CNAM Grenoble — Examen Probatoire en Informatique

Étude de l'environnement ouvert de développement intégré Eclipse dans l'optique d'une extension

Nathalie Lépy

Vendredi 1er juillet 2005

Composition du jury

Présidente :	Mme DONZEAU-GOUGE Véronique (CNAM Paris)
Examinateurs :	M. COURTIN Jacques (UPMF Grenoble)
	M. GIRAUDIN Jean-Pierre (CNAM, UPMF Grenoble)
	M. PLISSON André (CNAM Grenoble)
Tuteur :	M. LANG Frédéric (INRIA Rhône-Alpes)

Plan de la présentation

- Contexte et objectifs
- Eclipse en quelques mots
- Architecture de la plate-forme Eclipse
- Plan de travail Eclipse
- Extensions et *plug-ins*
- Illustration – Développement d'un *plug-in*
- Conclusion

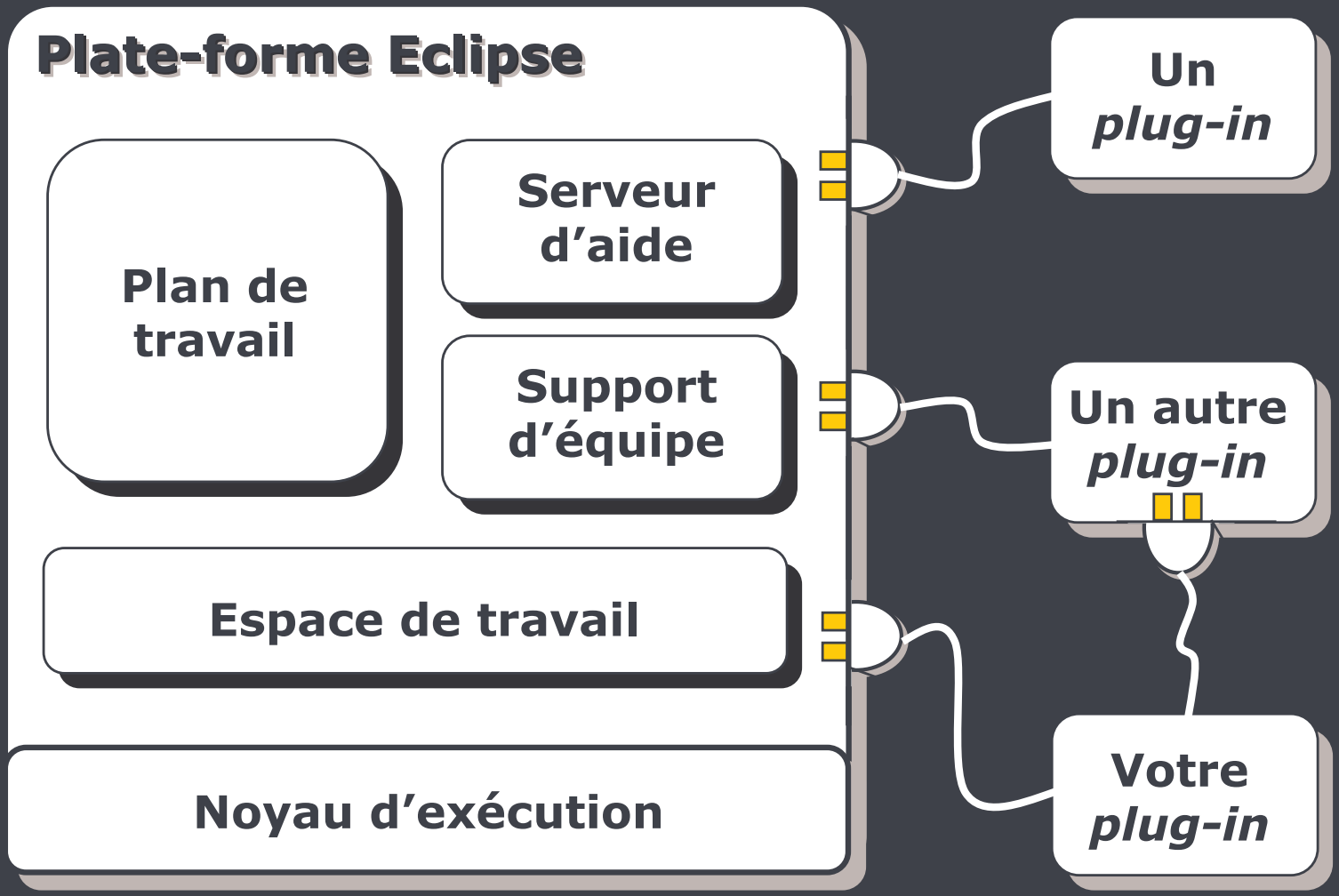
Contexte et objectifs

- Contexte : projet de recherche équipe VASY (*VAlidation de SYstèmes*), INRIA Montbonnot
  - Développement d'outils logiciels de vérification automatique de protocoles d'applications distribuées
  - Utilisation d'Eclipse imposée : interface commune de l'ensemble des outils utilisés
- Objectifs
  - Définir ce qu'est Eclipse : philosophie, organisation, extension
  - Comment intégrer d'autres outils et langages
    - utilisation de nouveaux langages (éditeurs)
    - accéder à des outils existants (actions)

Eclipse en quelques mots

- Plate-forme de développement universelle
  - multilangages : Java, C/C++, C#, Cobol, Python, Perl, Eiffel, PHP, HTML, XML, UML, LaTeX, …
  - multiplateformes : Windows, Linux, Mac OS X, Solaris 8, …
- Environnement de Développement Intégré (EDI) ouvert et évolutif
  - code *open source*
  - extension par *plug-ins*
- Développé par OTI (filiale IBM)

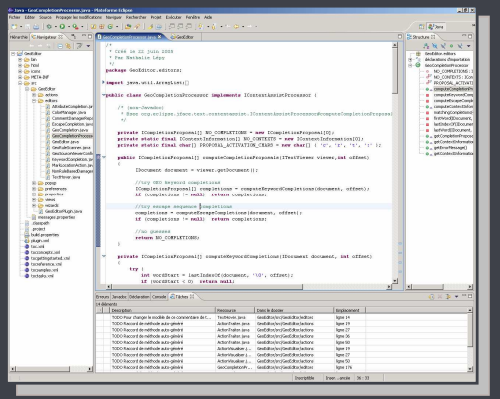
Architecture de la plate-forme Eclipse

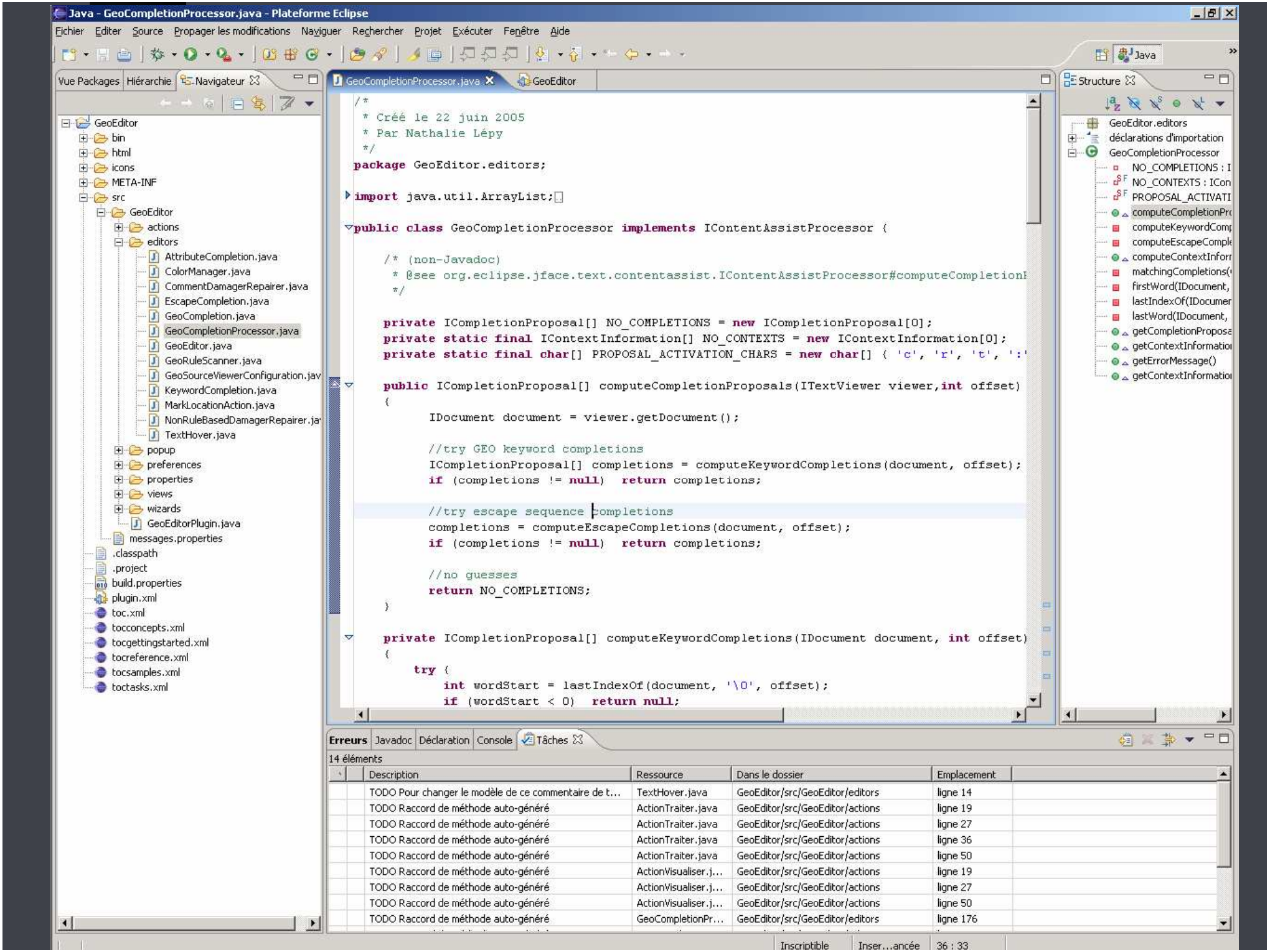


D'après http://eclipse.org/eclipse/presentation/eclipse-slides_files/v3_document.htm

Plan de travail Eclipse

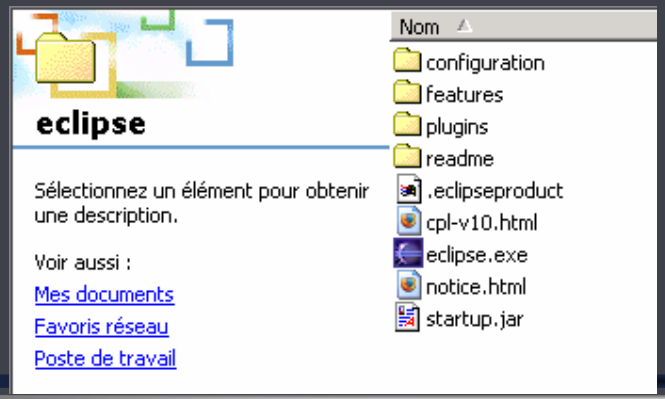
- **Interface Utilisateur (UI) multifenêtres**
  - n'utilise pas MDI (Multi Document Interface)
  - utilise un système de perspectives
- **Trois concepts importants**
  - vue
    - état des développements selon un certain point de vue (Holzner, 2004)
  - éditeur
    - permet de créer et modifier les fichiers ressources
  - perspective
    - ensemble pré-définis de vues et d'éditeurs

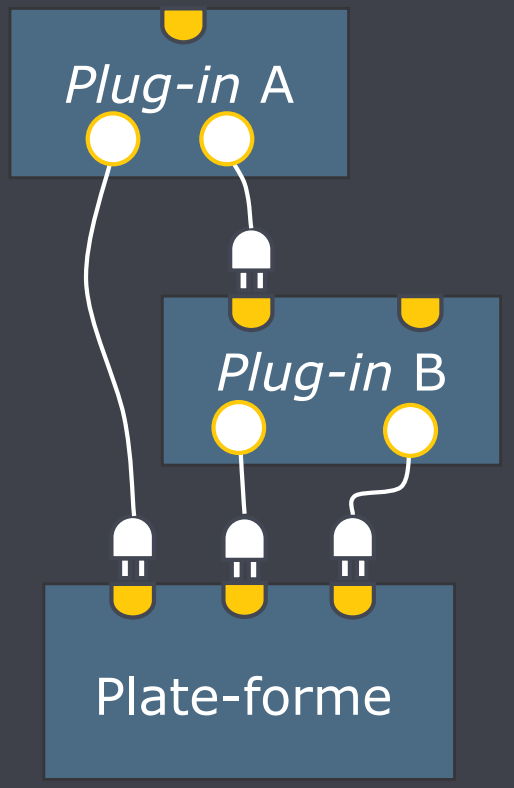



Java - GeoCompletionProcessor.java - Plateforme Eclipse
Fichier Editer Source Propager les modifications Naviguer Rechercher Projet Exécuter Fenêtre Aide
Vue Packages Hiérarchie Navigateur
GeoEditor
bin
html
icons
META-INF
src
GeoEditor
actions
editors
AttributeCompletion.java
ColorManager.java
CommentDamagerRepairer.java
EscapeCompletion.java
GeoCompletion.java
GeoCompletionProcessor.java
GeoEditor.java
GeoRuleScanner.java
GeoSourceViewerConfiguration.jav
KeywordCompletion.java
MarkLocationAction.java
NonRuleBasedDamagerRepairer.ja
TextHover.java
popup
preferences
properties
views
wizards
GeoEditorPlugin.java
messages.properties
.classpath
.project
build.properties
plugin.xml
toc.xml
tocconcepts.xml
tocgettingstarted.xml
tocreference.xml
tocsamples.xml
toctasks.xml
GeoCompletionProcessor.java
GeoEditor
/*
* Créé le 22 juin 2005
* Par Nathalie Lépy
*/
package GeoEditor.editors;
import java.util.ArrayList;
public class GeoCompletionProcessor implements IContentAssistProcessor {
/* (non-Javadoc)
* @see org.eclipse.jface.text.contentassist.IContentAssistProcessor#computeCompletionI
*/
private ICompletionProposal[] NO_COMPLETIONS = new ICompletionProposal[0];
private static final IContextInformation[] NO_CONTEXTS = new IContextInformation[0];
private static final char[] PROPOSAL_ACTIVATION_CHARS = new char[] { 'c', 'r', 't', ':'
public ICompletionProposal[] computeCompletionProposals(ITextViewer viewer,int offset)
{
IDocument document = viewer.getDocument();
//try GEO keyword completions
ICompletionProposal[] completions = computeKeywordCompletions(document, offset);
if (completions != null) return completions;
//try escape sequence completions
completions = computeEscapeCompletions(document, offset);
if (completions != null) return completions;
//no guesses
return NO_COMPLETIONS;
}
private ICompletionProposal[] computeKeywordCompletions(IDocument document, int offset)
{
try {
int wordStart = lastIndexOf(document, '\0', offset);
if (wordStart < 0) return null;
Structure
GeoEditor.editors
déclarations d'importation
GeoCompletionProcessor
NO_COMPLETIONS : I
NO_CONTEXTS : ICon
PROPOSAL_ACTIVATI
computeCompletionPr
computeKeywordComp
computeEscapeCompl
computeContextInfor
matchingCompletions(
firstWord(IDocument,
lastIndexOf(IDocumer
lastWord(IDocument,
getCompletionPropose
getContextInformatio
getErrorMessage()
getContextInformatio
Erreurs Javadoc Déclaration Console Tâches
14 éléments
Description Ressource Dans le dossier Emplacement
TODO Pour changer le modèle de ce commentaire de t... TextHover.java GeoEditor/src/GeoEditor/editors ligne 14
TODO Raccord de méthode auto-généré ActionTraiter.java GeoEditor/src/GeoEditor/actions ligne 19
TODO Raccord de méthode auto-généré ActionTraiter.java GeoEditor/src/GeoEditor/actions ligne 27
TODO Raccord de méthode auto-généré ActionTraiter.java GeoEditor/src/GeoEditor/actions ligne 36
TODO Raccord de méthode auto-généré ActionTraiter.java GeoEditor/src/GeoEditor/actions ligne 50
TODO Raccord de méthode auto-généré ActionVisualiser.j... GeoEditor/src/GeoEditor/actions ligne 19
TODO Raccord de méthode auto-généré ActionVisualiser.j... GeoEditor/src/GeoEditor/actions ligne 27
TODO Raccord de méthode auto-généré ActionVisualiser.j... GeoEditor/src/GeoEditor/actions ligne 50
TODO Raccord de méthode auto-généré GeoCompletionPr... GeoEditor/src/GeoEditor/editors ligne 176
Inscriptible
Inser...ancée
36 : 33

Extensions et *plug-ins* (1/4)

- Eclipse est structuré en *plug-ins*
  - *Plug-in* = ensemble d'extensions et points d'extension
  - Extension/contribution = code étendant les fonctionnalités d'un *plug-in existant*
  - Point d'extension = point de connexion du *plug-in*
- Utilisation de *plug-ins* existants
  - Installation : esprit « *plug-and-play* »



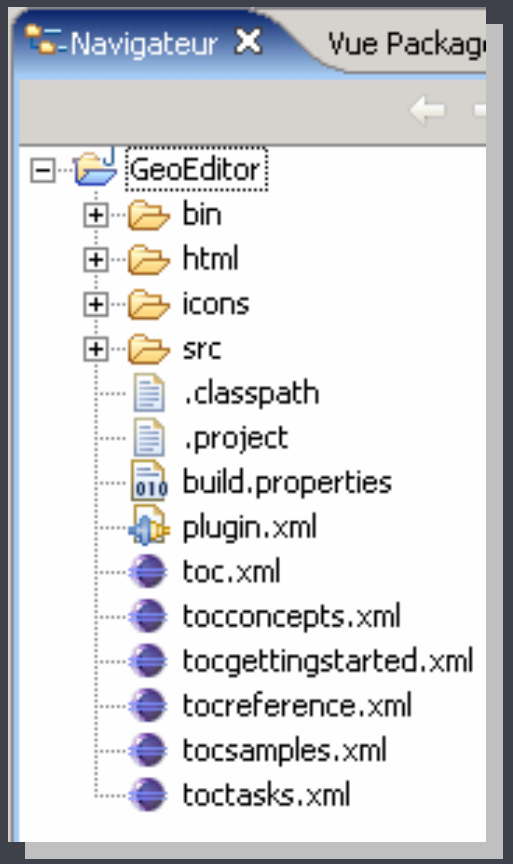


Extension/contribution
Point d'extension

D'après http://www.afceurope.com/club-java/slides/presentation-club-java.ppt

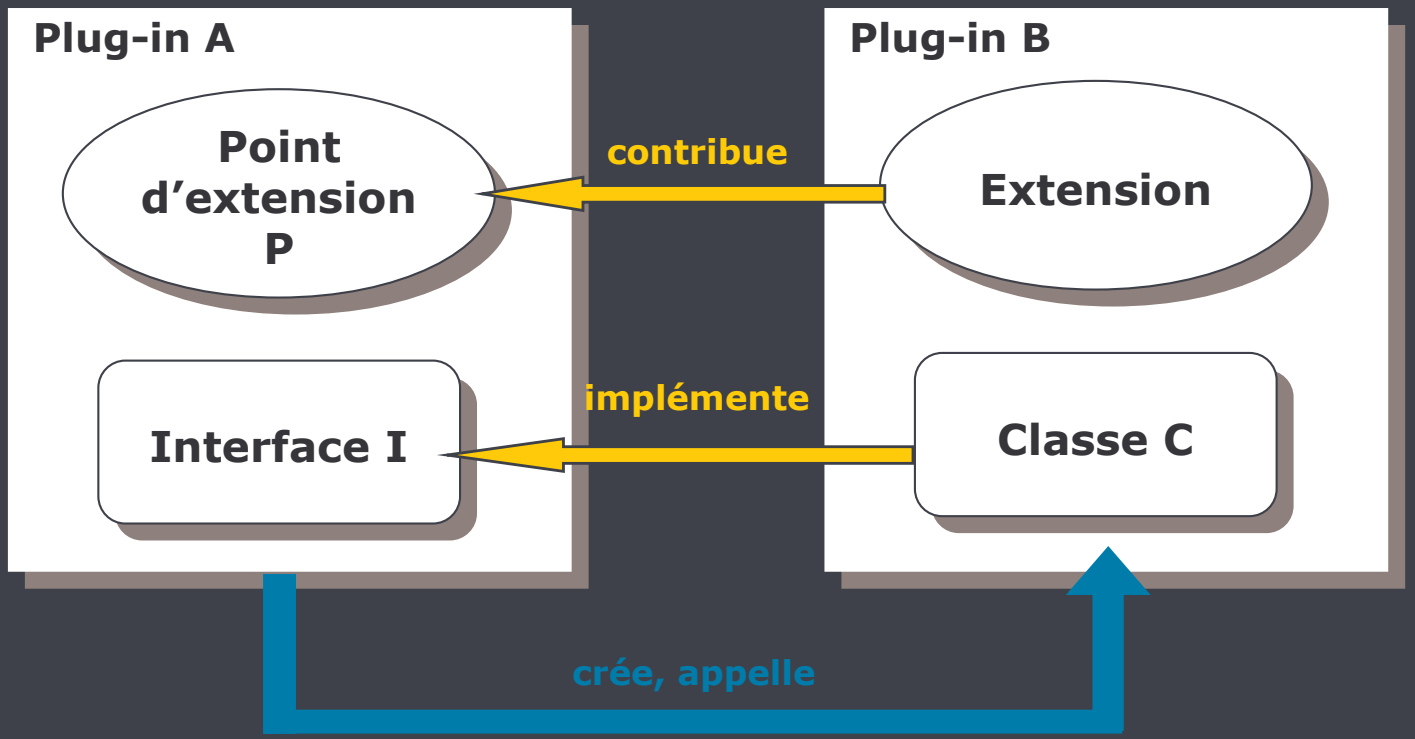
Extensions et *plug-ins* (2/4)

- Structure d'un projet *plug-in*
  - Fichier manifeste *plugin.xml*
    - déclarations des extensions et points d'extension
  - Code Java (sources et binaires)
    - implémentation
  - Autres ressources (images, fichiers html, …)



Extensions et *plug-ins* (3/4)

- Interconnexion des *plug-ins*



D'après http://eclipse.org/eclipse/presentation/eclipse-slides_files/v3_document.htm

Extensions et *plug-ins* (4/4)

- Extraits de fichiers manifestes (*plugin.xml*)

```
<plugin id="org.eclipse.ui">
    <extension-point
        name="Prefs"
        id="preferencepages"
        schema="schema/prefs.exsd"/>
    …
</plugin>
```

Plug-in A

définition du point d'extension

définition de la contribution / extension

```
<plugin id="myPlugin">
    <extension point="org.eclipse.ui.preferencepages">
        <page id="com.example.myprefpage"
            icon="icons/image.gif"
            title="My title"
            class="com.example.mywizard">
        </page>
    </extension>
  …
</plugin>
```

Plug-in B

Illustration – Développement d'un *plug-in* (1/4)

- Nouveau langage : GEO
  - Description de figures géométriques
    - carré : taille, positionX, positionY, codeCoul, rempli ;
    - triangle : taille1, taille2, taille3, positionX, positionY, codeCoul, rempli ;
    - rond : diamètre, positionX, positionY, codeCoul, rempli ;
- Outils à intégrer
  - Compilateur
  - Visualisation des figures
- Objectifs, fonctionnalités
  - Développer un nouvel éditeur
  - Ajouter des actions
  - Développer un assistant de création de fichiers
  - Compilation à la saisie
  - Aide globale
  - Aide contextuelle (pop-up)
  - Ajouter une vue
  - Modifier des vues existantes
  - Associer une perspective

Illustration – Développement d'un *plug-in* (2/4)

Méthodes

- tout faire ou presque
- partir d'un projet similaire
- utiliser PDE (*Plug-in Development Environment*)
  - nombreux assistants
  - modèles
  - éditeur de *plug-ins*

Étapes

- Création du projet GeoEditor
- Fichier manifeste *plugin.xml*

```
<plugin
    id="GeoEditor"
    name="GeoEditor Plug-in"
    class="GeoEditor.GeoEditorPlugin" >
    <requires>
        <import plugin="org.eclipse.ui"/>
        <import plugin="org.eclipse.core.runtime"/>
        ...
    </requires>
    <runtime>
        <library name="GeoEditor.jar"/>
    </runtime>
    <extension
       point="org.eclipse.ui.editors">
        <editor
            ...
            class="GeoEditor.editors.GeoEditor"
        >
        ...
    </extension>
    <extension
        point="org.eclipse.ui.actionSets">
        ...
    </extension>
    ...
</plugin>
```

Illustration – Développement d'un *plug-in* (3/4)

- Développer le code Java
  - Classes pour gérer les actions
    - exécution du compilateur (action Traiter)
    - exécution du visualiseur (action Visualiser)
  - Classes pour gérer les fonctionnalités de l'éditeur
    - coloration syntaxique
    - complétion et assistance au contenu (*content assist*)
    - affichage d'informations « volantes » (*text hover*)
  - Etc.

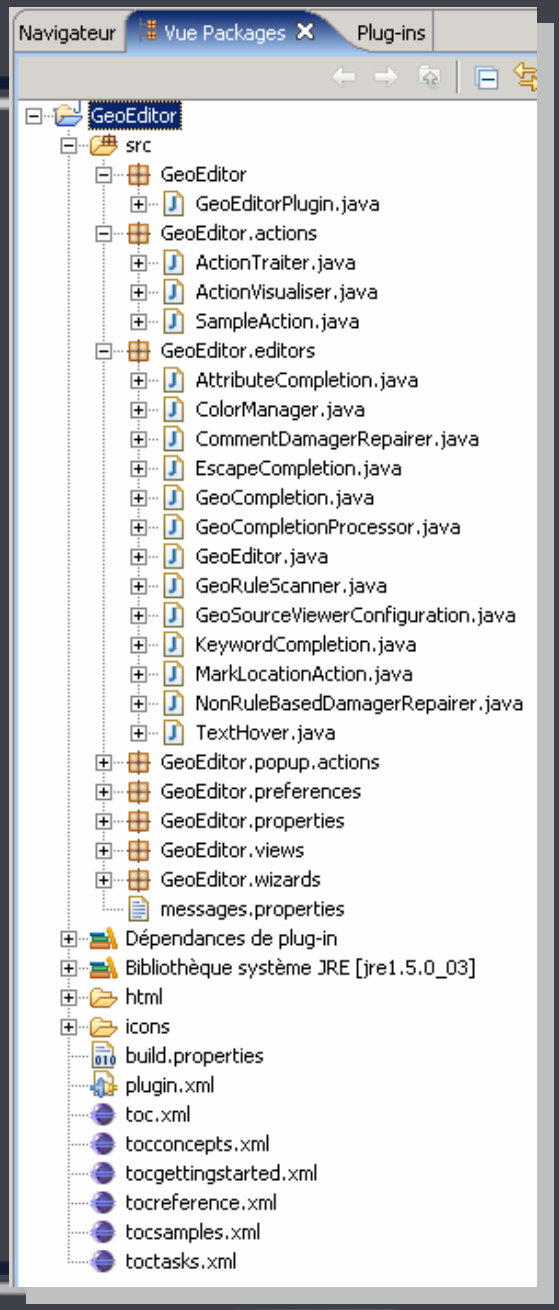
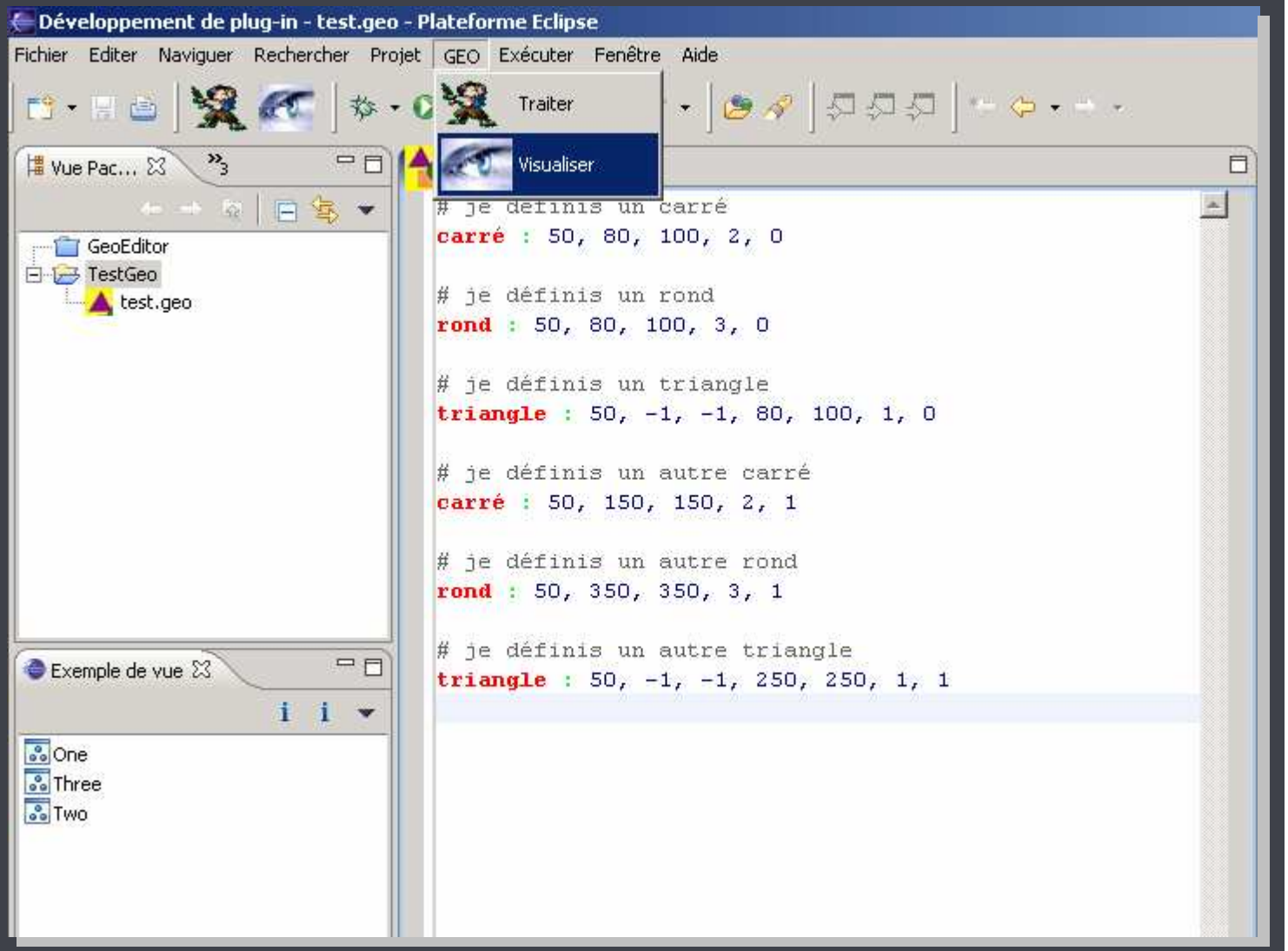


Illustration – Développement d'un *plug-in* (4/4)

- Tester le *plug-in* (plan de travail d'exécution)
- Déployer le *plug-in* (utiliser PDE)



Conclusion

- Eclipse = IDE puissant et complexe
  - environnement de programmation dans un langage connu
  - riche en fonctionnalités
  - personnalisation de l'environnement pour de nouveaux langages et fonctionnalités
- Répond aux objectifs concrets fixés
  - conception d'un éditeur pour un nouveau langage
  - intégration possible des outils existants
- « Prise en main » moyennement facile
  - compréhension rapide de l'organisation
  - MAIS architecture de *plug-ins* complexe

Références bibliographiques

- Holzner S. (2004). *Eclipse*. Paris : Éditions O'Reilly
  - utilisation de PDE
- Arthorne J., Laffra C. (2004). *Official Eclipse 3.0 FAQs*. Gamma E., Nackman L., Wiegand J. Eds. The Eclipse Series, Addison-Wesley
  - si on trouve la question, on a la réponse…
  - complet
- Aide en ligne Eclipse
  - chercher par mot-clés plutôt que par la table des matières
  - complet
- Gamma E., Beck K. (2004). *Eclipse: principes, patterns et plug-ins*. CampusPress
  - nécessite une bonne maîtrise de la terminologie
  - un zeste de philosophie
  - orienté contributions à org.eclipse