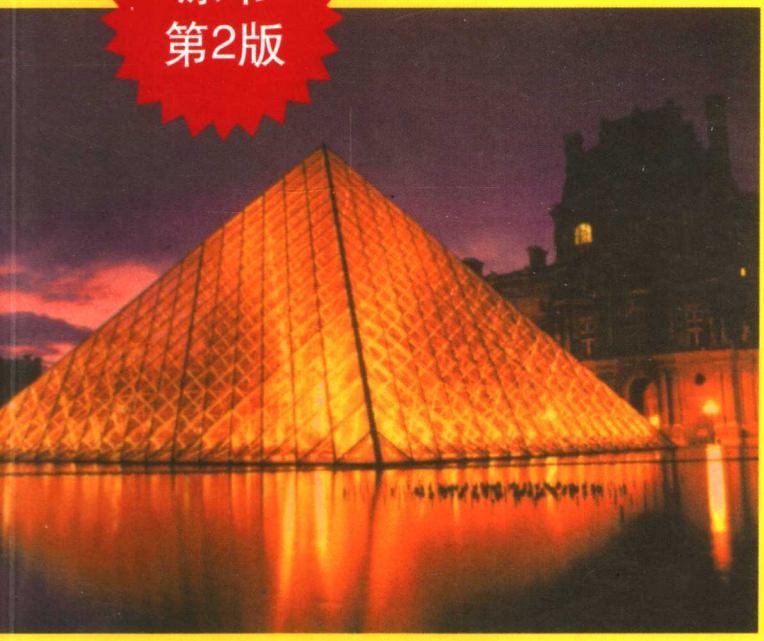
HZ BOOKS

PRENTICE HALL

Sun microsystems

Java Studio Creator 用户指南

原书第2版



（美）Gail Anderson
Paul Anderson 著

王海鹏 蔡黄辉 译

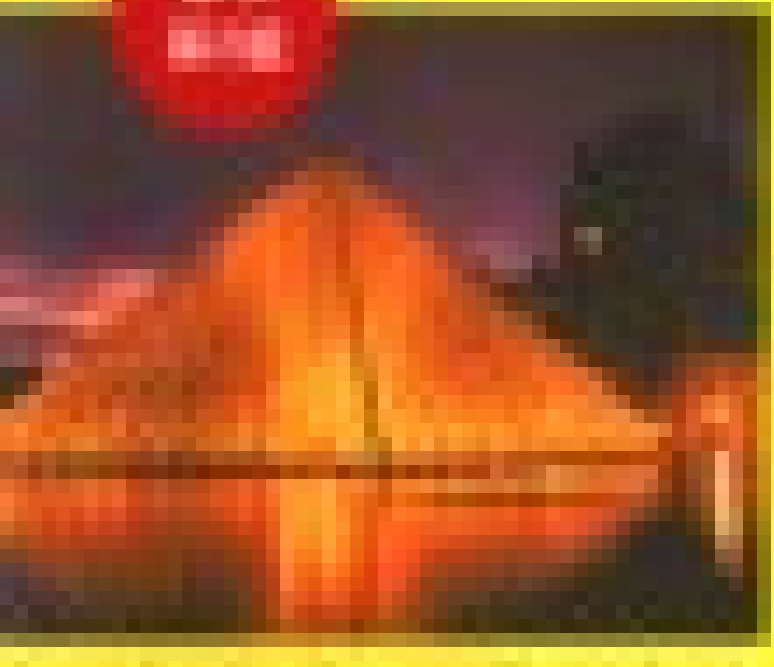
- 如何拖放组件、指定页面导航、访问Web服务和数据库。
- Creator提供了发挥IDE强大功能的建议与提示。
- 组件分类列表列出了每个组件、验证器和数据转换器，易于查找。
- 包含完整的Web服务访问案例研究，使用Google Web服务API。

Java Studio Creator Field Guide, Second Edition

Java之父James Gosling作序推荐

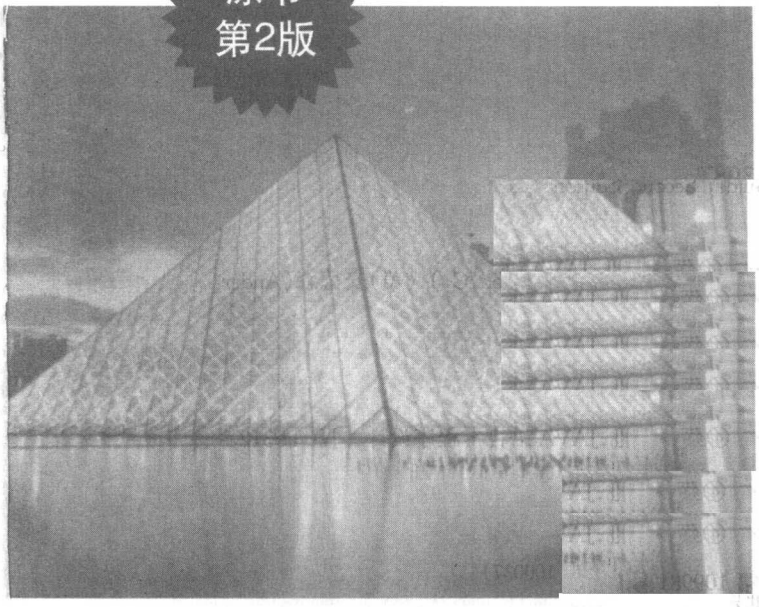
机械工业出版社
China Machine Press

Java Bundle Creator Arena



Java Studio Creator 用户指南

原书第2版



（美）Gail Anderson
Paul Anderson 著

王海鹏 蔡黄辉 译

Java Studio Creator Field Guide
Second Edition

机械工业出版社
China Machine Press

本书详细介绍 Java Studio Creator 集成开发环境的方方面面，主要内容包括：Java 技术概述、Creator 基础、Creator 组件、软件开发、页面导航、Creator 项目剖析、Web 页面设计、引入数据提供者、访问数据库、访问 Web 服务、使用 EJB 组件、Portlet、用 Creator 实现应用个性化、利用 Creator 调试等。本书讲解清晰、实例丰富，可帮助开发者极大地提高工作效率。

本书适合 Web 应用开发人员参考。

本书版权登记号：图字：01-2007-3087

图书在版编目（CIP）数据

Java Studio Creator 用户指南（原书第 2 版）/（美）安德森（Anderson, G.），（美）安德森（Anderson, P.）著；王海鹏，蔡黄辉译. －北京：机械工业出版社，2007.9

书名原文：Java Studio Creator Field Guide, Second Edition

ISBN 978－7－111－21918－7

Ⅰ.J…　Ⅱ.①安…　②安…　③王…　④蔡…　Ⅲ.JAVA 语言－程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2007）第 109981 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：李东震

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2007 年 9 月第 1 版第 1 次印刷

186mm×240mm · 27.5 印张

定价：52.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

本书详细地探讨了如何利用Sun微系统公司突破创新的Java Studio Creator集成开发环境来创建BS架构的J2EE应用程序，清晰地展示了Creator的强大功能以及如何极大地提高开发者的效率。若能掌握本书的内容，一定能够帮助您提高J2EE Web应用开发的效率和品质。

众所周知，J2EE的目标是为企业级的关键任务应用程序提供解决方案，因此，它由一组为数不少而又相当复杂的技术规范组成。这是由企业级关键应用本身的特点所决定的，正是这种特点，给开发者带来了挑战。

即使我们将范围局限在Web应用程序的表示层开发上，J2EE提供的解决方案也远不能满足开发者的要求，于是出现了各式各样的Java Web框架，如Struts、WebWorks、SpringMVC、Tapestry……如此之多的Java Web框架正好说明了一个问题：没有一种框架能够很好地解决问题。其中似乎Struts以其稳定性、清晰性而成为某种程度上的事实标准，这种情况一直持续到JavaServer Faces标准的推出。

JavaServer Faces综合了已有主要Java Web框架的设计优点，它有Tapestry那样的组件模型，有Struts那样的页面流转控制，有Spring那样的JavaBean支持。Struts的主要设计者Craig McClanahan也是JavaServer Faces 1.0(在JSR 127下开发的)联合规范的领导者，他曾在不同场合下多次表示JSF技术是Java Web框架的趋势。

JSF所缺的只剩下一个可视的组件组装RAD环境，像Delphi或VB那样的IDE，Java Studio Creator完成了这最后一块拼图。Creator2已趋于成熟，足以胜任实际的项目开发工作，McClanahan正是Creator产品团队中的一名架构师。好的东西使人一旦拥有，就别无所求。Java Studio Creator正是这样的产品，它是目前能找到的最好的可视化JSF应用开发工具。所有正在使用Struts或为选择哪一种Web框架而烦恼的开发者，请听从McClanahan的建议，赶快转到JSF和Creator上来吧！

在Sun公司的主打开发环境NetBeans 6中，所有Creator的功能都会合并进去，但这丝毫不会影响本书的价值：它仍是可视化JSF应用开发的最权威的指南。

在本书的翻译过程中，我们也受益颇多。我们在开发的项目正是使用Creator来开发Portlet。开发中遇到的许多问题，我们都在这本书中找到了答案。本书的1至6章由王海鹏翻译，7至14章由蔡黄辉翻译，全书由王海鹏统稿。

衷心地希望这本书能帮助您提高开发效率，享受技术带来的好处：有更多的时间享受生活！

王海鹏　蔡黄辉

2007年初夏于上海

序　言

决定创建 Java Studio Creator 的开发者面临一项艰巨的任务：让创建复杂的企业级应用程序变得简单。

构成 Java2 Enterprise Edition(J2EE)的一组技术是相当庞大的，好的方面是 J2EE 已经经过了实战的检验，证实了它作为大规模关键应用的基础是很出色的。许多优秀的书籍详细地介绍了 J2EE 的方方面面的特点，但所有这些导致了 J2EE 的不利的方面：它变得很难，需要花很多时间来学习和使用。为了让 J2EE 变得尽可能简单，人们做了大量的工作，但它仍然让人畏惧。

在简化基于 J2EE 的 Web 应用程序的开发过程方面，Java Studio Creator 迈出了一大步。开发者不需要处理这些琐碎的细节：Creator 会处理它们。开发者能够以一种简单而直接的方式关注他们的应用程序做什么，看起来怎样。他们通过简单的拖拉操作将数据源编织在一起。在 Creator 中开发应用程序只需要很少的 Java 或 J2EE 知识。Creator 不仅简化了过程，也加速了过程。

这本书包含了利用 Creator 生成企业级应用程序时您需要了解的所有内容。它不要求事先了解 J2EE，甚至不要求事先了解 Java，书中包含一些需要了解的背景知识。对于来自其他平台(例如 Basic!)的开发者来说，这是进入大规模、关键任务应用程序的绝佳入口。这很有意思，投身进去吧！

James Gosling

Sun Microsystems, Inc.

前　言

您将开始一次旅程，我们希望这次旅程是快乐而富有成果的。当然，任何应用程序开发工具的目标都是帮助开发者变得更有效率，让他们将时间花在创造性的任务上，让工具默默地替他们完成一些苦差事。从这一点来说，我们希望本书能够教给您关于 Creator 的方方面面的知识，这们您就能够快速地开始创建 Web 应用程序。

本书是怎样组织的

Creator2 在 Creator 第一个版本的基础上有了很大的改进，我们全面修订（并扩充）了这本书，以反映“全新的”Creator。总的来说，您会看到改进的设计时体验，先进的页面设计编辑器和组件样式编辑器。Creator2 包含一组完整的新 UI 组件（数量更多），支持改进的基于主题的外观，属性名称也很直观。数据提供器为持久层提供了一致的接口，NetBeans 为 IDE 提供了底层的支持。其他的改进还包括创建基于 Portlet 的项目、使用 EJB、使用支持 AJAX 技术的组件等。享受这一切吧！

第 1 章介绍 Java 的世界及其支持技术。Creator 是基于这些优秀的 Java 技术来完成它的工作的。利用 Java 编程语言和 XML，JavaServer Faces 组件系统，以及 NetBeans 工具的构建技术，Creator 集成了已有的标准。本章对这些主题进行了简要的介绍，让您能够了解 Creator 在整个 Java 世界中处于什么位置。我们也花了一些篇幅介绍 Java 编程语言，因为您在 Creator 中经常要使用到 Java 语言。如果您来自于其他的编程环境，我们希望您能很快适应基于 Java 的 Web 应用程序。

第 2 章介绍 Creator，目的是让您了解它的各种窗口、设计画面和编辑器。了解如何操作 IDE 目的是提高效率。您将从头开始创建第一个项目。

第 3 章是组件清单。这一章能够从 Creator 的组件、验证器和数据转换器中选择最适合应用组件。也指出了在本书的哪些项目中使用了这些组件。

第 4 章讨论使用 Java 源代码编辑器，用并发版本系统（CVS）管理代码，以及执行项目级的任务，如重构。Creator 是基于 NetBeans IDE 的，NetBeans 为这些软件管理任务提供了支持。

第 5 章介绍 Creator 中的页面导航。介绍如何在 Web 应用程序中指定页面流转，理解哪些组件适合实现页面导航。将讨论 Creator 的导航模型，并通过几个项目来展示页面导航。

第 6 章对 Creator 项目进行剖析。JavaBeans 组件为 Creator 提供了一种关键的支持技术。理解 JavaBeans 优点的开发者能够利用可复用的组件创造出稳定、强大的应用程序。这一章也讨论 JSF/Creator 页面请求生命周期，这样就可以充分利用 Creator 的生命周期回调方法。

第 7 章介绍 Web 页面设计和布局的相关特征。页面片段让应用程序能够保持一致的外观，一些布局组件帮助实现布局设计。Creator 的层叠样式表（CSS）编辑器能够创建页面和组件的样式规则。

第 8 章介绍数据提供器，它作为一个标准层，将 Web 应用的组件与持久层连接起来，如数据库、Web 服务、JavaBeans 组件或企业级 JavaBean（EJB）等。介绍了数据提供器接口，展示了在项目中使用数据提供器的一般方法。

第 9 章介绍如何在 Creator 中使用数据库。将创建一个项目进行基本的数据库操作，如读取、更

新、插入和删除。Creator 的数据感知组件使得连接数据库(通过数据提供器)变得简单而容易。

第 10 章介绍如何在 Creator 创建的应用程序中访问 Web 服务。随 Creator 一起提供了一些 Web 服务,您也可以在 IDE 中添加新的 Web 服务。在这一章中,将创建一个使用 Google Search Web 服务的应用程序。企业级 JavaBean(EJB)为分布式系统提供了一个强有力的模型。Creator 创建使用 EJB 的应用程序,对已部署的对象自动生成调用方法的代码。

第 11 章介绍如何调用 EJB 方法,如何通过标准的数据提供器操作返回的数据。利用 Creator,可以创建符合 Protlet 规范(JSR 168)的 JSF Portlet 应用程序。

第 12 章介绍 Portlet 应用程序开发,包含了使用 Web 服务、访问数据库以及 portlet Edit 模式和 Help 模式的一些例子。

第 13 章介绍如何利用 Creator 定制一个 Web 应用程序。了解如何进行应用程序的本地化和国际化。并介绍如何编写和安装定制的验证方法。您急于使用 AJAX 技术吗? 我们提供了几个例子,用到了支持 AJAX 技术的自动完成文本字段。

第 14 章展示如何在项目中使用 Creator 的调试器。了解如何设置断点、查看服务器日志文件、响应异常以及使用 NetBeans 的 HTTP 监听器。

关于示例

本书包含大量示例。我们在这些章节中提到的所有例子都包含在例子下载包中。数据库有关的一些章节包含了一个 Music 数据库的例子,也包含在了下载包中。例子代码可以在 Sun 微系统公司的 Creator2 网站上下载:

```
http://developers.sun.com/prodtech/javatools/
  jscreator/index.jsp
```

您也可以从作者的网站上下载例子代码:

```
http://www.asgteach.com
```

我们建议您访问以上两个网站,了解 Creator 的最新更新情况。

关于封面

贝聿铭先生的玻璃金字塔可以很贴切地表示 Java Studio Creator 技术。这个金字塔是巴黎卢浮宫博物馆的入口。它是真正国际化的,它由美籍华裔贝聿铭先生设计,贝先生既是工程师,也是艺术家。我们被玻璃和光的优雅简单所吸引,它拥有多面性,其结构既具有现代风格又体现古希腊风格。

Java Studio Creator 同样是建立在架构分层的概念上的。基于 JavaServer Faces 技术,Creator 利用了现有的 Java2 平台企业版(J2EE)的架构,当然也利用了 Java 及其运行环境作为其坚实的基础。与此相似,玻璃金字塔也是分层架构的一个范例,它的背景是传统的文艺复兴时期的宫殿,卢浮宫本身存放了自古以来的许多艺术珍宝,可能最著名的就是达芬奇的蒙娜莉莎。

Gail Anderson 和 Paul Anderson
Anderson Software Group, Inc.

致谢

如果没有其他人的帮助,不仅这本书在时间和准确性方面会受到影响,这本书是否能存在也是一

个问题。Sun公司的Jim Inscore宣布了这个项目，负责管理和协调技术支持。如果没有他的期望，本书的第2版将无法实现。

来自Sun公司Creator团队的许多人向我们提供了直接和间接的帮助，回答了技术问题，在设计和架构问题上提供了深刻见解，提供了更新的软件，使我们能使用最新的系统，并对本书的草稿提供了很有价值的反馈意见。我们特别要感谢Winston Prakash，他为我们提供了关键的技术支持，回答了很多的问题，他就是我们要找的人，怎样评价他在这个项目上的付出都不过分。

Octavian Tanase和Sandip Chitale也直接帮助我们，在本书的编写过程中解答了许多技术问题，Octavian也集中处理我们的问题，将它们反馈给对应的Creator团队成员。因此，我们也想感谢那些给予我们帮助的人：David Botterill、David Folk、Chau Nguyen、Tor Norbye、Matt Bohm、Craig McClanahan、Chris Kutler、Edwin Goei、Dongmei Cao、Vaughn Spurlin和Dusan Pavlica。Valerie Lipman参与了早期确定内容的讨论。我们的读者Les Hawkins也提供了有价值的反馈。

第2版是在第1版的基础之上完成的。Greg Doench是Prentice Hall的编辑，他使本书成功出版成为可能。德国Vietsbronn的Vicky Hilpert完成了德文翻译，Blanca Lazaro完成了西班牙文翻译。还要感谢家人(特别是Sara和Kellen)和朋友对我们的支持。

最后，感谢James Gosling给我们带来了Java。

目　录

第 1 章　Java 技术概述

欢迎来到 Creator 的世界！Creator 是一个帮助您构建 Web 应用的 IDE(集成开发环境)。虽然目前已经有许多 IDE 能完成这项工作，但 Creator 的独特之处在于，它建立在针对 Java 的一种分层技术之上，这种技术的核心是 Java 编程语言。Java 包括一个产生可移植的字节码的编译器和一个在各种处理器上运行这种字节码的 Java 虚拟机(JVM)。Java 是 Creator 的一个重要组成部分，因为它使您的应用可移植。

但 Java 不仅是一种编程语言，它也是一种“技术平台”，人们已经使用 Java 作为核心开发了许多大型系统，这些系统具有高度的可伸缩性，针对今天的一些大容量、分布式的计算环境提供一些服务和结构。

1.1　简介

Creator 依赖于多种技术，本章将对它们进行一些介绍。如果您是 Java 的初学者，它的许多组成部分和缩写可能使您畏缩。Java 技术被划分为一些相关的包(package)，这些包中包含了类和接口，为了构建一个应用，您需要一个系统中的某些部分以及另一个系统中的某些部分。本章为您提供了 Java 技术和文档资源的线路图，帮助您利用 Creator 来设计您的 Web 应用。

我们将从 Java 编程语言的概述开始，这有助于您习惯通过编写 Java 代码来定制自己的 Creator 应用。但在此之前，我们将展示如何找到 Java 类和方法的文档。这有助于在程序中有信心地使用它们。

大多数的 Java 应用编程接口(API)文档可以通过 Creator 的帮助系统进行访问，在主菜单的 Help 菜单下，有时候只需要包或系统的名称，就能找出类、接口或方法属于哪个 API。Java 包含了基本语言(java 下的所有包)和 Java 扩展(javax 下的所有包)。当您选定了一个包之后，就可以探索其中的接口和类，了解它们实现的方法。

也可以访问 Java 在线文档。下面是 Java API 文档的起点：

```
http://java.sun.com/docs/
```

这个页面包含了 Java2 平台标准版的一些链接，其中包含了核心的 API。它也包含了所有其他 Java API 的链接，可以在下面的页面中找到：

```
http://java.sun.com/reference/docs/index.html
```

Creator 也建立在 JavaServer Faces(JSF)技术之上。可以在下面的页面上找到当前的 JSF API 文档：

```
http://java.sun.com/j2ee/javaserverfaces/1.0/docs/api/
index.html
```

JSF 作为 J2EE Tutorial 的一部分进行了介绍，可以在下面的页面上找到：

```
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
```

这些是所有重要的参考资料。我们在本书的开始就引入它们，这样以后(当您深入到 Web 应用开发的挑战中时)就能容易地找到。现在，让我们从 Java 编程语言开始。然后我们将探讨作为 Cre-

ator 基础的一些支撑技术。

1.2 Java 编程语言

这里对 Java 编程语言的粗略概述是为来自于非 Java 编程环境的读者准备的。本书并不打算成为 Java 语言深度参考，而是作为了解 Java 的一个起点。Creator 中的许多工作涉及通过设计画面操作组件和组件的属性表。但是，有时候必须在 Java 页面 bean 中添加代码(针对 Web 应用页面的支持代码)，或在应用中使用 JavaBeans 组件。需要对 Java 有一个基本的了解才能更容易地使用 Creator。

面向对象编程

像 C 和 Basic 这样的语言是面向过程的，这意味着数据和函数是分离的。在编写程序时，您要么将数据作为参数传递给函数，或者把数据作为全局的，让函数可以访问。如果需要隐藏口令、客户的身份代码或网络地址等数据，这种安排可能会有问题。在编写简单程序时，面向过程的设计可以很好用，但它一般不适合于更复杂的任务，诸如分布式程序和 Web 应用。函数库有一定的帮助，但错误处理会很困难，而且全局变量会在程序维护时引起副作用。

面向对象编程将数据和函数组合成名为“对象”的单元。类似 Java 这样的语言会向用户程序隐藏数据(字段)，只暴露出函数(方法)作为公有的接口。这种“封装”的概念使您能够控制调用者如何访问您的对象。它使您能够将应用分解为一些行为方式较为简单的对象，这个概念称为“抽象”。在 Java 中，您用一个 Java 类来实现一个对象，对象的公有接口就成为它的“外部视图”。Java 支持继承，能从已有的数据类型进行扩展，创建新的数据类型。Java 也有接口，它允许对象实现特定类型的对象所要求的行为。所有这些概念都有助于分离对象的实现(内部视图)与它的接口(外部视图)。

从同一个类创建的所有对象都具有相同的数据类型。Java 是一种强类型的语言，所有对象都隐式地派生自“Object”类型(除了内建的 int、boolean、char、double、long 等原生类型)。您可以利用一个转换器将一种对象转化为另一类对象。只有当编译器知道这种转换时，强制转换到不同的类型才允许发生。通过动态语法分析和代码完成选择，Creator 的 Java 编辑器帮助您创建形式良好的语句。您将在第 2 章中看到它是如何工作的。

错误处理总是棘手问题，而且 Web 应用的错误处理更为困难。服务器可能出现处理错误，但需要以一种较好的方式告诉用户，Java 实现了异常处理，以将错误作为对象并优雅地进行恢复，Java 编译器强制程序员使用内建的异常处理机制。

另外，Java 不允许使用全局变量，这一限制有助于程序的维护。

创建对象

操作符“new”在 Java 中创建对象。您不需要关心对象的销毁，因为当程序不再使用这些对象时，Java 将通过一种垃圾收集机制自动销毁它们。

```
Point p = new Point();        // 在 (0, 0) 创建一个点
Point q = new Point(10, 20); // 在 (10, 20) 创建一个点
```

操作符“new”在运行时创建一个对象并将它在内存中的地址返回给调用者。在 Java 中，您使用“引用”(p 和 q)来保存对象的地址，这样将来您可以引用它们。每个引用都有一个类型(Point)，对象创建时可以带参数，初始化它们的数据。在这个例子中，我们创建了两个带 x 和 y 坐标的 Point 对象，一个具有默认坐标(0，0)，另一个坐标是(10，20)。

在您创建了一个对象之后，您可以通过引用来调用它的方法。

```
p.move(30, 30);          // 将对象 p 移动到 (30, 30)
q.up();                  // 将对象 q 在 y 方向上向上移
p.right();               // 将对象 p 在 x 方向上向右移

int xp = p.getX();       // 取得对象 p 的 x 坐标
int yp = p.getY();       // 取得对象 p 的 y 坐标
q.setX(5);               // 改变对象 q 的 x 坐标
p.setY(25);              // 改变对象 p 的 y 坐标
```

如上所示您可以对 Point 对象做许多事。可以将 Point 对象移动到一个新位置，或将它向上移或向右移，所有这些操作将影响 Point 对象的一个或两个坐标。我们还有一些取值方法和设置方法，分别返回和设置 x 和 y 坐标。

为什么值得这样做？因为 Point 对象的数据(x 和 y 坐标)是“隐藏”的。您操作 Point 对象的唯一方法就是通过它的公有方法。这使得维护 Point 对象的完整性变得更容易。

类

Java 在 API 中已经有了一个 Point 类，但为了讨论的方便，让我们来看自己定义的。下面是我们的 Point Java 类，描述了前面展示的函数。

程序清单 1-1 Point 类

```
// Point.java - Point 类
class Point {
// 字段
  private double x, y;        // x 和 y 坐标

// 构造方法
  public Point(double x, double y) { move(x, y); }
  public Point() { move(0, 0); }

// 实例方法
  public void move(double x, double y) {
    this.x = x;  this.y = y;
  }
  public void up() { y++; }
  public void down() { y--; }
  public void right() { x++; }
  public void left() { x--; }

  // 取值方法
  public double getX() { return x; }
  public double getY() { return y; }

  // 设值方法
  public void setX(double x) { this.x = x; }
  public void setY(double y) { this.y = y; }
}
```

Point 类被分成三个部分：字段、构造方法和实例方法。字段保存内部数据，构造方法初始化这些字段，实例方法由您通过对象引用调用。请注意 x 和 y 字段是私有的，这在面向对象编程中确保了数据封装，因为用户不能够直接访问这些值，但其他声明为公有的方法可以由所有客户程序访问。

Point 类有两个构造方法创建 Point 对象。第一个构造方法接受两个 double 类型参数，第二个构

造方法没有参数。请注意，两个构造方法都调用 move()方法来初始化 x 和 y 字段。move()方法利用 Java 的 this 关键字来区分方法中的局部变量名称和对象中的类字段名称。setX()和 setY()方法使用了相同的技术。[⊖]

大部分 Point 方法使用 void 作为其返回类型，这意味着该方法不返回任何东西。++ 和 -- 操作符分别让值加 1 和减 1。每个方法都有一个“识别标志(signature)”，[⊖]这是函数的参数列表的别名。请注意识别标志可能是一个空的列表。

包

文件 Point.java 保存了 Point 类的定义。在 Java 中，您必须以类名来命名文件名，这使得 Java 运行时的解释器在实例化(创建)对象时能够方便地找到类定义。如果所有的类都处于相同的目录，编译和运行 Java 程序是很容易的。

事实上，类必须放在不同的地方，所以 Java 利用“包(package)”来对相关的类进行分组，Java 中的包既是一个目录也是一个库，这意味着包的层次名称与文件在目录结构中的路径名之间存在着一对一的对应关系。唯一的包名通常采取逆序的因特网域名(com.mycompany)。Java 也提供通过类路径(class path)和 JAR(Java Archive)文件来访问包的方式。

假定您需要在名为 MyPackage.examples 的包中保存 Point 类，下面是具体做法：

```
package MyPackage.examples;
class Point {
  . . .
}
```

用点(.)分隔的包名直接对应路径名，所以 Point.java 放在 examples 目录下的 MyPackage 目录中。Java 的 import 语句使得程序中可以方便地使用类名，不用完全指定它的包名。import 语句也适用于 Java API 中的类。

```
// Another Java program
import java.util.Date;
import javax.faces.context.*;
import MyPackage.examples.Point;
```

第一个 import 语句为程序提供了 Date 类，它来自于 java.util 包。第二个 import 语句使用了通配符(*)，让程序能够使用 javax.faces.context 中的所有类。最后一个 import 让程序能使用 MyPackage.examples 中的 Point 类。

异常

我们在前面曾提到，面向过程语言的一个缺点是子程序库不能很好地处理错误。这是因为库只能检测到错误，但不能修复错误。即使库支持某些精心设计的错误处理机制，您也不能强制他人检查函数的返回值或查看全局的出错标志。出于这样或那样的原因，很难编写能够优雅地恢复错误的分布式软件。

类似 Java 这样的面向对象语言具有内建的异常处理机制，让您能够将出错的情况作为对象进行处理。当关键代码的 try 语句块中发生错误时，库方法可以抛出一个异常对象，交给 catch 处理代码进行处理。在用户代码中，这些 catch 处理代码可以调用异常对象中的方法，完成各种事情，例如显示出错信息，重试，或执行其他动作。

⊖ 如果参数名与属性名不同，就不必使用 this 引用。

⊖ signature 也有人译作签名。——译者注

异常处理机制依赖在 3 个 Java 关键字：throw、catch 和 try。下面的简单代码说明了它的工作原理：

```
class SomeClass {
  . . .
  public void doSomething(String input) {
    int number;
    try {
      number = Integer.parseInt(input);
    }
    catch (NumberFormatException e) {
      String msg = e.getMessage();
      // do something with msg
    }
    . . .
  }
}
```

假定一个调用了 doSomething()的方法需要将内存中的一个字符串(输入)转换成一个整型值(数字)。在 Java 中，调用 Interger.parseInt()将执行所需的转换，但是遇到形式不正确的字符串该怎么办？幸运的是，如果输入字符串包含无效的字符，parseInt()方法将抛出一个 NumberFormatException 异常。我们要做的就是在一个 try 语句块中调用它，并在捕捉到异常时利用 catch 处理代码来生成出错信息。

下面向您展示异常是如何抛出的。这常常被称为“**抛出点(throw point)**”。

```
class Integer {
  public static int parseInt(String input)
                      throws NumberFormatException {
    . . .
    // input string has bad chars
    throw new NumberFormatException("illegal chars");
  }
  . . .
}
```

静态的 parseInt()方法[⊖]展示了异常的两个要点。首先，方法识别标志中的 throws 子句说明 parseInt()会抛出类型为 NumberFormatException 的异常对象。这个 throws 子句让 Java 编译器强制要求错误处理。要调用 parseInt()方法，您必须将该调用放在一个 try 语句块中，或者放在具有同样 throws 子句的方法中。其次，操作符 new 调用 NumberFormatException 的构造方法创建一个异常对象。这个异常对象创建时带有一个错误字符串参数，会抛给与这类异常对象(NumberFormatException)相“匹配”的错误处理代码。[⊖]正如您所看到的，catch 处理代码调用异常对象的 getMessage()方法来得到出错信息。

为什么 Java 异常很重要？因为您在用 Creator 开发 Web 应用时，必须处理抛出的异常。幸运的是，Creator 有内建的调试器，可以帮助您监视异常。在第 14 章，我们将向您展示如何设置断点来追踪 Web 应用中的异常(参见 14.10 节)。

⊖ 在 Integer 类的内部，static 关键字意味着您不需实例化一个 Integer 对象来调用 parseInt()。您只要通过类名就能调用静态方法，不需要通过对象引用。

⊖ 这种匹配不一定是精确的。抛出的异常可以精确地与处理代码捕捉的对象匹配，也可以是通过继承得到的派生异常对象。要捕捉所有可能的异常，您可以使用超类 Exception。我们将在下面的小节中讨论继承。

继承

代码复用的概念是面向对象编程的一个主要目标。在设计一个新类的时候，您可以从一个已有的类派生。因此，继承实现了类之间的“是一种(is a)”关系。继承也使得挂接(hook)到已有的框架变得容易，这样您就可以实现新的功能。通过继承，您可以保持原有类既有的结构和行为，并对某些方面进行特别处理，从而满足您的需要。

在 Java 中，继承是通过对类进行“扩展”来实现的。当您从一个类扩展出另一个类时，“父类”的公有方法就成为“子类”公有接口的一部分。父类被称为“superclass”，子类被称为“subclass”。下面是一些例子：

```
class Pixel extends Point {
  . . .
}

class NumberFormatException extends IllegalArgumentException {
  . . .
}
```

在第一个例子中，Point 是父类，Pixel 是子类。例如，Pixel 可以“是一种”带颜色的 Point。在 Pixel 类的内部，带有取值方法和设置方法的颜色字段可以用于操作颜色。但是，Pixel 对象就是 Point 对象，所以您可以向上、向下、向左、向右移动它，而且可以得到或设置它的 x 坐标和 y 坐标。(您也可以通过 Pixel 对象的引用来调用 Point 的所有公有方法。)请注意，您不需要在 Pixel 类中编写代码来完成这些事，因为它们已经从 Point 类中继承。类似地，在 NumberFormatException 中，您可以引入新的方法，并继承 IllegalArgumentException 的功能。

关于继承的另外一点，就是您可以在子类中编写一个自己的方法，与父类中的方法拥有同样的名称和识别标志。例如，假定我们在 Point 类中添加了一个 clear()方法，将 Point 对象重置到(0, 0)。在 Point 类的派生类 Pixel 中，我们可以“覆写(override)”clear()方法[⊖]。这个新版本的方法会将 Pixel 对象移动到(0, 0)，并重置它的颜色。请注意，在 Point 类中的 clear()方法会被 Point 对象调用，但 Pixel 类中的 clear()方法会被 Pixel 对象调用。通过一个 Point 引用指向两种类型的对象，您在调用这个方法时会产生不同的行为。

重要的是理解 Java 中这种方法调用是在运行时解析的。这被称为“动态绑定(dynamic binding)”。在面向对象编程方式中，动态绑定意味着对象方法调用的解析延迟到运行程序时进行。在 Web 应用和其他类型的分布式软件中，动态绑定起到了关键的作用，决定了对象如何通过网络中的不同机制或通过多任务系统中的不同进程来调用方法。

接口

在 Java 中，一个带有识别标识而没有方法体的方法被称为“抽象(abstract)”方法。抽象方法必须在子类中覆写，抽象方法也用于定义“接口(interface)”。Java 的接口与类相似，但没有字段，只有抽象方法。接口很重要，因为它们规定了“契约(contract)”。实现一个接口的所有类都必须为接口中的方法提供代码实现。

下面是一个接口的例子：

⊖ Creator 也利用了这一特征，提供了一些方法，在 JSF 页面请求生命周期的不同时刻进行调用。您可以覆写这些方法，提供自己的代码，“挂接(hooking)”到页面请求生命周期中去。我们将在第 6 章中向您展示如何做到这一点(参见 6.4 节)。