Eclipse 平台入门

英文原文

使用 Eclipse 插件来编辑、编译和调试应用程序

级别：入门

David Gallardo (david@gallardo.org)
软件顾问
2004 年 01 月

本文为您提供关于 Eclipse 平台的概述，包括其起源和体系结构。本文首先简要讨论 Eclipse 的开放源代码性质及其对多种编程语言的支持，然后通过一个简单的程序例子展示 Java 开发环境。本文还将考查以插件扩展形式可用的一些软件开发工具，并展示一个用于 UML 建模的插件扩展。

内容：

相关内容：

将基于 Swing 的开发工具插入 Eclipse 中
国际化 Eclipse 插件
Use Eclipse to build a user interface for XM
developerWorks Toolbox subscription

在 Linux 专区还有：

教程
工具与产品
代码与组件
文章

Eclipse 是什么？

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。

虽然大多数用户很乐于将 Eclipse 当作 Java IDE 来使用，但 Eclipse 的目标不仅限于此。Eclipse 还包括插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许他们构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件，对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。

这种平等和一致性并不仅限于 Java 开发工具。尽管 Eclipse 是使用 Java 语言开发的，但它的用途并不限于 Java 语言；例如，支持诸如 C/C++、COBOL 和 Eiffel 等编程语言的插件已经可用，或预计会推出。Eclipse 框架还可用来作为与软件开发无关的其他应用程序类型的基础，比如内容管理系统。

基于 Eclipse 的应用程序的突出例子是 IBM 的 WebSphere Studio Workbench，它构成了 IBM Java 开发工具系列的基础。例如，WebSphere Studio Application Developer 添加了对 JSP、servlet、EJB、XML、Web 服务和数据库访问的支持。

Eclipse 是开放源代码的软件

开放源代码软件是这样一种软件，它们在发布时附带了旨在确保将某些权利授予用户的许可证。当然，最明显的权利就是源代码必须可用，以便用户能自由地修改和再分发该软件。这种用户权利的保护是通过一种称为 *copyleft* 的策略来完成的：软件许可证主张版权保护，除非明确授予用户这样的权利，否则用户不得分发该软件。copyleft 还要求同一许可证涵盖任何被再分发的软件。这实际上倒置了版权的目的——使用版权来授予用户权利，而不是为软件的开发者保留版权——copyleft 经常被描述为“保留所有版权”。

曾经四处蔓延的对开放源代码软件的许多恐惧、担忧和疑虑，都与某些 copyleft 许可证的所谓“病毒”性质有关——如果使用开放源代码软件作为您开发的程序的一部分，您将失去自己的知识产权，因为该许可证将“传染”您开发的专有部分。换句话说，该许可证可能要求与开放源代码软件一起打包的所有软件，都必须在相同的许可证之下发布。虽然这对最著名的 copyleft 许可证（即 GNU 通用公共许可证，例如 Linux 就是在该许可证之下发布的）来说可能是事实，当时还有其他许可证在商业化和社区考虑之间提供了较好的平衡。

开放源代码计划（Open Software Initiative）是一家非营利机构，它明确定义了开放源代码的含义及满足其标准的认证许可证。Eclipse 是在 OSI 认可的通用公共许可证（CPL）1.0 版之下被授予许可证的，CPL“旨在促进程序的商业化使用……”（欲获得指向通用公共许可证 1.0 版完整文本的链接，请参阅本文稍后的 参考资料）。

为 Eclipse 创建插件或将 Eclipse 用作软件开发应用程序基础的开发人员，需要发布他们在 CPL 下使用或修改的任何 Eclipse 代码，但是他们可以自由决定自己添加的代码的许可证授予方式。与出自 Eclipse 的软件一起打包的专有代码不需要作为开放源代码来授予许可证，该源代码也不需要提供给用户。

尽管大多数开发人员不会使用 Eclipse 来开发插件，或创建基于 Eclipse 的新产品，但是 Eclipse 的开放源代码性质所意味的，并不只是它使得 Eclipse 免费可用（尽管便于商业化的许可证意味着插件可能要花钱）。开放源代码鼓励创新，并激励开发人员（甚至是商业开发人员）为公共开放源代码库贡献代码。对此存在许多原因，不过最本质的原因或许是为这个项目作贡献的开发人员越多，这个项目就会变得对每个人都越宝贵。随着这个项目变得更加有用，更多的开发人员将会使用它，并围绕它形成一个社区，就像那些围绕 Apache 和 Linux 形成的社区一样。

Eclipse 是什么机构？

Eclipse.org 协会管理和指导 Eclipse 正在进行中的开发。在据说 IBM 花了 4000 万美元开发 Eclipse，并把它作为一个开放源代码项目发布之后，Eclipse.org 协会吸收了许多软件工具提供商，包括 Borland、Merant、Rational、RedHat、SuSE、TogetherSoft 和 QNX。从那以后还有其他公司相继加入，包括 Hewlett Packard、Fujitsu、Sybase。这些公司分别向理事会派了一名代表，这个理事会负责确定 Eclipse 项目的方向和范围。

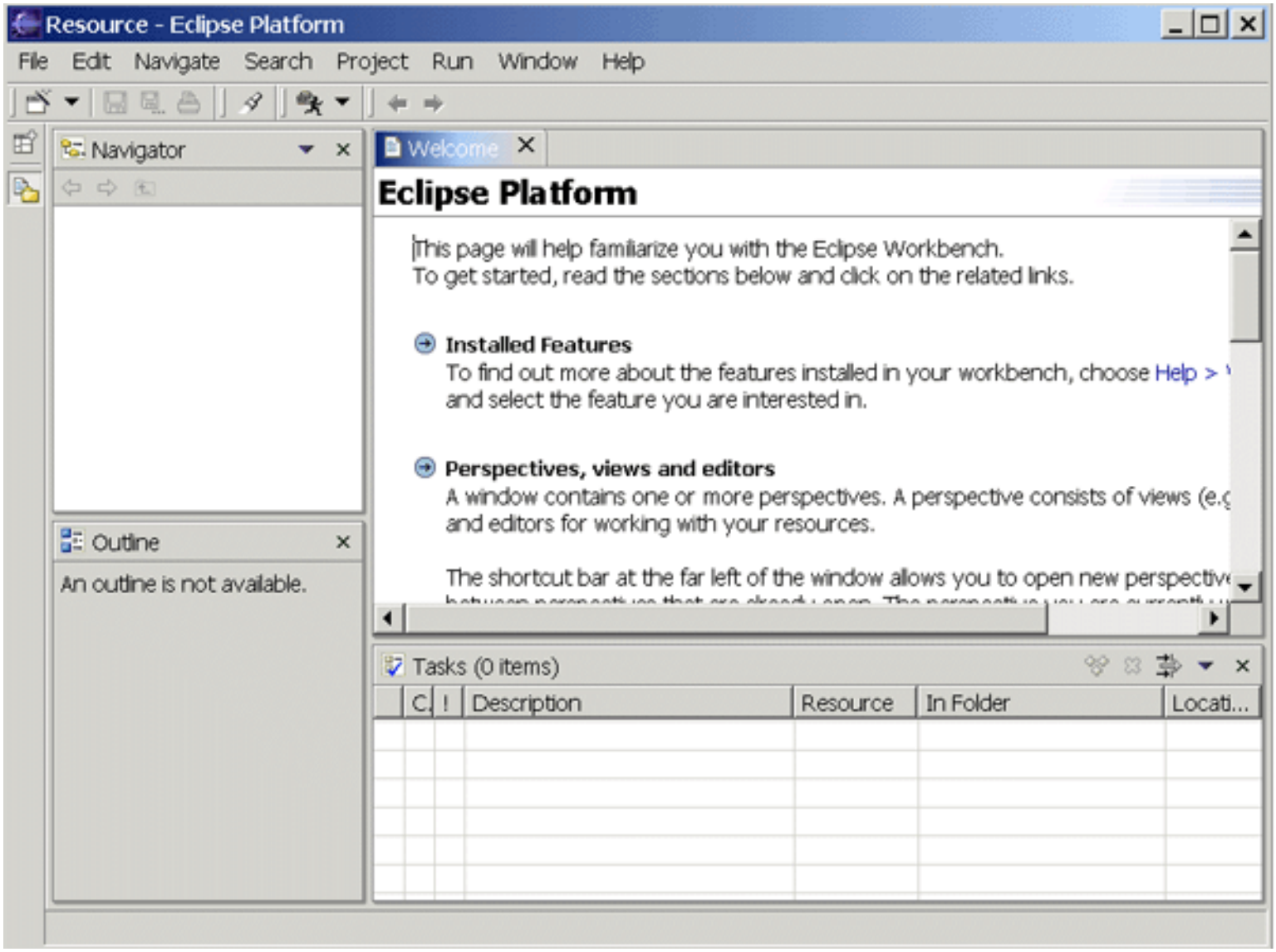
在最高层，项目管理委员会（Project Management Committee，PMC）管理着 Eclipse 项目。这个项目被划分为多个子项目，每个子项目都有一名负责人。大型子项目又被划分为组，每个组也有一名负责人。目前，这其中的大多数管理角色都由最初开发 Eclipse 的 IBM 子公司 Object Technology International (OTI)的人担任，但是作为一个开放源代码的项目，它欢迎任何人的参与。任何特定部门的职责是通过该部门对项目的贡献来争取的。

现在我们已经考察了 Eclipse 背后的一些理论、历史和管理，下面让我们考察该产品本身。

Eclipse 工作台

在第一次打开 Eclipse 时，首先看到的是下面的欢迎屏幕：

图 **1. Eclipse** 工作台



Eclipse 工作台由几个称为 视图（*view*） 的窗格组成，比如左上角的 Navigator 视图。窗格的集合称为 透视图（*perspective*）。默认的透视图是 Resource 透视图，它是一个基本的通用视图集，用于管理项目以及查看和编辑项目中的文件。

Navigator 视图 允许您创建、选择和删除项目。Navigator 右侧的窗格是 编辑器区域。取决于 Navigator 中选定的文档类型，一个适当的编辑器窗口将在这里打开。如果 Eclipse 没有注册用于某特定文档类型（例如，Windows 系统上的 .doc 文件）的适当编辑器，Eclipse 将设法使用外部编辑器来打开该文档。

Navigator 下面的 **Outline** 视图 在编辑器中显示文档的大纲；这个大纲的准确性取决于编辑器和文档的类型；对于 Java 源文件，该大纲将显示所有已声明的类、属性和方法。

Tasks 视图 收集关于您正在操作的项目的信息；这可以是 Eclipse 生成的信息，比如编译错误，也可以是您手动添加的任务。

该工作台的大多数其他特性，比如菜单和工具栏，都应该和其他那些熟悉的应用程序类似。一个便利的特性就是不同透视图的快捷方式工具栏，它显示在屏幕的左端；这些特性随上下文和历史的不同而有显著差别。Eclipse 还附带了一个健壮的帮助系统，其中包括 Eclipse 工作台以及所包括的插件（比如 Java 开发工具）的用户指南。至少浏览一遍这个帮助系统是值得的，这样可以看到有哪些可用的选项，同时也可更好地理解 Eclipse 的工作流程。

为继续这个短暂的 Eclipse 之旅，我们将在 Navigator 中创建一个项目。右键单击 Navigator 视图，然后选择 **New=>Project**。当 New Project 对话框出现时，选择左面的 Java。标准 Eclipse 只有一种 Java 项目类型，名为“Java Project”。如果安装了插件来提供 JSP 和 servlet 支持，我们会从这里看到一个用于 Web 应用程序的附加选项。眼下，请选择 Java Project，在提示项目名称时输入“Hello”，然后按 Finish。

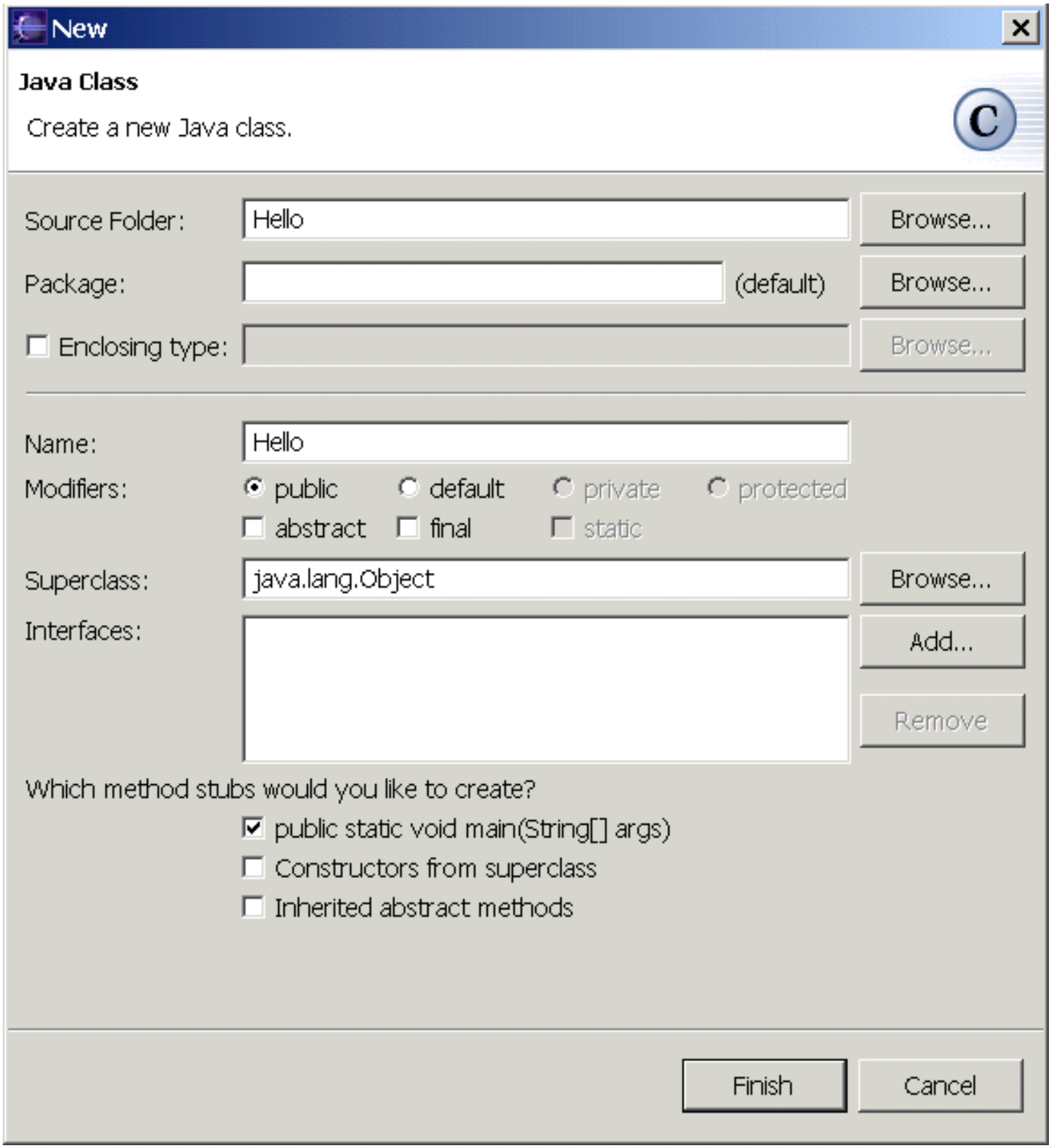
接下来，我们将检查一下 Java 透视图。取决于您喜欢的屏幕管理方式，您可以通过选择 **Window=>Open Perspective=>Java** 来改变当前窗口中的透视图，也可以通过选择 **Window=>New Window**，然后再选择这个新的透视图，从而打开一个新的窗口。

正如您可能预期的那样，Java 透视图包含一组更适合于 Java 开发的视图。其中之一就是左上角的视图，它是一个包含各种 Java 包、类、jar 和其他文件的层次结构。这个视图称为 **Package Explorer**。还要注意主菜单已经展开了——并且出现了两个新的菜单项：Source 和 Refactor。

Java 开发环境（JDE）

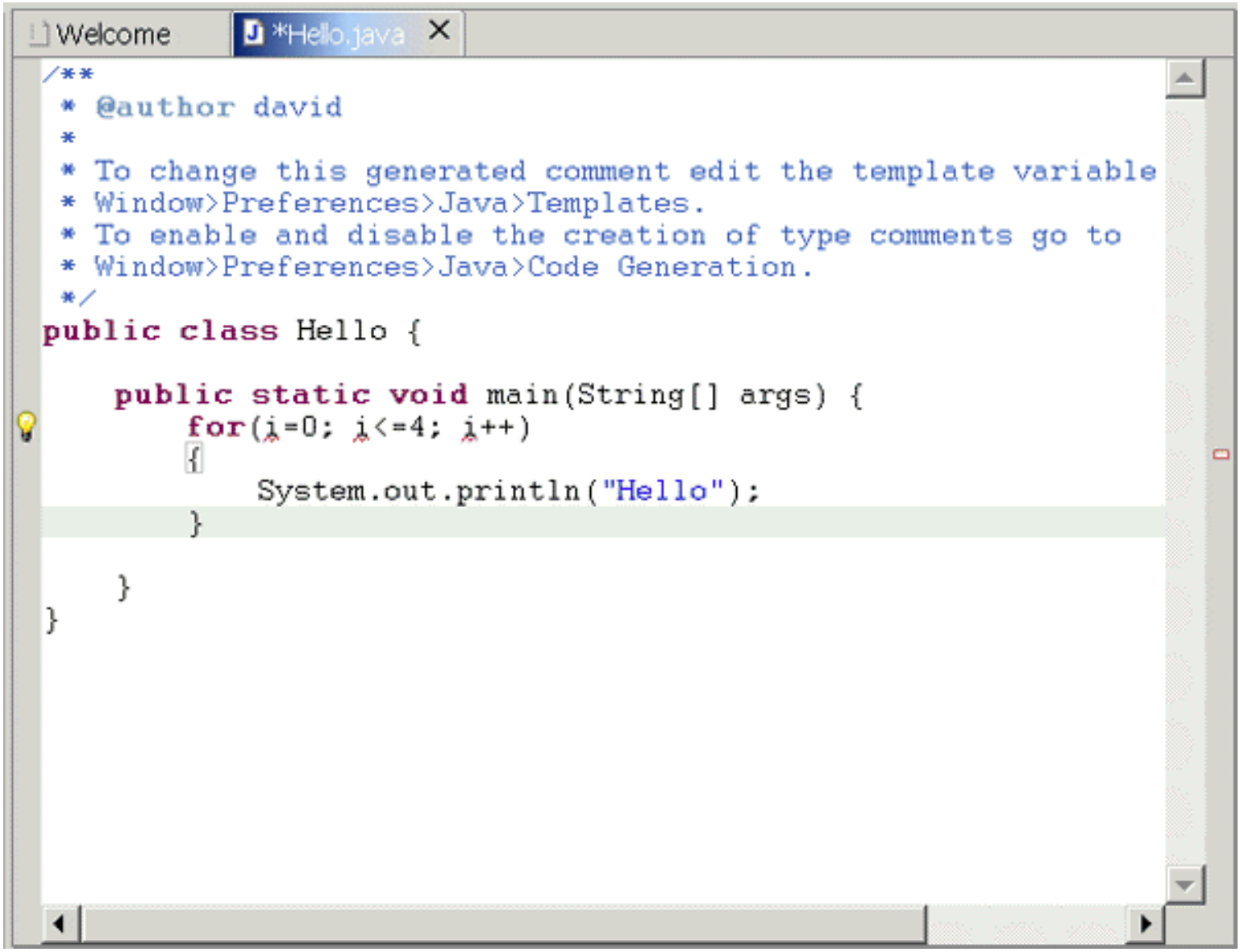
为试验一下 Java 开发环境，我们将创建并运行一个“Hello, world”应用程序。使用 Java 透视图，右键单击“Hello”项目，选择 **New=>Class**，如图 2 所示。在随后出现的对话框中，键入“Hello”作为类名称。在“Which method stubs would you like to create?”下面，选中“public static void main(String[] args)”复选框，然后按 Finish。

图 **2.** 在 **Java** 透视图中创建新类



这样将在编辑器区域创建一个包含 Hello 类和空的 main() 方法的 .java 文件，如图 3 所示。然后向该方法添加如下代码（注意其中 i 的声明是有意省略了的）：

图 **3. Java** 编辑器中的 **Hello** 类



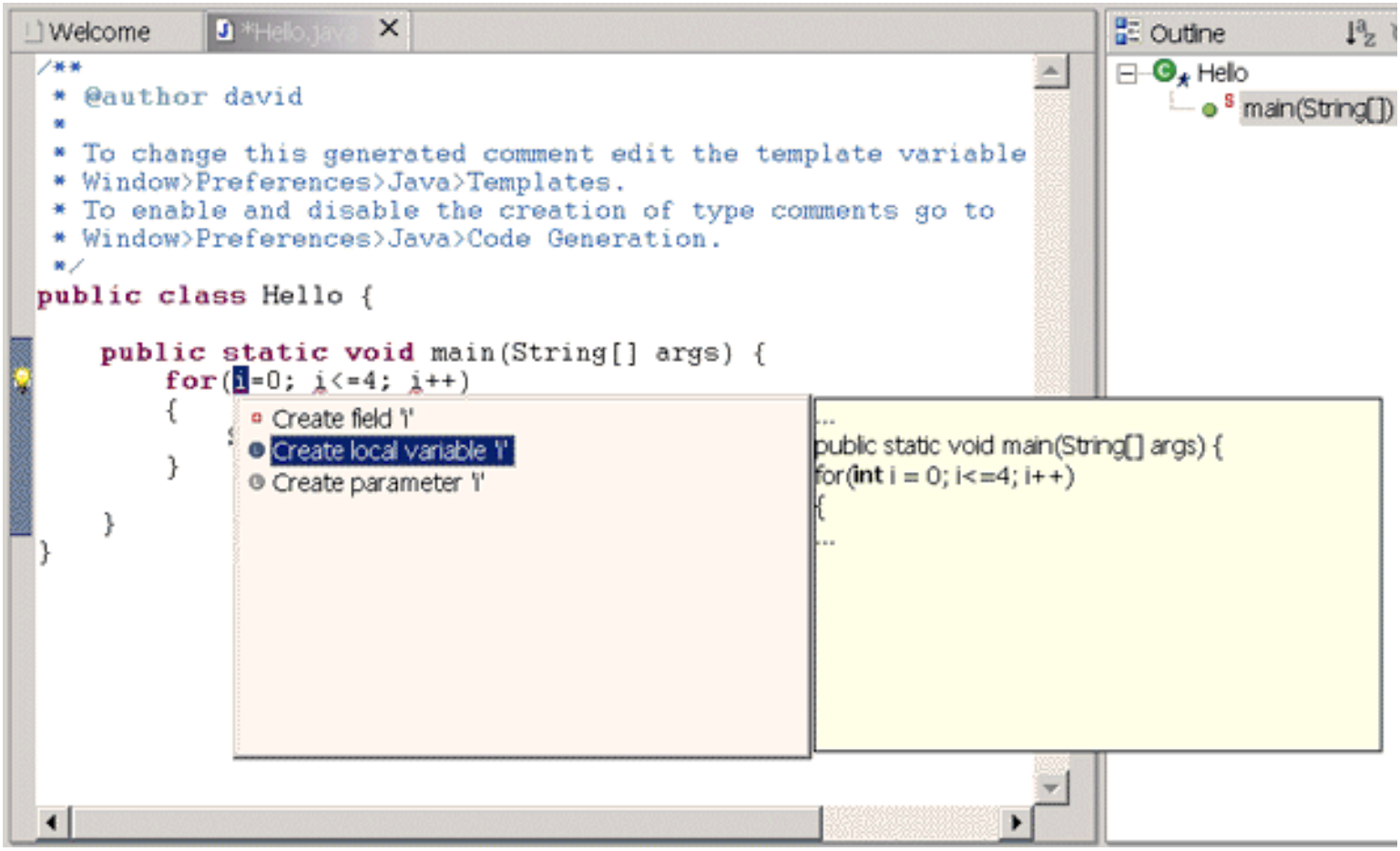
您会在键入时注意到 Eclipse 编辑器的一些特性，包括语法检查和代码自动完成。在 2.1 版（我曾下载 M2 版来试用过）中，当您键入开括号或双引号时，Eclipse 会自动提供配对的符号，并将光标置于符号对之内。

在其他情况下，您可以通过按 Ctrl-Space 来调用代码自动完成功能。代码自动完成提供了上下文敏感的建议列表，您可通过键盘或鼠标来从列表中选择。这些建议可以是针对某个特定对象的方法列表，也可以是基于不同的关键字（比如 for 或 while）来展开的代码片断。

语法检查依赖增量编译。每当您保存代码，它就在后台接受编译和语法检查。默认情况下，语法错误将以红色下划线显示，一个带白 “X” 的红点将出现在左边沿。其他错误在编辑器的左边沿通过灯泡状的图标来指示；这些就是编辑器或许能为您修复的问题——即所谓的Quick Fix（快速修复）特性。

上面的代码例子在 for 语句后面有一个灯泡状图标，因为 i 的声明被省略了。双击该图标将调出建议的修复列表。在此例中，它将提供创建一个类字段 i、一个局部变量 i 或一个方法参数 i 的建议；单击其中的每一个建议都会显示将要生成的代码。图 4 显示了该建议列表和建议创建一个局部变量之后生成的代码。

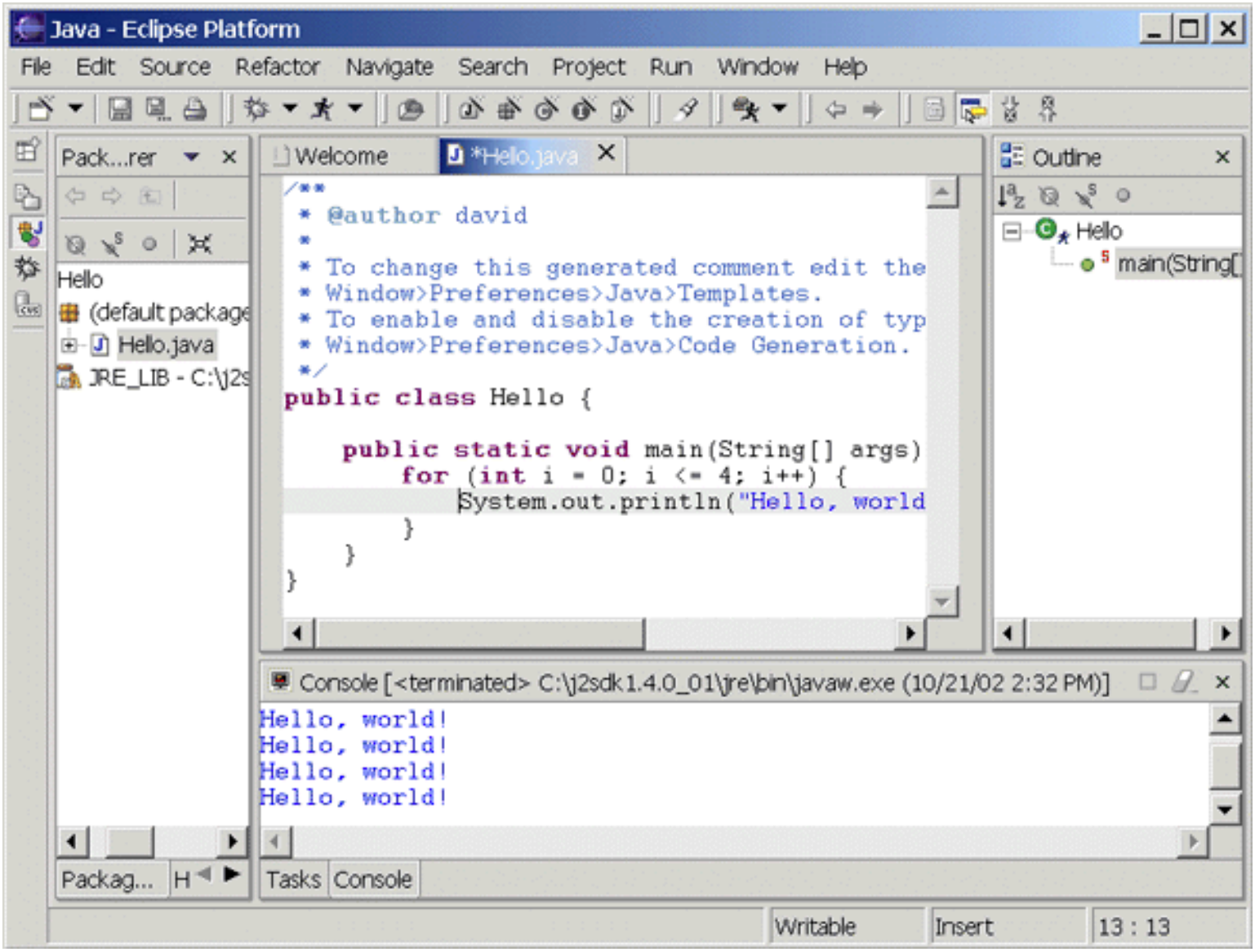
图 **4. Quick Fix** 建议



双击该建议就会把建议代码插入到代码中的恰当位置。

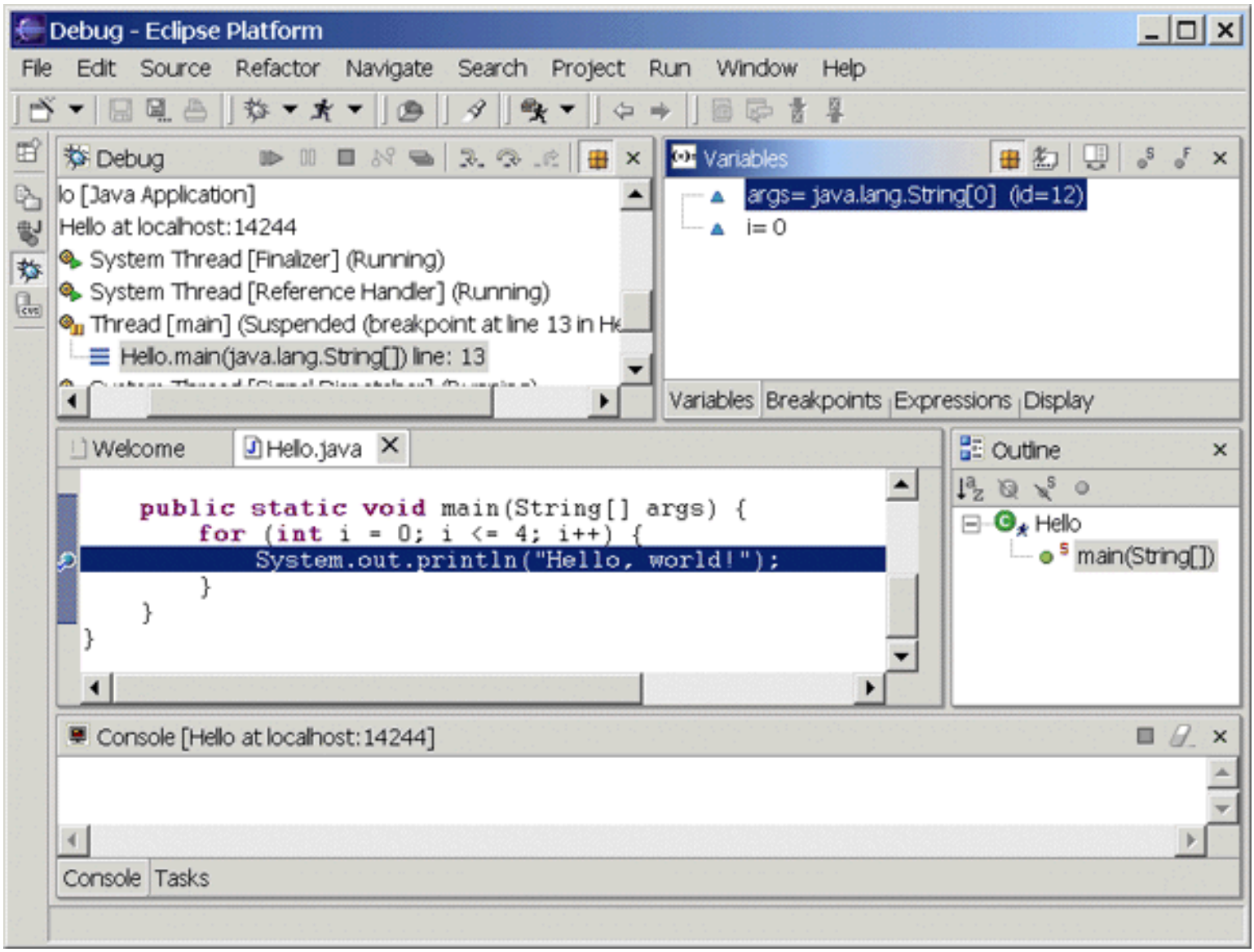
一旦代码无错误地编译完成，您就能够从 Eclipse 菜单上选择 Run 来执行该程序（注意这里不存在单独的编译步骤，因为编译是在您保存代码时进行的。如果代码没有语法错误，它就可以运行了）。这时会出现一个具有适当默认设置的 Launch Configurations 对话框；请按右上角的 Run 按钮。一个新的选项卡式窗格将出现在下面的窗格（控制台）中，其中显示了程序的输出，如图 5 所示。

图 **5.** 程序的输出



也可以在 Java 调试器中运行程序。首先双击编辑器视图左端的灰色边沿，从而在调用 System.out.println() 之后的 main() System.out.println() 中设置一个断点。一个蓝色的点将会出现在那里。然后从 Run 菜单上选择 Debug。正如上面描述的，这时会出现一个 Launch Configurations 对话框。请选择 Run。透视图将自动切换到 Debug 透视图，其中具有许多有趣的新视图，如图 6 所示：

图 **6. Debug** 透视图



首先，请注意该透视图左上角的 Debug 视图。这个视图显示调用堆栈，并且标题栏中有一个工具栏，它允许您控制程序的执行，包括继续、挂起或终止程序、跟踪下一个语句、单步执行下一个语句，或者从方法返回。

右上角的窗格包含许多选项卡式的视图，包括 Variables、Breakpoints、Expressions 和 Display。这里我单击了 Variables 视图，以便我们能够看到 i 的当前值。

可以通过上下文敏感的帮助，获得关于这些视图的更多信息：单击视图的标题，然后按 F1。

附加插件

除了像 JDT 这样用于编辑、编译和调试应用程序的插件外，还有些可用的插件支持从建模、生成自动化、单元测试、性能测试、版本控制到配置管理的完整开发过程。

Eclipse 标准地附带了配合 CVS 使用的插件，CVS 是用于源代码控制的开放源代码并发版本系统（Concurrent Versions System）。Team 插件连接到 CVS 服务器，允许开发团队的成员操作一组源代码文件，却不会相互覆盖其他人的更改。这里不打算进一步探讨如何从 Eclipse 内部进行源代码控制，因为这需要安装 CVS 服务器，不过支持开发团队而不只是独立的开发，这是 Eclipse 的一个重要的必备特性。

已经可用或已宣布要推出的一些第三方插件包括：

版本控制和配置管理

- CVS
- Merant PVCS
- Rational ClearCase

UML 建模

- OMONDO EclipseUML
- Rational XDE (代替 Rose)
- Together WebSphere Studio Edition

图形

- Batik SVG
- Macromedia Flash

Web 开发、**HTML**、**XML**

- Macromedia Dreamweaver
- XMLBuddy

应用服务器集成

- Sysdeo Tomcat launcher

欲了解可用插件的更完整列表，请参阅 参考资料 中的链接。

例子：一个用于 **UML** 建模的插件

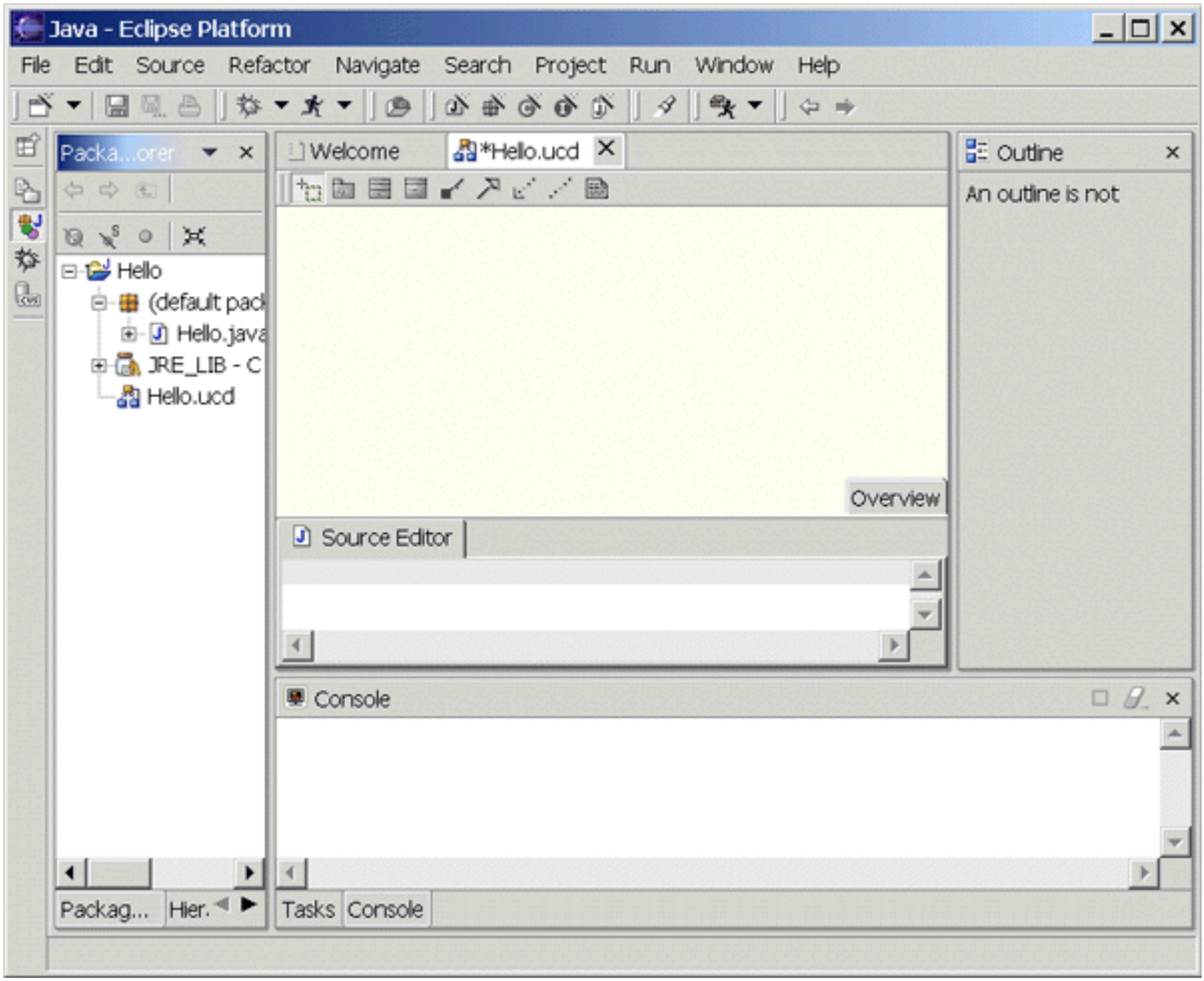
要查看插件的例子，以及查看它是如何与 Eclipse 集成的，请下载流行的 OMONDO EclipseUML（参阅 参考资料 中的链接）；您需要注册，不过该插件是免费的。这个插件依赖 GEF，即 Graphical Editor Framework，这是另一个 Eclipse 插件。GEF 是 Tools 子项目的一部分。要下载 GEF，请转到 Eclipse Web 站点（参阅 参考资料），选择“downloads”，然后单击“Tools PMC downloads page”链接。注意您需要下载 OMONDO 推荐的 GEF 版本（针对 OMONDO 1.0.2 的是 GEF 2.0 版）。

下载之后，插件的安装通常是通过解压缩下载文件，并将其内容复制到 Eclipse 插件目录来完成的。在此例中，GEF 需要解压缩到 Eclipse目录（它将自动从该目录进入插件目录）。为安全起见，您可能想将它解压缩到某个临时目录，再相应地从那里复制相关目录。如果 Eclipse 正在运行，您需要停止它然后再重新启动它，这样它才能识别新安装的插件。

一旦 EclipseUML（以及 GEF）安装完成，您就能够像创建一个 Java 类文件一样创建一个类图。在 Java 透视图中，右键单击 Package Explorer 中的“Hello”项目，然后从弹出菜单上选择 **New=>Other**。New 对话框的左边窗格中将会有一个用于 UML 的新选项。EclipseUML 的免费版本仅支持类图，因此右侧的惟一选项是 UML Class Diagram。请选择 UML Class

Diagram，然后为该类图键入一个名称，比如“Hello”：

图 **7. Class Diagram** 编辑器



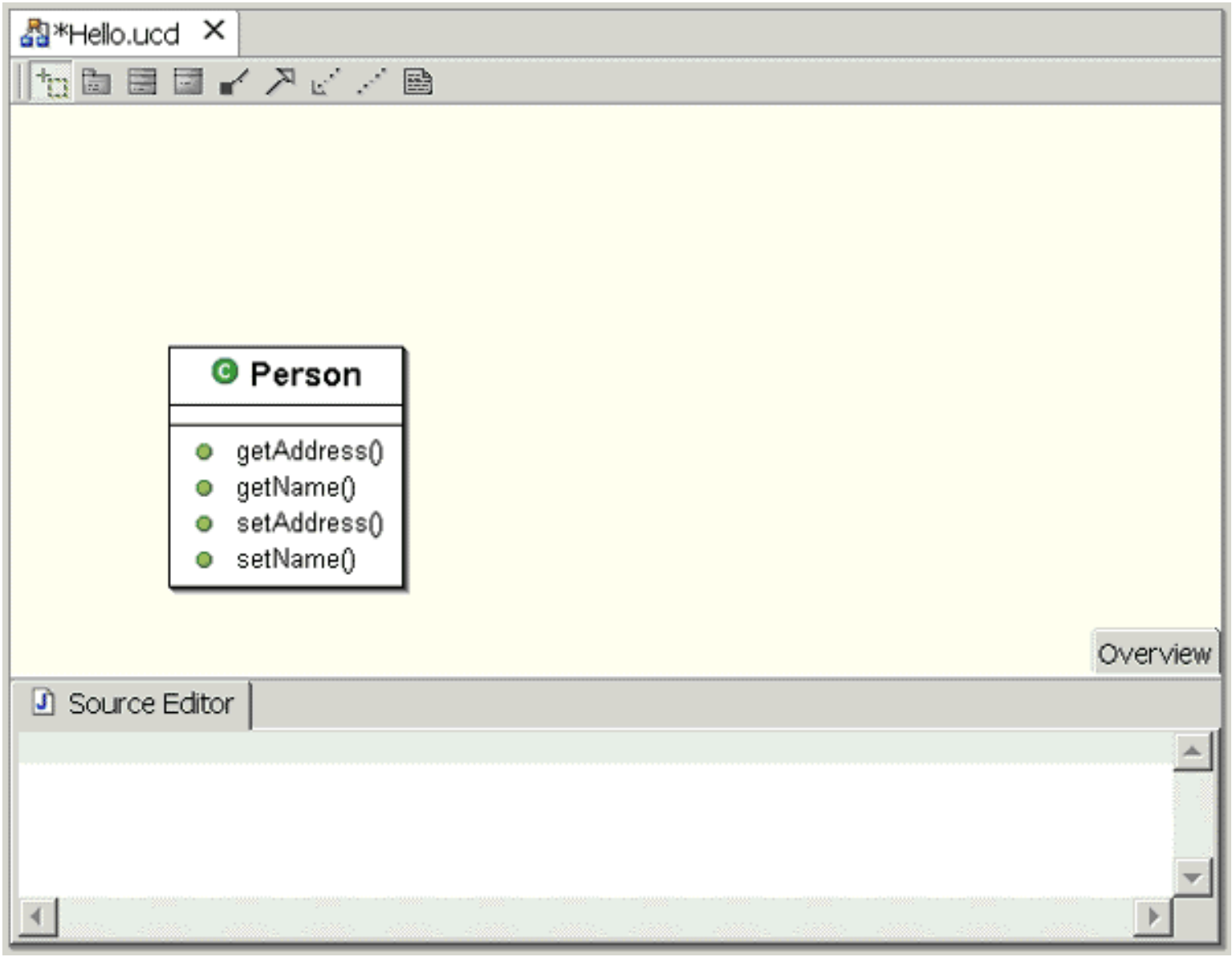
编辑器区域中将会出现一个图形编辑器，它带有用于绘制类图的画布。您可以通过两种方式创建类图：通过将 Java 文件从 Package Explorer 拖放到类图上，从而对现有代码进行逆向工程；或者使用空白类图上面工具栏中可用的绘制工具。要试验第一种方法，请创建一个名为 Person 的新类（使用 **File=>New=>Class**），然后赋予它下面列出的两个私有属性：

```
/** Person.java
 * @author david
 */
public class Person {
private String name;
private Address address;

/**
 * Returns the address.
 * @return Address
 */
public Address getAddress() {
       return address;
}

/**
 * Returns the name.
 * @return String
 */
public String getName() {
      return name;
}

/**
 * Sets the address.
 * @param address The address to set
 */
public void setAddress(Address address) {
      this.address = address;
}

/**
 * Sets the name.
 * @param name The name to set
 */
public void setName(String name) {
     this.name = name;
}

}
```

（应该承认，我仅键入了针对 name 和 address 的行。getter 和 setter 方法是通过 Eclipse 自动生成的，即右键单击源代码，然后从弹出菜单上选择 **Source=>Generate Getter and Setter** 。）

请保存并关闭 Person.java Hello.ucd。

图 **8. Person** 类图

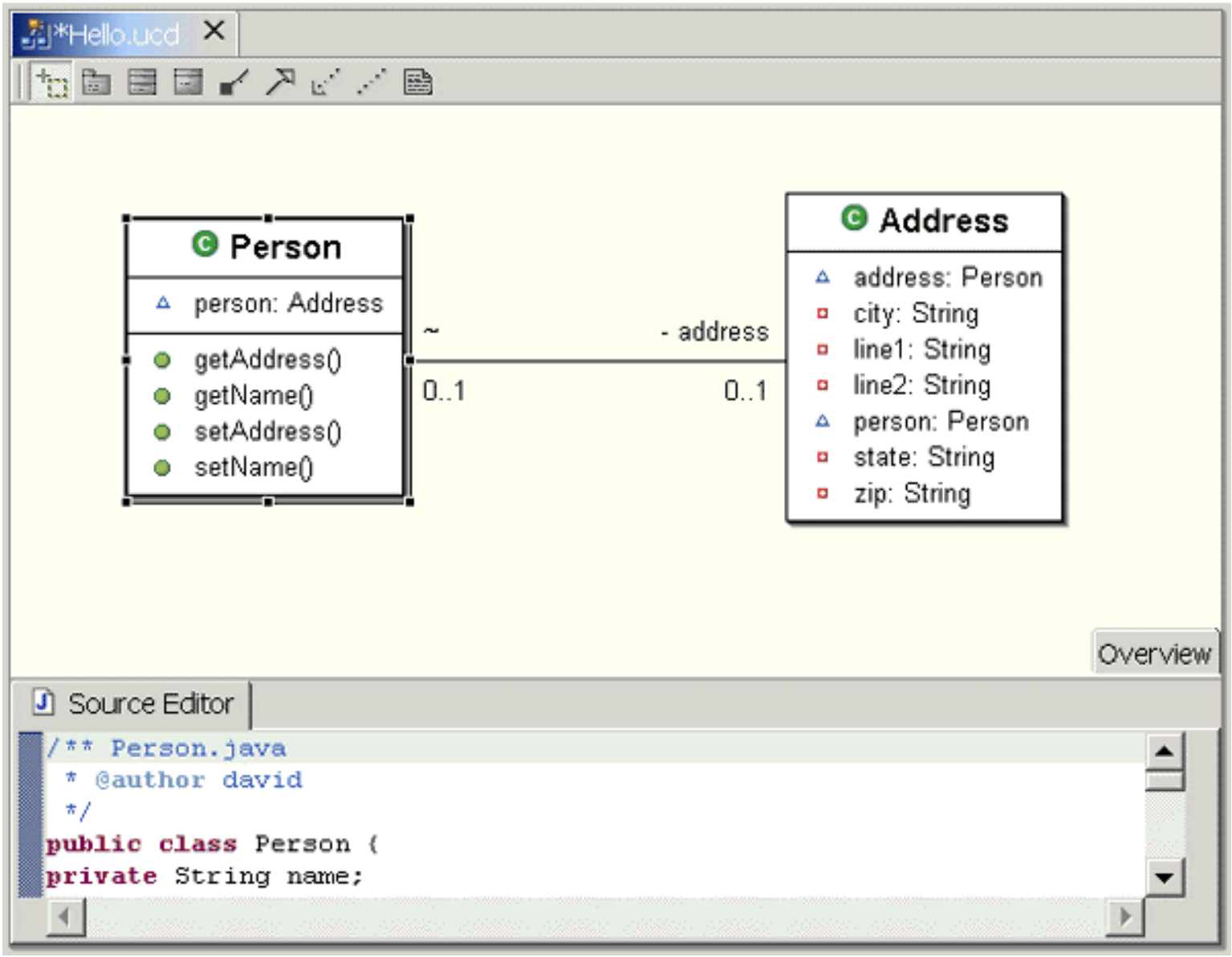


要从 UML 创建 Java 类，请单击类图窗口顶部工具栏上的“New class”按钮，即左起第三个按钮，然后单击类图。当 New 类向导打开时，请键入 Adress 作为类名称，然后按 Finish。

您可以右键单击类名称并选择 **New=>Attribute**，从而给类添加属性。在 New 属性对话框中，请输入属性名称、类型和可见性。然后右键单击类名称并选择 **New=>Method** 来添加方法。

当您更改类图时，图下面的 Source Editor 窗口将反映所做的更改。最后，您可以单击 Association 按钮（左起第五个），绘制一条从 Person 类指向 Address 类的线段，从而绘制这两个类之间的关系图。这样会调出另外一个对话框，您可以在其中输入关联属性（请参考 EclipseUML 帮助，以了解关于必需信息的更多内容）。完成后的图应该类似如下：

图 **9.** 关联



这个 UML 插件展示了 Eclipse 插件的几个典型特点。首先，它展示了工具之间的紧密集成。表面上绝对无法看出有多个组件在工作；与 Eclipse 平台和 JDT 的集成是无缝的。例如，当 Person 类被创建时，它显示语法错误是因为它的一个属性 Address 没有定义。一旦 Address 类在 UML 图中创建完成，这些组件就会分开显示出来。

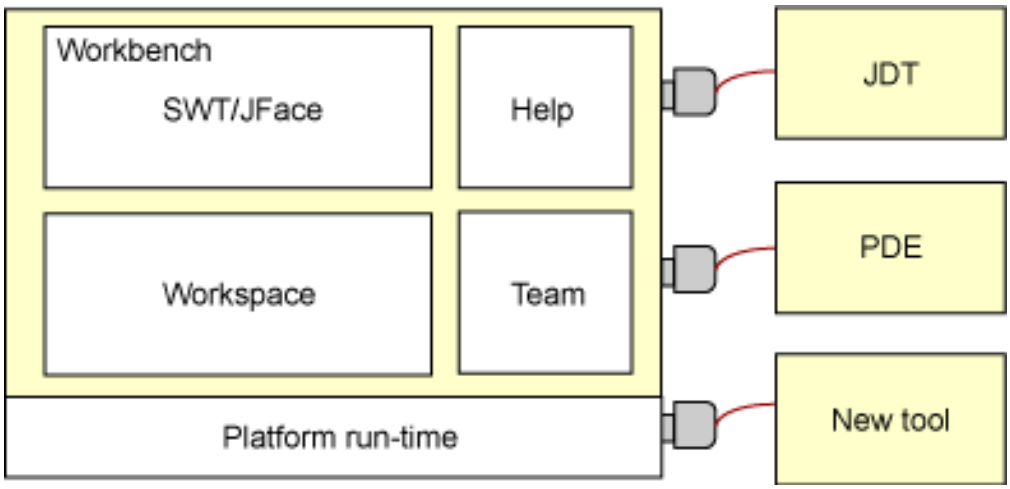
另一个特点是 EclipseUML 利用其他插件提供的功能的能力——在此例中是 GEF 插件，它提供用于开发可视化编辑器的工具。

还有另一个特点涉及 EclipseUML 插件使用多层次功能来分发的方式。支持类图的基本插件是免费的，但是更成熟的版本要付费才能使用。

Eclipse 平台体系结构

Eclipse 平台是一个具有一组强大服务的框架，这些服务支持插件，比如 JDT 和插件开发环境（PDE）。它由几个主要的部分构成：平台运行库、工作区、工作台、团队支持和帮助。

图 **10. Eclipse** 平台体系结构



平台
平台运行库是内核，它在启动时检查已安装了哪些插件，并创建关于它们的注册表信息。为降低启动时间和资源使用，它在实际需要任何插件时才加载该插件。除了内核外，其他每样东西都是作为插件来实现的。

工作区
工作区是负责管理用户资源的插件。这包括用户创建的项目、那些项目中的文件，以及文件变更和其他资源。工作区还负责通知其他插件关于资源变更的信息，比如文件创建、删除或更改。

工作台
工作台为 Eclipse 提供用户界面。它是使用标准窗口工具包（SWT）和一个更高级的 API（JFace）来构建的；SWT 是 Java 的 Swing/AWT GUI API 的非标准替代者，JFace 则建立在 SWT 基础上，提供用户界面组件。

SWT 已被证明是 Eclipse 最具争议的部分。SWT 比 Swing 或 SWT 更紧密地映射到底层操作系统的本机图形功能，这不仅使得 SWT 更快速，而且使得 Java 程序具有更像本机应用程序的外观和感觉。使用这个新的 GUI API可能会限制 Eclipse 工作台的可移植性，不过针对大多数流行操作系统的 SWT 移植版本已经可用。

Eclipse 对 SWT 的使用只会影响 Eclipse 自身的可移植性——使用 Eclipse 构建的任何 Java 应用程序都不会受到影响，除非它们使用 SWT 而不是使用 Swing/AWT。

团队支持
团队支持组件负责提供版本控制和配置管理支持。它根据需要添加视图，以允许用户与所使用的任何版本控制系统（如果有的话）交互。大多数插件都不需要与团队支持组件交互，除非它们提供版本控制服务。

帮助
帮助组件具有与 Eclipse 平台本身相当的可扩展能力。与插件向 Eclipse 添加功能相同，帮助提供一个附加的导航结构，允许工具以 HTML 文件的形式添加文档。

Eclipse 的前景

围绕 Eclipse 的开发正处于关键阶段。主要软件工具提供商都参与进来了，并且开放源代码 Eclipse 插件项目的数量正在与日俱增。

可移植、可扩展、开放源代码的框架并不是个新思想（您会想起 Emacs），但是由于它成熟、健壮和优雅的设计，Eclipse 带

来了全新的动力。IBM 价值 4000 万美元的世界级软件在开放源代码领域的发布，给业界带来了久违的震撼。

参考资料

- 可以从 Eclipse 项目网站 获得 Eclipse 的文档、文章以及下载 Eclipse。
- 查看 通用公共许可证 v1.0 的内容。
- 浏览完整的 Eclipse 插件列表。
- 下载流行的 OMONDO EclipseUML；您需要注册，不过该插件是免费的。
- 关于开放源代码软件的信息，包括诸如公共通用许可证这样的认证开放源代码许可证，可在 开放源代码计划 Web 站点 上找到。
- 关于 copyleft 的权威解释可在 免费软件基金会的Web 站点 上找到。
- 阅读 IBM 宣布向开放源代码社区捐赠 Eclipse 的新闻稿。
- 在以下这些 *developerWorks* 文章中了解关于 Eclipse 的更多信息:
  - “Interview with Marc Erikson about the Eclipse code donation”（*developerWorks*，2001 年 11 月）。
  - “Working the Eclipse Platform”（*developerWorks*，2001 年 11 月）。
  - “Getting to know WebSphere Studio Application Developer”（*developerWorks*，2001 年 11 月）。
  - “Help for reusing your assets”（*developerWorks*，2001 年 11 月)。
  - “Create native, cross-platform GUI applications”（*developerWorks*，2002 年 4 月）。
  - “国际化 Eclipse 插件”（*developerWorks*，2002 年 6 月）。
  - “测试您国际化的 Eclipse 插件”（*developerWorks*，2002 年 7 月）。
  - “将基于 Swing 的开发工具插入 Eclipse 中”（*developerWorks*，2003 年 1 月）。
  - “Working XML: Use Eclipse to build a user interface for XM”（*developerWorks*，2002 年 10 月）。
- 最近计算机新闻界还出现了许多关于 Eclipse 的文章。在“Promoting shared software through Eclipse”（*DTmag.com*，2002 年 11 月）中了解为什么 Carleton 大学的一位教授会选择基于 Eclipse 建立两个野心勃勃的项目。

关于作者

David Gallardo 是一名独立软件顾问和作家，他的专长是软件国际化、Java Web 应用程序和数据库开发。他成为专业软件工程师已经有 15 年了，他拥有许多操作系统、编程语言和网络协议的经验。他最近在一家 B2B 电子商务公司 TradeAccess, Inc.从事先进的数据库和国际化开发。在这之前，他是 Lotus Development Corporation 国际产品开发部的高级工程师，负责开发为 Lotus 产品（包括 Domino）提供 Unicode 和国际化语言支持的跨平台库。可以通过 david@gallardo.org 与 David 联系。

到页首

您对这篇文章的看法如何？

○ 真棒！(5) ○ 好材料 (4) ○ 一般；尚可 (3) ○ 需提高 (2) ○ 太差！ (1)

建议？

提交反馈意见