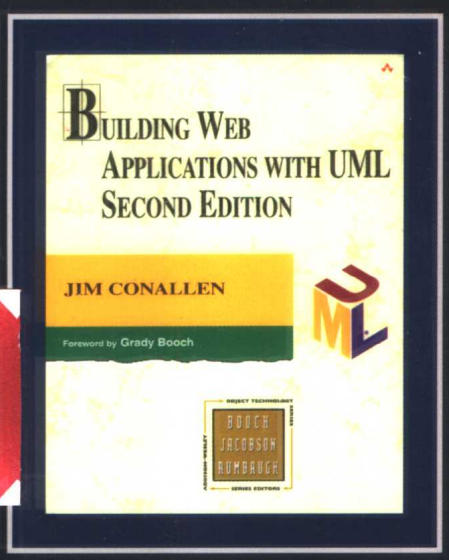
软 件 工 程 系 列

PEARSON Addison Wesley

Building Web Applications with UML Second Edition

用UML构建Web应用（第二版）

[美]Jim Conallen 著
陈起 英宇 译



- 适合Web软件项目经理、构架师、分析师、设计师
- 完整讲述UML在Web项目中的应用
- UML之父Grady Booch亲自作序

中国电力出版社
www.infopower.com.cn

软 件 工 程 系 列

Building Web Applications with UML

Second Edition

用 UML 构建 Web 应用

（第二版）

[美] Jim Conallen 著

陈起 英宇 译

中国电力出版社

Building Web Applications with UML Second Edition (ISBN 0-201-73038-3)

北京市版权局著作权合同登记号 图字：01-2003-3826 号

图书在版编目（CIP）数据

用 UML 构建 Web 应用/（美）肯那伦著；陈起，英宇译．第 2 版．—北京：中国电力出版社 2003
ISBN 7-5083-1557-X
Ⅰ.用… Ⅱ.①肯…②陈…③英… Ⅲ. 面向对象语言，UML－程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2003）第 057816 号

责任编辑：乔晶
丛 书 名：软件工程系列
书　　名：用UML构建Web应用（第二版）
编　　著：（美）Jim Conallen
翻　　译：陈起　英宇
出版发行：中国电力出版社
地址：北京市三里河路6号　邮政编码：100044
电话：（010）88515918　传真：（010）88518169
本书如有印装质量问题，我社负责退换
印　　刷：汇鑫印务有限公司
开　　本：787×1092 1/16　　**印　　张**：21.5　　**字　　数**：512 千字
书　　号：ISBN 7-5083-1557-X
版　　次：2003年11月北京第一版
印　　次：2003年11月第一次印刷
定　　价：39.00 元

序

在最近和 Java 的主要发明者 James Gosling 的一次会晤中，他谈到了开发一个复杂的软件密集型系统（Software-intensive system）所面临的挑战，并指出“当你手头拥有软件的许多片段时，绝大多数工具只把那些单独的代码行看作文本。如果不是只看到软件的一些单独的片段，而是看到整个的系统，那么常常就会很有用处。”接下来他还解释了他“为什么不像通常那样用文本的格式编辑代码，而是致力于寻找一种以可视化模型的方式编辑代码的方法。”[1]

整个软件工程的发展历史是以抽象水平的不断提高为标志的，这一点也很明显地体现在我们的编程语言、平台和方法当中。抽象是很重要的，因为这是一种人们用以把握复杂世界的基本手段。软件的图形建模仅仅是又一种更高程度的抽象而已，它使得开发人员可以可视化地研究系统、定义系统、建立系统的结构，为系统建立文档，还可以对系统进行推理研究。

以 Web 为中心的可视化建模系统确切地说是 Jim Conallen 工业做出的一个贡献。我第一次知晓 Jim 的工作是在 1998 年，那时他发表了一篇关于这一主题的早期论文。在那之前，那个最终导致 UML 的出现的统一化尝试正在进行之中，而对我们来说，能够对以 Web 为中心的系统进行建模是一个明确的目标。我们具有了对这类系统建模所必需的一些基本要素，但是 Jim 把一个解决建模问题的方法展现在了我们面前，那就是建立在 UML 扩展机制的基础之上。

这些年关注 Jim 的工作取得进展直至成熟是一件令人高兴的事情。将他的方法融入到一个商业建模工具，加上 Jim 本人对于各种有趣的以 Web 为中心的系统的涉猎，已经带领我们所有人去发现一些我们未知的事情。这一点，再加上平台（例如 J2EE）的优势，已经使得 Jim 的建模工作更加切题。

正因为如此，我很高兴有这个机会来介绍 Jim 的《Building Web Applications with UML》这本书的第二版。从第一版开始，Web 开发领域已经在某些微小的方面有所变化，主要标志是我们更深入地理解了一个声称是以 Web 为中心的体系结构看起来应该是什么样子，以及引导我们实现那个体系结构所要经历的开发过程。这两种变化都在本书的新版本中有所反映：Jim 谈到了在 J2EE 领域很多最新的发展状况，也介绍了关于 Rational Unified Process 的一些工作，这项工作主要是针对 Web 的。

在过去的几年里和 Jim 在一起工作真是一种乐趣。他是一个老练的建模者和设计师，而且更难得的是，他是一个善于表达的人，既掌握了技术的精髓，又能把他所知道的东西以一种近乎通俗的方式传递出来。

我想你会真正喜欢上本书的，当然我也会。

Rational 软件公司
首席科学家
Grady Booch

---

1 Berger，M.“JavaOne：Gosling Hits ‘Jackpot’ with Futuristic Tools.” InfoWorld，March 20，2002.

前　言

本书的第一版是在 1999 年末面市的。在很大程度上，它是基于我为零售行业和卫生部门开发基于 ASP 的应用的一些经验而写成的。我那时候已经是一个独立的顾问了，但是在完成那本书之前不久，我加入了 Rational 软件公司。在 Rational 公司工作，使得我有机会去访问一些机构并且与他们进行合作，那些机构正从事于构建以 Web 为中心的应用。我曾经见过各种各样的情况，有的是管理良好的、开发着顶尖质量的软件的开发小组，有的是管理混乱的、急于寻找正确的指导的开发小组，还有少数对这种应用持否定态度的开发者。

从第一版的公开出版到现在的两年时间里，我也已经感受到 J2EE 平台地位的提升，而且不得不说：自从那时起，我的绝大多数 Web 应用开发经验都是与 J2EE 体系结构相伴而行的。以至这个第二版中的绝大多数素材都是面向 Java 环境的。这并不意味着.NET 开发人员，甚至 Cold Fusion 和 PHP 开发人员就不能利用本书的指导来构建应用。事实上完全可以。只有在设计和实施的章节以及在参考应用里面的大多数例子，才是专门为基于 JSP 的应用而写的。

本书和前一个版本里面透露的一些思想，都是出于试图把我的面向对象的技能和 Web 应用开发领域相结合的一种期望的结果。在应用用例（use case）分析的时候，我很少遇到麻烦，直到我开始创建分析和设计模型的时候我才意识到事情开始变得艰难起来。当创建一个 Web 应用的时候，我的概念焦点一直集中在 Web 页上。而且，我关于一个模型的想法不停地围绕着一个站点地图的概念。我知道，为了理解一个应用，贯穿一个系统的导航路径将具有无可非议的重要性，而且任何系统模型将不得不包括它们。

我关于对 Web 应用进行建模的早期尝试是伴随着 Rumbaugh 的 OMT（Object Modeling Technique，对象建模技术）一起开始的，后来，当 UML（Unified Modeling Language，统一建模语言）0.8 版本开始公开发布后，我开始应用它。我知道，为了使得任何建模技术更有用，它需要捕捉的不仅要有与 Web 特征元素（例如 Web 页面和超链接）相关的语义，而且还要有它们与系统的后台元素（例如中间层对象和数据库）的关系。那时候，我发现了 OMT 和 UML 都不能够充分地表达某些东西，而在我看来，这些东西在一个 Web 应用里面是非常重要的。

作为一个在某种程度上还算成功的对象实践者和一个工程师，我突然得出了一个结论，那就是一个整套新的开发方法和理念是急需的。毕竟，如果业已存在的方法和符号不能够满足我的需要，那么最明显的解决方法就是去发明一个。当然，这是一个陷阱，这个陷阱曾使得许多像我们这样呆在软件行业的人们陷入其中。在我的闲暇时间里，我开始起草一个新的基于图形和语义的方法，去表达 Web 应用的体系结构。在为我自己的工作感到骄傲之余，我开始把它展示给我的两位同事——Joe Befumo 和 Gerald Ruldolph，他们都是经验丰富的对象实践者。他们的直接反应是：为什么？我努力去解释那些牵涉到 Web 应用开发的论点，以及那种对于在视觉上真实地表达他们的设计的需求。然而，所有听到我陈述的人都仍然认为开发一个新的方法和理念对于原来的方法和理念有点矫枉过正了。

我开始重新思考我所从事的这项工作。我并没有太多地为自己辩护，认为我是对的，而

其他人都是错的。还有更多的准备工作等着我去做。我重新审视了一下我的原始需求：为了用一个对于抽象和具体程度恰到好处的标准来表达 Web 应用的设计，而且更重要的是，这个标准要作为这个系统其余部分设计的一个部分。因为 UML 正在企业中运用得如火如荼，我意识到我做的任何工作都不能离开 UML。

于是，我重新转向了 UML。这时候它已经是 0.91 版本了，而且，一个新的概念被包括进来了——构造型（stereotype）。开始，我并不懂什么是构造型。毕竟，UML 规范并不是很轻易就能读懂的。它又长又难，但是我知道，在对 Web 应用建模的领域，任何成功必须是从这个方向开始的。最终，我开始明白了构造型和其他的扩展机制：标记值（tagged value）和约束（constraints）的含义是什么。我终于开始看到黑暗隧道尽头的一线光明。

现在我有了一个机制，利用它我可以介绍新的语义到 UML 语法里面，而不会对原有的语义有所影响。我一直明白，关键是要提供一个一致且连贯的思路，以一种正确的抽象标准，同系统的其他部分的模型一起，去对 Web 特有的元素进行建模。UML 的扩展机制提供给我这个框架去做这件事。

下一步是开始通过建立构造型、标记值和约束来定义扩展。在图里面通过构造型元素来使用自定义图标的能力大大帮助了我，使得我对于直观图的关注变得容易得多。而且，Rational Rose，即我所选择的可视化建模工具，已经介绍了在 Rose 模型里面利用你自己的构造型的一个方法。我很快创建了一套用于 Web 页面抽象的图标集。我试图让它们具有一致性，它们大都是矩形，而且在左上角带有构造型标记。我利用有填充的狗耳朵来表示页面，用没有填充的狗耳朵来代表组件。没有带狗耳朵的图标则通常代表被包含的类，它们是不能被 Web 浏览器直接请求的。Web 页面的组件的图标是三巨头曾经在他们的书《Unified Modeling Language User Guide》[1] 里面用过的图标的一个非常恰当的拷贝。

回头想想，我记得花了不到一天的时间来画这些图标。我没有对此花太多的时间，因为我一直相信最终某个具备更多经验的人会设计一些更有意义的图标。从那以后的四年时间里，那些图标基本上保持原样。然而，一个更简洁的版本出现了，它叫做“装饰”。本书的这个版本也会包括一些我如何手画这许多图标的例子，目的是为了说明在鸡尾餐巾纸上对 Web 系统建模都是可能的。（事实上，我在讨论会上对于这类事情做了相当多的建模工作和深入思考。）

随着扩展功能的发展，也随着许多细节和不连贯的部分被修正，我一直额外关注代码自动生成的能力。在我的心目中，建模技术如果可能（仅仅在理论上）清晰地生成代码并且对代码进行逆向工程编译（reverse-engineer），那么它就得到了验证。我甚至构造了一些 Rose 脚本的原型，它们的确限制了前向工程编译（forward-engineering）。从这一点开始，事情以飞快的速度进展。在奥兰多举行的 1998 年度 Rational 用户讨论会上，我公布了关于 Internet 的一份白皮书并提交了这一主题。Grady Booch 对此工作很感兴趣，并且鼓励我。Addison-Wesley 出版社问我是否有兴趣考虑把这一主题写成一本书。如果我当时就知道这本书写下去有这么难，我不敢确信我会答应。在起初的那本白皮书之后，我又为一些在线机构或者书面发行机构写了一系列其他的文章，而且开始对于扩展性进行经常性的电子邮件讨论。

自从这本书第一版公开发表后，Rational Rose 公司开始把本书中介绍的自动 Web 建模的

---

1 Grady Booch、James Rumbaugh 和 Ivar Jacobson 所著的《The Unified Modeling Language User Guide》（Addison-Wesley 出版社，1998 年出版）。

内容包括进来。在这一过程中，我开始有机会和一些顶尖的工程师——例如 Tommy Fannon 和 Simon Johnston 等一起工作，从而在 UML 具备正反向的设计功能之后应该是什么这一问题上，得到了更大的收获。因为他们的洞察力和其他很多 Rational 公司内外人士的努力，我相信这本书的新版和 Web 建模轮廓都会在当今的以 Web 为中心的构架应用方面更具健壮性和可行性。

本书面向的读者

本书目的是针对客户端/服务器系统的构架师和设计者介绍一些关于 Web 开发的要点和技术。它将有助于项目经理去理解有关于 Web 应用开发的技术和要点。因为本书建立在已经存在的面向对象（OO）的方法和技术之上，所以就没有再去介绍这些内容。我们希望读者稍微熟悉一点面向对象的原理和概念，特别是要熟悉 UML。我们同样希望读者至少熟悉一个 Web 应用的构架或环境。

对于客户端/服务器系统的构架师来说，本书可以作为一个理解 Web 应用的技术和要点的指南。在关于哪一种技术适合服务于商业需求这个问题上，系统构架师需要做出决定，这些是通过需求和用例（use case）来表达的。第 7 章定义了三种主要的客户层构架模式，它们有助于对 Web 应用的构架进行分类。通过审查这些模式和它们的优缺点，构架师能够做出决定，从而定义一个应用的技术范围。根据任何工程规则，构架师必须对系统构架中运用到的每一种技术进行折衷权衡。在对可采用的技术以及它们之间因果关系的可靠理解的基础之上，我们能够找到一个合适的组合，使之最好地符合商业需求。

对于分析者和设计者来说，本书介绍了对于 UML 的一个扩展，它适合表达 Web 应用的设计。这个扩展的关键目标在于：

- 对适当的工件（例如 Web 页面、页面之间的关系、导航路径、客户端脚本以及服务器端页面生成）进行建模。
- 在一个恰当的抽象和具体的标准上进行建模。
- 使得模型中 Web 特有的元素能够与其他系统元素进行相互作用。

分析者/设计者必须能够用 UML 模型的术语来表达系统商业逻辑的执行。其思想是要对系统的商业逻辑有一个统一的模型。在模型中，一些商业逻辑是由传统的服务器端对象和组件（例如中间件、事务处理监视器、数据库）执行的，也有一些是由 Web 元素（例如浏览器、客户端脚本）执行的。

对于项目经理来说，本书讨论了关于 Web 应用开发的一些潜在的问题和要点。本书同时对开发小组成员的责任、活动和角色等方面也给出了指导。除了讨论分析者/设计者和构架师之外，本书还讨论了开发过程中的其他角色。项目经理是对于整个项目的健康发展负有责任的人，需要对所有的角色和与这个开发过程相关的每个人的责任有一个清晰的了解。

本书包括了更多的例子和图表。根据读者的反馈，我意识到从一些构架良好的例子中，而不是从一个长长的散文式的讲述中，他们能够学得更多更快。为了使本书更完整，我还在书中提供了两个参考应用：一个是术语表应用的 J2EE 版本，它在第一版已经出现过，还有一个是电子零售应用的例子。然而，这个电子零售应用仅仅包括客户端和表示层，因为这是本书的建模工作的焦点所在。

更新原来的针对.NET 平台的基于 ASP 的术语表应用是我的初衷。然而，因为.NET 工具和环境的延迟发布，我没有办法开发一个应用，使得它能够准确地适应.NET 环境所能提供的所有功能。

本书的结构

本书分为 13 章和 5 个附录。从概念上讲，也可以分为两个主要部分。第 1 章~第 5 章实质上是对 Web 应用建模技术和概念的一个介绍。它们为本书第二部分奠定了基础。对于那些已经很熟悉 Web 应用构架的读者来说，这些章节可以忽略不读。但是，我建议大家至少对这一部分有一个粗略的浏览。

第 2 章是一个对于很基本的 Web 应用构架的介绍。这一章定义了一个术语“Web 应用”，同时定义了它的范围和焦点。本章同时定义了主要的通信机制和语言，还讨论了支持 Web 应用的技术。它们是把简单 Web 站点（Web 系统）转变为商业逻辑执行系统的基础设施。

当系统中由客户端执行一些商业逻辑的时候，绝大多数对 Web 应用的设计工作变得复杂起来。在第 3 章中讨论了允许其成为可能的技术。在这一章中讨论了普通的 Web 技术，例如 JavaScript、applet 和 ActiveX 控件。作为与客户端资源的主要对象接口，本章还介绍了 DOM（Document Object Model，文档对象模型）。

由在第 2 章和第 3 章中介绍的技术所描述的基本 Web 应用构架，能够产生非常有用的 Web 应用，而且对常见的 Internet 应用特别有用，例如零售店。对一些应用来说，这些基本成分不足以用来实现所需要的复杂功能。那些限制因素常常是 HTTP 和 HTML 自身的基础性技术。除了 HTTP 和 HTML 之外，Web 应用能够被扩展到容纳和利用其他通信和格式的技术中。在第 4 章里回顾和探讨了这些技术中最常见的技术。

第一部分的最后一章是第 5 章。不管一个应用多么地无趣，它的威胁性有多么地低，如果它在 Internet 上，那么安全性就是该考虑的因素。甚至对于 Intranet 应用来说，也应该考虑安全性。使得 Web 应用具有安全性在很多方面比起一个传统的客户端/服务器应用要更难。由于它们本身的特性，Web 服务器对于网络上的任何节点的请求都是开放的。保证一个应用是安全的诀窍在于对安全性危险自身特性的理解。不幸的是，你所能买到的产品或服务中没有一个能够保证是一个安全的应用。安全性必须在应用中设计，而且它必须在应用中不断地维持下去。新的安全性漏洞在现有软件中不断地被发现。最终，某一个会对你的应用构成威胁。在设计你的系统时牢记这一点，控制下一个将要出现的安全性危险将会更加容易一些。

本书的第二部分讲述构建 Web 应用的过程。它从第 6 章开始，回顾了开发面向对象系统的整个过程，还介绍了一个 Web 应用开发过程的例子。这个过程并不是一个完整的过程，但是它的确提供了足够的细节来确立了一个上下文环境，在这个上下文环境下面，过程中的模型和工件都能够被理解。

第 7 章讨论了定义 Web 应用构架的活动。虽然这些活动经常发生在对需求和系统用例的一个近乎于完整的检查之后，但是较早地讨论它可以帮助建立一个开发 Web 应用的思维模式。因为这里运用到的过程是迭代的和递增的，所以当定义和细化系统用例的时候，用例的说明者必须在他们的心中有一个 Web 系统的构架。从理论上说，这是不可能的。然而，实际上，在一个递增和迭代的过程里面，把用例放到一个特定构架的上下文环境里面，并不一定

是个错误。

第 8 章回顾了搜集系统需求和定义系统用例的过程。各种各样的需求都可以搜集起来，用于帮助详细说明一个特定的系统。其中搜集功能性需求最有用的技术之一是使用用例。用例提供了一种结构化的方法来搜集和表达一个系统的功能性需求，而且描述了一个系统用户（叫做参与者）和系统的交互。用例是些文本文档，它仅仅在语言的范围内描述了系统应该做什么，而没有详细说明它应该怎么做。至于应该怎么做，是在第 9 章和第 10 章讨论的。

第 9 章介绍了开发过程中的一个新模型——UX（User Experience，用户体验）模型。UX 模型描述了用户界面中对于构架具有重要意义的元素。通过建立 UX 模型，允许把外观问题和工程问题分离开来。本章中，UX 模型被描述为在用户体验小组和工程小组之间订立的一个契约。用户体验小组负责设计和构建用户界面视图，而工程小组则负责实现必要的商业逻辑和功能。之所以要在这里介绍 UX 模型，是因为在我访问过的很多组织当中，对用户体验工件（artifact）进行填充的工作都是在第一批需求集建立之后很快就进行的，也就是差不多在分析者刚刚开始对解决方案进行建模的时候。

大概是在开始调查用户体验的同时，分析小组便开始分析系统的用例和需求说明，并开始以对象的概念来表达它们。这就是第 10 章的主题。分析是把系统需求转换为能够在软件中实现的设计的活动。正如在用例中所定义的那样，这里创建了一个分析模型，这个分析模型包括了类以及用于展示系统行为的类协作。

第 11 章和第 12 章讨论了如何将分析模型转变成可以直接映射到系统组件（实际交付的模块）的东西。第 11 章对针对 UML 的 Web 应用扩展（Web Application Extension，WAE）的主体部分作了介绍。一旦设计模型完成后，它便可以被直接映射为可执行代码。

本书的最后一章，即第 13 章讨论了从 UML 模型中生成代码的问题。因为本书的这个版本提供几个参考应用，并对 WAE 代码映射（附录 A）作了详细描述，所以本章仅仅介绍一些实现 WAE 设计的例子。

附录在这个版本中占据很大篇幅。附录 A 是针对 UML 的 WAE 轮廓的一个概述，是本书中针对 UML 轮廓介绍的一个快速而又详细的参考。其他的附录是本书中一些应用的实际例子，这些例子已经用 WAE 轮廓进行过建模。附录 B 包括一个典型的在线零售店的关键的 UML 图表。这个应用一部分是用两种构架模式开发的：受控制的控制器（Controlled Controller）和主要模板（Master Template），它们各自在附录 C 和附录 D 中描述。附录 E 包括第二个参考应用——一个简单的在线术语表。这个术语表应用管理着针对软件开发小组的定义集，演示了如何应用复杂的 JavaScript 对象以及如何对这些 JavaScript 对象建模。源码可以从 www.wae-uml.org 下载到，这是我开设的用于支持本书的一个网址。

致谢

正如大多数书籍一样，你所读到的内容、词语和观点仅仅表现了所要出版的实际著作的一个片段。在 Addison-Wesley 公司工作的那些人以及为 Addison-Wesley 公司工作的那些人（尤其是 Joan Flaherty、Evelyn Pyle、Rob Mauhar、Dianne Wood、Marilyn Rash 和 Paul Bechker）都是好样的，本书的成功，他们大有功劳。他们理解技术细节以及从初稿中捕捉奇异的技术性错误的能力给我留下了深刻印象，而那些错误在很多情况下都被我忽略了。我为 Joan 的工

作感到非常骄傲，在和其他作者进行交谈时，我常常提到她的技术和耐心。Marilyn 和 Paul 也表现出了非凡的耐心和理解力，我确信这种素质已经远远超出他们的工作本身。并不是所有作者都有机会遇到如此优秀的支持团队的。

另外还要感谢那些对本版和原来版本中的素材提供技术评审的人们。特别是 Rational 软件公司的 Grant Larsen、Simon Johnston、Dave tropeano 和 Kelli Houseton 等人提供了非常细致的、富有洞察力的看法和评论，这些都极大地提高了本书的质量。

最后，我需要感谢我的妻子 Brenda，感谢她的耐心和对我的理解。写本书花费了不少时间。本版，就像第一版一样，大部分篇幅是在忙完一整天的工作之后的夜晚时间或者周末时间完成的。Brenda 在我费心写书的时候，心甘情愿地照顾小孩（Eion、Sean 和 Evan），这是一种巨大的牺牲，也是对我的一种馈赠，而这种牺牲和馈赠，仅有为人父母之后才能真正地理解。

目　录

第二部分　构建 Web 应用

第一部分

建模和与 Web 相关的技术概述

第 1 章

绪　论

1.1　本书内容

简单地说，本书是关于构建模型驱动的（model-driven）Web 应用的。它不是一本讲述如何利用特殊的工具、按部就班的方法或者新的方法论的书。它仅仅是一个向导，针对的是项目经理、构架师、分析/设计人员、Web 应用的实施者——任何想要利用面向对象技术建立一个健壮的、可扩展的、功能丰富的 Web 应用的人。这种面向对象技术是经过证明的，迄今为止，这种技术用于构建传统的客户端/服务器应用已有好几年的历史了。本书建立在面向对象的应用开发技术之上，而不是定义自己的一套东西。我之所以说是模型驱动，是因为在系统的开发工件的演化过程中，模型是基本的驱动因素。

本书中表达出来的大多数思想不是原始的，而是有充分理由的。本书中描述的许多概念和方法是在多个领域内的长期实践活动中发展演变而来的。这些面向对象的实践活动使得项目能够准时地交付，而且不超出预算，更重要的是使它们可以预测。本书描述的面向对象的原理绝大部分是基于 Grady Booch、Jim Rumbaugh 和 Ivar Jacobson 的集体工作，我们也知道他们被称作“三巨头”。

“三巨头”是统一建模语言（UML）的主要创立者。UML 是一种用于可视化地表达软件密集型系统（software-intensive system）的模型的符号。对于大多数人来说，UML 也代表了一种方法，虽然从技术的角度看这么说不够准确。UML 仅仅只是一种语言，但是像任何语言一样，表达事物总会有一定的偏差。特别之处在于 UML 是用面向对象的风格来表达系统模型和设计的，虽然 UML 被用于表达的系统类型千差万别，从普通的商业组织结构和过程，到实时的嵌入式系统设计。

本书的标题中就包含着术语 UML，因为在本书中关于构建 Web 应用的所有讨论中，它都占有核心地位。本书中的大部分初始工作都围绕 Web Application Extension for UML（WAE）进行的。在其中，UML 符号被附加的语义和约束所扩展，从而允许将 Web 相关的构架元素作为系统模型的一部分来建模。本书的一个主要的主题是：对所有系统的业务逻辑进行建模是一个很关键的问题，而不管它在系统中哪里或怎样被实施。对于 Web 应用来说，这意味着

在我们的 UML 模型中，我们需要捕捉在 Web 页面、客户端脚本和 Web 组件中的业务逻辑的执行。在拥有一个系统中所有业务逻辑的单个中心模型之后，我们可以更好地理解这个系统，并最终在该系统的未来发布中对它进行详细阐述。

本书第一版的大部分篇幅是在我加入 Rational 软件公司之前写成的。那时候，我是一个独立的顾问，专门致力于 Web 应用。当然，本书第一版中表达的想法并不被任何一个提供商提供的工具所支持。Rational Rose 过去和现在都是我建模时所选择的工具。利用它的可扩展性接口，我设法在这个工具里面得到了一些专门的图标，甚至创建了一些脚本，这些脚本可以将 ASP 组件从类图出发进行前向工程（forward engineer）。我很骄傲地说，自从那时开始，Rose 开发小组就开始把 ASP 和 JSP 组件的双向工程（round-trip engineering）功能全部包括进来了。

然而，对 WAE 规范的完全支持并没有包括在产品的当前版本中，将来也肯定不会。简单地说，有些东西不能够准确地进行双向工程。它们可能不适合自动化，但是它们对建模依旧很重要，因为建模的真实目标是便于理解和互通，而不是自动化。本书列举了所有的规范，以及一些关于哪些重要而哪些不重要的推荐意见。但是，到底你的模型中需要多少细节性的东西，最后还是要靠你自己和你的小组拿主意。

因为这是一本关于构建模型驱动的 Web 应用的书，所以它关注三个关键的主题。本章的其余部分将讨论建模、过程和 Web 构架。

1.2 建模的角色

我们采用建模来理解复杂事物。今天的软件比以往更加复杂，因此，我们对软件进行建模。我们的模型描述了我们所要建立的，我们正在建立的，和我们已经建立的系统。模型的使用贯穿了整个开发生命周期，而且还是关键性的工件。模型经常连接着不同类型的工件——从用例规范到数据库表，而且把责任和可溯性联系起来。

这些模型都是对现实的简化。模型存在于不同层次的抽象上。一个抽象层次预示着一个模型离现实有多远。高层次的抽象标准，导致模型会非常简单。[1]低层次的抽象则导致模型与建模的事物有着近似于 1:1 比例的映射关系。一个正好是 1:1 比例的映射，与其说是一个模型，不如说是某个事物从一种形态到另一种形态的转换。

模型的真正价值不在于它包含了什么，而在于在它背后隐藏着什么。例如，UML 模型中的类图没有包含每一个类操作的单独状态。这些细节隐藏在模型里面，仅仅在源代码中才可以得到。[2]通过不把所有的类操作、属性和关联显露在外的做法，类图更进一步地隐藏了很多东西。元素选择在图中是否被抑制是由其目标所决定的。每创建一个类图都是有一定原因的：用于交流或者解释某件事情。在通信或解释中不起作用的细节并不属于图表。于是，大多数图都包含一些类，这些类都只有一些相关的细节显露在外。

开发过程中包括了很多模型。每个模型都有它自己的视点（viewpoit）。实际上，模型定

1 幻灯片和你喜爱的剪辑艺术代表了一个系统所能具有的最高层次的抽象标准。

2 我认为源代码本身也是一个模型。因为变量名称和注释的可选性，它们都是增值的基本来源。