EclipseUML nécessite plusieurs plug-ins pour son fonctionnement avec Eclipse 3.0 :

- GEF (Graphical Editor Framework) 3.0.1
- EMF (Eclipse Modeling Framework) 2.0.1
- UML2 1.0.1

### 30.1. Installation

Il faut télécharger le fichier `eclipseUML_E301_freeEdition_2.0.0.beta.20041026.jar` sur le site d'Omondo : <http://www.omondo.com/download/free/index.html>

Ce fichier .jar contient un exécutable qui va installer EclipseUML et les différents plug-ins tiers nécessaires (GEF, EMF et UML2).

Si Eclipse est en cours d'exécution, il faut le quitter avant de lancer l'installation.

Pour lancer l'installation, il suffit de lancer la commande ci dessous dans une console DOS

```
java -jar eclipseUML_E301_freeEdition_2.0.0.beta.20041026.jar
```

Un assistant guide l'utilisateur dans les différentes étapes de l'installation :

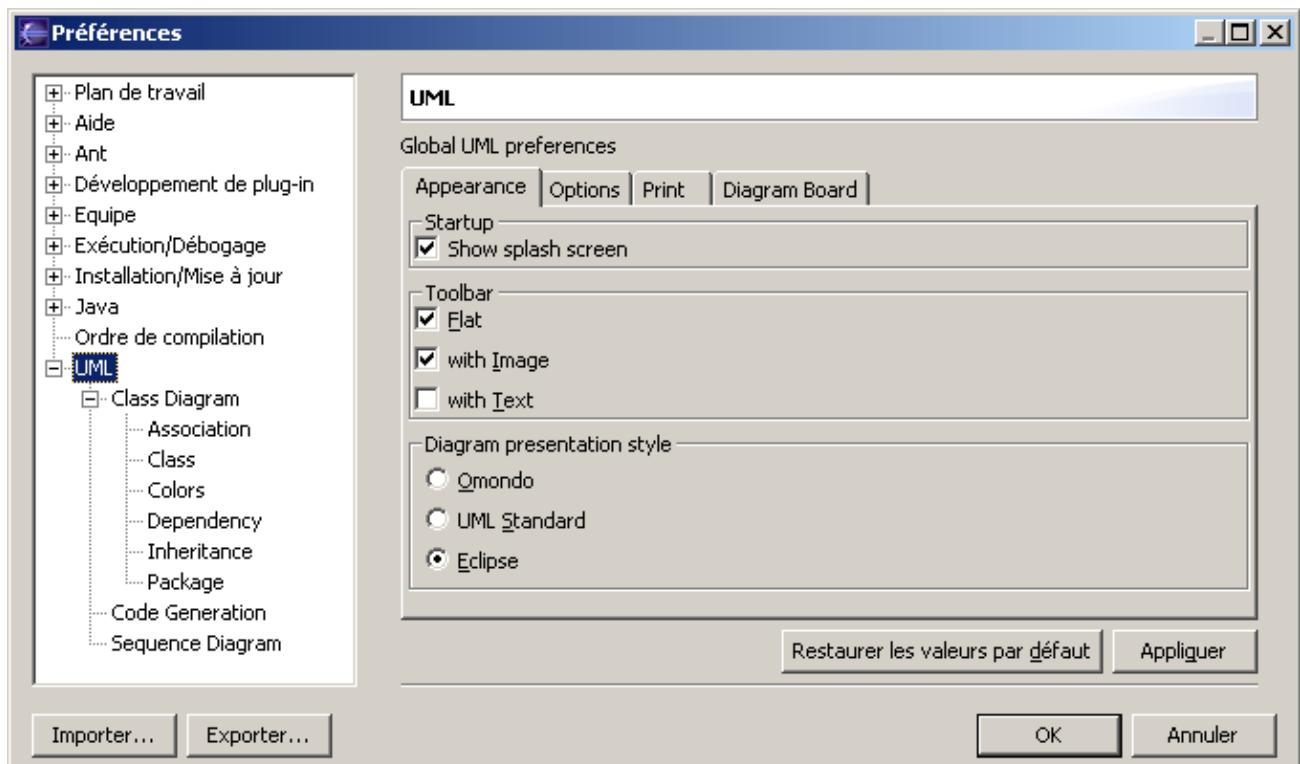
- l'écran d'accueil permet de sélectionner la langue de l'assistant : cliquez sur le bouton « Ok »
- l'écran « Introduction » vous souhaite la bienvenue : cliquez sur le bouton « Suivant »
- l'écran « Informations » affiche des informations à lire : cliquez sur le bouton « Suivant »
- l'écran « License » affiche la licence d'utilisation : lisez la licence et si vous l'acceptez, cliquez sur le bouton « J'accepte les termes de cet accord de licence » puis sur le bouton « Suivant »
- l'écran « Configuration » permet de sélectionner le répertoire d'installation d'Eclipse. Sélectionnez le

- répertoire d'installation d'Eclipse au besoin et cliquez sur le bouton « Suivant ».
- l'écran « Configuration » suivant permet de sélectionner les packages à installer : cliquez sur le bouton « Suivant »
  - l'écran « Installation » affiche la progression de la copie des fichiers : cliquez sur le bouton « Suivant »
  - l'écran « Install Complete » : cliquez sur le bouton « Quitter ».

Une fois l'installation terminée, il suffit de lancer Eclipse.

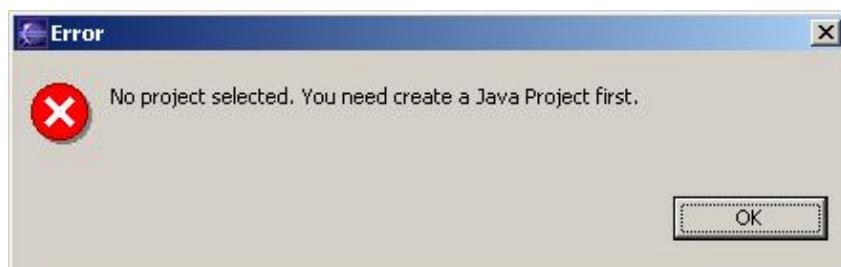
## 30.2. Les préférences

Les nombreux paramètres d'EclipseUML peuvent être réglés dans les préférences (menu Fenêtre/Préférences)

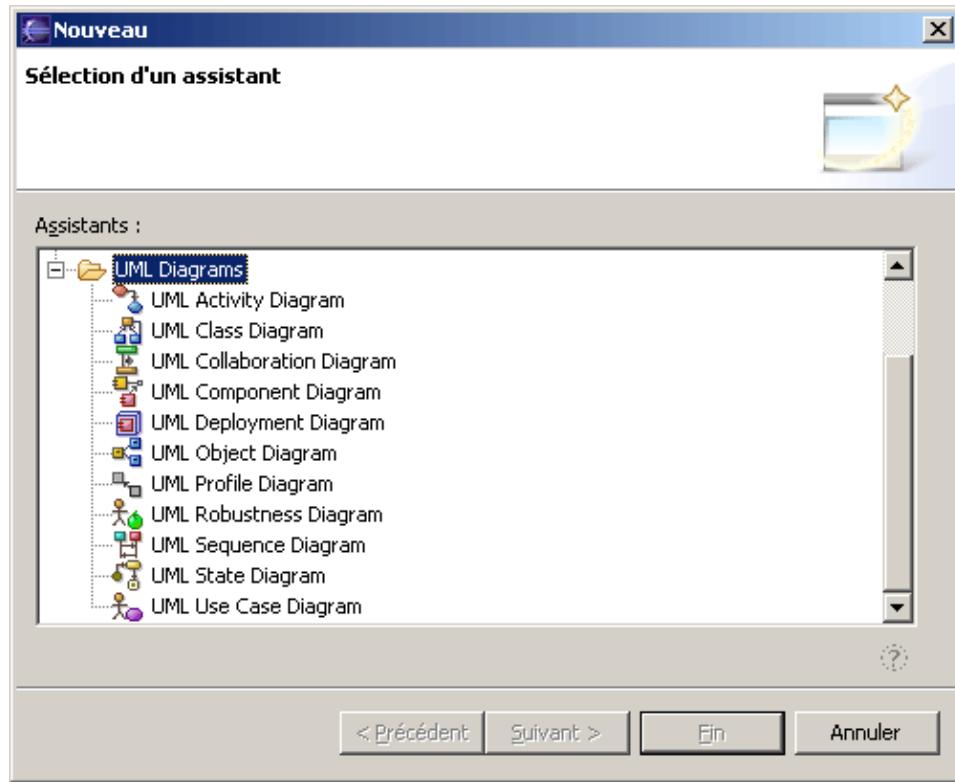


## 30.3. Mise en oeuvre du plug-in

L'utilisation des diagrammes UML nécessitent d'utiliser un projet Java. Si aucun projet Java n'est sélectionné, avant l'appel à l'assistant de création d'entités, ce dernier affichera un message d'erreur lors de son utilisation.



Pour créer un nouveau diagramme, il faut sélectionner un projet et utiliser l'option « Nouveau / Autres » du menu « Fichier ».

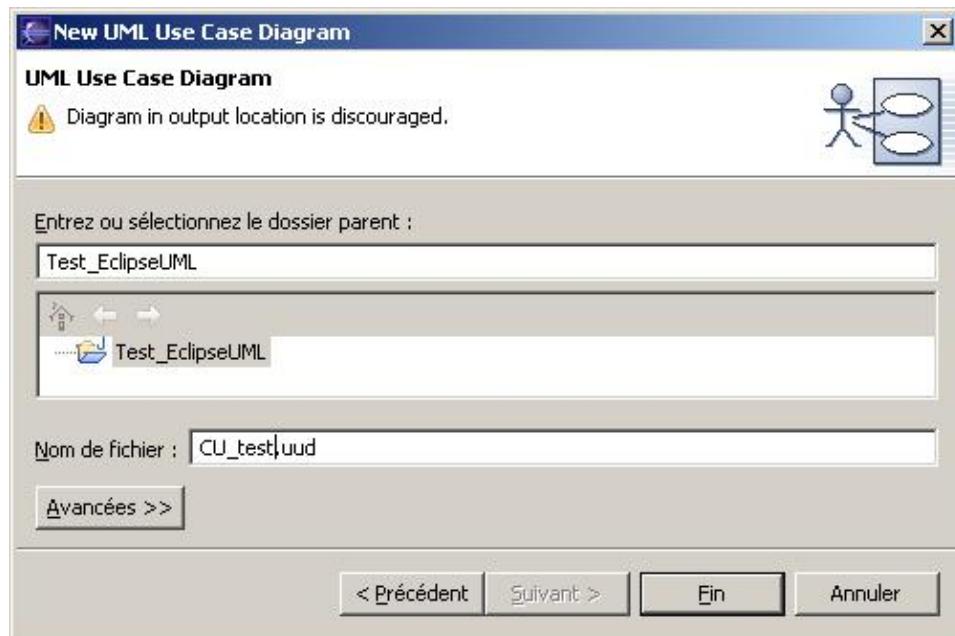


Il est possible d'exporter chaque diagramme dans plusieurs formats en utilisant l'option « Export » de l'éditeur de diagrammes. Les formats utilisables sont : SVG, JPEG et Gif.

Chacun des éditeurs proposent des fonctionnalités de zoom via plusieurs options du menu contextuel.

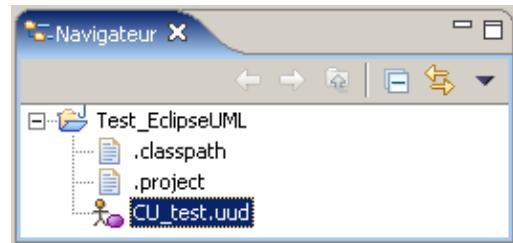
### 30.3.1. Création d'un diagramme de cas d'utilisation

Il faut créer une nouvelle entité de type « UML Diagrams / UML Use Case Diagram» et cliquer sur le bouton « Suivant »

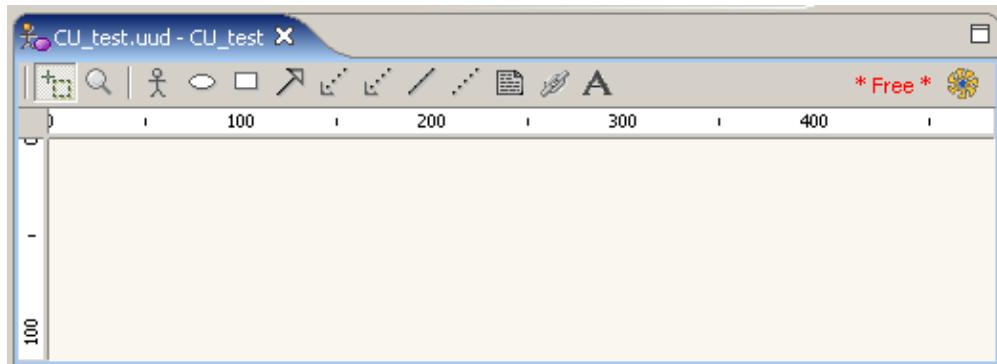


Saisissez le nom du fichier et cliquez sur le bouton « Fin ».

Le fichier est créé par l'assistant



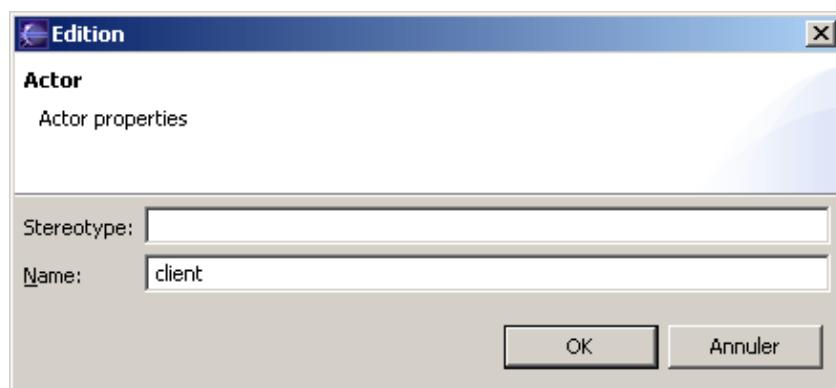
et une vue dédiée à l'édition du diagramme ouvre le fichier



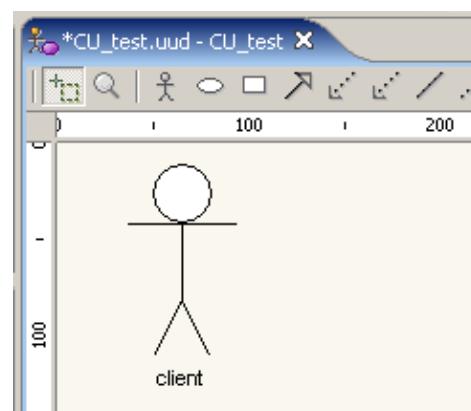
Il suffit alors de composer son diagramme.

Pour ajouter un acteur, il faut cliquer sur le bouton puis cliquer sur la surface de travail de la vue.

Une boîte de dialogue permet de saisir le stéréotype et le nom du nouvel acteur.



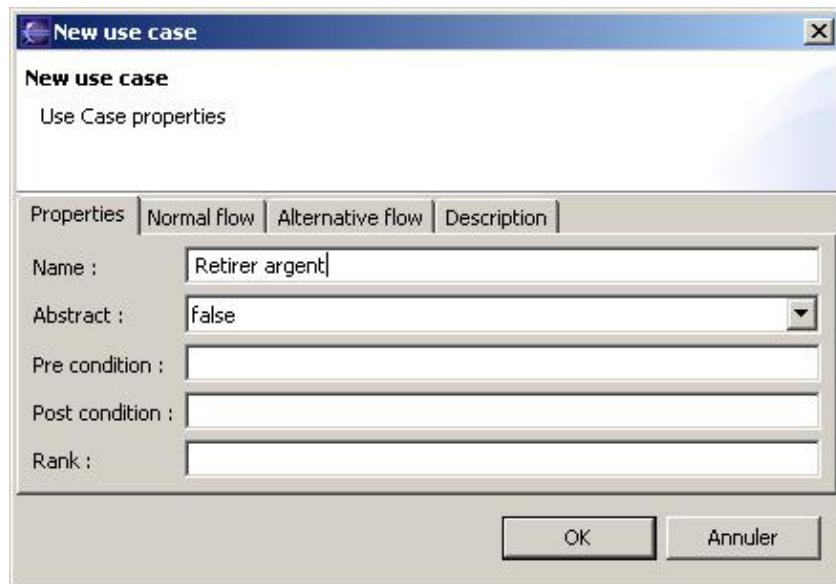
Le diagramme est enrichi avec le nouvel acteur.



Il est possible de répéter l'opération d'ajout d'un nouvel acteur en cliquant sur la surface tant que le bouton est activé.

La création d'un cas d'utilisation est similaire en utilisant le bouton .

Une boîte de dialogue permet de saisir les informations concernant le nouveau cas d'utilisation.



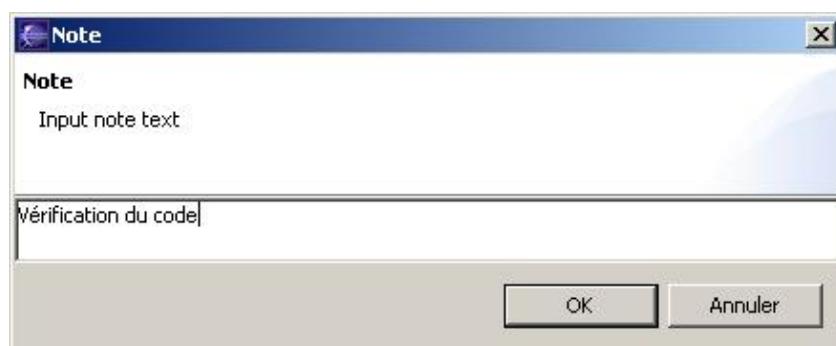
Il faut saisir les informations et cliquer sur le bouton « OK ».



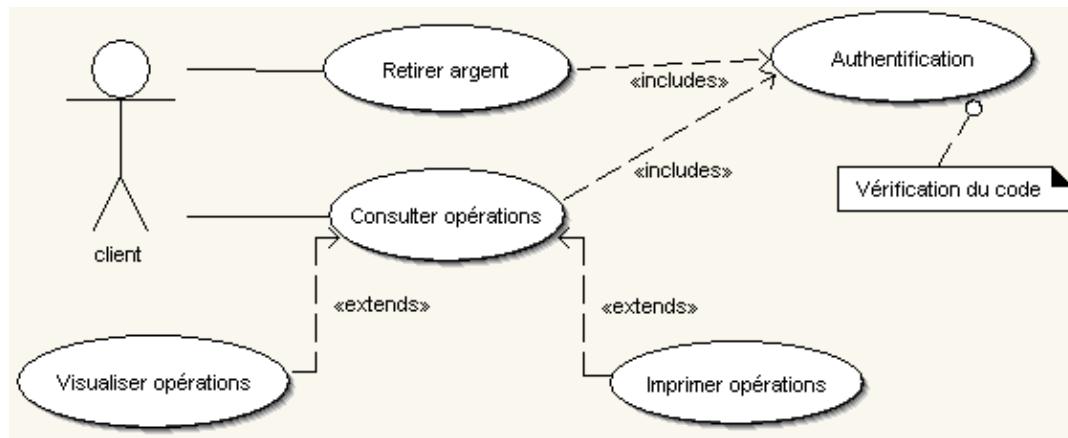
Pour ajouter des associations, il suffit de cliquer sur l'association désirée dans la barre d'outils, puis de cliquer sur la première entité (celle ci change de couleur au passage de la souris) puis sur la seconde (celle ci change aussi de couleur au passage de la souris).

Pour ajouter un commentaire, il suffit de cliquer sur le bouton .

Une boîte de dialogue permet de saisir le commentaire.

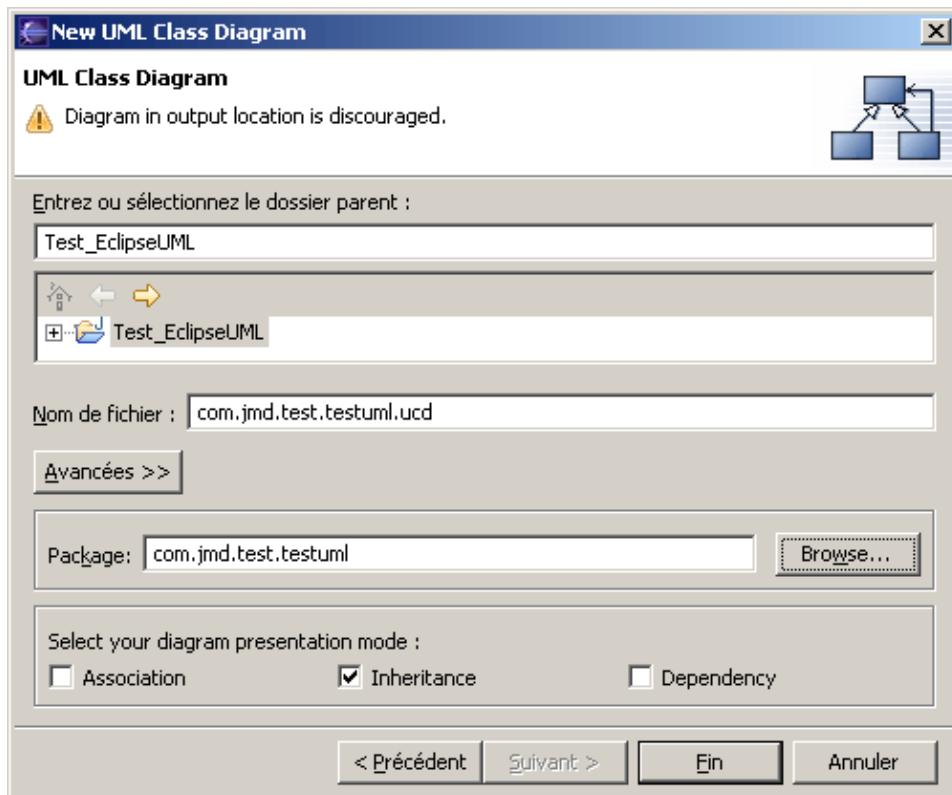


Voici un exemple de diagramme.

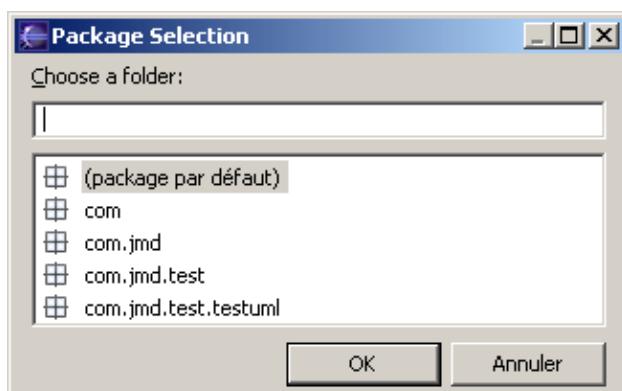


### 30.3.2. Création d'un diagramme de classe

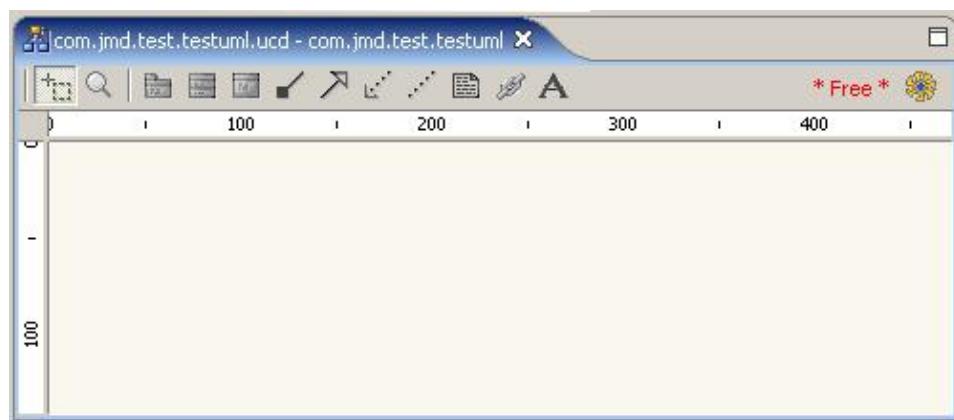
Pour créer un diagramme de classe, il faut sélectionner un package et créer une nouvelle entité de type « UML Diagrams/UML Class Diagram ».



Le bouton « Browse » permet de sélectionner le package concerné

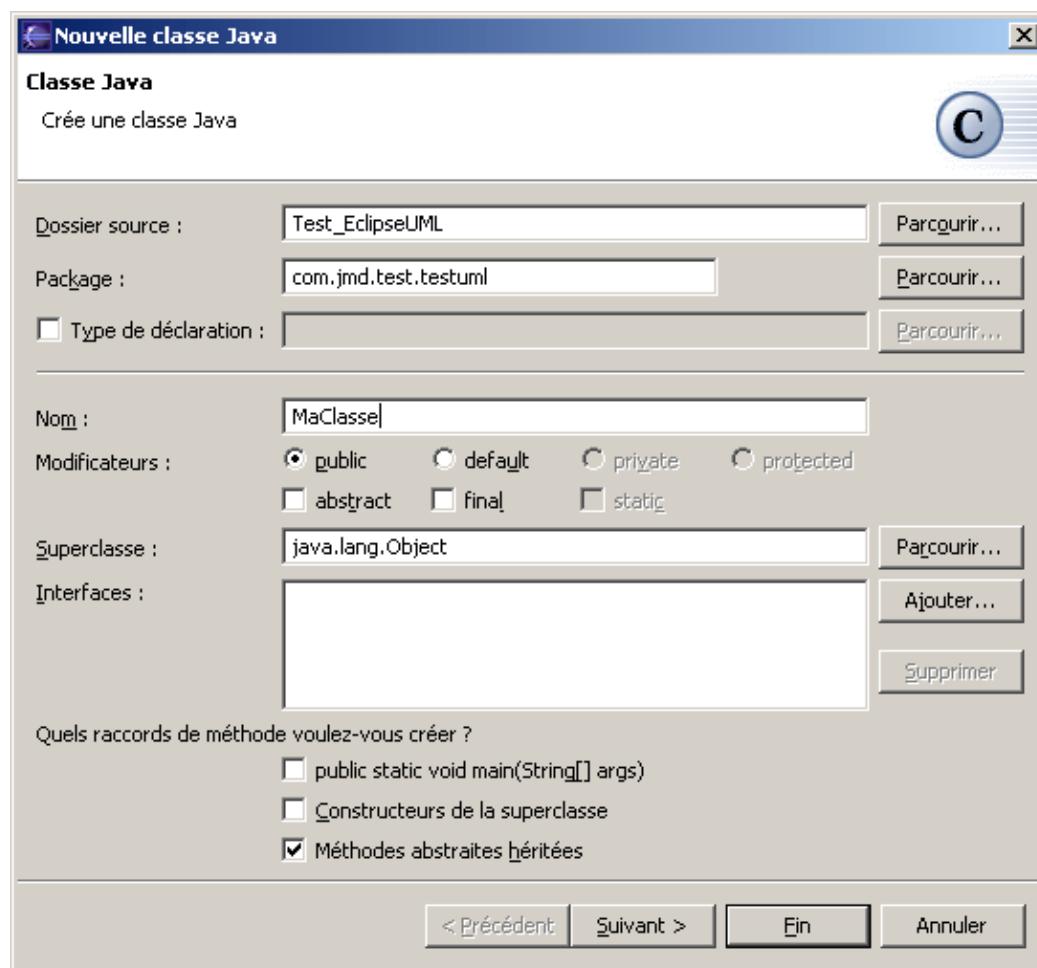


Saisissez le nom du fichier qui sa contenir le diagramme et cliquez sur le bouton « Fin » pour créer le fichier et ouvrir l'éditeur

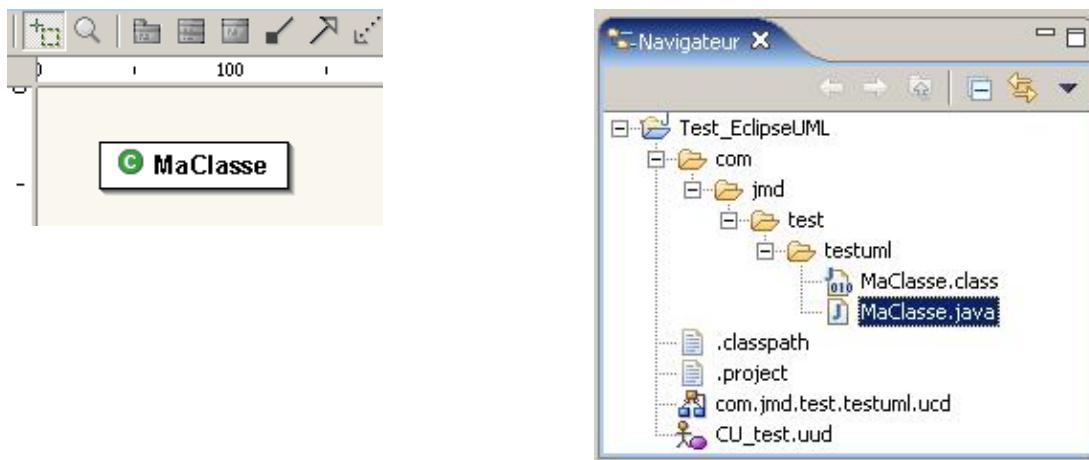


Pour créer une nouvelle classe, il suffit de cliquer sur le bouton puis de cliquer sur la surface de travail de l'éditeur.

L'assistant de création de classe s'ouvre.



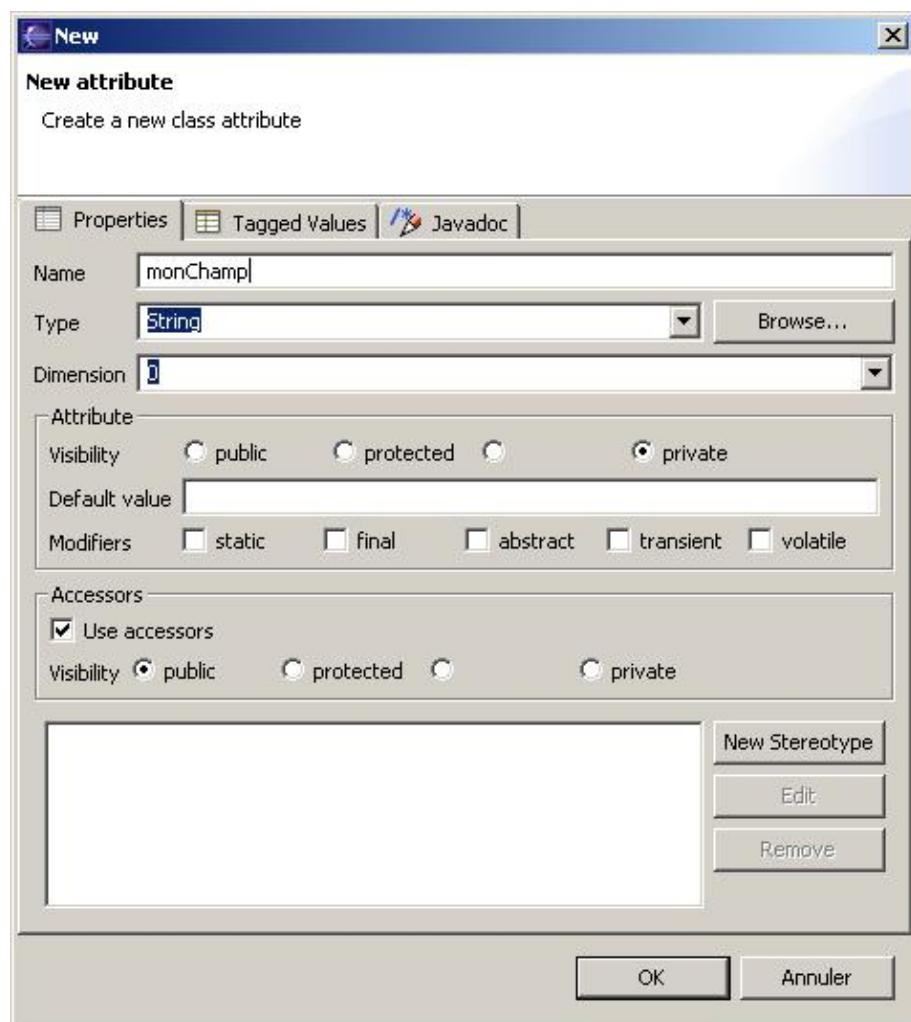
Le bouton suivant permet d'ajouter des stéréotypes. Une fois la saisie terminée, il suffit de cliquer sur le bouton « Fin ». La nouvelle classe est alors ajoutée dans le diagramme mais aussi dans l'espace de travail.



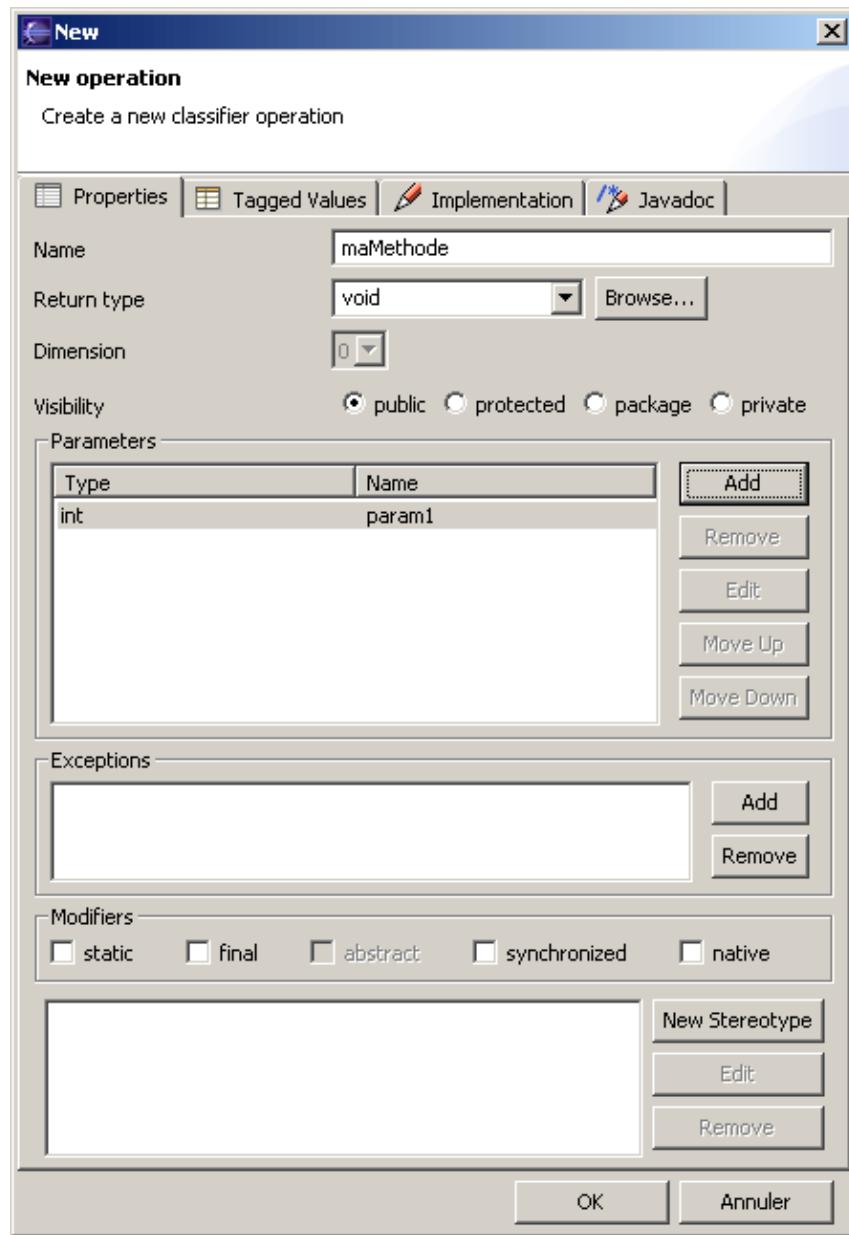
L'option « New » du menu contextuel de la classe dans l'éditeur permet d'ajouter des membres à la classe.

Une boîte de dialogue permet alors la saisie des informations sur le nouveau membre.

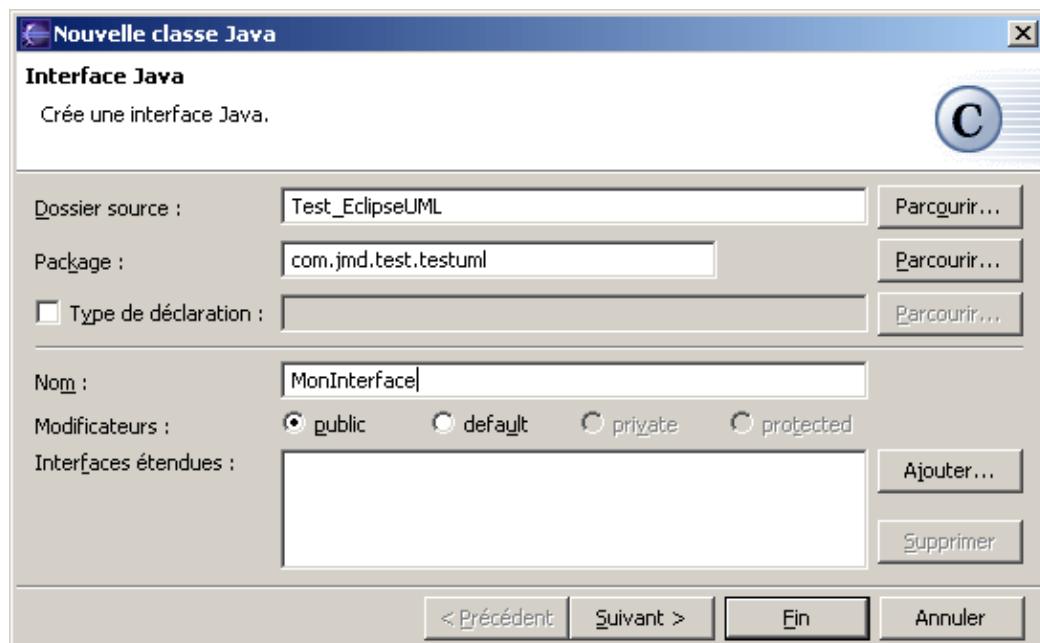
Par exemple pour ajouter un nouveau champ.



Par exemple, pour ajouter une nouvelle méthode

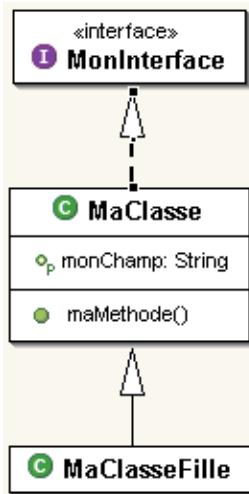


Pour ajouter une nouvelle interface, il suffit de cliquer sur le bouton puis sur la surface de travail de l'éditeur. Une boîte de dialogue permet de saisir les informations de la nouvelle interface.



La création d'une association de généralisation entre deux classes/interfaces se fait en cliquant sur le bouton , puis sur la classe fille et enfin sur la classe mère.

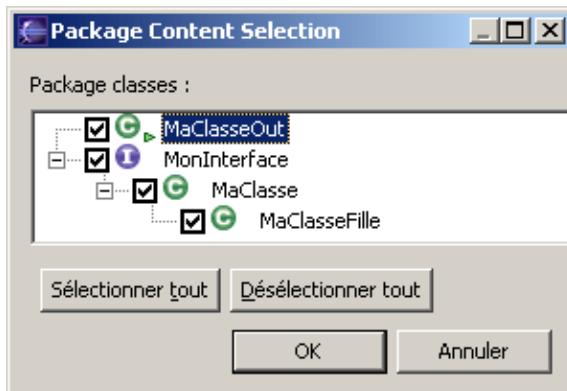
La relation générée (héritage ou implémentation) se fait automatiquement en fonction du type des deux entités concernées.



Pour mettre à jour le code source en fonction des relations ajoutés, il faut utiliser l'option « Model synchronize » du menu contextuel de la classe ou interface dans l'éditeur.

Les mises à jour sont bi-directionnelles : dans le code et sur le diagramme. Il faut donc faire attention notamment à la suppression d'une classe du diagramme qui la supprime aussi dans le package. Si le besoin est simplement de ne plus afficher la classe dans le diagramme, il faut simplement la masquer en utilisant l'option « Hide » du menu contextuel.

Pour faire réapparaître un élément masqué, il faut utiliser l'option « Package elements ... »



Il suffit de sélectionner l'élément et de cliquer sur le bouton « OK ».

## 31. Les bases de données et Eclipse

# Chapitre 31

Eclipse ne propose pas par défaut de plug-in capable de gérer ou de manipuler une base de données, mais il existe de nombreux plug-in permettant d'accomplir certaines de ces tâches. Ce chapitre va présenter quelques uns d'entre eux :

- Quantum
- JFaceDBC
- DBEdit
- Clay Database Modelling

### 31.1. Quantum

Quantum est un plug-in qui permet l'exécution de requêtes sur une base de données. Il affiche une vue arborescente des bases de données et propose plusieurs assistants pour effectuer des mises à jour sur les données.

Les bases de données supportées sont : Postgres, MySQL, Adabas-D, Oracle, DB2, DB2 for AS400.

Le site officiel est <http://quantum.sourceforge.net/>

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Quantum	2.2.2.

La version utilisée dans cette section nécessite obligatoirement un JDK 1.4 pour fonctionner.

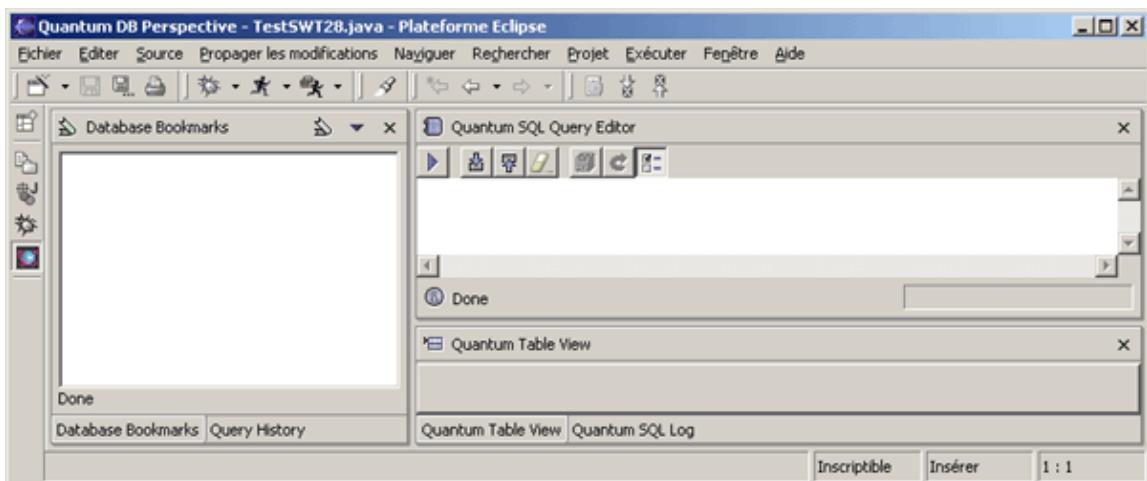
#### 31.1.1. Installation et configuration

Après avoir téléchargé le fichier quantum222.zip, il faut le décompresser dans le répertoire qui contient le répertoire d'Eclipse (attention : le chemin des fichiers de l'archive contient déjà le répertoire Eclipse).

Pour utiliser le plug-in, il faut ouvrir la perspective nommée "Quantum DB".

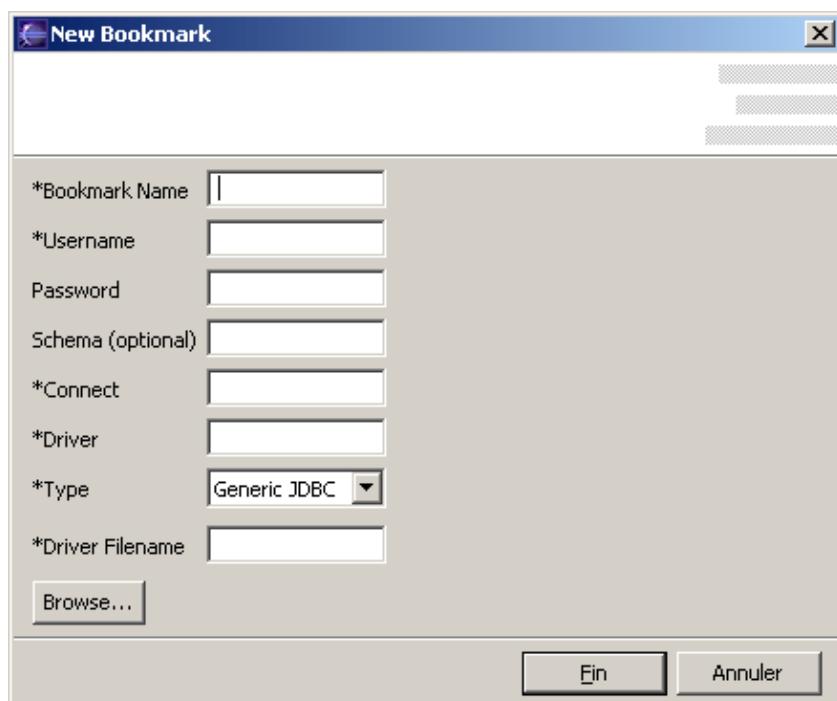


Quantum propose plusieurs vues :



Pour pouvoir utiliser une base de données, il faut l'enregistrer dans le bookmark. Pour cela, dans la vue « Database Bookmarks », cliquez sur le bouton ou sélectionnez l'option « New Bookmark » du menu contextuel.

Une boîte de dialogue permet de saisir les informations concernant la base de données.



L'exemple ci dessous contient les paramètres pour une base de données MySQL nommée "Test" avec le pilote MySQL Connector-J 3.0 (<http://www.mysql.com/products/connector-j/>).

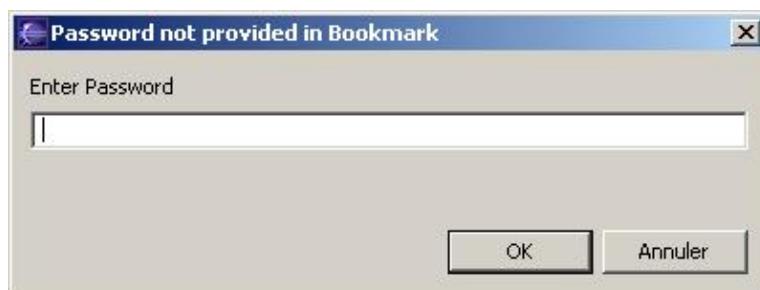
- « Bookmark name » : est le nom du bookmark (cette zone est libre)
- « Username » : est le nom de l'utilisateur pour la connexion à la base de données
- « Password » : est le mot de passe de cet utilisateur
- « Connect » : est la chaîne de connexion dépendante du pilote utilisé (jdbc:mysql://localhost/test dans l'exemple)
- « Driver » : est le nom de la classe du pilote JDBC (com.mysql.jdbc.Driver dans l'exemple)
- « Type : » : est le type de base de données utilisée
- « Driver Name » : est le fichier jar qui contient le pilote JDBC (D:\java\mysql-connector-java-3.0.11-stable\mysql-connector-java-3.0.11-stable-bin.jar dans l'exemple)

Un fois tous les paramètres saisis, il suffit de cliquer sur le bouton « Fin ».

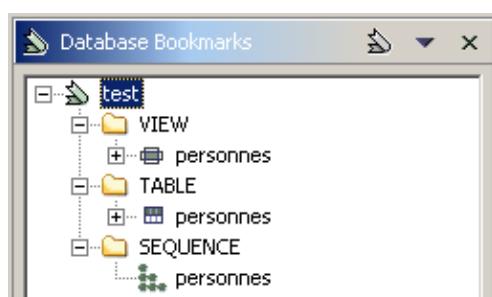


Le bookmark apparaît dans la liste. Le menu contextuel propose alors plusieurs options permettant de modifier ou de supprimer le bookmark. L'option "Connect" permet de demander la connexion à la base de données.

Aucun mot de passe n'ayant été fourni, une boîte de dialogue demande le mot de passe.



Si il n'y a aucun mot de passe, il suffit de cliquer sur le bouton "OK".



La vue "Database Bookmarks" affiche l'arborescence de la base de données. Dans l'exemple, la base de données ne contient qu'une seule table nommée "personnes".

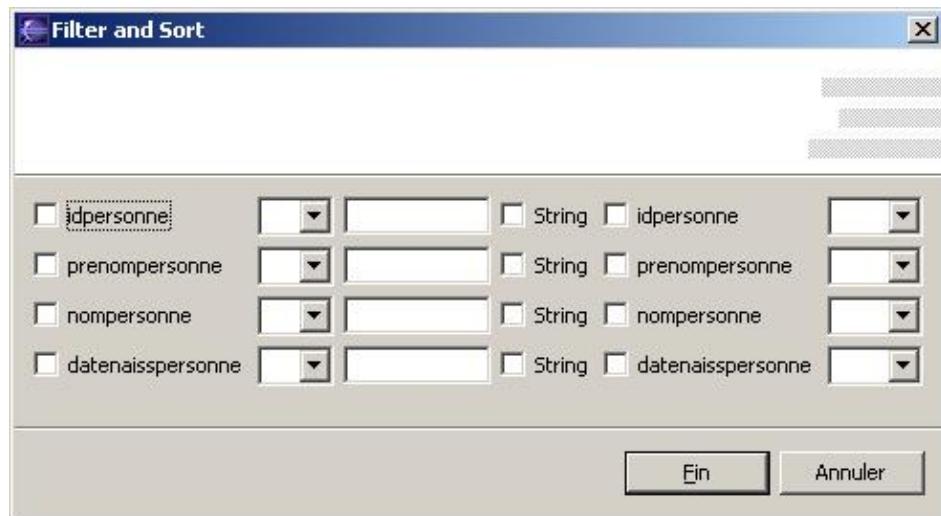
### 31.1.2. Afficher le contenu d'une table

Pour afficher le contenu d'une table, il suffit de double cliquer sur une table dans la vue « Data Bookmarks ».

La vue « Quantum Table View » affiche alors le contenu de la table.

The screenshot shows a table with four columns: idpersonne, prenompersonne, nompersonne, and datenaisspersonne. The first row contains the values 1, Prenom1, Nom1, and 1973-11-03 respectively. Below the table, it says "1 to 1 of 1". At the bottom, there are two tabs: "Quantum Table View" and "Quantum SQL Log".

Cette vue permet de trier et de filtrer le contenu de la table en cliquant que le bouton



Le filtre est défini dans la partie de gauche et le tri dans la partie de droite de la boîte de dialogue.

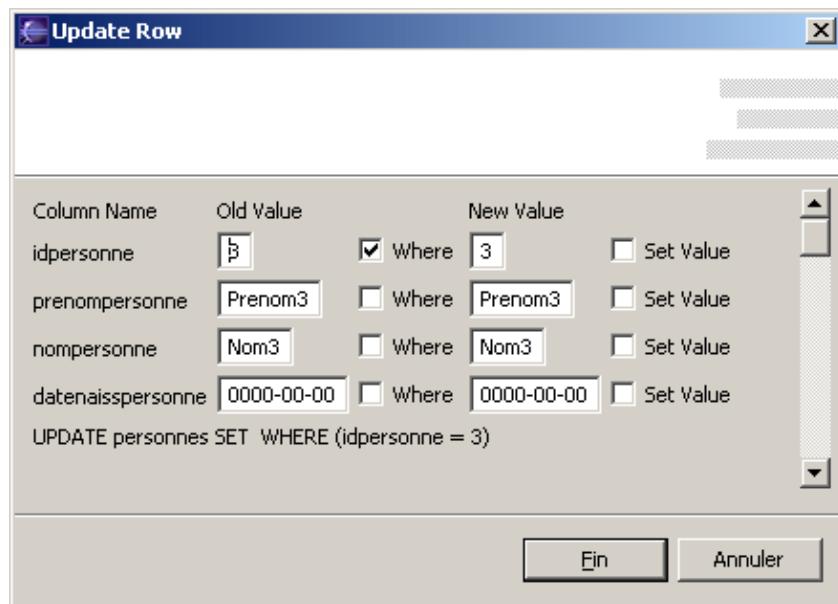
Le menu contextuel de la vue « Quantum Table View » propose des options pour faire des mises à jour des données de la table : insertion, modification et suppression.

Un assistant permet de saisir les données à ajouter dans la nouvelle occurrence de la table :

This dialog box is used for inserting new rows. It has a table with 'Column Name' and 'Value' columns. The values entered are: idpersonne (empty), prenompersonne (Prenom3), nompersonne (Nom3), and datenaisspersonne (empty). Below the table is a generated SQL query: "INSERT INTO personnes (prenompersonne, nompersonne) VALUES ('Prenom3', 'Nom3')". At the bottom are "Fin" and "Annuler" buttons.

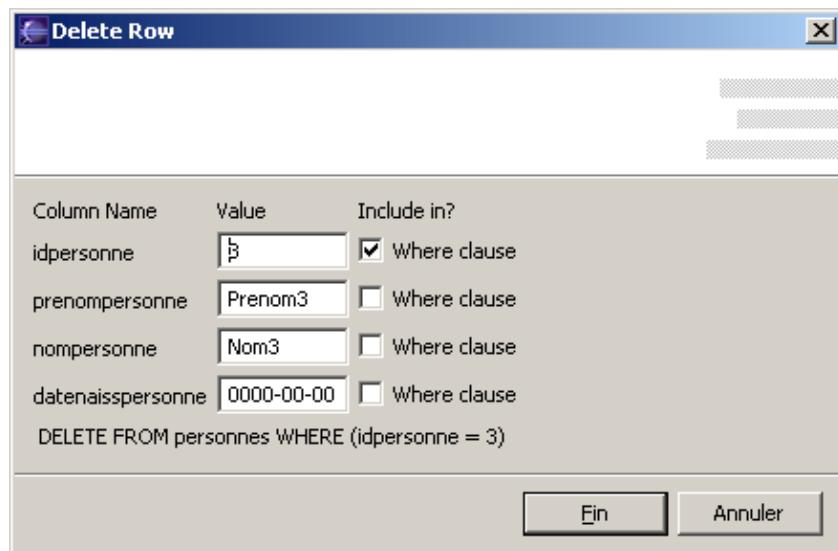
Une fois les données saisies, il suffit de cliquer sur le bouton « Fin » pour que la requête SQL générée soit exécutée. Pour voir les modifications dans la vue « Quantum Table View », il faut cependant explicitement demander le rafraîchissement en cliquant sur le bouton .

Un assistant permet de modifier les données d'une occurrence :



Le principe de la construction dynamique de la requête SQL est identique à celui de l'insertion d'une occurrence hormis le fait qu'il est possible de saisir les informations concernant la portée de la requête de mise à jour (clause Where).

L'assistant permettant la suppression d'une ou plusieurs occurrences s'utilise de la même façon :



### 31.1.3. Exécution d'une requête

L'éditeur « Quantum SQL Query Editor » permet de saisir des requêtes SQL avec une coloration syntaxique.

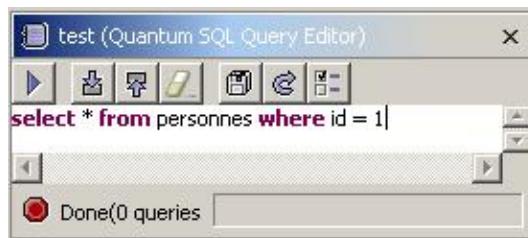


Le bouton  permet d'exécuter la requête.

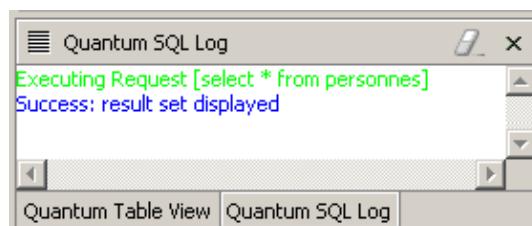
Si la requête contient un erreur, alors un message d'erreur est affiché :



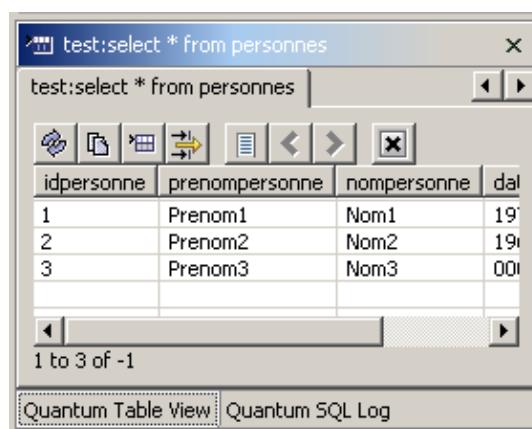
Un rond rouge en bas de la vue signale aussi une erreur lors de la dernière exécution.



La vue « Quantum SQL Log » affiche des informations sur l'exécution des requêtes.



Dans le cas d'une requête d'interrogation, le résultat est affiché dans la vue « Quantum Table View ».



Les boutons  et  permettent respectivement d'importer et d'exporter le contenu de l'éditeur dans un fichier possédant l'extension .sql par défaut.

La vue permet aussi de gérer les transactions. Le bouton  permet de préciser si le mode fonctionnement est auto-commit (bouton enfoncé) ou manuel. Dans le mode manuel, le bouton  permet de faire un commit des opérations non encore validées et le bouton  permet d'invalider les opérations non encore validées (rollback).

La vue « Query History » contient un historique des requêtes exécutées : un double clic sur une de ces requêtes permet de réafficher son contenu dans la vue « Quantum SQL Query ».

## 31.2. JFaceDbc

JFaceDBC est un plug-in qui propose de faciliter la manipulation d'une base de données. Ce plug-in possède quelques fonctionnalités intéressantes comme un système de gestion des connexions avec chargement dynamique des pilotes, l'affichage sous la forme graphique du schéma de la base de données ou la création pour une table des fichiers xml et java pour l'outil de mapping Hibernate.

Le site officiel est à l'url <http://jfacedbc.sourceforge.net/>

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
JFaceDBC	2.2.1.

La version utilisée dans cette section nécessite un JDK 1.4 pour fonctionner.

### 31.2.1. Installation et configuration

Après voir téléchargé le fichier net.sf.jfacedbc\_2.2.1.zip sur le site de JFaceDBC, il faut le décompresser dans le répertoire qui contient Eclipse. A l'exécution suivante, il est nécessaire de valider les modifications en attente et de relancer l'application.

Il est aussi possible d'installer JFaceDBC en utilisant le gestionnaire de mises à jour avec l'URL <http://jfacedbc.sourceforge.net/update-site/site.xml>

JFaceDBC propose une perspective particulière.



Cette perspective se compose de plusieurs vues et éditeurs.

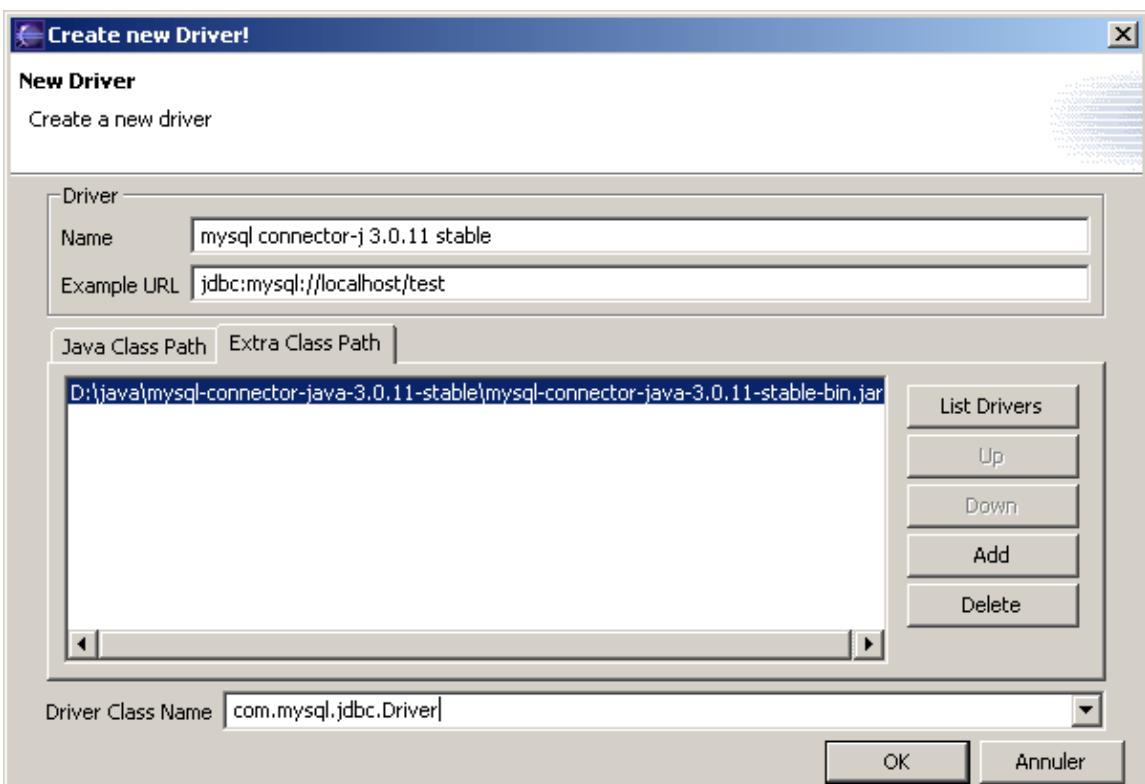


La vue "Drivers" permet de gérer les pilotes JDBC.



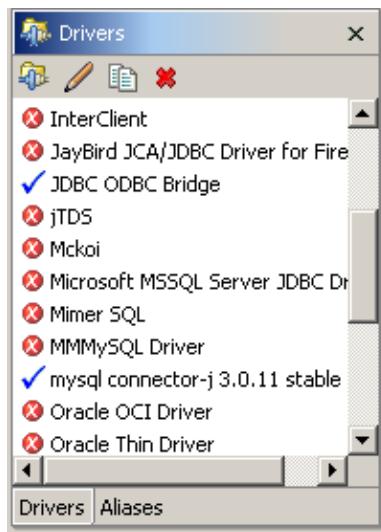
Un certain nombre d'entre eux sont préconfigurés et chargés dynamiquement si ils sont trouvés dans le classpath. Si le chargement a réussi alors une petite coche bleue est affichée au lieu d'une croix blanche dans un rond rouge.

Il est possible d'ajouter un nouveau pilote en cliquant sur le bouton . Voici un exemple avec le pilote MySQL Connector-J 3.0.



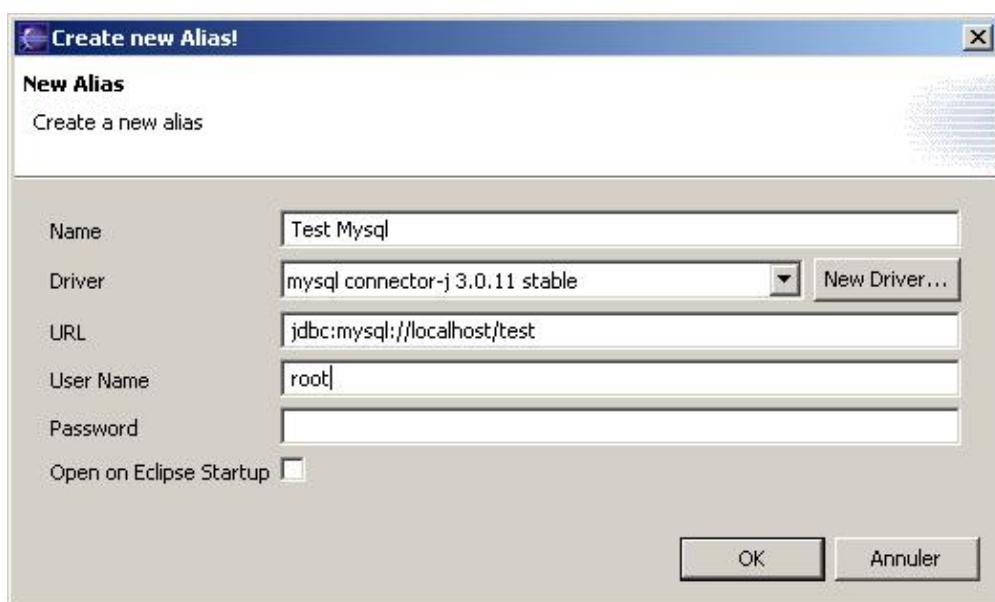
Il faut renseigner le nom du pilote, un exemple d'URL de connexion, sélectionner le fichier jar pour l'ajouter dans l'onglet « Extra Class Path », saisir le nom de la classe du Pilote et cliquer sur le bouton « OK »

Si le chargement du pilote a réussi, alors la petite coche bleue est affichée.

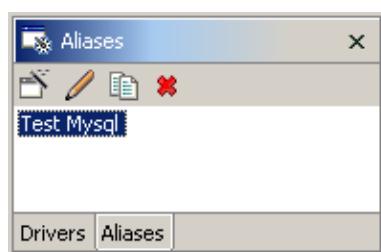


La vue "Aliases" permet de gérer des connexions. Pour ajouter un alias, il suffit de cliquer sur le bouton ou d'utiliser l'option « Create New Alias » du menu contextuel.

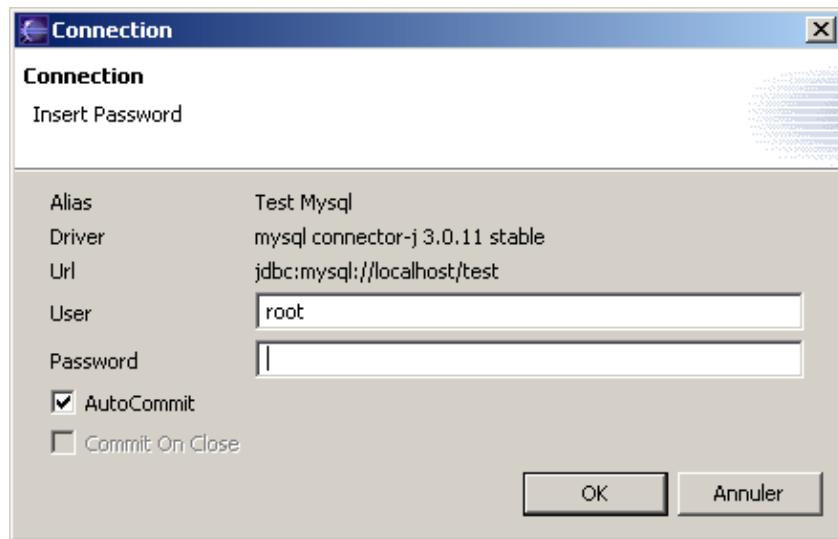
Une boîte de dialogue permet de saisir les informations concernant le nouvel alias de connexion.



Il faut saisir un nom, sélectionner le pilote parmis la liste de ceux définis, saisir l'URL de la connexion, le nom de l'utilisateur et son mot de passe.



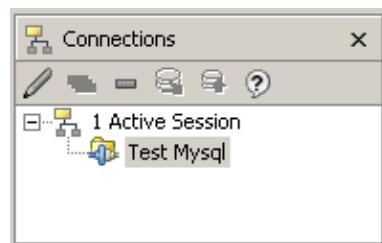
Pour ouvrir une connexion, il suffit de double cliquer sur l'alias ou d'utiliser l'option « Open » de son menu contextuel.



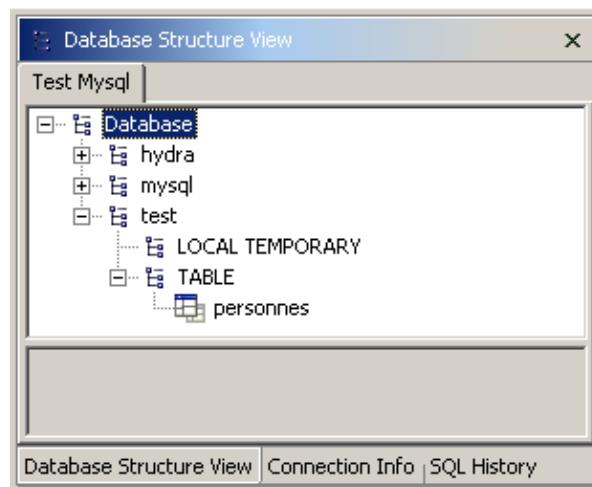
Saisissez les informations et cliquez sur le bouton "OK". JFaceDBC retrouve les informations de la base de données



La vue "Connections" affiche la ou les connexions actives :



La vue « Database Structure » affiche les bases de données accessibles via la connexion.



La vue « Connection Info » permet d'obtenir des informations sur les paramètres de la connexion .

Property	Value
Database Product Name	MySQL
Database Product Version	4.0.16-nt
Driver Major Version	3
Driver Minor Version	0
Driver Name	MySQL-AB JDBC Driver
Driver Version	mysql-connector-java-3.0.11-stable ( \$Date: 2005-09-22 11:00:00 +0000 \$ )
User Name	root@localhost
URL	jdbc:mysql://localhost/test
AutoCommit Mode	true
All Procedures Are Callable	false
All Tables Are Selectable	false
Nulls are sorted High	false
Nulls are sorted Low	true
Nulls are sorted at Start	false

Database Structure View | Connection Info | SQL History

Pour connaître la structure d'une table, il suffit de la sélectionner dans la vue « Database Structure View »

Database Structure View

Test Mysql

- Database
  - + hydra
  - + mysql
  - test
    - + LOCAL TEMPORARY
    - TABLE
      - personnes

Columns	Indexes	Primary Key	Foreign Key	Preview	Row Count	
Column...	Data T...	Size	Decim...	Defaul...	Accept...	Comm...
idperso...	int	11	0		NO	auto_i...
prenom...	varchar	50	0		YES	
nomper...	varchar	50	0		YES	
datenai...	date	10	0		YES	

Database Structure View | Connection Info | SQL History

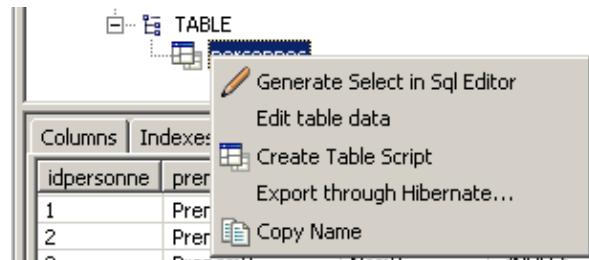
Les données peuvent être visualisées dans l'onglet « Preview ».

personnes

Columns	Indexes	Primary Key	Foreign Key	Preview	Row Count
idpersonne	prenompersonne	nompersonne	datenaisspersonne		
1	Prenom1	Nom1	1973-11-03		
2	Prenom2	Nom2	1968-03-23		
3	Prenom3	Nom3	<NULL>		

Database Structure View | Connection Info | SQL History

Une table possède un menu contextuel qui propose plusieurs options :



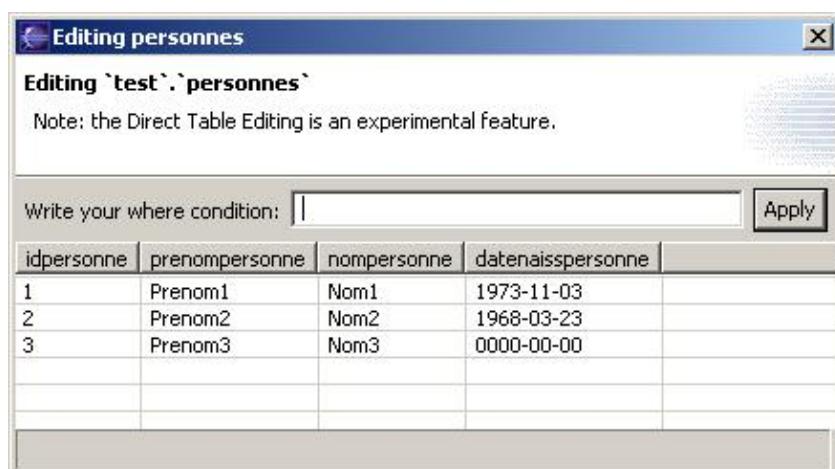
L'option « Generate Select in SQL Editor » ouvre l'éditeur SQL (« SQL Editor ») avec une requête permettant d'afficher le contenu de la table.

### 31.2.2. La mise à jour des données d'une table.

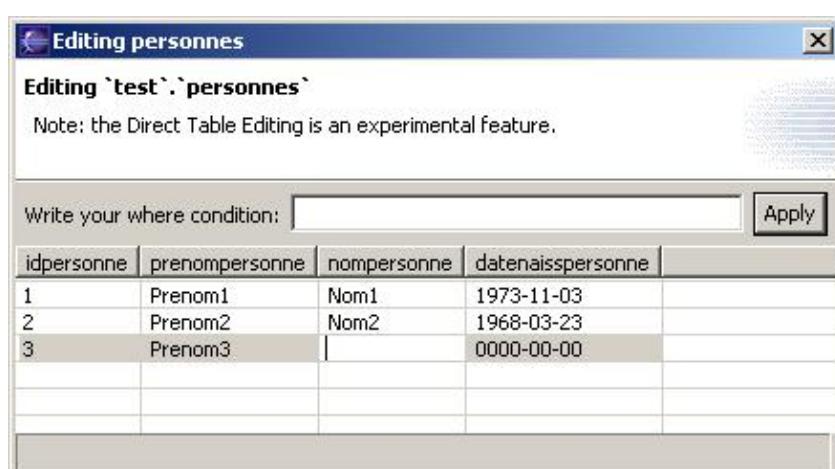
JFaceDBC propose une fonction pour mettre à jour les données d'une table.

Attention, cette fonctionnalité n'est utilisable qu'avec un pilote JDBC autorisant les déplacements des curseurs dans les deux sens.

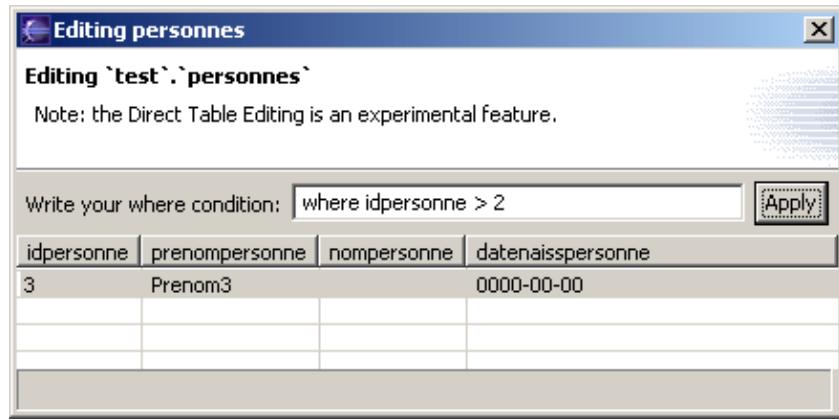
Pour utiliser cette fonctionnalité, il suffit d'utiliser l'option « Edit Table Data » du menu contextuel d'une table dans la vue « Database Structure ».



Il suffit alors de cliquer sur les données à modifier et de saisir sa nouvelle valeur :

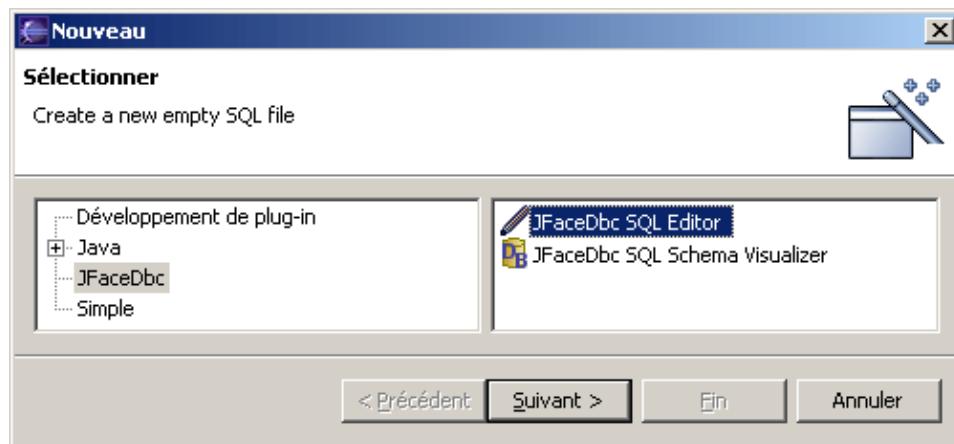


Il est possible d'appliquer un filtre pour limiter le nombre d'occurrences affichées dans la grille. Pour cela, il suffit de saisir la clause where qui sera ajoutée à la requête de sélection et de cliquer sur le bouton « Apply ».

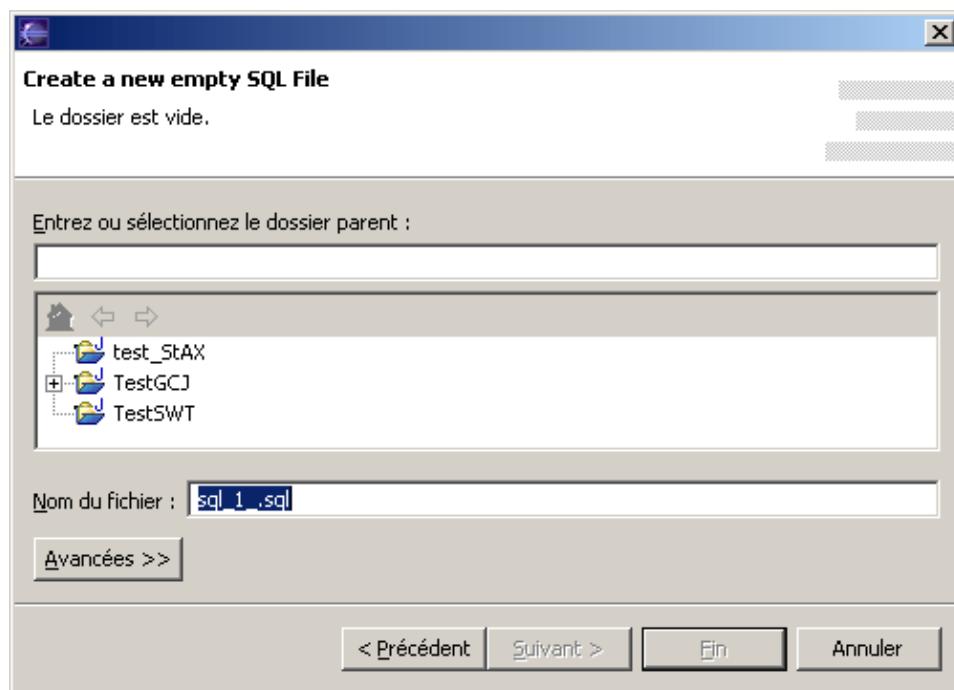


### 31.2.3. L'éditeur SQL

Pour ouvrir un nouvel éditeur SQL, il faut créer une nouvelle entité du type « JFaceDBC/JFaceDBC SQL Editor »



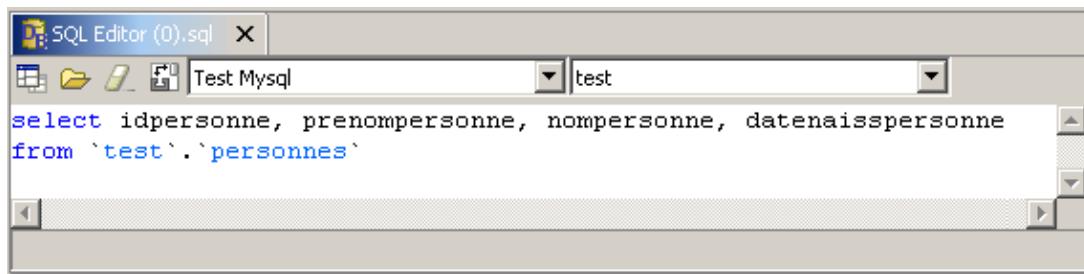
L'assistant demande ensuite le nom du fichier .sql qui sera créé et de sélectionner le projet qui va le contenir.



Un clic sur le bouton « Fin » crée le fichier et ouvre l'éditeur :



La première liste déroulante permet de sélectionner la connexion qui sera utilisée.



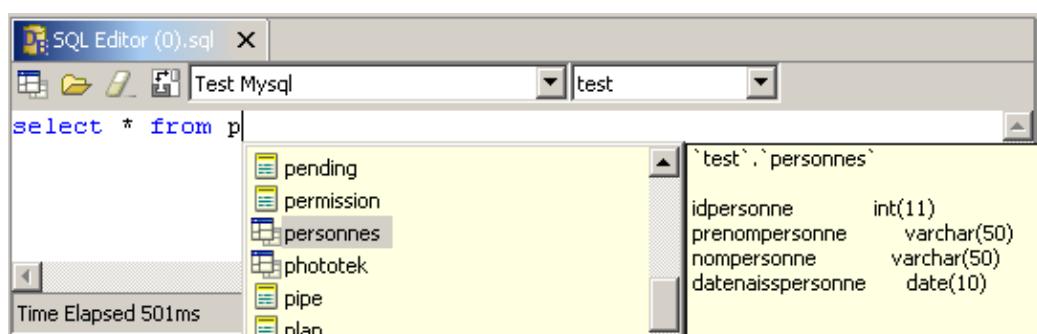
La seconde liste déroulante permet de sélectionner la base de données à utiliser.

Le bouton ou l'option « Execute SQL » du menu contextuel permet de lancer l'exécution de la requête.

Le résultat de la requête s'affiche dans la vue « SQL Results ».

SQL Results			
1			
	idpersonne	prenompersonne	nompersonne
1	1	Prenom1	Nom1
2	2	Prenom2	Nom2
3	3	Prenom3	Nom3

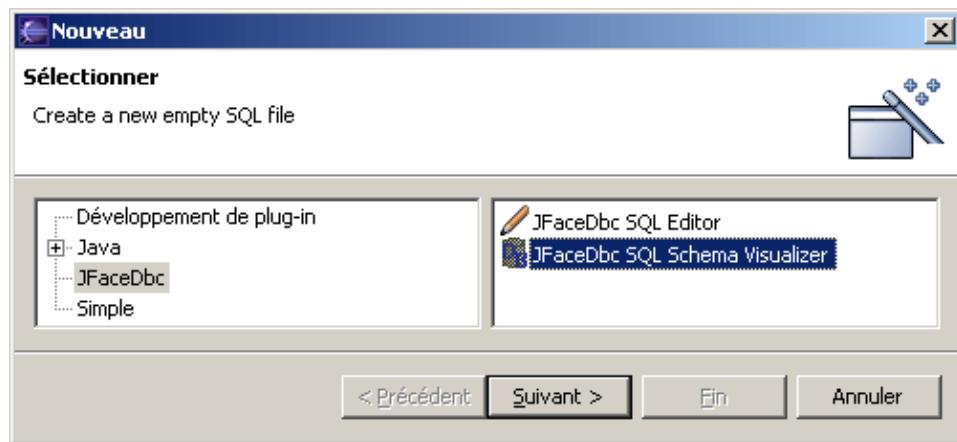
L'éditeur propose une coloration syntaxique des différents éléments de la requête mais aussi une complétion de code pour les tables et les champs, ce qui est très pratique. Cette fonction est appelée en utilisant la combinaison de touches "Ctrl + espace".



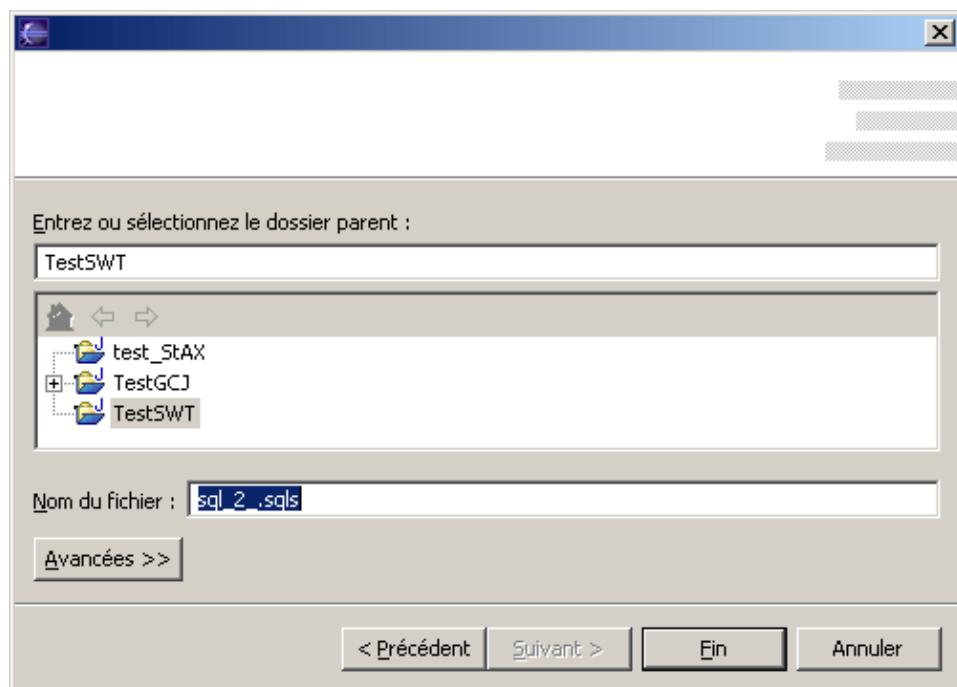
### 31.2.4. Vue graphique d'une base de données

Une autre fonctionnalité intéressante de JFaceDBC est de pouvoir afficher un schéma d'une base de données sous une forme graphique.

Pour cela, il faut créer une nouvelle entité de type "JFaceDBC/JFaceDBC SQL Schema Visualizer".



L'assistant demande ensuite le nom du fichier .sqls qui sera créé et de sélectionner le projet qui va le contenir.



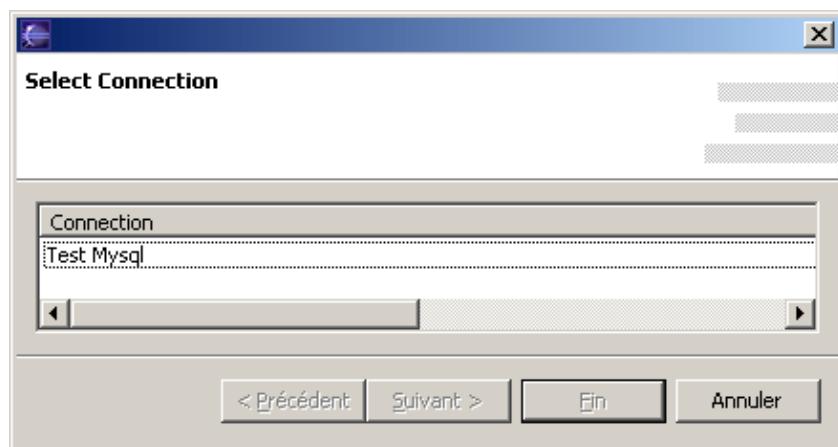
Un clic sur le bouton « Fin » créé le fichier et ouvre l'éditeur. Par défaut, ce dernier est vide.



Pour ajouter une nouvelle table au schéma, il suffit de faire un cliquer/glisser entre une table de la vue « Database Structure » et l'éditeur.

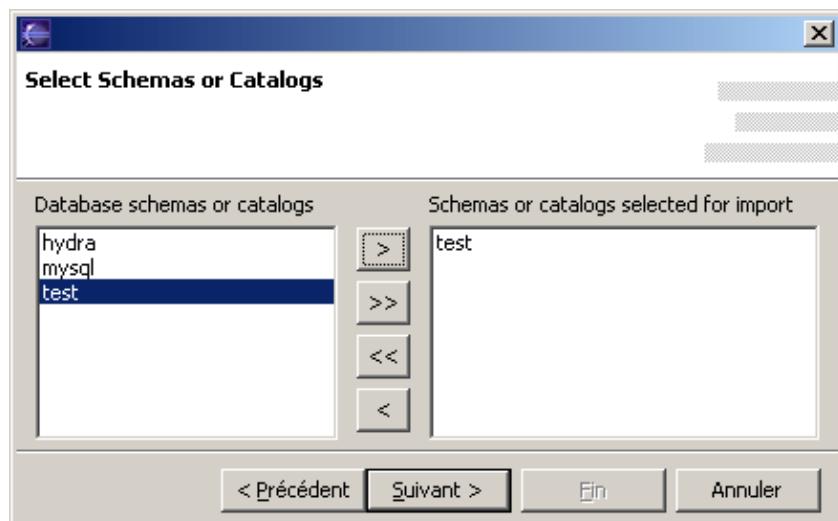
`test`.`personnes`	
idpersonne	int(11)
prenompersonne	varchar(50)
nompersonne	varchar(50)
datenaisspersonne	date

Le menu contextuel de l'éditeur propose l'option « Reverse Database ». Un assistant demande de sélectionner la connexion concernée.

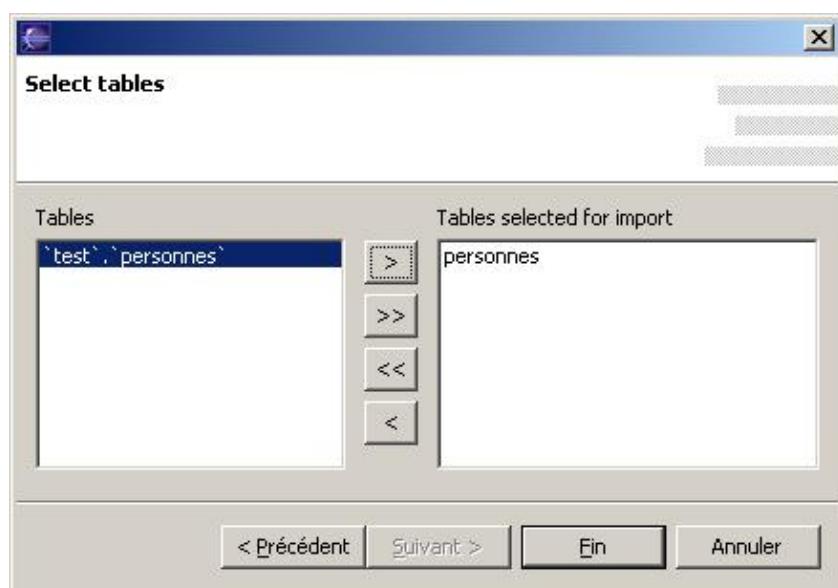


Il suffit de sélectionner la connexion concernée et de cliquer sur le bouton « Suivant ».

Il faut ensuite sélectionner la ou les bases de données à inclure parmi celles accessibles via la connexion.



Cliquer sur le bouton « Suivant ». L'assistant demande de sélectionner la ou les tables concernées.



Un clic sur le bouton « Fin » génère le schéma dans l'éditeur.

### 31.3. DBEdit

DBEdit est un plug-in qui permet l'exécution de requêtes sur une base de données.

Le site officiel est <http://sourceforge.net/projects/dbedit>

	Version utilisée
Eclipse	2.1.2
J2RE	1.4.2_02
DBEdit	1.0.1.

#### 31.3.1. Installation et configuration.

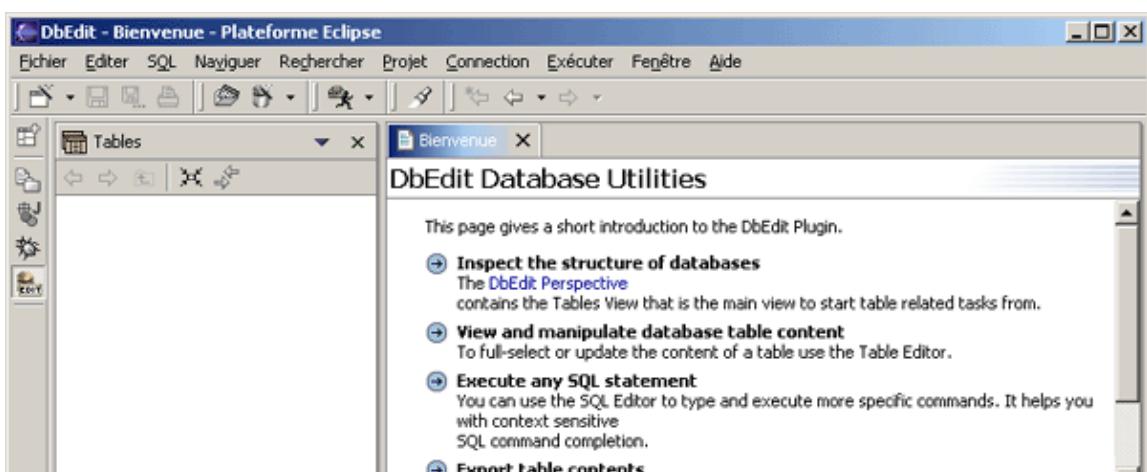
Il faut télécharger le fichier dbedit\_1.0.1.bin.dist.zip sur le site de DBEdit et décompresser son contenu dans le répertoire d'installation d'Eclipse. A l'exécution suivante, il est nécessaire de valider les modifications en attente et de relancer l'application.

Il est aussi possible d'utiliser l'URL [http://www.geocities.com/uwe\\_ewald/dbedit/site](http://www.geocities.com/uwe_ewald/dbedit/site) dans le gestionnaire des mises à jour.

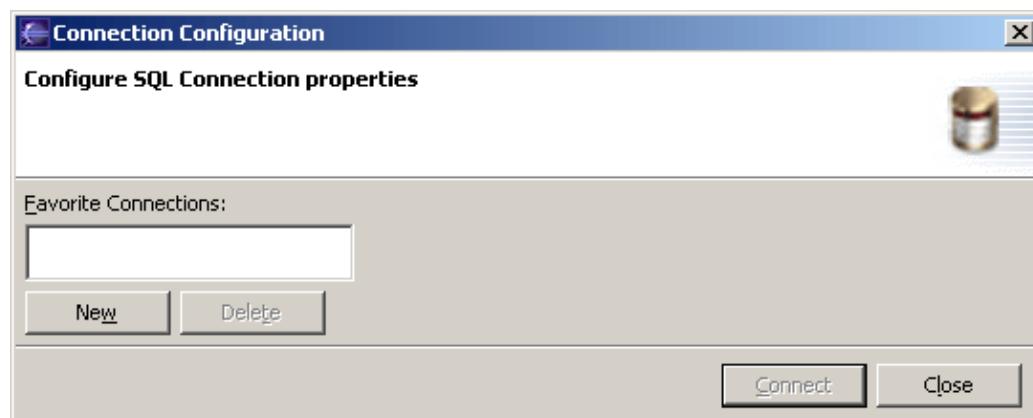
DBEdit propose une perspective particulière.



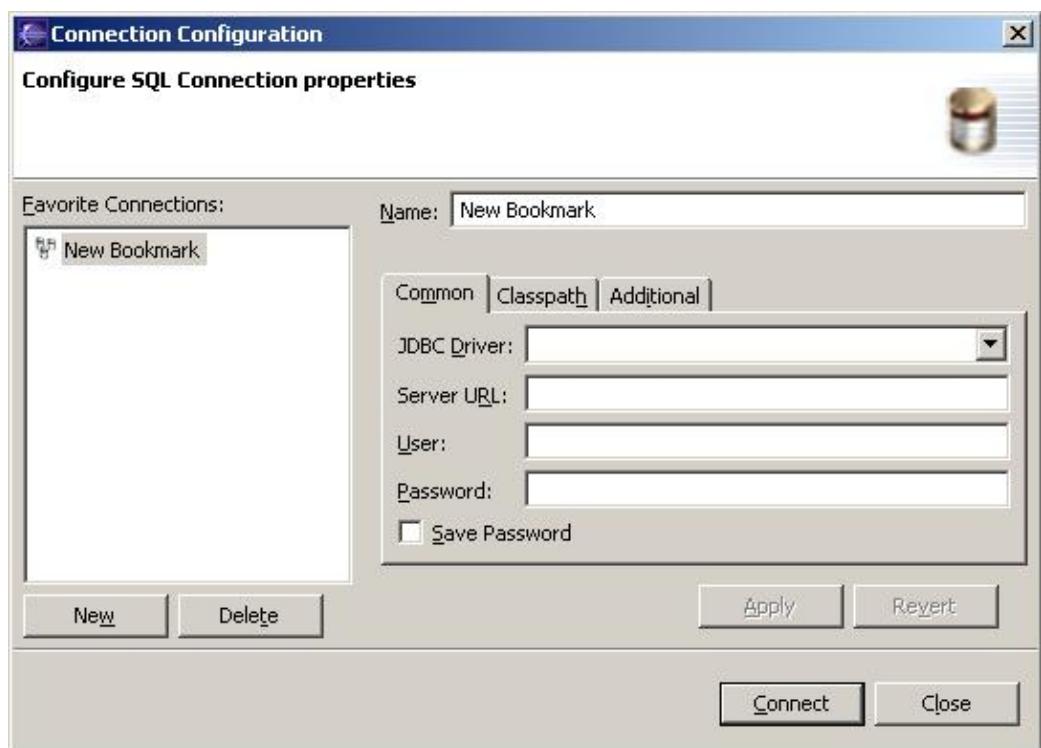
Cette perspective se compose de plusieurs vues et éditeurs.



La première chose à faire est de définir une connexion en utilisant l'option « Connection/Configure » du menu contextuel de la vue « Tables ». Une boîte de dialogue permet de gérer les connexions.



Pour ajouter une nouvelle connexion, il suffit de cliquer sur le bouton « New ».

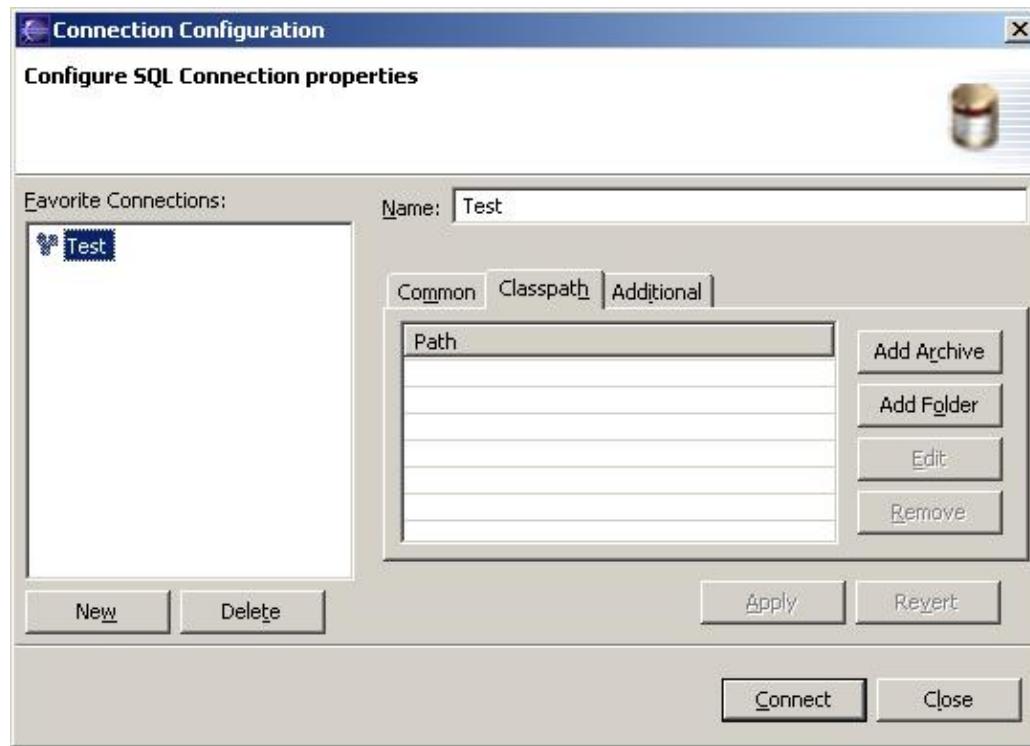


Il suffit alors de saisir les informations nécessaires :

- Name : est le nom de la connexion
- JDBC Driver : permet de préciser le nom de la classe du pilote JDBC
- Serveur URL : est l'URL de connexion
- User et password : sont le nom et le mot de passe de l'utilisateur

Un clic sur le bouton « Apply » valide les informations.

Il faut ensuite ajouter le pilote au classpath en cliquant sur l'onglet « Classpath »



Il suffit de cliquer sur le bouton « Add Archive » et de sélectionner le fichier Jar contenant le pilote à utiliser.

L'onglet « Additional » permet de préciser des paramètres supplémentaires au pilote JDBC.

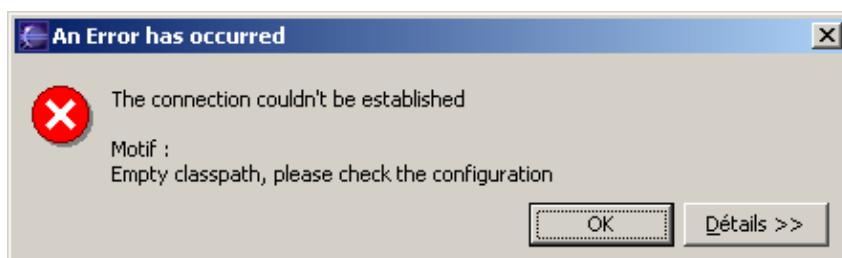
Un clic sur le bouton « Connect » permet d'ouvrir la connexion.

### 31.3.2. La vue « Tables »

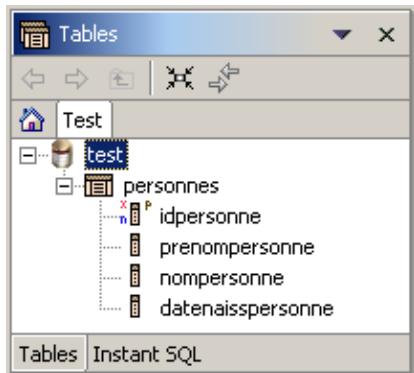
Par défaut, elle affiche la liste des connexions enregistrées. Pour ouvrir une connexion, il suffit d'utiliser l'option « Connection/Connect » du menu contextuel d'une connexion.



Si le pilote n'est pas inclus dans le classpath, un message d'erreur est affiché



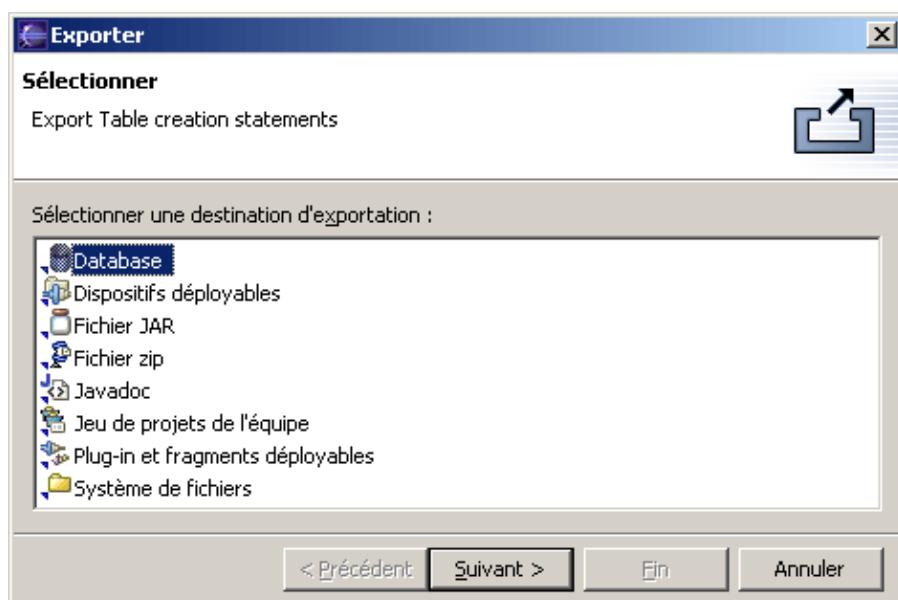
Une fois la connexion établie, la vue « Tables » affiche une arborescence contenant les éléments de la base de données : tables, vues, champs, index ....



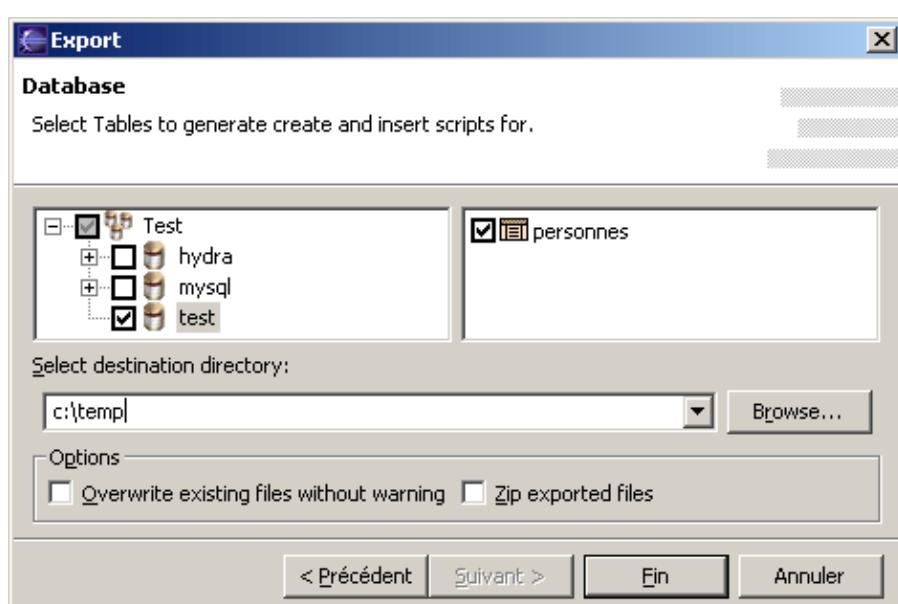
A partir du menu contextuel de cette vue, il est possible de réaliser plusieurs actions :

- créer une nouvelle entité : connexion, schéma, table, fichier SQL
- importer ou exporter une table
- ouvrir une table dans l'éditeur

L'export se fait avec un assistant :



Il faut sélectionner "Database" et cliquer sur le bouton « Suivant »

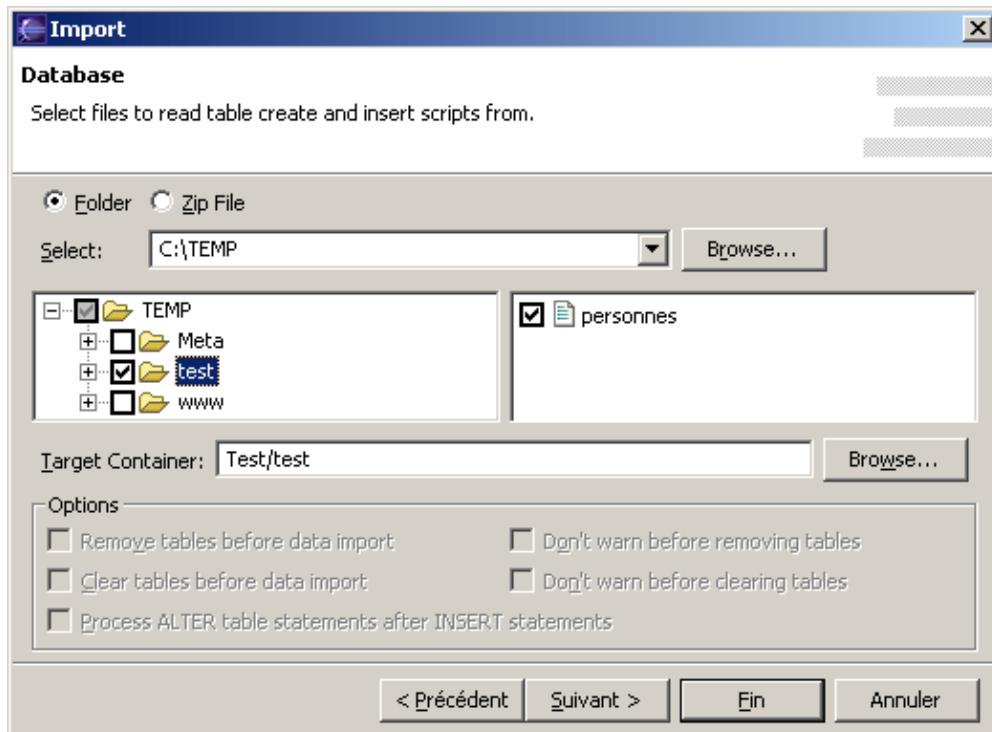


Il faut sélectionner la connexion et les tables concernées, sélectionner le répertoire de destination et cliquer sur le bouton « Fin ».

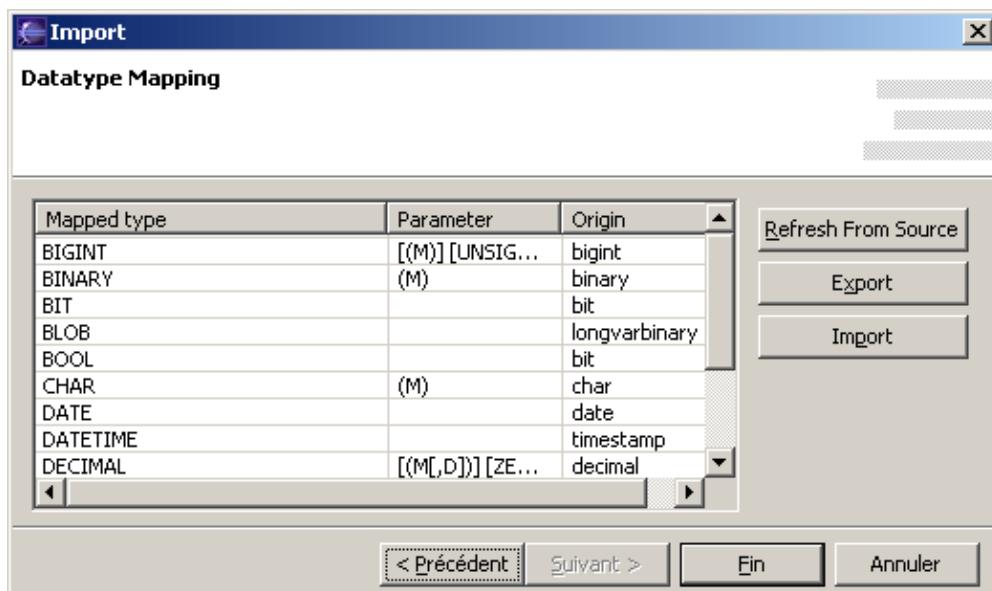
L'exportation se fait sous la forme de requêtes SQL :

```
1 quote='
2 create table `personnes` (`idpersonne` int not null, `prenomper
3 alter table `personnes` add constraint `PRIMARY` primary key (`
4 alter table `personnes` add constraint `idpersonnes` primary ke
5 insert into `personnes` (`idpersonne`, `prenompersonne`, `nompe
6 insert into `personnes` (`idpersonne`, `prenompersonne`, `nompe
7 insert into `personnes` (`idpersonne`, `prenompersonne`, `nompe
8 insert into `personnes` (`idpersonne`, `prenompersonne`, `nompe
9
```

L'importation se fait aussi avec un assistant

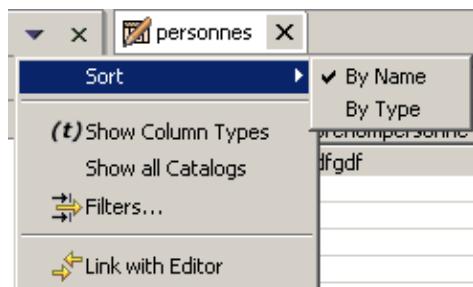


Il suffit de sélectionner les entités et de cliquer sur le bouton « Suivant ». La page suivante permet de préciser les options de correspondance des types de données.

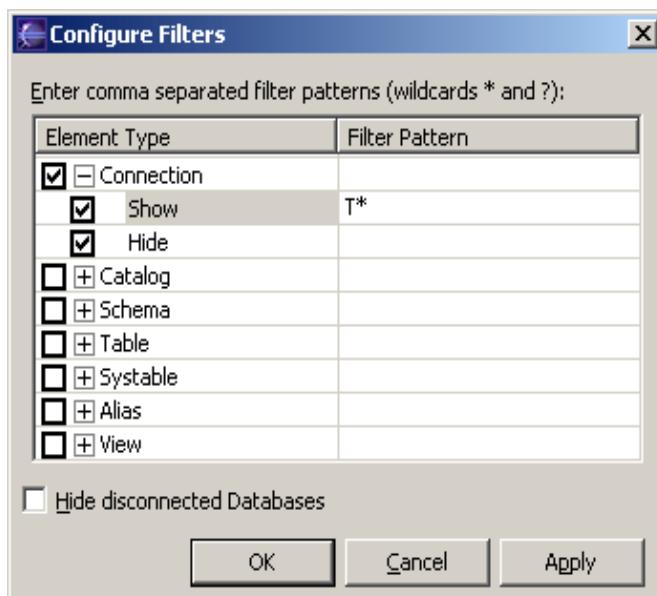


Un clic sur le bouton « Fin » permet de lancer l'importation.

Un clic sur le bouton  permet de régler certaines options d'affichage.



L'option "Filters" permet de restreindre les types d'éléments qui sont affichés :



### 31.3.3. L'éditeur « Table »

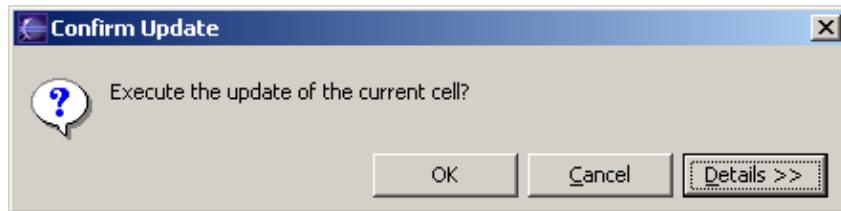
Sur la vue « Tables », l'option « Open with/ Table Editor » du menu contextuel d'une table permet d'ouvrir un éditeur avec les données de la table.

Row	idpersonne	prenompersonne	nompersonne	datenaisspersonne
1	1	Prenom1	Nom1	1973-11-03
2	2	Prenom2	Nom2	1968-03-23
3	3	Prenom3		<NULL>

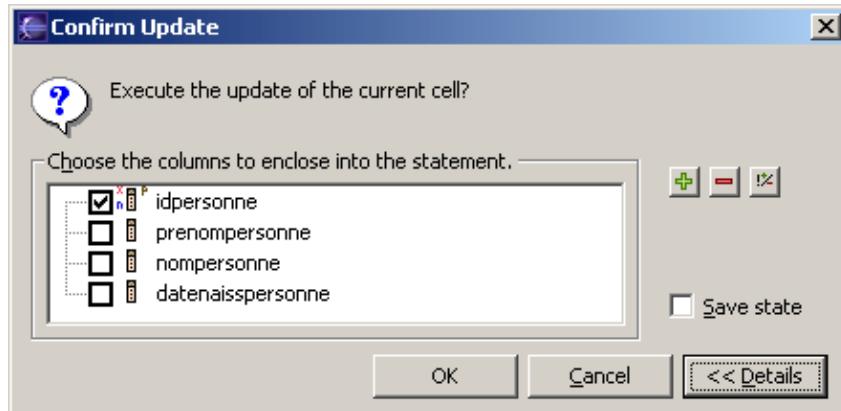
Par défaut, l'éditeur est en mode consultation. Le menu contextuel permet de réaliser certaines opérations :

- Edit Mode : permet de basculer dans le mode de mise à jour des données pour un seul champ
  - Insert Row : permet d'insérer une nouvelle occurrence dans la table
  - Delete Row : permet de supprimer l'occurrence sélectionnée

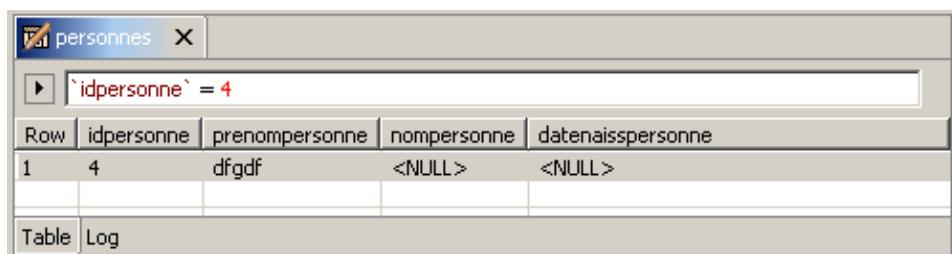
Chaque mise à jour doit être confirmée dans une boîte de dialogue



Un clic sur le bouton « Details » permet d'obtenir des informations précises sur l'opération.

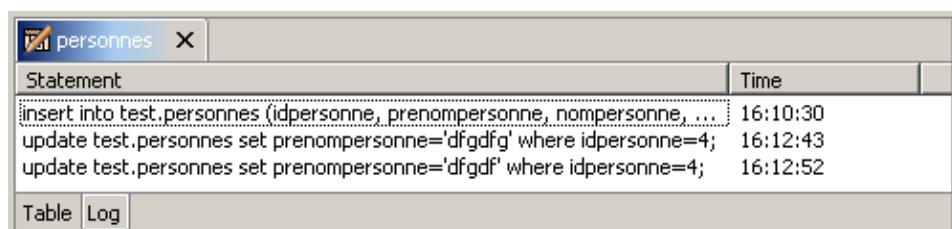


Le bouton dans la barre d'outils permet de rafraîchir les données de l'éditeur. Le bouton dans la barre d'outils permet de saisir un filtre.



### 31.3.4. La vue Log

Cette vue recense toutes les requêtes SQL qui sont exécutées.



### 31.3.5. L'éditeur Instant SQL

Cet éditeur permet de saisir et d'exécuter des requêtes. Il suffit de saisir la requête dans la zone d'édition

The screenshot shows the Instant SQL interface. The main window has a toolbar at the top with various icons. Below the toolbar is a text input field containing the SQL query: 'select \* from personnes'. A status message below the input field reads: 'To display a query result, open a connection, type an SQL statement into the upper area and execute it using the toolbar actions.' At the bottom of the window are two tabs: 'Tables' and 'Instant SQL', with 'Instant SQL' currently selected.

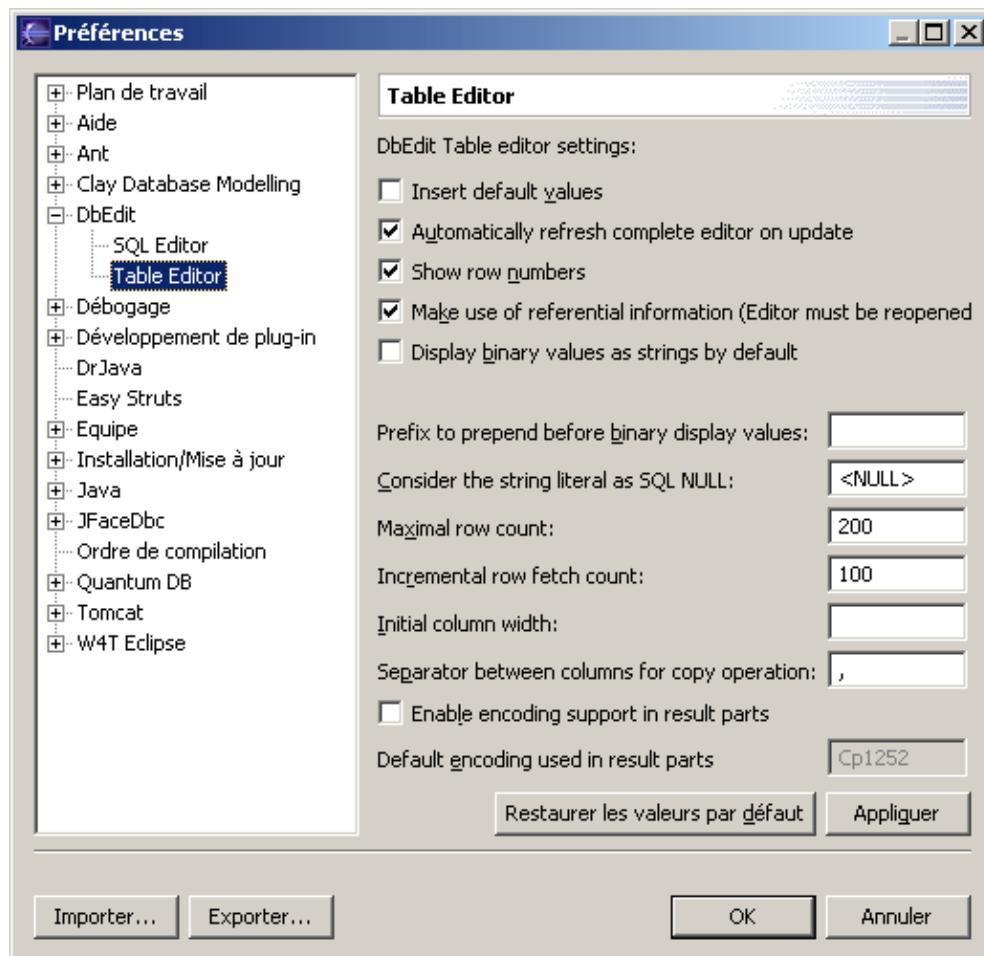
puis de cliquer sur le bouton pour lancer l'exécution

The screenshot shows the Instant SQL interface after executing the query. The main window now displays a table titled 'Result' with four columns: Row, idpersonne, prenompersonne, nompersonne, and datenaisspersonne. The table contains four rows of data:

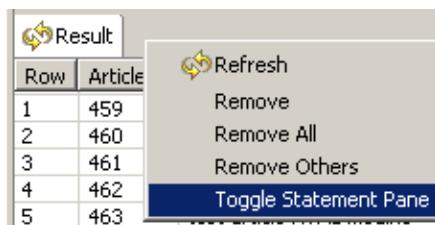
Row	idpersonne	prenompersonne	nompersonne	datenaisspersonne
1	1	Prenom1	Nom1	1973-11-03
2	2	Prenom2	Nom2	1968-03-23
3	3	Prenom3		<NULL>
4	4	df	<NULL>	<NULL>

At the bottom of the window are two tabs: 'Tables' and 'Instant SQL', with 'Instant SQL' currently selected.

Il est possible de limiter le nombre d'occurrences renvoyées par une requête. Ceci est nécessaire pour des tables possédant de nombreuses occurrences car dans ce cas un affichage de toutes les occurrences provoque une exception de type OutOfMemory. Ce nombre d'occurrences est réglable dans les préférences.



La version 1.0.2 propose une pagination sur les occurrences obtenues en fonction du nombre d'occurrences maximales à afficher. Pour l'activer, il faut utiliser l'option "Toogle" du menu contextuel de la barre de titre "Résultats".



L'activation de cette option affiche un panneau permettant la pagination.



## 31.4. Clay Database Modelling

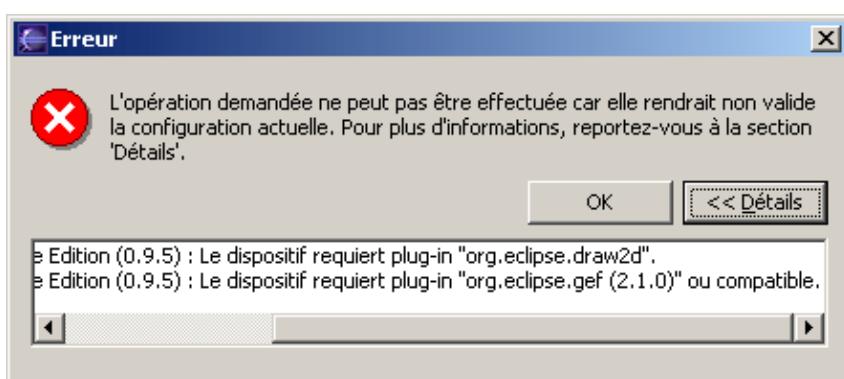
Ce plug-in permet de concevoir ou retro-concevoir une base de données graphiquement.

Les bases de données supportées sont : HSQLDB, MySQL, PostgreSQL, Firebird, ANSI SQL-92, SAP DB, McKoi.

Le site officiel est <http://www.azzurri.jp/en/software/clay/index.jsp>

	Version utilisée dans cette section
Eclipse	2.1.2
J2RE	1.4.2_02
Clay Database Modelling	0.9.5

Ce plug-in nécessite le plug-in GEF (Graphical Editor Framework). Si il est absent, un message d'erreur est affiché.



### 31.4.1. Installation et configuration.

Il faut télécharger le fichier jp.azzurri.clay.core\_0.9.5.bin.dist.20031201.zip sur le site du plug-in et décompresser son contenu dans le répertoire d'installation d'Eclipse. A l'exécution suivante, il est nécessaire

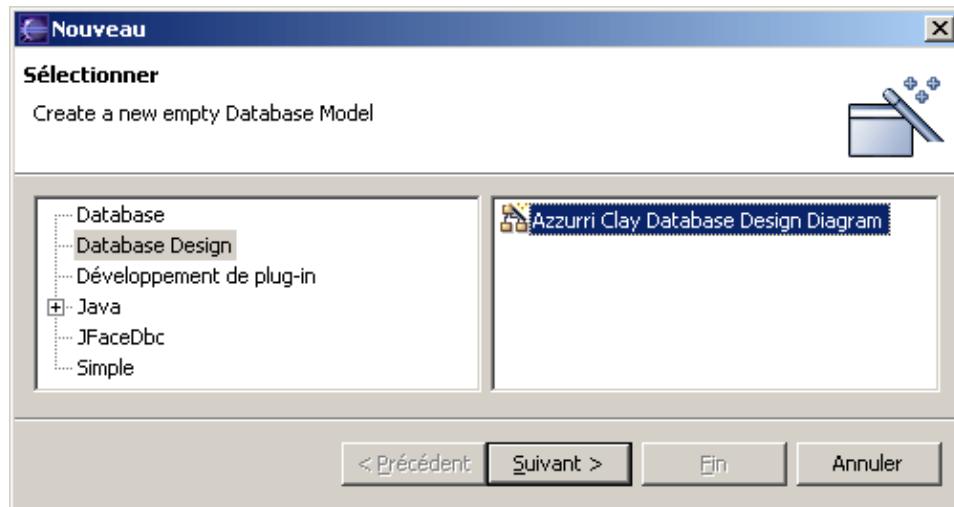
de valider les modifications en attente et de relancer l'application.

L'installation par le gestionnaire de mises à jour est possible en utilisant l'url  
<http://www.azzurri.jp/eclipse/plugins>

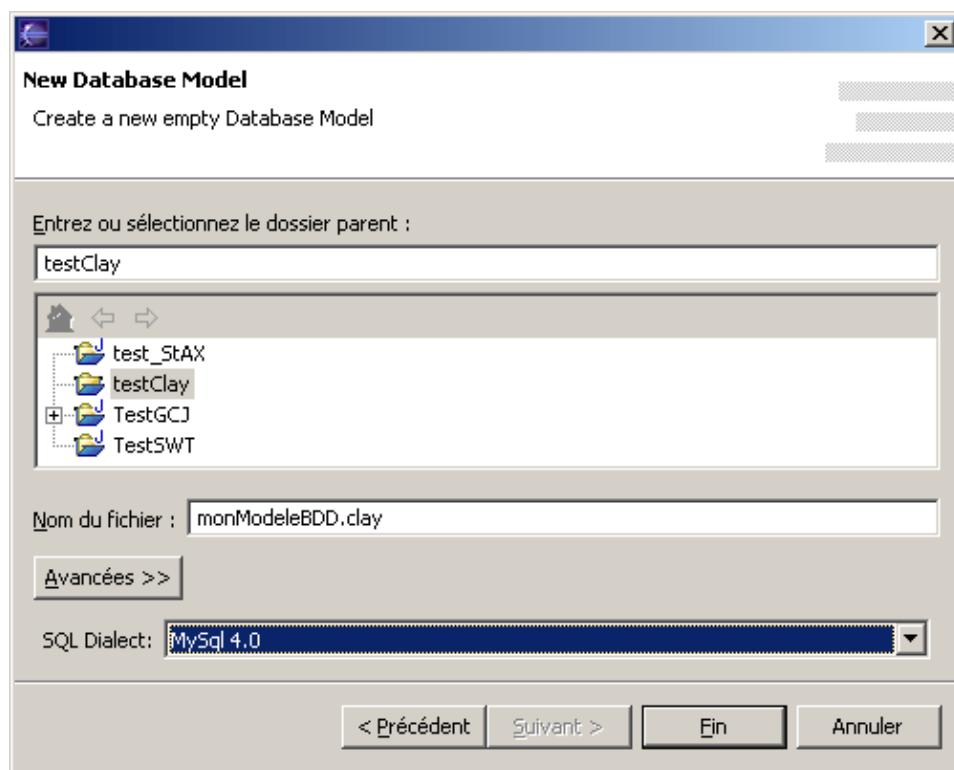
### 31.4.2. Mise en oeuvre

Clay Database Modelling ne propose pas de perspective particulière.

Il faut créer un nouveau projet ou utiliser un projet existant. Dans ce projet, il faut définir un nouveau modèle en utilisant l'option de création d'un nouvel élément de type « Autre ... ».

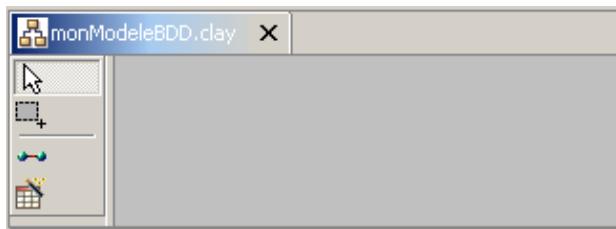


Il faut sélectionner « Database design » et « Azzurri Clay Database Design Diagram » puis cliquer sur le bouton « Suivant ».

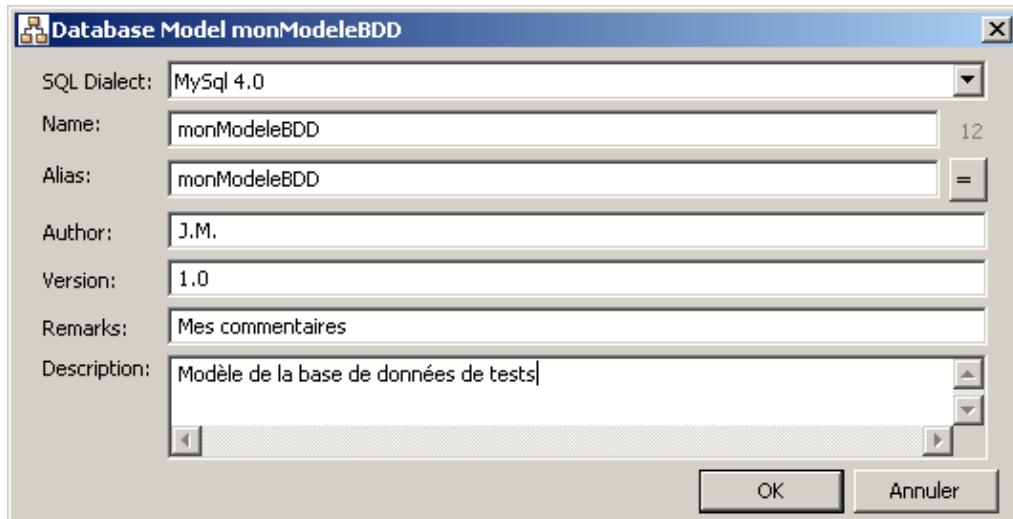


Il faut sélectionner l'emplacement du fichier de données (par défaut le répertoire du projet en cours), saisir le nom de ce fichier, et sélectionner le type de base de données cible dans lequel les scripts SQL seront générés.

Un éditeur dédié s'ouvre avec le fichier créé, par défaut vide.



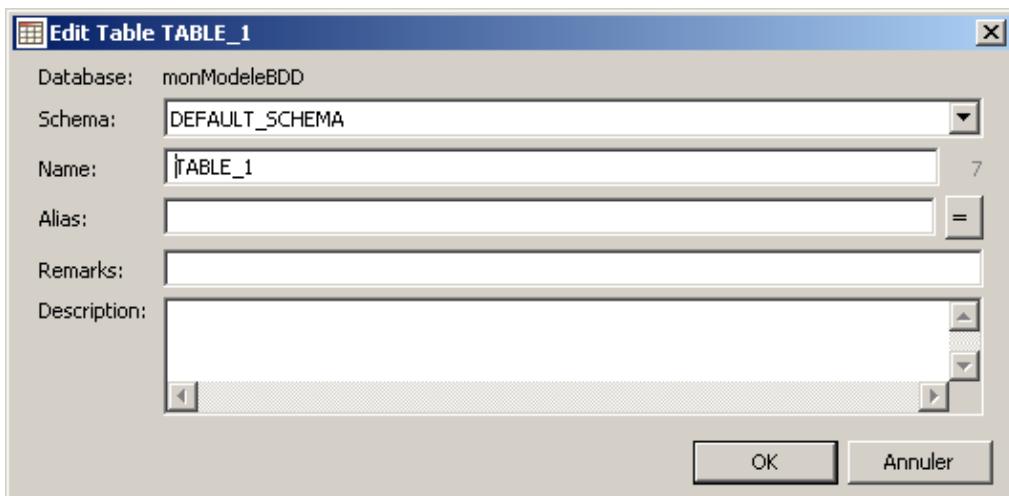
L'option « Edit Database Model » du menu contextuel permet de saisir des informations concernant le modèle.



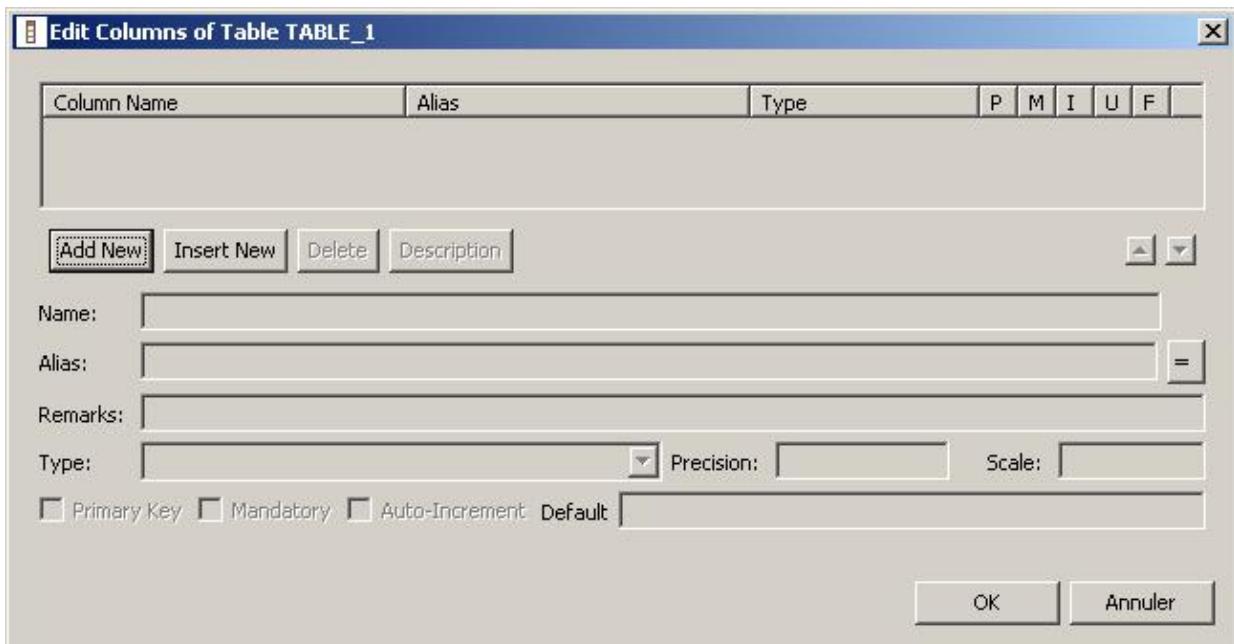
Dans cette boîte de dialogue, la donnée « SQL Dialect » est particulièrement importante car elle va conditionner les possibilités de l'éditeur selon le type de base de données sélectionnées, notamment par exemple au niveau des types de données utilisables dans les définitions de tables.

Le bouton permet d'ajouter une nouvelle table au modèle en cliquant sur le bouton puis sur la zone d'édition de l'éditeur.

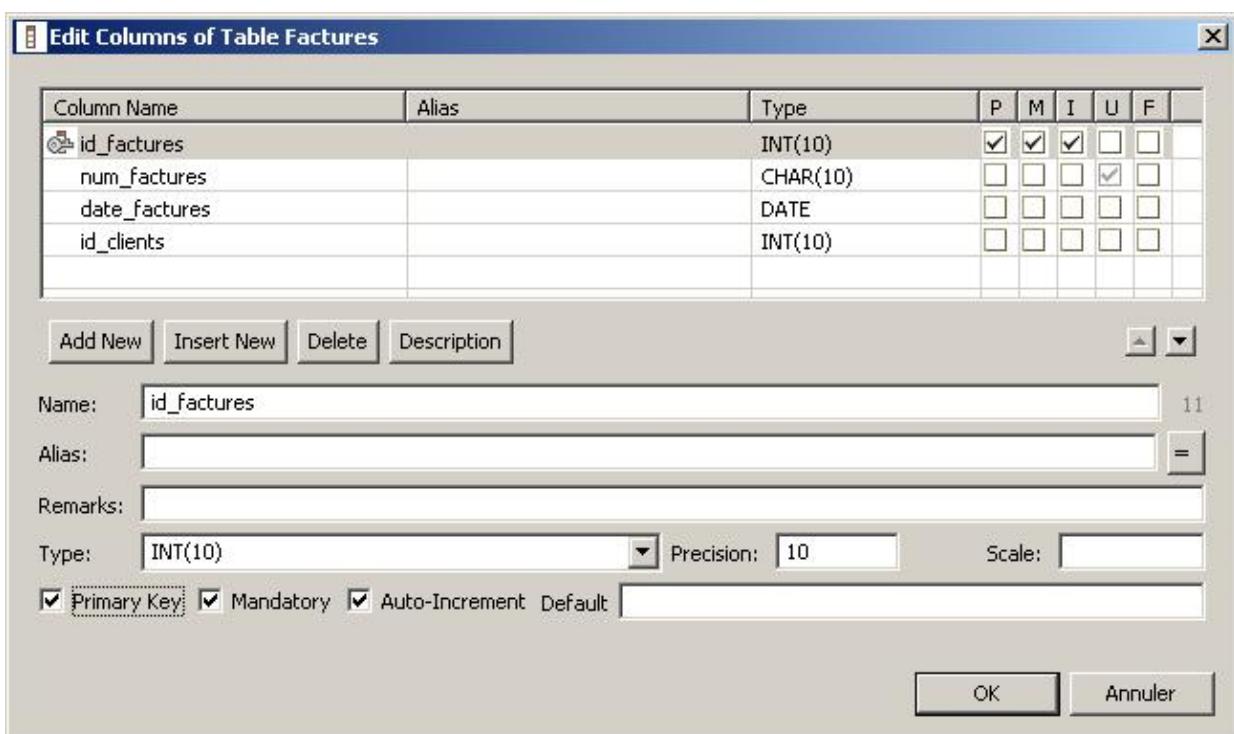
L'option « Edit Table » du menu contextuel permet de saisir des données générales de la table tel que son nom, une description ou des remarques.



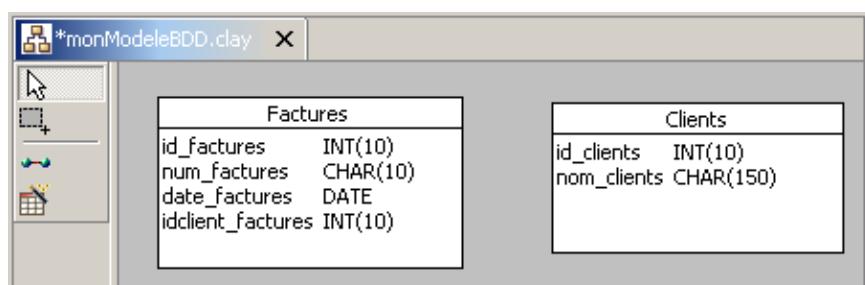
Un double clic sur la nouvelle table créée ou l'utilisation de l'option « Edit Table Column » permet de saisir la description des champs qu'elle va contenir.



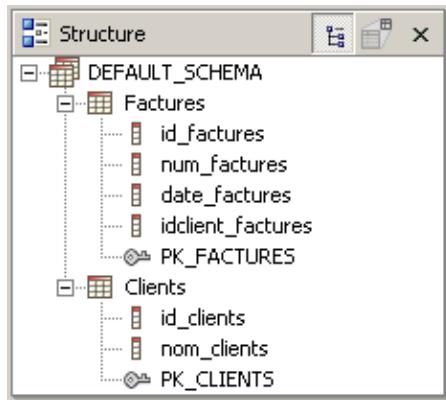
Pour ajouter une nouvelle colonne, il suffit de cliquer sur le bouton « Add New »



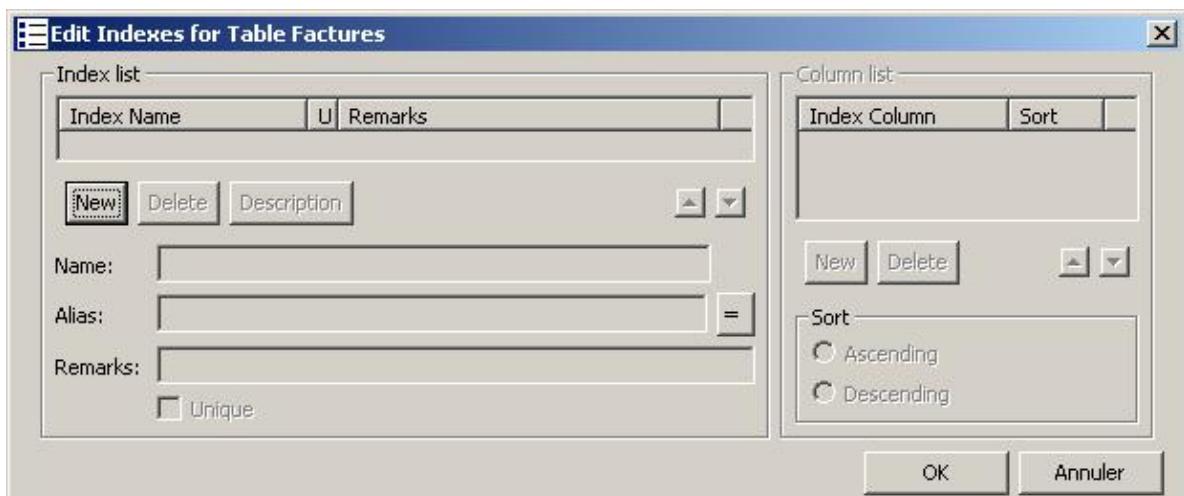
Il suffit alors de saisir les données du nouveau champ. L'opération doit être répétée pour chaque table du modèle. Exemple :



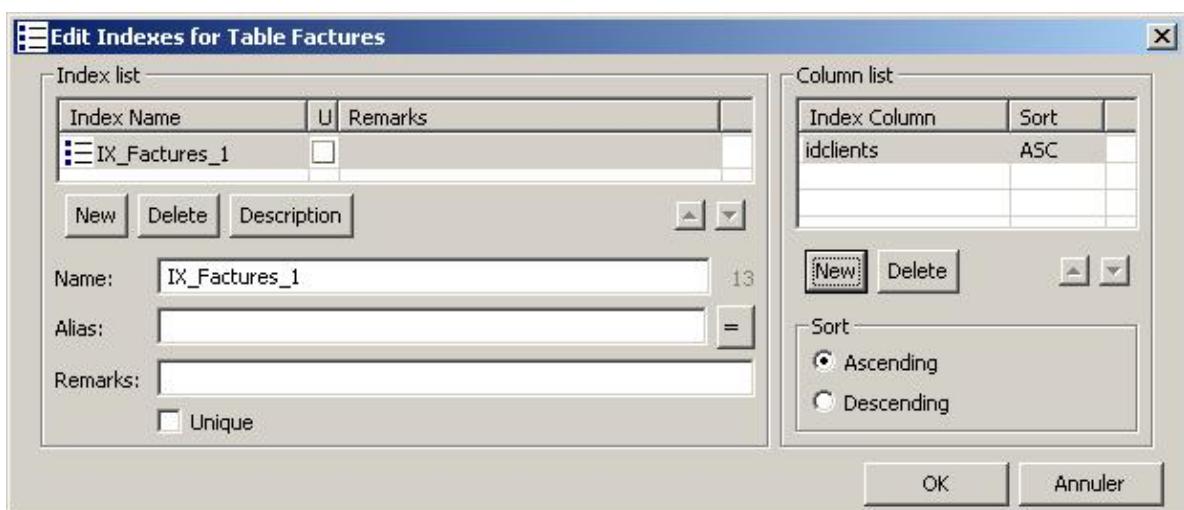
La vue "Structure" affiche sous une forme arborescente la structure du modèle avec les différents éléments qui la compose.



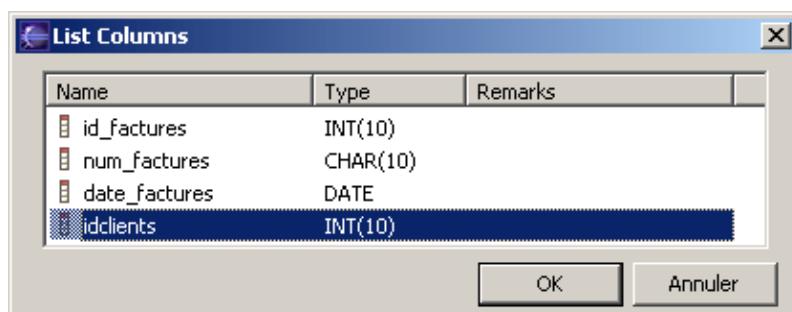
Il est possible de créer des index en utilisant l'option « Edit Table Indexes » du menu contextuel.



Pour ajouter un nouvel index, il faut cliquer sur le bouton « New »

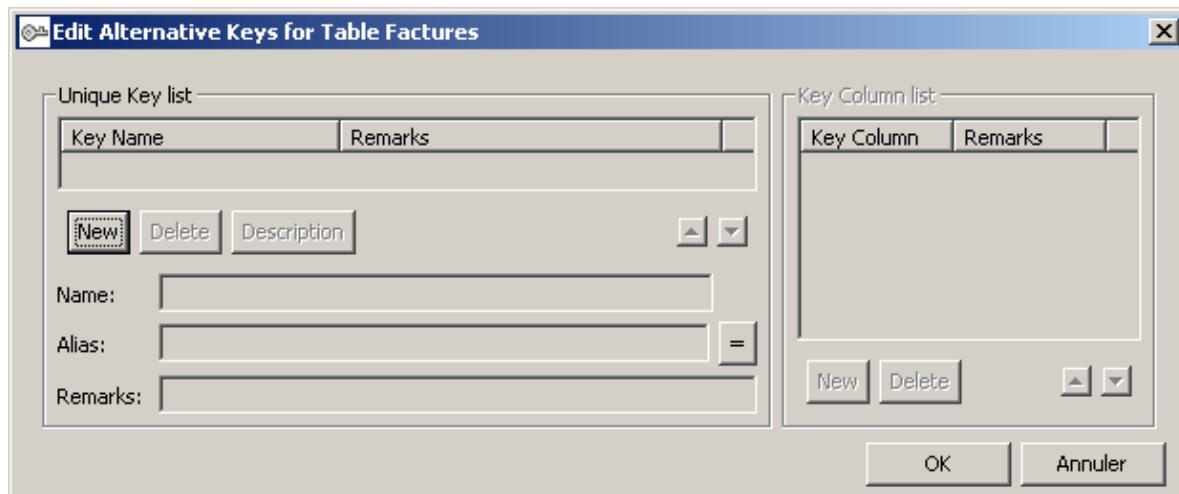


Pour ajouter un champ dans l'index, il faut cliquer sur le bouton « New » dans « Key Column List »

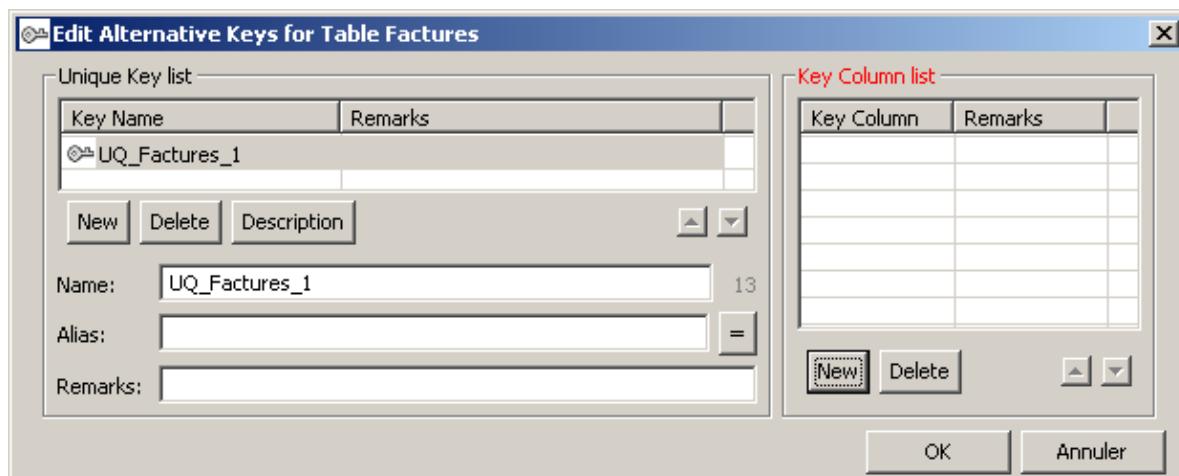


Sélectionnez le champ et cliquez sur le bouton « Ok ». Une fois toutes les informations saisies, il suffit de cliquer sur le bouton « OK ». Le nouvel index apparaît dans la vue « Structure ».

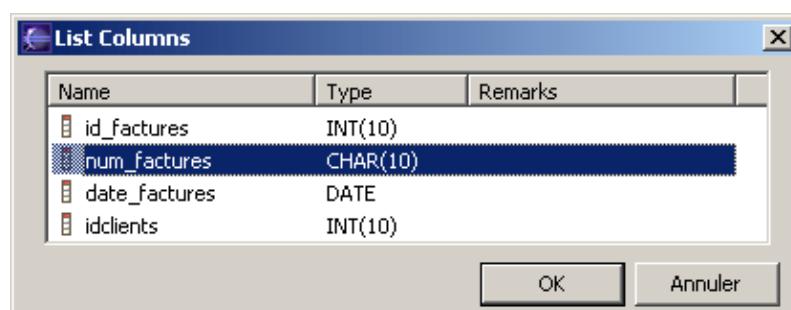
Il est possible de définir des index uniques en utilisant l'option « Edit Table Unique Keys » du menu contextuel.



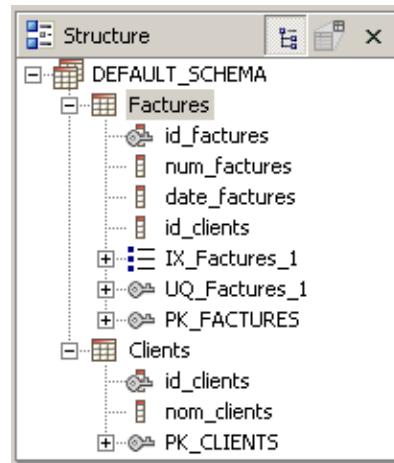
Pour ajouter un nouvel index, il faut cliquer sur le bouton « New »



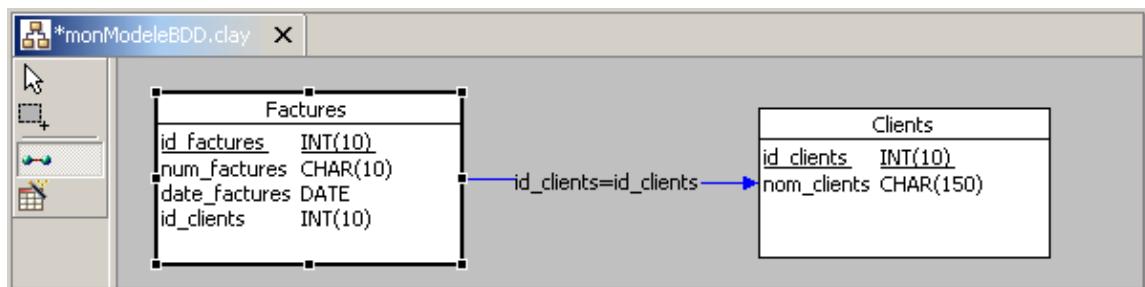
Pour ajouter un champ dans l'index, il faut cliquer sur le bouton « New » dans « Key Column List ».



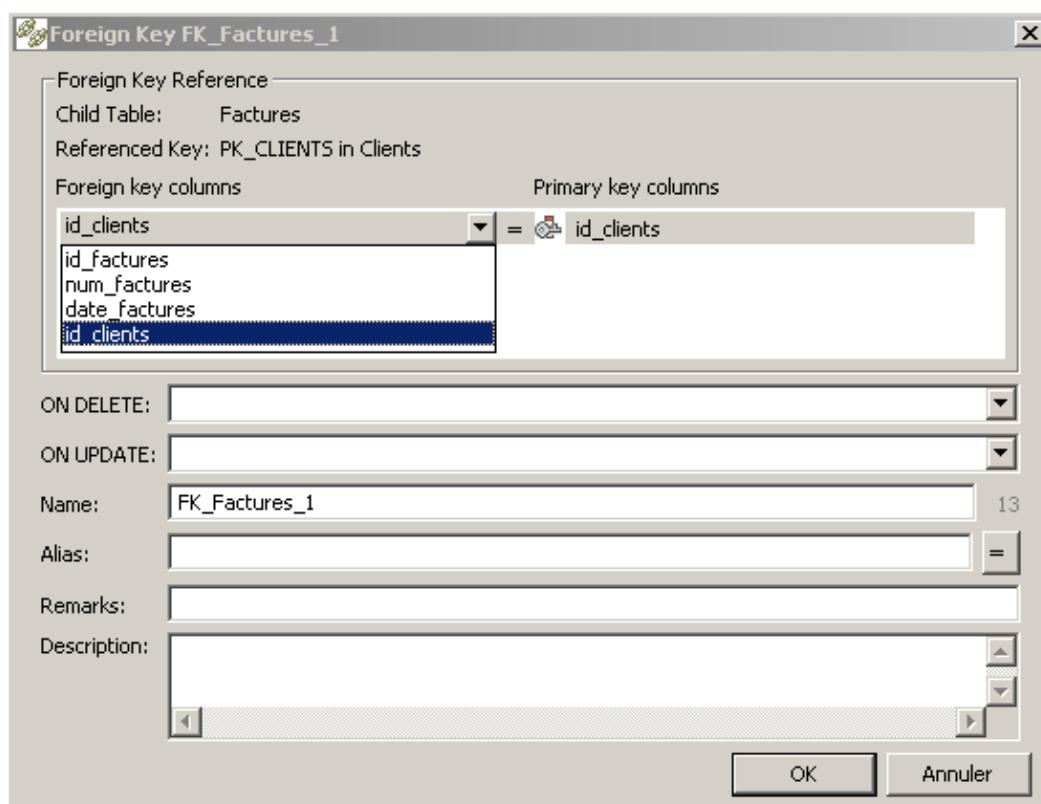
Sélectionnez le champ et cliquez sur le bouton « Ok ». Une fois toutes les informations saisies, il suffit de cliquer sur le bouton « OK ». Le nouvel index apparaît dans la vue « Structure »



Le bouton permet de créer des relations entre deux tables. Pour cela, il faut cliquer sur le bouton, cliquer sur la table contenant la clé étrangère puis cliquer sur la table référence.



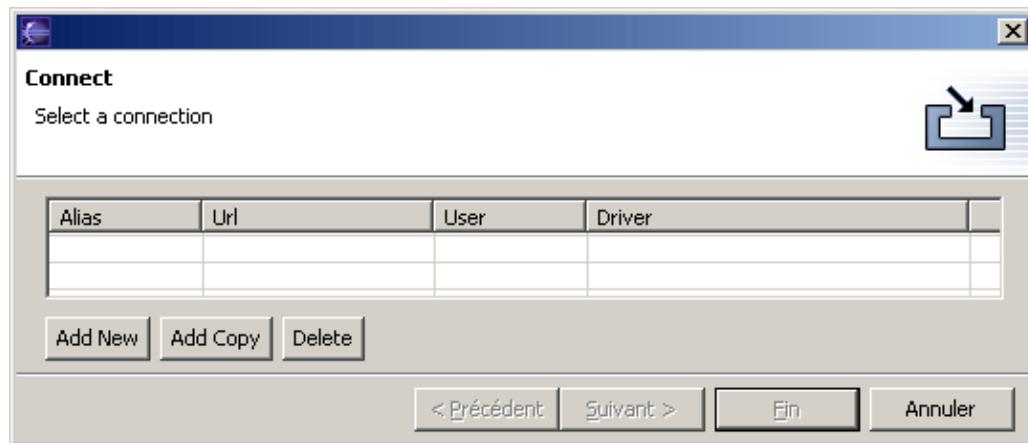
Le lien entre les deux tables apparaît sous la forme d'une flèche bleue entre les deux tables. Le menu contextuel « Edit Foreign Key » permet de modifier la clé étrangère.



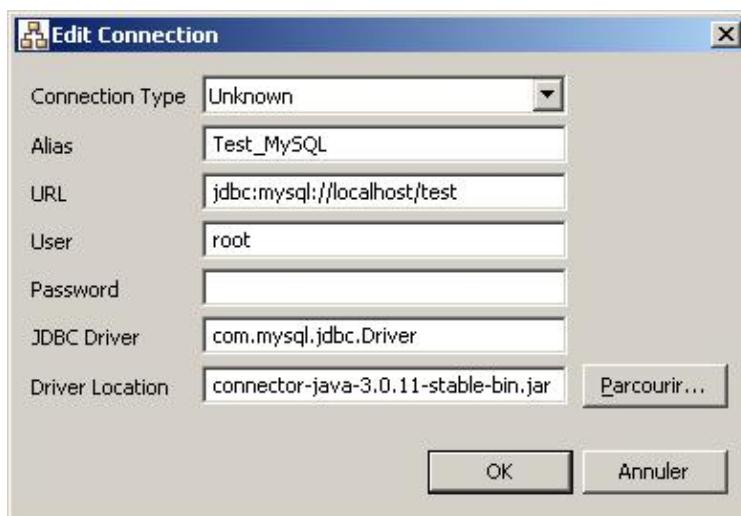
### 31.4.3. Rétro-conception d'un modèle

L'option « Reverse Engineer Database »  du menu contextuel de l'éditeur permet de rétro-concevoir un modèle à partir d'une base de données existante.

Un assistant permet de sélectionner ou de saisir les informations concernant la connexion à la base de données.



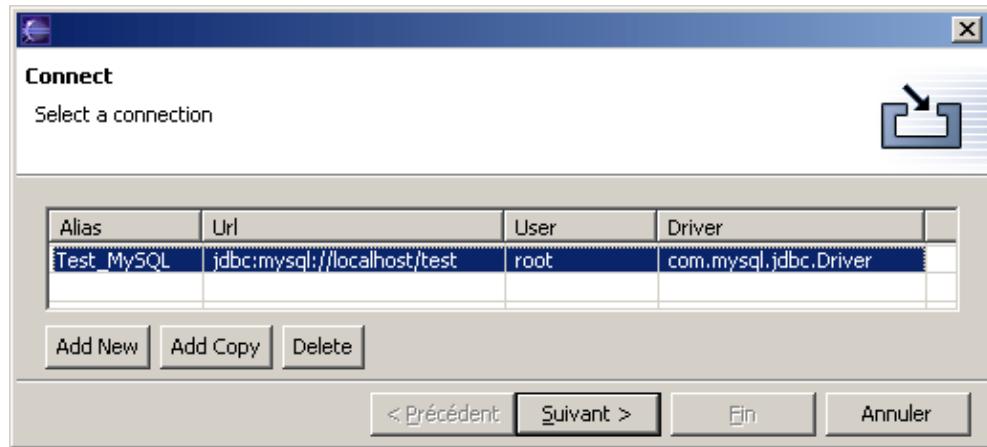
Cliquez sur le bouton « Add New »



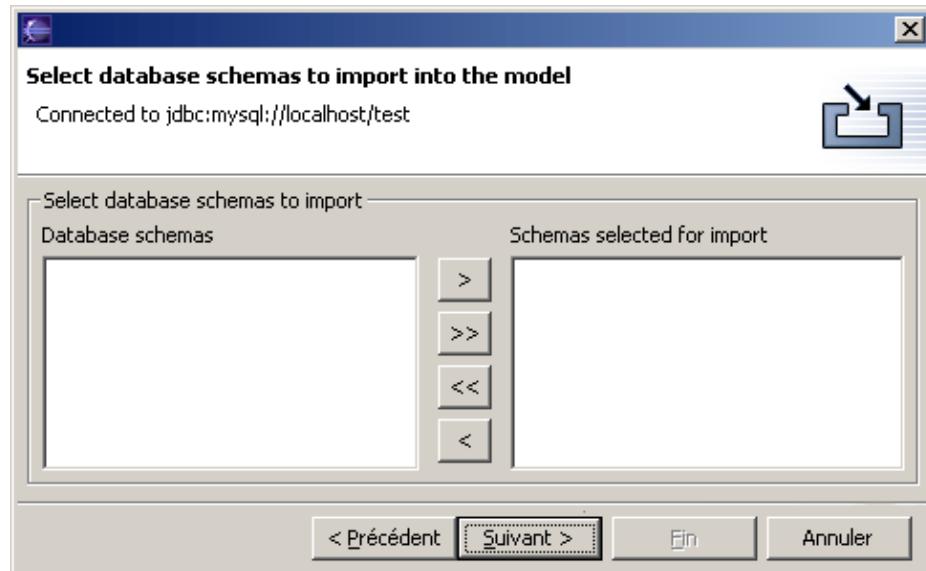
Une boîte de dialogue permet de saisir les informations nécessaires à la connexion :

- le type de connexion peut être choisi dans la liste : si celui ci n'apparaît pas dans la liste, alors il faut sélectionner « unknown »
- l'alias est un nom qui permet de reconnaître la connexion
- user et password sont le nom de l'utilisateur et son mot de passe utilisé pour la connexion
- JDBC Driver est le nom pleinement qualifié de la classe contenant le pilote à utiliser
- Driver Location est le chemin du fichier jar contenant la classe du pilote

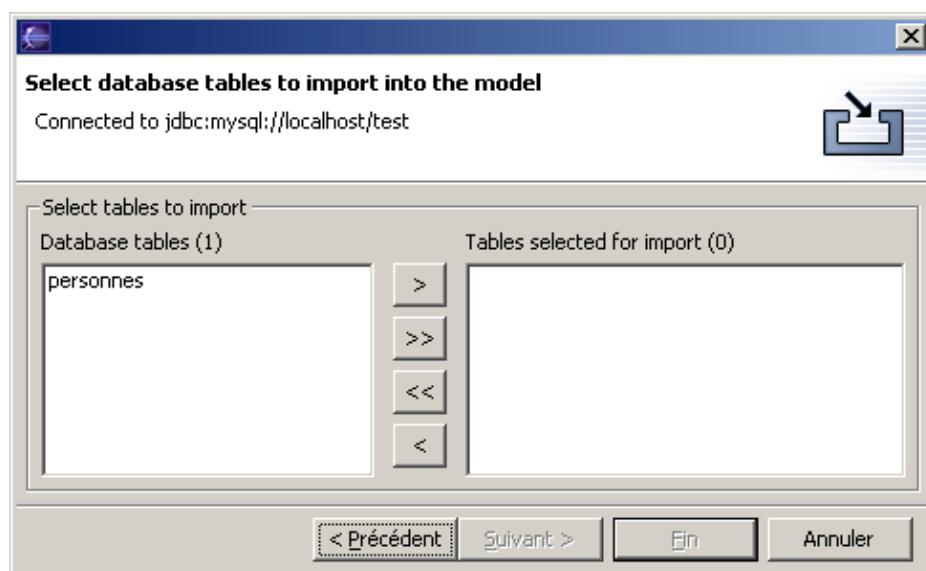
Cliquez sur le bouton « Ok ».



Sélectionnez l'alias et cliquez sur le bouton « Suivant ». La connexion à la base de données s'effectue et l'assistant demande la sélection du schéma. Dans l'exemple de cette section, il n'y en a pas.

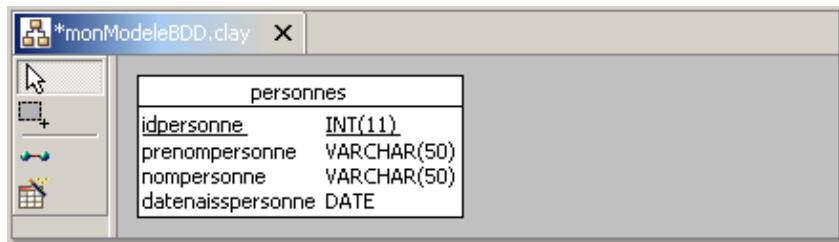


Cliquez sur le bouton « Suivant ». L'assistant demande de sélectionner les tables qui doivent être intégrées dans le modèle.



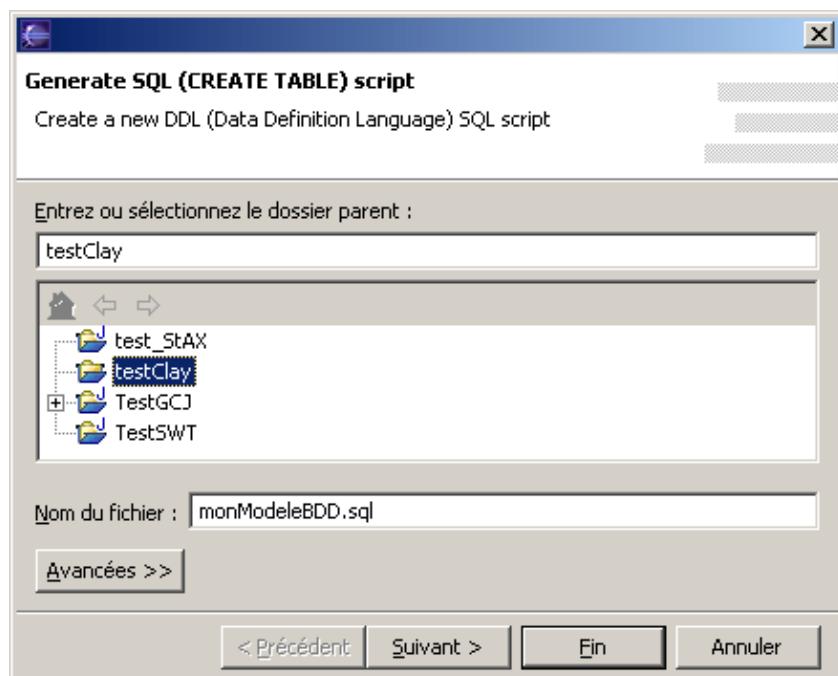
Pour sélectionner une table, il suffit de la sélectionner dans la liste de gauche et de la faire basculer dans la liste de droite en utilisant le bouton « >> ». Une fois la sélection terminée, il suffit de cliquer sur le bouton « Fin ».

L'assistant traite la demande et affiche le résultat des traitements dans l'éditeur.

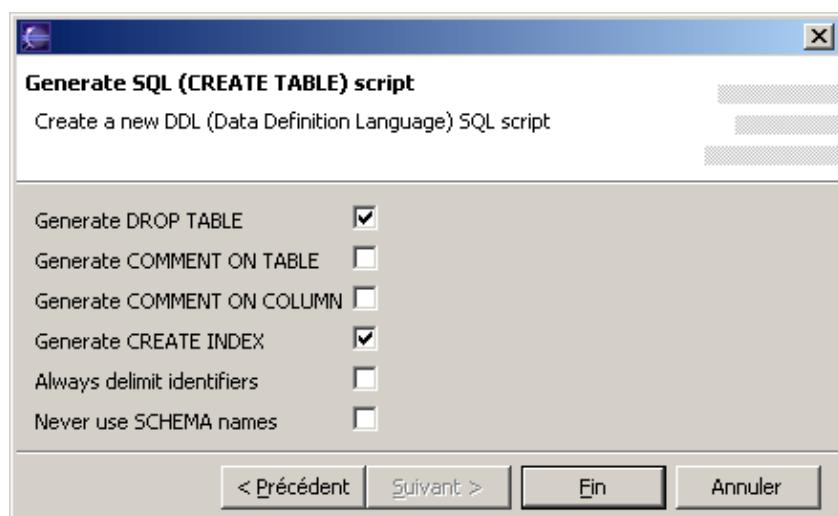


### 31.4.4. Génération du DDL

L'option « Generate SQL (CREATE TABLE) script » permet de demander la création d'un fichier contenant les ordres de création des éléments de la base de données grâce à un assistant.



Il faut sélectionner le dossier qui va contenir le fichier, saisir le nom de ce fichier et cliquer sur le bouton « Suivant ».



La page suivante permet de sélectionner les types d'ordres qui seront générés. Un clic sur le bouton « Fin » permet de générer le fichier et de l'ouvrir dans l'éditeur associé au fichier .sql.

Exemple :

```
DROP TABLE IF EXISTS Factures;
DROP TABLE IF EXISTS Clients;
CREATE TABLE Clients (
    id_clients INT(10) NOT NULL AUTO_INCREMENT
    , nom_clients CHAR(150)
    , PRIMARY KEY (id_clients)
) TYPE=InnoDB;
CREATE TABLE Factures (
    id_factures INT(10) NOT NULL AUTO_INCREMENT
    , num_factures CHAR(10)
    , date_factures DATE
    , id_clients INT(10)
    , UNIQUE UQ_Factures_1 (num_factures)
    , PRIMARY KEY (id_factures)
    , INDEX (id_clients)
    , FOREIGN KEYFK_Factures_1 (id_clients)
        REFERENCES Clients (id_clients)
) TYPE=InnoDB;

CREATE INDEX IX_Factures_1 ON Factures (id_clients ASC);
```

## **Partie 8 : Annexes**

## 28. Annexes

### Annexe A : GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

#### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

#### 1. APPLICABILITY AND définitionS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and

straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3

above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been

approved by an organization as the authoritative définition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the

terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

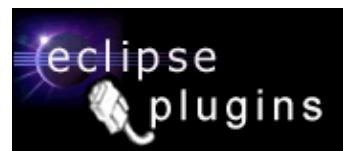
The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## Annexe B : Webographie



<http://www.eclipse.org/>  
le site officiel d'Eclipse



<http://www.eclipse.org/downloads/index.php>  
la page officielle pour le téléchargement d'Eclipse

<http://www.eclipsetotale.com/index.html>  
portail en français consacré au projet Eclipse et aux outils WebSphere Studio d'IBM



<http://eclipse-plug-ins.2y.net/eclipse/index.jsp>

<http://eclipsewiki.swiki.net/>



<http://www.sysdeo.com/eclipse/tomcatPluginFR.html>  
plug-in pour utiliser Tomcat dans Eclipse



<http://www.eclipse-workbench.com/jsp/>  
portail en anglais

[http://www.geocities.com/uwe\\_ewald/dbedit.html](http://www.geocities.com/uwe_ewald/dbedit.html)  
un plug-in qui permet de visualiser le contenu d'une base de données