Contenu

[Installation OpenLDAP 1](#_Toc413065132)

[Prérequis 1](#_Toc413065133)

[Installation du serveur 2](#_Toc413065134)

[Installer LDAP browser 4](#_Toc413065135)

[Installation des plugins pour Eclipse 6](#_Toc413065136)

[FindBugs (analyse statique de bytecode) 6](#_Toc413065137)

[Installation 6](#_Toc413065138)

[PMD (analyse statique de code source) 6](#_Toc413065139)

[Installation 6](#_Toc413065140)

[Checkstyle (analyse statique de formatage de code source) 7](#_Toc413065141)

[Installation 7](#_Toc413065142)

[Cobertura (couverture de code) 7](#_Toc413065143)

[Installation 7](#_Toc413065144)

[Paramétrage de Checkstyle 8](#_Toc413065145)

[Paramétrage de FindBugs 9](#_Toc413065146)

[Préférences Eclipse 10](#_Toc413065147)

# Installation OpenLDAP

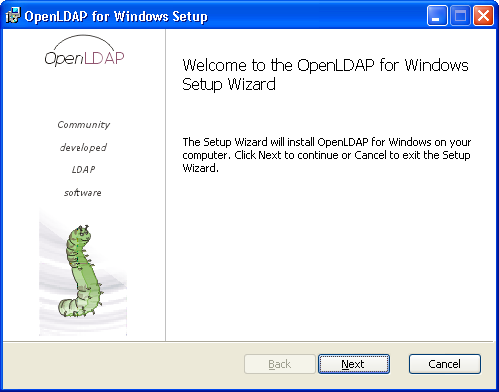
## Prérequis

Les fichiers suivants :

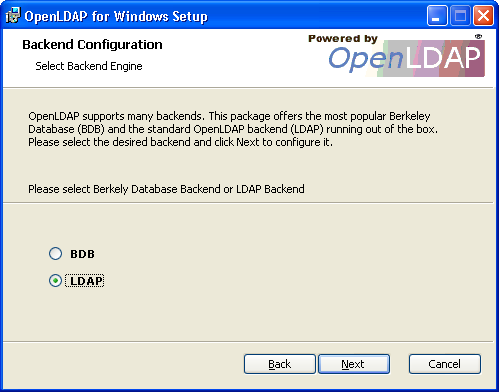
* openldap-for-windows.msi
* ldapbrowser.zip
* 20150302-export-annuaire-univ.ldif
* fichier « slapd.conf »
* dossier contenant les fichiers de configuration pour les règles de qualité : regles-qualite (pmd-cs-findbugs-formatter-cleanup)

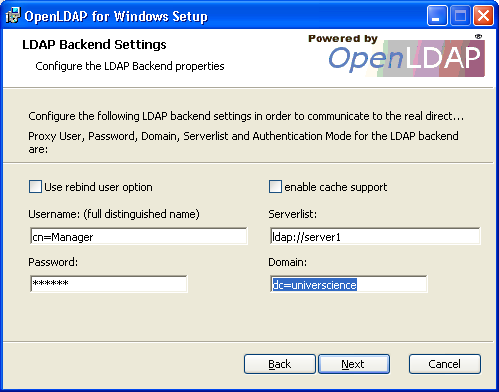
## Installation du serveur

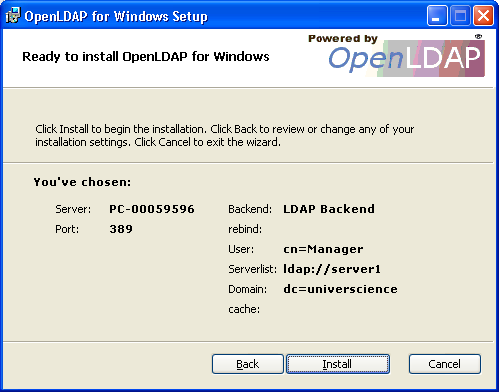
Lancer l’installer « openldap-for-windows.msi »



Cliquer Next > Accepter la licence > Next 4 fois



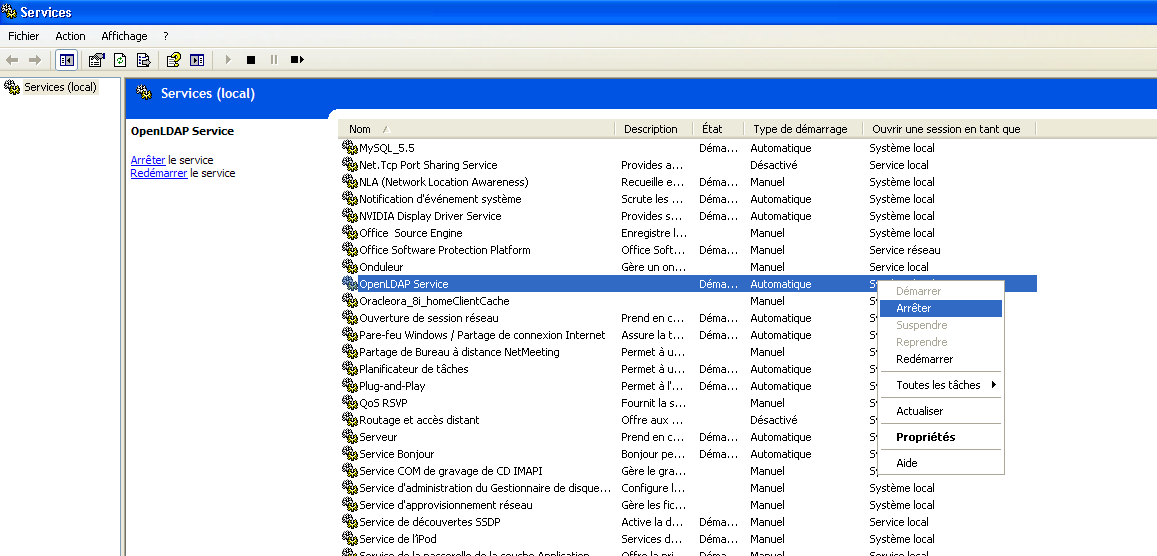
 Mot de passe : secret



Cliquer sur « Install » et enfin sur « Finish ».

Le serveur s’installe comme un service windows. Pour le configurer avant utilisation, arrêter le service windows associé :

Démarrer > Exécuter > services.msc



Se positionner dans le dossier d’installation d’OpenLDAP :

C:\ProgramFiles\OpenLDAP

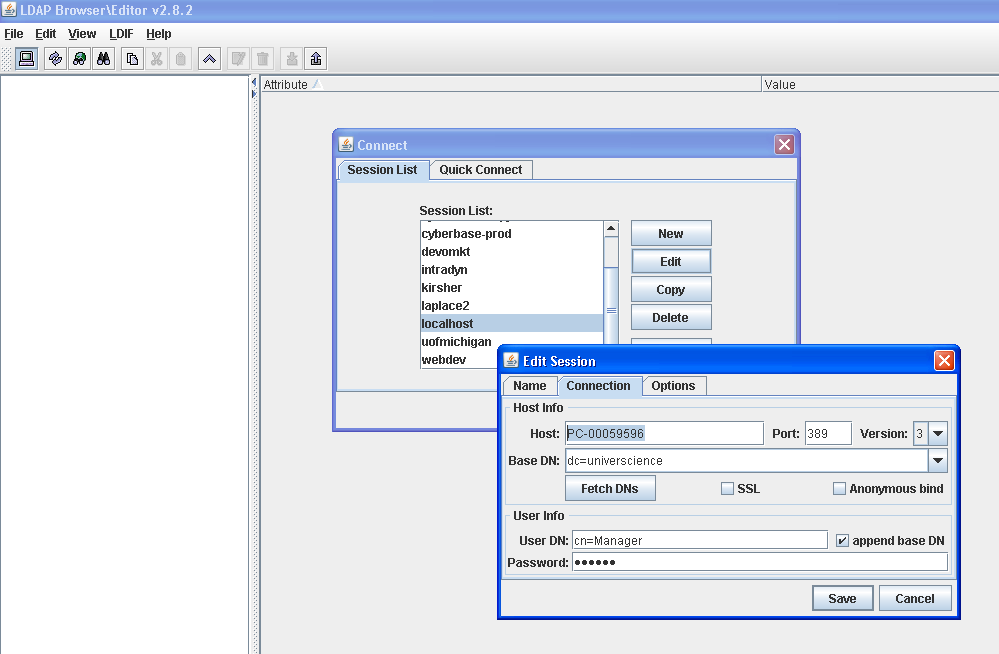
Modifier le contenu du fichier « slapd.conf » par le contenu de ce fichier 

Redémarrer le service windows « OpenLDAP Service »

# Installer LDAP browser

Décompresser le fichier « ldapbrowser.zip » dans un dossier, par exemple « D:\devs\tools\ldap »

Lancer le fichier « D:\devs\tools\ldap\ldapbrowser\lbe.bat »



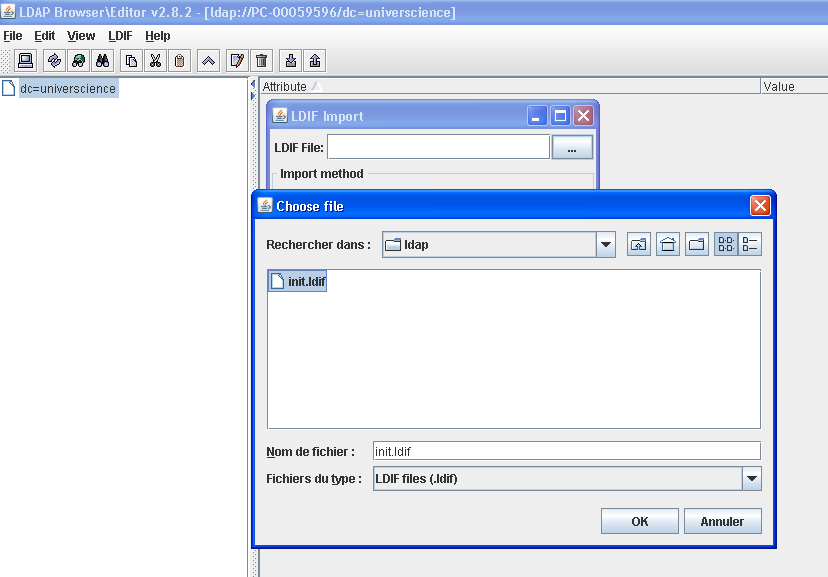
Editer la conf « localhost ». Remplacer le nom de « Host » par le nom de votre PC. Le mot de passe est me même que saisit lors de l’installation du serveur.

Cliquer sur « Save » puis « Connect »

Le browser est à présent connecté au serveur.

Il faut ensuite importer la racine de l’annuaire et quelques entrées initiales.

Menu LDIF > Import : pointer vers le fichier « 20150302-export-annuaire-univ.ldif »



L’annuaire est initialisé.

# Installation des plugins pour Eclipse

## FindBugs (analyse statique de bytecode)

FindBugs est un outil d’analyse statique de bytecode Java, diffusé sous licence LGPL. Comme il n’analyse pas le code source mais le bytecode après compilation, FindBugs ignore le formatage et la syntaxe Java d’origine.

FindBugs se concentre sur les erreurs potentielles comme le référencement de variables nulles et la performance comme les opérations inutiles ou l’utilisation de méthodes plus lentes que d’autres.

### Installation

* Plugin Eclipse : FindBugs Plugin for Eclipse
* Version 1.3.9
* Update site : <http://findbugs.cs.umd.edu/eclipse/>
* Plugin Hudson : [FindBugs Plug-in](http://wiki.hudson-ci.org/display/HUDSON/FindBugs+Plugin)
* Version 4.12
* Plugin Maven : org.codehaus.mojo.findbugs-maven-plugin
* Version 2.2

## PMD (analyse statique de code source)

Diffusé sous licence type BSD, PMD est un outil d’analyse statique de code source Java. Il dispose d’un large ensemble de règles couvrant l’ensemble des patrons d’erreurs typiques que l’on trouve dans les patrimoines Java. Cet outil n’est pas orienté mesure, mais parmi ses règles certaines vérifient que des mesures de volumétrie restent dans des seuils prédéfinis.

### Installation

* Plugin Eclipse : PMD for Eclipse 3
* Version 3.2.6
* Update site : <http://pmd.sourceforge.net/eclipse>
* Plugin Hudson : [PMD Plug-in](http://wiki.hudson-ci.org/display/HUDSON/PMD+Plugin)
* Version 3.9
* Plugin Maven : org.apache.maven.plugins.maven-pmd-plugin
* Version 2.5

## Checkstyle (analyse statique de formatage de code source)

Checkstyle est un outil d’analyse de code source Java diffusé sous licence LGPL. Assez proche de PMD, Checkstyle est comme son nom l’indique plus orienté vérification de « style » (formatage, indentation, nommage, commentaires, etc.). Dans ses évolutions successives Checkstyle a reçu des contrôles de quelques patrons de bug et des métriques, quelques fois d’ailleurs en doublon de PMD. Les deux outils sont cependant largement complémentaires.

### Installation

* Plugin Eclipse : Eclipse-CS
* Version 5.1.1
* Update site : <http://eclipse-cs.sf.net/update/>
* Plugin Hudson : [Checkstyle Plug-in](http://wiki.hudson-ci.org/display/HUDSON/Checkstyle+Plugin)
* Version 3.9
* Plugin Maven : org.apache.maven.plugins.maven-checkstyle-plugin
* Version 2.6

Le plugin Checkstyle présente un défaut qui l’empêche de récupérer son fichier de configuration par http lorsqu’invoqué par la cible site:site.

Contournement en attendant la version stable 2.6 qui corrigera cette anomalie :

Le plugin Checkstyle est invoqué explicitement lors du build par mvn checkstyle:checkstyle au lieu de l’être normalement lors de mvn site:site.

Ce contournement se trouve commenté dans le pom racine du projet csi-maven.

## Cobertura (couverture de code)

Cet outil instrumente l’exécution des batteries de tests unitaires d’un module lors de sa construction. Il essaie de détecter quelles sont les instructions qui ne sont exécutées par aucun de ces tests. Cela peut mettre en évidence du code mort ou du code insuffisamment testé.

Il ne s’agit pas de *preuve* ni de l’exhaustivité des cas de tests, ni de leur exactitude, mais une bonne mesure ayant trait aux caractéristiques de *testabilité* et de *non-résistance au changement*. En effet, disposer d’un large éventail de tests unitaires et de non-régression permet l’évolution et le refactoring en confiance, alors qu’à l’inverse en l’absence de tout test on s’abstiendra de modifier du code.

Dans l’idéal, la couverture de tests devrait être totale (100%), mais dans la pratique on peut fixer comme cible un taux de couverture de 60% à 80% (par exemple l’écriture d’un test pour la levée d’exceptions dans des cas très particuliers peut être coûteuse pour un maigre bénéfice).

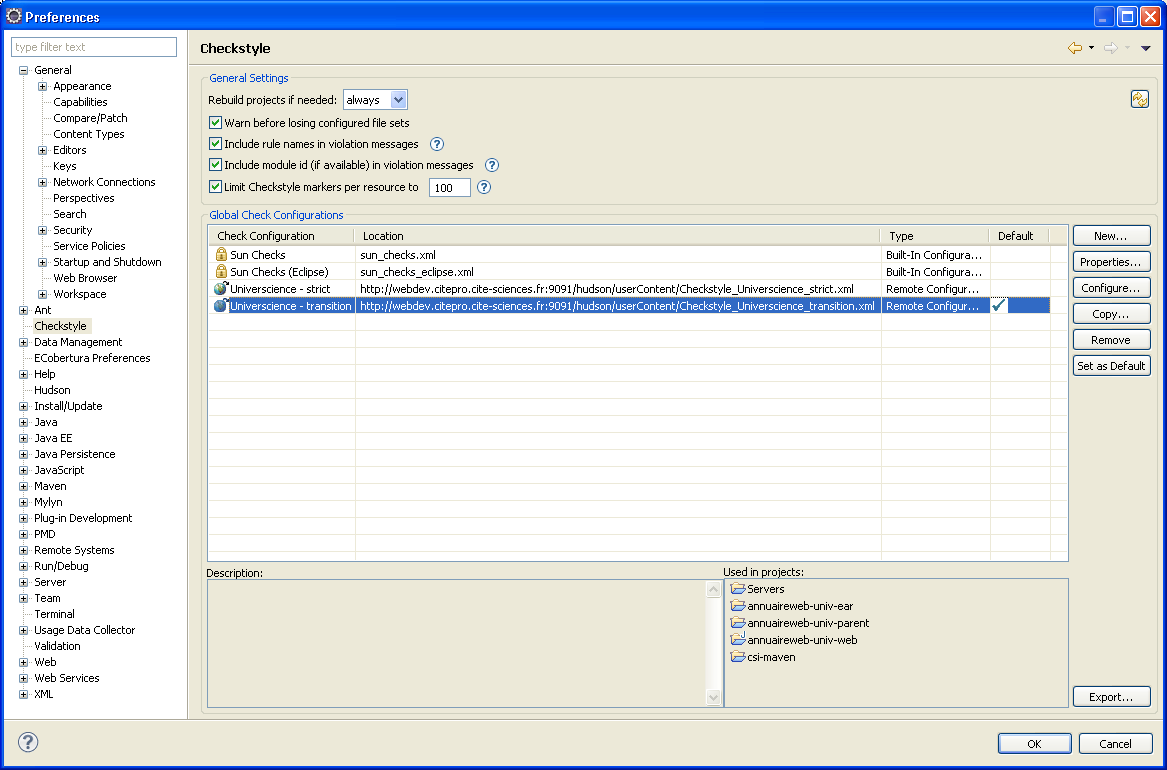
### Installation

* Plugin Eclipse : eCobertura
* Version 0.9.8
* Update site : <http://ecobertura.johoop.de/update/>

Plugin Hudson : [Hudson Cobertura plugin](http://wiki.hudson-ci.org/display/HUDSON/Emma+Plugin)

* Version 1.0
* Plugin Maven : org.codehaus.mojo. cobertura-maven-plugin
* Version 2.4

### Paramétrage de Checkstyle

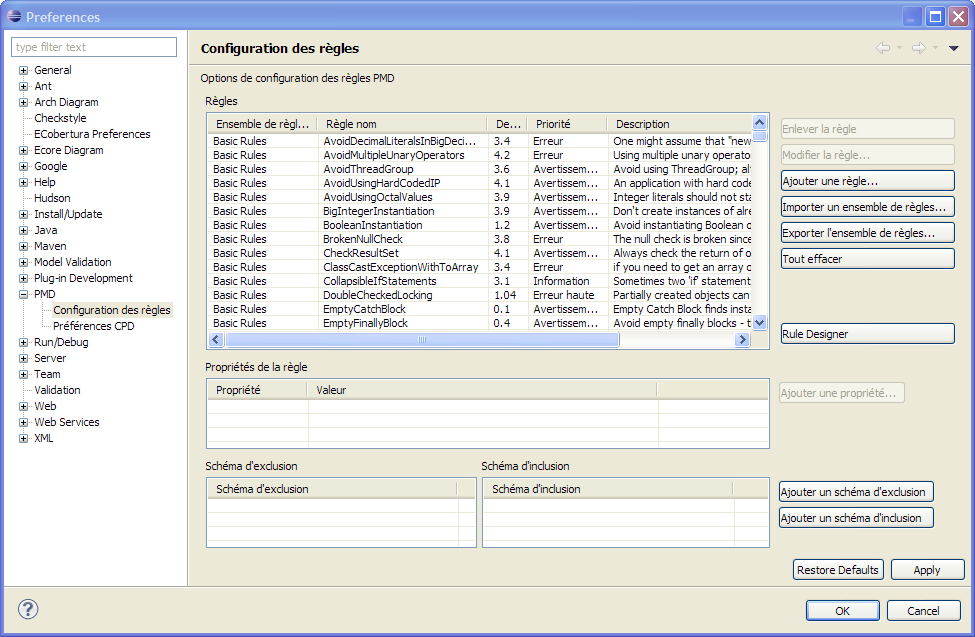


Name : Universcience - strict

Location : <http://webdev.citepro.cite-sciences.fr:9091/hudson/userContent/Checkstyle_Universcience_strict.xml>

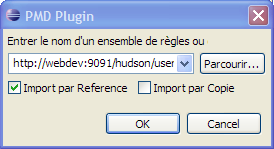
Name : Universcience - transition

Location : <http://webdev.citepro.cite-sciences.fr:9091/hudson/userContent/Checkstyle_Universcience_transition.xml>



Ce n’est pas clairement indiqué par l’interface, mais il est tout à fait possible d’importer directement depuis le serveur en donnant une URL complète dans la fenêtre d’importation :

<http://webdev:9091/hudson/userContent/PMDruleset_Universcience_strict.xml>

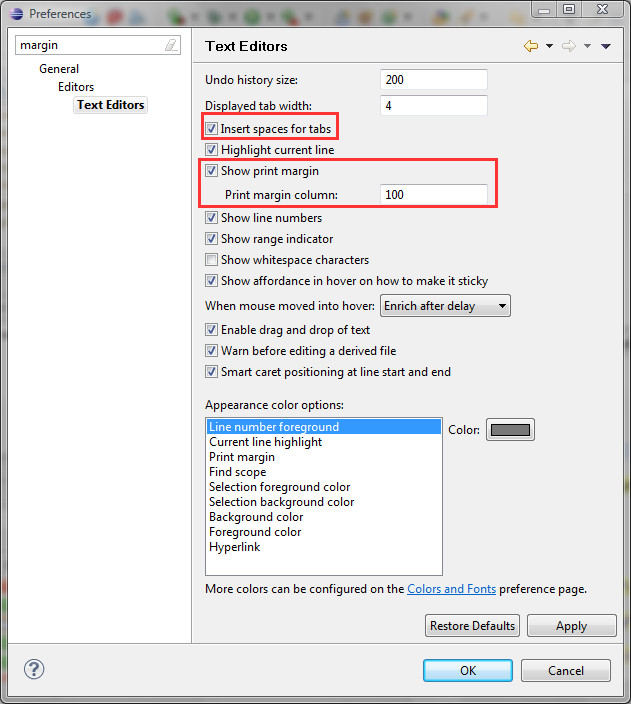


### Paramétrage de FindBugs

Toutes les règles par défaut de FindBugs sont utilisées.

## Préférences Eclipse

Afin de se conformer automatiquement aux conventions de formatage en vigueur dans Eclipse, il faut configurer les éditeurs ainsi :



Il faut également importer les fichiers de préférences suivants :

* EclipseFormatter\_Universcience pour les préférences de formatage de code Eclipse.
* EclipseCleanUp\_Universcience pour les préférences de nettoyage de code, à la volée (*Save Action*) ou à la demande (*Clean Up*)