使用 Eclipse 插件 Ruby Development Tools

级别: 中级

Neal Ford (neal.ford@gmail.com), 应用程序架构师, ThoughtWorks

2005 年 12 月 08 日

本文介绍如何使用 Eclipse 插件 Ruby Development Tools（RDT），这个插件使 Eclipse 能够成为一流的 Ruby 开发环境。那些希望利用 Eclipse 社区丰富的基础设施来支持开发的 Ruby 开发人员会从本文中受益，对有兴趣使用 Ruby 的 Java™ 开发人员也会有所收获。

为什么要使用 **Ruby**?

为什么 Java 开发人员会关心 Ruby? Ruby 是 10 年前在日本开发出来的通用脚本语言。与流行的信念相反，它是一种纯面向对象语言。与 Java 技术不同，Ruby 没有标量，所以所有东西（包括整数）都是一类对象。Ruby 的语法很大程度上借鉴于 Smalltalk、Python 和 Ada。与 Java 语言相同的地方是，Ruby 也是一种单继承语言，但是它提供了 Java 技术所缺乏的某些高级特性，比如闭包（与 steroids 上的匿名内部类相似）和 mix-ins（与接口相似，但是它们与类的绑定不太紧密）。Ruby 也具有很高的可移植性，可以在所有主流操作系统上运行。

Ruby 现在已经很流行了，人们开始用它建立各种应用程序。因为它是解释语言而且使用动态类型，所以可以在运行时做许多极其灵活的工作，而这在 Java 中是非常困难的。动态类型和表达语法所支持的神奇功能之一是，能够在 Ruby 中创建领域特定的语言，这使开发人员能够在更高的抽象级别上工作，从而脱离语言的"原始"语法。Ruby on Rails 是一个用于创建带后端数据库的 Web 应用程序的框架，它展示了这种优雅性。Rake（Make 和 Ant 相结合的 Ruby 版本）也展示了 Ruby 的这种强大能力。

使用 Ruby 的另一个理由是，许多敏锐的开发人员已经开始使用它了。那些在 1996 年就认识到 Java 技术即将流行的开发人员（比如 Glenn Vanderburg、Bruce Tate 和 Martin Fowler）现在已经开始使用 Ruby。即使您还不打算全面转向 Ruby，现在也应该研究一下这种语言了。

对于用 Ruby 进行广泛的开发，主要的限制因素之一是缺少一个出色的开发环境（对于那些不想学习 Emacs 的人来说尤其如此）。RDT 改变了这种状况。在您喜欢的 Eclipse IDE 中使用 Ruby，这会使您感到舒适。

回页首

准备

在开始使用 Ruby 之前，必须安装（或者验证已经具有了） Ruby 解释器和库以及 Ruby Development Environment。

获得 **Ruby**

可以获得适合所有主流平台以及几个次要平台的 Ruby 版本。实际上，您的 Linux® 或 Cygwin 发行版可能已经包含了 Ruby。转到命令提示符下并输入 `ruby -v`。

如果看到一个版本号，就说明 Ruby 已经有了。如果没有看到版本号，就需要获取 Ruby。先寻找适合您平台的 发行版。

如果您正在使用 Windows®，就更容易了。RubyForge（与 SourceForge 功能相似）有一个称为 One-Click Ruby Installer 的项目，这个程序会在 Windows 上建立 Ruby 环境（参阅 参考资料）。它还包含几个工具，包括称为 FreeRide 的 IDE，但是如果使用 RDT，则可以不用理会大多数这些工具。

回页首

获得文档

在开始使用一种新语言时，文档和其他参考资料是非常重要的。可以在 Ruby 主 Web 站点上获得 Ruby 文档的在线参考，但是您会发现这里的文档有点儿过时了（它针对 Ruby V1.6，而当前版本是 1.8.2）。这是因为最新的文档还没有从日文翻译成英文。但是在 Ruby-doc.org 上可以找到最新的文档。它包含 API 级文档（与 Javadoc 等同）以及一些教程和书籍。（参阅 参考资料。）

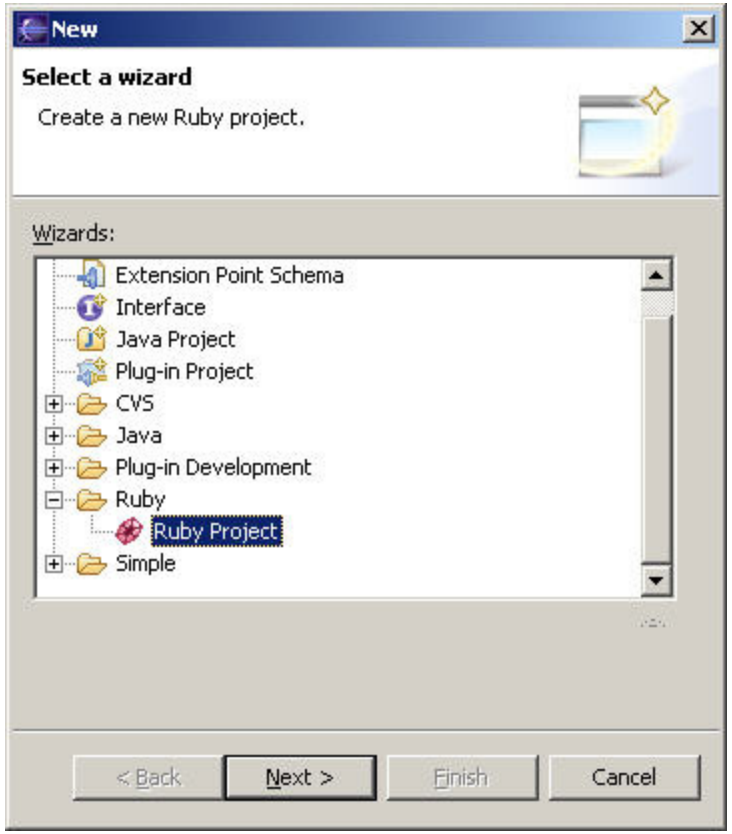
如果您关心 Ruby 开发，那么应该尽快获得 Dave Thomas 和 Andy Hunt 撰写的 *Programming Ruby: The Pragmatic Programmer's Guide*（参阅 参考资料）。这是对 Ruby 的标准介绍并且介绍了 Ruby 库的需求。在阅读这本书的同时，还可以阅读 Dave Thomas 撰写的 *Agile Development with Ruby on Rails*，其中介绍了 Ruby on Rails。

获得 **RDT**

既然已经在计算机上安装了可以工作的 Ruby 版本并且获得了文档，现在就需要 RDT 了（参阅 参考资料）。这个 Eclipse 插件提供了许多特性，您在编辑代码时会慢慢熟悉这些特性。RDT 是一个标准的 Eclipse 插件，具有特性和插件，所以可以将 zip 文件直接解压缩到 Eclipse 文件夹。存档文件中的路径会建立目录结构。

现在可以创建 Ruby 项目了（参见图 1）：

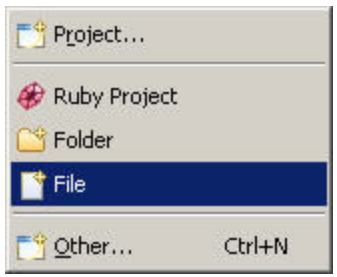
图 **1.** 创建新的 **Ruby** 项目



与 Java 技术相比，Ruby 对名称和目录结构的要求宽松多了。在 Ruby 中创建一个项目实际上只是创建一个目录和一个 `.project` 文件（这里不需要 `.classpath` 文件，因为 Ruby 没有类路径）。与 Java 技术相比，另一个显著差异是 Ruby 项目向导并不创建显式的 `src` 和 `bin` 目录。Ruby 是解释语言，所以没有输出文件夹。如果项目比较小，那么可以将 Ruby 源代码文件与项目文件放在同一个文件夹中。也可以创建自己的目录结构。您会发现，与 Java 语言相比，Ruby 不太关心目录结构。

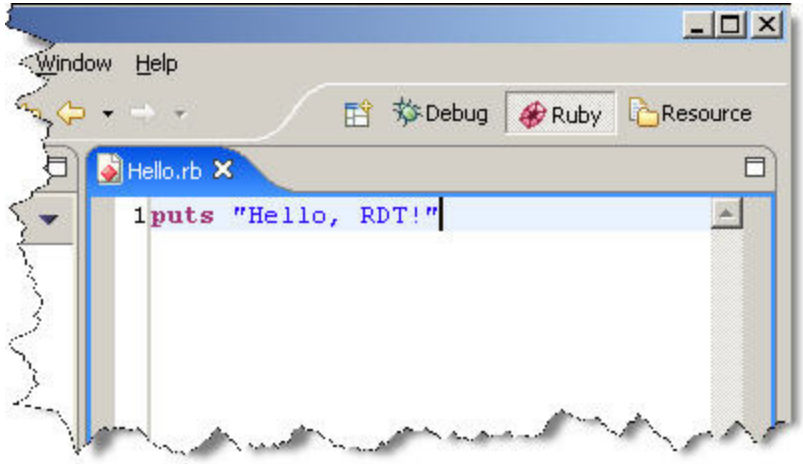
接下来，需要一个 Ruby 源代码文件。没有专门用于创建 Ruby 源代码文件的向导。与 Java 技术不同，对于 Ruby 源代码文件的结构没有什么要求，所以要创建 Ruby 文件，只需使用项目的右击菜单创建一个新文件。

图 **2.** 创建 **Ruby** 源代码文件



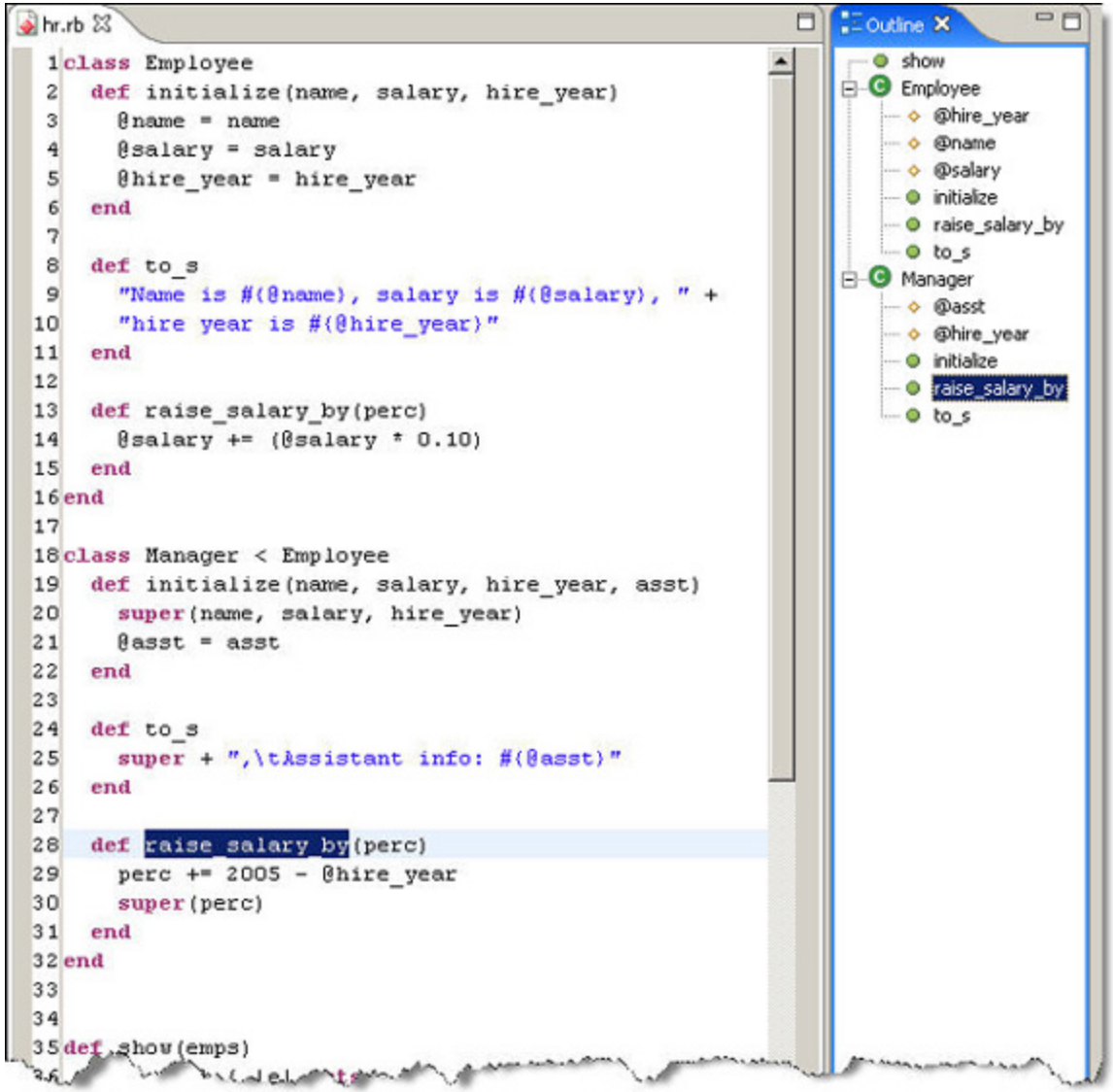
不要忘记在文件名后面加上标准的扩展名 `.rb`，这是一般的 Ruby 扩展名。创建 Ruby 文件应该会打开 Ruby 透视图。

图 **3.** 创建 **Ruby** 源代码文件会打开 **Ruby** 透视图



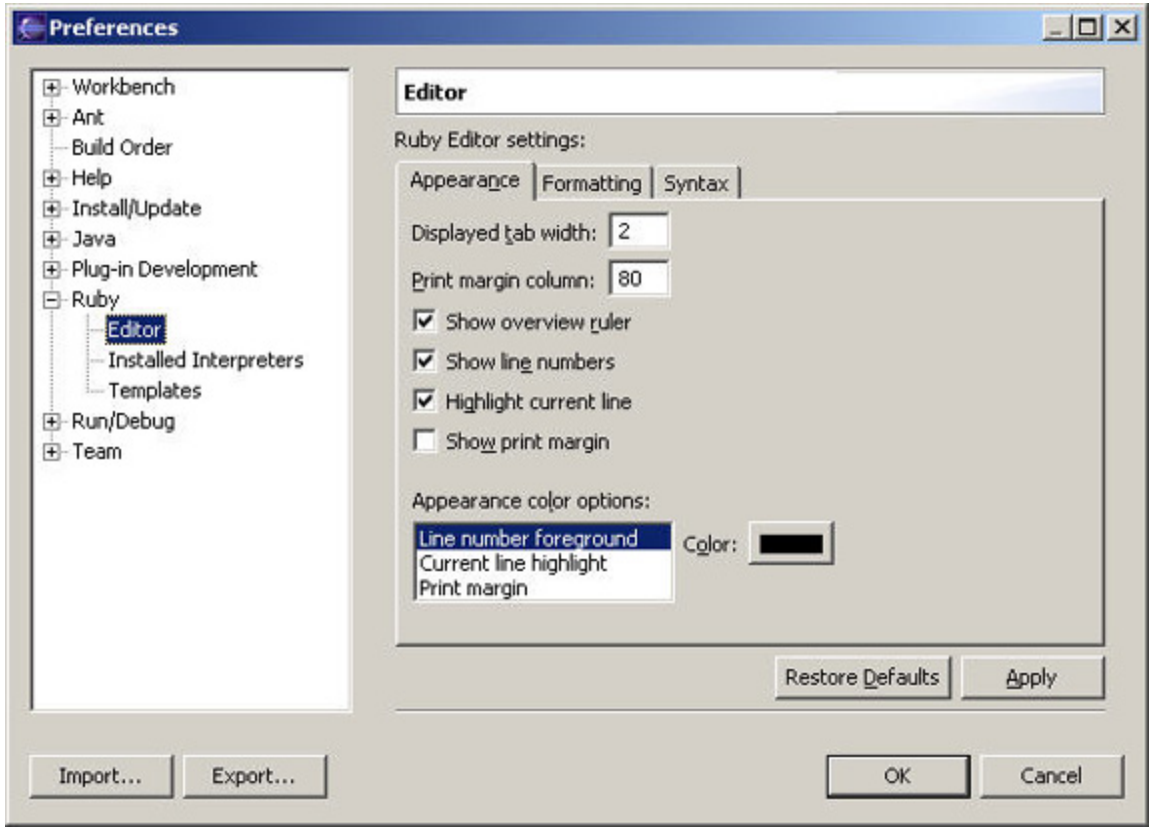
Ruby 透视图还提供一个大纲视图，这与 Java 透视图提供的大纲视图相似。与 Java 大纲视图相似，它允许导航 Ruby 源代码文件的元素。在图 4 中，`raise_salary_by` 方法在大纲视图和源代码视图中高亮显示。

图 **4.** 大纲视图允许在源代码文件中进行导航



与其他复杂的插件一样，RDT 也在 **Window > Preferences** 对话框中增加了定制特性。这个首选项对话框如图 5 所示。

图 **5. RDT** 的定制首选项



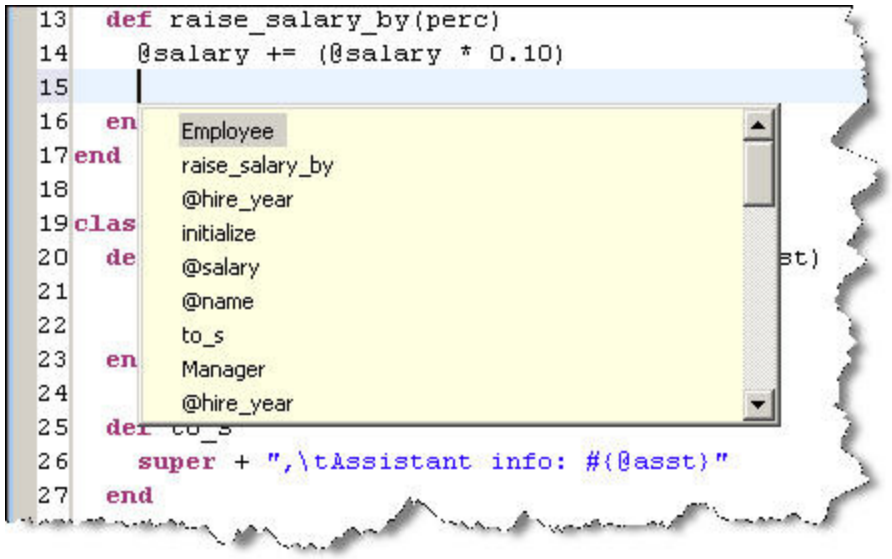
首选项菜单项允许修改语法高亮显示方式、格式化（在 Ruby 中标准的缩进是两个空格，不是四个，所以要做某些调整）等等。它还允许定制编辑器模板以及选择解释器。

回页首

RDT 编辑器

在 Java 技术环境中，我们已经习惯了高级的编辑器特性，这使我们在转移到没有提供这些特性的其他环境时不太适应。Ruby IDE 缺乏的特性之一是 Content Assist，这种特性针对标识符进行上下文相关的查找。RDT 编辑器对 Ruby 代码提供了 Content Assist。

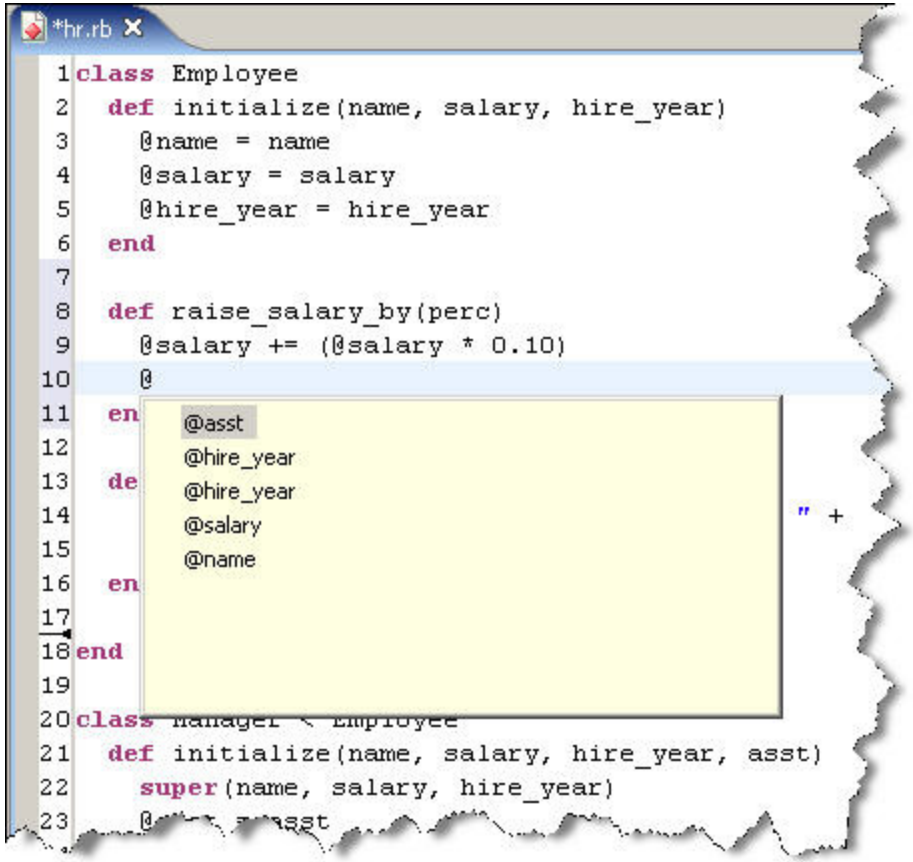
图 **6. RDT** 编辑器提供 **Content Assist**



RDT 编辑器中的 Content Assist 没有 Java 环境中那么有针对性，因为 Ruby 支持动态类型。在 Ruby 中，不能将类型赋给变量或者方法返回值。标识符上下文在运行时决定类型。Ruby 使用所谓的"duck typing" —— 也就是说，如果它接受"鸭叫"消息，那么它一定是鸭子。对于那些习惯于强类型语言的人来说，这可能像是一种阻碍，但是这种弱类型耦合使 Ruby 语言能够实现某些更强大的特性。例如，在 Ruby 中可以编写一个异常处理程序，如果调用一个不存在的方法就会触发这个程序，这个异常处理程序可以动态地合成这个方法并且调用它。在强类型语言中很难实现这种元编程。

Content Assist 弥补的特性之一是 Ruby 对标识符使用的特殊命名约定。例如，在 Ruby 中，所有成员变量在第一次使用时才存在，而且都由 @ 符号作为名称的第一个字符。如果使用 Content Assist 查找成员变量，那么可以输入 @，并且只看到成员变量。

图 **7. Ruby** 中的命名约定帮助实现 **Content Assist**



动态类型仍然会妨碍 Ruby 中的上下文敏感性。在图 7 中，合法的成员变量只是在上面方法声明中出现的成员变量，即 `@name`、`@salary` 和 `@hire_year`。Content Assist 挑选出的其他成员变量来自后面定义的另一个类。RDT 编辑器还不完善，无法过滤出所有语法上正确但是语义上不正确的条目。

回页首

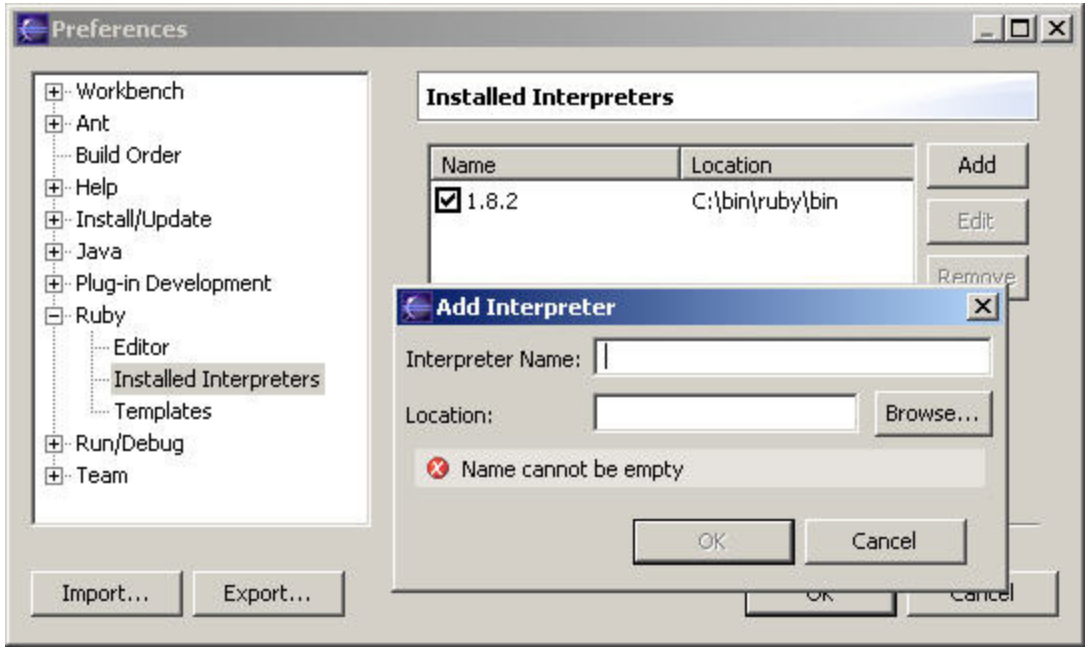
运行和调试

IDE 的关键特性之一是能够在构建应用程序的环境中对应用程序进行运行和调试。RDT 支持这两种功能。

指定解释器

Ruby 是一种解释语言，所以必须将一种解释器与环境相关联，然后 RDT 才能运行或调试应用程序。这种关联是在 **Windows > Preferences** 对话框中 **Ruby** 标题下面的 **Installed Interpreters** 项中设置的。

图 **8.** 将 **Ruby** 解释器与环境关联起来

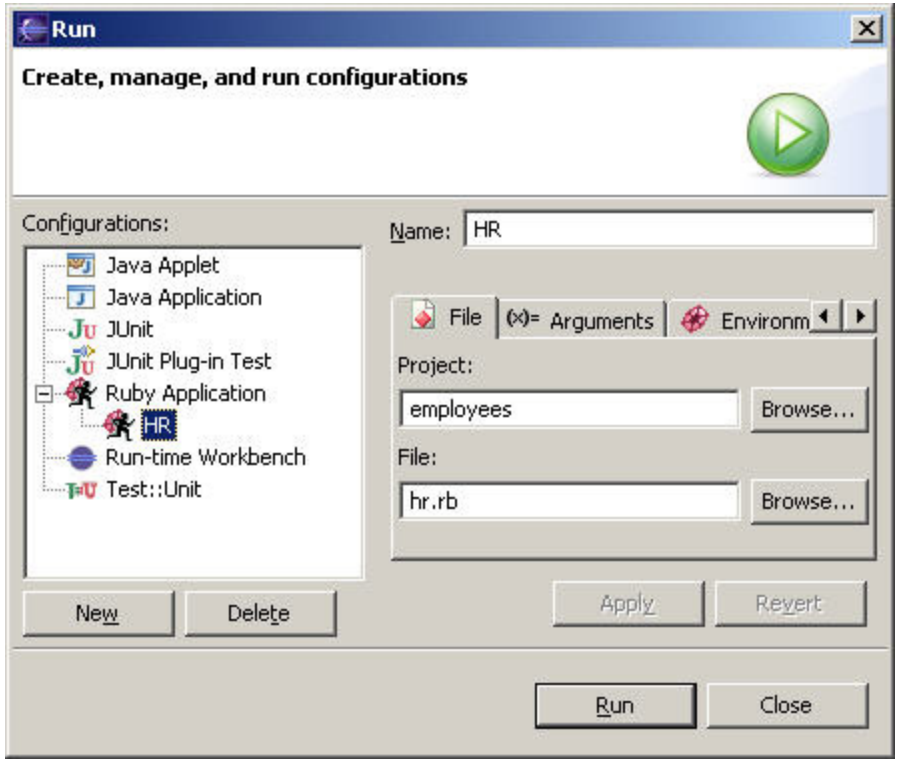


将"Location"文本域指向您使用的 Ruby 版本的 `bin` 目录。RDT 会找到所需的其他信息。关联了解释器之后，就可以运行应用程序了。

运行 **Ruby** 应用程序

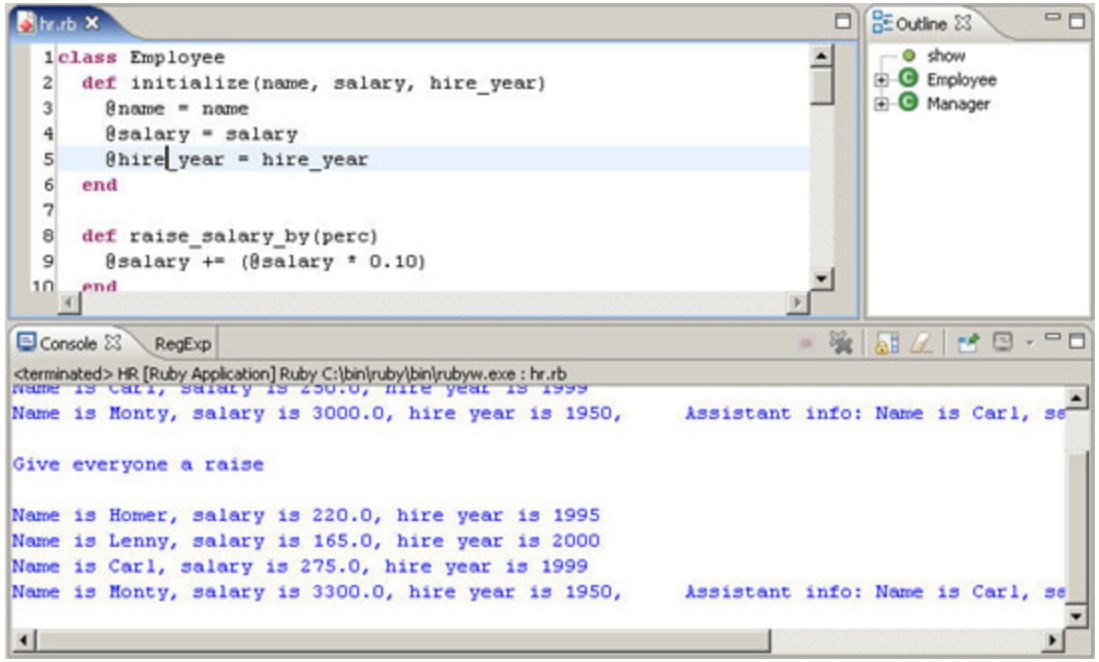
运行 Ruby 应用程序与运行 Java 应用程序实际上完全一样。使用 Run 菜单创建一个 Run 配置。

图 **9.** 在 **RDT** 中建立 **Run** 配置



在运行应用程序时，RDT 启动 Ruby 解释器并且在 Eclipse 工作区底部的一个控制台窗口中运行这个应用程序。

图 **10.** 在 **RDT** 中运行 **Ruby** 应用程序



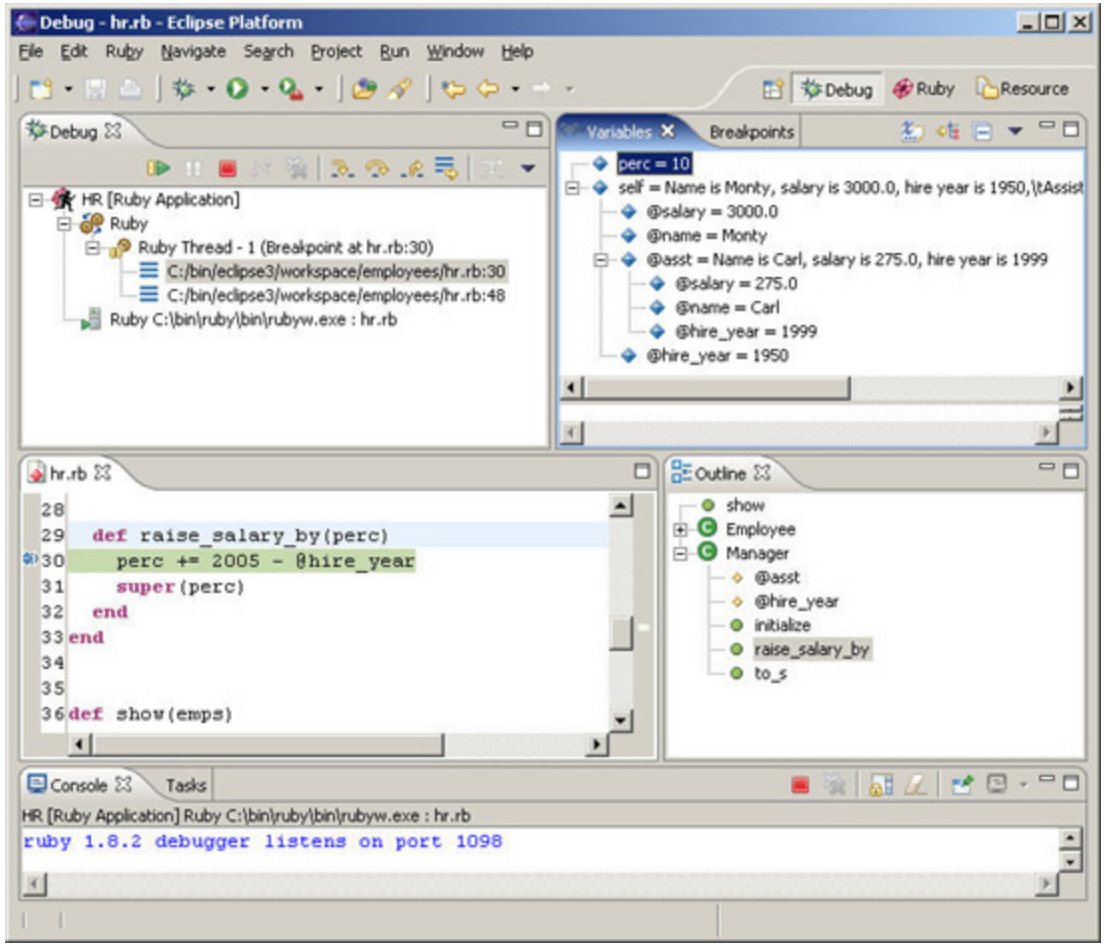
这个例子显示如何运行一个控制台应用程序，但是运行其他类型的应用程序（比如图形化应用程序）也是一样的。

用 **RDT** 进行调试

IDE 需要的最关键的特性之一是能够有效地调试应用程序。Ruby 解释器包含一个命令行调试器，但是在这个图形化工具的时代谁还想使用命令行调试器？幸运的是，Ruby 解释器还通过一个特定（可配置）的端口广播调试信息，RDT 这样的工具能够监听这个端口并且提供开发人员期望的调试支持类型。

要调试 Ruby 应用程序，需要创建一个 Debug 配置，就像前面创建 Run 配置一样。然后，在左边的空白处点击以设置断点，并使用调试器启动应用程序。与 Java 技术一样，IDE 将询问您是否想切换到 Debug 透视图。回答之后，将看到下面这样的屏幕：

图 **11.** 在 **RDT** 中调试 **Ruby** 应用程序



RDT 提供了与 Java 技术相同的调试支持级别。左上方的窗格显示当前正在执行的线程。右上方的窗格显示变量的值。与 Java 语言中一样，可以深入到对象中，查看它们的底层成员变量值。左边中间的窗格显示正在运行的应用程序的源代码，右边中间的窗格显示大纲视图（它在这里的工作方式与在编辑器中一样，让开发人员能够通过点击标识符来进行导航）。Debug 窗口的底部显示 Ruby 解释器在端口 1098 上广播调试信息，RDT 在这个端口上监听调试信息。

调试支持是 RDT 的关键特性。即使您的编辑器有出色的 Ruby 支持，但是仍然必须依赖命令行调试器来调试应用程序。拥有一个功能全面的调试器可以显著提高生产效率。

回页首

测试

对于 Java 开发人员，Ruby 最难掌握的特性之一是动态类型。如果您已经习惯了强类型语言，那么动态类型看起来可能会导致混乱。动态类型支持所有高级的元编程技巧，这在强类型语言中是很难实现的，甚至不可能。当然，还会失去编译时检查的安全保障。是否有办法同时获得这两种环境的优势呢？

对于语言，单元测试是必需的，但是在动态语言中它特别重要。单元测试能够比单纯编译揭示出更多的问题。实际上，您应该换个角度看待单元测试和编译之间的关系。近来，在一个软件开发专家讨论会上，软件开发思想家 Relevance 公司的 Stuart Halloway 指出，“在 5 年内，我们将看到编译成为一种比较弱的单元测试形式。”单元测试检验代码是否做了开发人员希望它做的事儿，而不只是对类型

进行拼写检查。

既然单元测试在 Ruby 环境中如此重要，那么当然希望 RDT 可以使运行单元测试更容易。它确实做到了。单元测试已经包含在 Ruby 中，所以不需要下载任何额外的扩展代码。Ruby 库包含一个 TestCase 类和 TestSuite 的概念。可以像创建其他 Ruby 类一样创建单元测试，这需要对 `Test::Unit::TestCase` 进行子类化。清单 1 是一个称为 `Employee` 的示例类。

清单 **1.** `Employee` 类

```
class Employee
  def initialize(name, salary, hire_year)
    @name = name
    @salary = salary
    @hire_year = hire_year
  end

  attr_reader :name, :salary, :hire_year
  def raise_salary_by(perc)
    @salary += (@salary * (perc * 0.01))
  end

  def to_s
    "Name is #{@name}, salary is #{@salary}, " +
    "hire year is #{@hire_year}"
  end

end
```

对应的单元测试是：

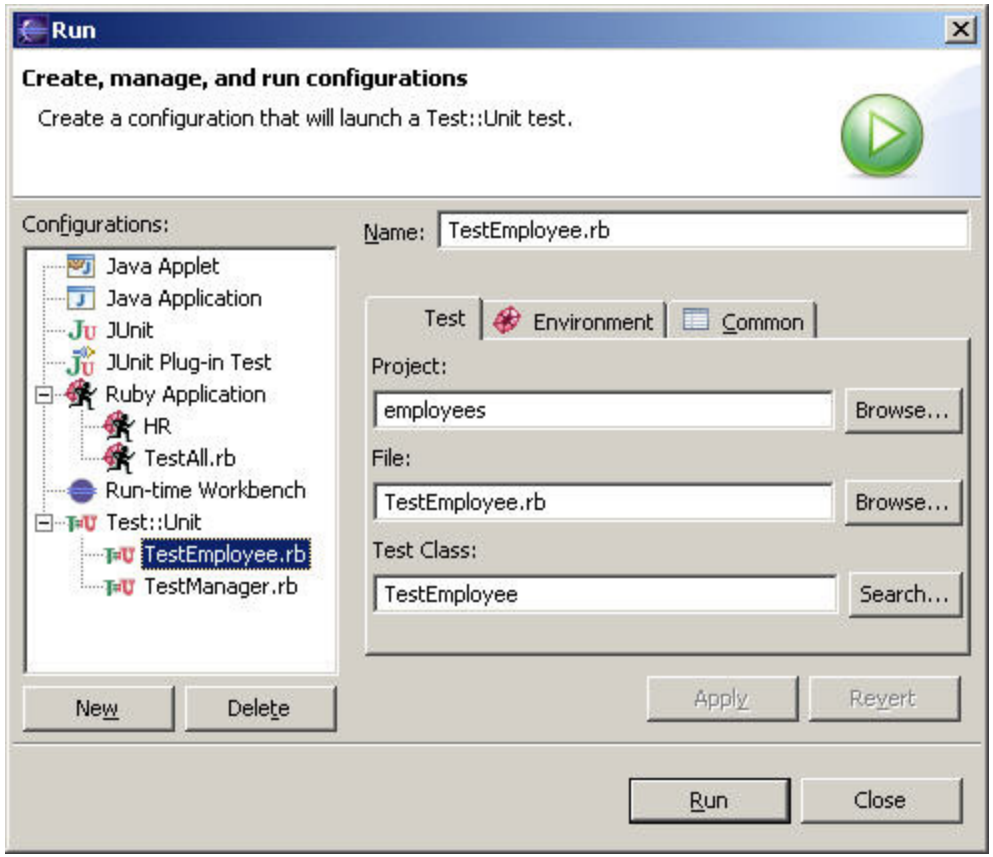
清单 **2.** `Employee` 类的单元测试

```
require 'test/unit/testcase'
require 'test/unit/autorunner'
require 'hr'
class TestEmployee < Test::Unit::TestCase
  @@Test_Salary = 2500
  def setup
    @emp = Employee.new("Homer", @@Test_Salary, 2003)
  end
```

```
  def test_raise_salary
    @emp.raise_salary_by(10)
    expected = (@@Test_Salary * 0.10) + @@Test_Salary
    assert( expected == @emp.salary)
  end
end
```

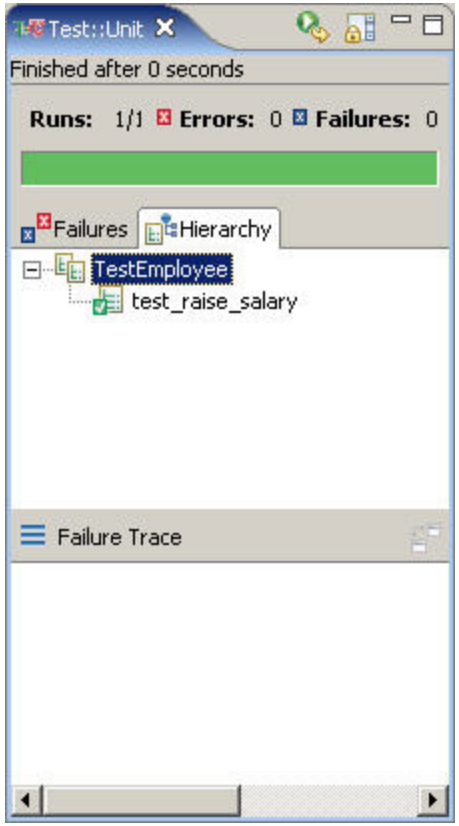
要运行这个单元测试，应该为单元测试类创建一个 `Test::Unit` 类型的 Run 配置。

图 **12. RDT** 包含 **`Test::Unit` Run** 配置



在运行这个测试时，可以获得与 Java 单元测试相同的支持元素，包括左下角与 JUnit 相似的面板。

图 **13.** 在 **IDE** 中运行的单元测试示例



在 Ruby 中还可以创建 TestSuites。TestSuites 是定义套件方法的 Ruby 类，这个方法返回 TestSuite。TestSuite 由在每个 TestCases 中自动定义的套件组成。清单 3 是两个类的 TestSuite 示例。

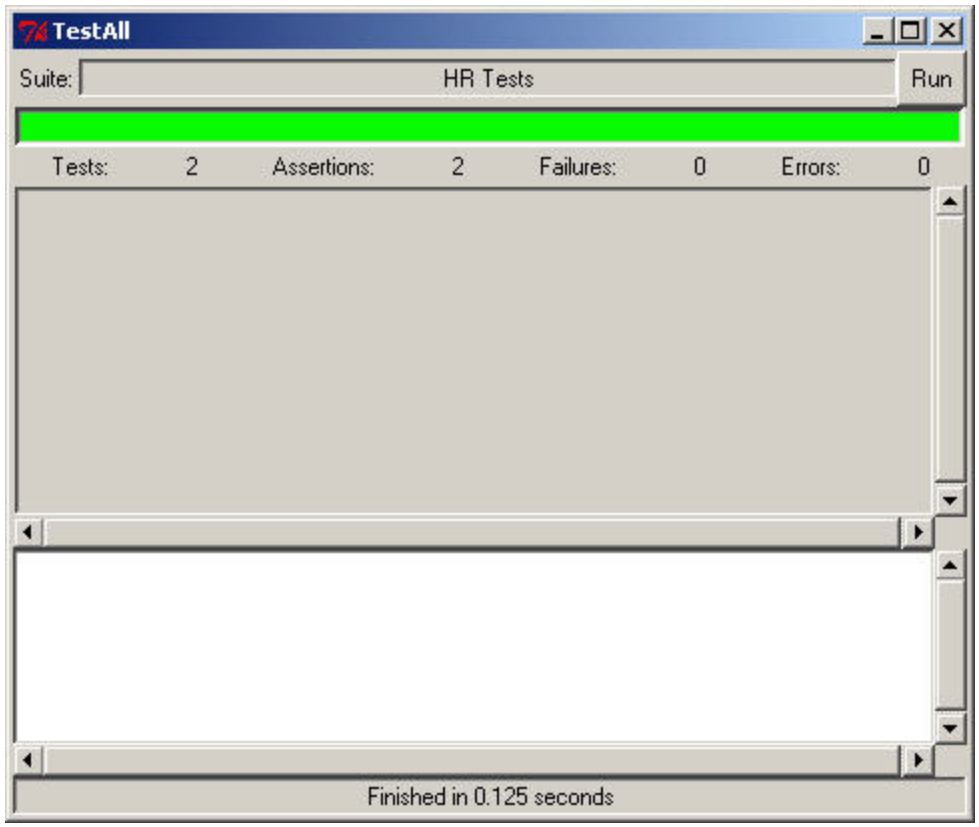
清单 **3.** 两个类的 **TestSuite** 示例

```
require 'test/unit/testsuite'
require 'test/unit/ui/tk/testrunner'
require 'test/unit/ui/console/testrunner'
require 'TestEmployee'
require 'TestManager'
class TestSuite_AllTests
    def self.suite
        suite = Test::Unit::TestSuite.new("HR Tests")
        suite << TestEmployee.suite
        suite << TestManager.suite
        return suite
    end
```

```
end
#Test::Unit::UI::Tk::TestRunner.run(TestSuite_AllTests)
Test::Unit::UI::Console::TestRunner.run(TestSuite_AllTests)
```

与前面运行单一 TestCase 的例子不同，套件作为单独的应用程序运行。Ruby 有两种显示 TestSuite 结果的方法。第一种是 Console Test Runner，它在控制台输出结果。第二种是 Tk TestRunner，它创建一个对话框来显示测试结果。Tk TestSuite 对话框见图 14。

图 **14.** 图形化的 **TestSuite** 对话框



回页首

未来的发展

RDT 的当前版本是 0.50。它的开发人员正在为下一个版本 0.60 而努力。下一个版本中计划的改进包括：

- 代码折叠 —— 可以将类和方法的代码折叠起来。
- 大纲视图 —— 更详细，支持局部变量。

RI 视图 —— 从 一个 RDT 视图使用 Ruby 的 ri 实用程序。

- 任务标记 —— 在 Ruby 注释中为可配置的关键字（比如 TODO、FIXME）创建任务。
- 编辑器改进 —— 自动补齐方括号、圆括号和单/双引号；更好的代码辅助。
- 审查的快捷方式 —— 对 Debug 会话期间经常使用的审查提供可配置的快捷方式，比如显示对象的所有方法、全局变量，等等。

下一个版本将更好地使用 JRuby 字节码编译器。JRuby 项目可以将 Ruby 代码编译为 Java 字节码。这意味着 RDT 的下一版本将更容易集成到 Eclipse 环境中，提供更好的支持。

回页首

结束语

Pragmatic Programmer: From Journeyman to Master 一书中最出色的建议之一是：您应该每年学习一种新的编程语言从而跟上潮流，并且通过新语言学习一些新知识。Ruby 就快要广泛流行了，这在某种程度上是由于 Ruby on Rails 项目获得了成功。到了将 Ruby 加进您的工具库的时候了。RDT 是您掌握 Ruby 的好帮手。

参考资料

学习

- 您可以参阅本文在 developerWorks 全球站点上的 英文原文
- Ruby 语言的官方站点 Ruby-lang.org 是通向其他 Ruby 站点和文档的门户。
- Ruby on Rails 是一个有影响力的基于 Ruby 的 Web 应用程序框架。这个站点允许下载 rails 并且查看示例应用程序。
- Rubyforge.org 提供 One-Click Ruby Installer，这个程序在 Windows 上建立 Ruby 环境。
- 在 Ruby-doc.org 上可以找到最新的 Ruby 文档。
- Dave Thomas 撰写的 *Programming Ruby, The Pragmatic Programmer's Guide, 2nd Edition* 是 Ruby 语言的指南。这本书采用一种非常容易阅读的方式编写，是新接触 Ruby 的人的首选书籍。即使您不关心 Ruby，也应该阅读这本经典的软件工程著作。
- 如果您对 Rails 感兴趣，那么 Dave Thomas 和 David Heinemeier Hansson 撰写的 *Agile Development with Rails: A Pragmatic Guide* 是不错的入门书。这本书明确地指导如何用 Rails 进行开发。
- Andrew Hunt 和 David Thomas 撰写的 *The Pragmatic Programmer: From Journeyman to Master* 建议开发人员应该每年学习一种新语言。
- Glenn Vanderburg 是一位软件思想家，常常提出有意思的观点。他是最先认识到并且宣传 Java 技术的潜力的人之一，他编写了最早的高级 Java 书籍，近几年他积极支持 Ruby。
- *Bitter Java* 和 *Bitter EJB* 的作者 Bruce Tate 也大力支持 Ruby 并且可能从 Java 技术转向 Ruby。他的几本书指导开发人员如何从 Java 人变成 Ruby 人。请阅读 developerWorks 上他的 "Secrets of lightweight development success" 系列。

- Martin Fowler 是软件工程领域中最响亮的名字之一。他理解深奥的概念并且能够极其透彻地表述这些概念。几年来，他大力宣传 Ruby 的潜力，促使大家转向 Ruby。他常常在自己的 blog 上介绍 Ruby。

- developerWorks 提供了许多 Ruby 文章和教程，包括“调试 Ruby 程序技巧 101”和“用 Rake 自动执行任务”。

- 请访问 developerWorks 的 开放源码专区，这里有丰富的 how-to 信息、工具和项目更新，可以帮助您利用开放源码技术进行开发并且将其用于 IBM 产品。

获得产品和技术

- 获得 Ruby Development Tools，这个 Eclipse 插件添加了 Ruby 支持，包括 Content Assist、调试和单元测试支持。

- 可以从 Rubyforge.org 获得 Rake，这是 Ruby 的构建工具。Rake 提供一种用于执行构建的语言，提供了 Make 和 Ant 两者的优秀特性。它还展示了 Ruby 灵活的语法如何轻松地创建具有高度针对性的领域专用语言。

- 使用 IBM 试用版软件 改进您的下一个开放源码开发项目，这些软件可以下载或者通过 DVD 获得。

讨论

- 通过参与 developerWorks blogs 加入 developerWorks 社区。

关于作者

Neal Ford 是 ThoughtWorks 的应用架构师，这是一家 IT 专业服务公司，它网罗了世界各地的天才，致力于从软件中获得更多价值。他是许多应用、教学资料、杂志文章、课件、视频/DVD 演示的设计者和开发人员，还是 *Developing with Delphi: Object-Oriented Techniques*、*JBuilder 3 Unleashed* 和 *Art of Java Web Development* 等书籍的作者。他主要关注的领域是大规模企业应用程序的构建。他还是在国际上受欢迎的演讲者，曾经在世界各地的许多开发人员会议上发言。他欢迎您通过 neal.ford@gmail.com 与他联系。

对本文的评价

- ◉ 太差！ (1)
- ○ 需提高 (2)
- ○ 一般；尚可 (3)
- ○ 好文章 (4)
- ○ 真棒！ (5)

建议？

反馈意见

回页首