

Programmation orientée objet en langage JAVA

Présentation d'Eclipse

Claude Duvallet

Université du Havre
UFR Sciences et Techniques
25 rue Philippe Lebon - BP 540
76058 LE HAVRE CEDEX
Claude.Duvallet@gmail.com
<http://litis.univ-lehavre.fr/~duvallet/>

Présentation de la plateforme Eclipse

- 1 Objectifs
- 2 Création d'un projet JAVA avec Eclipse
- 3 Création de la JavaDOC
- 4 La perspective DEBUG

Introduction à Eclipse

- Prévue pour fournir une plateforme ouverte de développement :
 - Fonctionne sur un grand nombre de systèmes d'exploitation.
 - Interface graphique très performante et facilitant le développement d'applications.
- Indépendance du langage de programmation :
 - Permet sans restriction l'utilisation plusieurs types de contenus.
 - HTML, Java, C, JSP, EJB, XML, GIF,...
- Facilite l'intégration de nouveaux outils :
 - Au niveau de l'interface et en profondeur.
 - Ajout de nouveaux outils pour les produits installés.
- Attire une grande communauté de développeurs :
 - Y compris des éditeurs de logiciels indépendants.
 - Capitalise la popularité de Java pour l'écriture de nouveaux Outils.

Genèse d'Eclipse

- Eclipse créé par l'OTI et les équipes d'IBM chargé pour les produits IDE :
 - IBM VisualAge / Smalltalk (Smalltalk IDE)
 - VisualAge IBM / Java (Java IDE)
 - VisualAge IBM / Micro Edition (Java IDE)
- Initialement composé de 40 développeurs à plein temps.
- Des équipes dispersées géographiquement de développement.
 - Ottawa OTI, Minneapolis OTI, Zurich OTI, IBM Toronto, OTI Raleigh, RTP, IBM Saint-Nazaire (France).
- Effort transition en projet open source
 - IBM a donnée le code de base d'Eclipse : Plate-forme, JDT, PDE.

Historique

1999

Mars Début des travaux sur Eclipse au sein de OTI/IBM.

2000

Mars Premiers composants Eclipse.

2001

Mars Ouverture du site <http://www.eclipsecorner.org/>.

Juin Version Eclipse 0.9.

Octobre Version Eclipse 1.0.

Novembre IBM fait don du code source d'eclipse.

Ouverture du site <http://www.eclipse.org/>

2002

Juin Eclipse 2.0

Septembre Eclipse 2.0.1

Novembre Eclipse 2.0.2

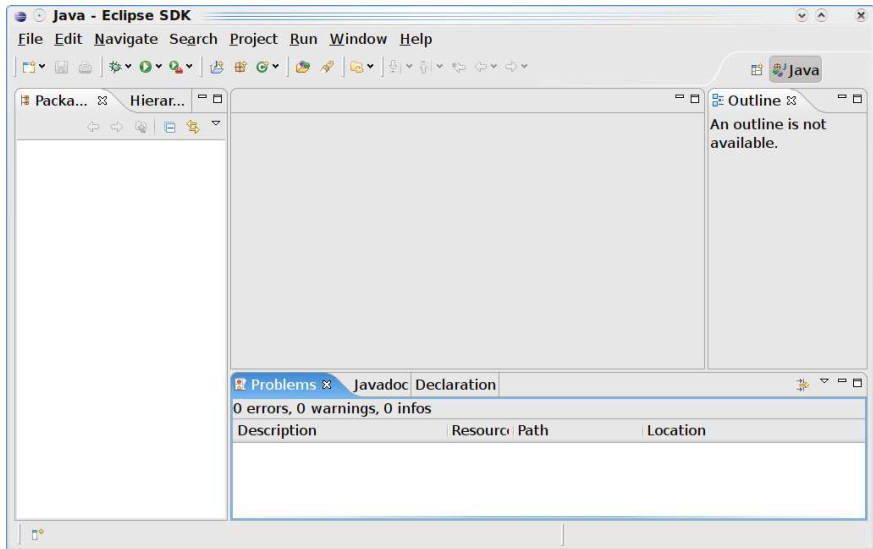
2003

Mars Eclipse 2.1

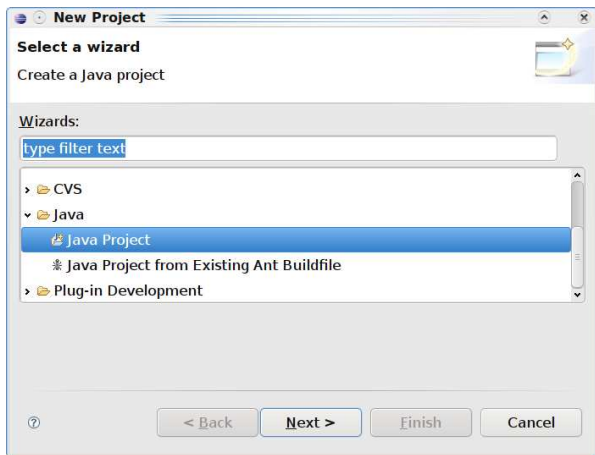
Eclipse : une plateforme de développement JAVA et plus

- Il fournit un ensemble d'outils permettant de créer facilement des classes.
- Possibilité de créer des classes JAVA comportant :
 - une méthode principale "main".
 - des méthodes hérités.
 - des accesseurs "get" et "set".
- Existence de nombreux plugins permettant de développer dans d'autres langages que JAVA.

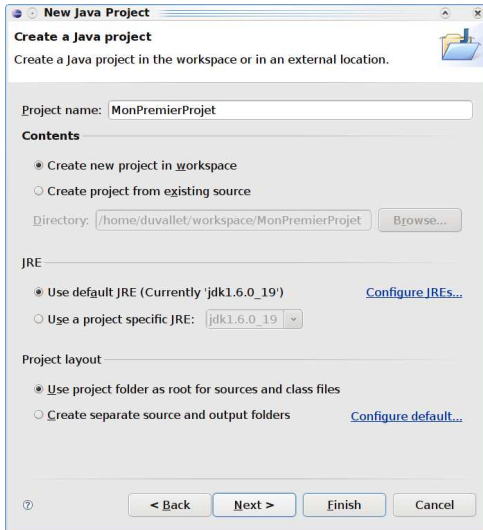
Présentation de la plateforme Eclipse



Création d'un projet JAVA



Nommer le projet et finaliser sa création



Création d'une nouvelle classe principale

Java Class

⚠ The use of the default package is discouraged.

Source folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)
 Constructors from superclass
 Inherited abstract methods

Do you want to add comments as configured in the [properties](#) of the current project?

Résultats de la génération

The screenshot displays the Eclipse IDE interface with the following components:

- Project Explorer:** Shows a project named 'MonPremierProjet' containing a package '(default package)' with the file 'MaClasse.java' selected.
- Editor:** Displays the source code for 'MaClasse.java':

```
public class MaClasse {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
    }  
  
}
```
- Outline:** Shows the class structure with 'MaClasse' expanded to show the 'main(String[])' method.
- Problems:** Shows '0 errors, 0 warnings, 0 infos'.
- Declaration:** Shows a table with columns 'Description', 'Resource Path', and 'Location'.

Lancement de l'application

The screenshot shows the Eclipse IDE interface. The main editor displays the source code of `MaClasse.java`. A context menu is open over the code, with the `Run As` option selected, and a sub-menu showing `1 Java Application` (with the keyboard shortcut `Shift+Alt+X`) and `Run...`. The code in the editor is as follows:

```
class MaClasse {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println ("Bonjour à tous !");  
    }  
}
```

The Outline view on the right shows the class structure with `main(String[])` selected. The Problems view at the bottom indicates 0 errors, 0 warnings, and 0 infos. The status bar at the bottom shows 'Writable', 'Smart Insert', and '9 : 49'.

Résultat de l'exécution

The screenshot shows the Eclipse IDE interface with the following components:

- Top Bar:** Title bar "Java - MaClasse.java - Eclipse SDK" and menu bar "File Edit Source Refactor Navigate Search Project Run Window Help".
- Left Panel (Project Explorer):** Shows the project structure: "MonPremierProjet" containing "(default package)" with "MaClasse.java" and "JRE System Library [jdk...]".
- Center Editor:** Displays the source code of `MaClasse.java`:

```
public class MaClasse {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println ("Bonjour à tous !");  
    }  
}
```
- Right Panel (Outline):** Shows the class structure with "MaClasse" containing the method "main(String[])".
- Bottom Panel (Console):** Shows the execution output: `<terminated> MaClasse [Java Application] /usr/local/jdk1.6.0_19/bin/java (9 avr. 10 13:2
Bonjour à tous !`

Création de la JavaDOC

The screenshot shows the Eclipse IDE interface. The main editor displays the source code of `MaClasse.java` with the following content:

```
/**
 * Description de la classe
 * @author Claude Duvallet
 *
 */
public class MaClasse {

    /**
     * Il s'agit de la méthode principale.
     * @param args paramètre permettant de récupérer les
     * arguments saisis en ligne de commande.
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println ("Bonjour à tous !");
    }
}
```

The Outline view on the right shows the class structure:

- MaClasse
 - main(String[])

The JavaDoc view at the bottom shows the generated documentation:

void MaClasse.main(String[] args)

Il s'agit de la méthode principale.

Parameters:
args paramètre permettant de récupérer les arguments saisis en ligne de commande.

At the bottom of the IDE, the status bar indicates: Writable | Smart Insert | 12 : 8

La perspective DEBUG (1/2)

- Elle offre plusieurs vues qui sont spécifiques au débogage :
 - La vue "Débogage" qui affiche sous la forme d'une arborescence, les différents processus en cours d'exécution ou terminés.
 - La vue "Variables" qui affiche les variables utilisées dans les traitements en cours de débogage.
 - La vue "Points d'arrêts" qui affiche la liste des points d'arrêts définis dans l'espace de travail.
 - La vue "Expressions" qui permet d'inspecter une expression en fonction du contexte des données en cours d'exécution.

La perspective DEBUG (2/2)

The screenshot displays the Eclipse IDE in the Debug perspective. The main editor shows the source code of `MaClasse.java` with a breakpoint at line 19. The Console shows the output of the program: `Bonjour a tous !` followed by `0123`. The Breakpoints view shows a breakpoint at line 19. The Outline view shows the class structure.

```
MaClasse.java
* @author duvallet
*
*/
public class MaClasse {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String che=Test;
        System.out.println ("Bonjour a tous !");
        for (int i=0; i<10; i++)
            System.out.print(i);
    }
}
```

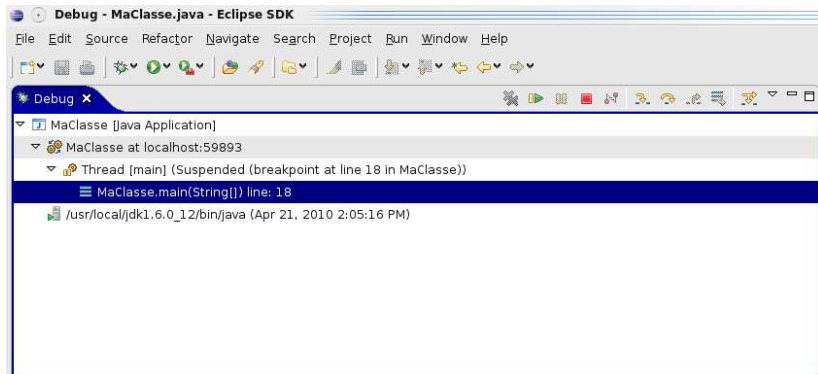
Console
MaClasse [Java Application] /usr/local/jdk1.6.0_12/bin/java (Apr 21, 2010 2:25:47 PM)
Bonjour a tous !
0123

Breakpoints
MaClasse [line: 17] [hit count: 1] - main(String[])
MaClasse [line: 19] [hit count: 5] - main(String[])

Outline
MaClasse
main(String[])

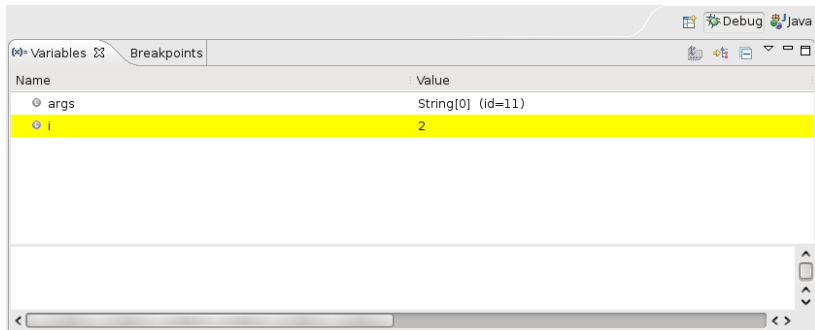
La vue "Débogage"

- Elle affiche les différents processus en cours d'exécution.
- Elle arrête l'exécution lors de la rencontre d'un point d'arrêt ou encore lorsqu'une exception se produit.



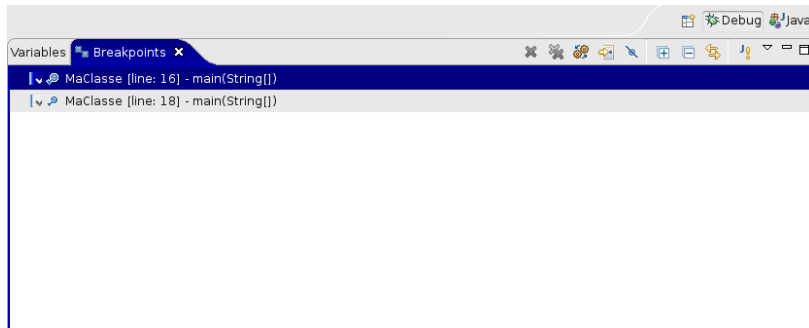
La vue "Variables"

- Permet de visualiser le contenu des variables :



La vue "Points d'arrêts"

- Elle permet de visualiser la liste des points d'arrêt.
- Il est possible de spécifier le nombre d'occurrences avant que l'exécution ne s'arrête sur le point d'arrêt.



La vue "Expressions"

- Permet d'inspecter la valeur d'une expression.
- Il faut ajouter les expressions à inspecter.

