自己动手写开发工具

——基于Eclipse的工具开发

杨中科◎编著

- 剖析现代开发工具的设计理念
- 讲解Eclipse插件开发技术
- 来自开发一线作者的经验结晶

清华大学出版社

自己动手写开发工具
——基于 Eclipse 的工具开发

杨中科　编著

清华大学出版社
北　京

内 容 简 介

本书系统地介绍了 SWT、Draw2D、GEF、JET 等与 Eclipse 插件开发相关的基础知识，并且以实际的开发案例来演示这些知识的实战性应用，通过对这些实际开发案例的学习，读者可以非常轻松地掌握 Eclipse 插件开发的技能，从而开发出满足个性化需求的插件。

本书以一个简单而实用的枚举生成器作为入门案例，通过该案例读者能学习到扩展点、SWT、JET 等 Eclipse 插件开发的基本技能；接着对 Eclipse 插件开发中的基础知识进行了介绍，并且对属性视图的使用做了重点介绍；最后以两个具有一定复杂程度的插件(Hibernate 建模工具和界面设计器)为案例介绍了 SWT、Draw2D、GEF、JET 等技术的综合运用。

本书不仅适合于 Eclipse 插件开发初学者学习，对于有一定相关开发经验的开发人员也具有很高的参考价值。

图书在版编目(CIP)数据

自己动手写开发工具——基于 Eclipse 的工具开发/杨中科编著. —北京：清华大学出版社，2007.10
ISBN 978-7-302-16217-9

Ⅰ. 自…　Ⅱ. 杨…　Ⅲ. 软件工具—程序设计　Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字(2007)第 152381 号

责任编辑：彭　欣　宋延清
封面设计：阳光智慧 + 荀博
版式设计：杨玉兰
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×260　**印　　张**：22.75　**字　　数**：516 千字
附光盘 1 张
版　　次：2007 年 10 月第 1 版　　**印　　次**：2007 年 10 月第1 次印刷
印　　数：1～4000
定　　价：45.00 元

序　言

自己动手写开发工具是很多开发人员的梦想，虽然市场上已经有了各种开发工具，但是在一些情况下还是有编写自己开发工具的需求的：

- 使用的编程语言没有合适的开发工具。比如在 Eclipse 出现之前，Python、Ruby、JavaScript 等语言都没有很好的全面支持代码编写、调试以及重构的开发工具，使用这些语言进行开发非常麻烦。
- 为自己开发的语言配备开发工具。有时我们会开发一款新的开发语言，为了方便使用，也需要为其提供相应的开发工具。
- 为控件库、框架等提供开发工具。Echo2、Tapestry、Spring 等都是非常优秀的产品，但是通过手工编码的方式使用这些产品仍然是非常麻烦的，如果能配备图形化的开发工具，那么通过简单地用鼠标拖曳就可以快速完成工作。
- 为产品提供二次开发工具。很多公司都有自己的产品，而这些产品一般都提供了二次开发的能力，开发人员只要进行少量的编码或者配置就可以很轻松地实现针对特定客户的个性化功能。由于二次开发人员的技术水平相对较差，如果能提供一个图形化的二次开发工具必将提高二次开发的效率及质量。

针对上面的这几种情况，已经有很多开发人员探索着实现了自己的开发工具，比如 Boa Constructor 就是一款用 Python 语言编写的 Python 开发工具，润乾报表提供了用 Swing 技术实现的报表设计器。这种将所有功能全盘地由自己实现的方式有如下缺点。

第一，必须自己处理所有的细节问题。比如实现一个语言的开发工具就必须自己处理语法高亮、语法分析、代码提示、调试、重构、可视化的界面编辑器以及代码生成等，这些问题的处理对开发人员的要求非常高，而且开发工作量也非常大。

第二，各个工具的差异性非常大，增加了用户的学习成本。

第三，不同的工具之间的集成非常困难。由于不同的工具是由各个厂商独立开发出来的，互相之间的集成非常麻烦，不仅使用的时候需要运行多个工具，而且经常需要在多个实现相似功能的工具之间做出取舍。

Delphi、Visual Studio .NET、JBuilder、NetBeans 等都提供了一定的扩展机制，我们只要按照要求编写插件就能在这些工具中开发扩展功能，但是这些工具提供的扩展功能是非常简单和有限的，我们几乎无法完成编写开发工具这样复杂的功能。

作为 IDE 界的一匹黑马，Eclipse 近些年来异军突起，很多开源项目或者商业化的产品都提供了相应的 Eclipse 插件，比如 Echo2、GWT、Struts 等开源产品以及 IBM Websphere、Crystal、金蝶、普元等商业公司的开发工具都基于 Eclipse 进行开发，甚至 Borland 也将新版本的 JBuilder 移植到 Eclipse 上。Eclipse 能够得到这么多厂商的支持，究其原因有如下几点：免费且开源；开放性；可扩展性强；对开发工具的开发提供了强大的支持；基于 Eclipse

的产品更专业；各种插件可以组合使用。

1. 免费且开源

大多数开发工具都是按用户数收费的，对于开发人员比较多的公司来说开发工具的支出是一笔不小的费用，而且基于这些开发工具开发出来的扩展插件在发布的时候也会涉及到授权的问题。Eclipse 是免费使用的，这样就为公司节省了不小的一笔开支，而且只要遵守 EPL 协议，那么基于 Eclipse 开发的扩展插件可以任意发布。Eclipse 是开源的，通过研读 Eclipse 的代码，我们能更快地开发出高质量的插件。

2. 开放性

Eclipse 并没有局限于 Java 语言，我们可以开发非 Java 语言的开发插件，比如 Ruby、Python、C/C++、C#以及 PHP 等语言都有了 Eclipse 上的开发插件。而且 Eclipse 也没有限定插件的应用领域，所以 Eclipse 成为了很多领域开发工具的基础，不仅 IBM、金蝶、普元等企业级系统开发商选择 Eclipse 作为其开发工具的基础，而且像风河系统公司、Accelerated 科技、Altera、TI 和 Xilinx 等嵌入式系统公司也将 Eclipse 平台作为自身开发工具的基础。

3. 可扩展性强

Eclipse 采用微内核架构，其核心只有几兆大小，大家平时使用的代码编辑、调试、查找以及重构等功能都是以插件的形式提供的。我们不仅可以扩展现有插件，而且还可以提供扩展点，这样其他用户同样可以基于我们的插件开发扩展插件，从而满足用户的个性化需求。所以通常我们只需要实现个性化的功能即可，而通用的功能由基础插件来完成。比如作者曾经开发过一个 Python 的远程调试插件，由于 PyDev 已经提供了本地调试的功能，所以作者对 PyDev 进行了少量扩展开发就完成了这个插件。

4. 对开发工具的开发提供了强大的支持

Eclipse 提供了新建向导、代码编辑、调试、运行、图形化界面以及代码生成等开发工具常见功能的支持，这大大简化了一个复杂开发工具的开发。只需数十行代码就可以实现语法高亮、代码提示等代码编辑功能，只需数百行代码就可以实现调试功能，同样只需数百行代码就可以实现所见即所得的图形化编辑器，这一切让开发一个专业的开发工具变得如此简单。这样厂商只须按照自己领域相关的逻辑进行定制，其他的基础功能则由 Eclipse 提供，这使得厂商能够把更多的精力投入到自己熟悉的业务领域。比如我们要开发一个 Python 的所见即所得的界面绘制工具，那么只需要基于 GEF 进行少量开发即可实现一个所见即所得的图形化编辑器，而生成的 Python 代码的编辑、调试以及重构等功能则由现有的 PyDev 插件来完成。可以想象如果没有 Eclipse 的话，我们从头开发一个 Python 的图形化编辑器需要处理多少技术难题！

5. 基于 Eclipse 的产品更专业

一个专业的开发工具通常需要考虑很多问题，比如需要考虑被选择对象的属性编辑方式、长时间操作的进度条展示、编辑窗口的布局方式以及工具选项的配置等问题。针对这

些问题，Eclipse 的开发人员已经替我们考虑好了，我们开发的插件将自动拥有这些功能，这使得我们的插件显得更加专业。

6. 各种插件可以组合使用

以前每开发一个开发工具，都需要实现代码版本控制等功能，而在 Eclipse 中已经有了支持 VSS、CVS 和 SVN 等版本控制协议的插件，我们只要实现自己的开发工具即可，这些版本控制插件可以正确地与我们的插件组合使用，并且用户可以选择任何他们喜欢的版本控制插件，使得我们的工具使用起来更加灵活。

现在市场上已经有了 XML 编辑器、版本控制、UML 绘制工具及 EJB 开发工具等插件，并且这些插件也有不同厂商实现的多个版本，这样用户可以随意挑选他们喜欢的插件，在同一个 Eclipse 环境中任意组合这些插件来完成复杂的功能。

Eclipse 的出现使得 IDE 市场出现了一个新的格局，主流的开发工具都开始向 Eclipse 靠拢，这不仅使得开发工具的开发变得更加容易了，而且连中小型企业甚至个人也能开发出一个实用的开发工具来。这些基于 Eclipse 的开发工具不仅能够提高开发效率，而且还将用户统一到 Eclipse 平台中，减少了用户的学习成本。相信基于 Eclipse 的插件开发将成为未来开发工具的主流，就让我们马上开始激动人心的 Eclipse 插件开发学习之旅吧！

前　言

Eclipse 是一款优秀的、高度可扩展的平台，如果我们只是使用 Eclipse 的现有功能的话是无法发挥 Eclipse 的潜力的，如果能根据需要开发基于 Eclipse 的插件，那么将会大大提高开发效率。现在市场上已经有了一些 Eclipse 相关的书籍，但是大部分都是偏重于 Eclipse 的使用，很少有涉及到基于 Eclipse 的插件开发的书籍，即使有讲解 Eclipse 插件开发的，其内容也是浅尝辄止，没有对有一定复杂程度和实用性的插件的开发过程进行详细讲解。

Eclipse 的插件体系是非常复杂的，学习门槛也非常高，为了帮助开发人员掌握 Eclipse 的插件开发技术，从而开发出满足自己要求的插件，本书将系统地介绍 Eclipse 插件各方面的知识，并且通过实际的开发案例来演示这些知识的实战性应用。

书中对应的 Eclipse 版本为 Eclipse 3.2，可以从 http://www.eclipse.org 网站免费下载。

本书各章内容安排如下。

- 第 1 章介绍常用的 Eclipse 插件的安装和使用。
- 第 2 章以一个枚举生成器插件的开发为案例讲解一个简单、实用的插件的开发。
- 第 3 章介绍 Eclipse 插件开发中常用的基础知识。
- 第 4 章介绍插件对属性视图的支持。
- 第 5 章以 Hibernate 建模插件为案例讲解有一定复杂程度和实用性的插件的开发。
- 第 6 章以界面设计器插件为案例讲解基于 GEF 技术的图形插件的开发。

本书以案例贯穿始终：枚举生成器案例将我们带入 Eclipse 插件开发的大门，通过它读者可以学习到 SWT、JET、扩展点、插件的部署等 Eclipse 插件开发最基础的知识，从而具备了开发简单插件的能力；Hibernate 建模插件使得我们能够将编辑器、向导、JET、属性视图等技术有机地结合起来，开发出有一定复杂程度和实用性的插件；界面设计器插件不仅巩固了前面所学的知识，而且以通俗易懂的语言讲解了 Eclipse 插件开发中最难的知识点——GEF，通过学习此案例，读者将具备使用 GEF 开发 UML 编辑器、报表设计器等图形化编辑器的能力。

本书的随书光盘包含书中所有案例的源代码，光盘中还包含一个讲解 Eclipse 插件开发入门知识的视频教程，手把手地引导读者进入 Eclipse 插件开发的大门。

在此，我要感谢 CowNew 开源团队的朋友们一直以来的支持，还要感谢清华大学出版社，特别要感谢彭欣和宋延清两位编辑，他们给我的帮助使得我们的合作非常愉快，也使得本书能够顺利地完成和出版。

如果您对本书有任何意见和建议，您可以给我发送邮件：about521@163.com，本书相关的后续资料将会发布到 CowNew 开源团队网站(http://www.cownew.com)中。

杨中科

目　录

第 1 章　Eclipse 插件

插件机制是 Eclipse 非常大的一个特色。说到插件我们并不陌生，IE 浏览器、FireFox 浏览器、PhotoShop、Microsoft Office 等都提供了插件功能供开发人员进行功能扩展，Delphi、Visual Studio .NET 之类的开发工具也提供了插件机制，更不用说 Eclipse 的竞争对手 NetBeans 了，但是与 Eclipse 的插件比起来，它们的插件机制可以说是小巫见大巫了。Eclipse 是迄今为止唯一能够把插件机制发展到顶峰的产品。可以毫不夸张地说，如果没有如此强大的插件机制，Eclipse 是根本不可能成功的。

Eclipse 的核心(即运行时平台)只是一个很小的运行引擎，我们平时用到的大部分 Eclipse 功能其实都是某个插件的功能，比如 Java 代码编辑器、CVS 管理等功能都是 Eclipse 的插件，整个 Eclipse 就是一堆插件的有机整体。最为精妙的是，Eclipse 中的插件本身也可以提供扩展点，供其他插件进行再次扩展。

Eclipse 官方插件越来越丰富，包括 Web 开发插件 WTP、数据库开发插件 DTP、C++ 开发插件 CDT、商业智能报表插件 BIRT 等，而且各种第三方厂商生产的 Eclipse 插件也层出不穷，比如大名鼎鼎的 J2EE 开发插件 MyEclipse、数据库开发插件 SQLExplorer、UI 设计插件 SWT-Designer、XML 开发插件 XMLBuddy 等。

1.1　插件的安装

我们首先来看一下 Eclipse 中插件的安装。Eclipse 插件有 3 种主要的安装方式：直接复制安装、links 安装和在线安装。

1.1.1　直接复制安装

使用这种方式安装插件的时候只要直接将插件复制到 Eclipse 的默认插件目录中即可。这种方式最简单，只需把插件直接解压到 Eclipse 安装目录的 plugins 目录下即可，方便快捷。具体的解压方式随不同插件的不同目录结构而略有不同。

(1)　一种是最简单的结构，以 CowNewStudio 为例，它的安装包结构如图 1.1 所示。

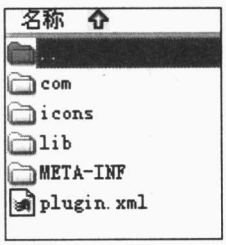

图 1.1　插件安装包结构

也就是说插件没有其他的层级结构，plugin.xml 就在安装包根目录中。对于这种目录结构的插件，只要在 Eclipse 安装目录的 plugins 目录下建立一个目录，比如名字为 CowNewStudio_1.0.0，然后把安装包中的内容解压到这个目录下即可。

(2) 另一种是标准的目录结构，以多语言配置插件 JInto 为例，它的安装包结构如图 1.2 所示。

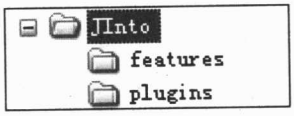

图 1.2 JInto 的安装包结构

也就是说安装目录下有 features、plugins 两个目录，plugin.xml 在 plugins 目录下，对于这种目录结构的安装包，安装起来会有一点麻烦，需要把安装包 plugins 目录下的文件解压到 Eclipse 安装目录的 plugins 目录下，然后把安装包 features 目录下的文件解压到 Eclipse 安装目录的 features 目录下。其实不安装 features 目录也是可以使用的，但是安装 features 目录以后能在 Eclipse 中更好地管理此插件。

安装完成后，重启 Eclipse 即可。

1.1.2 links 安装方式

以上的安装方式非常简单，但是当安装了许多插件之后，Eclipse 的尺寸会变得非常大，而且不便于更新和管理众多插件，如果要删除插件或者更新插件则需要找到对应的文件进行删除或者更新。而用 links 方式安装 Eclipse 插件，则可以每个目录安装一个插件，管理起来是非常方便的。

仍然以 JInto 为例，安装步骤如下所示。

(1) 解压插件到某文件夹，如 E:\eclipse3.2\3rdplugins\JInto\eclipse，如图 1.3 所示。注意最后一级文件夹名字必须为 eclipse，且插件解压以后的 features、plugins 文件夹必须是此文件夹的直接子文件夹。而 E:\eclipse3.2\3rdplugins\JInto 则被称为此插件的安装目录。

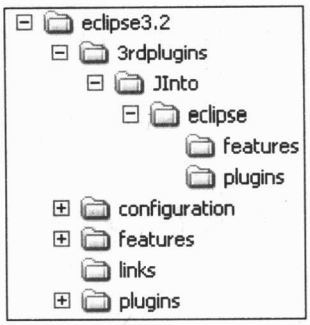

图 1.3 JInto 的 Links 安装方式

(2) 如果 E:\eclipse3.2 下没有 links 文件夹，则在 E:\eclipse3.2\下新建文件夹 links(该文

件夹应位于 eclipse.exe 文件所在的目录中)，这样就得到了 E:\eclipse3.2\links。

(3) 在 E:\eclipse3.2\links 下新建一个 link 文件，比如 JInto.link，在 JInto.link 中写入“path=E:/eclipse3.2/3rdplugins/JInto/”，或者“path=E:\\eclipse3.2\\3rdplugins\\JInto\\”。需要注意的是，这里的路径分隔符只能使用“/”或者“\\”而不能使用“\”，这个地方很容易出错。

(4) 重启 Eclipse。

采用上面两种安装方式，有的时候插件安装完成并重启 Eclipse 后，会出现在 Eclipse 中并没有看到此插件的情况，其原因有如下几个：

- Eclipse 没有发现此插件被安装，所以就没有加载插件。解决方法是，给 Eclipse 增加一个-clean 启动参数，比如在 Windows 下，可以在打开命令提示符窗口并进入 Eclipse 的安装目录，然后敲入如下命令“eclipse.exe -clean”来启动 Eclipse，一般就可以看到此插件了。以后启动 Eclipse 则无需每次都通过此种方式。网络上有的文章中提到，如果安装的插件没有在 Eclipse 中发现的话可以通过删除 configuration 目录来解决，这种方式虽然能够解决问题，但是由于在线安装方式的相关文件也是存储在此文件夹中的，删除此文件夹会造成以在线安装方式安装的插件在自动更新的时候出现问题。
- 插件所依赖的其他插件没有安装。比如很多插件都依赖于 GEF 插件，但是这个插件默认是不安装的，因此就会导致插件安装不成功。解决方法就是安装所有被依赖的插件。如何得知此插件依赖于哪些插件呢？如果您熟悉 Eclipse 插件开发，只要打开插件的 META-INF 目录下的 MANIFEST.MF 文件，查看并分析 Require-Bundle 段的内容即可；如果您不熟悉 Eclipse 插件开发的话就只能求助于插件作者或者他人了。
- 在涉及安装多个插件时，对应不同的插件，应该在 links 目录中使用多个*.link 文件。每个*.link 文件中应当只包含一个插件路径。这些*.link 文件的名称可以任意指定，但在双击 eclipse.exe 启动时通常需要尝试两次方能正确识别*.link 文件。
- Eclipse 版本不对。Eclipse 的插件平台在版本升级的时候会尽量保证向下兼容，但是由于各种原因，在低版本下开发的插件有的时候拿到高版本下有可能导致安装失败。所以安装的时候一定要注意插件相应的 Eclipse 版本号。
- JRE 版本问题。有一些插件是采用高版本的 JDK 开发的，比如 CowNewStudio 就是采用 JDK 1.5 开发的，因此如果运行在 JRE 1.4 下就会失败。

1.1.3 Eclipse 在线安装方式

可以看到，上面的两种安装方式都比较麻烦，稍有不慎就会安装失败，安装失败之后的解决也是很麻烦的。那么有没有更方便的安装方式呢？当然有，那就是在线安装方式。

下面以反编译插件 Jode 为例来讲解在线安装方式。在安装之前必须得到此插件的安装地址，一般可以在插件的网站中获得此地址，例如 Jode 的安装地址是“http://www.technoetic.com/Eclipse/update”。

(1) 选择主菜单中的【帮助】|【软件更新】|【查找并安装】命令，在弹出的对话框中选择【搜索要安装的功能部件】选项，单击【下一步】按钮，弹出如图 1.4 所示的对话框。

(2) 单击【新建远程站点】按钮，弹出如图 1.5 所示的对话框。在【名称】文本框中输入“Jode”(这个名字可以任意选择)，在 URL 文本框中输入“http://www.technoetic.com/eclipse/update”，然后单击【确定】按钮。

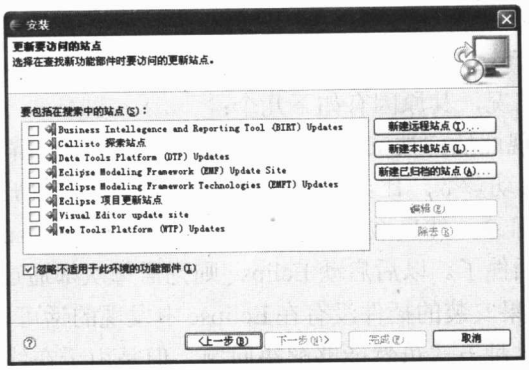

图 1.4 【安装】对话框

图 1.5 【新建更新站点】对话框

(3) 确保【安装】对话框中选中了刚才新建的“Jode”站点，单击【完成】按钮。

(4) 接着就会显示如图 1.6 所示的【更新】对话框以供选择要安装的插件。

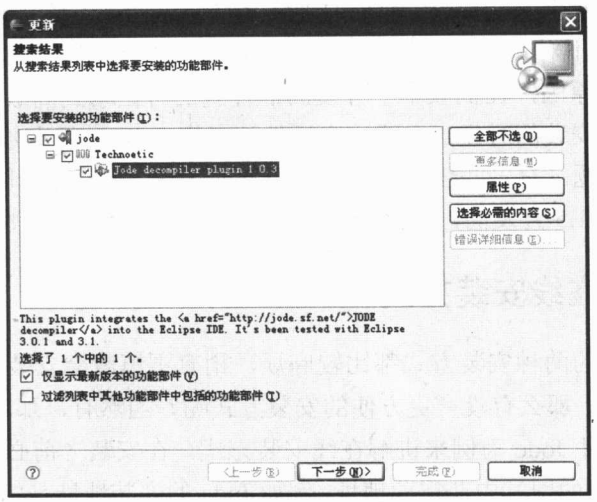

图 1.6 【更新】对话框

因为这里只有一个插件，所以我们选择“Jode decompiler plugin 1.0.3”，单击【下一步】按钮。

(5) 在协议页面中选中【我接受许可协议中的条款】单选按钮，单击【下一步】按钮。

(6) 在新的页面中要求我们选择插件安装的位置，如图 1.7 所示。

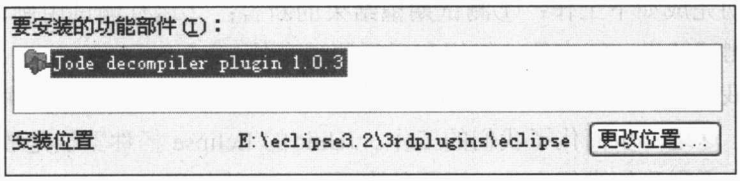

图 1.7 选择插件安装位置

单击【更改位置】按钮，选择要安装的功能部件的安装位置，然后单击【完成】按钮。

(7) Eclipse 就开始自动下载插件文件并安装。安装速度依网络速度的不同而有差异。安装完成后 Eclipse 会提示重启 Eclipse，我们选择重新启动即可。

1.2 内置 JUnit 插件的使用

单元测试是软件开发过程中一个很重要的步骤，它降低了测试的成本并提高了产品的质量。其实单元测试并不神秘，在 xUnit 之类的工具推出之前人们已经在进行大量的单元测试了。比如在写一个类库的时候，经常为这个类写一个 main 方法，在这个方法中写一些测试代码。比如：

```
public class MyMath
{
    public static int add(int i1,int i2)
    {
        return i1+i2;
    }

    public static void main(String[] args)
    {
        System.out.println(add(1,2));
        System.out.println(add(1,-1));
        System.out.println(add(0x7fffffff,0x7fffffff));
    }
}
```

有人认为写测试代码会浪费时间，而且在有些公司中测试代码的编写是不计工作量的，因此开发人员就忽略了测试代码的书写，导致代码质量变差，Bug 满天飞，花在找 Bug、解决 Bug 上的时间也越来越多，实际效率大大降低。

其实这些人的做法也是有一定道理的，写测试代码确实需要消耗一定的时间，而且每

次进行单元测试的时候也要花时间去检验输出结果是否正确。

为了解决这些问题，Erich Gamma 和 Kent Beck 编写了 JUnit。JUnit 是一个开放源代码的 Java 测试框架，用于编写和运行可重复的测试，它是单元测试框架体系 xUnit 的 Java 实现版本，类似的产品包括 .NET 下的 NUnit、C++下的 CPPUnit、Delphi 下的 DUnit 等。这个工具可以自动完成如下工作：①测试期望结果的断言；②运行测试用例，并通过图形或文本界面报告测试结果；③方便地组织和运行测试套件。

Eclipse 为我们提供了 JUnit 的 Eclipse 插件，这样我们就可以在 Eclipse 中书写测试代码、运行测试用例，这进一步简化了我们的工作。JUnit 的 Eclipse 插件默认是放在 Eclipse 的安装包中的，因此无需我们进行安装。下面就来看一下它的使用。

(1) 首先建立一个 Java 项目，然后添加一个 Java 类文件，这个类只有一个方法 getExtension，这个方法用来从文件全路径的字符串中取得文件的扩展名。这个功能用正则表达式写是最好的，但是为了更好地演示 JUnit 的使用，我们这里用最基本的字符串操作函数来完成，代码如下：

```
public class FileNameParser
{
    public String getExtension(String fileName)
    {
        int lastDotIndex = fileName.lastIndexOf('.');
        return fileName.substring(lastDotIndex+1, fileName.length());
    }
}
```

(2) 右击 FileNameParser.java 选项，选择【新建】|【JUnit 测试用例】命令，Eclipse 会弹出如图 1.8 所示的对话框。

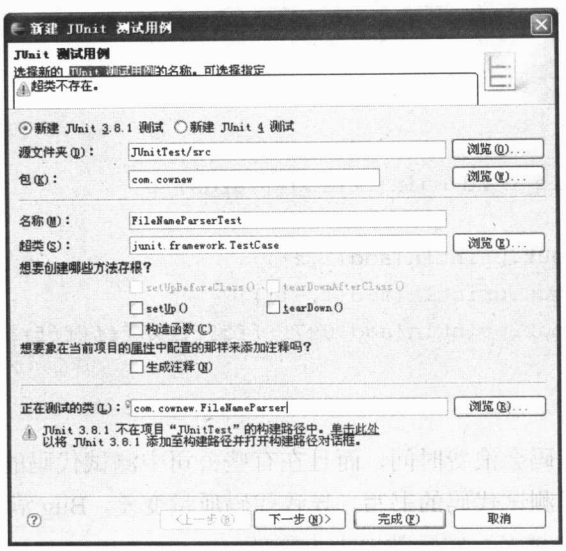

图 1.8 【新建 JUnit 测试用例】对话框

这里提示 JUnit 所用到的包没有在构建路径上，单击“单击此处”链接，将 JUnit 的包添加到构建路径中。由于我们需要在 setUp 方法中创建 FileNameParser 类的实例，因此可以在【想要创建哪些方法存根?】选项组中选中 setUp 复选框，然后单击【下一步】按钮，向导会转向如图 1.9 所示的界面。

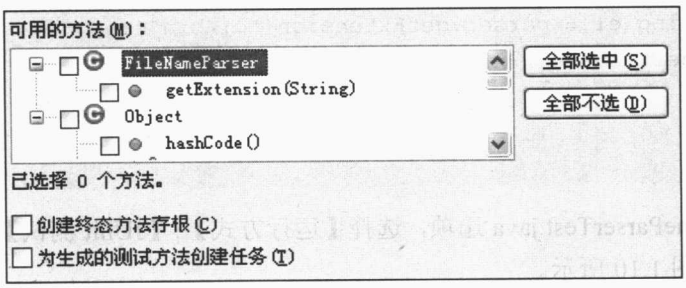

图 1.9　选择要测试的方法

在此处选择要测试的方法 getExtension，然后单击【完成】按钮。

JUnit 就为我们生成了下面的测试代码：

```
import junit.framework.TestCase;

public class FileNameParserTest extends TestCase
{
    protected void setUp() throws Exception
    {
        super.setUp();
    }
    public void testGetExtension()
    {
        fail("尚未实现");
    }
}
```

(3)　修改生成的代码为：

```
import junit.framework.TestCase;

public class FileNameParserTest extends TestCase
{
    private FileNameParser parser;
    protected void setUp() throws Exception
    {
        super.setUp();
        parser = new FileNameParser();
    }
```