WebSphere 快速入门

瞿裕忠 张剑锋 王丛刚 陈峥

东南大学计算机科学与工程系

摘要

我们已经走进电子商务时代，本书将带您进入电子商务应用开发的世界。本书第一章介绍电子商务理念及有关技术基础，包括 Internet、Web 和 Java。第二章介绍 IBM 电子商务应用框架，包括 WebSphere。第三章给您带来 WebSphere Studio 应用开发技术。第四、五章分别介绍新一代 Web 应用的关键技术：Java Servlet 和 JSP。第六章介绍新一代 Web 应用服务器软件：WebSphere 应用服务器。第七章带给您新一代 Web 应用编程技术：WebSphere 应用编程技术。本书附录中的实验指导带您一起走过使用 WebSphere 技术开发电子商务应用的过程。
本书特别适合于信息技术类的高年级大学生和研究生，以及从事电子商务应用开发的技术人员。通过学习本书，您将了解电子商务应用开发的先进技术，并快速掌握 WebSphere 电子商务应用开发的基本技术。

第一章 电子商务基础知识

Web 正改变着我们生活的方方面面，但任何领域都没有象商务运作方式那样经历着一场快速而巨大的变革。众多商家正在准备或已经走向转变到电子商务的道路，即使用 Internet 技术转变关键的商务过程。本章介绍电子商务理念及有关技术基础，包括 Internet、Web 和 Java。

1.1 电子商务理念

从 20 世纪 70 年代以来，很多机构依靠电子数据交换（EDI、Electronic Data Interchange）实现业务处理的自动化。EDI 着重于商业伙伴之间的事务处理标准化，但是 EDI 标准缺乏灵活性和可扩展性。进入 20 世纪 90 年代，随着 Web 技术的发展，尤其是在 1995 年 Java 出现以后，许多机构开始采用 Web 应用系统来支持电子商务。电子商务（e-business）是指借助 Internet 及相关技术进行商务活动，而一个电子商务(an e-business)是这样的一个机构，它通过内部网、外部网和 Web 将关键业务系统直接连接到客户、员工、供应商和业务伙伴。一个机构要转型为一个电子商务，就要使用 Internet 技术转变关键的业务过程，如客户关系管理、电子商贸、供应链管理、企业内部管理。这个转变过程也是一个综合使用 Internet 技术、信息技术、商务技术转变业务方式的过程。

然而，电子商务不仅仅是技术更新。转变到电子商务必须对需要做什么有一个明确的蓝图，以及对实现这个蓝图有一个清晰的发展指南。电子商务周期模型正是为企业开展电子商务提供了一个模型。电子商务周期（参见图 1-1）由四个组成阶段，包括转变、构造、运行和利用。想要转变到电子商务的机构无论何时均可以从任何一个阶段开始。这也是一个重复的过程。

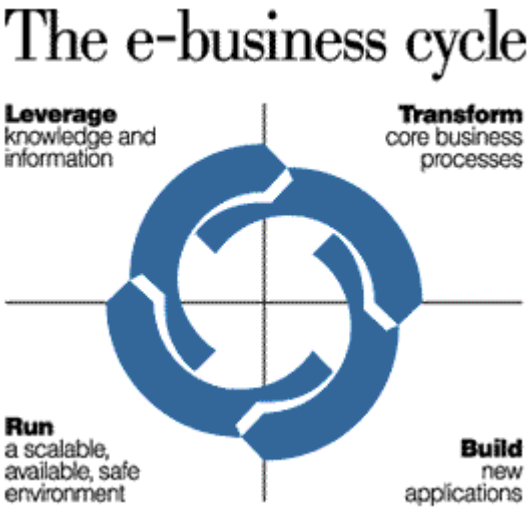


图 1-1：电子商务周期模型

1) 转变阶段是关于转变核心的商务过程，也就是要将现有的商务模型扩展到网络世界以创造一个电子商务模型。应用 Internet 技术为商务创造最大限度的价值，电子商务改变着客户关系管理、供应链和电子商贸的传统准则。在转变商务过程时，每一个商务过程应该放在整体环境中加以考虑。否则，充其量只是离散的各个更好的商务过程，无法带来期望的改善客户服务和提高电子商务价值的效果。
2) 构造阶段是关于构造新的应用系统。转变核心的商务过程需要新一代的应用系统。构造阶段也包括使用一个基于开放标准的途径将已有应用系统迁移到 Web 上。要求电子商务应用系统是基于标准的、以服务器为中心的、可伸缩的、可快速部署、易用和易管理的。
3) 运行阶段涉及一个可伸缩的、可用的、安全的运行环境。围绕着商务和应用系统通常有一个基础设施。基础设施提供的服务要求是可用的、可伸缩的、易管理的和安全的。
4) 利用阶段是关于知识和信息的利用。这里的焦点是知识管理，也就是说利用我们知道的东西。与信息管理不同，知识管理包括对显式知识和隐式知识的管理。传统的 IT 系统处理的是显式知识，即能写下来并能编程处理的。而隐式知识是人们知道的但没有被写下来的东西，它基于直觉、经验和洞察力。

从简单的开始，快速地增长。从现有的基础上构造电子商务应用系统，逐步将核心业务扩展到 Internet 上，最终实现电子商务带来的巨大的投资回报。然而，电子商务的技术基础主要包括 Internet、Web 和 Java，本章以下各节介绍这些技术的基础知识。

1.2 Internet 基础知识

Internet 已经成为企业、政府和研究机构共享信息的基础设施，同时也是开展电子商务的基础。Internet 的先驱是 ARPANET。美国国防高级研究计划局（Defense Advanced Research Project Agency、简称 DARPA）于 20 世纪 60 年代后期资助开发了一种叫做 ARPANET 的实验性通讯系统。起初，它仅用于连接美国军事机构的计算机网络，但随后不久，这个网络很快扩展到与国防有关的公司和研究机构。从此，面向特定应用需求的不同网络协议及网络技术相继出现，同时也带来了网络互连的问题。如果两个网络使用不同的协议，即使采用某种网络技术将它们在物理上互相连接起来，应用系统（如邮件系统）之间也无法相互沟通。为此，很多组织，如 CCITT（Consultative Committee on International Telephony and Telegraphy，现在成为 ITU-T，即 International Telecommunications Union - Telecommunication Standardization Sector）和 ISO(国际标准化组织)，开始考虑定义一套分层协议族，使得应用系统之间能相互通信，即使这些应用系统运行在不同的网络环境中（如不同的操作系统和不同的网络技术）。DARPA 从 1970 年左右开始研究一套称之为 TCP/IP 的分层协议族，于 1978 年左右基本定形。ARPANET 于 1980 年左右开始采用 TCP/IP 协议族，并于 1983 年全面实现。与此同时，TCP/IP 协议族在 UNIX 操作系统中

也得到了实现（由加利福尼亚大学伯克利分校完成），并免费分发。从此，TCP/IP 在大学和研究机构中迅速传播，并成为连接 UNIX 系统的标准协议。这些互相连接的网络广泛深入到大学和其它组织（一开始主要是非赢利组织）。由于个人计算机的迅速普及，该网络扩展到全球大部分地方，并且开始吸引成千上万的个人以及赢利组织加入，由此逐渐形成了所谓的 Internet（因特网）。Internet 指的是全球性互连网络。它由下列网络群构成：

1) 主干网：通常为大规模网络，这些网络主要用来与其它网络互连，如美国的NSFNET（NSF是指美国国家自然科学基金会）、欧洲的EBONE、大型的商用主干网。
2) 区域网：如连接大专院校的区域网。
3) 商用网络：为客户提供连接骨干网的服务的网络，或只供公司内部使用且连接到Internet的网络。
4) 局域网：如校园网。

90年代，Internet发展十分迅猛，这与1991年HTML的问世和Gopher的免费发放有关，也与1993年Mosaic的问世有关，Internet的商用也加速了Internet的扩张。到20世纪末，Internet已经成为一种通过服务器将小型网络连接起来的错综复杂的网络结构。大部分情况下，服务器通过专门进行 Internet 通讯的线路来传送数据。个人计算机则通过直接线路，或者通过电话线和调制解调器连接到这些服务器上。直接线路一般是高速的电讯线路，专门用于在建筑物之间或组织之间传送数据。而标准的电话线路，现在主要是综合业务数字网络（ISDN）的线路，则通常用于连接个人计算机。

1.3 Web 基本知识

Web（World Wide Web、万维网）技术是电子商务的核心技术。Web 的思想可追溯到 Tim Berners-Lee 于 1989 年 3 月在 CERN (Centre European pour la Recherche Nucleaire，或称 European Laboratory for Particle Physics、欧洲粒子物理实验室)写的一个关于信息管理的项目建议书（Information Management: A Proposal）。该建议书提出了分布式超文本系统的设想，旨在将 CERN 已有的几个信息服务器一体化，并提供一个简单的用户界面来存取各种形式的信息。1990 年 Web 浏览器和 Web 服务器使用面向对象技术相继在 CERN 实现。Berners-Lee 和他的合作伙伴成功引入了构成 Web 体系结构的基本元素：Web 服务器、Web 浏览器、浏览器与服务器之间的通信协议 HTTP(Hypertext Transfer Protocol 、超文本传输协议)、写 Web 文档的语言 HTML（Hypertext Markup Language、超文本标记语言)、以及用来标识 Web 上资源的 URL（Universal Resource Locator 、统一资源定位器)。1993 年，美国伊利诺斯大学国家超级计算应用中心 NCSA (National Center for Supercomputing Applications)的 Marc Andreesen 及其合作者发布了称为 Mosaic 的浏览器，这是第一个较健壮的易用的浏览器，它具有友善的图形用户界面。从此，Web 迅速成长为全球范围内的信息宝库。1994 年，W3 联盟在 Tim Berners-Lee 的领导下成立，该组织通过制定技术规范与提供参考软件来发展 Web 的技术标准并促进 Web 产品之间的互操作性。

URL（Universal Resource Locator 、统一资源定位器)用来唯一标识 Web 上的资源，包括 Web 页面、图象文件(如 gif 格式文件和 jpeg 格式文件)、音频文件(如 au 格式)、视频文件 (如 mpeg 格式文件)。URL 的格式为：协议://主机名<:端口号>/标识符(例如 http://www.seu.edu.cn:80/index.html)。协议可以是 HTTP、HTTPS（安全的超文本传输协议)、FTP；主机名用来标识被请求的服务器；端口通常为不同协议保留，例如 FTP 和 HTTP 守护进程侦听不同的端口，FTP 缺省的端口号为 21，HTTP 缺省的端口号为 80；标识符说明被请求的是什么，可以是文件名（含路径）或一个应用关键字 (如/cgi-bin/和/servlet/)加上一些信息（如一个脚本的名字和 servlet 的名字）。例如，用户键入 URL 格式的地址（例如 http://www.seu.edu.cn:80/index.html)；浏览器请求主机www.seu.edu.cn 在 80 端口提供的 HTTP 服务，并要求取得该服务器上的 index.html 文件；服务器接受请求，取得该文件；服务器把文件返回浏览器，并告诉浏览器这是一个 HTML 文件；浏览器在显示器上显示这个页面。在浏览器和 web 服务器之间使用的协议是 HTTP。

HTTP（Hyper Text Transfer Protocol、超文本传输协议)是用来在互连网上传输文档的协议，它是 Web 上最常用也是最重要的协议，也是 Web 服务器和 Web 客户（如浏览器）之间传输 Web 页面的基础。HTTP 是建立在 TCP/IP 之上的应用协议，但并不是面向连接的，而是一种请求/应答（Request/Response）式协议。

浏览器通常通过 HTTP 向 Web 服务器发送一个 HTTP 请求，其中包括一个方法、可能的几个头、一个体。常用的方法类型包括：GET（请求一个网页）、POST（传送一个表单中的信息）、PUT（存入这个信息、类似于 FTP 中的 PUT）和 DELETE（删除这个信息）。Web 服务器接受到 HTTP 请求之后，执行客户所请求的服务，生成一个 HTTP 应答返回给客户。HTTP 应答有一个状态行、可能的几个头、一个体。在头中可以定义返回文档的内容类型（MIME 类型）、Cache 控制、失效时间。MIME 类型包括："text/html"（HTML 文本）、"image/jpeg"（JPEG 图）、"audio/ra"（RealAudio 文件）。HTTP 本身也在不断完善和发展，目前，常用的是 HTTP1.1，它更好地利用 TCP 的特性，对 HTTP1.0 作了改进。

HTML (Hypertext Markup Language、超文本标记语言)是 Web 诞生与发展的要素之一，它旨在使得 Web 页面能显示在任何 HTML 使能的浏览器中，而与连网的机器平台无关。HTML 并不是一个程序设计语言，而是一个标记语言，它所提供的标记是由 SGML（Standard Generalized Markup Language，标准的通用标记语言）定义的。SGML 是 ISO（国际标准化组织）在 1986 年推出的一个用来创建标记语言的语言标准，它源自 IBM 早在 1969 年开发的 GML（Generalized Markup Language），该语言的名称也正好包含了三位创始人姓字的第一个字母，他们分别是 Charles F. Goldfarb, Edward Mosher，Raymond Lorie。SGML 是一种元语言，即用来定义标记语言的语言，它提供了一种将数据内容与显示分离开来的数据表示方法，使得数据独立于机器平台和处理程序。这些特性促使 Tim Berners-Lee 采用 SGML 来创建称之为 HTML 的标记语言。1993 年形成 HTML 1.0，以后不断完善，HTML 4.0 发表于 1997 年。特别需要指出的是 HTML 提供的链接机制是 Web 的本质特性之一。但是，HTML 更多的关注 Web 浏览器如何在页面上安排文本、图象和按钮等，过多地考虑外观使其缺乏对结构化数据的表示能力。另外，HTML 中有限的标记不能满足很多 Web 应用的需要，如基于 Web 的大型出版系统和新一代的电子商务，而为各种应用需要不断地往 HTML 中增加标记显然不是最终的解决方法，究其原因是 HTML 缺乏可扩展性。解决方案应该是简化 SGML 使之能应用到 Web 上。为此，从 1996 年开始，W3C（World Wide Web Consortium） 的一个工作组在 Jon Bosak 的领导下致力于设计一个超越 HTML 能力范围的新语言，这个语言后来被命名为 XML（Extensible Markup Language，可扩展标记语言）。1998 年 2 月，W3C 发布了 XML 1.0 作为其推荐标准。现在，W3C 已经用 XML 设计出一个与 HTML4.01 功能等价的语言，称为 XHTML1.0（Extensible HyperText Markup Language）。

Web 客户通常指的是 Web 浏览器，如 Netscape Navigator 和 Microsoft Internet Explorer。这种浏览器能理解多种协议，如 HTTP、HTTPS、FTP；也能理解多种文档格式，如 text、HTML、JPEG（一种图象文件格式）、XML(有的尚未支持)；也具备根据对象类型调用外部应用的功能。需要指出的是 HTML 文档中的链接在 Web 浏览器中通常以带下划线的方式显示，用户点击某个链接就能浏览到所链接的 Web 资源，这也是 Web 的魅力所在。

Web 服务器（或称 HTTP 服务器）提供 HTTP 服务。本来 Web 服务器只提供 " 静态 " 内容，即返回在 URL 里指定的文件的内容，一般具备将 URL 名映射到文件名的功能，并能实施某种安全策略。现在，可采用 CGI（通用网关接口）技术或 Java Servlet 技术从一个运行的程序里得出 " 动态 " 内容，可以采用应用关键字（如/cgi-bin/和/servlet/）来组织脚本文件和 Servlet 文件，而且现在的 Web 服务器通常还具备连接数据库的功能，这些形成了 Web 应用的出现。通常，一个 Web 服务器还提供其它服务，如 FTP 服务。有的还可作为代理服务器。一个代理服务器是一个可以从别的服务器上为它的客户取文件的服务器。代理服务器可以通过缓存应答（页面）使得响应时间更快，也可以降低网络流量，对外能隐藏内部网信息。

总之，URL、HTTP、HTML(以及 XML)、Web 服务器和 Web 浏览器是构成 Web 的五大要素。Web 的本质内涵是一个建立在 Internet 基础上的网络化超文本信息传递系统，而 Web 的外延是不断扩展的信息空间。Web 的基本技术在于对 Web 资源的标识机制（如 URL）、应用协议（如 HTTP 和 HTTPS）、数据格式（如 HTML 和 XML）。这些技术的发展日新月异，同时新的技术不断涌现，因此 Web 的发展前景不可限量。

1.4 Java 简介

现代技术的发展，尤其是网络技术，给现代企业带来了许多新的机遇和挑战，如改善客户服务、全球

化和信息检索等，这些在技术上反映为信息的获取、系统管理、系统集成、新技术的开发、Internet、Intranet等等与商业的结合。而这些要求一个随处可用的开放的结构和在不同的平台之间低成本的信息传递方式，Java 正好满足这些要求。

Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 程序设计语言（以下简称 Java 语言）和 Java 平台的总称。用 Java 实现的 HotJava 浏览器（支持 Java applet）显示了 Java 的魅力：跨平台、动感的 Web、Internet 计算。从此，Java 被广泛接受并推动了 Web 的迅速发展，常用的浏览器现在均支持 Java applet。另一方面，Java 技术也不断更新。

Java 平台由 Java 虚拟机（Java Virtual Machine）和 Java 应用编程接口（Application Programming Interface、简称 API）构成。Java 应用编程接口为 Java 应用提供了一个独立于操作系统的标准接口，可分为基本部分和扩展部分。在硬件或操作系统平台上安装一个 Java 平台之后，Java 应用程序就可运行。现在 Java 平台已经嵌入了几乎所有的操作系统。这样 Java 程序可以只编译一次，就可以在各种系统中运行。Java 应用编程接口已经从 1.1.x 版发展到 1.2 版。目前常用的 Java 平台基于 Java1.2。

Java 语言是一个支持网络计算的面向对象程序设计语言。Java 语言吸收了 Smalltalk 语言和 C++语言的优点，并增加了其它特性，如支持并发程序设计、网络通信、和多媒体数据控制等。主要特性如下：

1) Java 语言是简单的。Java 语言的语法与 C 语言和 C++语言很接近，使得大多数程序员很容易学习和使用 Java。另一方面，Java 丢弃了 C++ 中很少使用的、很难理解的、令人迷惑的那些特性，如操作符重载、多继承、自动的强制类型转换。特别地，Java 语言不使用指针，并提供了自动的废料收集，使得程序员不必为内存管理而担忧。
2) Java 语言是一个面向对象的。Java 语言提供类、接口和继承等原语，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为 implements）。Java 语言全面支持动态绑定，而 C++ 语言只对虚函数使用动态绑定。总之，Java 语言是一个纯的面向对象程序设计语言。
3) Java 语言是分布式的。Java 语言支持 Internet 应用的开发，在基本的 Java 应用编程接口中有一个网络应用编程接口(java.net)，它提供了用于网络应用编程的类库，包括 URL、URLConnection、Socket、ServerSocket 等。Java 的 RMI(远程方法激活)机制也是开发分布式应用的重要手段。
4) Java 语言是健壮的。Java 的强类型机制、异常处理、废料的自动收集等是 Java 程序健壮性的重要保证。对指针的丢弃是 Java 的明智选择。Java 的安全检查机制使得 Java 更具健壮性。
5) Java 语言是安全的。Java 通常被用在网络环境中，为此，Java 提供了一个安全机制以防恶意代码的攻击。除了 Java 语言具有的许多安全特性以外，Java 对通过网络下载的类具有一个安全防范机制（类 ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类 SecurityManager）让 Java 应用设置安全哨兵。
6) Java 语言是体系结构中立的。Java 程序（后缀为 java 的文件）在 Java 平台上被编译为体系结构中立的字节码格式（后缀为 class 的文件），然后可以在实现这个 Java 平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。
7) Java 语言是可移植的。这种可移植性来源于体系结构中立性，另外，Java 还严格规定了各个基本数据类型的长度。Java 系统本身也具有很强的可移植性，Java 编译器是用 Java 实现的，Java 的运行环境是用 ANSI C 实现的。
8) Java 语言是解释型的。如前所述，Java 程序在 Java 平台上被编译为字节码格式， 然后可以在实现这个 Java 平台的任何系统中运行。在运行时，Java 平台中的 Java 解释器对这些字节码进行解释执行，执行过程中需要的类在联接阶段被载入到运行环境中。
9) Java 是高性能的。与那些解释型的高级脚本语言相比，Java 的确是高性能的。事实上，Java 的运行速度随着 JIT(Just-In-Time)编译器技术的发展越来越接近于 C++。
10) Java 语言是多线程的。在 Java 语言中，线程是一种特殊的对象，它必须由 Thread 类或其子（孙）类来创建。通常有两种方法来创建线程：其一，使用型构为 Thread(Runnable) 的构造子将一个实现了 Runnable 接口的对象包装成一个线程，其二，从 Thread 类派生出子类并重写 run 方法，使用该子类创建的对象即为线程。值得注意的是 Thread 类已经实现了 Runnable 接口，因此，任何一个线程均有

它的 run 方法，而 run 方法中包含了线程所要运行的代码。线程的活动由一组方法来控制。 Java 语言支持多个线程的同时执行，并提供多线程之间的同步机制（关键字为 synchronized）。

11) Java 语言是动态的。Java 语言的设计目标之一是适应于动态变化的环境。Java 程序需要的类能动态地被载入到运行环境，也可以通过网络来载入所需要的类。这也有利于软件的升级。另外，Java 中的类有一个运行时刻的表示，能进行运行时刻的类型检查。

Java 语言的优良特性使得 Java 应用具有无比的健壮性和可靠性，这也减少了应用系统的维护费用。Java 对对象技术的全面支持和 Java 平台内嵌的 API 能缩短应用系统的开发时间并降低成本。Java 的编译一次，到处可运行的特性使得它能够提供一个随处可用的开放结构和在多平台之间传递信息的低成本方式。特别是 Java 企业应用编程接口（Java Enterprise APIs）为企业计算及电子商务应用系统提供了有关技术和丰富的类库。

1) JDBC（Java Database Connectivity）提供连接各种关系数据库的统一接口。
2) EJB(Enterprise JavaBeans)使得开发者方便地创建、部署和管理跨平台的基于组件的企业应用。
3) Java RMI(Java Remote Method Invocation)用来开发分布式 Java 应用程序。一个 Java 对象的方法能被远程 Java 虚拟机调用。这样，远程方法激活可以发生在对等的两端，也可以发生在客户端和服务器之间，只要双方的应用程序都是用 Java 写的。
4) Java IDL(Java Interface Definition Language) 提供与 CORBA(Common Object Request Broker Architecture)的无逢的互操作性。这使得 Java 能集成异构的商务信息资源。
5) JNDI(Java Naming and Directory Interface)提供从 Java 平台到的统一的无逢的连接。这个接口屏蔽了企业网络所使用的各种命名和目录服务。
6) JMAPI（Java Management API）为异构网络上系统、网络和服务管理的开发提供一整套丰富的对象和方法。
7) JMS(Java Message Service)提供企业消息服务，如可靠的消息队列、发布和订阅通信、以及有关推拉（Push/Pull）技术的各个方面。
8) JTS(Java transaction Service)提供存取事务处理资源的开放标准，这些事务处理资源包括事务处理应用程序、事务处理管理及监控。

在 Java 技术中，值得关注的还有 JavaBeans，它是一个开放的标准的组件体系结构，它独立于平台，但使用 Java 语言。一个 JavaBean 是一个满足 JavaBeans 规范的 Java 类，通常定义了一个现实世界的事物或概念。一个 JavaBean 的主要特征包括属性、方法和事件。通常，在一个支持 JavaBeans 规范的开发环境（如 Sun Java Studio 和 IBM VisualAge for Java）中，可以可视地操作 JavaBean，也可以使用 JavaBean 构造出新的 JavaBean。JavaBean 的优势还在于 Java 带来的可移植性。现在，EJB (Enterprise JavaBeans) 将 JavaBean 概念扩展到 Java 服务端组件体系结构，这个模型支持多层的分布式对象应用。除了 JavaBeans，典型的组件体系结构还有 DCOM 和 CORBA，关于这些组件体系结构的深入讨论超出了本书的范围。

第二章 电子商务应用框架

要转变传统的业务过程，就需要开发和部署电子商务应用系统的一个基础，电子商务应用框架正是这样的一个基础。许多企业希望电子商务应用系统具备下列特征：

1) 基于标准；
2) 以服务器为中心；
3) 可伸缩；
4) 能利用已有的核心系统；
5) 可快速部署和易用；

6) 易管理。

本章介绍的电子商务应用框架能满足企业开发电子商务应用系统的上述需求。所谓一个框架是指一个可复用的设计，表示为一组抽象的元素范例以及元素范例之间合作的接口。框架是有针对性的，如一个用户界面框架只为软件系统的用户界面提供了一个设计，而一个应用框架为整个应用系统提供了一个设计。一个应用框架中的元素范例也可称之为组件。本节介绍的电子商务应用框架基于业界标准；它为开发和部署电子商务应用系统提供了一组完整的服务；它提供的Web应用编程模型定义了Web应用拓扑结构以及使用框架提供的服务来设计Web应用的一个模型。这个框架基于独立于平台和提供商的技术标准，包括关于客户端、应用服务器、网络、数据和基础设施的标准。这些标准使得客户能在任何时候在网络上的任何地方存取有关数据和服务，也使得开发的应用软件只需写一次就能到处运行，并能即插即用各种组件。下面介绍电子商务应用框架的基本系统模型、体系结构和Web应用编程模型。

2.1 基本系统模型

电子商务应用框架为设计电子商务解决方案提供了一个模型。这个框架基于一个多层的分布式环境，在这个环境中，任何多的各层应用逻辑和商业服务分离为各种组件，这些组件通过网络相互通信。在它的最基本的形式中，可以被描述为一个“逻辑上”的 3 层计算模型，即分层是在逻辑上的，并不要求是物理上的。这个基本的 3 层系统模型包括客户、Web 应用服务器、服务器（见图 2-1）。在这 3 个逻辑层中的应用元素通过一组业界标准的协议、服务和软件连接器互相连接起来。

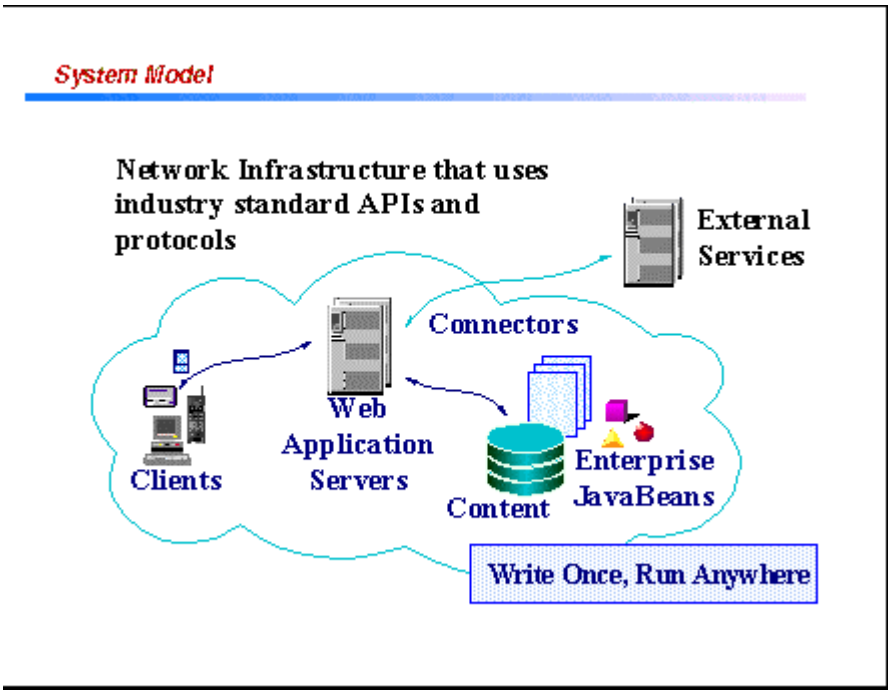


图 2-1:基本系统模型

1. 客户。这个应用框架支持广泛的客户端设备，从个人数字助理（PDA）、智能卡（smartcard）、数字无绳电话等大众普遍使用的设备到网络计算机和个人计算机。将这些客户端设备连结到 Web 应用服务器的思路是使用一组被广泛支持的基于 Internet 的技术和协议。客户端的主要作用是将应用产生的结果信息显示给用户。正因为如此，这种客户通常被称为“瘦客户”，也就是说在客户端执行的应用逻辑很少或没有，这样，很小的软件（如 Web 浏览器）需要在客户端安装。
2. Web 应用服务器。Web 应用服务器是这样的一个平台，它为应用的业务逻辑提供了一个运行环境。它包括 HTTP 服务器和企业 Java 服务，支持分布式网络环境下应用软件的快速开发和部署。应用软件在 Web 应用服务器及其内嵌的 JVM（Java Virtual Machine，Java 虚拟机）中运行。这些服务端的组件通

过 HTTP 或 IIOP（Internet Inter-ORB Protocol）与客户和其它组件通信，并利用网络基础架构提供的目录和安全服务。这些组件还可以利用数据库、事务处理、和群件等设施。

3. 连到外部服务的连接器。外部服务通常是企业在信息技术上多年投资的结果，是人们日常工作所依赖的应用和数据。这些应用和数据是重要的商务资源，需要以一种安全且可控的方式连接到 Web 上，使得企业充分发挥它们的作用为顾客、业务伙伴和员工服务。连接器就是使得它成为现实的一种机制。连接器将中间层内新增的业务逻辑连结到企业已有的应用和数据，从而将 Internet 的力量无缝地连接到企业中来。

这个电子商务基本系统模型集中体现了面向 Web 的网络计算风格，并结合了显示、业务逻辑、数据存贮这 3 层应用元素。这个基本系统模型的特性如下：

1. 基于 Web 浏览器/Java applet 使能的广泛的客户连接。
2. 易管理的客户，通过配置需要很少或无须本地的软件安装和数据备份。
3. 写一次、到处可运行的应用软件的快速开发及即时部署。
4. 提倡软件复用，使得新添程序量最小化、生产效率最大化，并提高软件质量。
5. 与外部服务的连接，在这些外部服务系统中驻留着已有业务应用和数据，充分发挥它们的作用为顾客、业务伙伴和员工服务。

2.2 体系结构

电子商务应用框架的体系结构为开发和部署电子商务应用系统提供了一组完整的服务。这个体系结构由下列关键元素构成（参见图 2-2）：

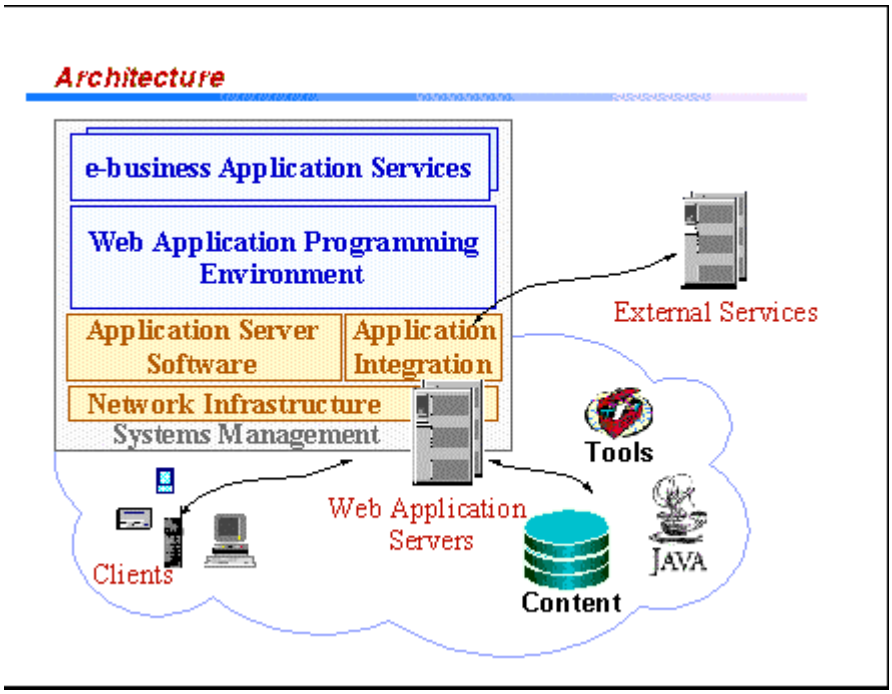


图 2-2：应用框架的体系结构

1) 客户基于 Web 浏览器/Java applet 模式，使得对应用系统的存取变得很普遍，并且应用组件能即时发送。
2) 网络基础设施提供了如 TCP/IP、目录和安全等服务，这些服务的能力可通过开放且标准的接口和协议来存取。
3) 应用服务器软件为电子商务应用系统提供了一个平台，包括 HTTP 服务器、数据库和事务处理服务、邮件和群件服务、和消息服务。

4) 应用集成使得异构应用系统间相互通信，使得 Web 能存取现有数据和应用系统。
5) Web 应用编程环境为创建动态和健壮的电子商务应用系统提供了服务端 Java 编程环境。
6) 电子商务应用服务为方便电子商务解决方案的创建提供了高层应用的特定功能。
7) 系统管理用来满足网络计算的管理需求，系统中的元素包括用户、应用、服务、基础构架、和硬件。
8) 开发工具用来创建、组装、部署、和管理应用系统。

1. 客户

客户通常为“瘦客户”，也就是说在客户端执行的应用逻辑很少或没有，这样，很小的软件（如 Web 浏览器）需要在客户端安装。在这个模型中，应用被安排在服务器上并动态地即时（on-demand ）下载到发出请求的客户。正因为如此，新一代应用的客户端部分应该用 HTML、DHTML、XML、Java applets 来实现。这种新一代应用支持广泛的客户端设备，从大众普遍使用的个人数字助理（PDA）和智能卡（smartcard）等设备到网络计算机和个人计算机。

2. 网络基础设施

它为整个体系结构提供了一个安全的可伸缩的分布式网络平台，包括下列均基于开放标准的服务：

1) TCP/IP 和网络服务。如 DHCP 和 WAP，DHCP 为进入网络的设备动态地分配 IP 地址，而 WAP（Wireless Application Protocol、无绳应用协议）将信息和电话服务发送到移动电话和其它的无绳设备。
2) 安全服务。基于公开密钥技术的安全服务支持用户辨认和鉴别、存取控制、保密、数据完整性和交易的无否认性。
3) 目录服务。基于 LDAP(Lightweight Directory Access Protocol、轻型目录存取协议)的目录服务定位网络中的用户、服务和资源。
4) 文件和打印服务。可以通过 Web 浏览器来存取和管理文件和打印服务

3. 应用服务器软件

应用服务器软件层为开发和支撑运行在 Web 应用服务器上的电子商务应用系统的商务逻辑提供了核心功能。包括下列服务：

1) HTTP 服务器。它协调、收集并组合包含静态和动态内容的 Web 页面，并将它们发送给客户。
2) 邮件和社团服务。它们提供 e-mail、工作日历、小组工作安排、聊天、新闻组讨论等服务。
3) 群件服务。它提供一个丰富的共享的虚拟工作间，并支持业务工作流程的协调。
4) 数据库服务。它将一个面向对象数据库的特性及功能与 Web 应用服务器集成起来。
5) 事务处理服务。它通过提供一个高可用的、健壮的、可扩充的、安全的事务处理应用执行环境来扩展 Web 应用服务器的功能。
6) 消息发送服务。它提供健壮的异步通信和消息代理设施来支持通信的发表/订阅模型和消息转换。

4. 应用集成

应用集成部分使得异构应用系统间在企业内或跨越企业相互通信，这些异构应用系统可能是用不同的程序设计语言实现的，也可能建立在不同的体系结构之上。企业目前的大量关键数据和应用程序（特别是事务处理程序）驻留在已有的系统中。应用集成使得 Web 客户以及服务器能与企业已有系统中的数据和程序一起工作，将 Internet 的力量无缝地连接到企业中来。所支持的集成方法包括连接器、应用消息发送服务、商务过程集成与工作流服务、组件集成服务。

5. Web 应用编程环境

Web 应用编程环境基于 Java servlets、Java Server Pages (JSP)、Enterprise Java services 和 Enterprise JavaBean 组件模型，为创建在 Web 应用服务器上的动态和健壮的商务应用提供了编程环境。该环境提供了有关服务来鼓励业务逻辑与显示的分离，使得应用能按用户兴趣和客户端设备来动态剪裁内容。

6. 电子商务应用服务

电子商务应用服务部分便利电子商务解决方案的创建。该部分包括的是面向高层应用的组件。它们建立在应用服务器软件及网络基础设施的基础上，并面向特定应用类型的功能需求按照应用框架的编程模型来实现的。比如支付服务和定单管理服务。

7. 系统管理

对于企业内部，系统管理服务为支持端到端的管理提供了核心功能，提供的有关工具和服务来支持应用系统整个生命周期的管理，从安装和配置到运作特性的监控。跨越企业，系统管理服务提供一个协作管理途径，包括策略管理和数据仓库等。

8. 开发工具

开发工具用来创建、组装、部署、和管理应用系统。

2.3 Web 应用编程模型

Web应用是这样的应用系统，它利用Web客户（如Web浏览器）、Web服务器和标准的Internet协议。通常，Web应用也能利用来自外部非Web服务的应用系统和数据。Web应用编程模型定义了Web应用拓扑结构以及使用框架提供的服务来设计Web应用的模型。

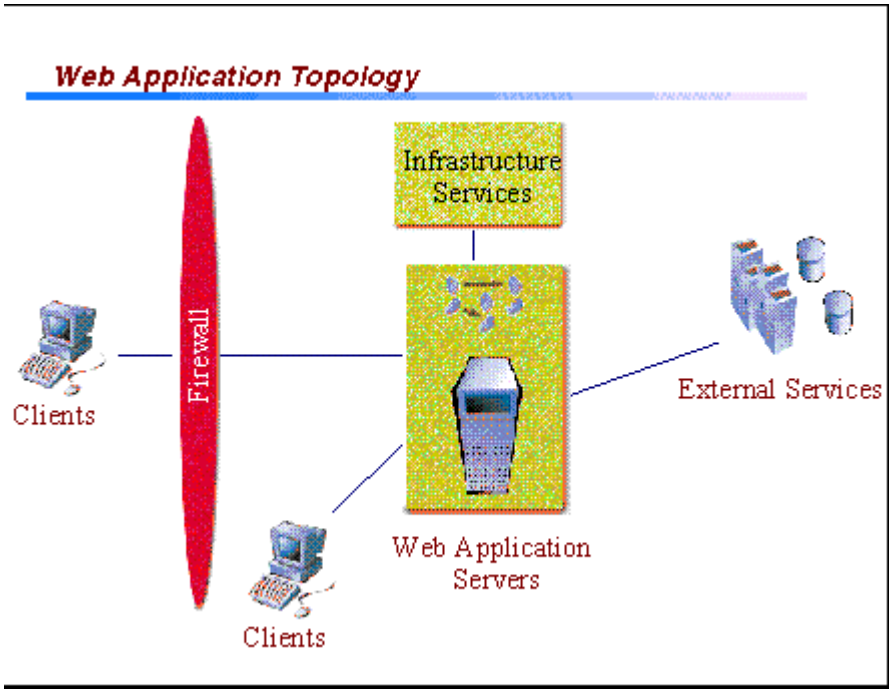


图 2-3：Web 应用的拓扑结构

图 2-3 显示了 Web 应用的拓扑结构（包括主要的元素）。值得注意的是，Web 应用服务器和外部服务是不同逻辑层，它们可能运行在同一台机器上。另外，Web 应用服务器的功能可能分散在多台机器上。通常，一个 Web 应用中的前端和业务逻辑部分运行在不同的机器上。Web 应用的拓扑结构包含的元素有客户、Web 应用服务器、基础设施服务和外部服务。

1. 客户

客户使用 Internet 技术标准(如 TCP/IP、HTTP、HTML 和 XML)与 Web 应用服务器通信来存取业务逻辑和数据。客户端的基本功能是接受并验证用户输入，显示从 Web 应用服务器到用户的返回结果。客户可以是 Internet 、Intranet（内部网）和 Extranet（外部网）中的客户。Web 应用编程模型的重要准则之一是 Web 应用的商务逻辑总是运行在服务端而不是在客户端。其优点如下：

1) 支持更广泛的客户端设备
2) Web 应用服务器能集成对资源（如数据库）的存取，从而简化应用的设计，增强可伸缩性，并提供对资源的更好保护。
3) 运行在服务端的商务逻辑容易得到保护、更新和维护。
4) 运行在服务端的商务逻辑使得用户的应用环境得到集中管理并能在不同的客户机上重建。

2. Web 应用服务器

Web 应用服务器是 Web 应用拓扑结构的核心，它为 Web 应用提供了广泛的程序设计、数据存取和应用集成等服务。我们可以把一个 Web 应用看作一个客户与 Web 站点之间一系列的交互作用。整个交互过程从显示在 Web 浏览器中一个页面开始。用户单击该页面上的一个按钮或链接就产生一个请求，该请求被送到 Web 应用服务器。Web 应用服务器对这个请求进行处理，产生新的页面，并送回到客户端。在 Web 浏览器中显示的新页面就是这一次请求的结果，可能也是下一次请求的开始。所以说，Web 应用包含了一组交互或处理步骤，每一步必须产生一个页面形式的响应，这个页面作为后继交互作用的入口。

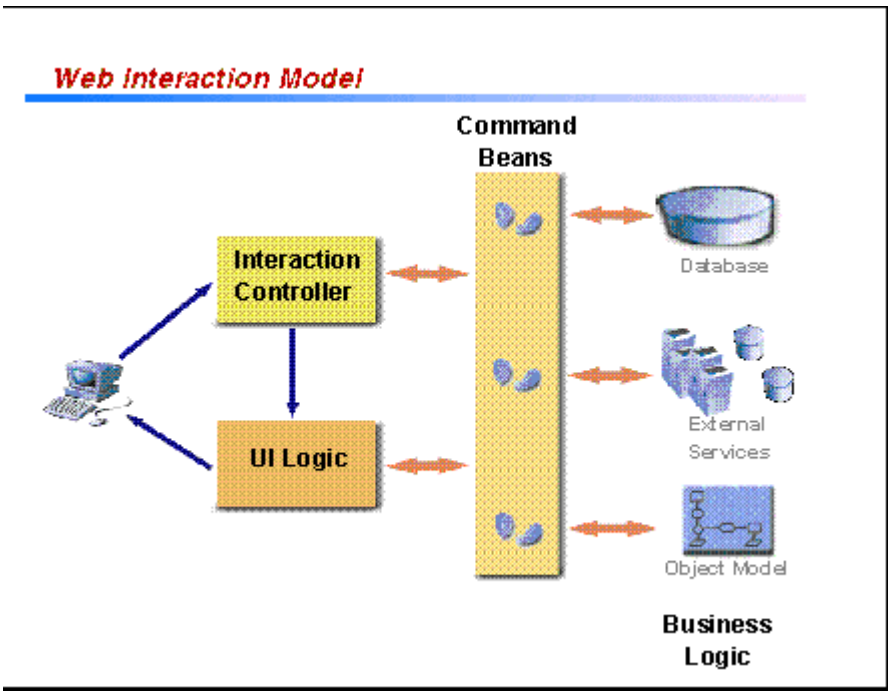


图 2-4：Web 应用的交互模型

深究单个交互的细节，不难发现这里有三个共用的处理要求，包括业务逻辑层、用户界面层和交互控制器层（参见图 2-4）。具体来讲：

1) 业务逻辑层。记录并处理用户输入的部分。比如：对应于在购物篮中添加一项商品的数据库更新操作，或从一个银行帐户到另一个的资金转帐操作。
2) 用户界面层。构造 HTML 页面的部分。构造出来的页面将被送回给用户，它决定了交互结果的显示形式和风格。
3) 交互控制器层。控制其它元素的部分。处理 HTTP 请求，从业务逻辑层选择要运行的组件，根据业务逻辑的运行结果从用户界面层选择相应的组件，以构造响应页面。

这三层正好对应到经典的 M/V/C（Model/View/Controller，模型/视图/控制器）范例。这一点很重要，因为不同层通常需要不同的开发技术和工具。应用框架为各层提供使用各种组件的支持，这些组件之间有一个良定的接口。

3. 基础设施服务

除了上述程序设计机制，应用开发者还需要运行时刻的服务来支撑 Web 应用。方便地定位应用组件、保证它们的可用性以及对它们安全的存取和执行等等都是在 Internet、Intranet 和 Extranet 环境中实施 Web 应用的关键要素。基础设施服务为支持 Web 应用提供下列设施：

1) 目录服务，本框架支持使用 JNDI 来存取基于 LDAP 协议的安全目录服务。
2) 认证授权，这个设施能产生用来鉴别用户和服务器的证书。公开密钥技术已经成为电子商务首选的可信赖的机制，而认证授权是公开密钥技术中的一个关键部分，它提供了数据保密、数据完整性、签名验证和用户鉴别等特性。
3) 防火墙，它作为可信赖的内部网络和不可信赖的外部网络之间的一个接口。通常，防火墙使用报文过滤器根据报文的源地址、宿地址、和服务类别（即端口号）来过滤报文流。使用防火墙来控制外部网络中的那些（IP 级）能允许存取内部网络、能使用那种类别的应用服务；同样的模式能控制内部网络对外部网络的存取。
4) 代理服务器，它起到为多个浏览器检索 Internet 上数据的作用。作为客户和 Web 应用服务器之间的一个管道，代理服务器将客户的请求递交给有关的服务器并把服务器的响应返回给客户。这种工作方式还可以提供附加功能，如地址安全性和缓冲功能。

4. 外部服务

在这些外部服务系统中驻留着企业已有的应用和数据、以及商业伙伴的服务系统，如支付服务和外部信息服务。通常，这些应用和服务系统控制着关键的商务过程，需要与 Web 服务器集成起来为顾客、业务伙伴和员工服务。

使用电子商务应用框架开发和部署电子商务应用系统带来下列优势：

1) 极大地方便电子商务应用系统的快速开发和部署。
2) 适应任何客户端设备。
3) 确保应用系统能移植到各种不同的服务器环境中。
4) 利用并扩展已有的信息技术资源。

这也是形成这个电子商务应用框架的重要原则。

2.4 WebSphere

电子商务的灵魂是软件，而电子商务应用系统通常采用以 Web 服务器为中心的计算模式（参见基本系统模型），因此，Web 应用服务器软件是电子商务市场的焦点。IBM WebSphere Application Server（WebSphere 应用服务器）是众多 Web 应用服务器软件中的杰出代表。

事实上，IBM WebSphere 产品系列是一套典型的电子商务应用开发工具及运行环境。该产品系列包括：

1) NetObject Fusion 提供许多构建和组织站点的工具，可用来建立和管理整个 Web 站点；
2) NetObject BeanBuilder 是一个构建 applets 的可视化写作工具；
3) NetObjects ScriptBuilder 提供了 Smart HTML、JavaScript、JSP 等的编辑器；
4) Lotus Domino Go Webserver 是一个 Servlet 使能的 Web 服务器；
5) IBM WebSphere Studio 提供了项目工作台和有关向导，这些向导可用来存取数据库、创建 JavaBean 和 servlet 等等；
6) IBM WebSphere Application Server（WebSphere 应用服务器）是一个 Web 应用服务器（内含 IBM Http Server），它本质上是适合于 servlet 的 Web 服务器插件，提供了增强的 Servlet API 和 Servlets 管理工具，并集成了 JSP 技术和数据库连接技术。

第三章 WebSphere Studio 应用开发

本章结合 IBM WebSphere Studio 产品的使用，介绍电子商务应用开发技术。

3.1 Lotus Domino Go Webserver

Lotus Domino Go Webserver 是一个 Web 服务器软件。它具有下列特性：

- 支持 CGI、Servlet 和一些其它的应用编程接口。这些应用编程接口用来创建动态页面。
- 具备代理服务器（Proxy Server）的功能。一个代理服务器是一个请求代理（Broker）。浏览器把请求发到代理服务器，代理服务器从网上或从缓存里取这个页。
- 支持 SSL 3.0。SSL 用来确保在 web 服务器和 web 浏览器之间安全通信。
- 支持 PICS（Platform for Internet Content Selection）。PICS 使得用户可以过滤他们遇到的材料，按照材料的分级（由 Web 管理员设置）接受或者拒绝材料。
- 集成了搜索引擎。集成的搜索引擎是 NetQuestion.
- 易维护性。如很容易地维护用户和访问控制列表

下面就 Domino Go Webserver 的特性及有关使用技术加以详述，包括 Web 服务器的配置、启动 Web 服务器、URL 与目录映射规则、服务器日志、访问控制、代理服务器。

1．Web 服务器的配置

配置 Lotus Domino Go Webserver 有两种方式，一种是使用 Web 浏览器来配置和管理，另一种是直接编辑配置文件。第一种方法更简单。安装 Lotus Domino Go Webserver 之后，启动 Web 服务器，使用 web 浏览器打开http://127.0.0.1/frntpage.html，输入用户名和口令后，浏览器就显示配置和管理 Web 服务器的表单。.这些表单是 CGI 程序和 HTML 表格的结合，为配置 Web 服务器或查看 Web 服务器的当前配置提供一个简单的方法。浏览器可以和服务器在同一台机器上或在访问 Web 服务器的远程的客户机上。在远程的客户机上进行配置和管理时需要访问http://your.server.name/，当然要求输入正确的用户名和口令。另一种方法是直接编辑配置文件。缺省地，配置文件名为 httpd.cnf，在系统目录下，通常是 C:\WINNT。配置文件是由指示语句（directive）构成的。通过更新这些 directives 来设置 Web 服务器的配置项，如主机名（Hostname）和 URL 传递规则（Passing Rules）。

2．启动 Widows NT 上的 Web 服务器

有两种方式来启动 Widows NT 上的 Web 服务器。第一种是从服务面板启动(NT Service Panel->StartUp Panel)，有 3 个可选的启动类型（Auto、Manual、Disable），并提供了从系统帐号启动的方式，当选择从系统帐号启动 Web 服务器时，还可以选择在桌面上启动图形用户界面来显示日志或跟踪信息。另一种是用命令行启动，命令的语法为：whttpg [-p port] [-r config_file]。其中，端口号（port）指定 Web 服务器侦听的端口，而配置文件名（config_file）指定 Web 服务器的配置文件，通常不是缺省的配置文件（在 WINNT 目录下的 httpd.cnf 文件）。

需要指出的是在一个 Widows NT 上可以运行多个 Web 服务器，但是有两个注意事项。第一，同时运行的 Web 服务器必须使用不同的端口。第二，最多有一个 Web 服务器可以从服务面板启动，也就是说，附加的 Web 服务器需要用命令行启动，并使用不同的端口，但可以使用同一个配置文件（建议使用不同的配置文件为妥）。另外，一个 Web 服务器可以有多个 IP 地址，每个 IP 地址有不同的主机名，这称为多 IP 服务器。一个 Web 服务器也可以只有一个 IP 地址，但是有多个主机名，这称为虚拟主机。多 IP 和虚拟主机可以为不同的虚拟主机提供不同的信息，尽管不同的虚拟主机是由同一个 Web 服务器服务的。

3．URL 与目录映射规则

URL（Uniform Resource Locator、统一资源定位器）是指定 Web 上资源项的地址的约定规则。它包括协议、后面跟着完整的主机名、和请求。一个 URL 的语法时：

协议：//主机名[：端口][/[目录名[/...][/文件名[？变量名= 值[&...]]]]]。

协议指的是提供 http、https、ftp 等服务的有关协议，Go WebServer 服务器提供自己的 http 和 https 服务，

并代理 ftp、gopher、wais 等服务。通常，Web 服务器把 URL 的主机名和端口后面的部分映射到一个路径和文件名，即所谓的目录映射。Go WebServer 的 Web 文档根目录缺省为 c:\WWW\HTML，在安装该 Web 服务器软件时可以指定 Web 文档根目录。当服务器收到的 URL 请求没有规定文件名字时，Web 服务器会在相应的目录里搜索某个文件名列表，该文件名列表称为欢迎页面(Welcome Pages)列表。欢迎页面列表通常包括 Welcome.html 和 index.htm 等。关于目录映射，Go WebServer 使用 URL 传递规则来实现所需的目录映射。传递模板及示例如下：

传递模板	动作	请求模板	替代的文件路径
示例	Pass	/test/*	c:\myhome\test*

其中，请求模板代表被请求的 URL，可以包含（*）通配符。当一个 URL 匹配请求模板时，Web 服务器按照替代的文件路径访问系统资源。这些规则可以用来维护物理文件系统的独立性。而动作类型及其含义如下:

(1) Map：Web 服务器按照替代的文件路径改变 URL，继续比较。

(2) Pass：Web 服务器接受请求。

(3) Fail：Web 服务器拒绝请求。

(4) Exec：Web 服务器接受请求，并按照替代的文件路径运行一个 CGI 程序。

(5) Redirect：Web 服务器使用一个由替代的文件路径确定的 URL 来响应。

(6) NameTrans：Web 服务器接受请求，在处理请求的名字转换步骤中，运行一个由替代的文件路径确定的 API 应用。

(7) Service：Web 服务器接受请求，在处理请求的服务步骤中，运行一个由替代的文件路径确定的 API 应用。

如果设置了下列传递规则：

- Map /beta/* /text/*
- Exec /cgi-bin/* c:\www\cgi-bin*
- Pass /* c:\www\html*
- Pass /abc.html c:\abc.html

URL 请求与实际的系统资源之间的对应举例如下：

表 3-1：URL 到系统资源之间的对应举例

URL 请求	对应的系统资源	应用的规则
/cgi-bin/search.exe	c:\www\cgi-bin\serch.exe	规则 1）、2）
/index.html	c:\www\html\index.html	规则 3）
/beta/download.html	c:\www\html\test\download.html	规则 1）、3）
/abc.html	c:\www\html\abc.html	规则 3）

在匹配了 Map 规则以后，服务器会继续比较余下的规则。服务器会在 Exec 和 Pass 规则后停止比较。第 4 个规则永远不会被使用，因为任何可以匹配第 4 个规则的都会匹配第 3 个规则。因此，Pass /* 规则通常应该放在最后。

4．服务器日志

缺省地 web 服务器会在 c:\www\logs 目录下记录所有东西，Web 服务器每天在午夜启动一个新的日志文件（如果那时它在运行的话）。否则，Web 服务器会在某天首次启动时开始一个新的日志文件。日志文件通常包括下列类型：

(1) httpd-log：记录 Web 被访问的统计情况。

(2) agent-log：记录客户使用的 Web 浏览器的情况。

(3) referer-log：记录引用到请求页面的页面的情况。

(4) httpd-error：记录 Web 服务器内部错误的情况

(5) cgi-error：记录 CGI 错误的情况。

日志的删除可以选择在若个天后删除，或在若个 MB 存储空间被用完后删除，或运行“ User Exit ” 删除。也可以通过配置:对某些请求不作记录。

5．访问控制

Web 服务器中的资源通常需要各种不同程度的保护。可以在 Web 服务器的配置文件里进行保护设置，也可以在每一个目录里设置 ACL（Access Control List），ACL 是一个包含保护规则列表的文件，用来保护同一个目录的文件。保护的类型有用户名保护和地址保护等方式，用户名保护决定谁可以访问这个资源（可以要求口令验证），地址保护使用地址模板来决定哪些客户机（IP 地址）可以访问这个资源。

6．代理服务器

代理服务器是一个可以为它的客户从别的服务器那里取得文件的服务器。一个代理可以降低网络流量，对 Internet 隐藏内部网信息。可以代理 http、ftp、gopher、wais 和 SSL 隧道（如 https 和 snews），通常通过设置 URL 传递规则来实现。为启用 SSL 代理，还必须启用 connect 方法。要使用 Web 服务器提供的代理功能，浏览器必须设置该 Web 服务器为代理服务器并指定这个 Web 服务器的端口。

启用的代理缓存功能可以提高运行效率。代理缓存是在代理服务器端保存 http、https 或 ftp 等缓存文件以减少不必要的重复的请求处理。有几个设置参数需要注意：缓存文件的一个给定的存储空间称为缓存容量，存放缓存文件的目录称为缓存文件的根目录，对某些 URL 进行缓存（其它的不需要）称为缓存过滤。另外，未被使用的缓存文件的时间限制用来规定服务器保存未被使用的缓存文件的最长时间，而存储空间回收用来移走过期的缓存文件。

3.2 IBM WebSphere Studio

Web 站点的内容是由静态和动态内容组成的。静态内容以 HTML、图片等形式出现。除了在客户端的动态效果以外，在一个 Web 服务器上动态内容通常以服务器端程序的形式存在，它们在被请求时产生动态内容。许多站点已经使用 applet 来达到在 web 上的动态效果。虽然这个方法能做到这一点，但是 java 和网络性能的问题阻碍了这种方法的大规模应用。服务器端生成动态内容在大多数情况下是更好的方法。因为：

- 更容易支持瘦客户。
- 提供更大的灵活性和安全性。
- 在大多数情况下动态内容来自数据库。
- 服务器端程序可以执行数据库操作，然后向浏览器报告结果。

服务器端生成动态内容符合瘦客户或超瘦客户设计模式，它使得软件开发者在一个 web 应用的客户端放尽量少的商务或数据访问逻辑。而胖客户是指在一个应用的客户端放置了太多商务逻辑和访问。这会影响网页的下载时间，可能引起其它问题。服务器端数据库访问是一个好的解决方案，可以更好地利用服务器，从客户减轻客户端的负担。通过 Java 访问数据库是利用 JDBC(Java Database Connectivity)应用编程接口进行的。总之，在为一个网站提供动态内容时，会涉及到服务器端应用程序。CGI 和/或 Servlets 将是典型的部署载体。Servlet 是生成动态内容的焦点。然而，大多数动态内容会涉及到数据库。

WebSphere Studio 提供了项目工作台和有关向导。项目工作台用来把各种生成的组件集成进一个项目，而各种向导可用来存取数据库、创建 JavaBean 和 servlet 等等。

1．项目工作台

项目工作台是一个在资源控制管理下的 Web 站点对象管理工具，其站点管理能力与 NetObject Fusion 大致雷同。项目工作台具有下列特性：

(1) 同任何网站开发工具互操作。通过将站点对象的文件类型注册到相关的应用，项目工作台可用来启动应用（工具）来操作（打开，显示）站点对象。从而能有效的管理那些与站点有关的对象资源，与资源控制环境的交互使得 Web 站点的小组开发得以有效地进行。因此它是能集成很多网站开发工具的理想的工具集成环境。

(2) 提供灵活的站点发布能力。项目工作台为两类对象资源提供各自独立的发布路径，一类是

Servlets 和 JavaBeans，另一类是所有其它的东西。项目工作台提供了网站的本地发布或远程发布两种方式。

WebSphere Studio 有一个双框图形用户界面，还有许多与桌面应用程序相似的功能特性。以下分别介绍框架、分解视图、具栏和按钮、过滤器。

(1) 框架

左边的框架显示所选项目及其文件夹的图形分层。右边的框架显示在左边框架中选择的一些项目的细节。

- 一个项目可包含文件和文件夹。
- 文件夹可包含文件或其它文件夹。
- 文件包含数据。

您可以按喜欢的方法来组织文件，并使它容易地将一个项目中的有关文件作为一个单元进行管理，比如，公布一个项目中的若干个文件或所有文件至目标服务器上（如 WebSphere 应用服务器）。

(2) 分解视图

当您从视图菜单中选择分解视图表，文件夹及子文件夹中的所有文件将显示在详细视图表中。表格包括附加的路径信息以帮助您区分它们。文件夹本身不显示。 快捷方式按钮位于工作台的右下方，您可对分解视图的开或关进行切换。

(3) 工具栏和按钮

此界面包括菜单栏、工具栏、状态栏和过滤器按钮的垂直行。

- 菜单栏允许您访问所有的可用功能。
- 工具栏允许您快速访问大多数的普通功能和向导。
- 过滤器通过文件扩展名或文件类型来控制显示文件。
- 状态栏列出一般信息，例如您的视图的大小和对象的数目，也包括被过滤器隐藏的文件数。

(4) 过滤器

过滤器按钮位于主窗口的右边。它通过文件类型和文件扩展名来控制哪些文件显示在详细框架中。单击它可进行开和关的切换。打开过滤器，请单击过滤器按钮，在详细视图中将显示此种文件类型的文件。过滤器类别是：HTML 文件、图像文件、SQL 文件、Java 文件、Java 类文件、文本文件。同时要打开多个过滤器，请在单击每个过滤器按钮时按下 Ctrl 键。要暂时关闭所有过滤器，请先从视图菜单中选择“ 过滤器 ”，然后选择 “ 显示全部 ”。

在分解视图上使用过滤器能够管理在分组中选择文件类型的所有文件，而不考虑它们的相对位置。例如，若要打开一个包含文件夹及子文件夹的项目，但其中只有 HTML 文件， 您就可打开分解视图和 HTML 文件过滤器来显示整个项目中的所有 HTML 文件。

2．向导

有关向导及其基本功能如下：

(1) SQL 向导：创建一个可被数据库访问 Servlet 使用的 SQL 语句。语句可查询关系数据库，并从表中选择数据显示在 Web 页面上。SQL 向导要求连接数据库，以验证连接参数，提取元数据来支持 SQL 语句的开发。

(2) 数据库访问向导 :创建 Servlet 来访问关系数据库。使用一个生成的 SQL 文件，数据库访问 sevlet 向导生成 5 个文件：servlet、servletBean、输入 HTML 表格、输出页面（JSP）、和可选的错误页面（JSP）。其中，可选的输入表格（HTML）用来为 SQL 命令传参数，ServletBean 提供一个可执行方法来调用 SQL 命令，Servlet 创建那个 ServletBean 的一个实例，执行它的方法，然后决定调用有关结果页面。

(3) 基本 Servlet 向导：创建使用指定的 JavaBean（包括 Navigator 和为 VisualAge for Java Enterprise Access Builder 建立 VisualAge 时的命令）的 Servlet。它同时产生一个调用 Servlet 及捕捉用户输入的输入页面和显示结果的输出页面。

(4) 到期 HTML Servlet 向导：创建随日期而更改内容的 Servlet。则它生成 Servlet 以及包括 HTML 的变量部分的页面。

(5) 注册 Servlet 向导：创建一个可在 WebSphere 应用服务器 的 UserProfile 特性中设置或获得数据的 Servlet。这使可维护 Web 访问器的详细信息，并且此访问器对于用户注册和其它的 Web 应用程序都是一样的。 向导同时产生一个调用 Servlet 及捕捉用户输入的输入页面和显示结果的输出页面。

(6) JavaBean 向导：创建能够进行编辑和扩展的 JavaBean 的基本实现。 指定在 Bean 中所需的属性，向导将创建方法来设置和获得其值。 一旦完成 Bean，就可在基本向导中使用它。

上述向导对快速地制作用于创建和显示动态内容的中间层（和客户层）元素的原型有用，对提供模板代码来定制有用。而且与 WebSphere 应用服务器的“ 编程模型 ”紧密集成。下面以使用数据库访问向导为例来创建访问数据库的 Servlet。数据库访问向导帮助创建 Java Servlet，它用来从关系数据库中检索指定数据并在 HTML 页面中显示信息。

向导需要 SQL 语句文件，Studio SQL 向导将帮助创建它。该 SQL 语句告诉数据库访问 Servlet 连接哪个数据库、搜索哪个表格、选择哪个数据以及如何排序。实际上，无需知道 Java 或 SQL 语法，就可创建数据库访问 Servlet。浏览至想放置 .jsp 文件的文件夹。

从“ 工具 ”菜单，选择“ Studio 向导 ”。

(1) 选择“ 数据库访问 Servlet”。

(2) 单击“ 下一步 ”启动向导，继续标记的页面。

(3) 当完成所有要求后单击“ 完成 ”。

(4) 当完成后，可“ 照原样 ”使用文件或根据需要进行定制。

向导将创建：

(1) 调用 Servlet 的输入页面 (.html)

(2) 提供实现指定 SQL 语句数据库访问逻辑的 JavaBean 文件 (.class 和 .java)

(3) 配置 Servlet 的 Servlet 配置文件 (.servlet)

(4) 包含选中数据库列 JSP 和 HTML 的输出标记的 JSP 文件 (.jsp)

(5) 按照下列步骤完成数据库访问 Servlet 向导：

注意：可使用这样的方式建立和测试数据库 Servlet，使用某一现有数据库，通过修改 .servlet 配置文件中的一个或多个以下值，将其创建为另一个 Servlet：

- URL
- driver
- userID
- password

此方法可能需要数据库服务器的模式别名。请参阅 Servlet 配置文件，了解更多关于 .servlet 文件的详细信息。

3．脚本编辑

IBM WebSphere Studio 能与 NetObjects ScriptBuilder 很好的集成起来。从 WebSphere Studio 的项目工作台项可以启动 ScriptBuilder 来对有关文件进行编辑，可编辑的文件类型包括 HTML、JavaScript、Java、JSP 和 XML 等。

NetObjects ScriptBuilder 是适合于 Web 的脚本开发工具，为 Web 站点的客户端和服务器端脚本提供了一个快速的开发方法。它将强大的脚本编辑器与丰富的可视化开发工具有机结合，以加速脚本的开发。它也是架构 Netscape,、Microsoft、 IBM、Sun Microsystems 等各大公司的不同 Web 脚本环境的桥梁。它支持 CFML (Cold Fusion Markup Language)、Dynamic HTML、 ECMAScript、 HTML、JSP、Java、LotusScript、ASP、CDF (Channel Definition Format)、DOM (Document Object Model)、Jscript、VBScript、JavaScript、LiveWire、Perl、和 WML (Wireless Markup Language)。使用 NetObjects ScriptBuilder, 开发者能够：

(1) 检查脚本的语法错误

(2) 创建面向对象的脚本组件。

(3) 访问语言的参考书目

(4) 使用 XML 定制语言参考书

(5) 检查文档与浏览器的兼容性

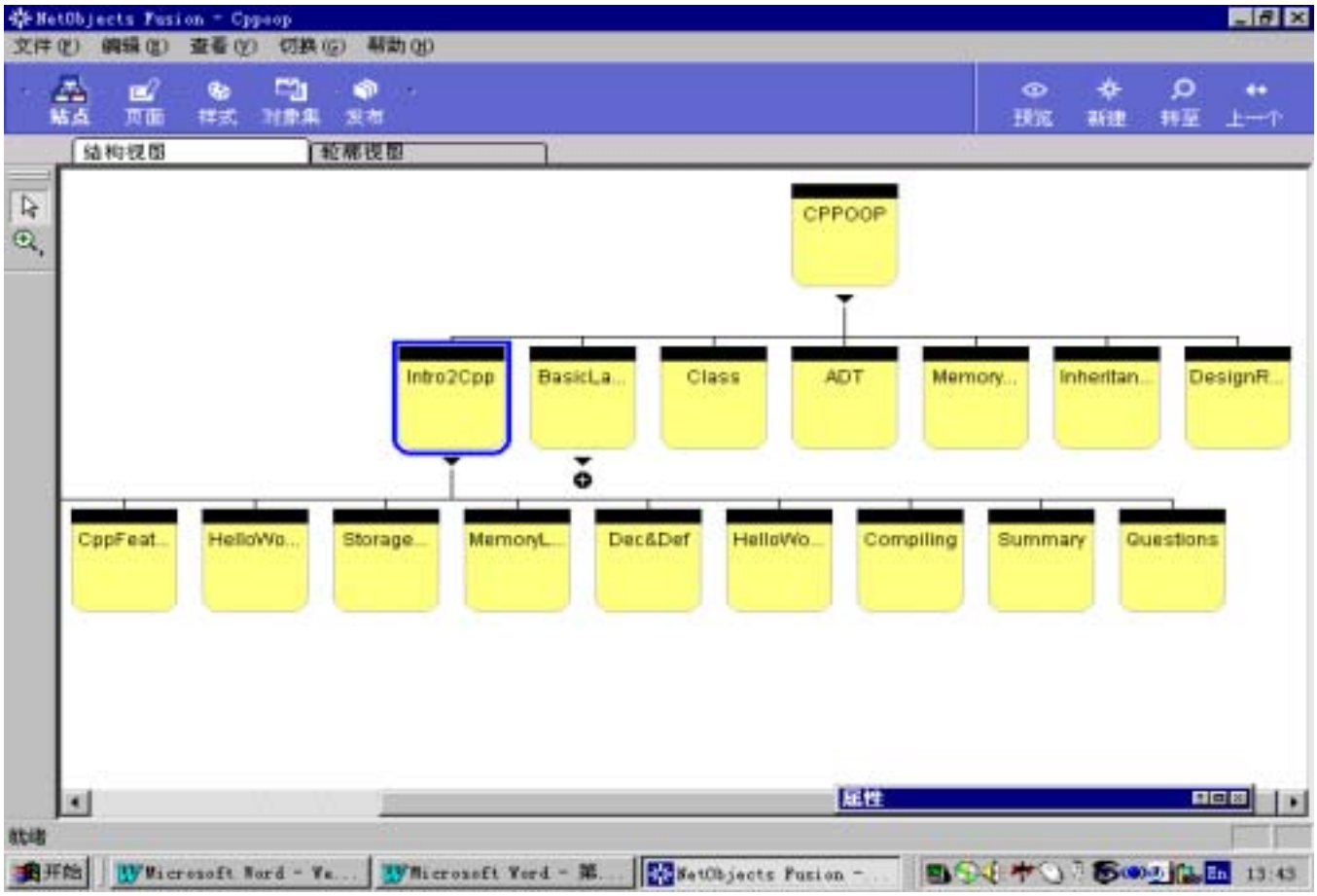
(6) 把所需的语言元素拖放到文档中。
(7) 使用标准的 Web 语言或其它语言
(8) 快速导航到嵌入的函数和组件
(9) 自动化重复性的任务
(10) 预览文档

3.3 NetObjects Fusion

NetObject Fusion 是一个可视化的 Web 站点的构建和管理工具。NetObjects Fusion 是 Web 站点构建过程的中心。NetObjects Fusion 是一个面向站点的工具，而不是仅仅处理单个页面。在 Web 站点开发的过程中，会创建许多页面，页面之间有很多链接。手工地更改这个结构是相当困难的。用 NetObjects Fusion 可以设计 Web 站点的通用元素，创建并维护 Web 站点的结构，构建各个页面，并能容易地维护这些页面。总之，使用 NetObject Fusion，可以设计 Web 站点、构建各个页面、容易地维护 Web 站点。下面介绍 NetObject Fusion 中的站点视图（Site view）、页面视图（Page view）、样式视图（Style view）、对象集视图（Assets View、或称资源视图）、和发布视图（Publishing view）。

1．站点视图

站点的结构视图（参见图 3-1）是站点视图的一种，单击“ 轮廓视图 ”标签就显示站点的轮廓视图。在结构视图中，页面的颜色是可以由开发者选择，通常用来为页面分组，但是页面颜色对实际页面没有影响。用户站点的文件格式为“ *.nod ”，模板文件格式为“ *.nft ”， 样式文件格式为“ *.ssf ”。使用站点视图，



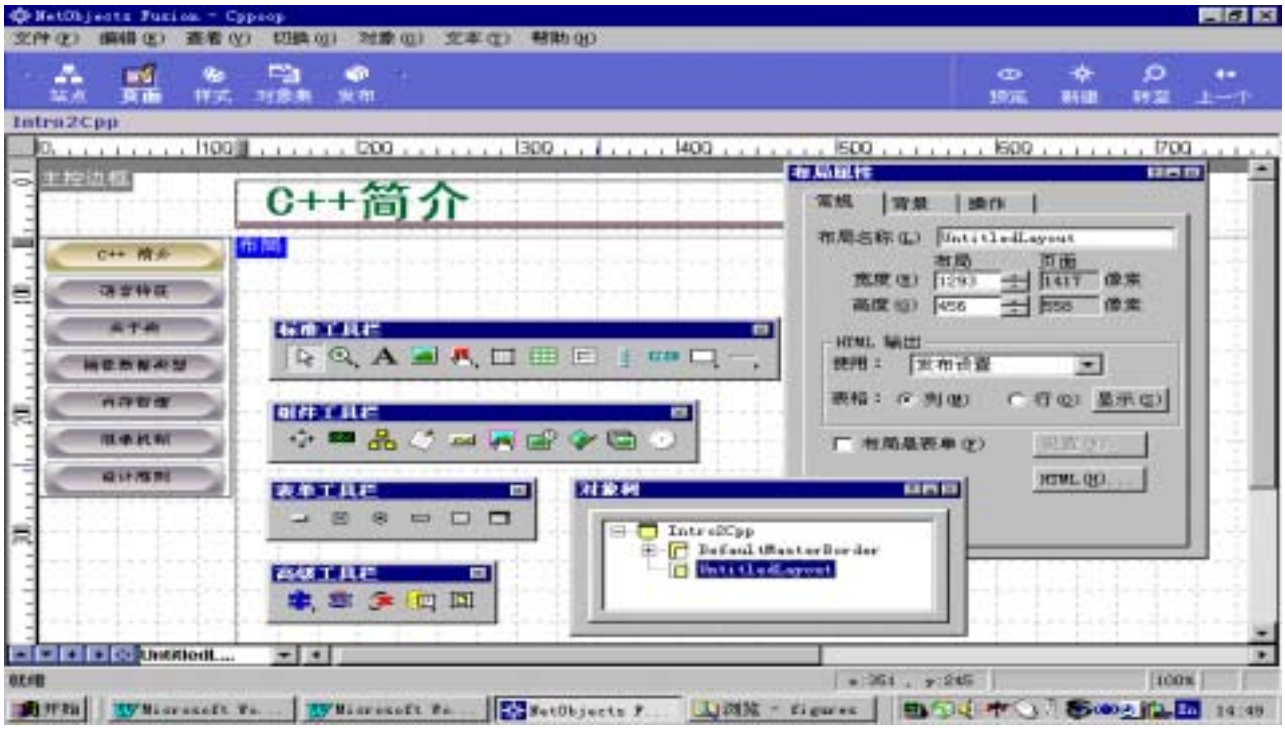
可以创建（或修改）一个站点的层次结构、操纵页面、引入/导出（Import/Export）站点、引入模板以改变站点的风格、打印站点的结构图等等。例如，可以引入一个已有的站点将它转成 NetObjects Fusion 的格式或保留为外部 HTML 页面。如果这些被引入的页面不会被改变，通常可以将引入的页面保留为外部 HTML 页面。

图 3-1：站点视图

2．页面视图

站点的页面视图（见图 3-2）包括页面布局（Page Layout）和主控边框（MasterBorder），并提供面板和工具栏。面板包括属性面板和对象树，工具栏包括标准工具栏、表单工具栏、组件工具栏、和高级工具栏。对象树是关于当前页面中所有对象的树状结构，开发者可以通过对象树来定位目标对象，也可以直接选择目标对象。一旦选定目标对象，属性面板就显示该对象的属性，对象属性依赖于对象的类型，不同类型的对象（如文本和 applet）有不同的属性，可以使用属性面板来设置对象属性，当然有的属性可以可视地编辑。

在页面视图中，可以可视地或通过属性面板来定义主控边框和页面布局。主控边框勾画出一个页面的五个区域：中间的布局区域和四个（左、右、上、下）边缘区域。这些边缘区域包含多个页面共有的元素，如水平标注和导航条，可以通过属性面板设置主控边框的自动边框（AutoFrame）属性，把某些边缘设置



成框架（frame）。关于布局，而布局区域用来放置该页面的内容。需要注意的是页面布局包括页面的实际内容，不仅仅是页面的布局，也许一个页面有好几个布局或版本，例如有框架（frame）的和没有框架的版本。NetObjects Fusion 使用表格（table）来组织它生成的 HTML 页的内容，也可以通过设置布局属性来选择布局是表单。也可以通过弹出菜单或单击属性面板中的“ HTML ”按扭在来显示生成的 HTML 文档。

图 3-2　页面视图

工具栏通常在窗口的边上，也可以拖进窗口里面（如图 3-2）。工具栏包括标准工具栏、表单工具栏、组件工具栏、和高级工具栏。这些工具栏为页面的制作提供了许多实用工具。

(1) 标准工具条

- 选择（Select）：选择一个特定的组件并移到另一个地点或重新设置大小。
- 缩放（Zoom In/Out）：使页面的视图围绕鼠标击中的区域放大或缩小。
- 文本（Text）：向页面增加一个包含一些文本的组件。
- 图片（Picture）：向页面增加一个图片。可以是一个.gif 或.jpg 文件。Fusion 接受 BMP，PCX，PICT，GIF，JPEG 的图。
- 热点（Hotspots）：把一个图片中的某个区域标记为可单击（链接到某个 URL），从而使该区域成为热点。这个区域可以是圆形、矩形、和多边形
- 布局区域（Layout Region）：在布局区域或主控边框里创建一个布局区域，它可以拥有与包含元素不同的布局属性（例如，一个不同的背景色）。

● 表格（Table）：创建一个 HTML 表格，可以规定表格的行数和列数。

● 表单区域（Form Area）：创建一个或多个表单。可以选择创建基于布局区域的表单、基于表格的表单、或基于文本的表单。也可以把一个表单加到边缘区域，从而能在多个页面里显示。

● 导航条（Navigation Bar）：用来创建导航条。可以使用按钮或文字形式，从当前页面链向站点里的别的页面，比如在 Web 站点结构里的父层、当前层或子层。

● 水平标注（Banner）。用来在页面的任何地方创建一个水平标注，反映这个页面的名字。通常在页面的顶部，来标志这个页面。

● 绘制（Draw）：绘制工具，这些工具用来在一个页面里画圆形、矩形、圆角矩形和多边形。

● 直线（Line）：画直线的工具，这些工具用来在一个页面里画水平线、任意角度的在线、站点样式的水平线。

需要注意的是 Hotspots、Draw、Line 这三个工具还包括可选的子工具，按下按扭不放就会显示可选的子工具，保持按下按扭并移动就可选择所要的工具。

(2) 表单工具栏

这些工具用来创建表单。表单可以用来收集信息，然后送回服务器端 CGI 程序或 Java Servlet。

● 表单按扭（Button）：一个表单按钮通常被用来启动一个后端服务程序或来重置表单。

● 表单复选框（Check Box）：一个复选框代表一个布尔值，选中表示真值，否则表示假。

● 表单单选按扭（Radio Button）：一个单选按钮是一组单选按扭的一部分，这组单选按扭是互斥的，即只能有一个为真，而别的都为假互斥的。

● 表单编辑字段（Edit Field）：表单上的一个区域，用户可以输入一行信息如名或姓。

● 表单多行文本（Multi-Line）：表单上的一个区域，用户可以输入多行信息，如对一个产品的说明或目录里的一项。

● 表单组合框（Combo box）：一个下拉列表，用户可以选择。一个例子是：头发颜色的列表。用户可以选一个。

(3) 组件工具栏

● 动态按扭（DynaButtons）：对按钮使用站点风格

● 磁带（Ticker Tape）：一个水平滚动文字区域，不断重复。

● 站点映射器（Site Mapper）：一个用来显示站点结构的按扭。

● 消息板（Message Board）：一个 BBS 设施

● 表单句柄（Form Handler）：实现一个简单的 CGI 程序。用来检查表单中的字段

● 图片反转（Picture Rollover）：一个反转按扭。当鼠标滚过图片，图片改变成指定的图片反转。当鼠标在图片上单击，图片会改变，一个 URL 链接会显示。

● 基于时间的图片（Time Based Picture）：在这个地点的不同时间显示不同的图。

● 图片加载器（Picture Loader）：使用其它资源中的图片

● 旋转图片（Rotating Picture）：简单动画。通常用来显示广告。

(4)高级工具栏

● Media：一个子工具条，包括了一些项目象 Shockwave、Quicktime、Video、Sound 和别的插件

● Java：允许往页面上加一个 Java applet。

● ActiveX Control：允许往页面上加一个 ActiveX 控件。

● Data：一个子工具条，允许你访问内部和外部数据对象。

● External HTML：包括一些需要特殊的、手工写的 HTML 页面。可以在外部引用 HTML 文件并把它们放在站点的任何地方，而不是引入并转换这个 HTML。NetObjects Fusion 在发布 Web 站点时不分析这个 HTML。

上述工具栏提供的有关工具可用来增加页面的互动性。如可以在页面上增加脚本、放置 ActiveX 控件、增加 Java applets 等等。脚本可以加到一个 Fusion 生成的 HTML 元素的前面、后面或里面。增加 Java applets 时使用 Java 对象属性设置有关参数，Fusion 将增加的 class 文件作为对象集(assets)中的元素。

3．样式视图

样式视图（参见图 3-3）可用来选择站点样式、编辑样式、或创建新的样式。Fusion 提供的样式适合于简单的站点或者可用来取得用户的反馈信息，重要的站点通常需要创建新的样式。Fusion 用样式中指定的字体生成水平标注和按钮上的文字。因此，应该为水平标注和按钮选择任何你的系统里的字体，Fusion 将使用指定的字体生成有关图象，浏览器只看到一个图象。对于在 HTML 里创建的文字组件，浏览器可以控制自己使用的字体。

图 3-3：样式视图

4. 对象集视图

对象集视图（参见图 3-4）是一个管理所有对象的地方，包括文件、链接、数据对象、变量，对象通过别名来访问。Fusion 提供预定义的变量，如日期、时间和站点名等等。开发者可以创建自己的变量，并在对象集视图中管理。每当开发者需要选择图象或别的文件时，打开文件对话框（Open File dialog）上会有一个对象集标签（Assets tab），单击该标签就显示对象集中的有关文件，开发者就可以选择所需要的文件。这种就对象进行统一管理的方式支持“一次更新，每个地方都更新”。例如标识公司的图案（logo），如果一个公司改变了它的标识图案，只要在对象集视图中改变有关标识图案的文件，Fusion 可以改变这个站点里的每一页。

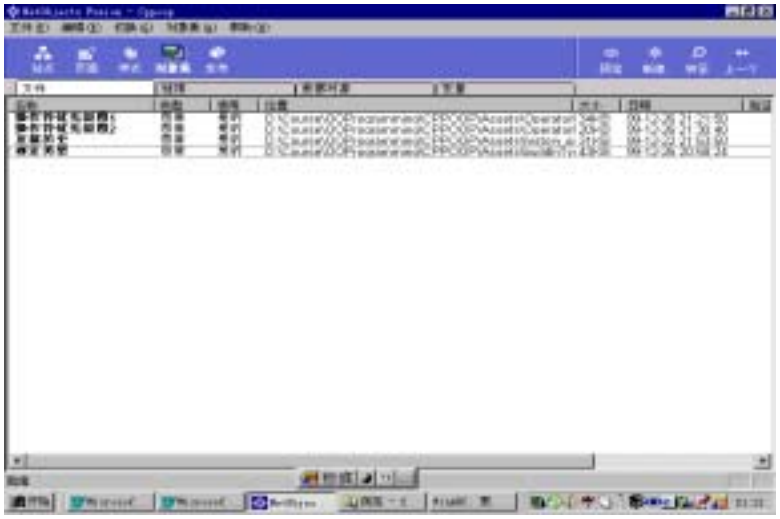


图 3-4：对象集视图

5．发布视图

发布视图（参见图 3-5）用来配置发布选项，并进行发布。发布选项包括本地发布位置和远程发布位置等。本地发布是指发布到本机的文件系统里，远程发布需要使用 FTP 把文件传输到服务器。如果你选择发布到一个远程站点，这个站点必须支持 FTP，发布者必须知道所需的用户名和口令。通常在正式发布之前需要发布到一个临时服务器上进行测试，并需要使用不同平台上的不同浏览器进行测试。发布时，Fusion 将生成 HTML 页面以及目录结构，拷贝所有的对象到指定的目录，这些目录里包含 HTML 文件、图象、Java 类等等。目录结构可以是单一的（flat）、基于对象类型的（asset）、或基于站点区域的（section）。所有生成的 HTML 使用相对路径。在发布之前还可以使用预览功能。预览创建 HTML 文件，生成任何需要的图象，但它不拷贝任何别的对象，从而不显示嵌入的组件。在按下 CTRL 键时单击预览按扭就可以只预览当前页。

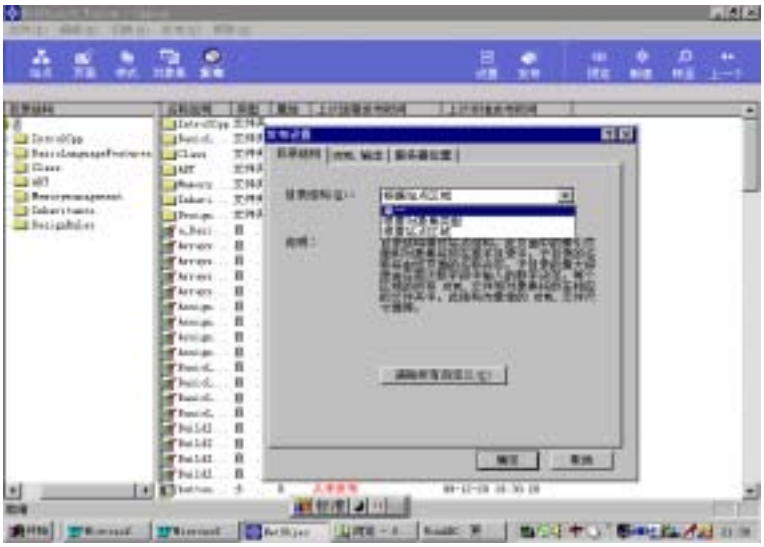


图 3-5：发布视图

总之，Fusion 是一个 Web 站点的组装工具，对管理站点范围的格式模板有强力支持，支持丰富的“对象”，链接和对象集管理使得部署到多个站点变得很容易。可以使用 Fusion 设计站点结构、构建和管理站点页面、设计和写页面内容、设计站点的图形化风格、管理和测试站点的对象集、把站点发布到本地或远程服务器、更新和维护站点。

3.4 NetObjects BeanBuilder

NetObjects BeanBuilder 是创建多媒体 Java applet 的可视化写作工具。它的前身是 Lotus BeanMachine。NetObjects BeanBuilder 使得开发者能用多媒体、特效、灵巧的表单和动态数据来丰富网站，而不要写一行 Java 代码，或者只需写一些简单的 Java 代码。BeanBuilder 是通过把 JavaBean 结合进 applet 的可视化编辑工具来实现这一强大的功能。开发者只要从 BeanBuilder 的样式库（Palette）里选择丰富的可重用的 JavaBean，设置几个属性，就完成了多媒体 Java applet 的开发。如果在样式库里没有需要的那个 JavaBean，可以用 BeanBuilder 的部件引入向导（Import Part Wizzard）引入 JavaBean 或 Java 类。甚至也可以引入已有的 applet。

1. Applet 概述

Applet 是 Panel 的一个子类，可以包含 GUI 组件。与一般的应用程序不同的是，applet 只能在浏览器里运行。Applet 使用称之为沙箱（“sandbox”）的安全模型。在这个模型中，网上下载的 applet 缺省地是不可信代码（不可信 Applet），不可信 Applet 有下列约束：

(1) 不能读或写本地文件系统；

(2) 不能进行网络操作，除非是对于 applet 的源主机；

(3) 不能装载某些类，如 SecurityManage 子类和 URLContentFactory 等；

(4) 不能访问它所运行的线程组以外的线程或线程组；

(5) 不能使用广播字节套（multicast sockets）；

(6) 不能访问 java.security 包

这个安全模型使得用户放心地使用 applet。然而，这些安全限制使编程的任务变得更困难。在 JDK1.1 里，Applets 可以被签署（sign）。如果运行时环境的安全政策认为“签署者”是可信任的话，这些被签署的 applets 可以被认为是“可信任的”代码。在 JDK1.2 里，所有的 Java 代码都是能被限制的或签署的。

Applets 通常被嵌入 HTML 文档中，applet 标记是在一个 Web 页面里使用 applet 的标准机制。举例如下：

```
<Applet
        CODE= Register.class
        WIDTH=500
        HEIGHT=500
        CODEBASE= classes
        ARCHIVE= Register.jar>
        <PARAM NAME= fullName VALUE= "Henry Burns">
        <PARAM NAME= age VALUE= "12">
</Applet>
```

CODE，WIDTH，HEIGHT 属性是必需的选项，其它的属性和包含的标记是可选的，如 ARCHIVE 属性和 CODEBASE 属性和 PARM 标记。有关属性和标记，及其它们的含义如下：

(1) CODEBASE= Java 代码所在的 URL（不是 document base）。CODEBASE 允许规定.class 文件所在的路径，使用页面服务器上的路径或 URL。

(2) CODE= Applet 类文件的名字。CODE 属性可以指定为在页面服务器上的路径或 URL，值对大小写敏感，而且必需包括.class 扩展名。

(3) WIDTH= applet 显示区域的宽度。

(4) HEIGHT= applet 显示区域的高度。WIDTH 和 HEIGHT 用像素点规定浏览器定位的空间。

(5) ALT=如果浏览器不能理解标记的话，要使用的文字

(6) ARCHIVE=允许使用 JAR 文件，作为发现 applet 文件的地方

(7) NAME= applet 的名字。多个 applets 之间的通信要用到这个。

(8) ALIGN= 告诉浏览器怎样在显示区域里排列 applet

(9) VSPACE= applet 和别的组件在垂直方向上的间隔

(10) HSPACE= applet 和别的组件在水平方向上的间隔

(11) PARM 标记，要求一个 NAME 和一个 VALUE 成对出现。使用 PARAM 标记来指定一个 applet 的参数。PARM 标记必须在<APPLET>和</APPLET>之间出现。NAME 部分是大小写不敏感的，而 VALUE 部分是大小写敏感的。所有传给 applet 的参数都被当作字符串。

在上述嵌入 HTML 文档的 applet 例子中，Applet 类由 Register.class 定义，该文件存放在 Web 服务器中 HTML 文档目录的 classes 子目录中，而且 Register.class 文件很可能被包含在 Register.jar 文件中。Jar 文件是包括一个附加的”manifest”文件的压缩 ZIP 文件，通常用来把一个 applet 需要的二进制文件、源文件和资源文件等有关文件结合进一个文件。浏览器取得 jar 文件之后，需要解压缩得到所需的.class 文件及可能的相关资源文件。Jar 文件也可以用于 Java 应用程序或 Servlet，但一般用于 Applet。如果一个 Jar 文件用于一个应用程序或 Servlet，在运行它们以前会有一个额外的解压缩的过程。对 applets 而言，这个额外的解压缩处理是值得的，因为节约了网络传输时间。最后，jar 文件也可以被用作一个分发 Java 软件包的机制。

在 applet 代码中，程序员可以使用 getDocumentBase()方法来获得包含 applet 标记的 HTML 页面的 URL。使用 getCodeBase（）来找出 CODEBASE 属性选项的设置。如果 CODEBASE 属性选项是空白的，getCodeBase()方法返回与 getDocumentBase()一样的值。使用 getParameter(String)方法取得参数的值，通常需要对传进 Applet 的 VALU 还要进行差错处理。

2．标准样式库

样式库中包含了许多用于组成 applet 的 bean。BeanBuilder 提供了缺省的样式库，为了把样式库中的某个 bean 放置在 applet 中，可以在样式库中单击这个 bean 的图标。此操作可将 bean 装载到光标上，并将光标变成十字指针形状，然后在设计器上再次单击鼠标放置 bean。样式库上的 beans 包括附件 Beans、控件 Beans、多媒体 Beans、和网络 Beans 四种类型。

A.附件分类

(1) 布尔求值：用于对布尔值进行 AND 和 OR 运算；

(2) 数学公式：用于执行基本的数学运算；

(3) 数字求值：用于比较两个数字的大小；

(4) 文本源：用于访问文本文件的内容，对于一个远程文件可以使用一个 URL 表示，或对于一个在可信任的 applet 里的本地文件使用一个文件名表示。

B.控件分类

(1) 按钮：按钮 bean。

(2) 复选框：复选框 bean。

(3) 选择框：下拉列表 bean。

(4) 标签：显示单行文本的 bean。

(5) 列表框：列表框 bean。

(6) 面板：可以包容其他 bean 并且具有布局样式的 bean。

(7) 文本区：可以输入多行文本的 bean。

(8) 文本域：只能输入单行文本的 bean。

这些是标准的 AWT 控件，但是，这些控件提供的有关事件更丰富。

C.多媒体分类

(1) 动画：用于显示一组连续的图片。

(2) 声音：用于播放声音剪辑。

(3) 时钟：用于显示日期和时间。

(4) 图像：用于显示图片文件。

(5) 动作：用于沿着路径移动其它 bean。

(6) 跳跃文本：用于显示在屏幕上跳动的文本。

(7) 感应按钮：当鼠标经过时可以改变图像。

(8) 阴影文本：显示带有阴影的文本。

(9) 打字机：用于显示文本字符并伴随播放声音。

(10) 文本：用于显示文本。

(11) 行走文本：用于在屏幕上显示移动的文本。

(12) 计时器：用于计算时间。

D.网络分类

(1) 数据库：用于显示数据库中的数据。

(2) 电子邮件：用于组织和发送电子邮件而不用浏览器邮件对话框。

(3) 新闻摘要：用于显示包含 URL 链接的滚动文本列表。

(4) HTML 参数：用于读取 HTML 参数。

(5) 邮件链接：用于显示浏览器的邮件对话框。

(6) URL 链接：用于连接到其他 Web 页面。生成的 URL 以./开头的，即使用相对于 HTML 页面的相对路径，正斜杠和反斜杠都是一样的。

(7) 大字标题；用于显示带 URL 链接的滚动文字列表

BeanBuilder 提供的控件同 VisualAge for Java 提供的不一样。例如，BeanBuilder 有出色的多媒体和动画控件。

3．添加与定制样式库

在样式库中添加 JavaBean。只要单击添加按钮，就将出现对话框（参见图 3-6）以进行选择。典型情况下可以从 .jar 文件添加现有的 bean。为此目的，单击“从 jar 文件中添加 bean”单选按钮，在文本框中输入 jar 文件的路径或单击查找...按钮以查找所需的 .jar 文件。使用分类域中的下拉列表以指定添加 bean 的样式库分类。单击确定增加 bean。如果指定的 jar 文件有多个 bean，所有的将添加到指定的分类中。如果想要在添加 bean 时添加新的分类，只要在分类域中输入新名称。然后单击确定添加 bean 和新的分类。注意：也可通过将 .jar 文件放置到 BeanBuilder 的“Beans”文件夹将 bean 添加到样式库中。“Beans”文件夹包含所有 bean 的 BeanBuilder 样式库。下一次 BeanBuilder 启动时，它将会探测到新添了 bean，并自动将它添加到样式库中。BeanBuilder 选择 jar 文件的名称作为 bean 分类的名称。通常也可以用 VA Java 来创建 beans，然后添加到 BeanBuilder 样式库。

除了向 BeanBuilder 样式库添加以外，还可以定制样式库。要定制样式库，单击样式库上的定制按钮。这将显示对话框以提供选项。可以重新排列 bean、隐藏 bean 或删除它们。

(1) 要重新排列样式库中的 bean，可在列表中选择想要移动的 bean，并使用箭头按钮在列表中移动它到想要的位置。甚至可以移动它到另一个分类。

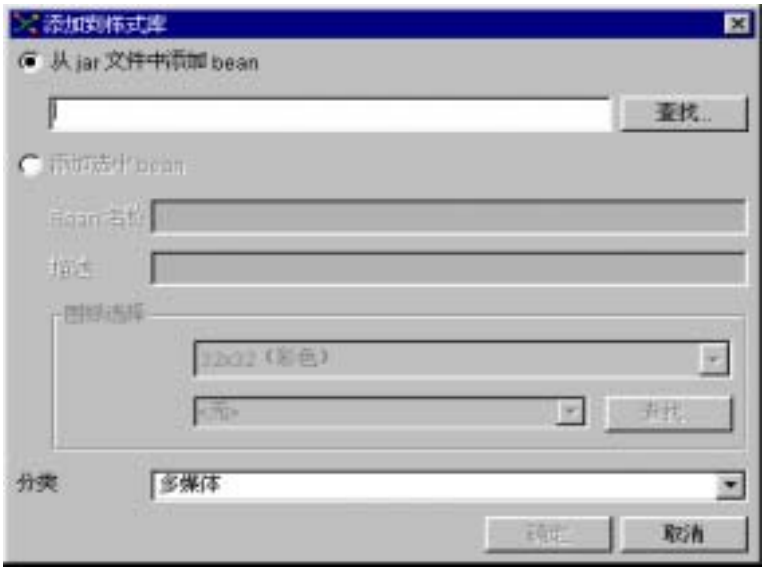


图 3-6：添加到样式库

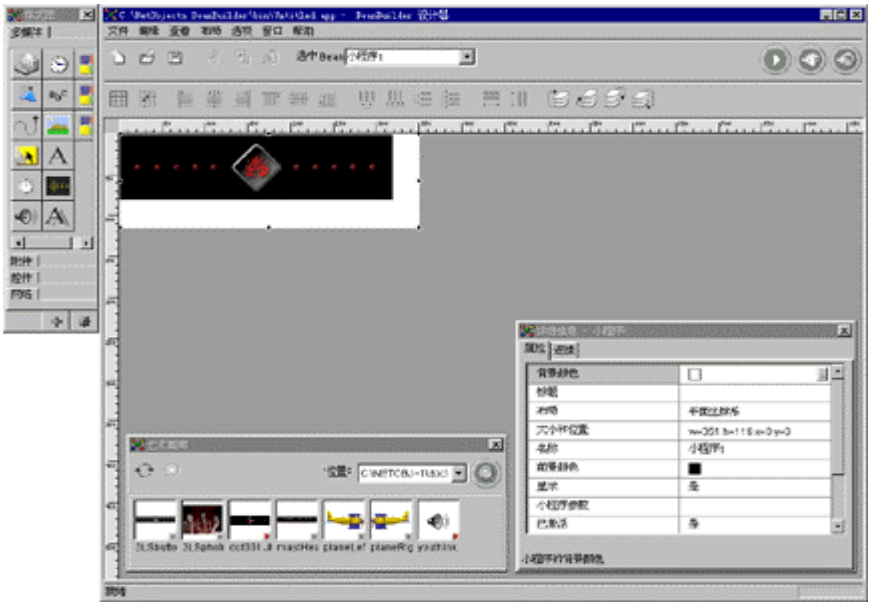
(2) 也可以隐藏特定的 bean 或整个分类的 bean。要隐藏 bean 或分类，可单击它旁边的复选框。如果框包含复选标记，此特定 bean 或分类将显示在样式库中。如果框不包含复选标记，bean 或分类将在视图中隐藏。

(3) 要完全删除 bean，突出显示该 bean 并单击删除按钮。如果可能以后需要该 bean，可使用隐藏替代删除，这样能够快速恢复此 bean 而不用重新增加它到样式库中。

注意：虽然可以删除创建的分类，但不能删除 BeanBuilder 所带的分类。要注意到删除按钮在这些分类突出显示时，是不能使用的。如果删除分类，所有的 bean 将会消失。

4．构建 applet

单击 Windows 的“开始”按钮，从“程序”菜单中启动 BeanBuilder 进入“新建 applet 向导”。可以通过这个向导快速创建第一个 applet。阅读并熟悉向导，在准备就绪之后单击“下一步”，在“动画”附签中选择“是”。单击“查找”定位动画的第一帧，将 BeanBuilder 安装目录的 doc 文件夹中的 dot001.jpg 作为动画的第一帧。BeanBuilder 使用相同的基础文件名（dot002.jpg、dot003.jpg 等）在第一帧图像所在的文件夹中搜索其他序号的文件，从而自动定义动画的其余帧。在本样例中，BeanBuilder 查找到 7 个文件来完成动画。BeanBuilder 自动识别动画中剩余帧。选择“中速”定义动画的播放速度。可以单击“预览”按钮预览动画的播放效果。这样就简单的完成了动画的创建。单击“完成”继续下面的操作。此时的设计



器窗口如图 3-7 所示：

图 3-7：设计器窗口

下面让我们通过创建“连接”来使它动起来。连接可以看作是两个部件之间的命令或交互作用。连接的作用方式为“当事件发生时，执行操作”。操作可以是诸如播放声音剪辑或者设置属性（背景颜色）之类的任意动作。在本样例中，操作就是播放动画。记住，所有的操作都要由事件触发。通过直接定义触发操作的事件来创建连接。在本样例中，我们需要在单击按钮之后开始播放动画。因此在创建连接之前，首先添加一个按钮。

BeanBuilder 中提供了多种按钮，可以使用标准按钮或是更为生动的感应按钮。感应按钮中附加了一些内置特性，当将鼠标移动到感应按钮上，或者在其上按下鼠标时，可以自动显示不同的图像。首先确保样式库中显示多媒体分类；如果没有，从下拉框中选择它。当多媒体分类出现后，单击感应按钮 bean。此时的光标将转换成十字指针，并且装载感应按钮 。下一步，将光标移动到 applet 画布上，单击鼠标并放置 bean。需要设置某些属性，以便定制感应按钮。可以在详细信息窗口的属性附签中将感应按钮链接到创建的图片文件上。根据按钮状态的不同，显示的图片也有所不同：

(1) 正常图片：鼠标指针没有移动到按钮上时的图片。
(2) 移入时图片：鼠标指针移动到按钮上时的图片。
(3) 按下时的图片：鼠标单击按钮时的图片。
(4) 禁止时图片：用户不能操作按钮时的图片。

在详细信息窗口的属性附签中为不同的感应按钮状态指定不同的图片。选择图片属性，然后单击右边的按钮。屏幕上将显示“打开”对话框，可以指定 3LSbutton3.jpg 作为图片属性，3LSbutton3P.jpg 和 3LSbutton3R.jpg 分别作为按下时图片和移入时图片属性。将按钮的名称属性修改为 3LostSoulsButton。如果没有定义感应按钮的移入时图片、单击时图片或者禁止时图片，那么 BeanBuilder 将使用已经指定的艺术图片。

在创建连接之前，选择动画 bean 并且将它的自动启动属性设置为否。如果将这个属性设置为是，自动启动属性将告诉 applet 在启动时自动播放动画。但是，在本样例中，我们希望单击感应按钮之后再播放动画。现在，我们开始创建连接。首先，必须确定选中按钮。通过下面的三种方法可以知道当前选中的 bean：选中 bean 的四周会显示选取控制端点，它的名称会显示在详细信息窗口的标题条中，它的名称会显示在主工具条的选定 Bean 的下拉列表中。如果选中了感应按钮，那么详细信息窗口（参见图 3-8）的标题条将显示详细信息—3LostSoulsButton。单击连接附签，由于还没有定义任何连接，此时连接列表的内容为空。在事件下面选择第一个元素，屏幕上将显示一个下拉列表，可以在这里查看感应按钮的事件。因为我们希望单击按钮后播放动画，因此从列表中选择“单击”。接着选择 Bean 下面的元素，在这里查看当前 applet 中可以与感应按钮进行连接的 bean。在列表中选择动画 1，最后的单元是操作列表，可以在这里查看单击感应按钮时动画 bean 执行的操作。为使动画在单击按钮时开始，选择启动操作。

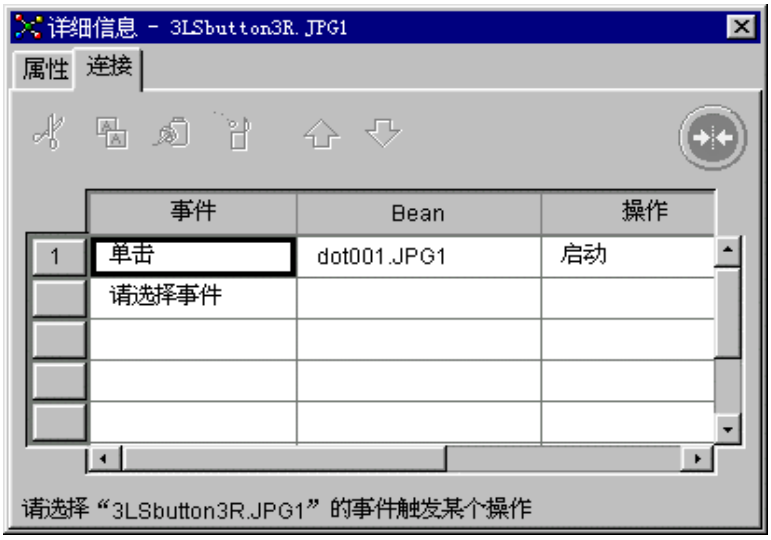


图 3-8：详细信息窗口

现在，请保存这个 applet。保存之后，可以单击主工具条上的运行按钮进行测试。短暂的停顿之后，屏幕上将出现 applet 查看器（Applet Viewer）窗口并显示刚创建的运行 applet。单击按钮测试动画的播放效果。测试完成之后，关闭 applet 查看器窗口。

通过类似的方法，可以添加声音并创建一些有趣的连接。单击主工具条上的试演按钮可以进行试演。单击主工具条上的发布按钮可以将刚才创建的 3 Lost Souls applet 中包含的图像文件、动画文件以及 BeanBuilder 创建的 Java (.class) 文件放到服务器上。单击主工具条上的发布按钮启动发布向导，在向导的指导下将 applet 放置在本地文件系统或远程的任何地方。在这里，我们将 applet 发布到本地文件系统。定位域中已经添入文件夹名 Publish。这是运行 applet 所需的所有文件的位置。当然，可以修改此文件夹为所需文件夹。在发布向导上单击完成按钮之后，BeanBuilder 整理 applet 中包含的所有文件并把它们复制到指定的文件夹中。如果已经设置好 Web 服务器，那么只需把这些文件从发布文件夹复制到服务器的文件夹中即可。记住，必须把 applet 中的所有文件和 Java (.class) 文件放置在服务器的同一个文件夹中，以便 applet 能够正确运行。这样就完成 Java 动画的开发和发布。

总之， BeanBuilder 是一个 applet 开发环境。它提供了非常灵活的手段，帮助用户发挥 Java 的强大功能，并自带丰富的 JavaBeans。它使得放置可视化控件、创建动画 applet、写简单的 Java 代码变得容易。对需要客户端动态的页面，applet 是理想的，而 BeanBuilder 是开发 applet 的及其优越的可视化制作工具。

3.5 VisualAge for Java

如果您熟悉 Java 程序设计，可以使用 VisualAge for Java(VAJ)来创建和定制 JavaBean。在您使用 Studio 的向导构建 Servlet 时很可能需要复杂的 JavaBean，这些 JavaBean 一般可使用 VAJ 来开发。也可使用 VAJ 来修改与完善由 Studio 向导生成的 Java Servlet 和 Bean。

VAJ 是一个全功能的 Java 开发环境，包含一个智能化集成开发环境，内含丰富的 Java 类库、高性能 Java 编译器和虚拟机，集成了包括向导和调试器在内的多种工具，支持 JavaBeans 的开发，通过先进的版本控制技术和贮存（Repository）机制支持团队开发(team development)。

VAJ 的工作台为你提供各种视图，每个视图作为笔记薄（Notebook）中的一页

- 项目－在工作区中所有项目及所含包、类的树状视图
- 包－所有包、类、方法的树状视图
- 类－工作区中的所有类的列表
- 接口－工作区的所有接口及其方法的列表
- 所有问题－所有出错元素及相应错误信息的列表

项目页允许你为一个元素作标签，这样只需简单的一次单击就可定位到它。另外，当你保存文件时，将进行编译和错误标志，更正错误无需长时间编译，从而体现了应用的快速开发。使用 VAJ 的工作台进行 Java 编程的步骤如下：

- 创建项目（和包）
- 创建类（可使用向导）
- 添加方法（可使用向导）
- 在源代码页中编辑方法，在菜单上单击保存－将进行快速编译
- 单击“运行”按钮测试代码

使用 VAJ 的导入／导出功能可以将有关文件（JAR、class、java）导入或导出 VAJ 的开发环境。值得关注的是，WebSphere 可在整个 VAJ 中运行，允许你在 VAJ 环境中编写和调试 servlet 和 JSP。对于 VAJ2.0 来说，需要下列步骤：

- 安装 VisualAge for Java update
- 载入 JSP 执行监视器，改正语法错误
- 在 com.ibm.servlet 包中启动 SERunner web 服务

而 VAJ3.0 已经和 WebSphere Application Server、WebSphere Studio、DB2 Universal Database 紧密集成，从而成为 IBM 电子商务应用框架的关键点。VAJ3.0 的新的特性主要包括：

- WebSphere 测试环境
- JSP/Servlet 开发环境
- 存贮过程构建器
- SQLJ 工具
- EJB 开发环境（VAJ3.0 企业版）
- Domino 企业存取构建器（VAJ3.0 企业版）
- XML 元数据交换(XML Metadata Interchange、XMI)工具包（VAJ3.0 企业版-技术预演）

第四章 Java Servlet 技术

4.1 Servlet 简介

1. Servlet 是什么？

Servlet是使用Java Servlet 应用程序设计接口（API）及相关类和方法的 Java 程序。除了 Java Servlet API，Servlet 还可以使用用以扩展和添加到 API 的 Java 类软件包。Servlet 在启用 Java 的 Web 服务器上或应用服务器上运行并扩展了该服务器的能力。Java servlet对于Web服务器就好象Java applet对于Web浏览器。Servlet装入Web服务器并在Web服务器内执行，而applet装入Web浏览器并在Web浏览器内执行。Java Servlet API 定义了一个servlet 和Java使能的服务器之间的一个标准接口，这使得Servlets具有跨服务器平台的特性。

Servlet 通过创建一个框架来扩展服务器的能力，以提供在 Web 上进行请求和响应服务。当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应。 当启动 Web 服务器或客户机第一次请求服务时，可以自动装入 Servlet。装入后， Servlet 继续运行直到其它客户机发出请求。Servlet 的功能涉及范围很广。例如，Servlet 可完成如下功能：

(1) 创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML页面。

(2) 创建可嵌入到现有 HTML 页面中的一部分 HTML 页面（HTML 片段）。

(3) 与其它服务器资源（包括数据库和基于 Java 的应用程序）进行通信。

(4) 用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如，Servlet 可以是多参与者的游戏服务器。

(5) 当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至applet的新连接，并将该连接保持在打开状态。当允许客户机和服务器简单、高效地执行会话的情况下，applet也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准（如 IIOP）进行通信。

(6) 对特殊的处理采用 MIME 类型过滤数据，例如图像转换和服务器端包括（SSI）。

(7) 将定制的处理提供给所有服务器的标准例行程序。例如，Servlet 可以修改如何认证用户。

2．Servlet 的生命周期

Servlet 的生命周期始于将它装入 Web 服务器的内存时，并在终止或重新装入 Servlet 时结束。

(1) 初始化

在下列时刻装入 Servlet：

●如果已配置自动装入选项，则在启动服务器时自动装入

●在服务器启动后，客户机首次向 Servlet 发出请求时

●重新装入 Servlet 时

装入 Servlet 后，服务器创建一个 Servlet 实例并且调用 Servlet 的 init() 方法。在初始化阶段，Servlet 初始化参数被传递给 Servlet 配置对象。

(2) 请求处理

对于到达服务器的客户机请求，服务器创建特定于请求的一个“ 请求 ”对象和一个“ 响应 ”对象。服务器调用 Servlet 的 service() 方法，该方法用于传递“ 请求 ”和“ 响应 ”对象。service() 方法从“ 请求 ”对象获得请求信息、处理该请求并用“ 响应 ”对象的方法以将响应传回客户机。service() 方法可以调用其它方法来处理请求，例如 doGet()、doPost() 或其它的方法。

(3) 终止

当服务器不再需要 Servlet, 或重新装入 Servlet 的新实例时，服务器会调用 Servlet 的 destroy() 方法。

3．Java Servlet API

Java Servlet 开发工具（JSDK）提供了多个软件包，在编写 Servlet 时需要用到这些软件包。其中包括两个用于所有 Servlet 的基本软件包：javax.servlet 和 javax.servlet.http。可从sun公司的Web站点下载 Java Servlet 开发工具。 下面主要介绍javax.servlet.http提供的HTTP Servlet应用编程接口。

HTTP Servlet 使用一个 HTML 表格来发送和接收数据。要创建一个 HTTP Servlet，请扩展 HttpServlet 类， 该类是用专门的方法来处理 HTML 表格的 GenericServlet 的一个子类。 HTML 表单是由 <FORM> 和 </FORM> 标记定义的。表单中典型地包含输入字段（如文本输入字段、复选框、单选按钮和选择列表）和用于提交数据的按钮。当提交信息时，它们还指定服务器应执行哪一个Servlet（或其它的程序）。 HttpServlet 类包含 init()、destroy()、service() 等方法。其中 init() 和 destroy() 方法是继承的。

(1) init() 方法

在 Servlet 的生命期中，仅执行一次 init() 方法。它是在服务器装入 Servlet 时执行的。 可以配置服务器，以在启动服务器或客户机首次访问 Servlet 时装入 Servlet。 无论有多少客户机访问 Servlet，都不会重复执行 init() 。

缺省的 init() 方法通常是符合要求的，但也可以用定制 init() 方法来覆盖它，典型的是管理服务器端资源。 例如，可能编写一个定制 init() 来只用于一次装入 GIF 图像，改进 Servlet 返回 GIF 图像和含有多个客户机请求的性能。另一个示例是初始化数据库连接。缺省的 init() 方法设置了 Servlet 的初始化参数，并用它的 ServletConfig 对象参数来启动配置， 因此所有覆盖 init() 方法的 Servlet 应调用 super.init() 以确保仍然执行这些任务。在调用 service() 方法之前，应确保已完成了 init() 方法。

(2) service() 方法

service() 方法是 Servlet 的核心。每当一个客户请求一个HttpServlet 对象，该对象的service() 方法就要被调用，而且传递给这个方法一个“ 请求 ”（ServletRequest）对象和一个“ 响应 ”（ServletResponse）对象作为参数。 在 HttpServlet 中已存在 service() 方法。缺省的服务功能是调用与 HTTP 请求的方法相应的 do 功能。例如， 如果 HTTP 请求方法为 GET，则缺省情况下就调用 doGet() 。Servlet 应该为 Servlet 支持的 HTTP 方法覆盖 do 功能。因为 HttpServlet.service() 方法会检查请求方法是否调用了适当的处理方法，不必要覆盖 service() 方法。只需覆盖相应的 do 方法就可以了。

- 当一个客户通过HTML 表单发出一个HTTP POST请求时，doPost（）方法被调用。与POST请求相关的参数作为一个单独的HTTP 请求从浏览器发送到服务器。当需要修改服务器端的数据时，应该使用doPost()方法。
- 当一个客户通过HTML 表单发出一个HTTP GET请求或直接请求一个URL时，doGet()方法被调用。与GET请求相关的参数添加到URL的后面，并与这个请求一起发送。当不会修改服务器端的数据时，应该使用doGet()方法。

Servlet的响应可以是下列几种类型：

一个输出流，浏览器根据它的内容类型（如text/HTML）进行解释。

一个HTTP错误响应，重定向到另一个URL、servlet、JSP。

(3) destroy() 方法

destroy() 方法仅执行一次，即在服务器停止且卸装Servlet 时执行该方法。典型的，将 Servlet 作为服务器进程的一部分来关闭。缺省的 destroy() 方法通常是符合要求的，但也可以覆盖它，典型的是管理服务器端资源。例如，如果 Servlet 在运行时会累计统计数据，则可以编写一个 destroy() 方法，该方法用于在未装入 Servlet 时将统计数字保存在文件中。另一个示例是关闭数据库连接。

当服务器卸装 Servlet 时，将在所有 service() 方法调用完成后，或在指定的时间间隔过后调用 destroy() 方法。一个Servlet 在运行service() 方法时可能会产生其它的线程，因此请确认在调用 destroy() 方法时，这些线程已终止或完成。

(4) GetServletConfig（）方法

GetServletConfig （） 方法返回一个 ServletConfig 对象，该对象用来返回初始化参数和 ServletContext。ServletContext 接口提供有关servlet 的环境信息。

(5) GetServletInfo（）方法

GetServletInfo（）方法是一个可选的方法，它提供有关servlet 的信息，如作者、版本、版权。

当服务器调用sevlet 的Service（）、doGet（）和doPost（）这三个方法时，均需要 “ 请求 ”和“ 响应 ”对象作为参数。“ 请求 ”对象提供有关请求的信息，而“ 响应 ”对象提供了一个将响应信息返回给浏览器的一个通信途径。javax.servlet 软件包中的相关类为ServletResponse和ServletRequest，而 javax.servlet.http 软件包中的相关类为HttpServletRequest 和 HttpServletResponse。Servlet 通过这些对象与服务器通信并最终与客户机通信。Servlet 能通过调用“ 请求 ”对象的方法获知客户机环境，服务器环境的信息和所有由客户机提供的信息。Servlet 可以调用“ 响应 ”对象的方法发送响应，该响应是准备发回客户机的。

4.2 创建 HTTP Servlet

创建一个 HTTP Servlet，通常涉及下列四个步骤：

1. 扩展 HttpServlet 抽象类。

2. 重载适当的方法。]如覆盖（或称为重写）doGet() 或doPost()方法。

3. 如果有 HTTP 请求信息的话，获取该信息。用 HttpServletRequest 对象来检索 HTML 表格所提交的数据或 URL 上的查询字符串。“ 请求 ” 对象含有特定的方法以检索客户机提供的信息，有3个可用的方法：

- getParameterNames()，
- getParameter()，
- getParameterValues()。

4. 生成 HTTP 响应。HttpServletResponse 对象生成响应，并将它返回到发出请求的客户机上。它的方法允许设置“ 请求 ” 标题和“ 响应 ”主体。“ 响应 ”对象还含有 getWriter() 方法以返回一个 PrintWriter 对象。使用 PrintWriter 的 print() 和 println() 方法以编写 Servlet 响应来返回给客户机。或者，直接使用out对象输出有关HTML文档内容。

一个servlet样例(ServletSample.java)如下：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletSample   extends HttpServlet {       // 第一步： 扩展 HttpServlet 抽象类。

public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException       {       // 第二步：重写doGet()方法

      String myName = "";                                          // 第三步：获取HTTP 请求信息
      java.util.Enumeration keys = request.getParameterNames();
      while (keys.hasMoreElements());
      {
          key = (String) keys.nextElement();
          if (key.equalsIgnoreCase("myName"))
                myName = request.getParameter(key);
      }
      if (myName == "")
                myName = "Hello";
                                                          // 第四步：生成 HTTP 响应。
      response.setContentType("text/html");
      response.setHeader("Pragma", "No-cache");
      response.setDateHeader("Expires", 0);
      response.setHeader("Cache-Control", "no-cache");

      out.println("<head><title>Just a basic servlet</title></head>");
      out.println("<body>");
      out.println("<h1>Just a basic servlet</h1>");
      out.println ("<p>" + myName +    ", this is a very basic servlet that writes an HTML page.");
      out.println ("<p>For instructions on running those samples on your WebSphere应用服务器, "+
                  "open the page:");
```

```
        out.println("<pre>http://<em>your.server.name</em>/IBMWebAs/samples/index.html</pre>");
        out.println("where <em>your.server.name</em> is the hostname of your WebSphere应用服务器.");
        out.println("</body></html>");
        out.flush();
    }
}
```

上述ServletSample类扩展 HttpServlet 抽象类、重写doGet()方法。在重写的doGet()方法中，获取 HTTP 请求中的一个任选的参数（myName），该参数可作为调用的 URL 上的查询参数传递到 Servlet。使用示例如下：http://your.server.name/servlet/ServletSample?myname=Michael。

4.3 调用 Servlet

要调用 Servlet 或 Web 应用程序，请使用下列任一种方法：由 URL 调用、在 <FORM> 标记中调用、在 <SERVLET>标记中调用、在 JSP 文件中调用、在 ASP 文件中调用。

1．由 URL 调用 Servlet

这里有两种用 Servlet 的 URL 从浏览器中调用该 Servlet 的方法：

(1) 指定 Servlet 名称：当用 WebSphere应用服务器 管理器来将一个 Servlet 实例添加（注册）到服务器配置中时，必须指定“ Servlet 名称 ”参数的值。例如，可以指定将 hi 作为 HelloWorldServlet 的 Servlet 名称。要调用该 Servlet，需打开 http://your.server.name/servlet/hi。也可以指定 Servlet 和类使用同一名称（HelloWorldServlet）。在这种情况下，将由 http://your.server.name/servlet/HelloWorldServlet 来调用 Servlet 的实例。

(2) 指定 Servlet 别名：用 WebSphere应用服务器 管理器来配置 Servlet 别名，该别名是用于调用 Servlet 的快捷 URL。快捷 URL 中不包括 Servlet 名称。

2．在 <FORM> 标记中指定 Servlet

可以在 <FORM> 标记中调用 Servlet。HTML 格式使用户能在 Web 页面（即从浏览器）上输入数据，并向 Servlet 提交数据。例如：

```
<FORM METHOD="GET" ACTION="/servlet/myservlet">
  <OL>
    <INPUT TYPE="radio" NAME="broadcast" VALUE="am">AM<BR>
    <INPUT TYPE="radio" NAME="broadcast" VALUE="fm">FM<BR>
  </OL>
（用于放置文本输入区域的标记、按钮和其它的提示符。）
</FORM>
```

ACTION 特性表明了用于调用 Servlet 的 URL。关于METHOD 的特性，如果用户输入的信息是通过 GET 方法向 Servlet 提交的，则 Servlet 必须优先使用 doGet() 方法。反之，如果用户输入的信息是通过 POST 方法向 Servlet 提交的，则 Servlet 必须优先使用 doPost() 方法。使用 GET 方法时，用户提供的信息是查询字符串表示的 URL 编码。无需对 URL 进行编码，因为这是由表单完成的。然后 URL 编码的查询字符串被附加到 Servlet URL 中，则整个 URL 提交完成。URL 编码的查询字符串将根据用户同可视部件之间的交互操作，将用户所选的值同可视部件的名称进行配对。例如，考虑前面的 HTML 代码段将用于显示按钮(标记为 AM 和 FM)，如果用户选择 FM 按钮，则查询字符串将包含 name=value 的配对操作为broadcast=fm。因为在这种情况下，Servlet 将响应 HTTP 请求，因此 Servlet 应基于 HttpServlet 类。Servlet 应根据提交给它的查询字符串中的用户信息使用的 GET 或 POST 方法，而相应地使用 doGet() 或 doPost() 方法。

3．在 <SERVLET> 标记中指定 Servlet

当使用 <SERVLET> 标记来调用 Servlet 时，如同使用 <FORM> 标记一样，无需创建一个完整的 HTML 页面。作为替代，Servlet 的输出仅是 HTML 页面的一部分，且被动态嵌入到原始 HTML 页面中的其它静态文本中。所有这些都发生在服务器上，且发送给用户的仅是结果 HTML 页面。建议在 Java 服务器页面（JSP）文件中使用 <SERVLET> 标记。请参阅有关 JSP 技术

原始 HTML 页面中包含 <SERVLET> 和 </SERVLET> 标记。 Servlet 将在这两个标记中被调用，且 Servlet 的响应将覆盖这两个标记间的所有东西和标记本身。如果用户的浏览器可以看到 HTML 源文件，则用户将看不到 <SERVLET> 和 </SERVLET> 标记。要在 Domino Go Webserver 上使用该方法，请启用服务器上的服务器端包括功能。部分启用过程将会涉及到添加特殊文件类型 SHTML。当 Web 服务器接收到一个扩展名为 SHTML 的 Web 页面请求时，它将搜索 <SERVLET> 和 </SERVLET> 标记。对于所有支持的 Web 服务器，WebSphere应用服务器 将处理 SERVLET 标记间的所有信息。下列 HTML 代码段显示了如何使用该技术。

```
<SERVLET NAME="myservlet" CODE="myservlet.class" CODEBASE="url" initparm1="value">
<PARAM NAME="parm1" VALUE="value">
</SERVLET>
```

使用 NAME 和 CODE 属性带来了使用上的灵活性。可以只使用其中一个属性，也可以同时使用两个属性。 NAME 属性指定了 Servlet 的名称（使用 WebSphere应用服务器 管理器配置的），或不带 .class 扩展名的 Servlet 类名。CODE 属性指定了 Servlet 类名。使用 WebSphere应用服务器 时，建议指定 NAME 和 CODE，或当 NAME 指定了 Servlet 名称时，仅指定 NAME。如果仅指定了 CODE，则会创建一个 NAME=CODE 的 Servlet 实例。装入的 Servlet 将假设 Servlet 名称与 NAME 属性中指定的名称匹配。然后，其它 SHTML 文件可以成功地使用 NAME 属性来指定 Servlet 的名称，并调用已装入的 Servlet。NAME 的值可以直接在要调用 Servlet 的 URL 中使用。如果 NAME 和 CODE 都存在，且 NAME 指定了一个现有 Servlet，则通常使用 NAME 中指定的 Servlet。由于 Servlet 创建了部分 HTML 文件，所以当创建 Servlet 时，将可能会使用 HttpServlet 的一个子类，并优先使用 doGet() 方法（因为 GET 方法是提供信息给 Servlet 的缺省方法）。另一个选项是优先使用 service() 方法。另外， CODEBASE 是可选的，它指定了装入 Servlet 的远程系统的 URL。请使用 WebSphere 应用服务器 管理器来从 JAR 文件配置远程 Servlet 装入系统。

在上述的标记示例中，initparm1 是初始化参数名，value 是该参数的值。可以指定多个 “ 名称-值 ” 对的集合。利用 ServletConfig 对象（被传递到 Servlet 的 init() 方法中）的 getInitParameterNames() 和 getInitParameter() 方法来查找参数名和参数值的字符串数组。在示例中，parm1 是参数名，并在初始化 Servlet 后被才被设置某个值。因为只能通过使用 “ 请求 ” 对象的方法来使用以 <PARAM> 标记设置的参数，所以服务器必须调用 Servlet service() 方法，以从用户处传递请求。要获得有关用户的请求信息，请使用 getParameterNames()、getParameter() 和 getParameterValues() 方法。

初始化参数是持续的。假设一台客户机通过调用一个包含某些初始化参数的 SHTML 文件来调用 Servlet。并假设第二台客户机通过调用第二个 SHTML 文件来调用同一个 Servlet，且该 SHTML 中未指定任何初始化参数。那么第一次调用 Servlet 时所设置的初始化参数将一直可用，并且通过所有其它 SHTML 文件而调用的所有后继 Servlet 都不会更改该参数。直到 Servlet 调用了 destroy() 方法后，才能重新设置初始化参数。例如，如果另一个 SHTML 文件指定了另一个不同的初始化参数值，虽然已此时已装入了 Servlet，但该值仍将被忽略。

4．在 JSP 文件中调用 Servlet

可以从 JavaServer 页面（JSP）文件中调用 Servlet。请参阅JSP技术部分。

5．在 ASP 文件中调用 Servlet

如果在 Microsoft Internet Information Server（IIS）上有遗留的 ASP 文件，并且无法将 ASP 文件移植成 JSP 文件时，可用 ASP 文件来调用 Servlet。在 WebSphere应用服务器 中的 ASP 支持包括一个用于嵌入 Servlet 的 ActiveX 控制，下面介绍ActiveX 控制AspToServlet 的方法和属性。

该方法说明如下：

(1) String ExecServletToString(String servletName)；执行 ServletName，并将其输出返回到一个字符串中。

(2) ExecServlet(String servletName)；执行 ServletName，并将其输出直接发送至 HTML 页面。

(3) String VarValue(String varName)；获得一预置变量值（其它格式）。

(4) VarValue(String varName, String newVal)；设置变量值。变量占据的总大小应小于 0.5 个千字节（Kbyte）。且仅对配置文件使用这些变量。

其属性如下：

- Boolean WriteHeaders；若该属性为真，则 Servlet 提供的标题被写入用户处。缺省值为假。
- Boolean OnTest；若该属性为真，服务器会将消息记录到生成的 HTML 页面中。缺省值为假。

下列ASP 脚本示例是以 Microsoft Visual Basic Scripting（VBScript）书写的。

```
<%
    ' Small sample asp file to show the capabilities of the servlets and the ASP GateWay ...
%>
<H1> Starting the ASP->Java Servlet demo</H1>
<%
    ' Create a Servlet gateway object and initialize it ...
    Set javaasp = Server.CreateObject("AspToServlet.AspToServlet")
    ' Setting these properties is only for the sake of demo.
    ' These are the default values ...
    javaasp.OnTest = False
    javaasp.WriteHeaders = False
    ' Add several variables ...
    javaasp.VarValue("gal") = "lag"
    javaasp.VarValue("pico")= "ocip"
    javaasp.VarValue("tal") = "lat"
    javaasp.VarValue("paz") = "zap"
    javaasp.VarValue("variable name with spaces") = "variable value with spaces"
%>
<BR>
Lets check the variables
<%
    Response.Write("variable gal = ")
    Response.Write(javaasp.VarValue("gal"))
%>
<BR>
<%
    Response.Write("variable pico = " & javaasp.VarValue("pico"))
%>

<BR>
<HR>
<%
    galout = javaasp.ExecServletToString("SnoopServlet")
    If javaasp.WriteHeaders = True Then
%>
```

```
Headers were written <%
    Else
%>
Headers were not written <%
    End If
    Response.Write(galout)
%>
<H1> The End ...</H1>
```

第五章 JSP 技术

5.1 JSP 简介

JSP（JavaServer Pages）是一种基于 Java 的脚本技术。在 JSP 的众多优点之中，其中之一是它能将 HTML 编码从 Web 页面的业务逻辑中有效地分离出来。用 JSP 访问可重用的组件，如 Servlet、JavaBean 和基于 Java 的 Web 应用程序。JSP 还支持在 Web 页面中直接嵌入 Java 代码。可用两种方法访问 JSP 文件：浏览器发送 JSP 文件请求、发送至 Servlet 的请求。

1. JSP 文件访问 Bean 或其它能将生成的动态内容发送到浏览器的组件。图 5-1 说明了该 JSP 访问模型。当 Web 服务器接收到一个 JSP 文件请求时，服务器将请求发送至 WebSphere 应用服务器。WebSphere 应用服务器 对 JSP 文件进行语法分析并生成 Java 源文件（被编译和执行为 Servlet）。Java 源文件的生成和编译仅在初次调用 Servlet 时发生，除非已经更新了原始的 JSP 文件。在这种情况下，WebSphere 应用服务器 将检测所做的更新，并在执行它之前重新生成和编译 Servlet。

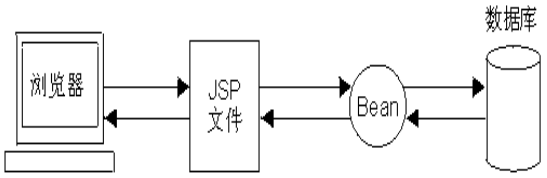


图 5-1：浏览器发送 JSP 文件请求

2. 发送至 Servlet 的请求生成动态内容，并调用 JSP 文件将内容发送到浏览器。图 5-2 说明了该访问模型。该访问模型使得将内容生成从内容显示中分离出来更为方便。WebSphere 应用服务器 支持 HttpServiceRequest 对象和 HttpServiceResponse 对象的一套新方法。这些方法允许调用的 Servlet 将一个对象放入（通常是一个 Bean）请求对象中，并将该请求传递到另一个页面（通常是一个 JSP 文件）以供显示。调用的页面从请求对象中检索 Bean，并用 JSP 来生成客户机端的 HTML。

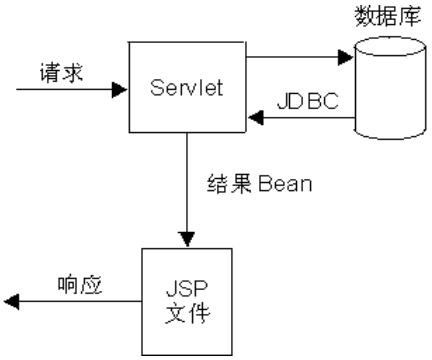


图 5-2：发送至 Servlet 的请求

5.2 JSP 示例

浏览器通过一个 Web 页面中的 HTML 表单请求一个 servlet（PopulateBeanServlet），该 servlet 创建一个名为 dataBean 的 DataBean 实例，并调用 JSP 文件将内容发送到浏览器。Servlet 示例和 JSP 文件示例说明了启用内容分离的 JSP 访问模型。

A. Servlet 是由下列 Web 页面中的 HTML 表单来调用的。

```
<HTML>
<BODY>
<H1>运行 PopulateBeanServlet</H1>
<P>您是否希望运行 PopulateBeanServlet?
<FORM action="/servlet/PopulateBeanServlet" method="GET">
<INPUT type="SUBMIT" value="Yes">
<INPUT type="SUBMIT" value="No">
</FORM>
</BODY>
</HTML>
```

B. 被请求的 servlet 为 PopulateBeanServlet， 其源代码如下：

```
/**********************************************************************
*Servlet 示例：PopulateBeanServlet.java
*这个 servlet 创建一个名为 dataBean 的 DataBean 实例，设置 dataBean 的若干个属性，
*将 dataBean 放置在当前“请求”对象中，
*调用 JSP 文件（DisplayData.jsp）来格式化并显示 dataBean 的数据
**********************************************************************/
import java.io.*;
import java.beans.Beans;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.ejb.CreateException;

public class PopulateBeanServlet extends HttpServlet {
    public void Service(HttpServletRequest req, HttpServletResponse res)
          throws ServletException, IOException {
    try {
            dataBean = (DataBean) Beans.instantiate(this.getClass().getClassLoader(), "DataBean");
        }
    catch (Exception ex) {
            throw new ServletException("Can't create BEAN of class DataBean: "
                  metaData.setSQL(getSQLString());
    }
    // Set some Bean properties (content generation)
    dataBean.setProp1("Value1");
    dataBean.setProp2("Value2");
    dataBean.setProp3("Value3");
```

```
    // To send the Bean to a JSP file for content formatting and display
    // 1) 将 dataBean 放置在当前“请求”对象中，
    ((com.sun.server.http.HttpServiceRequest) req).setAttribute("dataBean", dataBean);
    // 2) 使用 callPage 方法调用 JSP 文件，文件名为 DisplayData.jsp，并把请求对象传递给 JSP。
    ((com.sun.server.http.HttpServiceResponse) res).callPage("/DisplayData.jsp", req);
  } //end of service mehtod
} /* end of class PopulateBeanServlet */
```

C. 被调用的 JSP 文件为 DisplayData.jsp，其内容如下：

```
<!-- 该 JSP 文件获得在请求对象中传递的 dataBean，并显示该 Bean 的属性。 -->
<html>
<head>
<title>Bean Data Display</title>
</head>

<!-- Get the Bean using the BEAN tag     -->
<bean name="dataBean" type="DataBean" introspect="no" create="no" scope="request">
</bean>
<body>
<!-- There are three ways to access Bean properties -->
<!--       Using a JSP scriptlet    -->
<% out.println("The value of Bean property 1 is " + dataBeans.getProp1());
%>

<!-- Using a JSP expression -->
<p>The value of Bean property 2 is
<%= dataBean.getProp2()    %> </p>

<!--Using the INSERT tag     -->
<p>The value of Bean property 3 is
<insert bean=dataBean property=prop3 default="No property value" >
</insert></p>

</body>
</html>
```

5.3 JSP 语法

JSP 文件（扩展名为 .jsp）可以包含指令（或称为指示语句）、Class-wide 变量和方法、直接插入的 Java 代码（scriptlet）、访问 JavaBean、变量数据的 HTML 模型、变量数据的 Java 表达式的组合。

1．JSP 指令

使用 JSP 指令（在<%@ 和 %>内的）来指定所使用的脚本语言、Servlet 实现的接口、Servlet 扩展的类、Servlet 导入的软件包。JSP 指令的一般语法形式为：<%@ 指令名 ="值" %>。其中有效的指令名为：

(1) language：文件中所使用的脚本语言。此时对于 Java 程序设计语言来说，仅 java 为有效值和缺省值。该指令作用于整个文件。当多次使用该指令时，只有第一次使用是有效的。示例如下：<%@ language

="java" %>。

(2) Method：由嵌入的 Java 代码（scriptlet）生成的方法的名称。生成的代码会成为指定方法名的主体。缺省的方法是 service。当多次使用该指令时，只有第一次使用是有效的。示例如下：<%@ method ="doPost" %>。

(3) Import：Servlet 导入的 Java 语言软件包名或类名列表，该列表是用逗号分隔的。在 JSP 文件中，可以多次指定该指令来导入不同的软件包。示例如下：<%@ import ="java.io.*,java.util.Hashtable" %>。

(4) content_type ：生成的响应的 MIME 类型。缺省值为 text/html。当多次使用该指令时，只有第一次使用是有效的。 该指令可用以指定在其中对页面进行编码的字符集。示例如下：<%@ content_type ="text/html; charset=gb2312" %>。

(5) mplements :用于生成 Servlet 实现的 Java 语言接口列表,该列表是用逗号分隔的。可以在一个 JSP 文件中多次使用该命令，以实现不同的接口。示例如下：<%@ implements = "javax.servlet.http.HttpSessionContext" %>。

(6) extends：Servlet 扩展的 Java 语言类的名称。该类必须是有效的，且不能是一个 Servlet 类。该指令作用于整个 JSP 文件。当多次使用该指令时，只有第一次使用是有效的。示例如下：<%@ extends ="javax.servlet.http.HttpServlet" %>。

2．class-wide 变量和方法

使用 <SCRIPT> 和 </SCRIPT> 标记来说明 Servlet 类的 class-wide 变量和 class-wide 方法。一般语法如下：

```
<script runat=server>。
 // class-wide 变量和方法的代码
 </script>
```

属性 runat=server 是必需的，它表明该标记是用于服务器端处理的。指定 class-wide 变量和方法的示例如下：

```
<script runat=server>
    // class-wide 变量
init i = 0;
String foo = "Hello";
    // class-wide 方法
private void foo() {
    // 该方法的代码
}
</script>
```

3．访问 JavaBean

JSP 对 JavaBean 的支持使您能通过 Web 站点重复使用组件。JavaBean 可以是类文件或串行化 Bean，也可由 Servlet 动态生成。请使用 <BEAN> 标记来创建一个 Bean 实例，以使从 JSP 文件的任何地方都可对该实例进行访问。标记 <BEAN> 的语法为：

```
<bean name="Bean_name" varname="local_Bean_name"
    type ="class_or_interface_name" introspect="yes|no"
    beanName="ser_filename" create="yes|no"
    scope="request|session|userprofile" >
    <param property_name="value">
</bean>
```

其中的属性及其含义如下：

(1) name：用于在相应的范围（由 scope 属性指定）内查找 Bean 的名称。例如，这可能是用于存储 Bean 的会话（session）关键值。该值是分大小写的。

(2) varname：用于在 JSP 文件中的任何地方引用 Bean 的名称。该属性是可选的。缺省值为 name 属性值。该值是分大小写的。

(3) type：Bean 的类文件名。该名称用于以代码说明 Bean 实例。缺省值为 Object 类型。该值是分大小写的。

(4) Introspect：当该值为真时，JSP 处理器检查将所有的请求属性，并调用与请求属性匹配的 set 属性方法集（该属性是在 BeanInfo 中传递的）。该属性的缺省值为是。

(5) BeanName：Bean 的 .class 文件名、Bean 软件包名或包含 Bean 的串行化文件（.ser 文件）的文件名。（该名称是给 Bean 实例程序的）。仅当 Bean 不在指定的范围中，且创建属性被设置成是时，才使用该属性。该值是分大小写的。文件路径必须在应用服务器的 Java 类路径中指定，除非文件在 applicationserver_root\servlets 目录下。

(6) Create：当该值为真时，如果处理器在指定的范围内没有找到 Bean，则 JSP 将创建一个 Bean 实例。缺省值为真。

(7)Scope：Bean 的生命期。该属性是可选的，其缺省值为 request 。有效值为：

● request - 由 Servlet 将 Bean 设置成请求的上下文，该 Servlet 使用 JSP API 中描述的 API 调用 JSP 文件。如果 Bean 不是请求上下文的一部分，则将创建该 Bean，并将其存储在请求上下文中，除非创建属性已设置为否。

● session - 如果当前对话中存在 Bean，则 Bean 已被重用。如果不存在 Bean，且创建属性设置为是，则它已被创建并作为会话的一部分存储起来。

● userprofile - 由 Servlet 请求对象检索、造型成指定的类型，并进行内省的用户简要表文件。（在 IBM WebShere 应用服务器中，缺省类型为 com.ibm.servlet.personalization.userprofile.UserProfile）。

(8) param：属性和值配对的列表。属性是在 Bean 中用内省自动设置的。只在实例化 Bean 时，才对属性进行设置。

除了使用 <param> 属性来设置 Bean 属性外，还可以使用其它三种方法：第一，在请求包含 Bean 的 Web 页面（JSP 文件）的 URL 时，指定查询参数。必须将内省属性设置成“是”。其示例如下：http://www.myserver.com/signon.jsp?name=jones&password=d13x，其中 Bean 属性名将设置为 jones。第二种方法，将属性指定成由 HTML <FROM> 标记提交的参数。必须将 mothod 属性设置成 post。将 action 属性设置成调用 Bean 的 JSP 文件的 URL。必须将 introspect 属性设置成“是”。其示例如下：

```
<form action="http://www.myserver.com/SearchSite" method="post">
  <input type="text" name="Search for: ">
  <input type="submit">
</form>
```

第三中方法是使用 JSP 语法设置 Bean 属性。

在指定了 <BEAN> 标记后，就可以从 JSP 文件中的任何地方访问 Bean。这里有三种访问 Bean 属性的方法：使用 JSP scriptlet、使用 JSP 表达式、使用 <INSERT> 标记（如同 变量数据的 HTML 模板语法中所说明的）。请参阅 JSP 示例（DisplayData.jsp 文件）以获取三个访问 Bean 方法的每个方法示例。

4．直接插入的 Java 代码（scriptlet）

可以在 JSP 文件的 <% 和 %> 标记间直接嵌入任何有效的 Java 语言代码。这样嵌入的代码称为 scriptlet。如果没有指定 method 指令，则生成的代码将成为 service 方法的主体。用于 Servlet 的 scriptlet 可以使用一个预定义的变量集，该变量集符合基本的 Servlet、输出和输入类：

(1) request：由 javax.servlet.http.HttpServletRequest 定义的 Servlet 请求类

(2) responses：由 javax.servlet.http.HttpServletRequest 定义的 Servlet 响应类

(3) out：由 java.io.PrintWriter 定义的输出转录程序类

(4) in：由 java.io.BufferedReader 定义的输入阅读程序类

其示例如下：

```
<%
foo = request.getParameter("Name");
```

```
out.println(foo);
%>
```

5．变量数据的 HTML 模板语法

当页面被返回给浏览器时，应用服务器 HTML 模板语法使您能将变量字段放在 HTML 页面上，并使 Servlet 和 JavaBean 可利用数据库的值动态地替换变量。该功能是 JSP 的一个 IBM 扩展，它使引用变量数据变得十分容易。该语法只能用于 JSP 文件中。HTML 模板语法包括：

- 基本 HTML 模板语法；
- 替代 HTML 模板语法；
- <REPEAT>标记。

这些标记通过 HTML 编写工具被设计成传递交互操作的标记。每一个标记有一个相应的结束标记。每一个标记是分大小写的，它们的一些属性也是分大小写的。IBM WebSphere Studio 使开发 JSP 文件以包含 HTML 模板语法变得更为容易。

(1) 基本 HTML 模板语法

<INSERT> 标记是用于指定变量字段的基本标记。一般的语法为：

```
<insert requestparm=pvalue requestattr=avalue bean=name
   property=property_name(optional_index).subproperty_name(optional_index)
   default=value_when_null>
</insert>
```

其中的属性及其含义如下：

- requestparm：要在请求对象内进行访问的参数。该属性是分大小写的，并且不能与 Bean 和 property 属性一起使用。
- Requestattr：要在请求对象内进行访问的属性。属性应使用 setAttribute 方法设置。该属性是分大小写的，并且不能与 Bean 和 property 属性一起使用。
- Bean：由 <BEAN> 标记在 JSP 文件中说明的 JavaBean 名。请参阅访问 JavaBean 以获得 <BEAN> 标记的解释。该属性的值是分大小写的。当指定了 Bean 属性，但未指定 property 属性时，在替换中将使用完整的 Bean 。例如，如果 Bean 是类型 String 且未指定属性，则将替代 string 的值。
- Property：访问替换的 Bean 的属性。该属性的值是分大小写的，并且是属性的独立场所名。该属性不能与 requestparm 属性一起使用。
- Default：当 Bean 属性值为空时，显示可选的字符串。如果字符串包含多个字，则该字符串必须包含在一对双引号中（例如 "HelpDesk number"）。该属性的值是分大小写的。如果未指定值，则当属性值为空时，用空字符串替代。

基本语法的示例如下：

```
<insert bean=userProfile property=username></insert>
<insert requestparm=company default="IBM Corporation"></insert>
<insert requestattr=ceo default="Company CEO"></insert>
<insert bean=userProfile property=lastconnectiondate.month></insert>
```

在大多数情况下，property 属性的值就是属性名。但是，可以通过指定 property 属性的全格式来访问属性的某一属性（子属性）。这个全格式也提供选择项给您来指定索引属性的一个索引。该可选的索引可以是一个常数（例如 2）或如重复设置 HTML 标记中描述的索引。使用属性标记的全格式示例如下：

```
<insert bean=staffQuery property=address(currentAddressIndex)></insert>
<insert bean=shoppingCart property=items(4).price></insert>
<insert bean=fooBean property=foo(2).bat(3).boo.far></insert>
```

(2) 替代 HTML 模板语法

HTML 标准不允许在 HTML 标记中嵌入 HTML 标记。因而，就无法在另一个 HTML 标记中嵌入 <INSERT>标记。作为代替，请使用 HTML 模板替代语法。要使用替代语法：

- 请使用<INSERT>和</INSERT>来包含 HTML 标记，在该标记中指出替代内容。

- 指定 Bean 和 property 属性：

— 要指定 Bean 和属性特性，请使用下列格式：$(bean=b property=p default=d)，其中 b、p 和 d 作为描述基本语法的值。

— 要指定 requestparm 属性，请使用下列格式：$(requestparm=r default=d)，其中 r 和 d 作为描述基本语法的值。

— 要指定 requestattr 属性，请使用下列格式：$(requestattr=r default=d)，其中 r 和 d 作为描述基本语法的值。

替代 HTML 模板语法的示例如下：

```
<insert>
  <img src=$(bean=productAds property=sale default=default.gif)>
</insert>
<insert>
  <a href="http://www.myserver.com/map/showmap.cgi?country=$(requestparm=country default=usa)
  &city$(requestparm=city default="Research Triangle Park")&email=
     $(bean=userInfo property=email)>Show map of city</a>
</insert>
```

(3) <REPEAT>标记

<REPEAT>标记的语法为：

```
<repeat index=name start=starting_index end=ending_index>
</repeat>
```

其中：

- index：是用于标识该重复数据块的一个可选的名称。该值是分大小写的。
- Start：是用于该重复数据块的一个可选的开始索引值。缺省值为 0 。
- End：是用于该重复数据块的一个可选的结束索引值。最大值是 2,147,483,647。如果结束属性的值小于开始属性的值，则忽略结束属性。

下面的示例 1、2 和 3 显示了如何使用<REPEAT>标记。如果所有的索引属性拥有 300 个或更少的元素，则这些示例将产生相同的输出。如果拥有的元素多于 300 个，示例 1 和示例 2 将显示所有的元素，而示例 3 将只显示前 300 个元素。示例 1 用缺省开始和结束索引显示了隐式索引：使用最小索引属性数的 bean 限制了循环重复的次数。

```
<table>
<repeat>
  <tr><td><insert bean=serviceLocationsQuery property=city></insert></tr></td>
  <tr><td><insert bean=serviceLocationsQuery property=address></insert></tr></td>
  <tr><td><insert bean=serviceLocationsQuery property=telephone></insert></tr></td>
</repeat>
</table>
```

示例 2 显示了索引、开始索引和结束索引：

```
<table>
<repeat index=myIndex start=0 end=2147483647>
  <tr><td><insert bean=serviceLocationsQuery property=city(myIndex)></insert></tr></td>
  <tr><td><insert bean=serviceLocationsQuery property=address(myIndex)></insert></tr></td>
  <tr><td><insert bean=serviceLocationsQuery property=telephone(myIndex)></insert></tr></td>
</repeat>
</table>
```

示例 3 用隐式开始索引显示了显式索引和结束索引。虽然指定了索引属性，仍可对索引过的属性城市进

行隐式索引，因为不需要（i）。

```
<table>
<repeat index=myIndex end=299>
   <tr><td><insert bean=serviceLocationsQuery property=city></insert></tr></td>
   <tr><td><insert bean=serviceLocationsQuery property=address(myIndex)></insert></tr></td>
   <tr><td><insert bean=serviceLocationsQuery property=telephone(myIndex)></insert></tr></td>
</repeat>
</table>
```

可以嵌套<REPEAT>数据块。独立索引每个数据块。该能力对交叉两个 bean 上的交错属性或含有子属性的属性非常有用。在示例中，将两个<REPEAT>数据块嵌套，用以显示用户购物手推车上每一张压缩光盘上的歌曲列表。

```
<repeat index=cdindex>
   <h1><insert bean=shoppingCart property=cds.title></insert></h1>
      <table>
   <repeat>
      <tr><td><insert bean=shoppingCart property=cds(cdindex).playlist></insert>
      </td></tr>
      </table>
</repeat>
</repeat>
```

6．变量数据的 Java 表达式

当处理 JSP 文件时，要指定分解一个 Java 语言表达式，请使用 JSP 表达式标记 <%= 和 %>。该表达式被评估和转换成一个字符串，并加以显示。原语类型，如 int 和 float 都自动转换成字符串表示法。在该示例中，foo 是在 <SCRIPT> 中加以说明的 class-wide 变量，示例请见 Class-wide 变量和方法：

```
<p>翻译欢迎辞<%= foo %>.</p>
```

当获得 JSP 文件时，文本为：翻译欢迎辞 Hello。

5.4 JSP API

有两种接口支持 JSP 技术。这些 API 提供了一种将内容生成（业务逻辑）从内容演示（HTML 格式）中分离出来的方法。这种分离使 Servlet 能生成内容并将它（如在 Bean 中）存储在请求的上下文中。生成上下文的 Servlet 通过将请求的上下文传递至 JSP 文件来生成一个响应，该 JSP 文件包含 HTML 格式。<BEAN> 标记便提供了对业务逻辑的访问。支持 JSP 的接口有：

- com.sun.server.http.HttpServiceRequest：该类实现了 javax.servlet.http.HttpServletRequest 接口和用于设置根据名称定义的属性的方法 setAttribute() 。
- com.sun.server.http.HttpServiceResponse：该类实现了 javax.servlet.http.HttpServletResponse 接口，并添加了一个使 Servlet 能调用 JSP 文件和可选地传递上下文的 callPage() 方法。

1．callPage() 方法

用 callPage() 方法在 Servlet 中对 JSP 进行服务。所服务的页面（一个 JSP 文件）作为响应返回到浏览器中。调用 Servlet 还可以通过请求对象传递某些上下文。应该对所服务的页面标题进行编码，以将用于告诉浏览器不要对文件执行高速缓存的指令包含在内。callPage() 方法的语法如下：

```
public void callPage(String fileName, HttpServletRequest req) throws ServletException, IOException
```

其中：

- fileName：用于标识文件（该文件用于生成输出并表示内容）的 URL 名。如果文件名以斜杠（/）开头，则可以假设文件位置与文档根目录有关。如果文件名不是以斜杠开头，则可假设文件位置与当前调用的请求有关。callPage() 方法不支持调用文件扩展名为 .html 的页面。如果需要使用 callPage() 方法来

调用 HTML 页面，则必须首先将 HTML 文件重命名成扩展名为 .jsp 的文件。

● Req：调用该方法的是 Servlet 的 HttpServletRequest 对象。最通常的是，将上下文作为 Bean，在请求对象的上下文中进行传递。

要使用 callPage() 方法，必须将响应对象造型成 com.sun.server.http.HttpServiceResponse 类型。

2．使用 setAttribute() 方法

用 setAttribute() 方法来将一个特性存储在请求的上下文中。语法是：public void setAttribute(String key, Object o),其中，key 代表存储的特性名，而 o 表示用 key 来存储的上下文对象。要使用 setAttribute() 方法，必须将请求对象造型成 com.sun.server.http.HttpServiceRequest 类型。

第六章 WebSphere 应用服务器

IBM WebSphere 应用服务器（简称 WebSphere 应用服务器）是一个 Web 应用服务器,它提供了增强的 Servlet API 和 Servlets 管理工具，并集成了 JSP 技术和数据库连接技术。

6.1 基本特性

WebSphere 应用服务器使您能够为 Servlet 的开发实现 “一次写成，各处使用 ”的目的。该产品包括一个基于 Java 的 Servlet 引擎，独立于 Web 服务器和它所基于的操作系统。WebSphere 应用服务器提供了服务器插件的选项，与大多数流行的应用程序设计接口（API）兼容。所支持的 Web 服务器有:

- IBM HTTP 服务器
- Apache Server
- Domino
- Lotus Domino Go Webserver
- Netscape Enterprise Server
- Netscape FastTrack Server
- Microsoft Internet Information Server

除了 Servlet 引擎及插件外，WebSphere 应用服务器还提供：

1. JavaSoft Java Servlet API 实现，以及这些 API 的扩展和附加。

2. 示例和文档，示例 Servlet 和 Web 站点应用程序演示了简单和高级技术。联机和可打印文档供您快速入门，并帮助掌握服务器高级功能的使用演示基本类及其扩展的示例应用程序。

3. WebSphere 应用服务器的管理器，使用图形界面，易于:

(1) 为装入本地和远程的 Servlet 设置选项
(2) 设置初始化参数
(3) 管理 Servlet
(4) 指定 Servlet 别名
(5) 创建 Servlet 链和过滤
(6) 启用轻量级目录访问协议（LDAP）目录支持
(7) 记录 Servlet 消息
(8) 启用 JVM 调试
(9) 监控 WebSphere 应用服务器 使用的资源
(10) 监控已装入 Servlet、活动 Servlet 会话和 JDBC 连接
(11) 控错误、事件、异常情况和日志输出
(12) 创建转储和数据快照
(13) 动态地启用和禁用跟踪

4. 缓存和再使用与 JDBC-从属数据库连接的连接管理功能。当一个 Servlet 需要数据库连接时，它

可从可用连接的缓冲池获得，从而消除了为每个请求打开一个新连接的所需花费的系统开销。

5. 附加的 Java 类，遵循 JavaBeans 规范，允许程序员访问 JDBC-从属数据库。当隐藏使用关系数据库的复杂度时，这些数据访问 Bean 可提供增强功能。它们可用于可视集成开发环境中。

6. JSP 的动态页面内容支持。JSP 技术使您能够通过服务器端脚本生成动态 Web 页面。其结果是将呈示逻辑（例如，定义 Web 站点结构和外观的 HTML 代码）从业务逻辑（例如，访问数据库以获得要显示在 Web 站点上的信息的 Java 编码信息）中分隔出来。灵活性方面，JSP 文件可包括任何直接插入的 Java 的组合：<SERVLET> 标记、NCSA 标记和 JavaBean。

7. 启用 LDAP（轻量级目录访问协议）支持的目录服务。

8. 用于集成 WebSphere 应用服务器和 Web 服务器（如 Apache Server） 到 Tivoli 管理环境（Tivoli Management Environment）以获得分布式监控和操作的模块和命令行界面。Tivoli 模块并不与 WebSphere 应用服务器一起发行，请访问 Web 站点（http://www.software.ibm.com/webservers/appserv/）以获得更多信息。

WebSphere 应用服务器对 JSP 的支持是通过 JSP 处理器来实现的。在 Web 服务器上安装 WebSphere 应用服务器时，Web 服务器的配置被设置成将对 JSP 文件（即文件扩展名为 .jsp）的 HTTP 请求传递至 WebSphere 应用服务器。WebSphere 应用服务器配置则被设置成将 JSP 文件传递至其 JSP 处理器（pageCompile）。

JSP 处理器对每一个 JSP 文件创建并编译 Servlet。该处理器还为每个 JSP 文件产生两个文件：

(1) java 文件：包含可用于 Servlet 的 Java 语言代码；

(2) class 文件：编译过的 Servlet。

JSP 处理器把 .java 和 .class 文件放在目录 servlets\pagecompile\JSP_file_path 下，其中 JSP_file_path 是 JSP 源文件所在的目录路径。例如，若 JSP 文件在 WebSphere\AppServer\samples\Web Bank 目录下，则.java 和.class 文件就在 WebSphere\AppServer\servlets\pagecompile\WebSphere\App Server\samples\WebBank 目录下。.java 和 .class 文件的文件名相同。处理器使用了命名约定，该命名约定包括将下划线字符和后缀添加到 JSP 文件名中。例如，如果 JSP 文件名是 login.jsp，则生成的文件为 _login_xjsp.java 和 _login_xjsp.class。

如同所有的 Servlet，由 JSP 文件生成的 Servlet 是 javax.servlet.http.HttpServlet 的子类或子孙类。如果 Servlet 类是软件包的一部分，则 Servlet Java 代码包含了用于一些必需类和软件包语句的导入语句。如果 JSP 文件包含 JSP 语法（例如指令和 scriptlets），则 JSP 处理器会将 JSP 语法转换成等价的 Java 代码。如果 JSP 文件包含 HTML 标记，则处理器添加 Java 代码，以使 Servlet 能一个接一个字符地输出 HTML。

6.2 安装与配置

对于任何软件，都需要一些计划和具体步骤以确保成功安装。对于安装与配制 WebSphere 应用服务器及其组件也是如此。下面介绍在 Windows NT 上安装与配置 WebSphere 应用服务器

1．内存和软件要求

(1) 内存：至少 128 MB RAM，建议 256 MB。WebSphere 应用服务器安装光盘包括 Java 开发工具（JDK）。

(2) 操作系统：Microsoft NT 版本 4.0 与服务包 3 。

(3) Web 服务器：WebSphere 应用服务器需要下列服务器之一。

- IBM HTTP 服务器版本 1.3.3 Windows NT 版（WebSphere 应用服务器安装光盘包括 IBM HTTP 服务器）
- Apache Server 版本 1.3.2 Windows NT 版
- Domino 版本 5.0 Windows NT 版

- Lotus Domino Go Webserver 版本 4.6.2.5 Windows NT 版
- Microsoft Internet Information Server 版本 3.x 和 版本 4.0 Windows NT 版
- Netscape Enterprise Server 版本 3.01 和 版本 3.51 Windows NT 版（建议使用版本 3.5.1）
- Netscape FastTrack Server 版本 3.01 Windows NT 版

(4) Java 开发组件（JDK）：建议使用与 WebSphere 应用服务器一起提供的 JDK1.1.6。

(5) Java servlet API：WebSphere 应用服务器包括 JSDK 版本 2.0 或更高版本。

(6) Web 浏览器：WebSphere 应用服务器的管理器是用来管理 servlet 的用户界面。要运行管理器，需要 appletviewer 或支持 JDK1.1 的浏览器，例如：

- Netscape Navigator 4.06 或 4.0.7，包括集成的 JDK 1.1 支持并由以下 Web 站点提供：http://developer.netscape.com/software/jdk/download.html。
- 或带有修正包的 Microsoft Internet Explorer 4.01 或更高版本。
- Sun HotJava 1.1 或更高版本

一些旧的浏览器不能够正确地处理启用本机语言支持的文本。如果在用户界面上看到外来字符，例如“ sEnable”，而不是“ Enable”，可以通过升级浏览器校正。

2．安装 Web 服务器

如果计划安装 IBM HTTP 服务器，请在安装 WebSphere 应用服务器之前安装它。可以从 WebSphere 应用服务器安装光盘安装 IBM HTTP 服务器。安装 WebSphere 应用服务器会更改 Web 服务器 httpd.conf 文件。如果安装 Web 服务器是在安装 WebSphere 应用服务器之后安装的，将不会进行更改，并且 WebSphere 应用服务器将不能够正确运行。安装 IBM HTTP 服务器之后，需要一些配置以启用 SSL 支持。请参阅 IBM HTTP 服务器文档以获得指示信息。

对于计划安装其它的 Web 服务器（如 Apache Server），也请在安装 WebSphere 应用服务器之前安装它。理由同上。

3．安装 WebSphere 应用服务器之前

安装 WebSphere 应用服务器版本 2.0 之前，请卸装所有以前的版本。卸装之前，备份 WebSphere 应用服务器版本 1.x 文件。

(1) 文件备份

从 Windows NT 卸装以前版本的 WebSphere 应用服务器之前，确保要移植的文件已经或者将要保存。安装 WebSphere 应用服务器版本 2.0 时显示出的图形用户界面备份 WebSphere 应用服务器目录中的文件，包括类、领域、Servlet、属性文件，其中，属性文件包括 servlet.properties、admin_port.properties、rules.properties、jvm.properties、aliases.properties、connmgr.properties、userprofile.properties。如果有文件驻留在这四个目录之外(例如,如果在 WebSphere 应用服务器 安装中创建自己的目录),请在安装 WebSphere 应用服务器 版本 2.0. 之前，在 WebSphere 应用服务器 安装之外的位置备份文件。

此处包括移植进程的第一部分。第二部分必须在安装 WebSphere 应用服务器版本 2.0 之后执行。请参阅有关文档以获得详细信息。

(2) 卸载前一版本

对于 Windows NT，使用开始 --> 控制面板中的添加/删除选项，或从开始 --> 程序 --> IBM WebSphere --> WebSphere 应用服务器版本 1.x 卸装。

注意：当已安装了某版本的 WebSphere 应用服务器，它将复制 Web 站点配置文件作为备份文件，然后修改原始配置文件。当使用 Web 服务器时这个已被修改过的文件就成为活动的配置文件。当卸装 WebSphere 应用服务器时，不会恢复以前的配置文件，它仍然是备份文件。为了使这些设置再次活动，必须将它们从备份文件转换为活动的 Web 服务器配置文件。

另外，需要清除 CLASSPATH。如果在 CLASSPATH 中已经有来自以前版本的 WebSphere 应用服务器的信息，安装版本 2.0 之前请删除这些信息。对于在 Windows NT 上使用 Go Webserver 的 Web 服务器来说，当安装 WebSphere 应用服务器时能自动卸装 Go Webserver 上的 Java 支持。其它的请查阅有关文档。

4．安装 WebSphere 应用服务器

在即将安装 WebSphere 应用服务器之前，请确保已经：

(1) 备份所有未通过安装程序自动备份的文件（从以前的 WebSphere 应用服务器进行安装）。

(2) 安装您计划使用的 Web 服务器。Web 服务器必须在安装 WebSphere 应用服务器之前安装。

(3) 如果 Web 服务器正在运行，请停止它。

注意：在 WebSphere 应用服务器的安装期间，如果指定使用 IBM HTTP 服务器或 Apache Server，将提示您确认 Web 服务器 httpd.conf 文件的位置。

安装在 Windows NT 上，插入 WebSphere 应用服务器安装光盘，转至以 Windows NT 操作系统命名的子目录，运行可执行安装程序（setup.exe）。一系列面板将指导您完成安装。

5．配置 WebSphere 应用服务器

下面介绍启用 WebSphere 应用服务器和它的组件以使之协同工作的必要配置。完成这些任务之后，WebSphere 应用服务器便能够主要通过使用缺省设置运行所有功能。

(1) 配置 Apache Server

如果使用 Apache Server 作为 Web 服务器，请确保 httpd.conf 文件包含此行：AddModule mod_app_server.c。

(2) 使用数据库。

要确保 WebSphere 应用服务器 维护和使用与关系数据库（如 Oracle 或 DB2）的连接，请添加数据库.zip 文件到位于文件 <as_root >/properties/bootstrap.properties 中的 java.classpath 属性，或使用 WebSphere 应用服务器的管理器界面的 Java 引擎页面来指定文件。同样，确保 java.classpath 包含用于数据库连接的有效的驱动程序。请查看产品 Web 站点上的 WebSphere 应用服务器 版本 2.0 自述文件可得到附加的技巧。

(3) 运行模式

ose.mode 属性控制 WebSphere 应用服务器是作为 Web 服务器的一部分（进程内），还是在独立模式下运行（进程外）。该属性位于 <as_root> /properties/bootstrap.properties 文件中。对于所有的服务器，ose.mode 缺省值是 out。如果使用 Apache Server 或 IBM HTTP 服务器，必须设置 ose.mode 属性为 out，这意味着 WebSphere 应用服务器在独立模式下运行。对于其它 Web 服务器，可以（但不建议）更改 ose.mode 为 in 并且作为 Web 服务器的一部分运行 WebSphere 应用服务器。 无论何时如有必要请复位 ose.mode。例如，如果从作为 Webserver 一部分运行的 Netscape Enterprise Server （ose.mode=in）转换为 IBM HTTP 服务器（要求 ose.mode=out），不要忘记在运行 IBM HTTP 服务器之前，将 ose.mode 属性更改为 out。

与手工编辑包含 ose.mode 属性的 bootstrap.properties 文件相对比，管理器界面的 Java 引擎页面提供了一个简单的方法来锁住该属性值。 Java 引擎页面提供了一个可用来指示是以 Web 服务器的一部分（ose.mode=in）或以独立模式（ose.mode=out）运行 WebSphere 应用服务器的单选按钮。请参阅有关文档学习如何访问管理器。

ose.mode 的值影响。作为 Web 服务器一部分运行 WebSphere 应用服务器为 Servlet 和其它应用程序提供较高的性能，但安全性较差。作为 Web 服务器的一部分运行 WebSphere 应用服务器，允许当关闭 Web 服务器时 WebSphere 应用服务器自动停止。在独立模式下运行 WebSphere 应用服务器需要其它步骤。 请参有关文档获得更多的详细信息。

6．启动和停止 WebSphere 应用服务器

当启动 Web 服务器时 WebSphere 应用服务器自动启动。WebSphere 应用服务器的管理器，是通过 Web 浏览器访问的，为查看和更改 WebSphere 应用服务器的设置和性能提供了界面。请参阅定制配置设置。

如果将 WebSphere 应用服务器 作为 Web 服务器的一部分运行，当关闭 Web 服务器时 WebSphere 应用服务器 将自动停止。如果以独立模式运行，WebSphere 应用服务器不会自动停止。如果使用 Apache Server 或 IBM HTTP 作为 Web 服务器， WebSphere 应用服务器必须在独立模式下运行。

当在 Windows NT 上进程外运行 WebSphere 应用服务器 时，停止 Web 服务器之后，请停止 WebSphere Servlet 服务以停止 WebSphere 应用服务器。从开始 --> 设置 --> 控制面板 --> 中选择 WebSphere Servlet 服务，并按“停止”按钮。

7．安装的检查和故障寻找

要验证 WebSphere 应用服务器已安装好并正确配置，可调用 WebSphere 应用服务器提供的 snoop servlet。使用 Web 浏览器打开 servlet URL: http://your.server.name/servlet/snoop。Snoop Servlet 应回送客户机发送的 HTTP 请求及 servlet 的初始化参数。SnoopServlet 和其它 servlet 的代码位于<as_root>/servlets 目录。如果 Servlet 失败，请尝试下列步骤:

(1) 如果已通过手工编辑 .properties 文件或使用 WebSphere 应用服务器的管理器界面更改了任何 WebSphere 应用服务器的配置设置，请检查这些文件以确保未引入任何非法的或不正确的值。特别要检查 <as_root> /properties/bootstrap.properties 文件。文档中心包含关于手工配置这些属性的资料和关于每一配置的可接受性及缺省值的讨论资料。

(2) 为 Web 服务器打开本地日志和跟踪。找到 WebSphere 应用服务器 bootstrap.properties 文件（在 <as_root> /properties 目录中）。设置 ose.trace.enabled 属性为 true，设置 ose.trace.to.webserver 属性为 true。停止 Web 服务器并重启动。记住如果 WebSphere 应用服务器运行在独立模式，当停止 Web 服务器时它不会相应停止。检查 Web 服务器出错日志及 WebSphere 应用服务器 <as_root>/logs 目录下的日志以查看错误。

(3) 启用调试控制台并重新启动 Web 服务器。WebSphere 应用服务器的调试控制台为收集和查看跟踪及监控数据提供了集中场所。例如，从调试控制台，可以作为一组启动和停止列在收集和监控服务器数据中的监控程序。调试控制台的服务器控制台标签允许查看 servlet 的 stdout 和 stderr 流。缺省情况下，调试控制台未启用。启用该控制台。在 WebSphere 应用服务器 debug.properties 文件中设置 debug.server.console.enabled 属性为 true 并重新启动 Web 服务器使得改动生效。或者，运行 http://your.server.name/servlet/DebugConsoleServlet，启用调试控制台。在 Windows NT 上，要成功地查看调试控制台，必须配置 Windows NT 以允许一个或多个服务与 Windows 桌面交互。如果使用作为 Windows NT 服务运行的 Web 服务器：

- 选择开始 --> 设置 --> 控制面板 --> 服务。
- 选择 Web 服务器相应的服务。
- 单击启动按钮。
- 在结果对话框中，选择允许服务与桌面交互的复选框。
- 重新启动 Web 服务器以使更改生效。

对于 Microsoft Internet Information Server，对与 Web 服务器相关的每个服务（如 Web 发布和 FTP 服务），执行以上过程。这些服务必须允许与 Windows NT 桌面交互。如果 WebSphere 应用服务器运行时未启动任何 Web 服务器相关的进程，则为 WebSphere Servlet 服务执行以上过程，允许服务与桌面交互。

8．从 版本 1.x 移植到版本 2.0

在安装 WebSphere 应用服务器之后，通过下列步骤完成移植：

(1) 请检查用户 Servlet、领域、类、JavaServer 页面（JSP）和其它在第一部分的移植过程中保存的与编程相关的文件已经被安放在它们所属的 WebSphere 应用服务器 版本 2.0 目录中。安装程序应已为您完成了这些步骤。如果已备份了您创建在 WebSphere 应用服务器 安装中用户文件或目录，请将它们安放在现在的新安装处。

(2) 传输 jvm.properties 文件设置到 WebSphere 应用服务器版本 2.0 bootstrap.properties 文件。

- 对于 WebSphere 应用服务器 类路径，仅移植用户指定的 类路径部分到 bootstrap.properties 中 – 不要传输 JVM 库或 WebSphere 应用服务器 版本 1.x .jar 文件路径，因为这些是 WebSphere 应用服务器版本 1.x 缺省配置的一部分，而您未曾自行设置过它们。换言之，只传输您添加的部分类路径。不要除去 WebSphere 应用服务器 版本 2.0 缺省类路径；简单地加入 版本 1.x 项目即可。
- 对于 Java 库路径和路径，也只移动用户指定的部分。不要除去 WebSphere 应用服务器 版本 2.0 缺省的库路径和路径设置；简单地加入 版本 1.x 项目即可。
- 对于其它属性，仅当它们有定义在 bootstrap.properties 中的对应者时才移植。

要使用 WebSphere 应用服务器的管理器来管理在安装 WebSphere 应用服务器之前已存在于 Web 服务器中的 Servlet，必须首先移植这些 Servlet。为了移植现有的 Servlet，将 Servlet 从它们目前的位置移

动到 <as_root> \servlets 目录。WebSphere 应用服务器 监控该目录并且当 Servlet 更改时自动重新装入 Servlet。如果有 Servlet 在其它目录中并且不想将它们移动到 <as_root> \servlets 目录，可使用管理器界面中 Java 引擎页面的“ 可重装 Servlet 类路径 ”字段来指定其余要监控的目录。使用 WebSphere 应用服务器中的配置页面来重新配置以前的 Servlet 参数。请参阅定制配置设置。

6.3 定制配置

本节介绍如何启动和使用 WebSphere 应用服务器的管理器（一个图形界面）为 Servlet 活动和 WebSphere 应用服务器的组件定制基本设置参数。

1．启动 WebSphere 应用服务器的管理器

要启动 WebSphere 应用服务器的管理器，在 Web 浏览器中输入 URL：http://your.server.name:9527/。其中 your.server.name 是主机的全限定名。注意：如果在安装了 WebSphere 应用服务器的同一机器的浏览器中启动 WebSphere 应用服务器的管理器，使用 http://localhost:9527/ 以获得更高性能。

管理器启动并显示登录页面。若是首次登录至管理器，请使用 admin 作为登录用户标识符和口令。单击“ 确定 ”。为安全起见，应该更改登录口令。要运行管理器，需要支持 Java 开发组件（JDK）1.1.6 的 appletviewer 或浏览器。请参阅准备安装 WebSphere 应用服务器中的软件要求，查看选项。

2．使用 WebSphere 应用服务器的管理器

管理器左边的浏览区域允许: 为不同 WebSphere 应用服务器 组件定制设置、配置 Servlet 和设置别名及过滤、建立和维护安全性、收集和监控 WebSphere 应用服务器、连接和 Servlet 数据。

(1) 定制基本属性

在使用 WebSphere 应用服务器管理 Servlet 之前，请为 Servlet 活动和 WebSphere 应用服务器的组件属性配置基本设置参数。在管理器浏览区域，单击“ 设置 ”显示可定制不同设置的项目页面。

- 管理页面：更改登录至管理器的用户标识符和口令，并为管理器指定新的端口号。
- 连接管理页面：设置连接缓冲池，从而减少用于维护与数据服务器（如 IBM DB2 关系数据库）的连接所花费的资源。
- 目录管理页面：为目录服务器指定设置，允许从 Web 服务器、操作系统、WebSphere 应用服务器 及其它软件产品的中央位置管理安全性数据。
- Java 引擎页面：指定 Java 编译器设置并指出 WebSphere 应用服务器 是作为 Web 服务器一部分运行以获得较高性能，还是作为独立模式运行以获得更好安全性。
- 会话跟踪页面：指定维护用户会话（来自于相同浏览器的相关用户请求系列）的状态信息的设置。
- 用户简要表文件页面：指定维护关于 Web 站点访问者永久信息的设置。
- 虚拟主机页面：指定 Servlet 的替代路径，允许 Web 服务器根据 Servlet 请求期间客户机指定的不同域提供不同文档。

(2) 配置 Servlet、别名和过滤

放置在<as_root>\servlets 目录下的 servlet 在请求时自动装入和重新装入（如果更新过）。也可以使用 WebSphere 应用服务器 管理器，通过初始化参数和创建 Servlet 别名和过滤更为直接地管理 Servlet。要管理 Servlet，单击管理浏览区域中的 Servlet 并选择页面：

- 配置页面：为个别 Servlet 定义配置信息和初始化参数，如关联的类文件，是否在启动时装入 Servlet 以及 Web 服务器是否从远程装入 servlet。
- 别名页面：指定路径映射规则，允许用户输入快捷 URL 来调用特定 servlet。
- 过滤页面：联系 Servlet 和 MIME-类型，从而每当生成一个特定的 MIME-类型响应时，调用一个特殊的 Servlet。

(3) 维护安全性

通过定义用户、组、资源和存取控制表建立和维护安全性。通过为每个用户、组和资源指定特定访问设置，可精确地控制如何使用服务的资源，及由谁使用。单击管理器浏览区域中的安全性显示这些页面:：

- 用户页面：指定允许谁访问由 WebSphere 应用服务器 提供的 Web 页面及其它资源，如 servlet。

- 组页面：将用户与命名列表相关联，允许同时对整个组赋予访问权限。
- 存取控制表页面：为用户和组指定访问许可。
- 资源页面：为特定目录、文件和 WebSphere 应用服务器 上的 servlet 指定安全性参数。

(4) 收集和监控服务器数据

通过查看日志文件监控 Servlet 活动、已装入的 Servlet 的状态和资源的实时使用。单击监视器浏览区域中的“ 服务器执行分析 ”显示页面列表：

- JVM 调试页面：启用 JVM 调试和指定调试设置。
- 事件页面（在日志文件下）：监控事件日志中收集的信息 。
- 出错页面（在日志文件下）：监控出错日志中收集的信息。
- 活动会话页面（在监控下）：监控关于 Web 服务器上当前活动的用户会话的信息，包括关于个别会话的信息和所有活动会话的摘要信息。
- 数据库缓冲池连接页面（在监控下）：监控连接缓冲池信息，包括关于缓冲池和个别连接的统计数字。
- 转储面板页面（在监控下）：指定何时及在何处创建 Servlet 转储和活动快照。
- 异常情况状态（在监控下）：监控来自于 Java 引擎和 servlet 的异常数据 。
- 已装入 Servlet 页面（在监控下）：监控个别 servlet 的状态和统计数字。
- 日志输出页面（在监控下）：查看事件或出错日志输出的记录。
- 资源使用页面（在监控下）：监控服务资源如何被使用，包括内存、请求句柄对象缓冲池、服务请求和服务响应时间。
- 跟踪页面：监控跟踪数据，通常由 IBM 服务人员提出请求。
- 注销：要快速注销，单击管理器浏览区域中的“ 注销 ”，返回管理器登录屏幕。

6.4 部署 Servlet

在 WebSphere 应用服务器上部署 Servlet 需要四个步骤：编译 Servlet 或 Web 应用程序、将类文件放到 WebSphere 应用服务器上、将相关的 HTML、JSP 和 SHTML 文件放到 WebSphere 应用服务器上、用 WebSphere 应用服务器的管理器来配置初始化参数，并设置其它选项。

1．编译 Servlet 和 Web 应用程序

编译 Servlet，有下列注意事项：

(1) 确保系统 CLASSPATH 环境变量中包括 JDK classes.zip 文件和相应的 WebSphere 应用服务器 JAR 文件。 lib 目录中有几个 WebSphere 应用服务器 JAR 文件。根据所需导入的类的不同，可能还需要将那些未列在该示例中的 JAR 文件包括在内：

如果使用的是 Windows NT，请分别输入下列命令（在同一行中）：

```
set CLASSPATH=.;JAVA_HOME\lib\classes.zip;
applicationserver_root\lib\ibmwebas.jar;
applicationserver_root\lib\jst.jar;applicationserver_root\lib\jsdk.jar;
applicationserver_root\lib\xml4j.jar;
applicationserver_root\lib\databeans.jar;%CLASSPATH%
```

(2) 用下列命令将 PATH 环境变量设置成包括 java/bin 目录的变量：

对于 Windows NT，该命令为：set PATH=JAVA_HOME\bin;%PATH%

(3) 通过发出下列命令来测试相应的 Java Development Kit (JDK) 是否在路径中：

```
java -version
```

该命令应该返回 JDK 版本的状态消息。

(4) 通过发出下列命令来编译 Servlet

```
javac filename.java
```

2．将类文件放到 WebSphere 应用服务器上

缺省情况下，WebSphere 应用服务器在 Servlet 根目录 applicationserver_root\servlets 下查找 Servlet 类文件。请将编译过的 Servlet 类文件复制到该目录下。要从替代 Servlet 目录中装入 Servlet，请配置可重装 Servlet 目录。要从远程系统中装入 Servlet，请在用 WebSphere 应用服务器的管理器配置 Servlet 时，指定该远程系统。

(1) 如果 Servlet 在软件包中，则将软件包结构镜像成 servlet\ 或可重装 Servlet 目录下的子目录。例如，如果 Servlet SignonServlet.class 和 AccountBean.class 在名为 WebBank 中的软件包中，请将 Servlet 放在目录 servlet\WebBank 下。

(2) 如果 Servlet 导入您所开发的非 Servlet 类，建议将那些类复制到 applicationserver_root\servlets 下。

根据 jvm.properties 文件中的设置，将决定所有的 Servlet 标准输出是到 applicationserver_root\logs\ncf.log 文件还是 Java 控制台窗口。请参阅 启用 Java 控制台中的有关的说明。

3．将 HTML、JSP 和 SHMTL 文件放到 WebSphere 应用服务器上

将与 Servlet 有关的 HTML、JSP 和 SHMTL 文件复制到 Web 服务器的 HTML 文档根目录 server_root\HTML_directory 下。该目录是由特定服务器配置（传递、别名和虚拟主机规则的设置）所决定的。

4．配置 Servlet

若要从远程系统上的 JAR 或 SER 文件装入 Servlet，或设置初始化参数，请使用 WebSphere 应用服务器的管理器来配置 Servlet 或使用 XML Servlet 配置。

6.5 连接管理器

连接管理器使您可以控制并减少由 Web 应用程序使用的资源。相对于非 Web 应用程序，基于 Web 的应用程序对数据服务器的访问会导致更高的和不可预料的系统开销，这是由于 Web 用户更为频繁的连接和断开。通常连接与断开连接所用的资源大于交互所用的资源。由于 Internet 的“冲浪”性质，用户的交互过程一般都较短。通常，公司外（Internet，而非 intranet）的用户会将使用卷变得很大，并难以预料。连接管理器通过建立用户 Servlet 可用的连接缓冲池将连接的系统开销分摊给多个用户请求。换言之，每个用户请求仅占用连接／断开连接所用系统开销成本的一小部分。在使用初始资源建立缓冲池中的连接后，其余连接／断开连接所用的系统开销就不大了，因为这只是重复使用已有的连接而已。

Servlet 以如下方式使用连接缓冲池：当一个用户通过 Web 向 Servlet 请求时， Servlet 从缓冲池使用一个已有的连接，这意味着用户请求不会引起数据服务器的连接系统开销。当满足请求时，Servlet 将连接返回至连接管理器缓冲池供其它 Servlet 使用。因而用户请求不会引起数据服务器的断开连接的系统开销。

连接管理器还使您能控制到数据服务器产品的并发连接数。当数据服务器的许可证协议限定用户数量时，这一特性是非常有用的。可以为数据服务器创建一个缓冲池，并将连接管理器缓冲池的“最多连接数”参数设成数据服务器产品许可证中限定的最大用户数。如果用其它程序而不用连接管理器连接到数据服务器，则不能保证该方法有效。

1．连接管理器结构

连接管理器维护一个连接到特定数据服务器产品处于打开状态的数据服务器缓冲池。每个数据服务器可以有一个或多个等同的或非等同的缓冲池。连接管理器的一个运行实例可以支持多个数据服务器。图6-1说明了在连接管理器与一个正在连接管理器的连接缓冲池中寻找可使用的连接的 Servlet 之间的典型交互作用。

(1) 当第一个 Servlet 试图与连接管理器通信时，由WebSphere应用服务器装入运行在WebSphere应用服务器下的连接管理器。只要WebSphere应用服务器在运行，连接管理器就一直被装入。

(2) WebSphere应用服务器将用户请求传递给一个 Servlet。

(3) Servlet 用连接管理器使用的方法从缓冲池中请求一个连接。

(4) 缓冲池给 Servlet 分配一个连接。

(5) Servlet 使用连接与数据服务器直接对话，这一过程中使用的是特定数据服务器的标准 API。

(6) 数据服务器通过与 Servlet 的连接返回数据。

(7) 当 Servlet 结束与数据服务器通信时，Servlet 把连接归还给连接管理器缓冲池，以供其它 Servlet 使用。

(8) Servlet 通过WebSphere应用服务器向用户发回响应。

在 Servlet 请求一个连接时，缓冲池中不一定有可用的连接。在这种情况下，连接管理器直接与数据服务器通信。连接管理器将：

- 请求一个新的连接（参见图6-1中的9 ）。
- 将连接添加到缓冲池中（参见图6-1中的10）。如果缓冲池中的连接数达到了规定的上限，连接服务器将不会把新的连接加入缓冲池中。
- 将新的连接交给 Servlet（参见图6-1中的4）。

2．性能特性

为缓冲池创建一个新的连接是一项系统开销很高的任务，新的连接将使用数据服务器上的资源。因此连接管理器尽量用缓冲池中的现有连接来满足 Servlet 的请求。 同时，连接管理器必须尽可能地最小

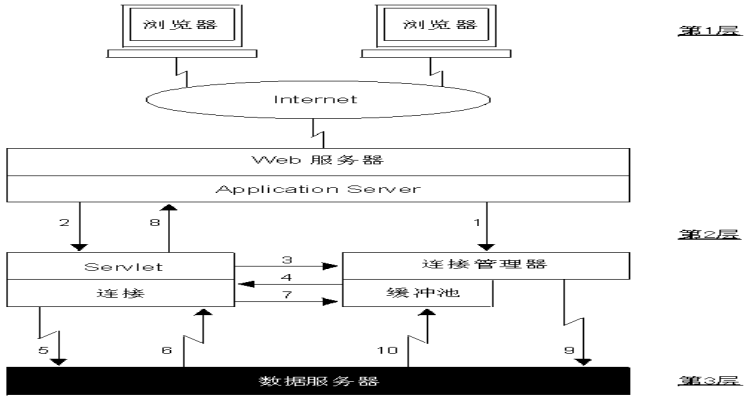


图6-1：连接管理器与Servlet 之间的交互

化缓冲池中的空闲连接，因为这是对系统资源的极大浪费。连接管理器与 Servlet 一同执行这些最小化和最大化任务。对于可选的性能，请适当地设置连接管理器参数。

连接管理器维护每个连接的验证时间标记、最近使用时间标记和正在使用标志。当某个 Servlet 第一次获得连接时，连接的验证时间标记和最近使用时间标记被设置为当前时间，连接的正在使用标志则被设置为真。

可将连接服务器配置成从某个 Servlet 中移走一个长时间未使用的连接。这个时间长度是由连接管理器的 “最长周期 ” 参数指定的。如果 Servlet 准备在较长一段时期内使用连接与数据服务器多次通信，可能希望将代码加入 Servlet 中，以便在每次使用连接之前，确认它仍占有这个连接。连接管理器从缓冲池中除去空闲的连接，因为它们会浪费资源。为了确定哪个连接是空闲的，连接管理器将检查连接标志和时间标记，这个操作是通过周期性地获取连接缓冲池信息来实现的：

(1) 连接管理器查看正在使用连接的最近使用时间标记。如果最近使用时间和当前时间之间的时间差大于 “ 最长周期 ” 配置参数，则本连接将被认为是一个残留连接，这就表明占有它的 Servlet 已终止或者没有响应。残留连接将被归还给缓冲池以供其它 Servlet 使用，它的正在使用标志被设置为假，且验证和最近使用时间标记被设置为当前时间。

(2) 连接管理器检查未被任何 Servlet 使用的连接（这些连接的正在使用标志为假）。如果最近使用时间与当前时间的时间差超过了 “ 最长空闲时间 ” 配置参数，将认为本连接是空闲的。空闲连接将被从缓冲池中除去，降至 “ 最少连接数 ” 配置参数指定的较低限定值。

3．监控连接管理器

WebSphere 应用服务器管理器为名为数据库缓冲池连接的连接管理器提供了一个监控程序。可以使用这些信息来查看如何执行连接缓冲池，并建议对连接缓冲池参数进行可能的更改。可以在更改参数之后对缓冲池执行监控，从而查看缓冲池特性的更改并帮助进一步对缓冲池参数进行调整。可以选择特定缓冲池，以从选择列表中对其进行监控。

6.6 用户简要表

Application Server 含有 com.ibm.servlet.personalization.userprofile 软件包中的类，这些类使维护关于 Web 站点访问者的持久信息和利用这些信息来定制 Web 页面变得更为容易。UserProfile 类包括了访问者的完整名称、邮政地址和电子邮件地址、电话号码等数据成员，并含有用于存储所选的语言、职业和用户定义的组信息的字段。另外，它还含有类属信息、购物手推车和剪贴板（一个 Java 散列表）。这就使您很容易地将其它您选择的对象合并到这些数据中，并将它们作为 UserProfile 类的一部分进行处理。因为这些对象在用户简要表文件的后继实例一直存在，所以它们必须是可串行化的。如果使用了一个 JDBC 数据库，则会将对象存储在数据库中。否则，会将对象作为文件存储。在 Application Server 版本 2.0 中，UserProfile 类使用了数据库连接管理器来维护 JDBC 连接。

可以使用“用户简要表文件”页面配置 UserProfile 类，该类用于定义和维护有关 Web 站点访问者的信息。UserProfile 类使用连接管理器来维护与 JDBC-从属数据库的连接。用户简要表文件的基本设置如下。

1. 启用用户简要表文件

(1) 查看“设置” -> “用户简要表文件”页面。

(2) 单击“启用”标签。

(3) 在“是否使用用户简要表文件?”字段中单击“是”。

(4) 对于“类名”字段，可以使用缺省值“com.ibm.servlet.personalization.userprofile.UserProfile”或为 UserProfile 类创建的扩展名。

(5) 单击“保存”按钮。

2．指定数据库信息

(1) 查看“设置” -> “用户简要表文件”页面。

(2) 单击“数据库”标签。

(3) 指定数据库产品的名称（如 IBM DB2 的 db2）。

(4) 指定数据库的 JDBC 驱动程序（如 COM.ibm.db2.jdbc.app.DB2Driver）。注意：这包括 Application Server 的 Java 类路径中的驱动程序 .zip 或 .jar 文件（如 DB2 的 db2java.zip）。

(5) 指定用户简要表文件的数据库名称以存储数据。如果不存在该数据库，将不会创建它。因此，可以指定现有数据库的名称，或在使用该用户简要表文件之前用该名称创建一个数据库。无需在数据库中创建任何表格。

(6) 指定数据库的所有者，即创建数据库的用户的标识符。

(7) 指定保存用户简要表文件数据的数据库表格的名称。无需创建该表格，因为 UserProfile 类会在初始化时创建它。

(8) 指定用于访问数据库和其表格的用户标识符。

(9) 指定与用户标识符相关的口令。

(10) 单击“保存”按钮。

3．配置连接管理属性

(1) 查看“设置” -> “用户简要表文件”页面。

(2) 单击“连接管理”标签。

(3) 指定要使用的连接缓冲池。

(4) 指定在“连接超时”指定的时间过后，连接仍不可用的情况下，指定是否等待并再试一次连接（与

“连接管理”页面中相同)。

(5) 单击“保存”按钮。

UserProfile 类和 Application Server 的其余部分之间的唯一的耦合在于，UserProfile 类和 IBM 的会话实现（ IBMSessionData）含有同一个用户名称段，该用户名称段可用于在两个类之间创建一些增效作用。用户简要表文件对象持有关于用户的信息，并通过唯一的用户名与用户的 HttpSession 对象相关联。用户名的值由 SessionData 类 getUserName() 和 setUserName() 方法设置并返回。

也可以扩展 UserProfile 类以创建一个更适合业务需求的子类，并用 Application Server 管理器指定专门 UserProfile 子类，该子类可用于所有的 Web 应用程序。

6.7 会话跟踪

Web 应用服务器具有会话跟踪（即跟踪用户请求）的功能。使用管理器的“会话跟踪”页面配置会话跟踪。会话跟踪程序可将同一用户的几个相关请求合并为单个会话（即一个 HttpSession 对象）。会话跟踪程序也能使 Web 应用服务器的多个实例能共享会话的一个公共缓冲池（即一个会话群集器)。

1．启用会话跟踪及其部件

查看 “会话跟踪”页面。单击“启用”标签。在“启用”标签中可以设置如下字段。

(1) 启用会话：是否启用会话跟踪，即请求和响应对象的相关会话方式是否有效。缺省值为是。

(2) 启用 URL 重写：指会话跟踪是否启用重写 URL 来获取会话标识符。若被启用，则会话跟踪程序将识别到达 URL 的会话标识符，若有必要的话，还将重写 URL 和发送该会话标识符。缺省值为。

(3) 启用 Cookie：指会话跟踪是否使用 cookie 来携带会话标识符。若为“是”，则会话跟踪程序将识别作为 cookie 到达的会话标识符，并用 cookie 作为发送会话标识符的方式。缺省值为是。

(4) 启用协议转换重写：当 URL 要求从 HTTP 转换成 HTTPS，或从 HTTPS 转换成 HTTP 时，是否将会话标识符添加到 URL 中。缺省值为否。

2．创建携带会话标识符的 Cookie

指定 Cookie 的设置，会话跟踪程序使用该 Cookie 将相关的个别请求置入逻辑用户会话中。当 Servlet 请求一个 HttpSession 对象表示一个用户会话时，会话跟踪程序可以使用一个 Cookie 将唯一的会话标识符传回给发出请求的浏览器。当 Servlet 发出后继请求时，Cookie 则获准在 Servlet 和会话跟踪程序间的往返传送。会话跟踪程序使用它来查找用户 Servlet 的现有 HttpSession 对象。换而言之，会话跟踪程序使用 Cookie 中的标识符确定该请求与该用户的其它请求相关。

查看 “会话跟踪”页面。单击“Cookie”标签。指定 Cookie 的名称，缺省值为 sesessionid。可选地指定下列字段：

(1) “域”字段的值以限制 Cookie 的发送位置。

(2) 会话 Cookie 在客户机浏览器上的活动时间（即“最长周期”字段，以毫秒计)。如果该字段为空，则当关闭浏览器时将删除该 Cookie。

(3) “路径”字段的值以指定 Cookie 发送所至的服务器路径。

(4) 在“安全”字段中单击“是”以限制仅允许 HTTPS 会话的 Cookie 进行交换。

3．设置会话跟踪活动的时限和参数

查看“会话跟踪”页面。单击“间隔”标签。

(1) 指定会话跟踪程序检查会话是否空闲及估计会话是否无效的时间间隔（以毫秒计)。

(2) 指定会话跟踪程序估计是否因为内存中会话过多，而将会话交换到磁盘的频率（以毫秒计)。

(3) 指定一次允许保留在内存中的会话数（最多驻留数)。

(4) 指定一个会话将允许在会话跟踪程序停止验证之前保持空闲的时间（以毫秒计)。

4．保留持续数据

使用“持续性”标签指定在会话跟踪程序关闭重新并启动后，以验证会话数据是否交换到磁盘。如果“持续性”字段设置为否，则会话跟踪程序将在每次启动时删除旧的会话交换文件。查看“会话跟踪”页面。单击“持续性”标签。在“持续性”字段中单击“是”。指定会话跟踪程序用于存储要保持持续性的

会话数据。注意：交换目录必须是一个空目录。

5．指定会话群集和协议转换设置

可以将 WebSphere 应用服务器配置成以如下方式运行：独立主机、会话群集服务器、会话群集客户机、协议转换主机、独立主机。查看“会话跟踪”页面，单击“主机”标签。

(1) 若 WebSphere 应用服务器实例为一个独立主机，则它不参与会话群集。在独立模式下, WebSphere 应用服务器将维护它自己的会话信息，而对客户的会话信息请求将不予响应，并且不从服务器请求会话信息。缺省值：是。

(2) 在用如 IBM eNetwork Dispatcher 之类的产品群集 Web 服务器的环境中，指定 WebSphere 应用服务器实例的主机名和 IP 地址，其中该实例为会话群集服务器。

(3) 若将 WebSphere 应用服务器实例配置成 URL 重写和协议转换（在“启用会话跟踪”标签中），则指明群集主机的主机名或 IP 地址。例如，可能为 IBM eNetwork Dispatcher 主机。

要使更改生效，请在更改“独立主机”或“会话群集服务器”字段后，重新启动 Web 应用服务器。更改这些字段后对所有配置所做的更改都保存在配置文件中，直至重新启动后方才生效。

6.8 安全性

WebSphere 应用服务器具有很好的安全性支持。安全性简单地说就是确定谁可访问重要的系统资源，这些系统资源包括文件、目录、程序、连接和数据库。以独立模式运行 WebSphere 应用服务器比作为 Web 服务器的一部分运行具有更高的安全性。如果安全性要求超出了 Web 服务器提供的安全性，那么请以独立模式运行 WebSphere 应用服务器。下面介绍使用存取控制表保护资源、选择认证方案和协议、在 WebSphere 应用服务器中使用目录服务。

1．使用存取控制表保护资源

建立了一个设置安全性的基本过程。

与安全性相关的概念包括用户、组、资源、许可权、领域、和存取控制表（ACL），这些安全性概念是紧密相关的，现说明如下：

(1) 用户：一个可被 Web 服务器认证的身份。用户可以是人也可以是计算机。

(2) 组：用户的集合。组提供了一个管理大量的用户的有效方法，这是因为管理员可以一次指定一个组的许可权。通常，组成员之间有一些共同特征。例如，一个公司中可能包括一个由管理级雇员组成的组，和另一个由非管理级雇员组成的组。其中，可能对非管理级用户赋予查看销售数据文件的访问权，而对管理级用户赋予查看和编辑销售数据文件的访问权。可以将一个人定义成单个用户，同时也将其定义成组的成员。例如，个人 Amy 可能代表单个用户 amy，也可能代表组 managers 的成员。

(3) 资源：通过 Web 服务器进行存取的有价资源包括 HTML 文件和目录（Web 页面）、其它文件和目录（如 FTP 文件）、Web 应用程序（Java Servlet 或 CGI 程序），而通过 WebSphere 应用服务器进行存取的资源包括 Java Servlet、启用访问企业资源和应用程序（如数据库）的定制 Servlet、连接、套接字、文件和其它可由 Servlet 使用的资源。

(4) 许可权：许可权表示请求访问资源的特权。管理员可通过建立存取控制表向用户和组授予许可权，以保护资源。许可权是与特定的资源相关的。想象一下，如果一个用户具有查看文件 A 和修改文件 B 的许可权。那么该用户不能用修改文件 B 的许可权来修改文件 A。每个许可权仅适用于特定的资源，且不能传递。单个用户的许可权优先于组的许可权。例如，如果用户 amy 对文件具有读、写权，但同时 amy 所属的组对该文件仅有读取权，那么 amy 仍对该文件具有读、写权，这是因为单个用户的许可权将覆盖具有更多限定的组许可权。（即使组的许可权限制性低于单个用户的许可权，但用户的许可权仍将覆盖组的许可权）。

(5) 领域：领域是用户、组和存取控制表的数据库。为了使用户能访问领域中的资源，必须在该领域中定义需访问资源的用户。一个用户可以属于几个领域，但在同一领域中用户标识符不能重复。例如，用户 amy 可属于 fileRealm 和 anyRealm 领域。但每个领域中仅可包含一个名为 amy 的用户。可在每个领域中赋予用户 amy 访问不同资源的不同许可权。

(6) 存取控制表：与资源关联的存取控制表指定了领域中的哪些用户和组可以访问资源。

存取控制表（ACL）、领域和资源的关系如下：

- 一个领域可以包含许多 ACL。
- 一个领域可以包含许多资源。
- 一个 ACL 可以仅属于一个领域。
- 一个资源可以仅属于一个领域。
- 一个资源仅与一个 ACL 相关。
- 一个 ACL 可与许多资源相关。

WebSphere 应用服务器附带了一些前期建立的领域：

● defaultRealm 定义了用户如何访问本地定义的资源。也可建立存取控制表，以确定哪些用户和组对哪些资源具有访问权。

● NT 领域（NTRealm）和 UNIX 领域（UnixRealm）定义了在操作系统中拥有帐户的用户如何使用 WebSphere 应用服务器资源。 操作系统中定义的用户可由 WebSphere 应用服务器共享。只要他们存在于下层系统中，这种共享就一直存在。WebSphere 应用服务器管理器界面使您能查看该领域；但若要更改它，必须使用操作系统所提供的设施。目前，WebSphere 应用服务器可以共享操作系统定义的用户，但不能共享组。

● servletMgrRealm 定义了 Servlet 如何访问远程定义的资源，如远程装入的 Servlet。 servletACL 是该领域中的唯一一张存取控制表。当远程装入且带有数字签名的 Servlet 试图访问一个受保护的资源时，将对 Servlet 中的数字证书与 servletMgrRealm 中与用户关联的数字证书进行比较。 servletACL 定义了是否授予或拒绝许可权。例如，假设用户 X 的数字证书封装在 anyServlet.JAR 文件中。如果用户 X 被添加在 servletMgrRealm 中，则所有包含与该用户相同数字证书的 Servlet （anyServlet.JAR 文件中）可以执行和访问赋予该用户的资源。

● 最后，如果在 WebSphere 应用服务器管理器的“ 目录管理 ”页面中启用目录服务，将显示 LDAPRealm。WebSphere 应用服务器可以共享定义在目录服务中的用户和组，且这种共享将一直持续下去，直至除去他们或禁用目录管理支持。WebSphere 应用服务器管理器界面使您能查看该领域；但若要对它进行更改，必须使用 LDAP 服务器所提供的设施。如需更多有关 LDAP、目录服务和 LDAPRealm 的信息，请参阅有关使用目录服务的文档。

WebSphere 应用服务器使您能设置各种许可权。例如，其中包括发送和接收文件、删除文件、读取和写入文件、装入 Servlet、链接至库文件、打开和侦听套接字等许可权。某些情况下，服务不需要其客户存在于存取控制表中。例如，许多 Web 页面（HTTP）服务向所有用户公开其文档，而不要求他们在存取控制表中进行注册。

可通过为每个资源在单个领域中建立单个存取控制表（ACL）来保护该资源（下面将进行讨论）。 ACL 将指定可以访问或修改资源的用户或组。对于要保护的每个资源，需要指定存取控制表、安全性领域、认证方案（用来验证访问资源的用户的方法）。下面介绍实现安全性的一个示例。

使用各种管理器页面，以用存取控制表来建立安全性的基本步骤显示如下。

(1) 定义用户。

使用“ 安全性 ”-> 使用“ 用户 ”页面来定义用户，即定义被允许访问资源的个人和计算机。例如，定义名为 bopeep 的用户：

- 选择 defaultRealm。
- 单击页面左上部的“ 添加 ”按钮。
- 输入用户名 bopeep。
- 输入口令 sheep。再一次输入该口令以进行验证。
- 单击“ 确定 ”按钮。
- 验证 bopeep 显示在“ 定义的用户 ”列表中。

(2) 可选地定义组

使用“ 安全性 ”-> 使用“ 组 ”页面来定义用户所属的组，以使管理更加容易。首先，选择一个领域。

下一步，选择将组添加到该领域中。可以用您喜欢的名称来命名组，然后将用户从非成员状态转换为成员状态。例如，将用户 bopeep 添加到组 mypeeps 中：

- 选择 defaultRealm。
- 单击页面左上部的“ 添加 ”按钮。
- 输入组名 mypeeps。
- 在“ 非成员 ”列表中选择用户 bopeep。
- 单击“ 非成员 ”列表旁的“ 添加 ”按钮。
- 验证 bopeep 已转换到成员框中。

(3) 创建 ACL

使用“ 安全性 ”-> 使用“ 存取控制表 ”页面来创建 ACL。例如，创建名为 sheepcontrol 的 ACL：

- 单击定义“ 定义的存取控制表 ”列表旁的“ 添加 ”按钮。
- 输入 ACL 名称 sheepcontrol。
- 单击“ 确定 ”按钮。

(3) 定义资源

使用“ 安全性 ”-> 使用“ 资源 ”页面以将要保护的资源添加到领域中。（注意：必须在执行“ 创建 ACL”步骤后才能执行该步骤，这是因为必须指定新资源所属的 ACL）。例如，将 CheckMessage Servlet 添加到领域 defaultRealm 中的 sheepcontrol ACL 中：

- 选择 defaultRealm。
- 单击“ 添加 ”按钮。
- 选择基本方案作为认证方案。（“ 方案 ”选项将在下文中讨论）。
- 选择 sheepcontrol ACL。
- 选择 Servlet，并指定 CheckMessage Servlet。
- 单击“ 确定 ”按钮。
- 验证 CheckMessage 现已作为 ACL sheepcontrol 的资源列示出来。

(4) 赋予许可权

返回 ACL 并赋予用户、组和计算机访问 ACL 资源的许可权。例如，赋予 bopeep 具有 GET 和 PUT 许可权，以使它可以访问领域 defaultRealm 中 sheepcontrol ACL 的文件和文件夹：

- 确保已在“ 定义的存取控制表 ”列表框中选定 sheepcontrol。
- 单击页面底部“ 主许可权 ”字段旁的“ 添加 ”按钮。
- 当显示“ 在 defaultRealm 中的 ACL sheepcontrol 添加许可权 ”框时，单击“ 文件与文件夹 ”。
- 下一步，选择许可权选项为 bopeep。由于 bopeep 是一个用户，所以单击“ 用户 ”。在列表中选择 bopeep。（注意，如果选择的是“ 组 ”，则将看到“ 主名 ”列表中的组 mypeeps）。
- 使用复选框赋予 bopeep 具有 GET 和 PUT 许可权。
- 单击“ 确定 ”按钮。
- 通过查看“ 主许可权 ”框来验证 bopeep 具有正确的许可权。如果没有显示许可权，请单击 bopeep 左边的加号。这时将显示一个列表，该列表显示了 bopeep 的许可权。

2．选择认证方案和协议

首先讨论的是认证协议，即 HTTP 和 HTTPS。接下来将会讨论的是认证方案，即基本、摘要、定制和证书认证。最后讨论的是有关组合方案和协议的策略。

(1) 关于认证协议

HTTPS 则是 HTTP 和 SSL 协议的组合。SSL（安全套接字层）是一个网络安全协议，它用来提供服务器和客户机之间必要的安全性。如果选择 HTTP，认证数据的接收无任何保护。如果选择 HTTPS，认证数据在每个 SSL 协议中加密。如果要使用证书认证，那么必须使用 HTTPS。

(2) 关于认证方案

WebSphere 应用服务器支持的认证方案包括基本认证、摘要认证、定制认证、证书认证。

- 基本认证：使用 HTTP 或 HTTPS 请求来自客户机的用户名和口令。 用普通文本将用于验证的

信息发送给服务器验证。所有浏览器都支持基本认证。如果一个用户标识符和口令提供了足够的认证，那么请考虑使用基本认证。

- 摘要认证：使用 HTTP 或 HTTPS 请求来自客户机的用户名和口令。将用于验证的用户名和口令的加密形式（使用摘要）发送给服务器。并非所有浏览器都支持摘要认证。（当前仅 Sun HotJava 浏览器支持该认证方案）。如果浏览器不支持摘要认证，那么其用户将无法访问由该协议进行保护的资源。
- 定制认证：使用 HTTP 或 HTTPS 来请求使用 HTML 格式定制的客户机信息。由 CGI 和 Servlet 将这些用于验证的信息用普通文本发送到服务器上。当需要除了标识符和口令之外的用户认证时，可使用定制认证。例如，可以请求一个社会安全号的用户认证。使用该协议，可建立 HTML 格式以询问用户数据。认证是由服务器端代码（CGI 和 Servlet）执行的，而不是由 IBM WebSphere 应用服务器运行时应用程序执行的。如果使用定制认证，请使用 HTTPS 保护数据。
- 证书认证：使用 HTTPS 以请求一个客户机证书。必须启用 SSL 客户机认证选项。将用于验证的信息发送给服务器。认证使用的数字证书具有很高的安全性，且证书认证通常对用户是透明的。系统或站点管理员会对客户机证书进行管理。通常这些任务是由证书权威服务器软件授权的，例如 IBM Vault Registry 产品。

(3) 组合认证方案和协议

正如前述，除非是在一个与安全无关的环境中，否则 HTTPS 通常更为可取。对于不同的安全性要求可以对方案和协议进行组合，策略如下：

- 对于基本安全性要求，使用基本、摘要或 HTTP 上的定制认证。
- 对于较高安全性要求，使用基本、摘要或 HTTPS 上的定制认证。
- 对于最高安全性要求，使用 HTTPS 上的证书认证。

3．使用目录服务

一个目录服务是信息仓库、存取法和相关服务的组合。信息仓库通常是用于存储位置信息和其它有关资源（例如用户、打印机、文件服务器和 WebSphere 应用服务器）的详细信息的数据库。存取法是指轻量级目录访问协议（LDAP）或其它可用来与目录服务组件进行通信的存取法。相关服务是指目录服务提供的用于查询、操纵和认证数据库中信息的设施。

对集中式安全性数据通常需要使用目录服务。目录服务可以为整个网络提供单个管理点。使用目录服务，可以一次定义用于所有应用程序（包括 WebSphere 应用服务器）的用户和组，而无需为每个应用程序分别定义用户和组。目录服务对于实现安全性很有帮助，即通过认证用户和控制对资源的访问。没有目录服务，可能必须为不同的软件产品（例如 WebSphere 应用服务器、Web 服务器和操作系统）重复定义相同的用户和组，这是因为这些软件产品不共享安全性数据。

WebSphere 应用服务器版本 2.0 支持使用轻量级目录访问协议（LDAP）V2 的目录服务。这些目录服务包括 Domino Directory 版本 4.6.x 和 Netscape Directory Server 版本 3.x。WebSphere 应用服务器版本 2.02 （本地语言版本）也支持 IBM Suites（不包括 IBM Suites 版本 1.0）的 eNetwork Directory 组件。使用目录服务的基本步骤如下：

(1) 设置目录服务，或使用一个现有的目录服务。

(2) 使用 WebSphere 应用服务器的管理器界面的“ 目录管理 ” 页面，以标识至 WebSphere 应用服务器的目录服务，并指定基本设置。

(3) 查看管理器中的目录服务用户和组信息。

(4) 在 WebSphere 应用服务器中定义存取控制表（或已有的 ACL），以利用目录服务用户和组信息。

(5) 用 ACL 保护 WebSphere 应用服务器资源。

使用“ 目录管理 ” 页面指定目录服务的步骤如下：

- 查看“ 设置 ” -> “目录管理 ” 页面。
- 在“ 配置 ” 标签中，请在“ 是否启用目录?”字段中单击“ 是 ”
- 使用下拉列表选择目录类型。选择：
  - —“ netscape”使用 Netscape Directory Server；
  - —“ domino4.6”使用 Domino Server；

— “ enetworkibmsuite”使用 IBM Suite 的 eNetwork Directory Server 组件。

- 输入驻留目录服务的计算机主机名。
- 输入服务的端口号。
- 输入将作为目录服务的 LDAP 搜索起始点的基本分辨名（Base DN）。例如，o=raleigh.ibm.com.
- 可选功能。输入联编分辨名和联编口令。这些字段空白表示 WebSphere 应用服务器匿名联编目录服务。如果指定了联编分辨名和口令，则请确认将目录服务配置成使用同一分辨名和口令来认证 WebSphere 应用服务器，否则认证将告失败。

当使用“ 目录管理 ”指定目录服务时，管理器将提供的功能有：

- 动态地建立、初始化和维护与 LDAP 目录服务相关联的 ldapRealm。
- 使用管理器的“ 安全性 ”页面关联 WebSphere 应用服务器资源和 ldapRealm。
- 将 WebSphere 应用服务器存取控制表（ACL）与 ldapRealm 的资源关联起来。

IdapRealm 是一个包含已定义在目录服务中的用户和组的 WebSphere 应用服务器安全性领域。可以使用 WebSphere 应用服务器的管理器界面的“ 安全性 ”页面，以建立存取控制表和 IdapRealm 之间的关联。ldapRealm 对资源进行保护，并且存取控制表允许属于该领域的用户和组（在这种情况下，用户和组在目录服务中）访问资源。

ldapRealm 使用非 SSL LDAP V2 协议来与目录服务进行通信，并根据在“ 目录管理 ”页面中指定的设置进行联编。当受 IdapRealm 保护的 WebSphere 应用服务器资源需要认证时，将根据 IdapRealm 来验证客户机的用户标识符和口令。该过程如下：

- 客户机向 WebSphere 应用服务器发出 Servlet 请求。
- WebSphere 应用服务器确定请求的资源是否受 ACL 保护。
- 因此 WebSphere 应用服务器将一个认证要求发送给客户机。
- 客户机输入用户标识符和口令。
- 目录服务验证用户标识符和口令与保护资源的 IdapRealm 是否相符合：目录服务发出 LDAP 查找，以查找与指定用户标识符相符的人。为该用户创建一个分辨名（DN）。
- 目录服务使用该用户分辨名和口令执行一个 LDAP 连接。
- 当目录服务认证用户后，WebSphere 应用服务器 Java 引擎将调用 Servlet 以处理与客户机获得的许可权一致的客户机请求。

可以通过访问 ldap://act:389/o=raleigh.ibm.com 来测试目录服务是否正在运行，其中：act 是运行目录服务的机器的主机名，389 是服务的端口号，o=raleigh.ibm.com 是基本分辨名，它还代表了目录服务中的搜索起始点。

第七章 WebSphere 应用编程

WebSphere应用服务器中包括它自己的软件包，该软件包用于扩展和添加 Java Servlet API。这些扩展和添加简化了维护用户信息、个性化 Web 页面和分析 Web 站点使用情况。WebSphere应用服务器 API 的 Javadoc 安装在产品的 doc/apidocs 目录下。WebSphere应用服务器 API 软件包为：

1. com.ibm.servlet.personalization.sessiontracking

对 JDSK 的这种 WebSphere应用服务器 扩展记录了分派页面，该页面引导访问者进入 Web 站点，该设备同时在站点内跟踪访问者的位置，并关联了会话和用户标识符。IBM 还将会话群集支持添加到 API 中。

2. com.ibm.servlet.personalization.userprofile

提供维护有关 Web 访问者的详细信息的方法，并在 Web 应用程序合并这些信息， 以便能够提供一个个性化的用户经历。

3. com.ibm.servlet.connmgr

使 Servlet 能与连接管理器进行通信，该连接管理器对打开的数据库服务器和 JDBC 从属数据服务器产品的连接的缓冲池进行维护。 当 Servlet 从缓冲池接收到连接时，它可以用它的 API 直接与数据服务器进行通信。

4. com.ibm.db

将类包括在内，以简化对关系数据库的访问并同时提供增强访问功能，如结果高速缓存、 通过执行高速缓存进行更新和查询参数支持。

5. com.ibm.servlet.personalization.sam

使您能用站点活动监控程序（Site Activity Monitor）注册自己的 Web 应用程序，它是一个提供给您动态、实时查看您 Web 站点活动的applet。

6. com.ibm.servlet.servlets.personalization.util

包括特殊的 Servlet，它允许 Web 站点管理员动态地登记公告牌，并允许 Web 站点访问者之间交换信息。

7.1 数据库应用编程

1. Servlet 如何使用连接管理器

所有使用连接管理器的 Servlet 都将执行下列步骤：

(1) 创建连接规范：Servlet 为连接到基本数据服务器准备了一个用以标识必需信息的规范对象。其中某些信息是仅针对于特定数据服务器的，而一般信息则适用于所有下层数据服务器。Servlet 仅准备一次规范并将它应用于所有请求，或者为每个用户请求准备一份新的规范。如果要为每个用户请求准备新的规范，则必须在第 3 步（使用规范）之前，执行该操作。

(2) 获得连接管理器：Servlet 为与连接管理器交流而获取有关连接管理器的引用。这一操作只需在 Servlet 的生命期中执行一次。

(3) 获得与连接管理器的连接：Servlet 请求连接管理器以连接到特定数据服务器，该数据服务器使用第 1 步中准备的连接规范。返回的连接对象来自于连接管理缓冲池，并且是在连接管理器 API 中定义类别的实例，它不是一个来自于数据服务器下属 API 集合中的类的对象。先调用它来暂时连接一个连接管理器或一个 CM 连接。通常 Servlet 为每个用户请求获得一个 CM 连接。

(4) 使用 CM 连接以访问预先建立的数据服务器连接：Servlet 调用第 3 步中返回 CM 连接的方法，检索定义在下层数据服务器的 API 集合中的对象。调用该数据服务器连接对象（或暂时调用一个数据连接）以从 CM 连接中将它分辨出来。这个数据连接不同于 CM 连接，它是来自下层数据服务器 API 集合。这个数据连接不是为 Servlet 而创建 -- 相反这个 Servlet 使用预先建立的数据连接因为它拥有一个缓冲池到缓冲池的连接。将使用来自下层数据服务器 API 集合的方法，以将该数据连接用于与数据服务器的交互。例如，在 Servlet 代码示例中下层数据服务器是一个 JDBC 数据库，而且数据连接对象则是来自 JDBC API 的连接类。

(5) 和数据服务器的相互作用：Servlet 与数据服务器相互作用（如：检索数据和更新数据）使用的是数据连接对象的方法。这些方法针对的是下层数据服务器，因为数据连接实际上来自下层数据服务器的 API 集合。不同的下层数据服务器的数据连接具有不同的方法。方法的全部文档可用于特定数据服务器产品的 API 文档。例如为了实现一个 JDBC 数据连接，可能需要查看关于 java.sql 程序包的文档，以及任何正使用的启用 JDBC 的关系数据库文档。如果在同一用户请求中对多个数据服务器的交互使用数据连接，可能希望在 Servlet 中仍有关联的 CM 连接的其它交互之前进行验证。要验证 Servlet 中仍有连接，Servlet将依次调用 verifyIBMConnection() 方法以使连接管理器检查连接的验证时间标记。如果 Servlet 中仍有连接，则将最近使用时间标记自动更新为当前时间，并作为调用 VerifyIBMConnection() 方法的一部分。然后 Servlet 使用连接来与数据服务器进行通信，确认连接生效。当 Servlet 完成连接时，它将连接释放回连接缓冲池。连接管理器将正在使用标记设置为假，并将验证和最近使用时间标志设置为当前时间。

(6) 释放连接:：Servlet 把 CM 连接返回到连接管理器缓冲池，释放连接以供另一个 Servlet 或该 Servlet 的另一个请求使用。

(7) 预备和发送响应：Servlet 准备响应，并将该响应发回给用户。在该步骤中可能不会使用任何连接管理器 API。

2．CM示例

```
// ********************************************************************
// * IBMConnMgrTest.java - 连接管理器示例的Sevlet，简称CM示例的servlet
// ********************************************************************
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
import com.ibm.servlet.connmgr.*;     // 使用了IBM的连接管理器软件包com.ibm.servlet.connmgr

public class IBMConnMgrTest extends HttpServlet {
    // Use to communicate with connection manager.
    static IBMConnMgr connMgr = null;

    // Use later in init() to create JDBC connection specification.
    static IBMConnSpec spec     = null;          // the spec
    static String DbName          = null;          // database name
    static String Db              = "db2";        // JDBC subprotocol for DB2
    static String poolName       = "JdbcDb2"; // from Webmaster
    static String jdbcDriver    = "COM.ibm.db2.jdbc.app.DB2Driver";
    static String url             = null;          // constructed later
    static String user            = null;          // user and password could
    static String password       = null;          // come from HTML form
    static String owner           = null;          // table owner

    // Name of property file used to complete the user, password,
    // DbName, and table owner information at runtime。
    static final String CONFIG_BUNDLE_NAME = "login";

    // ****************************************************************
    // * Initialize Servlet when it is first loaded                   *
    // ****************************************************************
public void init（ServletConfig config）throws ServletException {
    super.init(config);
    try {
            // Get information at runtime (from an external property file
            // identified by CONFIG_BUNDLE_NAME) about the database name
            // and the associated database user and password.    This
            // information could be provided in other ways.    It could be
            // hardcoded within this application, for example.
```

```
            PropertyResourceBundle configBundle =
            (PropertyResourceBundle) PropertyResourceBundle.getBundle(CONFIG_BUNDLE_NAME);
            DbName    = configBundle.getString("JDBCServlet.dbName");
            url          = "jdbc:" + Db + ":" + DbName;
            user         = configBundle.getString("JDBCServlet.dbUserid");
            password = configBundle.getString("JDBCServlet.dbPassword");
            owner       = configBundle.getString("JDBCServlet.dbOwner");
    }
    catch (Exception e)   {
            System.out.println("read properties file: " + e.getMessage());
    }

    try {
                        // 第一步：创建连接规范
            spec = new IBMJdbcConnSpec(poolName, true, jdbcDriver, url, user, password);
                        // 第二步：获得连接管理器
            connMgr = IBMConnMgrUtil.getIBMConnMgr();
    }
    catch (Exception e) {
            System.out.println("set connection spec, get connection manager: " +
                                    e.getMessage());
    }
}
    // ****************************************************************
   // * Respond to user GET request                                                *
   // ****************************************************************
public void doGet(HttpServletRequest req, HttpServletResponse res) {
        IBMJdbcConn cmConn      = null;
        Connection dataConn    = null;
        Vector firstNameList = new Vector();
        try {
            // 第三步：获得与连接管理器的连接
            cmConn = (IBMJdbcConn)connMgr.getIBMConnection(spec);

            // 第四步：使用 CM 连接以访问预先建立的数据服务器连接
           dataConn = cmConn.getJdbcConnection();

            //第五步：和数据服务器的相互作用
            Statement stmt = dataConn.createStatement();
            String query      = "Select FirstNme " +
                                    "from " + owner + ".Employee " +
                                    "where LASTNAME = 'PARKER'";
            ResultSet rs      = stmt.executeQuery(query);
            while(rs.next())
            {
```

```
            firstNameList.addElement(rs.getString(1));
        }
        // Invoke close() on stmt, which also closes rs, freeing
        // resources and completing the interaction.    You must
        // not, however, close the dataConn object.    It must
        // remain open and under the control of connection manager
        // for possible use by other requests to this servlet or to other servlets.
        stmt.close();
    } catch (Exception e) {
        System.out.println("get connection, process statement: " +
                                e.getMessage());
    }
                //  第六步：释放连接
    finally {
        if(cmConn != null) {
            try {
                cmConn.releaseIBMConnection();
            } catch(IBMConnMgrException e) {
                System.out.println("release connection: " + e.getMessage());
            }
        }
    }
    //  第七步：预备和发送响应
    // say Hello to everyone whose last name is 'Parker'
    res.setContentType("text/html");
    // Next three lines prevent dynamic content from being cached
    res.setHeader("Pragma", "no-cache");
    res.setHeader("Cache-Control", "no-cache");
    res.setDateHeader("Expires", 0);
    try {
        ServletOutputStream out = res.getOutputStream();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Hello DBWorld</TITLE></HEAD>");
        out.println("");
        if(firstNameList.isEmpty()) {
            out.println("<H1>Nobody named Parker</H1>");
        }
        else {
            for(int i = 0; i < firstNameList.size(); i++)
            {
                out.println("<H1>Hello " +
                                firstNameList.elementAt(i) + " Parker</H1>");
            }
        }
        out.println("</BODY></HTML>");
```

```
            out.close();
        } catch(IOException e) {
            System.out.println("HTML response: " + e.getMessage());
        }
    }
}
```

3．连接管理器 API

结合上述示例，下面讨论部分的连接管理器 API。

(1) IBMJdbcConnSpec 类

创建该类的规范对象以将连接规范记录到期望的数据服务器上。这通常是在步骤 2 中执行的。规范对象实际上并没有设置规范，但是它却被另一个方法当作一个变量使用，该方法用于设置规范。请在 IBMConnMgr 类中参阅 getIBMConnection() 方法。

目前，连接服务器仅支持遵从 JDBC 的数据服务器，因此只有 JDBC 规范的类是可用的。当支持其它数据服务器时，其它规范类也将变得可用。在建立规范对象后，请使用 get 和 set 方法指定连接要求。通常，将在规范对象的初始建立过程中指定所有的要求。

public IBMJdbcConnSpec（String poolName，boolean waitRetry，String dbDriver，String url，
String user，String password)

- 缓冲池名称：包含希望使用的连接类型的连接管理器缓冲池。请查询 Web 管理员以获知缓冲池名称。
- 等待重试：如果当前缓冲池中没有可用的连接时，是否要等待连接释放（若要等待，请指定为真）。Web 管理员使用“ 连接超时 ”参数来设置等待整个缓冲池的时间长度，因此如果在给定的时间内仍不能使用连接，请求将告失败。如果缓冲池中的连接不可用，请指定为假表示立即失败。
- 数据库驱动程序：由特定的 JDBC 产品多提供的驱动程序类，或提供 JDBC-ODBC 桥接器的驱动程序名。请参阅 Web 管理员或数据库管理员以获得驱动程序名。
- URL ：数据库 URL。
- 用户：作为代表进行连接的数据库用户。
- 口令：用户的口令。

(2)IBMConnMgrUtil 类

使用该类的一种方法以获得有关连接管理器的引用。这通常是在步骤 2 步中执行的。将使用获得的引用来和连接管理器进行通讯并使用它所提供的服务。这里只讨论一个静态方法：

public static IBMConnMgr getIBMConnMgr()；

该方法将引用返回到连接管理器。

(3) IBMConnMgr 类

正在运行的连接管理器实例是该类的一个实例。将在步骤 2 步中获得有关连接管理器的参考。Servlet 从未创建一个连接管理器的实例，而是使用一个现有实例的参考。这里只讨论该类的一种方法：就是用于从缓冲池获得一个 CM 连接的 getIBMConnection() 方法（在步骤 3 中使用）。

public IBM Connection get IBM Connection (IBM ConnSpec connSpec) throws
IBM ConnMgr Exception

参数connSpec包含了对一个专用下层数据服务器的详细连接要求。在示例中，它是 IBMJdbcConnSpec 对象（步骤 1 中创建的）。该方法从连接管理器缓冲池返回一个 IBMConnection 对象。

如果 CM 连接是可用的，则 getIBMConnection() 方法为 Servlet 从缓冲池中获得一个 CM 连接。仅获通过的参数是连接的规范对象（在步骤 1 中创建的）。如果一个 CM 连接不是马上可用，并且如果规范对象中的 waitRetry 设置为真，则 Servlet 可以等待 CM 连接变成可用为止。等待的时间长度是由 Web 管理员用“ 连接超时 ”参数设置的。Web 管理员还可以禁止等待或不确定地延长等待时间。如果在等待周期过后 CM 连接仍不可用，getIBMConnection() 方法将报告 IBMConnMgrException 异常。如果

waitRetry 设置为假，获取 CM 连接的失败将导致 getIBMConnection() 立刻报告异常。

返回的 IBMConnection 对象是一个“普通”连接对象，必须将它造型成您需访问的下层数据服务器的特定连接对象。例如，在示例 Servlet 中，返回的 IBMConnection 对象被造型成一个 IBMJdbcConn 对象。缺少了适当的造型，连接对象就不能使用其方法在步骤 5 中到达下层数据服务器。

(4) IBMJdbcConn 类

考虑到要访问的下层数据服务器，需将步骤 3 中检索的 IBMConnection 对象造型成一个连接对象。否则，就不存在与下层数据服务器关联的 API 访问。当前，只有一个这样的连接类可用，供访问 JDBC 数据服务器的 IBMJdbcConn 类。通过 IBMJdbcConn 对象访问 JDBC 服务器的工作通常是在步骤 4 中建立的。IBMJdbcConn 类只关心一种方法：

public Connection getJdbcConnection()

该方法将一个连接对象返回至下层 JDBC 数据服务器。连接类来自 JDBC API ，并由 java.sql 软件包存档。连接类的方法让您可使用 JDBC 数据服务器进行交互。该文档的其它部分较多地使用了连接类，而非数据连接，以将其和 CM 连接区分开。 CM 连接不是下层数据服务器 API 集合的一部分。

(5) IBMConnection 类

IBMJdbcConn 类是 IBMConnection 类的一个扩展。因此，IBMConnection 中的多数方法也包括在 IBMJdbcConn 类的实例中。这里讨论的是 IBMConnection 类（即 IBMJdbcConn 类）的两种方法：

public boolean verifyIBMConnection() throws IBMConnMgrException

使用该方法以验证 CM 连接是否仍然有效。可步骤 5 中可选地使用该方法。如果 CM 连接在指定的时间内一直是非活动的（由 Web 管理员检查以确定该由连接管理器配置的操作），则连接管理器可能从 Servlet 中移走 CM 连接。如果在一个用户请求中对几个交互操作使用数据连接，可能希望在每个交互操作在缓冲池中检查 Servlet 中是否仍有关联的 CM 连接之前，调用 verifyIBMConnection() 方法。如果 Servlet 中仍有连接，该方法将返回真，且调用的方法也将重新设置一个最近使用时间标记。

如果预见 Servlet 与数据服务器的交互操作（从一个用户请求中）将在几秒中之内完成，并且如果连接管理器缓冲池的“最长周期”参数至少长达有几分钟，则可能不需要使用 verifyIBMConnection() 方法，请求将在连接管理器移走连接的很早以前就完成。

public void releaseIBMConnection() throws IBMConnMgrException

使用该方法将 CM 连接返回到缓冲池。可在步骤 6 中可选地使用它。当一个 Servlet 不再需要 CM 连接对象时， Servlet 便会使用这种方法将连接释放回缓冲池。这应该在每个用户请求的最后完成。

4．数据访问 JavaBean

访问遵从 JDBC 关系数据库的 Java 应用程序（和 Servlet）通常使用 java.sql 软件包中的类和方法来访问数据。数据访问JavaBeans 是遵守JavaBeans 规约的 Java 类，这种类是用来存取数据库中数据。由于数据访问类是 JavaBean，可以使用在集成开发环境（IDE）下，如 IBM 产品 VisualAge for Java，允许程序员用可视方法操纵 Java 类，而非编辑 Java 代码的行。由于 JavaBean 也是 Java 类，所以程序员在写 Java 代码时，可以象使用普通的类一样使用 JavaBean。

可以使用软件包com.ibm.db 中的类和方法，来替代使用 java.sql 软件包。软件包 com.ibm.db 提供了：

(1) 查询结果的高速缓存：可以立刻检索所有的 SQL 查询结果并将其放入高速缓存。应用程序（或 Servlet）可以在高速缓存中向前和向后移动或直接转移到高速缓存中的任何结果行。比较它与 java.sql 软件包的灵活性，java.sql 软件包每次从一个数据库中检索一行（仅在向前方向中），而且新的检索行覆盖上一次的检索行，除非又写入了扩展功能代码。对于大量的结果集合，数据访问 JavaBean 提供了检索和管理包的方法，完整的结果设置的子集。

(2) 通过结果高速缓存更新：Servlet 可以使用标准的 Java 语句（而不是 SQL 语句）修改、添加或删除结果高速缓存中的行。更改至可以传播到下属关系表格的高速缓存。

(3) “查询”参数支持：用参数替代某些实际值，以将基本 SQL 查询定义为 Java 字符串。当查询运行时，数据访问 JavaBean 提供了用运行时生成的可用值来代替参数的方法。例如，参数值可能由用

户用 HTML 表格提交。

(4) 元数据支持：StatementMetaData 对象包含基本 SQL 查询。可以将高级别的数据（元数据）加入对象，以帮助将参数传递到查询中并使用返回的结果。“ 查询 ”参数可以是便于 Java 程序使用的 Java 数据类型。当查询运行时，会使用元数据规格将参数自动转换成适合 SQL 数据类型的表格。为了阅读查询结果，元数据规格可以将 SQL 数据类型自动转换成最便于 Java 应用程序使用的 Java 数据类型。

5．DA示例

```
// ********************************************************************
// * IBMDataAccessTest.java - 使用数据访问 JavaBean 的示例Servlet，简称DA示例的Servlet
// ********************************************************************
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
import java.math.BigDecimal;
import com.ibm.db.*;                    // 使用了IBM的软件包com.ibm.db
import com.ibm.servlet.connmgr.*;       // 继续使用IBM的连接管理器软件包com.ibm.servlet.connmgr

public class IBMDataAccessTest extends HttpServlet {
        // Use to communicate with connection manager.
    static IBMConnMgr connMgr = null;

    // Use later in init() to create JDBC connection specification.
    static IBMConnSpec spec     = null;         // the spec
    static String DbName        = null;         // database name
    static String Db            = "db2";        // JDBC subprotocol for DB2
    static String poolName      = "JdbcDb2"; // from Webmaster
    static String jdbcDriver    = "COM.ibm.db2.jdbc.app.DB2Driver";
    static String url           = null;         // constructed later
    static String user          = null;         // user and password could
    static String password      = null;         // come from HTML form
    static String owner         = null;         // table owner

    // Name of property file used to complete the user, password,
    // DbName, and table owner information at runtime.   ".properties"
    static final String CONFIG_BUNDLE_NAME = "login";

    // Single metaData object, used by all user requests, will be
    // fully defined in the init() method when the servlet is loaded.
    com.ibm.db.StatementMetaData metaData = null;     // 与CM示例不同，增加了变量metaData

    // ********************************************************************
    // * Initialize Servlet when it is first loaded                       *
    // ********************************************************************
public void init（ServletConfig config）throws ServletException {
```

```
super.init(config);
try {
        PropertyResourceBundle configBundle =
        (PropertyResourceBundle) PropertyResourceBundle.getBundle(CONFIG_BUNDLE_NAME);
        DbName    = configBundle.getString("JDBCServlet.dbName");
        url          = "jdbc:" + Db + ":" + DbName;
        user         = configBundle.getString("JDBCServlet.dbUserid");
        password = configBundle.getString("JDBCServlet.dbPassword");
        owner       = configBundle.getString("JDBCServlet.dbOwner");
} catch (Exception e) {
        System.out.println("read properties file: " + e.getMessage());
}

try {
                        // 第一步：创建连接规范
        spec = new IBMJdbcConnSpec(poolName, true, jdbcDriver, url, user, password);
                        // 第二步：获得连接管理器
        connMgr = IBMConnMgrUtil.getIBMConnMgr();
 }
 catch (Exception e) {
        System.out.println("set connection spec, get connection manager: " +
                                e.getMessage());
 }
    // 增加数据访问JavaBeans代码
 String sqlQuery = "SELECT ID, NAME, DEPT, COMM " +
                                "FROM " + owner + ".STAFF " +
                                "WHERE ID >= ? " +
                                "AND DEPT =    ? " +
                                "ORDER BY ID ASC";
    // Start defining the metaData object based on the query string.
 metaData = new StatementMetaData();
 metaData.setSQL(sqlQuery);
 try {
        // Add some more information to the metaData to make Java
        // programming more convenient.    The addParameter() method allows
        // us to supply an input parameter for the query using a
        // convenient Java datatype, doing a conversion to the datatype
        // actually needed by SQL.    The addColumn() method makes things
        // convenient in the other direction, retrieving data in a
        // datatype convenient for Java programming, doing a conversion
        // from the underlying SQL datatype.    The addTable() method
        // identifies the relational table and makes it possible for
        // result cache changes to be folded back onto the table.
        metaData.addParameter("idParm",     Integer.class,      Types.SMALLINT);
        metaData.addParameter("deptParm", String.class,       Types.SMALLINT);
```

```
            metaData.addColumn("ID",              String.class,       Types.SMALLINT);
            metaData.addColumn("NAME",            String.class,       Types.VARCHAR);
            metaData.addColumn("DEPT",            Integer.class,      Types.SMALLINT);
            metaData.addColumn("COMM",            BigDecimal.class, Types.DECIMAL);
            metaData.addTable("STAFF");
    } catch(DataException e) {
            System.out.println("set metadata: " + e.getMessage());
    }
}

   // ****************************************************************
   // * Respond to user GET request                                  *
   // ****************************************************************
public void doGet(HttpServletRequest req, HttpServletResponse res) {
    IBMJdbcConn cmConn     = null;
    Connection dataConn    = null;
    SelectResult result    = null;
    Try {
            // 第三步：获得与连接管理器的连接
            cmConn = (IBMJdbcConn)connMgr.getIBMConnection(spec);

            // 第四步：使用 CM 连接以访问预先建立的数据服务器连接
           dataConn = cmConn.getJdbcConnection();

            //第五步：和数据服务器的相互作用，但与CM示例不同
            // Make use of the externally managed connection dataConn gotten
            // through the connection manager.    Our dataAccessConn object
            // should be local (a new object for each request), since there
            // may be multiple concurrent requests.
            DatabaseConnection dataAccessConn = new DatabaseConnection(dataConn);
            // Begin building our SQL select statement - it also needs to be
            // local because of concurrent user requests.
            SelectStatement selectStatement = new SelectStatement();
            selectStatement.setConnection(dataAccessConn);
            // Attach a metadata object (which includes the actual SQL
            // select in the variable sqlQuery) to our select statement.
            selectStatement.setMetaData(metaData);
            // Make use of the facilities provided through the metadata
            // object to set the values of our parameters, and then execute
            // the query.    Values for dept and id are usually not hardcoded,
            // but are provided through the user request.
            String    wantThisDept    = "42";
            Integer wantThisId = new Integer(100);
            selectStatement.setParameter("deptParm", wantThisDept);
            selectStatement.setParameter("idParm", wantThisId);
```

```
        selectStatement.execute();
        // The result object is our cache of results.
        result = selectStatement.getResult();
        // Try an update on the first result row.    Add 12.34 to the
        // existing commission, checking first for a null commission.
        BigDecimal comm = (BigDecimal)result.getColumnValue("COMM");
        if(comm == null) {
            comm = new BigDecimal("0.00");
        }
        comm = comm.add(new BigDecimal("12.34"));
        result.setColumnValue("COMM", comm);
        result.updateRow();
        // Close the result object - no more links to the relational
        // data, but we can still access the result cache for local
        // operations, as shown in STEP 7 below.
        result.close();
    } catch (Exception e) {
        System.out.println("get connection, process statement: " +
                                    e.getMessage());
    }
          // 第六步：释放连接
    finally {
        if(cmConn != null) {
            try {
                cmConn.releaseIBMConnection();
            } catch(IBMConnMgrException e) {
                System.out.println("release connection: " + e.getMessage());
            }
        }
    }
     // 第七步：预备和发送响应，与CM示例有区别。
     res.setContentType("text/html");
     // Next three lines prevent dynamic content from being cached
     res.setHeader("Pragma", "no-cache");
     res.setHeader("Cache-Control", "no-cache");
     res.setDateHeader("Expires", 0);
     try {
        ServletOutputStream out = res.getOutputStream();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Hello DBWorld</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<TABLE BORDER>");
        // Note the use of the result cache below.    We can jump to
        // different rows.    We also take advantage of metadata
        // information to retrieve data in the Java datatypes that
```

```
            result.nextRow();
            out.println("<TR>");
            out.println("<TD>" + (String)result.getColumnValue("ID") + "</TD>");
            out.println("<TD>" + (String)result.getColumnValue("NAME") + "</TD>");
            out.println("<TD>" + (Integer)result.getColumnValue("DEPT") + "</TD>");
            out.println("<TD>" + (BigDecimal)result.getColumnValue("COMM") + "</TD>");
            out.println("</TR>");
            result.previousRow();
            out.println("<TR>");
            out.println("<TD>" + (String)result.getColumnValue("ID") + "</TD>");
            out.println("<TD>" + (String)result.getColumnValue("NAME") + "</TD>");
            out.println("<TD>" + (Integer)result.getColumnValue("DEPT") + "</TD>");
            out.println("<TD>" + (BigDecimal)result.getColumnValue("COMM") + "</TD>");
            out.println("</TR>");
            out.println("</TABLE>");
            out.println("</BODY></HTML>");
            out.close();
        } catch (Exception e) {
            System.out.println("HTML response: " + e.getMessage());
        }
    }
}
```

DA示例用了数据访问 JavaBean 来代替使用 java.sql 软件包中的类，以与数据库进行交互操作。而CM示例使用 java.sql 软件包中的类来与数据库进行交互操作。DA 示例的优点来自于连接管理器的优化和资源管理成为可能。程序员编写 DA 示例的优点来自于数据访问 JavaBean 提供的特点和功能。DA 示例仅有一小部分区别于 CM 示例。这里只讨论区别的部分。

CM 示例 Servlet 的步骤 1，2，3，4 和 6 在 DA 示例 Servlet 中未作更改。这些步骤包括了连接管理器的设置、从缓冲池中获得连接和将连接释放回缓冲池。对CM 示例的主要更改是：

(1) metaData 变量

该变量是在代码开始的“ 变量 ”段中加以说明的，它在所有方法之外。它允许对到达的用户请求使用单个实例。变量的说明是在 init() 方法中完成的。

(2) init() 方法

已将新的代码加入 init() 方法中，以便在初次装入 Servlet 时对 metaData 对象进行一次初始化操作。新的代码从将基本查询对象 sqlQuery 创建为 String 对象开始。注意 :idParm 和 :deptParm 参数的占位符号。 sqlQuery 对象指定了 metaData 对象内的基本查询。最后，提供高级别的数据（元数据）和基本查询给 metaData 对象，这将对运行查询和使用结果有帮助。代码显示了：

- addParameter() 方法注释，当运行查询时，为了便于 Servlet 进行操作，参数 idParm 将被支持为 Java Integer 数据类型，但当运行查询时，idParm 应该转换（通过 metaData 对象）以执行重要关系数据的 SMALLINT 关系数据类型的查询。
- deptParm 参数的使用相似于 addParameter() 方法，要注意相同的下属 SMALLINT 关系数据类型，第二个参数将会在 Servlet 中以不同的 Java 数据类型存在 - 如 String 类型而不是 Integer 类型。这样的话，参数可以是便于 Java 应用程序使用的 Java 数据类型，并可以自动地由 metaData 对象转换成和运行查询时所要求的关系数据类型一致的数据类型。
- addColumn() 方法执行一个类似于 addParameter() 方法的功能。对于要从关系表中检索的数据的

每个列， addColumn() 方法将关系数据类型映射到 Java 应用程序最便于使用的 Java 数据类型。从结果高速缓存中读取数据时和对高速缓存进行更改（然后对下属关系表进行更改）时会使用映射。

● addTable() 方法明确地指定了下属关系表。如果将更改到结果高速缓存传播至下属关系表，那么就需要信息。

(3) 步骤 5

步骤 5 已经重新写入使用数据访问 JavaBean 以执行 SQL 查询来代替执行 Java.sql 软件包中的类。查询使用 selectStatement 对象运行，该对象是一个 selectStatement 数据访问 JavaBean。步骤 5 是对用户请求的响应处理的一部分。当步骤 1 至 4 已运行时，可以使用连接管理器的 dataConn 连接对象。代码显示了：

● 创建 dataAccessConn 对象（数据库连接 JavaBean）以建立数据访问 JavaBean 和 数据库连接（dataConn 对象）之间的链接。

● 创建 selectStatement 对象（选择语句 JavaBean），用于通过 dataAccessConn 对象指向数据库连接，并通过 metaData 对象指向查询。

● 查询是用 setParameter() 方法指定参数来“ 完成 ”的。

● 查询是用 execute() 方法来执行的。

● 结果对象（SelectResult JavaBean）是一个包含查询结果的高速缓存，它是用 getResult() 方法创建的。

● 数据访问 JavaBean 提供了一套丰富的用于使用结果高速缓存的特性 - 就这一点，代码显示了如何用标准 Java 代码更新结果高速缓存（和下属关系表 ）中的第一行，而不需要使用 SQL 语法。

● close() 方法切断了结果高速缓存和下属重要关系表间的链接，但是高速缓存中的数据仍可由 Servlet 进行本地访问。使用 close() 方法后，数据库连接就不是必需的了。步骤 6（未对 CM 示例 Servlet 作修改 ）中将数据库连接释放回连接管理器缓冲池，以供另一个用户请求使用。

(4) 步骤 7

步骤 7 已完全重写（对照 CM 示例），以使用步骤 5 中检索出的查询结果高速缓存，来准备发送给用户的响应。结果高速缓存是一个选择结果数据访问 JavaBean。虽然结果高速缓存不再链接至下属关系表，但它仍可以由本地进程访问。在该步骤中，准备响应并将它发送给用户。代码显示了如下内容：

● nextRow() 和 previousRow() 方法用于浏览结果高速缓存。可以使用其它浏览方法。

● getColumnValue() 方法用于从结果高速缓存中检索数据。由于先前在创建 metaData 对象时的属性设置，因此可以很容易将数据造型成便于 Servlet 使用的格式。

7.2 会话编程

1．会话跟踪程序

会话是起源于同一浏览器上同一用户的一系列请求。使用该会话跟踪框架，服务器就可以保持会话的状态信息。处理会话跟踪的类是IBMSessionContextImpl（在IBM的软件包com.ibm.servlet. personalization.sessiontracking中）。缺省情况下，通常由WebSphere应用服务器激活它。IBM的软件包com.ibm.servlet.personalization.sessiontracking 包括下列3个类和1个接口：

● IBMSessionData （该类扩展了com.sun.server.http.session.SessionData）

● IBMSessionContextImpl （该类扩展了com.sun.server.http.session.SessionContextImpl）

● IBMHttpSessionBindingEvent（该类扩展了javax.servlet.http.HttpSessionBindingEvent，并实现了Serializable，用于会话群集）

● IBMHttpSessionBindingListener（扩展了HttpSessionBindingListener和Serializable，用于会话群集）当用户首次发出请求时，便启用了会话跟踪程序，并创建了 HttpSession 对象，且将会话标识符作为一个 cookie 发送至浏览器。在后继请求中，浏览器将会话标识符作为一个 cookie 送回给用户，会话跟踪程序用它找到与该用户关联的 HttpSession 。当 HttpSession 包含用于实现

IBMHttpSessionBindingListener 的对象时，当实现侦听程序接口的对象与会话连接或断开连接时，会话跟踪程序会通知该对象。由于 HttpSession 对象自己是作为 IBMHttpSessionBindingEvent 的一部分进行传递的，所以 IBMHttpSessionBindingListener 接口允许您在除去会话之前，保存会话的任何部分。这与自动结束会话还是由指定的请求结束会话无关。

下面说明在集成的 Web 服务器环境中会话跟踪是如何进行工作的：

(1) 每当用 HttpServletRequest 实现的 getSession 方法获得一个会话时，拥有会话的主机在会话上加一个锁并将会话传播到所需的地方。在任何会话修改和 HttpServletRequest 实现的服务方法结束后，会话将自动发送回服务器以更新会话的副本并释放会话中所加的锁。可以将使用 HTTP 请求获得的会话作为 Servlet 的当前会话或相关联的会话，或是当前 HTTP 请求拥有的会话。

(2) 还可以用 HttpSessionContext（IBMSessionContextImpl）实现的 getSession 方法获得会话。Servlet 可以先访问 HttpSessionContext 实现的 getIds 方法，再用上下文中的 getSession 方法以访问其它几个与 HTTP 请求关联的会话。当用这种方法获得请求时，程序员必须用 IBMSessionContextImpl 的 同步化（synchronized）方法，手工地对会话解锁。同步化方法也会自动更新会话的拥有者主机的版本。

(3) 当 HttpSessionBindingListener 和 HttpSessionBindingEvent 用于集成的 Web 服务器环境时，在WebSphere应用服务器中将在会话驻留的地方激发该事件。如果会话超时，那么这个位置将是拥有者主机。如果通过调用会话对象的无效方法使会话无效，那么该位置可能是会话群集客户机或拥有者主机。如果用 removeValue() 方法除去会话对象，则将在主机上使用调用的地方激发该事件。要提高获取会话无效事件通知单的能力，请使用 IBMHttpSessionBindingListener 和 IBMHttpSessionBindingEvent，它们提供了设置和检索会话群集客户机或会话群集服务器的主机名时所需的扩展名，以接收通知单。

(4) 在将WebSphere应用服务器的一个实例配置成一个会话群集客户机后，所有其它配置参数（除了启用会话支持参数和会话交换参数）都由指定的会话群集服务器进行访问。参数设置的本地副本保留在会话群集客户机的磁盘上。当会话群集客户机将会话跟踪程序更改为另一总模式时，会话跟踪程序将使用参数设置。这种设计保证了会话群集配置的一致性。

(5) 在当前的 Java Servlet API（如 Sun Microsystems 指定的）中，HttpSession 接口的 putValue() 方法的定义对集成环境的可能性不做解释。如果添加一个不能实现会话的可串行化接口的对象，则将无法用给定的对象来传播对象。当客户机中做了会话更新时，将不能从群集客户机发送对象至群集服务器。要使应用程序在集成环境中可移动，则必须将所有对象放置在可串行化的会话中。

(6) 在会话群集环境中，添加到会话中的 Java 对象必须在每个群集主机和客户机的WebSphere应用服务器类路径中。正如前面所讨论的，这种对象将被串行化（即存储到磁盘上）。

2．使用 HttpServletRequest 创建会话

下面结合示例（SessionSample.java），介绍使用 HttpServletRequest 创建会话的编程技术。示例的程序清单如下：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public
class SessionSample extends HttpServlet {
public void doGet (HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
        // 第一步：获取 HttpSession 对象
        boolean create = true;
        HttpSession session = request.getSession(create);
        // 第二步：在会话中存储并检索用户定义的数据。
       Integer ival = (Integer) session.getValue ("sessiontest.counter");
```

```
        if (ival == null) ival = new Integer (1);
        else ival = new Integer (ival.intValue () + 1);
        session.putValue ("sessiontest.counter", ival);
          // 第三步：输出一个包含 HttpSesson 对象的数据的 HTML 响应。
        response.setContentType("text/html");
        out.println("");
        if(firstNameList.isEmpty()) {
            out.println("<head><title>Session Tracking Test</title></head>");
            out.println("<body>");
            out.println("<h1>Session Tracking Test</h1>");
            out.println ("You have hit this page " + ival + " times" + "<br>");
            out.println ("Your " + request.getHeader("Cookie"));
            out.println("</body></html>");
        }
}
```

编程步骤如下：

(1) 获取 HttpSession 对象。

使用 HttpServletRequest 对象的 getSession() 方法来获得会话。当首次获得 HttpSession 对象时，会话跟踪程序会创建一个唯一的会话标识符并典型地将它作为 cookie 发送回浏览器。该用户（在同一浏览器上）的每个后继请求会传递 cookie 。会话跟踪程序用会话标识符来查找用户的现有 HttpSession 对象。

在代码示例的第一步中，将 Boolean 设置为真，这样如果未找到HttpSession，便会创建HttpSession。

(2) 在会话中存储并检索用户定义的数据。

会话建立后，就可以将一个检索用户定义的数据添加到会话中。HttpSesson 对象中添加、检索和除去Java 对象的方法类似于 java.util.Dictionary 中的方法。通常用 HttpSession.putValue() 来连接对象和会话，并用 HttpSession.removeValue() 来切断对象和会话之间的连接。会话的失效也暗示着将切断当前所有会话与对象的连接。如果启用了会话持续性（缺省设置）或在会话群集环境中，请确保将要添加到会话中的 Java 对象（.class, .jar 和 .ser 文件）放置在WebSphere应用服务器类路径。

在代码示例的第二步中，Servlet 从 HttpSession 对象中读取一个整数对象，对其增量，并将之写回。可以用任何名称(例如示例中的 sessiontest.counter)来标识 HttpSession 对象中的值。因为 HttpSession 对象是由用户可访问的 Servlet 共享的，所以建议采用站点端的命名约定规则，以避免 Servlet 之间的共享冲突。

(3) 输出一个包含 HttpSesson 对象的数据的 HTML 响应（可选的）。

在代码示例的第三步中，Servlet 生成了一个 Web 页面，每次用户在会话期间访问该页面时，它就会显示 sessiontest.counter 的值。

(4) 使用 IBMHttpSessionBindingListener 接口，以在会话结束时保存数据（可选的）。

会话结束前，将通知存储在实现 IBMHttpSessionBindingListener 接口会话中的对象会话即将终止。这就允许执行后续会话处理，例如将数据保存到一个数据库中。HttpSessionBindingListener 接口提供了下列方法，必须在实现接口时实现这些方法：

- valueBound()；将对象连接到会话时调用该方法
- valueUnbound()；切断对象与会话间的连接时调用该方法

IBMHttpSessionBindingListener 接口提供了下列方法，用自己的程序来覆盖这些方法：

- setReceiverHostname()；设置启动事件的主机名，并支持对会话群集环境中会话失效事件的响应。
- getReceiverHostname()；返回接收器主机名。

(5) 结束会话

可由会话跟踪程序或指定的 Servlet 处理自动终止一个会话。

7.3 PageListSevlet 技术

PageListSevlet 技术是 Webphere 应用服务器特有的一种 Servlet 技术。

1．使用 XML 文件来配置 Sevlet

Webphere 应用服务器支持使用 XML 文件来配置 Sevlet，这些配制文件称为 XML Servlet 配置文件。一个 XML Servlet 配置文件的示例如下：

```
<?xml version="1.0"?>
<servlet>
  <code>SimplePageListServlet</code>
  <description>Shows how to use PageListServlet class</description>
  <init-parameter name="name1" value="value2"/>
  <page-list>
    <default-page>
      <uri>/index.jsp</uri>
    </default-page>
    <error-page>
      <uri>/error.jsp</uri>
    </error-page>
    <page>
      <uri>/TemplateA.jsp</uri>
      <page-name>page1</page-name>
    </page>
    <page>
      <uri>/TemplateB.jsp</uri>
      <page-name>page2</page-name>
    </page>
  </page-list>
</servlet>
```

页面列表（<page-list>）描述了 servlet 调用的那些 JSP 文件的抽象名，从而避免了将被调用的 JSP 文件的 URL 编码到 servlet 中。页面列表还可以包含一个缺省页面、出错页面和其它根据 HTTP 请求而调用的 JavaServer 页面。

XML Servlet 配置文件（如名为 SimplePageListServlet.servlet 的 XML 文档）包含如下内容：

- Servlet 类文件的文件名
- Servlet 说明
- Servlet 初始化参数
- 包含 Servlet 可以调用的 JavaServer 页面的 URL（统一资源定位器）的页面列表。该页面列表可以包括一个缺省页面、一个出错页面、一个或多个已装入的目标页面（如果它们的名称出现在 HTTP 请求中）。

该 .servlet 文件存储在 WebSphere 应用服务器的 servlets\ 目录下，或 Java 类路径目录下。当 WebSphere 应用服务器接收到一个对 Servlet 实例的请求时，它将在 servlet.properties 文件中查找该 Servlet 的配置信息。当使用 WebSphere 应用服务器的管理器配置 Servlet 初始化参数和装入选项时，管理器将把这些设置信息存储在 servlet.properties 文件中。如果 WebSphere 应用服务器在 servlet.properties 文件中没有找到

所请求的 servlet 的配置信息，它将检查 servlets\ 目录和 Java 类路径下属于该 Servlet 实例的 .servlet 文件。当 WebSphere 应用服务器找到 Servlet 配置信息时，它将装入该 Servlet 实例。

在下列情况下，XML Servlet 配置文件是十分有用的：

● Servlet 使用 callPage() 方法调用 JSP 时。为了调用 JSP，XML Servlet 配置文件的 page-list 元素和 PageListServlet 类（将在下一章节中讨论）消除对 URL 硬编码。如果引用的页面发生更改时，只需更新 .servlet 文件，而无需更新 Servlet 代码和重新编译该 Servlet。每当 .servlet 文件发生更改时，WebSphere 应用服务器就会自动装入 Servlet 实例。

● 希望将一个 Servlet 的配置信息与其它的 Servlet 的配置信息分隔开时。每当使用 WebSphere 应用服务器的管理器来配置 Servlet 初始化参数和装入选项时，管理器将把这些设置信息存储在 servlet.properties 文件中。如果需分隔一个 Servlet（为了易于管理安全性、展开或除去应用程序）时，XML Servlet 配置支持是十分有帮助的。

WebSphere Application Studio 提供了用于生成 Servlet 的向导。这些向导实现了 XML Servlet 配置（即生成了一个 .servlet 文件）。在没有这个工具的情况下，可以创建一个扩展 PageListServlet 类的 Servlet，并使用文本编辑器创建一个 XML Servlet 配置文件，或使用 XMLServletConfig 类来为 Servlet 实例创建一个 XML Servlet 配置文件。必须注意的是，目前，使用 XML Servlet 配置文件的 Sevlet 必须是 PageListServlet 的子类或子孙类。

2．使用 PageListServlet 类

PageListServlet 包含一种 callPage() 方法，该方法调用了一个 JavaServer 页面，并将其作为对页面列表中某个页面的 HTTP 请求的响应。PageListServlet callPage() 方法接收 在 XML 配置文件中的页面名、HttpServletRequest 对象、和 HttpServletResponse 对象。与此相反，HttpServiceResponse 类的 callPage() 方法通常接收一个 URL 和 HttpServletRequest 对象。

SimplePageListServlet 是一个扩展 PageListServlet 类的 Servlet 示例：

```
public class SimplePageListServlet extends com.ibm.servlet.PageListServlet {
    public void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    {
        try{
            setRequestAttribute("testVar", "test value", req);
            setRequestAttribute("otherVar", "other value", req);

            String pageName = getPageNameFromRequest(req);
            callPage(pageName, req, resp);
        }
        catch(Exception e){
            handleError(req, resp, e);
        }
    }
}
```

请注意：PageListServlet 在软件包 com.ibm.servlet 中，该类扩展了 javax.servlet.http.HttpServlet。

3．创建 XML Servlet 配置文件

可以使用文本编辑器创建一个 XML Servlet 配置文件，或使用 XMLServletConfig 类来为 Servlet 实例创建一个 XML Servlet 配置文件。可以编写一个使用 XMLServletConfig 类的 Java 程序，以生成 Servlet 配置文件。 XMLServletConfig 类提供了关于设置和获取 file 元素和其内容的方法。

4．部署 Servlet 和 .servlet 文件

为了在 WebSphere 应用服务器上使用经编译的 Servlet 和其 XML Servlet 配置文件：

(1) 将经编译的 Servlet 和其 .servlet 文件放置在下列任一路径中：

(2) applicationserver_root\servlets，其中 applicationserver_root 是 Application Server 安装的根目录。如果 Servlet 有软件包名称， 则请确保该 Servlet 被放置在正确的 servlets\ 子目录下。

(3) 一个在 Application Server 管理器 Java 设置屏幕上的“应用服务器 Java 类路径”字段中指定的目录。

(4) 请确保在页面列表中引用的 JSP 文件位于 Web 服务器的 HTML 文档目录中。

(5) 如果希望 WebSphere 应用服务器在启动时自动装入 Servlet 实例，请使用管理器添加 Servlet 实例名称，并指定在启动时装入。

WebSphere 应用编程技术涉及 Java Servlet API 和 WebSphere 应用服务器 API，请参阅有关文档获得更详细的信息。另外，附录中的实验指导带您一起走过使用 WebSphere 技术开发电子商务应用的过程。

参考文献

1. G. Cornell, C. S. Horstmann. Core Java. SunSoft Press, 1996.
2. Marc Abrams, ed. World Wide Web - Beyond the Basics, Prentice Hall, 1998.
3. John Akerley etc. Developing an e-business Application for the IBM WebSphere Application Server. International Technical Support Organization，1999。（http://www.redbooks.ibm.com）
4. Jurgen Friedrichs etc. Java Thin-Client Programming. International Technical Support Organization，1999。（http://www.redbooks.ibm.com）
5. Goldfarb, Paul Prescod. The XML handbook. Prentice Hall, Upper Saddle River, NJ 07458, 1998.
6. Sean McGrath. XML by Example: Building E-Commerce Applications. Prentice Hall, Upper Saddle River, NJ 07458, 1998.
7. Web上的参考站点：
   - Internet FAQ Consortium：http://www.faqs.org/。
   - WWW联盟(World Wild Web Consortium)：http://www.w3.org/
   - 面向对象信息资源：http://iamwww.unibe.ch/~scg/OOinfo/
   - 对象管理组织：http://www.omg.org/。
   - IBM 电子商务：http://www.ibm.com/e-business/。
   - Java 技术：http://java.sun.com/。
   - 结构化信息标准发展组织（Organization for the Advancement of Structured Information Standards）：http://www.oasis-open.org。
   - XML 组织：http://xml.org/。
   - IBM WebSphere：http://www-4.ibm.com/software/webservers/

附录 I IBM WebSphere Studio 实验以及应用开发实验的设置指导

特别提醒

1. 请在开始以前读完所有指导。确信你能得到所有需要的软件，理解正确设置的所有步骤。
2. 设置的步骤会根据你是创建一个要在多个机器上复制的映像（master image）还是只在一个机器上安装

而不同，参看这个文件末尾的“其它考虑”。

3. 我们强力推荐：只要有可能，就安装 WebSphere Application Server 高级版和 VisualAge for Java 企业版。
4. 本实验需要 3 个压缩文件：WasDESLabfiles.zip、WasDEVLabfiles.zip、JKToysSite.zip。

引言

WebSphere Studio 实验以及应用开发实验可以在同一个映像上进行，因为这两个实验的缘故，建议所有东西都安装在 C 盘上。

一些有用的提示

1.包含空格的长文件名必须被括在引号里。例如，cd “websphere workshop”将目录改变到 WebSphere Workshop。注意：大多数路径结构是大小写无关的，然而为了安全起见，还是用正确的大小写。
2.为了避免打字错误，你可以在你的浏览器里打开这个文件，用编辑菜单的拷贝和粘贴功能来传递特定的信息到别的安装地点。
3 **不要用 NotePad**，而要用 WordPad 作为你的编辑器。
4.安装一个解压缩产品如 WinZip，该软件可以从http://www.winzip.com得到。
5.确信你的显示器的分辨率为 1024X768，否则你也许会在安装 VisualAge for Java 时遇到麻烦。

需要的软件

需要为这个工作室安装的软件包括：

1. Windows NT 4.0 Workstation 或 Server，带 Service Pack 3 或 Service Pack 4
2. Netscape 最新版本(可以从 http://www.netscape.com/download/ 得到)
3. IBM DB2 UDB (Version 5.2)
   1) 可以从 IBM 内部网 ftp://ftp3.torolab.ibm.com/pub/db2install/db2_v520/NTEE/ 得到
   2) 在外面的 Internet 上的是: http://www.software.ibm.com/data/db2/udb/downloads.html
4. Sun Microsystems 的 JDK 1.1.7 (或 1.1.8)。 (不要下载 JDK 1.2.x) JDK 可以从 http://java.sun.com/products/jdk 下载 JDK 和 API 文档
5. WebSphere Application Server V2.0 Advanced Edition (GA)
6. Lotus Domino Go Web server (V 4.6.2.5) - IBM WebSphere Studio 1.0 Shrink wrap 里有
7. WebSphere Studio V1.0 (GA)
   1) IBM 内部网 http://wsstudio.raleigh.ibm.com 上有
   2) 外部 Internet 上 http://www.software.ibm.com/webservers 有
   3) 这里下载的产品包括：
      (1) NetObjects Fusion 3.0.1 (contained in WebSphere Studio)
      (2) NetObjects BeanBuilder 1.0 (包含在 WebSphere Studio 里)
      (3) NetObjects ScriptBuilder 3.0 (包含在 WebSphere Studio 里)
      (4) VisualAge for Java V2.0 Professional Edition (包含在 WebSphere Studio 里)
         - VisualAge for Java Professional Update (http://www.software.ibm.com/vadd) (ProUpNt.zip)
   4) 可选： (VisualAge for Java Enterprise)
      - VisualAge for Java Enterprise Update (http://www.software.ibm.com/vadd) (EntUpNt.zip)
8. 实验练习和设置的材料：
   1) WebSphere Studio Workshop Lab Files (WasDesLabfiles.zip)
   2) WebSphere Developer Workshop Lab Files (WasDevLabfiles.zip)

3) JKToys Web site for the Developer Workshop (JKToysSite.zip)

如果你是一个 IBM 员工，你可以从 IBM 内部网上得到一些软件。如果你不能访问到 IBM 内部网，你必须购买那些不能免费下载的软件。

需要的硬件（至少）

1. P200+
2. 128 Mb 内存
3. 2 gig DASD (C 盘 2 gig)
4. 连接到 LAN 上，用来复制
5. 单独的上课环境（可选）
6. 建议一个学生一台机器

本指导分成下列 4 个部分：
第 1 部分：两个实验都需要的设置；
第 2 部分：WebSphere Studio 实验的设置和确认
第 3 部分：WebSphere 应用开发实验的设置和确认
第 4 部分：其它考虑

第 1 部分 两个实验都要的设置

1.操作系统

Windows NT Workstation 或 Server 4.0 (带 Service Pack 3 或 4)标准安装，并安装网络。每一台机器应该有自己的 IP 地址和单独的工作站名字（为了用微软网络）。如果机器单独运行（不在一个网络里）：加 MS Loopback Adapter 或别的 LAN 适配器:

1. 设置 IP address 1.2.3.4
2. 设置子网掩码 255.0.0.0
3. DNS 设置里设置域名为 "ibm.com"
4. 重启后确认 Loopback，输入 "netstat -r":

活动路由显示:				
Network Address	Netmask	Gateway Address	Interface	Metric
1.0.0.0	255.0.0.0	1.2.3.4	1.2.3.4	1
1.2.3.4	255.255.255.255	127.0.0.1	127.0.0.1	1
1.255.255.255	255.255.255.255	1.2.3.4	1.2.3.4	1
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
224.0.0.0	224.0.0.0	1.2.3.4	127.0.0.1	1
255.255.255.255	255.255.255.255	1.2.3.4	1.2.3.4	1

确认 Display Desk Top Area 设为 1024 x 768 象素点。
改变虚拟内存参数：

1. 双击"我的电脑"
2. 双击"控制面板"
3. 双击"系统"
4. 单击"性能"

5. 在系统参数窗口单击“ 虚拟内存 ”改变框
6. 改变初始尺寸 (MB)为 150-200
7. 改变最大尺寸为 (MB) 200-300
8. 单击“ 设置 ”
9. 单击“ 确认 ”
10. 在系统属性窗口单击“ 确认 ”

创建路径“ WebSphere Workshop”

在命令提示创建子路径: 输入 md "WebSphere Workshop" 来保存下载的文件。

增加新的用户名

为了增加新的用户名，单击开始—程序—管理工具（公用）—用户管理。单击用户—新用户.....

1) 用户名=USERID –确信你输入大写
2) 口令=PASSWORD – 确信你输入大写
3) 确认口令=PASSWORD – 确信你输入大写
4) 不选 "用户必须改变口令"
5) 选 "Password Never Expires"
6) 按"组"按钮，增加 NT 管理员权限
7) 按“ 确认 ”
8) 按“ 确认 ”
9) 关闭用户管理
10) 退出登录，Atl-Ctl-Del 按 "退出登录..."; 登录 USERID 和 PASSWORD。

2. WinZip 7.0

运行 winzip70.exe 文件，缺省安装。

3. Netscape

缺省安装。

4.DB2 UDB (如果安装 WebSphere Standard Edition 才作这一步)

从 CD 安装 DB2 Universal Database Workgroup Edition (setup.exe)

1) 安装 UDB 选择 "DB2 Universal Database Workgroup Edition"
2) 按“ 典型 ”安装
3) 当被要求 DB2 管理员 id/口令,使用 上一步设置的"USERID" 和 "PASSWORD"
4) 安装后选择"是的, 现在我要重启我的计算机”
5) 重启后创建 SAMPLE 数据库
6) 把命令中心移到桌面，以免重启后启动
   (1) 开始->寻找->文件和文件夹
   (2) 输入 "startup"回车
   (3) 双击"C:\WINNT\PROFILE\..."
   (4) 把控制中心拖到桌面
   (5) 关闭所有窗口

7) 安装后清空 c:\temp 子目录。

5.IBM Java Development Kit 1.1.7

运行 jdk117-win32.exe (安装 JDK 的自解压文件). (只选择 Program Files; Library and Header Files; Applets)
这样只有下面的组件被安装：

1) Program Files
2) Library and Header Files
3) Demo Applets

解压 jdk1.1.x 文档文件。

6.Lotus Domino Go Webserver Version 4.6.2.5 只安装:

1) Lotus Domino Go Webserver
2) Security Files
3) NT Service

当被要求 ID 和口令时，回答 "admin", "admin"

7.IBM WebSphere Application Server 2.0 Advanced Edition

安装 WebSphere Application Server

1) 选择缺省目的目录(c:\WebSphere\AppServer)
2) 选择所有 Application Server 组件
3) 选择安装过的 JDK (也许已经高亮了)
4) 选择 Application Server 插件(IBM HTTP Server, 选择 version 1.3.3 和 Lotus Domino Go WebSphere 4.6.2.5)

在安装过程里，IBM HTTP Server 1.3.3 将被安装: 接受设置文件的缺省目录
IBM DB2 UDB 也会在这个过程里被安装(只是 Advanced Edition)。

8.设置 IBM HTTP Server

1) 打开 Windows NT 服务 "面板". (开始->设置->控制面板->服务)
2) 卷屏，在服务列表里找到 IBM HTTP Server。在这个服务上双击。
3) 在启动类型, 选择"手工"; 然后按 OK
4) 测试 HTTP Server，从命令行启动
   (1) c:
   (2) cd Program Files\IBM HTTP Server\
   (3) Apache
5) 如果 HTTP Server 启动失败，可能需要作以下 2 个修改：
   (1) 从服务面板, 在 IBM HTTP Server 的启动下输入 USERID 和 PASSWORD 作为"这个帐号"信息。
   (2) 编辑 c:\Program Files\IBM HTTP Server\conf\httpd.conf 文件；找到有 :#ServerName somename.host.com 的行，把它改为: ServerName "hostname as found on network host identification entry" (比如：ServerName student1)

9.IBM WebSphere Studio V1.0

为 WebSphere Studio 运行 setup.exe 程序，下面的组件应该被安装:

1) IBM WebSphere Studio V1.0
2) NetObjects BeanBuilder 1.0
3) NetObjects Fusion V3.0.1 (对 Fusion 3.0.1 当要保存以前的 Fusion 站点时选择 “ 不 ”，选择典型安装)
4) VisualAge Java Professional V2.0 (推荐安装 VisualAge for Java 的 Enterprise Edition，安装 Studio 附带的 Professional Edition 是最后的手段)

安装后选择 “ 是，我要现在重启机器 ”。确信你能从各自的开始.程序菜单项目里启动以下的程序:

1) WebSphere Studio (c:\Websphere\Studio\bin\WSStudio.exe)
2) NetObjects BeanBuilder (c:\Websphere\BeanBuilder\bin\bbuilder.exe)
3) NetObjects ScriptBuilder (c:\Websphere\ScriptBuilder\SBuilder.exe)
4) NetObjects Fusion (c:\NetObjects Fusion 3.0.1\Fusion.exe)

如果任何一个开始.程序快捷方式不见了，请确认这个程序真的安装了，创建一个可执行程序的快捷方式（见上），放到桌面上。

10.IBM VisualAge for Java V2.0 Enterprise Edition (如果有的话，第 6 步里不要安装 Java Pro)

运行 setup.exe 开始安装 VAJava：

1) 选择完全安装
2) 选择"本地库"选项
3) 启动 VisualAge for Java 来完成安装。

11.我的电脑。

双击我的电脑图标，打开目录文件夹。选择查看—>选项。
在文件夹页面选择第二个按钮，标记为：用一个窗口浏览文件夹.....
在查看页面，确信只有以下的项目被选择：

1) 显示所有文件
2) 在标题条显示完整的路径

按确认按钮来保存。

第 2 部分 WebSphere Studio 实验的设置和确认

设置 IBM WebSphere Studio 实验

确信你用管理员组的 USERID PASSWORD 登录.

用 WinZip 解压 c:\WebSphere Workshop\WasDESLabfiles.zip 到 c:\.
从命令窗口启动 DB2，运行 c:\JKToys Workshop\setup\intranet\db2\db2demo

1. 打开一个 DB2 命令行处理器窗口，开始->程序 ->DB2 for Windows NT->命令窗口
2. cd c:\JKToys Workshop\setup\intranet\db2\
3. 输入 db2demo 回车

为了使以上的步骤生效，启动 DB2 命令中心 (开始->程序->DB2 for Windows NT->管理工具->控制中心);

按 "+"号指导本地数据库显示出来。你应该看到 SSDEMO01 数据库.

确认 IBM WebSphere Studio 实验

1. 确认 DB2 服务自动启动，用服务面板:开始->设置->控制面板->服务->DB2-DB2.
2. 停止 IBM HTTP Server (启动-> 程序-> IBM HTTP Server -> 停止 Server)
3. 用 c:\JKToys Workshop\go-intra.bat 启动 web 服务器。选择开始->运行; 按“ 浏览 ”找到 c:\JKToys Workshop\go-intra.bat; 按确认来运行
4. 检查现存的站点
   1) 打开 Netscape，浏览 URL http://<your_hostname>:8080/ (your_hostname 是你的机器的名字,你可以用 127.0.0.1 或者你可以在一个命令提示下输入 hostname 或者按 Ctrl-Alt-Del 找到你的机器的名字)或者在命令提示输入“ hostname”。 注意：只有在你已经安装了非 MS Loopback 的网络适配器，它在 WINNT 子目录下创建了一个主机表的情况下，才会工作正常。
   2) 按市场 & 销售 然后按 JKToys 销售， 为 Quarter 输入 Q1; 按提交
   3) 按雇员服务，按雇员查看，为姓输入 D，按提交

第 3 部分 WebSphere 应用开发实验的设置和确认

设置 IBM WebSphere 应用开发实验

确信你用管理员组的 USERID 和 PASSWORD 登录。
用 WinZip 解压缩 c:\WebSphere Workshop\WasDEVLabfiles.zip 到根目录 c:\.
用 WinZip 解压缩 c:\WebSphere Workshop\JKToysSite.zip 到根目录 c:\.

1. 从命令提示输入 cd c:\WASDevDBSetup，输入 db2cmd dbsetup.cmd。为了让以上步骤生效，启动 DB2 命令中心 (开始->程序->DB2 for Windows NT->管理工具->控制中心)；按"+"号直到本地数据库出现。你应该看到 JKDB 和 JKTOYS 数据库。
2. 安装 VisualAge for Java Rollup2Obtain，安装文件 rollup2_vaj20_win.zip。这个文件解压缩到 VisualAge 的安装盘上的 IBMVJava 目录。解压缩后，启动 VisualAge 完成安装。
3. VisualAge for Java update, March 1999 version.
   1) 为了升级 VisualAge for Java Enterprise, 解压缩 enu399nt.zip 到 c:\updateinstall. 然后，执行 updateinstall 目录里的 setup.exe。
   2) 为了升级 VisualAge for Java Professional, 解压缩 pru399nt.zip 到 c:\updateinstall. 然后，执行 updateinstall 目录里的 setup.exe。
4. 创建一个桌面图标来启动 VA Java (拖一个 c:\IBMVJava\IDE\PROGRAM\IDE.EXE 的拷贝到桌面).
5. 从这个桌面图标启动 VAJava. (在启动时，要载入一些 VAJava 特性).
6. 载入两个特性到 VA Java 库.
   1) 从 VA Java Workbench 菜单选择文件 -> 快速启动
   2) 选择特性，从快速启动对话框增加特性(按确认)
   3) 选择 JSP 运行监视器和 IBM WebSphere 测试环境特性 (按确认)
7. 导入 WASDev 项目，把它加到 VisualAge for Java 环境。那个文件 waslab6.dat 在 c:\ directory
8. 将 waslab6.dat 导入到 VisualAge for Java 仓库：
   1) 文件->载入; 单击库，选择一个载入源;
   2) 单击 Next; 单击浏览;
   3) 单击"c:\";
   4) 选择 waslab6.dat; 单击详细情况;

5) 选择 WASDev; 单击确认；单击结束.

9. VisualAge for Java 的下一个设置步骤是设置资源路径.从工作台选择"窗口"菜单，选择 "选项"，单击 "资源" tab（如果它还没有高亮），按"编辑"按钮，插入以下的语句:
   1) C:\Program Files\sqllib\java\db2java.zip; (如果 WebSphere Standard Edition 被安装的话，这将是 c:\sqllib\java\db2java.zip)

   按“ 确认 ”。按"应用"，按"确认".
10. 增加项目到 VisualAge Java 环境:
    1) 按"增加新的或者已有的项目到 Workspace"; (项目文件夹在"奔跑的人"图标的右边;把鼠标拖过工具条上的图标来显示描述.)
    2) 按"增加项目到库”;
    3) 往下卷屏，选择 WASDev;
    4) 只选择 [Startup Labs V2.0];
    5) 按结束.
11. 选择类 SERunner
    1) 按项目边上的+号- IBM WebSphere 测试环境
    2) 定位包 com.ibm.servlet -- 按+ 号
    3) 选择类 SERunner

    在类 (SERunner)上右击，选择属性，在项目路径行，按编辑按钮选择 WASDev.
12. 打开 JSP 执行监视器里的语法错误报告
    1) 选择菜单：工作间->工具-> JSP 执行监视器; (停顿以下，一个对话框会显示)；选择标有"检索语法错误信息" 的 checkbox，按确认。
13. 解压缩 c:\properties.zip 到 c:\ 这放置了运行‘ 应用开发实验 ’的更新特性文件，让 IBM WebSphere 测试环境监听端口 80。

确认 IBM WebSphere 开发者工作室

1. 你需要确认网站既能通过 IBM HTTP Server 运行，又能在 VisualAge for Java 里运行
   1) WebSphere 内部设置检查
   2) 启动 IBM HTTP Server webserver (开始->程序-> IBM HTTP Server -> 启动 Server) .
   3) 启动 WebSphere Application Server Admin GUI (开始-> 程序-> IBM WebSphere -> Application Server V2.0 ->管理).
   4) 输入用户名/口令 (都是 admin) –按登录.
   5) 在下一屏,在左边的面板选择设置, Java 引擎
   6) 在右边的面板,确信下面的被输入 (确信 c: 是你安装的正确驱动器字母!)
      (1) Application Server Classpath: 确信 C:\Program Files\sqllib\java\db2java.zip 在类路径的最后 (如果你安装了 Standard Edition of the Application Server，它将不会在那儿，你需要增加 c:\sqllib\java\db2java.zip 到类路径的最后.)
      (2) Java Libpath: c:\jdk1.1.7\bin
      (3) Java Path: c:\jdk1.1.7\bin
      (4) 按保存
2. 下面测试网站，输入下面的 URL: http://127.0.0.1/JKToys/index.html 你应该看到 JKToys Internet Site.
3. 关闭 IBM HTTP Server (开始->程序 ->IBM HTTP Server->停止 Server)
4. 启动 VisualAge Java 开始->程序 ->IBM VisualAge Java for Windows->IBM VisualAge for Java.
5. 选择"SERunner" 类
   1) 按项目边上的+号 - IBM WebSphere 测试环境

2) 定位包 com.ibm.servlet -- 按+ 号
3) 选择类 SERunner
4) 单击奔跑的人按钮.

6. 在 Netscape, 输入 URL http://localhost/JKToys/index.html. localhost 是你的机器的名字.你可以用 127.0.0.1 或者你可以在一个命令提示输入 hostname 来找到你的机器名.
7. 按"Kids Zone". 一个页面显示，上面有一家飞机飞过中间。
8. 还输入 URL http://127.0.0.1/servlet/SimpleServlet 这会显示一个简单的消息在屏幕上。

第 4 部分 其它考虑

在教室里的设置被复制到个人的机器上后，这里有一些项目需要在个人的机器上更新。
1.**计算机名**。. 从网络属性窗口确信网络里的每一台机器都有唯一的计算机名。这只是在机器在一个网络里是才需要。
2.**唯一的 IP 地址**. 如果复制的机器在一个网络里，每一台机器需要一个唯一的 IP 地址和使用 DHCP 来获得一个动态 IP。
3.**HTTP Server**。计算机名（看步骤 1），应该被输入到 IBM HTTP Server 设置文件 (c:\Program Files\IBM HTTP Server\conf\httpd.conf), 参见第 1 部分中的步骤 8，设置 IBM HTTP Server.。
4.**桌面图标。**应该让它们的属性被选中，来确信它们指向本地磁盘驱动器上的文件 (不是网络驱动器).
5.**DB2** 的 DB2SYSTEM 注册条目必须同计算机的 TCPIP 名字一样。使用"NT 注册编辑器"来改变这个条目的现存的值为第一步设置的计算机名，作以下的步骤：选择开始->运行，输入"regedit"，单击确认，扩展: HKEY_LOCAL_MACHINE SOFTWARE IBM DB2 选择 GLOBAL_PROFILE。确信 DB2SYSTEM 条目的值被设置为现在的计算机名。如果不对，双击 DB2SYSTEM，在"Value data" 域输入计算机名，按确认. 关闭注册编辑器
6.**NT 安全 ID**。如果复制的机器在一个网络里， NT 要求每一个机器有一个唯一的安全 id。大多数复制软件包如 Ghost 和 Drive Image 有"sid walkers"，会处理这个任务。根据使用的包，"sid walker" 会作为复制步骤的一部分自动执行，或者会在复制后专门执行一下。看你的复制软件的文档来获得细节。

附录 II IBM WebSphere Studio 实验指导

你将用 WebShpere 建立 JKToys（Just for Kids Toys）网站的一些部分，JKToys 是一个假想的玩具公司。在将要做的所有实验里，你会需要事先准备好的许多文件。实验指导员应告诉你这些文件所在的目录，我们不妨用 X：来代表这个目录。

Lab1A(Fusion 1)：创建 JKToys 网站

在这个练习里你将用 NetObjects Fusion
1. 创建一个新的 Web 站点，包括：
   — 一个新的站点样式
   — 一个新的主控边框
   — 新的 HTML 页面
   — 页面之间的链接
2. 发表这个网站

这个网站称为 JKToys（Just for Kids Toys)。在开始以前创建一个目录，例如，***C:|JKToys***，来放置你的练习。

第一部分：创建一个新的 NetObjects Fusion 站点叫做 JKToys

1. 启动 NetObjects Fusion。选择 Start->Programs->NetObjects->NetObjects Fusion3.0.1。你会看到“ 欢迎使用 NetObjects Fusion ”对话框。
2. 如果你还没有创建 JKToys 站点，选择“ 空白站点 ”，按“ 确定 ”按钮。在“ 新建站点 ”对话框中输入文件名 *JKToys*，按“ 保存 ”按钮。
3. 如果你已经创建了一个站点，在“ 欢迎使用 NetObjects Fusion ”对话框上按“ 取消 ”按钮。从菜单上选择“ 文件 ”->“ 打开站点 ”Site。在“ 打开 ”对话框选择 *JKToys*，按“ 打开 ”按钮。在下一个“ 打开 ”对话框，选择 JKToys.nod，按“ 确定 ”按钮。

第二部分：为 JKToys 站点创建一个新的站点风格

1．选择“ 样式 ”图标，然后选择菜单“ 样式 ”->“ 新建样式 ”。
2．输入新建样式名称 JKToys。选择“ 确定 ”。
3．从左边的面板上选择你刚建立的 JKToys 站点样式。
4．为这个风格创建按钮：
a.双击右边面板上的第一级导航条.
i. 按“ 正常 ”标签右边的“ 浏览 ”按扭。浏览到包含按钮图的 NetObjects 目录：x:\JKToys Workshop\labs\Ex1-Fusion\Images，双击 StandardButton.gif。
ii. 按“ 突出显示 ”标签右边的“ 浏览 ”按扭。浏览到包含按钮图的 NetObjects 目录：x:\JKToys Workshop\Labs\Ex-1-Fusion\Images，双击 StandardButtonOn.gif
iii. 按“ 确定 ”
b. 双击右边面板上的第二级导航条.
i. 按“ 正常 ”标签右边的“ 浏览 ”按扭。浏览到包含按钮图的 NetObjects 目录：x:\JKToys Workshop\labs\Ex1-Fusion\Images,双击 AltStandardButton.gif。
ii 按“ 突出显示 ”标签右边的“ 浏览 ”按扭。浏览到包含按钮图的 NetObjects 目录：x:\JKToys Workshop\Labs\Ex-1-Fusion\Images，双击 AltStandardButtonOn.gif
iii.按“ 确定 ”
5.按“ 设置样式 ”图标。

第三部分：为 Welcome page 创建 MasterBoard

1. 在 Page 视图里打开 Welcome page（在 Site 视图里双击那个页面，或在 Site 视图里选择那个页面，在工具栏上按 Page 按钮）。
2. 把 MasterBorder 的名字改为 jkmainmb.
a)单击 MasterBorder 区域（在 Layout 正方形外面的任意地方）。
b)在属性对话框中，确信选中 General 页(如果属性对话框看不到，选择：视图->面板->属性面板).
c)按 Name field 边上的 Add/Edit 按钮
d)按 Edit MasterBorder List 对话框，选择 Add 按钮。
e)在 New MasterBorder 对话框，在 Name 输入 jkmainmb，按“ 确定 ”。
f) 在 MasterBorder 区域(在 Layout 正方形外面的任意地方)。从上面 MasterBorder 边白删除 banner(the top marque with the X in it)，从左边的 MasterBorder 边白删除左边的 navigation bar(选择每个元素，按 Delete 键或 Edit->Clear)。
3. 把 BasterBorder 的尺寸改为以下(用 Step 2 里同样的 Properties 对话框的同样的面板)：Left:10;Top:170;Bottom:133,Right 保留缺省值。

4. 参见图 1，在 Welcome 页面的 MasterBorder 区域增加一些图(gif 文件)：StaticMasterhead.gif(上边界)、toysonly.gif(下边界)、nclogo.gif(下边界)。你需要的所有图都存放在 x:\labs\Ex1-Fusion\images 目录(x 是包含 JKToys workshop CD 图片的驱动器或目录)中。可从资源管理器里进行拖放来增加一个图，或者：
   a) 如果页面没有被打开，在 Page 视图里打开页面(在 Site 视图里双击那个页面)
   b) 在屏幕左边的 Tools 面板里选择 Picture 图标。如果面板看不到，选择 View->Toolbars->Standard Tools。在 toolbar 选择 picture 按钮。
   c) 在 MasterBorder 区域你想放图的地方单击。
   d) 从 File Selection 对话框里选择图。
   e) 把图拖到正确的地方。
5. 在页面的底部排放图。一个方法是使用 Multi-Object 面板里的 Position 页。另一个方法是选择多个元素(SHIFT－鼠标左键用来选择多个元素)，用菜单条的 Object->Distribute Object->Horizontally 或 Object->Align objects->Vertical Center 来排列。
6. 保存站点：File->“保存” Site。你也可以预览这个站点，如果你要的话你可以单击 Preview 按钮。注意，Control->Preview(在单击 Preview 按钮时按住 Control 键)预览单独一个页面。

第四部分：为 JKToys 站点创建初始 HTML 页面

1．如果你不在 Site 视图，单击 Site 图标。把 Home Page to Welcome 的名字改为 JKToys。要做到这一点，选择这个页，在 Properties 对话框，确信 Page 页被选中(如果 Properties 对话框看不到，选择 View->面板 s->Properties 面板)。在 Name 字段类型输入 Welcome to JKToys(你也可以在页面图标上直接改变名字)。选择 Custom Names 按钮，编辑 Navigation Button 字段为 Welcome.单击“确定”。
2．创建 5 个空白页，作为 Home Page 的孩子；这样作：选择新的 Home Page,然后：
   a) 从弹出菜单选择 New Page，或
   b) 从菜单条选择 Edit->New Page
3．就象在 step 1 里所作的，把 5 个页面的 name 和 Custom Name 改成以下：

Name	Custom Name->Navigation Button
JKToys Shopping	Shopping
JKToys Self-service	Self-service
JKToys Kids Zone	Kids Zone
JKToys Information	Informaiton
JKToys Search	Search

4. 创建以下的页面。这些页面里的某些没有内容，但是需要完成按钮和链接。为每一个页面选择一个父页面，然后从菜单条或弹出菜单里选择 New Page。

Parent Page	New Page Name	**Custom Names->Navigation Button**
JKToys Self-service	JKToys Registation	Registration
JKToys Self-service	JKToys Literature	Literature
JKToys Self-service	JKToys Parts Locator	Parts Locator

JKToys Self-service	JKToys Orders	Orders
JKToys Self-service	JKToys Accounts	Accounts
JKToys Self-service	JKToys Returns	Returns
JKToys Kids Zone	JKToys Kids Zone Registration	Kids Registration
JKToys Information	JKToys Employment	Employment
JKToys Information	JKToys Store Locator	Store Locator

第五部分：完成 Welcome page

1. 在 Welcome page 的 Page 视图中，在 Layout 区域中增加 4 个图：welcome.gif、links1.gif、soldier.gif、legos2.gif。用这个方法把图加到页面里：
   a) 在 Page 视图里打开页面(在 Site 视图双击页面).
   b) 在 Windows 资源管理器里选择包含图的文件。这些文件在 X:\JKToys Workshop\Ex1-Fusion\images.
   c) 把文件拖到 NetObjects 的 Layout 区域.
   d) 放置文件，位置如图 1 所示。
2. 增加文字(从 Tools 面板里选择 Text 按钮，创建一个正方形，设置大小，在里面放置你的文字)。输入以下文字：Use our Site Map to move around the site。
3. 增加 Site Map：选择 View->Toolbars->Component Tools，从 Component Tools 面板里选择 Site Mapper。画一个正方形，在里面放置 Site Map。不要担心正方形的尺寸。关闭 Component Tools 面板.
4. 从菜单条选择 Object->Size Layout to Objects，使 Layout 区域的大小适合你放进去的对象。
5. 创建三个链接，分别从 Welcome 页面到 Shopping,Information,Store Locator:
   a) 在 Tools 面板单击 Hotspot:Rectangle。
   b) 围绕 SHOP ONLINE 画一个正方形，在 Link 对话框选择 JKToys Shopping。按 Link 按钮完成链接。
   c) 重复步骤 a)，然后围绕 ABOUT OUR COMPANY 画一个正方形，从 Link 对话框的页面列表里选择 JKToys Information。
   d) 重复步骤 a),然后围绕 STORE LOCATOR 画一个正方形,从 Link 对话框的页面列表里选择 JKToys Store Locator。

   注意：你可以在 NetObjects Fusion 里用 Follow Link 功能跟踪链接。为了做到这个，或者右击一个链接元素，然后从弹出菜单选择 Follow Link，或者选择元素，从菜单条里选择 Go->Follow Link。为了跟踪一个文字链接，先击文字，这样文字光标出现。
6. 预览你站点：击 Preview 按钮。为了看到 site map，你要发表站点。

第六部分：发表你的站点

1．设置 NetObjects Fusion Publish Setting
   a) 从 Site 页面单击 Publish 按钮。从菜单条选择 Publish->Publish Setup.
   b) 在 Directory Structure 页，确保目录结构是 By Asset Type。这会生成 asset 和 html 子目录，是 JKToysMasthead applet(将在 BeanBuilder 里创建)正确工作所需要的。
   c) 按“确定”，结束 setup 的这个部分。
2．发表你的站点：从 Site 页面单击 Publish 按钮。选择 Publish->Publish Site。从下拉列表里选择 My Computer。如果你没有预先设置 MyComputer 位置，你需要执行以下的步骤：
   a) 从下拉列表里选择 My computer。按 location 旁边的 Edit 按钮。
   b) 在 Location Properties 对话框里，确保 Server Name 是 My Computer,选择 Local 单选按钮按钮，规定目录为 C:\www\internet\，C:是 web server 安装的驱动器的。
   c) 按“确定”完成编辑。

3．按“确定”发表你站点。
4．确保 internet site 的 web server 实例这样启动的:
   a) 运行 X:\JKToysWorkshop 目录里的 go-internet.bat。
5．在 Netscape 的 Location 区域输入以下内容，打开主 internet 页面。
   a) http://127.0.0.1/
   b) 测试链接，确保你的所有页面正确地发表了。

Lab1B(Fusion2)：继续创建 JKToys 网站

在这个练习里，你将为你的 JKToys 网站创建新的页面，包括：
— 图片轮放
— 图像映射
— 发布站点
并发表这个网站。这里的指导没有上一个练习那么详细。

第一部分：创建别的 MasterBorders

1.创建一个新的 MasterBorder，叫做 jkhowtomb。
  a).打开 JKToys Kids Zone 页面。在 Page 视图，在 JKToys Kids Zone 的 MasterBorder 里单击。
  b).在 MasterBorder Propertises 对话框的 General 页，按 Name 边上等 Add/Edit 按钮。
  c).在 Edit MasterBorder 列表对话框，击 Add 按钮。
  d).在 New MasterBorder 对话框，规定名字 jkhowtomb，把它基于 jkmainmb(从 Based On 输入区旁边的下拉列表选择)。选择“确定”，然后 Close。这将创建一个新的 MasterBorder 叫做 jkhowtomab,看上去跟你为 Welcome 页面创建的一样。
  e).用 Property 对话框重新设置 MasterBorder 的尺寸：Top:180,Left:125,Right 25,Bottom 133.
2.在 jkhowtomb MasterBorder 的左边创建一个框架。
  a).在 MasterBorder 的任何地方单击。
  b).打开 Properties 面板。
  c).在 Properties 对话框，单击 AutoFrame 下面的 Left 图标。
  d).不要选 Generate HTML frame borders 勾选框。
  e).单击框架区域，然后在 Properties 对话框的 Frame 页单击。
  f).选择 Solid Color 单选按钮。
  g).为背景色选择紫色。如果你被要求使用近似的颜色，也可以。
  h).从标准工具栏里选择 Navigation Bar，把它垂直地放在框架里(画一个垂直长方形，把它放在里面)，Navigation Bar 的顶部同 Layout 区域的顶部相平。
3.创建一个新的 MasterBorder 叫做 jkinfomb.
   a).打开 JkToys Information 页面。在 Page 视图,在 JKToys Information 页面的的 MasterBorder 区域。
   b).在 MasterBorder 对话框的 General 页单击 Name 旁边的 Add/Edit 按钮。
   c).在 Edit MasterBorder 列表对话框里，单击 Add 按钮。
   d).在 New MasterBorder 对话框里，规定名字为 jkinfomb,基于 jkmainmb(从 Base On 输入区域旁边的下拉列表里选择)。选择“确定”，然后 Close。这将创建一个新的 MasterBorder 叫做 jkinfomb，看上去跟你为 Welcome page 创建的一样。
   e).用 Properties 对话框重新设置大小，Top:180,Left 125,Right 25,Bottom 133.
4. 在 jkinfomb MasterBorder 的左边创建一个框架。
  a).在 MasterBorder 的任何地方单击。

b).打开 Properties 面板。
c).在 Properties 对话框，单击 AutoFrame 下面的 Left 图标。
d).不要选 Generate HTML frame borders 勾选框。
e).单击框架区域，然后在 Properties 对话框的 Frame 页单击。
f).选择 Solid Color 单选按钮。
g).为背景色选择绿色。如果你被要求使用近似的颜色，也可以。
h).从 Standard Tools 面板里选择 Navigation Bar，把它垂直地放在框架里(画一个垂直长方形，把它放在里面)，Navigation Bar 的顶部同 Layout 区域的顶部相平。

5.改变 jkinfomb Navigation Bar 来显示 Child Level 链接。
a).选择 Navigation Bar。
b).在 Properties 对话框，确信 Vertical Navigation Bar Properties 对话框出现。
c).单击 Display 区域右边的 Options。
d).单击 Nav Bar Display 对话框的 Child Level 单选按钮。
e).选择 Include Home Page checkbox。单击“确定”。
6.在 JKToys Search 页面使用 jkhowtomb MasterBorder。
a).在 Page 视图打开 JKToys Search page.
b).选择 MasterBorder 区域。
c).打开 Properties 对话框。
d).从 Name 右边的下拉列表里选择 jkhowtomb MasterBorder。

第二部分：完成 JKToys Shopping page

1.在 JKToys Shopping 页面增加一个表。
a).在 Tools 面板里选择 Table 按钮
b).在 layout 区域里你想放表的地方画一个长方形。
c).在 Table 对话框输入 2 列 3 行，别的项目为缺省值
d).选择 Tools 面板里的 Text 按钮，增加文字
e).重新设置表的尺寸，把文字包含进去。

Register Here	Join our club!
Guest Shopper	You’re JustLooking
Registered Shopper	JKToys Members Only

2.创建从表到正确的页面的链接，对每一格：
a).双击格子里的文字，高亮它。
b).如果没有打开 Properties 对话框的话，打开它。
a)单击链接
b) 从页面列表里选择链接。链接是这样的：

Register Here	JKToys Registration
Guest Shopper	JKToys Store Locator
Registered Shopper	JKToys Orders

3.向 JKToys Shopping page 增加一个文字元素。
a).在 Tools 面板选择 Text 工具

b).在页面上你想放文字的地方画一个边界框
c).输入图 2 里的文字

注意：当你加文字时，按 Enter 创建一个新的段落，按 SHIFT Enter 创建一个新的行。

4.向 JKToys Shopping page 增加一个图 welcome.gif
a).从 Tools 面板里选择 Picture 工具
b).在上面的表里单击页面
c).选择 X:\JKToys Workshoop\labs\Ex1-Fusion\images\welcoss.gif.

第三部分：完成 JKToys Self-service page

1.向 JKToys Self-service 页面增加一个 Navigation Bar
a).在 Page 视图里打开 JKToys Self-service 页。
b).从 Standard Tools 面板里选择 Navigation Bar，把它垂直地放在紧靠在 layout 区域左边的框架里(画一个垂直的长方形，把它放在里面)，Navigation Bar 的顶部，同 Layout 区域的顶部相平。
c).单击 layout 区域的左边。
d).打开 Properties 面板
e).单击 Display 域旁边的 Options 按钮
f).选择 Child Level
g).单击“ 确定 ”
2.向图 3 里的页面增加 4 个文字元素。注意，你必需选择文字来改变它的塑性，而不只是选择文字元素。同样地，要输入文字的下一行，按住 shift 按 Enter.

第四部分：完成 JKToys Kids Zone page

1.在 Page 视图打开 JKToys Kids Zone 页面
2.增加图 planerun.gif 和 twartists.gif，就象图 4 这样。注意，planerun.gif 是一个 multi-part(animated)GIF,第一帧是空的，所以在页面上什么也看不到。
3.创建一个有小孩图案的 rotating picture of the refrigerator.
a).在 View->Toolbars 面板,选择 Component tools，然后在 Component Tools 面板单击 Rotating Picture.
b).在你想放置组件的地方画一个长方形。
c).一个占位图像出现在页面上并且 Rotating Picture Properties 面板 页出现
d).在 Component 页里设置 Rotating Picture Component 参数。在 External Link 页输入 External Link URLs:
Pause Time:2
Number of Images:3
Image 1:frkids.gif
External link URL for Image 1:www.yahooligan.com
Image 2:frsb.gif
External link URL for Image 2:www.disney.com
Image 3:frsf.gif
External link URL for Image 3:www.warnerbros.com

第五部分：完成 JKToys Information page

1.增加图 5 里的 2 个文字元素到页面上。注意，文字的被链接的四行是一个文字元素。从四个链接的文字行创建链接如下：

Employment Opportunities	JK Toys Employment
Franchise Opportunities	JK Toys Literature
Investing in the JKToys Company	JK Toys Literature
Latest new about JKToys	JK Toys Literature

2.保存站点：File->“保存” Site

第六部分：发表你的站点

1．设置 NetObjects Fusion Publish Settings.
   a)从 Site 页单击 Publish 按钮。从菜单条选择 Publish->Publish Setup.
   b) 在 Directory Structure 页，确信目录结构是 By Asset Type。这会产生 asset 和 HTML 子目录，是 JKToysMasthead applet(在 BeanBuilder 里建成)正确工作所需要的。
   c)单击“确定”结束设置的这个部分。
2．发表你的站点：从 Site 页单击 Publish 按钮。从菜单条里选择 Publish->Publish Site。从下拉菜单里选择 MyComputer。如果你没有预先设置 MyComputer 地点，你需要作以下的步骤。
   a)从下拉列表里选择 My Computer。单击 location 旁边的 Edit 按钮。
   b) 在 Location Properties 对话框，确信 Server Name 是 My Computer，选择 Local 单选按钮，规定目录为 C:\www\internet\，C 是安装 web server 的目录。
   c)单击“确定”结束编辑。
3．单击“确定”发表你的站点。
4．再一次，确信”internet”web server 在运行(go-internet.bat)
5．在 Netscape 的 Location 区域输入地址打开 main internet page
   a).http://127.0.0.1/
   b).测试不同的链接,确信你的所有的页面已经正确地发表。

第七部分（可选）：增加 Store Locator Page

1.创建一个新的页面，作为 Store Locator 页面的 child。命名为 JKToys Store Locator.
2.在 Page 视图里打开新的 JKToys Store Locations 页面。
3.用 Windows Explorer，把 X:\JKToys Workshop\labs\Ex1-Fusion\Store.txt(X 是 CD 的目录)拖进 layout 区域. 格式化文字，使得 state 用大些的字体，颜色同 city 不同，city 同 address 不同。在每一个 state name 的前面和旁边创建以下的 anchors.要创建一个 anchor,单击你要 anchor 的文字区域，在 Properties 对话框单击 Anchor 按钮。

State	Anchor
New York	NY
Pennsylvania	PA
Ohio	OH
North Carolina	NC
Florida	FL
Texas	TX
California	CA

4.在原始的 JkToys Store Locator 页面，不是在你刚才创建的 Store Location 页面上。
a)把 X:\JKToys Workshop \labs\Ex1-Fusion\images\usa2.gif 图加进页面。
b) 在 Store Locator 图的每一个红色区域，创建一个多边形 Hot Spot：
c)在 Tools 面板上按住 HotSpot:Rectangle 按钮，然后从按钮右边显示出来的选择列表里选择 Hotspot:Polygon。
d) 单击多边形的点，绘出区域轮廓.
e)双击，结束多边形。
f)从每一个多边形创建一个链接到 JKToys Store Location 页面，如上表所示的正确的 state anchor，选择正确的 anchor。
5.入第六步所说的，发表站点

Lab2：使用 NetObjects BeanBuilder 创建 Applet

在这个练习里,你将
1. 用 NetObjects BeanBuilder 创建一个 Applet， 用：
   - 图
   - Rollover Pictures
   - URL links
   - Java methods
2. 发表这个 Applet 来用在 NetObjects Fusion
3. 增加 applet 到你在练习 1 里创建的站点。

第一部分：开始创建 applet,把图载入 Gallery,放在 applet canvas 上放一个图

1.从开始菜单选择 Programs->IBM WebSphere->NetObjects BeanBuilder 1.0->BeanBuilder 启动 NetObjects BeanBuilder。
2.按“取消”按钮关闭 New Applet Wizard。我们将在没有 Wizard 的情况下工作。如果你不想在每次启动 BeanBuilder 时看到这个窗口，你必须复位选择框：show this window at start time.
3.把需要的图载入 Gallery：
a)如果 Gallery 窗口看不到，选择菜单条项目 Window->Gallery
b) 在 Gallery 里单击 Find 图标(有一个箭头的文件夹)。
c)从 X:\JKToys Workshop\labs\Ex2-BeanBuilder\images 里选择和双击任何一个图，或图上单击，然后单击

“打开”。Gallery 自动装载在那个目录里的所有图(X:是 JKToys workshop 目录所在的目录，可能是你的 CD－ROM，也可能是你的硬盘上的目录)

4. 在 Gallery 单击 jkhead.jpg，单击 applet canvas(在 Composer 窗口的左上方的白色正方形)，把第一个图加进 applet canvas.
5. 单击图 jkhead.jpg，在 Details 窗口的 Properties 页面把 name 设置为 jkhead。

第二部分：把 4 Rollover Buttons 和 5 URl 链接放到 applet canvas 上

注意：作为以下指导的另一个替代，你可以在 applet canvas 上放 1 个 rollover button，设置图，拷贝这个按钮三次，在每一个拷贝里手工编辑图的名字。你可以对 URL 链接采取同样的步骤。

1．在面板窗口选择 Multimedia 页。
2．单击 Rollover 图标，在 Jkhead 图下单击。注意 applet canvas 怎样重新设置尺寸接收一个新的元素。
3．当对话框出现,用 Find 按钮为 rollover button 定位 3 个图：
   a)Button Picture X:\JKToys Workshop\labs\Ex2-BeanBuilder\images\nbash.gif
   b) Pressed Picture ：X:\JKToys Workshop\labs\Ex2-BeanBuilder\images\nbashp.gif
   c)Rollover Picture：X:\JKToys Workshop\labs\Ex2-BeanBuilder\images\nbashO.gif
4．为别的 rollover buttons 重复步骤 2 和 3 三次，使用文件：
   a)nbass.gif,nbassP.gif,nbassO.gif
   b) nbakz.gif,nbakzP.gif,nbakzO.gif
   c)nbajkt.gif,nbajktP.gif,nbajktO.gif
5．选择所有的 rollover buttons（用 Shift 鼠标左键单击），然后从菜单条选择 Layout->Align->Top.
6．安排 rollover buttons，和 jkhead 图，如图 1 所示。确信 rollover buttons 没有重叠。重新设置 applet canvas 的大小，使得 rollover buttons 和图适合白色区域。
7．在面板窗口选择 NetWorking 页，单击 URL Link.
8．把 URL Link 放在 composer 的任何地方(你可以放在 applet 的里面或外面，运行时它们是看不到的)。
9．重复上面的步骤四次，把另外 4 个 URL 链接放进 composer。
10．在你在练习 1 的开头创建的目录里保存你的 applet(选择“文件”>“另存为”，保存为 JKToysMastHead.app).运行这个 applet 看看你到现在为止都做了些什么。练习的这个阶段的答案可以在 x:\JKToys Workshop\Labs\Solutions\Ex2-BeanBuilder\JKToysMastHead1.app 找到。

第三部分：设置所有的属性，建立 URL 链接的连接

在这些 URL 里你不需要 http:，因为你将创建一个 Java 方法来实际创建正确的 URL。

1.单击 URL Link1，在 Details 窗口设置 name 为 shoppingURL,URL address 为/html/jktoys_shopping.html,Status Message 为 Go to the Shopping Page,target frame 为 parent frame.在这个练习里不需要 target name。
2.单击 URL Link2，在 Details 窗口设置 name 为 SelfServiceURL，URL 地址为/html/jktoys_self-service.html，Status Message 为 Go to the Self-Service Page，target frame 为 parent frame.
3.单击 URL Link3，在 Details 窗口设置 name 为 KidsZoneURL，URL 地址为/html/jktoys_kids_zone.html，Status Message 为 Go to the Kids zone Page，target frame 为 parent frame.
4.单击 URL Link4，在 Details 窗口设置 name 为 infoURL,URL 地址为/html/jktoys_information.html，Status Message 为 Go to the About JK Toys Page，target frame 为 parent frame.
5.单击 URL Link5，在 Details 窗口设置 name 为 homeURL，URL 地址为/index.html，Status Message 为 Go to

the Go to the JK Toys Home Page，target frame 为 parent frame.

6.单击 jkhead 图，在 Details 窗口选择 Connection 页。记住总要选择这个项目，即使它显示为缺省。

a)选择 When->mouse clicked,Bean->homeURL,Do->show document.

b) 选择 When->mouse entered,Bean->homeURL,Do->show status

7.单击 Rollover1

a)选择 When->mouse entered,Bean->shoppingURL,Do->show status.

b) 选择 When->clicked,Bean->shoppingURL,Do->show document

8.单击 Rollover2

a) 选择 When->mouse entered,Bean->selfServiceURL,Do->show status.

b)选择 When->clicked,Bean->selfServiceURL,Do->show document

9.单击 Rollover3

a)选择 When->mouse entered,Bean->kidsZoneURL,Do->show status.

b)选择 When->clicked,Bean->kidsZoneURL,Do->show document

10. 单击 Rollover4

a)选择 When->mouse entered,Bean->infoURL,Do->show status.

b)选择 When->clicked,Bean->infoURL,Do->show document

11. 保存你的工作，再次运行 applet，看看是否 Show Status 功能正常。你不能检查链接是否正常，除非你在 Netscape 里运行 applet。这个练习的这个阶段的答案可以在

x:\JKToys Workshop\Labs\Solutions\Ex2-BeanBuilder\JKToysMastHead2.app

找到

第四部分：增加两个 Java 方法，创建正确的 URL

因为一个用户可以从不同的页面访问 masthead applet，applet 必须基于当前页面建立正确的 URL。

1．从菜单条选择 Window->Java

2．规定一个 import 语句

a)从 Java 窗口的右上角的 Other Methods Choice 选择 imports

b) 在现存的代码行的最后加上 import java.net.*;

c)单击“ 保存 ” Method.

3．创建第一个方法，单击 New Method(最左边)图标。

4．把 Java 窗口的内容换成 x:\JKToys Workshop\labs\Ex2-BeanBuilder\method1.txt

a)用 notepad 打开 method.txt

b) 选择文件里的所有文字

c)拷贝文字到 Clipboard

d) 用 shift-insert 把文字粘贴到 Java 窗口。

5．单击“ 保存 ” Method 图标

6．创建第二个方法，单击 New Method(最左边)图标。

7．把 Java 窗口的内容换成 x:\JKToys Workshop\labs\Ex2-BeanBuilder\method2.txt

a)用 notepad 打开 method.txt

b)选择文件里的所有文字

c)拷贝文字到 Clipboard

用 shift-insert 把文字粘贴到 Java 窗口。

8．单击“ 保存 ” Method 图标.

9．把 applet 的 started event 联到 initialize 方法：

a)打开 Details 窗口

b) 单击 applet canvas 的空白区域(白色区域)

c)在 Details window,when 之下，单击在 started 上。

d) 在 Bean 之下，单击 Applet1

e)在 Do 下单击 initialize.

10．保存你的工作，运行 applet，看看你是否有 Java 错误。注意你不能测试链接是否工作直到你在 Netscape 里运行 applet。这个练习的这个阶段的答案可以在 x:\JKToys Workshop\labs\Solutions\Ex2-BeanBuilder\JKToysMastHead.app.

第五部分：发表 applet 作为一个 NFX 组件

1．选择 File->Publish 来显示 Publish Wizard。在 Wizard 作以下的：

a)按 Next 按钮

b) 在 Publish As 页面，规定你要 applet 发表为一个 web 的 applet.

c)按 Next 按钮

d) 在 Local 页面，规定你要发表本地文件夹，选择发表为 NetObjects Fusion component.选项

e)选择你为练习 1 创建的目录（也许是 c:\JKToys）.

f)在 Finish 上单击。

第六部分：在你在 NetObjects Fusion 里创建的站点里载入 Applet

1.启动 NetObjects Fusion，打开你在练习 1 里创建的站点

2.打开 JKToys Welcome 页面.

a)删除 banner(StaticMastHead.gif)

b) 在 Component tool bar 单击 NetObjects Fusion Components(NFX Components)图标

c)单击 MasterBorder 区域里 banner 删除以前在的地方

d) 单击 Add.选择你刚刚创建好的 NFX 组件(C:\JKToys\JKToysMastheadComp.nfx)

e)单击“ 打开 ”.

f)在 Installed Components list 里选择 JKToys\JKToysMasthead-BeanBuilder,单击“ 确定 ”。注意：移动组件，这样它就适应 MasterBorder.

3.在 page 视图,放好 masthead,使得它在上边界的左上角，选择 Object->Size Layout to Objects.如果需要，重新设置 top MasterBorder margin,直到 masthead 适合空间。

4.预览页面，用 Control-Proview.注意你不能在 proview mode 看到组件。

5.对 jkinfomb 和 jkhowtomb MasterBorder 重复步骤 a,b,c,和 2 的 f。这些 MasterBorders 分别用在 JKToys Information page 和 JKToys Kids Zone page。

第七部分：发表你的站点

1．设置 NetObjects Fusion Publish Settings.

a)从 Site 页面单击 Publish 按钮。从菜单条选择 Publish->Publish Setup.

b) 在 Directory Structure 业，确信目录结构是 By Asset Type。这将产生 asset 和 html 子目录，将是 JKToysMasthead applet 正确工作需要的。(在 BeanBuilder 里创建)

c)按“ 确定 ”结束设置的这个部分。

2．发表你的站点：从 Site 页面单击 Publish 按钮。从菜单条选择 Publish->Publish Site。从下拉列表里选择

My Conputer.如果你以前没有设置过 MyComputer location,你需要执行以下步骤：

a)从下拉列表里选择 My Computer。按 Edit 按钮 next to location.

b) 在 Location Properties dialog,确信 Server Name 是 My Computer,选择 Local 单选按钮，规定目录为 c:\www\internet\,c 是 web server 安装的驱动器的驱动器字母。（忽略说 applet 的.class 文件丢掉的消息——它被放在.jar 文件里〕

c)按“ 确定 ”结束编辑。

3．单击“ 确定 ”发表你的站点。

4．确信“ internet”site 在运行，(go-default.bat)

5．在 main intenet page 打开 NetScape,在 Netscape 的 Location 区域输入以下：

a)http://127.0.0.1/

b) 测试不同的链接，确信所有你的页面已经被正确地发表了。

Lab3：WebShpere Studio & BeanBuilder with WebRunner Beans

在这个练习里，你将用 WebRunner chart beans 和 NetObjects BeanBuilder 创建一个商务应用，可以用 IBM WebSphere Studio 生成的 servlet 从 DB2 里取出数据，用图表表示。在你开始以前，把 WebRunner bean 加进 NetObjects BeanBuilder：

a)拷贝文件 x:\JKToys Workshop\Lectures\L5-BeanBuilder\Beans*.jar 到 c:\WebSphere\BeanBuilder\Beans

b) 选择 Start->NetObjects BeanBuilder 1.0->BeanBuilder 启动 NetObjects BeanBuilder,检验 Chart Bean 已经被加进 Chartbb 页。

第一部分：创建 Applet JKChart

1.改变 applet 的布置到 Border Layout.

a)在 composer 窗口，选择 applet(左上角的白色 box)

b) 在 Properties 页的 Details，选择 layout 输入区域，单击 position 或...按钮。当对话框出现，选择 Border Layout，然后 Horizontal 和 Vertical gap 为 5。

c)单击“ 确定 ”

2.增加 webrunner chart bean 到 Applet

a)在 Palette 窗口，选择 Chartbb 页.

b) 接着，在 Palette 窗口单击 ChartBean，然后在 Composer 窗口单击 applet 的中部把 bean 放进 applet.

c)确信 Chart Bean 被放进 applet 的 Center 位置。

1)单击 Chart Bean，然后选择 Details 窗口的 Properties 页。

2)选择 size and position 输入区域，改到 Center(如果还没有设置为 Center)

d) 选择 File->“ 保存 ”保存 applet.

e)选择 File->Run 测试 applet.

3.改变 Chart Bean 的属性

a)选择 ChartBean，然后选择 chartTitle 域，改变值为”Sales”.

b) 选择 name 域，改变值为“ SalesChart”

4.创建一个方法来用 applet 参数设置 chart 数据

a)从 Composer 的菜单条里选择 Window->Java 菜单

b) 从 Java 窗口的工具条上单击 New Method 图标。

c)用 Explorer,打开文件 x:\JKToys Workshop\labs\Ex3-Studio-DB\initparam.txt,选择文件里的所有 Java 代码，Edit->Copy 它。

d) 把 Java 窗口里的内容更换为从文件里拷贝来的 java 代码。代码看上去应该是这样的：

```
public Object initParms(EventObject anEvent){
    int rows=
    Integer.valueOf(getParameter("NumberOfRows")).intValue();
    SalesChart.setChartTitle(getParameter("Title"));
    SalesChart.getChartLabels().removeAllElements();
    SalesChart.getChartData().removeAllElements();
    for (int i=1;i<=rows;i++){
        SalesChart.getLabels().addElement(getParameter("Label"+i));
        SalesChart.getChartData().addElement(
            Integer.valueOf(getParameter("Value"+i)));
    }
    return null;
}
```

e)用 Java 窗口的“ 保存 ”图标保存方法。

5.从 applet 的 started 事件调用 initParams 方法：

a)在 Composer 窗口选择 applet.

b) 从 Details 窗口的 connection 页，从 When 列选择 started。

c)从 Bean 列选择 Applet1 bean

d) 从 Do 列选择 initParms 方法

6.用 hard coded data 测试 bean

a)选择 Applet1,Details 窗口，然后是 Properties 页。如果 Details 窗口还没有打开，你可以选择 window->Details 打开

b) 在 Properties 页,选择 applet parameters，在值列按...按钮加名字－值对。

c)把以下的名字－值对加进去，按“ 确定 ”按钮

Label1	Q1
Label2	Q2
Label3	Q3
Label4	Q4
Value1	138
Value2	133
Value3	102
Value4	122
NumberOfRows	4
Title	Sales Targets

d) 从文件菜单选择“ 保存 ”选项(File->“ 保存 ”)保存 applet 的改变和 hard-coded parameters

e)选择 File->Run 测试 applet

Lab4：创建 JSP 页面来执行 JKToys 数据库上的雇员搜寻

在这个练习里你将用 Studio SQL 和 Data Access wizard 来允许内部网用户执行雇员搜寻。这个宏从雇员服务页面/intranet/jkemp 链接出来。读者应该熟悉基本 SQL，HTML 和浏览器来检查结果。按照这个练习，你将能用 IBM WebSphere Studio 来查询雇员数据。在你开始以前，请

1．如果你的机器没有联网，设置你的浏览器的代理设置为直接连接。

2．确信 WebSphere Studio CLASSPATH 正确设置来访问 DB2 驱动器和 Studio 里的类。设置运行时 CLASSPATH：

a)从 Windows NT Control 面板 l,(Start->Settings->Control 面板 l),双击 System 图标打开它

b) 从显示的窗口里选择 Enviroment 页

c)在 User Variable 部分，选择 CLASSPATH variable

d) 确信 c:\sqllib\java\db2jaba.zip;在 Value 域，c:是\sqllib 所在的驱动器。如果没有，在末尾加上它

e)按 SET 按钮应用你的改变

f) 关闭窗口

3．把 db2jaba.zip jar 文件放在 WebSphere Application Servers' ClassPath.

a)从 Start Menu,选择 Programs->Ibm WebSphere->Application Server v1.0->Administration 启动 WebSphere Application Server Administration 页面。

b) 用 admin 用户 ID 和 admin 口令登录

c)按 Manage 按钮到 administer servlets

d) 当弹出窗口显示，选择 Setup 按钮和 Basic 选项。

e)选择 Basic 页,在 java ClassPath 文字域，增加下面的内容到行的最后：

c:\sqllib\java\db2java.zip;

c 是\sqllib 安装的驱动器

4．在这个练习里，当在 Netscape 里打开页面，确信你显式地用浏览器的 Shift-Reload 重新载入页面.这将保证你得到最新版本的页面。

第一部分：用 SQL 向导创建 SQL 查询

SQL Query 是一个有.sql 扩展名的文件，将被 Studio Wizard 用创建一个 Database Servlet。所有 SQL 规定都放在 SQL Wizard 创建的.sql 文件里。

1．如果还没有启动 WebSphere 的话，启动 WebSphere。

2．创建一个新的项目叫做 JKToysEmployees

a)从菜单条选择 File->New project...。在 General 页，规定项目的名字为 JKToysEmployees.单击 OK。看图 1

3．用 SQL Wizard 创建一个对 Sample Database 里的 Project Table 的查询

a)从 Tools 菜单选择 SQL Wizard 启动 SQL Wizard

b) 通过向导的 tabbed windows,规定下表的信息：

Tabbed Window	Action Explanation
Welcome	1.输入 JKToysEmp 作为 SQL 查询的名字 向导会增加.sql 扩展名
Logon	2.输入向导需要的连接数据库的信息。例如： Database name:jdbc:db2:SSDEMO01 Userid:USERID Password:PASSWORD Driver:IBM DB2 UDB Local 3.按 Connect,向导把你连接到规定的数据库 注意：Userid 和 Password 的大小写很重要——都应该是大写
Table	4.在 Statement type 单击 Select 5.选择 USERID.JKPDF 表 这个练习会对 JKPDF 表使用 SELECT 语句
Join	没有动作
Columns	6.从表的下拉列表里选择 USERID.JKPDF,选择下列列： a. LNAME b.FNAME c.ADDR d.CITY e.PHONE f.TITLE g.DEPT h.EMPNO 7.按 Add>>按钮，把这些列移动到 Columns to include 当 SQL 查询被使用，数据会从被选中的列里被 SELECT 语句取出来。在选择完列后，你能选择列的名字，然后用 Move Up 和 Move Down 按钮来改变它们出现的顺序
Condition 1	8.从表的列表里选择表 JKPDF 9.从列域里选择 LNAME 列 10.从 opetator 域里选择”start with the character(s)” 11.在值的第一个域里单击，按 Parameter 按钮 12.在 Create New Parameter Dialog 输入”Lastname”,按“确定” 13.按 Find on another column 按钮另一个情况——Condition 2 “Lastname”变成生成的 SQL 语句的一部分，最后会被从网页上来的一个值填入。这将使数据的结果集合按照输入的 lastname 减少
Condition2	14.从表的列表里选择 JKPDF 表 15.选择”Find more rows(OR)”单选按钮 16.从列域里选择 DEPT 列

	17.从 operator 域里面选择”is exactly equal to” 18.在值的第一个入口域里单击，按 Parameter 按钮 19. 在 Create New Parameter Dialog 里输入”Department”，按“ 确定 ” “Department”在变成生成的 SQL 语句的一部分，最后会被一个从网页里来的值填入。这会使得数据的结果集合根据输入的 department number 减少
Sort	20.从 Columns box 选择 LNAME 和 FNAME 21.按 Add>>按钮 被选中的列用来对取得的数据排序。在你把类 Columns to sort on 以后，你可以用 Move Up 和 Move Down 按钮改变它们的顺序。
SQL	22.按 Next，不要作任何动作 这个 tab 展示了完成的 SQL 语句。你可以拷贝到剪贴板，在另一个应用里使用。
Finish	23.按 Finish 这个完成的 SQL 文件会作为 JKToysEmp.sql 在项目里出现

从 WebSphere Studio File 菜单里选择 File->“ 保存 ” Project，保存项目

第二部分：建立 Database Servlets

1.用 Studio Wizard 创建一个 database Servlet
a)从 WebSphere Studio 主窗口选择 JKToysEmployees 项目
b) 从 Tools 菜单里选择 Studio Wizard 图标来 Studio Wizard
c)从 Welcome 页选择 Database servlet，按 Next
d) 通过 wizard 的 tabbed window，规定下表的信息：

Tabbed Window	Action Expalnation
Welcome	1.在 Name 域输入 JKToysHR 这个值被用来作为 Database Servlet 文件的名字。Wizard 将增加正确的文件扩展名 2.保持 Java 包的名字不变 3.输入一个说明“ Show Employee Data” 4.让 Output Folder 的值保持不变
SQL 语句	5.按 Select from an active project 6.从列表里选择 JKToysEmp.sql 文件 Wizard 使用规定的文件里的 SQL 语句来从数据库请求数据
Input Page	7.按 Yes please 单选按钮 8.选择 Lastname 和 Departname 作为你想能从网页规定输入值的域 一个 input form 将为这些域创建
Input Layout	9.输入一个 Page Title 为”Employee Search”

	为了调用数据库查询，需要一个提交按钮。重命名 Reset 按钮为 Clear.Clear 按钮将清除表格的域
Output Page	10.按 Next,不要作任何事情 接受缺省设置将使得 SQL 语句里得来的所有列出现在生成的 HTML 页面里。
Output Layout	11.输入一个 Page Heading 为”Employee Information Query Results” 12.对别的选项接受缺省值
Custon Error Page	13.单击 Next，不规定任何值。
Finish	14.单击 Finish. Database Wizard 将生成和编译文件

从 WebSphere Studio File 菜单选择 File->“ 保存 ” Project 来选择项目。

第三部分：发表生成的 servlet

1.在发表以前，生成的 servlet 需要改变为指向你将发表你的 HTML 页面的位置。

a)在 WebSphere Studio, 双击 JKToysEmployees\classes\JKToysEmployees\JKToysHRServlet.Servlet，在 NetObjects ScriptBuilder 里打开它。

b) 找到文字<default-page>,改变跟着的行为

<uri>/employee/JKToysHROutputPage.jsp</uri>

2.发表 JKToysEmployee 项目

a) 选择 JKToysEmployee 项目，从菜单条的 File 菜单里单击 Publish 选项(File->Publish).

b) publishing server 为 local,按 Server 边上的 Edit 按钮

c) 在出现的对话框里，确信 Local 单选按钮被选中，html 的 Path 为 c:\www\intranet\HTML\Employee，类的 Path 为 c:\WebSphere\AppServer\classes. Employee 目录也许不在你的机器上。你可以在剩下的路径里键入 Employee 目录名字。按“ 确定 ”按钮。

d) 在 Select Publishing Server 对话框，按 Continue 按钮，使得 wizard 生成文件到磁盘。如果你被要求创建 Employee 目录，选择 Yes 来创建它。

e) 在 Publish to Local 对话框，按 Publish 按钮。

第四部分：测试生成的 JSP

1．测试 JSP

a)运行 c:\JKToysWorkshop 目录里的 go-intat.bat 来启动 Lotus Domino Go，运行内部网站点。

b) 在 Netscape 的 Location 区域输入以下的东西，在内部网主页上打开 Netscape:

http://127.0.0.1:8080/

c)选择 Employee Service 图标，去雇员服务页面。

d) 接着，选择 Employee Lookup,注意你将看不到生成的页面，因为到你刚才发表的新的 HTML 页面的链接还没有建立。

2．链接到调用你刚才创建的 JSP 的新的 HTML 页面

a)选择 Start->Programs->NetObjects ScriptBuilder->NetObjects ScriptBuilder 3.0，打开 NetObjects ScriptBuilder

b) 选择 File->“ 打开 ”来打开页面 c:\www\intranet\html\intranet\jkemp.html

c)找到文字 Employee Lookup，改变为

<p>Employee Lookup is now available.

d) 选择 File->“ 保存 ”保存文件，File->Close 关闭。关闭 NetObjects ScriptBuilder.

3．关闭和重启 Webserver 和 Netscape。向上面一样测试。你应该看到 Employee Lookup input 页面。HTML 页面上的输入参数是大小写敏感的。姓都以大写开头。试一下字母 D。你会得到所有姓以字母 D 开头的条目。

附录III IBM WebSphere 应用开发实验指导

Servlet 1A:创建一个简单的客户注册 servlet

在这个练习里，你将创建一个简单的 servlet。这个 servlet 的目的是确认一个新客户在 JKToys 站点的成功注册。

在这个练习里你将访问 JKToys 网站。你从主页选择 Kids Zone 区域，注册为一个新客户。注册表格和网站的余下部分已经由网站设计者为你提供。在注册表格上按 Register me now!按钮，将会调用服务器上的 JKRegistration servlet。注册 servlet 将会用表格的信息注册一个用户，在 JKTOYS 数据库的 CUSTOMER 表创建一个新的条目。在注册时，指派一个唯一的客户号码。在下一个练习里你将用这个信息提供一个定制的 HTML 页面确认注册。

第一部分

对所有的练习，确信你用了 id:USERID 和口令：PASSWORD 登录进系统。

这个练习的前 5 个步骤只是给你的信息，请花时间读它们。实际的代码从第二部分开始。

1．你将用 IBM VisualAge for Java,version 2,Enterprise Edition+Enterprise Update(从现在开始用 VAJ 指代)来编辑，编译和发表你的练习。现在双击桌面上的图标打开 VAJ 产品。如果你看到了 VAJ 的欢迎窗口，选择“ 取消 ”按钮，这将放上 Workbench 窗口。你将看到带有几个包的 WASDev 项目已经为你预先装载了。选择 com.ibm.waslab.servlet1 包。在这个包里你将能看到 RegistratonServlet 类。这个类包含 RegistrationServlet 和 startup methods required for you to complete this exercise.在这个练习里，你将集中注意力于 doPost()方法。

2．JKToys 网站已经布置到安装在你的机器上的 web server 上。这个站点的根目录是:[x]:\www\HTML\JKToys；[X]是 web server 所在的驱动器字母。网站是用 NetObjects Fusion 创建的。这个站点的 HTML 页面在 html 子目录里。JKToys 注册页面的完整的路径是：[x]:\WWW\HTML\JKToys\html\body_jktoys_kids_zone_registraiton.html

3．你还记得在讲座里提到，当用户按了一个表格上的提交按钮，在我们这里按钮上写着“ Register me now!”，浏览器调用 HTML <FORM>标记的 ACTION 子句里规定的 URL。看一下 HTML 文件你将看到<FORM>标记包含以下文字：

<FORM NAME=”RegistrationForm”

ACTION=http://127.0.0.1/servlet/JKRegister METHOD=POST>

向网站设计者提供<FORM>标记的正确的参数是你的责任。

在 ACTION 子句里的 URL 指向主机 127.0.0.1，这是一个指向本地主机的 IP 地址，换句话说，指向你自己的机器。网站设计者也许用“ localhost”这个字代替 IP 地址。

注意：在某些情况下用"localhost"会降低你的服务器的性能，为了避免这个问题，你必须在你的机器里的一个叫 HOSTS 的文件里加一行，把 localhost 和 IP 地址对应起来。

在 URL 里跟在 IP 地址后面的 servlet 应用键，使得 web server 调用 WebSphere Applicaiton Server(把 HTTP 请求传给它)。应用服务器在 servlet classpath 里配置的某个子目录或 jar 文件里定位 servlet。IBM WebSphere Application Server 的缺省 servlet classpath 是：[x]:\WebSphere\AppServer\servlets 目录，[x]是 WebSphere 安装的目录。

JKRegister 是 servlet 名字。这不是 servlet 的真实名字，而是在 WebServer 里给 servlet 的名字。在实践中，给 servlets 名字是很常见的，它降低显示在浏览器上的 URL 的复杂性和混乱。它还允许你向 WebSphere 里的你的 servlet 提供初始参数。按我们的配置，JKRegister 被对应为：
com.ibm.waslab.servlet1.RegistrationServlet，这是这个 servlet 的包和类的名字。

METHOD＝POST 子句表示在 servlet 被调用时，servlet 的 doPost()方法将被 service()方法调用。

4．正如上面提到的，你的 servlet 将在 doPost()方法里取得控制。选择 RegistrationServlet。定位 doPost()方法。注意给 doPost 的两个参数，HttpServletRequest 和 HttpServletResponse。这些对象让你找出你从哪里来，到哪里去，输入 HTML 表格的值是什么。在下面的步骤里你将学会怎样用它们。

5．doPost 的开头几行提取输入到表格中的值，在数据库里注册新客户。现在你不需要关心哪个方法实际在数据库里注册了用户，在以后的练习里，你将自己写这个方法，你将学几个访问数据库的方法。你将在这个练习的后半部分写 getFormImput()方法。

```
//Get the input form contents
Properties formInput=getFormInput(req);

//Register new customer in the database
register(formInput);
```

第二部分

在这个 servlet 里，当 doPost()得到控制，第一个任务是取得用户输入和注册新客户。这是前面讨论的两行代码所作的。最后一步是通知客户注册成功了。

你在第二部分的任务是写 doPost()方法的余下部分，通知用户注册成功。现在打开 doPost()方法，看看有些什么：

```
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException,jiava.io.IOException
{
    //Get the input form contents
    Properties formInput=getFormInput(req);

    //Register new customer in the database
    register(formInput);

    //Obtain a PrintWriter from response object

    //Set content type on the response header

    //Output confirmation message
    return;
```

}

注释说明了你的代码将要执行的任务。第一个任务是从 response 取得一个 PrintWriter.看看你的课堂笔记，找个例子。如果你还是不懂，看最后的答案部分。得到一个 PrintWriter 后，你将需要设置 response header 的内容类型为”text/html”。你再看以下课堂笔记，找个例子吧。

最后，你将需要输出确认消息。页面的标题应该是”Registration Complete”,包含以下文字：

Greetings:

Your registration has been recorded in the database.

Thank you for your interest in JKToys.

We look forward to serving you in the future!

你的课堂笔记给出了在一个 Servlet 里打印文字的例子。如果你不懂，看最后的答案部分。

按 Ctrl-s，或从右键选项选择” “ 保存 ” ”，保存 doPost().记住，在 VAJ 保存你的方法的时候也编译了你的方法。如果你在编译时得到错误，VAJ 会警告你，有时会提供正确的方法名或正确的语法来建议你怎样改正错误。改正你的错误，保存它。如果你不能解决错误，请向你的辅导员请求帮助。

一旦编译完成而没有错误，你可以试试你的 servlet。为了调试的目的，你将在 VAJ 里运行你的 servlet。你的 workspace 已经用 IBM WebSphere Application Server libraries 配置过了。用这个配置你可以在 VAJ 里使用 HTML 和 servlets。这将允许你在 VAJ 环境里调试你的 servlets.

测试你的第一个 servelt 的步骤

1. 在 VAJ 里启动 web server。打开 IBM WebSphere Test Enviroment 项目，在包 com.ibm.servlet 里，选择 SERunner 类，按 Run 按钮来启动服务器。你可以看控制台来确定 web server 是否在运行。它将打印许多行输出。
2. 然后,启动你的浏览器,在 URL 输入域输入http://127.0.0.1/JKToys/index.html,按 enter。这将显示 JKToys 的主页。从 banner 选择 Kids Zone。在 kids zone,单击拖着标语的飞机，这将把你带到注册页面。填入表格的信息，按注册按钮，几秒钟之后你应该得到 servlet 的回应，表示一个成功的注册。

如果你真的要确信你的记录已经在数据库里,你可以在 JKTOYS 数据库的 CUSTOMER 表上执行一个查询，来看看你自己。你可以在 DB2 命令行里，或 DB2 文件夹里的 Control Center 工具作这个。

在这个练习里你作了些什么

在这个练习里你创建了一个将被 JKToys 的新客户注册表格调用的 servlet。你在 doPost()方法里增加代码，来提供给客户的反馈，告诉他们注册成功。你现在理解了表格，servlet 和回应 HTML 页面之间的控制流。

你在你的 servlet 里动态创建一个 HTML 页面。在下一个练习里你将使用客户提供的信息进一步定制 servlet 给客户的回应。你还将检查错误，如表格信息遗漏或数据库问题。

答案

在你能向客户送回信息以前，你需要创建一个输出流，你将用这个同浏览器通信。HttpServletResponse 对象有一个方法来获取这样一个流。把下面的代码加到 doPost()方法里 register()调用的下面：

```
//Obtain output stream form response object
PrintWriter out=res.getWriter();
```

也可以创建别的类型的流，要看什么会回到浏览器。PrintWriter 尤其适用于送字符串，因为它有打印新行和在一个现存行后面打印的方法，分别是 println()和 print()。

现在你已经有一个输出流，你可以开始把你的消息送回浏览器。记住，你实际作的是编辑一个 HTML 页面，它将被在用户的浏览器上显示。任何有效的 HTML 页面都可以送回。

在你送实在的文字以前，你应该设置回应的 MIME 类型。你用 setContentType()方法在回应头里设置内容类型。输入以下行：

```
//Set content type on the response header
res.setContentType("text/html");
```

你可以输入这些行完成你的 HTML 页面。

```
//Output confirmaiton message
out.println("<html>");
out.println("<head><title>Registration Complete</title></head>");
out.println("<body>");
out.println("<h2><br><br>Greetings:</h2>");
out.println("<br>Your registration has been recorded in the database");
out.println("<br>Thank you for your interest in JKToys");
out.println("<br>");
out.println("<h2>We look forward to serving you in the future!</h2>");
out.println("</body></html>");
```

最后，你可以刷新和关闭流。在一个 servlet 的末尾这么作是可以的，只要你知道这个 servlet 将不会被另一个 servlet 调用来提供输出到流。你会在以后的练习里看到，一个 servlet 可以调用另外的 servlets，每一个 servlet 的输出只是要完成一个 HTML 页面的所有的信息的一部分。如果任何一个 servlet 关闭流，主调 servlets 将不能使用这个流来完成页面的变换。所以当你关闭输出流时要小心。输入以下的行来完成 servlet 的编码。

```
outflush();

//Closing the output stream is not really necessary
out.close();
```

注意：在设置和在这些练习的指导里，你将会被要求编辑.properties 文件。确信你没有使用 Notepad。用 Wordpad 或别的编辑器，保存你的操作系统的原始的句子结束符。

所有的练习从 JKToysDBInfo.properties 文件获得访问数据库所需要的参数。在这个文件里，你会看到 USERID 和 PASSWORD 等等的定义。如果你需要改变这个文件来适应你的配置，可在以下目录找到：

d:\IBMVJava\ide\project_resources\WASDev

d:\WebSphere\AppServer\servlets

第一个在运行在 VAJ 里是用，第二个在运行在一个网络服务器时用。

Servlet 1B:改进客户注册 servelt

在这个练习里你将在输入表格上执行错误检查，发现用户没有填写的字段。对这个例子来说，所有字段都是强制的。
你将从上一个 servlet 你丢下的地方开始这个练习。你将从草稿重写 getFormInput()方法。这个方法从表格提取参数，返回一个 Properties 对象，用字段名字作键值。这个方法还检查空白或空参数。你将使用属性对象中的值来定制送回去的确认消息。

第一部分：创建 getFormInput()方法

1．在 VAJ 里打开 com.ibm.waslab.Servlet1 包里的 RegistrationServlet。改变现存的 getFormInput()方法的 signature，增加以下的”throws”列表。

```
Private Properties getFormInput (HttpSErvletRequest req) throws
    InvalidParameterException
    {
    }
```

注意：这个方法抛出一个 InvalidParameterException，这个异常的类已经提供给你了，在 VAJ 的 com.ibm.waslab.exception 包里。

getFormInput()方法声明为 private，返回一个 Properties 对象，它将包含表格的参数名字和值。它把 HttpServletRequest 对象作为一个参数，因为它包含原始形式的输入表格的值。

2．然后，增加代码检查空白字段。
注意：在这些指导里，因为印刷的行的长度限制，有很多行代码在纸上被分开了。你应该在 VAJ 里把这些行作为一个单独的行输入。

```
Properties formInput=new Properties();
Enumeration fieldNames=req.gerParameterName();
while (filedNames.hasMoreElemenets())
{
        String paramName=(String)fieldNames.nextElement();
        String paramValue=(String)
              req.getParamterValues(paramName)[0];

        //check for missing parameters(code to be added)
        if (paramValue.length()= =0)
              throw(new InvalidParameterException(“Required parameter missing”));
        formInput.put(paramName,paramValue);
}
return formInput;
```

确信你理解所有已经有的代码。第一行声明一个新的 Properties 对象，命名为 formInput,来保存参数名字/值对。然后这个 request 对象的 getParameterNames()方法是用来取得一个所有参数名字的枚举。这些是表格里字段的名字，将被用作 formInput properties 对象的键值。

反复通过枚举，你取得参数名字和值。你检查，来确保每一个值有一个比 0 大的长度。如果长度等于 0，一个异常被抛出和处理。如果值的长度比 0 大，它被保存在 formInput，用字段名字作为键值。当枚举里没有更多的元素时，循环结束，返回填好的 forminput Properties 对象。

保存 getFormInput()方法，重新打开 doPost()方法。

第二部分：处理 InvalidParameterException 异常

因为 getFormInput()现在可以抛出一个异常，doPost()里的代码得处理这个情况，把这个方法的调用放在一个 try/catch 块里。修改调用 getFormInput()的代码段来处理这个异常。如果异常被抛出，调用方法 missingParameterResponse(),这将生成一个错误 HTML 页面。在异常的情况下，你还将需要从 doPost()方法返回，结束事务，不写任何东西到数据库。

重新提交一个表格来测试代码，这次有一个参数（输入表格上的一个字段）没有填入。错误页面应该被显示，而不是你在前一个练习里编码的成功注册消息。

第三部分：产生一个定制的输出 HTML 页面

1．在这个部分你将定制用户在记录被存入数据库后看到的信息。你会在页面里显示客户的名字和新指派给客户的客户号码。在 register()调用后面输入以下的行：

```
//Output confirmation message
out.println("<html>");
out.println("<head><title>Registration Complete</title></head>");
out.println("<body>");
//modified or additional lines of code are in bold
out.print("<h2><br>Greetings:");
out.println("formInput.getProperty("FNAME")+"</h2>");
out.println("<br>Your registration has been recorded in the database");
out.println("<br><br>You have been assigned customer number"<strong>"+formInput.gerProperty("CUSTNO")+"</strong>);
out.println("<br>Thank you for your interest in JKToys");
out.println("<br><h2>We lo " 确定 "  forward to serving you in the future!</h2>");
out.println("<a href=\""+ServletUtil.getClientURL(req)+"/jktoys/htm/jktoys_kids_zone.html\">"+"Back to the kids Zone!!!</a><br>");
out.println("</body></html>");
out.flush();
out.close();
return;
```

在这个练习里我们假设写数据库成功。在一个真实的应用里，你应该捕捉在 register()方法里的任何异常，显示合适的错误信息。

注意字段的值怎样从属性里提取，插入被建立的 HTML 页面的正确的地方。

从底向上第四行建立一个回到注册页面的超链接。ServletUtil.gerClientURL(req)的调用返回 URL 的第一部分，包含一些东西，象：http://127.0.0.1。http 和主机名从请求对象里得到，组合进一个字符串，用来形成这个这个页面的完整的 URL。web server 的端口号，如果不是 80，也要带上。ServletUtil 是一个在 com.ibm.waslab.common 包里向你提供的类。

2．Property 对象里的客户号条目不是原始表格的一部分，而是插入 register（）方法。这个唯一的号码是从数据库得到的：查询一个现有的最大的客户号，再加 1。
3．保存 doPost()方法。如果你在编译期间得到错误，返回到源代码，改正它们，再编译。如果你不能解决错误，请向你的辅导员请求帮助。
4．现在你已经准备好测试你的改进的 servelt 了。遵从前一个练习里的同样的测试步骤，通过正常的执行路径，输入所有的字段，按 register，现在应该得到新的，改进的和定制的祝贺。
5．再试一次，让一个或两个字段空白。按 register 按钮，你应该得到自己的错误 HTML 页面。

第四部分：输出和在 VisualAge 外面运行 servlet

1．要从“生产”环境运行 Registration Servlet，先停止 WebSphere Text Enviroment(SERunner)。
2．从 VisualAge 输出类和源代码到 WebSphere Application Server。选择 WASDev 项目，鼠标右击，选择 Export。选择”directory”，击”Next”。按“Browse”按钮，浏览到 c:\WebSphere\AppServer\servlets 目录。选择.class 和源代码。按 Finish。这把类文件布置到 AppServer。
3．启动 HTTP Server(Start->Programs->IBM HTTP Server->Start Server)。
4．从浏览器访问 Registration 页面，象以前一样重新运行 Registration 场景。

你在这个练习里做了些什么

在这个练习里，你学会怎样提取一个表格里的字段的值，和怎样把它们包含在属性里。你使用 Properties 对象里的信息来生成一个定制的动态的 HTML 页面。你也处理表格里提供的信息不全的情况，显示一个错误页。

答案

```
Properties formInput=null;
//Get the input form contens
try
{
    formInput=getFormInput(req);
}
catch (InvalidParameterException e)
{
    missingParameterResponse(out);
    return;
}
```

Servlet2:JKToys Login 和 Toy Display Servlets

在这个练习里你将学会在一个 web 会话里用户在许多 HTML 页面和 servlets 里移动时维护信息，完成一个

事务。

按照这个练习，你将能用 HttpSession 类存储在会话里持续的信息。你还将学习怎样阻止在浏览器里缓存页面，来防止缓存的页面显示前一个会话或查询的错误信息。在这个练习里，你将建立三个 servlets：

- Login
- DisplayToys
- TotalPurchase

这个三个 servlets 的组合将满足用户的需求。对每一个方法，大多数答案代码可以在一个叫 cutAndPaste 的方法里找到。这将使你在敲键盘敲烦的时候能”cut-and-paste”。

第一部分：修改 Login servlet 来保存会话数据

1．在 VAJ 选择 com.ibm.waslab.Servlet2 包。从这个包里选择 LoginSevlet。在 doPost()里定位包含下面文字的代码行：

```
//Enter code here to retrieve the customer number form the form data
```

2．在那个下面输入以下的代码。这个代码调用 getCustomerNumber()方法来提取输入到 HTML 表格里的客户号。如果 Login 按钮被按下时客户号码为空，这个方法抛出一个 InvalidParameterException;你用 outputLoginFailedMessage()显示一个错误消息来处理这个异常。

```
String customerNumber=null;
try
{
        customerNumber=getCustomerNumber(req);
}
catch (InvalidParameterException e)
{
        //Output failure message
        outputLoginFailedMessage(out,”You did not enter your customer number.”);
        return;
}
```

3．保存方法。你将得到错误，因为 getCustomerNumber()和 outputLoginFailedMessage()方法还没有被编码。带着错误保存方法，继续下一步。

4．创建 getCustomerNumber()方法。这个方法比前一个练习里的同名方法简单得多，因为只有一个字段要从表格里取出来。以下行表示这个方法：

```
private String getCustomerNumber(HttpServletRequest req) throws
InvalidParameterException{

        String paramValue=(String)
        req.getParameterValues(“CUSTNO”)[0];
        //check for missing parameters
        if(paramValue.length()==0)
                throws(new InvalidParameterException(“Required parameter missing”));

        return paramValue;
}
```

保存这个方法，返回到 doPost()。

5．在 doPost()方法的下一行，调用 locate()方法来取出对应输入的客户号码的记录：

```
//Look for this customer in the database
```

JKToysCustomer customer=locate(customerNumber);

如果没有在数据库里发现客户，返回一个 null 客户对象，否则的话它包含先前为这个客户号码保存在数据库里的信息。

6．如果客户号码没有在数据库里找到，你需要提供一个合适的消息。在 locate()调用下面输入这些行：

```
//customer will be null if customer not found in the database
if(customer= =null)
{
        //Output failure message
        outputLoginFailedMessage(out,”The customer number you entered was not found in our database.”);
        return;
}
```

7．继续下面的代码，处理客户号码在数据库里找到的情况：

```
else
{
        //create a session and store the customer information
        HttpSession session=req.getSession(true);

        //if there is an existing session,invalidate it and re-create
        if (session.isNew()= = false)
        {
              session.invalidate();
              session=req.getSession(true);
        }
        session.putValue(“CUSTOMER”,customer);
        //Output confirmation message

        //Ensure that no caching takes place in the browser
        res.setHeader(“Pragma”,”no-cache”);
        res.setHeader(“Cache-Control”,”no-cache”);
        res.setDateHeader(“Expires”,0);

        String clientURL=ServletUtil.getClientURL(req);

        out.println(“<html>”);
        out.println(“<head><title>Login complete</title></head>”);
        out.println(“<body>”);
        out.print(“<p><h2>Greetings:”);
        out.println(customer.getFirstName()+”                    ”+customer.getLastName()+”,from
“+customer.getCity()+”</h2></p>”);
        out.println(“<p>Welcome back to JK Toys</p>”);
        out.println(“<p>what can we do for you? Please select from the options below:</p>”);
        out.println(“<a          href=\””+clientURL+”/jktoys/html/body_jktoys_kids_zone_change.jsp\”>Update
registration information</a><br>”);
        out.println(“a href=\””+clientURL+”/servlet/JKOrderStatus\”>Status of my last order</a><br>”);
        out.println(“a     href=\””+clientURL+”/servlet/JKDisplayToys\”>Display     toys     for     my     age
group(“+customer.getAge()+”)</a><br>”);
```

```
        out.println("<p>Thank you for your interest in JKToys</p>");
        out.println("</body></html>");
    }
```

else 语句后面的最初几行创建了一个会话。然后你检查是否是一个新的会话。如果这个会话不是新的，就把它失效掉，再创建一个新的会话。现下你有一个有效的会话了。用一个键值 CUSTOMER 保存会话里的 customer Properties。这个数据将被同一个会话里的 servlets 使用。

接下去的三行是必需的，确保浏览器不会缓存 servlet 产生的页面。如果页面被缓存，它将可能在页面上显示从一个以前的缓存页面得来的错误的信息。

在这一页有一些定制。我们用客户的全名和城市来祝贺客户。我们也提到他在哪一个年龄组，在：Display Toys for my age group choice。同前一个练习一样，这个客户的数据是用 getProperty()方法从 customer Properties 对象里取出来的。如果你希望，你可以修改 HTML 标记来修改消息的内容或格式为你更喜欢的样子。

注意，Display Toys choice 的 href 标记会调用 JKDisplayToys servlet。你将在这个练习的下一个步骤里继续做这个 servlet。

保存 doPost()方法。现在我们已经准备好写 outputLoginFailedMessage()方法来完成这个 servlet。这个有用的方法在 doPost()里已经被调用了两次，因为需要输出同样的 HTML，这两个只有一个句子不一样。我们可以在 doPost()方法里编写 HTML 代码，但是，这是更好的方法。在"New Method"SmartGuide 里创建 outputLoginFailed(PrintWriter out,Stirng message)方法，输入以下的代码：

```
/**
*This is method prints the generic failure page with a specified message
*@param out java.io.PrintWriter
*@param message java.lang.String
*/
private void outputLoginFailedMessage(PrintWriter out,String message)
{
    out.println("<html>");
    out.println("<head><title>Login Failed</title></head>");
    out.println("<body>");
    out.println("<p><h2>Sorry,could not log you in</h2>");
    out.println("<p><font color=\"red\">+message+"</font></p>");
    out.println("<p>Please press the <strong>Back</strong>button,and resubmit.</p>");
    out.println("<p>Thank you for your interest in JKToys!!!");
    out.println("</body></html>");
    out.flush();
}
```

一旦你输入这个代码，最后的错误应该从 doPost()里消失。
确信服务器在 VAJ 里运行，测试 Login servlet。
进入 JKToys Kids Zone 页面里工作。在有飞行的飞机的页面上，击右边的椭圆图案。它将把你带到 Login

表格。输入一个你知道的已经存在的客户号码，按 Login 按钮。你将得到 servlet 的祝贺。现在试一下无效的客户号码，和一个空白客户号码。确信你在每种情况下都得到了正确的消息。

第二部分：创建 DisplayToysServlet servlet

它将搜查数据库，按照客户的年龄显示玩具。
在 VAJ 的 Servlet2 包里，你将找到建了一半的 servlet。打开 DisplayToysServlet 类。

1．因为这个 servlet 不是从一个表格调用的，而是从网页上的一个 href　HTML 标记调用的，它的 doGet()方法将从 servlet 的 service()方法调用。在 doGet()方法里开始工作。在设置回应头不要缓存的代码后面输入这些行：

```
//Get customer information from session
HttpSession session=req.getSession(false);

//We should have a previous session,otherwise it's an error
if(session= = null)
{
        //Output failure message
        showError(out);
        return;
}
```

2．注意，这次你把 false 作为参数传递给获取 session 的调用。这么作会只返回一个已经存在的 session，而不会创建一个 session。如果一个存在的 session 被找到，下一步是获取包含在 session 里的 JKToysCustomer 对象，它包含了客户信息。这个信息是这个练习的第一部分里的 LoginServlet 存进去的。如果我们有一个 session,但是用键 CUSTOEMR 找不到值，我们就有一个无效的 session,你会显示一个错误，然后退出。

```
 //Create a new Customer and initialize withthe value in the session
 JKToysCustomer customer=(JKToyscustomer)session.getValue("CUSTOMER");

 //We should have a customer stored in the session,if not ,it's an error
 if(custmer= =null)
 {
        //Output failure message
        showError(out);
        return;
 }
```

3．下一行 findToys()方法。客户的年龄作为参数被传进去。在这个方法里，数据库被访问，返回一个玩具的向量(vector)。如果没有为这个特定的客户找到玩具，一个 null vector 被返回。

```
//Look for toys in the database for customer's age group
Vector toys=findToys(customer.getAge());
```

4．在检查了玩具的向量不为空，displayToys()方法被调用，来建立一个表，显示向量里的玩具。类 toy 在 com.ib.waslab.common 包里提供给你。displayToys()方法接受请求对象、玩具向量、客户对象和 PrintWriter 作为参数。在显示了玩具以后，向量被保存在会话里，以备客户要定购。

```
if(toys!=null)
{
```

```
        displayToys(req,toys,customer,out);
        session.putValue("TOYS",toys);
    }
```

5．输入以下代码完成 displayToys()方法。研究这个代码，看看 HTML 表是怎样创建的，向量是怎样循环取得所有元素的。

```
    out.println("<head><title>Your JK Toys,toys selection</title></head>");
    out.println("<body>");
    out.println("<p><h2>"+customer.getFirstName()+",here are the toys we have in stock for your age group("+customer.getAge()+")</h2></p>");
    //show selected toys in a table
    out.println("<BR>");

    //Create form to display and select toys
    out.println("<FORM NAME=\"JKPurchaseForm\" ACTION=\""+ServletUtil.getClientURL(req)+"/servlet/JKTotalPurchase\" METHOD=POST>");

    out.println("<TABLE>");
    out.println("<TABLE BORDER=1");
    out.println("<tr bgcolor=#993366>");
    out.println("<TH><font color=#FFFFFF>Picture</font>");
    out.println("<TH><font color=#FFFFFF>Name</font>");
    out.println("<TH><font color=#FFFFFF>Description</font>");
    out.println("<TH><font color=#FFFFFF>Price</font>");
    out.println("<TH><font color=#FFFFFF>Buy it?</font>");
    Enumeration selectedToys=toys.elements();
    Toys aToy=null;
    int checkboxNumber=0;
    while (selectedToys.hasMoreElements())
    {
        aToy=(Toy)selectedToys.nextElement();
        out.println("<TR>");
        out.println("<TD><IMG SRC=\"..."+aToy.getThumbnailFile()+"\">");
        out.println("<TD>"+aToy.getShortDescription());
        out.println("<TD>"+aToy.getFullDescription());
        out.println("<TD>"+aToy.getPriceAsString());
        String boxId=String.valueOf(checkboxNumber++);
        out.println("<TD><INPUT id="+boxId+"\"TYPE=\"checkbox\" NAME=\""+boxId+"\">");
    }
    out.println("</TABLE>");
    out.println("<br><br>");
    out.println("<INPUT TYPE=\"submit\" NAME=\"PurchaseButton\" VALUE=\"Purchase selected toys\" id=\"PurchaseButton\">");
    out.println("</FORM>");
    out.println("<p><h2>Thank you for your inteest in JKToys!!!</h2>");
    out.println("</body></html>");
    out.flush();
```

```
out.close();
return;
}
```

注意，显示玩具的 table 是在一个 form 里建立的。table 的最后一列包含一个 checkbox，用来让客户选择哪些玩具他希望购买。checkbox 的名字是从 0 开始的顺序的数字。它们将被用作索引，从包含所有玩具的向量里提取选择的玩具。这将在 TotalPurchaseServlet 里作。

每一个玩具的价格存在玩具对象里，这个值作为 float 保存。floats 的格式不能被正确地打印，所以 toy 类有一个方法 getPriceAsString()，它执行格式化。
保存这个方法。

6．最后，创建一个 showError()方法。它将在会话信息不管因何种原因无效时被调用。输入以下行实现这个方法。

```
private void showError(PrintWriter out){

//Outpur failure message
out.println("<head><title>JK Toys error page</title></head>");
out.println("<body>");
out.println("<p><h2>Sorry,we don't have a record of your customer name make a valid selection</h2>");
out.println("<p>Please press the <strong>Back</strong>button,and resubmit.</p>");
out.println("<p>Thank you for your interest in JKToys!!!");
out.println("</body></html>");
out.flush();

return;
}
```

保存这个方法。

这个完成了 DisplayToysServlet。你应该登录进 JKToys Kids Zone，在确认你成功登录进站点的页面上选择 Display toysfor my age group 链接，以测试这个 servlet。

第三部分：完成 servlet，显示在 DisplayToysServlet 里选中的玩具

这个 servlet,TotalPurchaseServlet，也总计选中的玩具的价格，显示总数。客户有个机会确认购买，完成定单，或取消定单。

1．从 com.ibmwaslab.servlet2 包里选择 TotalPurchaseServlet。打开 doPost()方法。最初的几行已经为你输入好了：

```
//Obtain output stream from response object
PrintWriter out=res.getWriter();
//Set content type on the response header
res.setContentType("text/html");

//Get the session and retrieve the customer information
```

```
HttpSession session=req.getSession(false);

Vector toys=(Vector)session.getValue("TOYS");
```

这些行执行一些你在前面建立的 servlets 里作的典型的任务。PrintWriter 从 resposne 对象里取得，你设置 MIME 内容，从会话里取得玩具向量。这个向量包含所有这个客户的年龄组的玩具。

2．在下面的步骤里，你将向 doPost()方法里增加代码来完成功能。在 doPost()输入：

```
//declare an object to hold the toy order
JKToysOrder toysToBuy=null;

//Get the input form contents
toysToBuy = getFromInput(req,toys);
```

实现你声明一个 null JKToyOrder 对象。这个对象将包含客户选择的玩具的集合。JKToyOrder 类还包含定单的价格总和。

下一行调用了 getFormInput()方法，你将在下一个步骤里建立它。保存 doPost()方法，你将得到错误，因为你还没有些 getFormInput()方法。继续，带着错误保存。

3．创建 getFormInput()方法。这个方法循环通过前面表格里的 checkboxs，用 checkbox 的名字作为玩具向量的索引。被选中的玩具被加进定单。

注意：只有被选中的(checked)checkbox 被从表格发送回 servlet。submit 按钮也被传送。按钮的值被忽略，看下面的代码。

输入下面的代码来完成 getFormInput()方法。

```
private JKToyOrder getFormInput(HttpServletRequest req,Vector allToys){

        JKToyOrder selectedToys =new JKToyOrder();

        Enumeration fieldName=req.getParameterName();

        int i=0;
        while (fieldNames.hasMoreElements())
        {
              String paramName=(String)fieldName.nextElement();
              String paramValue=(String)req.getParameterValue(paramName)[0];
              try
              {
                    int toyIndex=Integer.parseInt(paramName);
                    selectedToys.addToy(allToys.elementAt(toyIndex));
              }
              catch (NumberFormatException e)
              {
                    //Skipping the button widget,only interested in checkboxes
                    //with numeric names
              }
        }

  return slectedToys;
}
```

你也许要看看 JkToyOrder 类，看它是怎么工作的，它可以在 com.ibm.waslab.common 包里找到。保存方法，打开 doPost()。

4．在你输入的最后一行后面，继续建立 doPost()方法，输入以下的代码：

```
//toysToBuy order will be empty if nothing was selected
if(toysToBuy.isEmpty())
{
        //Output failure message
        out.println(“<html>”);
        out.println(“<head><title>No toys selected</title></head>”);
        out.println(“<body>”);
        out.print(“<p><h2>Nothing selected</h2>”);
        out.println(“<p><font color=\”red\”> You did not select any toys!</font></p>”);
        out.println(“<p>Please press the <strong>Back</strong> button,and resubmit.</p>”);
        out.println(“<p>Thank you for your interest in JKToys!!!”);
        out.println(“</body></html>”);
        out.flush();
        return;
}
else
(
        displaySelections(res,req,toysToBuy);
}
```

这段代码检查是否至少有一个玩具被选择，如果 toysToBuy 定单是空的，一个错误消息被显示。如果定单不空，displaySelections()方法被调用来显示一个包含被选中的玩具的表。

保存 doPost()方法，应该没有错误。

5．完成 displaySelections()方法。这个方法同 DisplayToysServlet 里的 displayToys 方法很相似，有一些差别。输入以下的代码来完成 TotalPurchase Servlet 类里的一个新的方法：

```
//Enumeration selectedToys=toys.elements();
Toy aToy=null;

while(!toys.isEmpty())
{
        aToy=(Toy) toys.getNextToy();
        out.println(“<TR>”);
        out.println(“<TD><IMG SRC=\”..”+aToy.getThumbnailFile()+”\”>”);
        out.println(“<TD>”+aToy.getShortDescription());
        out.println(“<TD>”+aToy.getFullDescriptin());
        out.println(“<TD>”+aToy.getPriceAsString());

}

out.println(“<TR>”);
out.println(“<TD><INPUT    TYPE=\”submit\=”
        NAME=\”ConfirmationPurchaseButton\”
```

```
        VALUE=\"Confirmation Purchase\"
        id=\"ConfirmPurchaseButton\">");
    out.println("<TD><strong>Your total purchase amount is:</strong>");
    out.pritnln("<TD><strong>"+toys.getTotalPriceAsString()+"</strong>");
    out.println("</TABLE>");


    }
```

6．这个方法负责显示选中的玩具，和显示定单的总价格。这里有两个按钮在表格上，一个处理购买，另一个取消定单；在这个课程里这两个按钮调用的 servlets 没有被实现，学生在业余时间实现它们。

在这个练习里你处理客户登录，基于客户的年龄显示一个选中的玩具的表，让客户从列表里选择玩具，显示最后的选中的玩具的总和和这次定购的总额。你学习了怎样用会话保持 servlets 和 HTML 页面之间的状态。你还学会了几种从一个表格里显示和选择项目的方法。

JSP3:JSP 页面

在这个练习里你将创建一个 JavaServer Page。servlet 的目的是允许 Customer Profile 更新信息。
客户登录进 JKToys 的 Kids Zone 部分以后，客户可以作的一个选择是：更新他们的客户信息。为了处理这个，我们需要一个网页显示这个 JKToysCustomer 对象相联系的每一个信息（客户 ID 除外），作为编辑区域。这些字段需要用当前的客户记录的数据初始化。这些字段将是一个客户记录更新表格的一部分，将被提交给 ProfileUpdateServlet 来处理。

第一部分：JSP 文件

1． 你将从一个 HTML 文件开始，[x]:\www\html\JKToys\html\body_jktoys-kids_zone_change_start.html(x 是 Lotus Domino Go Webserver 安装的驱动器字母，文件是用 NetObjects Fusion 编辑的。第一步是在你的浏览器里打开这个文件（作为本地文件）。你看到的是一个 Table,字段对应 JKToysCustomer properties。每一个字段（加上一个隐藏字段）都需要在运行时用 live Customer data 生成。我们将用 JSP 语法作这个。

2． 用 NetObjects ScriptBuilder 打开 HTML 文件(Start->Programs->NetObjects ScriptBuilder->NetObjects ScriptBuilder 3.0)。ScriptBuilder 是一个简单的（基于文字的）编辑器，重要用来编辑基于 HTML 的文件(.html,.asp,.jsp,.shtml,....)在这个文件里你将注意到几个 HTML 注释表明需要使用 JSP 代码的地方。定位第一个这样的注释，看上去应该象：

   <!—The value field should be the Customer Number retrieved from the Customer object saved in the Session -->

   你将需要先写一个 scriptlet 来从 HttpSession 里取得 JKToysCustomer。下面的代码使用了 JSP 声明和 JSP Scriptlet 来完成这个：

   <%@ import=com.ibm.waslab.common.* %>

   <%KToysCustomer CUSTOMER=(JKToysCustomer)request.getSession(false).getValue("CUSTOMER");%>

   Declaration provides a global import for the Servlet which will be generated during page compilation.这样 com.ibm.waslab.common 包里的类在我们的 JSP scriptlet 和 expression 里就不要用全名了。把以上的行加进 HTML 文件。

3． 然后，我们将提供 JKToysCustomer 对象的 customerNumber 属性作为 HIDDEN 输入域的值。在同 INPUT 标签的 VALUE=element 相联系的引号里加入以下的 JSP 表达式:

   <%=CUSTOMER.getCustomerNumber() %>

4． 我们现在需要把每一个字段放进表，用从 CUSTOMER 对象里来的它们当前的值。当你往下扫过文件，你会发现几个 HTML 标记看上去象：
<INPUT id=”FormsEditFieldx” TYPE=”text” NAME=”xxx” VALUE=”” SIZE=....>
对这些中的每一个，你要在值域的引号之间增加一个 JSP 表达式。（看注释一确定你应该输入什么。有 6 个这样的字段。）

5． 我们要显示的最后一个输入域是一个下拉列表，让客户选择年龄组。This needs to be populated and have the “currently” specified age group,”selected”.这需要更多的 scriptlet 来有效地作这个。你所要的是产生 5 个<OPTION>标签标记(在<SELECT>块的里面)。每一个标签应该是“ Under 5”,”5-7”,”7-9”,”9-13”和”Over 13”。接着，这些”subtags ”中有一个应该看上去这样：
<OPTION SELECTED> age_group
这个规定了初始选中的项目。

第二部分：发表、运行和调试 JSP

1．把改过的文件保存在[x]:\www\html|JKToys\html\body_jktoys_kids_zone_change.jsp。注意，这个文件名和文件类型同提供的不一样。

2．接着输出你的 WASDev 项目(VA Java)到两个分开的目录树里(x;\WebSphere\AppServer\servlets 和 x:\IBMVJava\ide\project_resources\IBM WebSphere Test Enviroment\servlets).(第一个目录允许在 VA Java 外面运行，而第二个在 SERunner 的类路径里。)

3．下面通过网站，登录进 Kids Zones。(你应该从 VAJava 里面运行 SERunner。)在登录确认页面，你选择 Update registratio information。一旦你选择了这个，IBM WebSphere Application Server 将调用页面编译。一个 servlet 将从.jsp 文件产生，这个 servlet 将被编译。如果编译有错误(因为你的.jsp 文件里的不合法的 JSP 语法)，编译错误消息将在一个 HTML 错误页里返回(回到浏览器)。(注意，”Retrieve syntax error infromation”必须在 JSP Execution Monitor Options dialog 里被启用。选择 Workspace->Tools->JSP Execution Monitor;等对话框窗口出现；确认”Retrieve syntax error infromation” checkbox 被 checked。)

你将需要查看任何一个编译错误消息，修改你的.jsp 文件，重复处理。如果你被错误消息弄糊涂了，你可以查看生成的 servlet 代码。这个代码被写进 VA Java 里的”JSP Page Compiler Generator Code”项目。

4．在某些时候，如果你要调试你的 JSP 文件，你需要停止 SERunner。确信”JSP Page Compiler Generator Code”项目被加进 SERunner 的类路径。重启 SERunner，重新访问站点。就象你在你的 servlet 里加断点一样，你可以把它们加进 JSP Servlet。

5．你也许还希望看看页面编译的翻译过程。打开 JSP Execution Monitor(JSP Execution Monitor Options dialog)，在监视器里一步一步地访问.jsp 文件。

你在这个练习里作了什么
这个练习说明了用 JavaServer Pages 来建立“ HTML 模板 ”文件，这个文件会被 JSP 页面编译过程转化为可执行的 Servlets。得到的模板文件(.jsp)可以被 WYSIWYG HTML 编辑工具处理，可以用这个格式来维护而不是代码的格式(也就是 servlet).

不同的 JSP 语法被试探，JSP Declaration,JSP Scriplets 和 JSP Expressions 在被建立的.jsp 文件里用到了。

答案
这里是建立下拉列表的 scriptlet

```
<%
        String[] ages=
            {“Under 5”,”5-7”,”9-13”,”Over 13”};
        for (int i=0;i<ages.length;i++){
            out.print(“<OPTION”);
            if ((CUSTOMER.gerAge().trim()).equals(ages[i]))
                out.print(“SELECTED”);
            out.println(“>”+ages[i]);
        }
>
```

JSPBeans4:JavaServer Pages with Beans

在这个练习里，你将更新一个 JavaBean，用它来在两个 servlets 之间通信，其中第二个 servlet 是从一个 JavaServer Page 生成的。JSP 的目的是显示客户选择的可能要购买的玩具。
许多网页有复杂的布置。这很难在一个 servlet 里用编程的方法表达。这是服务器端脚本技术的主要理由。JSP 编程模型的力量在于隔离可重用的组件(JavaBeans)里的应用代码，而不污染演示 HTML 模板。
你将把一个 JavaBean 的一些状态扩展为属性，以便使用 HTML 模板语法的 JSP 容易地使用这个 JavaBean。你还将用 JSP 标记和 JSP HTML 模板标记(insert、repeat)来扩充 HTML 文件。

第一部分：更新 JKToys JavaBean

1.在这个练习里我们将在 com.ibm.waslab.common 包里工作。我们从在类 JKToyOrder 上打开一个类浏览器开始。选择这个类的 BeanInfo view。这个类是用来维护当前选择的要定购的玩具的。It is populated by the TotalPurchaseServlet。我们会让一个 JSP 表示“ order review ”信息，而不是用现在输出 HTML 和动态内容的 displaySelections 方法。
为了做到这个，我们需要一个新的方法来以”JavaBean”风格的方式循环过一个选择的玩具。这需要把列表作为一个 Indexed Property。

创建一个新的 IndexedProperty,orderedToy，在类 JKToyOrder 里。为了做到这个，单击 Property Wizard 按钮。在结果对话框 check readable,writeable,indexed。提供 Property 的名字为 orderedToy，选择 Property 的类型为 com.ibm.waslab.common.Toy(这个将被向导转化为 Toy[])。对我们的目的而言，uncheck 规定 property 为 bound 的 box。按 Finish.

2.返回到类的 Methods 或 Hierarchy 视图，来实现刚才生成的 4 个方法。如果你看类定义，你会发现一个新的实例变量名字叫 fieldOrderedToy，类型为 Toy[]。这个是 Bean Property wizard 产生的。这个实例变量并不需要，因为玩具已经存在一个实例变量叫 orderedToys,类型为 java.util.Stack。这个实例变量已经在类的其它地方被使用了。
删除变量 fieldOrderedToy，保存类。你将会在步骤 1 里 Property wizard 产生的三或四个附属方法里得到错误标记。
3.编辑四个附属方法的每一个。使用 java.util.Stack 和 java.util.Vector 的正确的方法(例如 CopyInto,elementAt,setElementAt,....)来实现这个四个方法的每一个。(需要帮助的话，看这个练习的末尾的答案。)
应该注意到，setOrderedToy(int Toy)和 getOrderedToy(int)方法也许会抛一个 ArrayOutOfBoundsException 异常。这个异常会触发 JSP 里<REPEAT>标记里循环的结束。

4.验证类 JKToyOrder 实现 java.io.Serializable。

5.最后，输出 com.ibm.waslab.common 包到以下路径：

[x:]\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\servlets

[x:]\WebSphere\AppServer\servlets(只有在要在 VAJava 外面运行时才需要)

第二部分：调用 JSP

1．打开 com.ibm.waslab.servlet2.TotalPurchaseServlet。编辑 doPost()方法。在方法的底部(在另一个块里)，displaySelections 被调用。这个需要被对 JSP 的调用掉换。第一步是把 JKToyOrder 对象放进 HttpSession 对象或 HttpServletRequest 对象。

为了把定单对象加进会话，增加以下的代码：

session.putValue(“OrderedToys”,toysToBuy);

注意，定单从会话里取得的键是”OrderedToys”。

可选的，JavaBean 可以被放在请求范围(而不是会话),把 setAttribute 方法用在 com.sum.serve.http.HttpServiceRequest 对象。

2．下一步是调用 callPage()方法来调用 JSP。CallPage 在回应对象里被调用。callPage 是一个 com.sun.server.http.HttpServiceResponse 类上的方法。你必须先把 res 造型为类型 com.sun.server.http.HttpServiceResponse。以下的代码将足够了(如果你加了一个 import 语句到类定义里面)。

```
HttpServiceResponse response=(HttpServiceResponse) res;
response.callPage(“/JKToys/html/jktoys_display_order.jsp”,req);
```

第三部分：Populating the JSP

1.最后一步是创建 JSP 文件。文件 jktoys_display_order_start.jsp(在 x:\www\html\JKToys\html 目录里找到)包含从 displaySelection 方法里提取的 HTML（从 com.ibm.waslab.servlet2.TotalPurchaseServlet)。你将需要为”OrderedToys”Bean 增加一个 BEAN 标记。这个可以被放在文件的任何地方（在 Bean 被引用以前)。

2.你还将要把 TABLE 的<tr>块包进一个<REPEAT>块。用 insert 标记和 JSP 表达式把动态内容插入 HTML 表格。

注意：你还将要增加一个 scriptlet 来在 REPEAT 块的开头测试一个 ArrayOutOfBoundsException(在任何静态内容被推到回应流里去以前)。也要注意，getPriceAsString()方法没有同一个 Property(在 JkToyOrder 类或 Toy 类)联系起来。<INSERT>标记不能用来显示这个信息，除非方法在每一个各自的类里被 prompted 为一个 Property。如果你不想”prompt “ 这个属性的话，这个方法可以用 scriptlet 来调用。

或者，你可以只用 scriptlet 和 JSP 表达式而不是<REPEAT>和<REPEAT>标记。

3.保存更新过的 JSP 文件为 jktoys_display_order.jsp。(注意新的文件名 ！！！)测试正确的操作。

你在这个练习里作了什么

在这个练习里，你作了一个典型的 Servlet->Bean->JSP“ 控制 ”流。注意，JSP 文件可以被 HTML WYSIWYG 编辑器维护和编辑（同在 Java 方法里管理它不一样)。

答案：

public Toy getOrderedToy(int index) throws

```
        ArrayIndexOutOfBoundsException{
        return (Toy)orderedToys.elementAt(index);
}

public com.ibm.waslab.common.Toy[] getOrderedToy(){
        com.ibm.waslab.commom.Toy[] toys=new
com.ibm.waslab.common.Toy{orderedToys.size()};
        orderedToys.copyInto(toys);
        return toys;
}

public void setOrderedToy(int index,Toy orderedToy)throws
        ArrayIndexOutOfBoundsException{
        orderedToys.setElementAt(orderedToy,index);
}

public void setOrderedToy(com.ibm.waslab.common.Toy[] orderedToy)
{
        orderedToys=new java.util.Stack();
        int max=orderrdToy.length;
        for (int i=0;i<max;i++)
                orderedToys.addElement(orderedToy[i]);
}
```

JSP 的答案参见：x:\www\html\JKToys\jspSolutions\jktoys_display_order_soln.jsp。

Studio5：WebSphere Studio Servlet Wizards

在这个练习里，你将使用 IBM WebSphere Studio 的一个向导来创建一个客户管理界面。产生的 servlet 和 JSP 页面是让站点管理员能查看数据库里所有 JKToys 客户的记录。按照这个这个练习，你将能用 Studio servlet wizard 来建立更复杂的 servlets 和 JSP 的骨架。

对所有的练习，确信你用 USEID 为 id 和 PASSWORD 为口令登录进去的。

1．你将用 IBM WebSphere Studio,version 1.0（从此以后用 Studio 指代），建立一个简单的管理应用。选择 Start,Programs,IBM WebSphere,Studio 1.0,WebSphere Studio，启动 Studio。这将带来 Developer Workbench,Studio 的主窗口。Workbench 给出一个基于项目的网站视图。从 File 菜单选择 New Project 选项创建一个新的项目。在 Name 域，输入 JKToys,按“确定”按钮。

2．建立管理应用的第一步是建立一个 SQL 查询对象。为了做到这个，你将利用 Workbench 提供的 SQL 向导。从 Tools 菜单选择 SQL Wizard...项目。

3．向导显示为一个对话框，有一个”tabbed”象 notebook 的外观。当它最初弹出时，对话框显示 Welcome 页面，问将要产生的 SQL 命令文件的名字。输入 ViewAllCustomer 到 name 输入域，按 Next 按钮。

4．在下一页(Logon)，输入 jdbc:db2:JKToys 到数据库 URL 输入域；USERID 到用户输入域；PASSWORD 到口令输入域。在 Driver 输入域选择 IBM DB2 UDB local。输入这些值以后，按 Connect 按钮。这会连接数据库，取得 JKToys 数据库的元数据。

5．确信 Select Statement 类型被 checked。还 check USERID.CUSTOMER 表。按 Next 按钮。

6．因为只有一个表，Join 是不需要的。按 Next 按钮。

7．对 Columns 入口，先按 Select all 按钮，然后是 Add 按钮。然后按 Next.

8．你在建立的查询要从数据库取得所有的行。所以，你不要 where 子句。按 Next 按钮。
9．按 Finish 来完成 SQL 语句。
10．接着我们将运行 Data Access Servlet Wizard。为了启动向导，从 Tools 菜单选择 Studio Wizard...项目。Studio Wizard 是一个同 SQL wizard 相似的对话框。
11．在 Welcome tab，选择 Data Access Servlet，按 Next 按钮。
12．在“第二个”Welcome tab，输入 ViewCustomer 到 Name 输入域，按 Next。
13．在 SQL 语句页面，“select from an active project”选项应该被选中，ViewAllCustomers.sql 应该被选中。按 Next。
14．在输入页，确信”Yes please”被选中。按 Next。
15．对 InputLayout,输入 :为 Page title 输入 View All Customers;清除 User prompt 的输入 ;Check Submit button check box,把文字改为 View All Customers;uncheck Reset button 输入。按 Next。
16．对输出页面接受缺省值（所有列）。按 Next。
17．在 Heading 输入域增加文字 JKToys Customers.然后选择输出字段(每次一个)，用上下箭头移动字段，使得它们可以以以下的顺序出现：LNAME，FNAME，AGE，ADDR，CITY，STATE，ZIP，CUSTNO。按 Next。
18．选择不要创建一个 custom Error Page。按 Next。
19．按 Finish 来生成 Servlet,DataBean,Output JSP 和 Input HTML 文件。一旦确认对话框消失，你应该看到以下文件在 Workbench 在 JKToys 项目里：ViewAllCustomers.sql;ViewCustomersOutputPage.jsp;ViewcustomersInputPage.html 。如果你看 classes/JKToys 文件夹，你还会发现：ViewCustomersServlet.java;ViewCustomersServlet.servlet;ViewCustomersServletBean.java;ViewCustomersServlet.class;ViewCustomersServletBean.class。
20．在发表这个应用之前，编辑 ViewCustomersServlet.servlet，在 Workbench 里双击这个文件。这个文件是 XML 的 servlet 配置文件。我们要改变”default-page”的路径。在第 5 行，在<URL>后面，插入”/JKToys/html”。缺省页面的路径现在应该是/JKToys/html/ViewCustomersOutputPage.jsp。
21．下一步是发表到 JKToys 站点。为了作这个，先创建一个“Publishing Server”。从 Options 菜单里选择 Publishing servers...选项。
22．按 Add 按钮。在 New Publishing Server Settings 对话框，输入 JKToys 到 Name 输入域。Check Local radio button。然后，选择 html 目标，按“浏览”按扭。在文件浏览器里，定位/打开 x:\www\html\JKToys\html(x 是 IBM HTTP Server 文件根所在的驱动器字母)。然后，选择 html 目标，按“浏览”按扭。在文件浏览器里，定位/打开 x:\WebSphere\Appserver\servlets。最后，按“确定”按钮。
23．为了发表，在 Workbench 左手的树视图里选择 JKToys 站点。从 File 菜单选择 Publish 选项。当 Select Publishing Server 对话框弹出来，选择 JKToys 作为服务器；按 Continue 按钮。然后按 Publish 按钮。最后按“确定”让对话框消失。
24．为了测试这个应用，我们将在 VA Java 外面运行，使用 IBM HTTP Server。确信 SERunner 被停止。然后启动 IBM HTTP Server。
25．在一个浏览器，打开 URL http://127.0.0.1/JKToys/html/ViewCustomerInputPage.html。按 View All Customers 按钮。这将调用 servlet，它将创建一个访问 bean 的实例，ViewCustomersServletBean，然后调用执行 Bean 的方法来发布 SQL 查询。Bean 被放在请求范围内，键是 viewCustomersServletBean。Servlet 对 ViewCustomersOutputPage.jsp 执行一个 callPage，在一个 HTML 表格里显示 SQL 查询的输出。
26．最后的步骤是看产生的代码。尤其是，看输出 JSP 和 ServletBean.

IBM WebSphere Studio 里的两个不同的向导被用来建立小的网络应用来查询客户数据库，返回内容作为动态内容。有许多不同的工具可用来执行业务数据的 Web 网关。而向导能有效地建立 servlet、”servlet bean”、JSP，从而支持 JSP 编程。

JDBCLab:用 JDBC 访问一个数据库

在这个练习里你将学习使用 JDBC 访问数据库和插入一个记录到数据库的基础知识。
在这个练习里，你将写 RegistrationServlet 类的 register 方法。这个类和你在 Servlet1A 练习里用的一样。在这个练习里，你将使用数据库的功能。这个练习将展示实现用户需求的步骤。

第一部分：理解 regsiter()方法在 RegistrationServlet 中的工作原理

在这个练习里，你将在 com.ibm.waslab.JDBC 包上工作。扩展 RegistrationServlet 类，理解 register()。这个方法抛一个 SQLException。完整的方法声明是：

```
public synchronized void register(Properties formInput)
        throws SQLException{


}
```

register()方法在 servlet 取得控制时从 doPost()方法里被调用。register()方法实际上写一个新的记录到数据库。我们将在第二部分写 register()方法。同时，我们需要确信一些设置步骤已经完成。

1．打开 init()方法，在 super.init()的调用后面输入以下的代码：

```
//Load JDBC driver for DB2
try
{
 Class.forName(JKToysDBInfo.gerDriver());
}
catch (ClassNotFoundException e)
{
 erroLog("JDBC Driver not found"+e);
}
```

保存 init()方法。记住，在驱动器管理可以得到一个连接以前，一个正确的数据库驱动器必须被 servlet 装载。每一个 servlet 只在 servlet 的 init()方法里作一次。这个方法并不真地建立一个连接，它只是允许连接被建立。

注意：在这个情况下，我们装载 DB2 app 驱动器。这个驱动器是在数据库装在 servlet 将要运行的同一台机器上时使用的。如果你在访问另一台机器上的数据库，你要使用 net 驱动器：

```
COM.ibm.db2.jdbc.net.DB2Driver
```

当用 net 驱动器得到一个连接，你需要在 getConnection()里使用的 URL 里提供更多的信息。数据库所在的机器名或 URL，和 DB2 Java Gateway 侦听的端口号。一个有效的 URL 参数看上去是这样的：

```
jdbc:db2://servrid:8888/databasename
```

第二部分

1．现在，回去，重新打开 register（）方法。代码的第一行创建一个数据库连接：
Connection conn=DriverManager.getConnection(URL,USER,PASSWORD);
你用驱动器管理器的静态方法 getConnection()，把数据库的 URL，一个有效的用户 ID 和口令传递给它。url,userid 和 password 在类被装载时从一个属性文件里得到(看静态变量声明)。
2．现在到了有挑战性的部分了。我们要作的第一件事是找到已经被分配掉的最后一个客户号码，这样我

们可以分配给正在注册的新客户一个新的客户号码（比以前的最高的还要高）。我们要存储这个新的客户号码在一个 int 变量名为 nextId。为了做到这个，你会需要确信使用 Statement 类和 ResultSet 类。以下的 SQL；

“SELECT MAX(CUSTNO) FORM”+DBOWNER+”.CUSTOMER”

将允许你获得当前最高的客户号。试试自己写这个代码，基于课程笔记的例子。如果你需要帮助，看答案页。

3．接着你将用一个 prepared statement 对象来把客户属性对象里的信息插入数据库。输入以下的行：

```
//Insert record in the database
  PreparedStatement insertStatement=
conn.prepareStatement(“INSERT
INTO”+DBOWNER+”.CUSTOMER(FNAME,LNAME,ADDR,CITY,STATE,AGE,ZIP,CUSTNO)”+”VALUES(
?,?,?,?,?,?,?,?)”);
```

上面的行要求连接创建一个 prepared statement 对象叫 insertStatement。SQL 语句作为参数被传递。数据库行的每一列的值用问号代表。每一个问号必须被一个正确类型的值代替。输入以下行：

```
insertStatement.setString(1,formInput.getProperty(“FNAME”));
insertStatement.setString(2,formInput.getProperty(“LNAME”));
insertStatement.setString(3,formInput.getProperty(“ADDR”));
insertStatement.setString(4,formInput.getProperty(“CITY”));
insertStatement.setString(5,formInput.getProperty(“STATE”));
insertStatement.setString(6,formInput.getProperty(“AGE”));
insertStatement.setString(7,formInput.getProperty(“ZIP”));
insertStatement.setInt(8,nextId);
```

4．insertStatement 执行对数据库的更新。输入下面两行：

```
insertStatement.executeUpdate();
```

如果有错误出现在创建连接时，语句或访问数据库，这个方法会抛出一个 SQLException。

5．最后你需要存储客户号码到 formInput 属性对象。输入下面行：

```
formInput.put(“CUSTNO”,new Integer(nextId).toString());
```

6．保存方法。应该没有错误。

7．接着你将需要为 JKRegister servlet 编辑 WebSphere 配置文件。（编辑 ”[x:]\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\properties\server\servlet\servletservice\serrvlets.properties”.）

改变看上去这样的行：

```
servlet.JKRegister.code=com.ibm.waslab.servlet1.RegistrationServlet
```

为这样：

```
servlet.JKRegister,code=com.ibm.waslab.JDBC.RegistrationServlet
```

8．如果你的 webserver 在运行，关掉它。在 VAJ，启动 SERunner。打开你的 web 浏览器，到 JK Toys 网站的注册页面。在表格里输入注册信息，按注册按钮。你的新的 servlet 将被调用。登录进 JKToys 站点确信注册已经发生。你的新的客户号码应该能在数据库找到。

在这个练习你作了什么

在这个练习，你编写了 Registration Servlet 的 register 方法。这个方法用 JDBC 来创建一个新的客户号码，注册一个新的客户到数据库。现在你有了在你的 servlets 里访问数据库的基本知识。

答案

以下的代码创建一个 SQL 语句，执行在前一页里定义的 SQL 查询。

```
        //Get next customer number
        Statement sqlStatement=conn.createStatement();
        ResultSet result=sqlStatement.executeQuery("SELECT MAX(CUSTNO) FROM"+DBOWNER+".CUSTOMER");
```

查询的结果被放在 ResultSet 对象叫 result 里面。结果的集合包含一个游标，最初指向返回的第一行前面。为了得到查询的值游标必须用 next()往前移动。用这个方法，游标只能向前移动。下面几行移动游标到返回的值，增加 1。

```
int nextId=0;
if(result.next()= =true)
{
        nextId=result.getInt(1)+1;
}
```

if 检查确定是否有行被返回。如果没有，结果为 false。我们不处理错误的情况，但是它可以简单地用把 nextId 设置为 1 来处理。我们还可以用 ResultSet 的方法 getInt(String)，列的名字为"CUSTNO"，但是因为我们知道只有一列会被返回，我们选择用 getInt(int)..

安全：Servlet 安全

这个练习展示你怎样使用 WebSphere 的 Servlet 安全特性来创建一个用户组，一个用户集合，和一个访问控制列表，然后增加一个 servlet 到被这个访问控制列表保护的资源列表。

这个练习将有三个部分。在第一个部分我们将作一些初始设置，包括取得前一个练习(WebSphere studio lab)里创建的 servlet，给它一个名字。第二个部分将展示怎样在 WebSphere 里创建用户和用户组，最后一个部分会介绍创建 ACL 和把资源(包括 servlet)指派到 ACL。

第一部分：初始设置和 Servlet 命名

1. 这个练习从你以前完成的 WebSphere studio 练习(练习 5)的结果开始。如果你没有完成那个练习，解开文件 c:\Lab5Solutions.zip 到 c:\。
2. 把文件 ViewCustomersServlet.servlet 从 \WebSphere\Appserver\servlets\JKToys 目录拷贝到 \WebSphere\Appserver\servlets 目录。
3. 下一个任务是让 JKToys.ViewCustomersServlet 类成为一个 named servlet。WebSphere 的访问控制特性依靠 servlet 命名——对没有命名的 servlet 它们不工作。为了做到这个，我们将用 WebSphere Application Server 管理工具工作。为此你需要从 IBM HTTP Server 启动 Application Server(记住首先在 VA Java 里停掉 SERunner)。在 Web Server 和 Application Server 被启动以后，从你的浏览器用 http://localhost:9527 登录进 WebSphere 管理页面。记住缺省的 WebSphere 管理用户的 ID 和口令为 admin,admin。当管理页面打开，打开标签为"Servlets"的树节点，选择"Configuration"来打开 Servlet Configuration page。
4. 按"Add"按钮。我们将命名这个 servlet 为"ViewCustomersServlet"。在"Servlet Name"域键入"ViewCustomersServlet"。View Customers servlet 的类是"JKToys.ViewCustomersServlet"。在"Servlet Class:"域键入这个，按"Add"按钮。(如果你要，按"Test"按钮来测试这个 servelt,然后释放说测试被通过的对话框。)
5. 我们接着需要作一些整理工作，检验我们的命名是否工作正常。在你的硬盘上的[x:]\www\html\JKToys\html(或 jspSolutions)目录里找到"ViewCustomersInputPage.html"文件。用 notepad 或别的文字编辑器打开那个文件。找到行"<FORM METHOD=POST ACTION="/servlet/JKToys.ViewCustomersServlet"> "改成，"<FORM METHOD=POST ACTION="/servlet/ViewCustomersServlet"> "。保存这个文件。现在，往你的浏览器里输入以下的 URL："http://localhost/JKToys/html/ViewCustomerInputPage.html".。确信"View Customers"按钮出现，一旦按钮

被按下，包含客户信息的表出现。

第二部分：增加用户和用户组

1．既然我们的重命名已经完成，我们可以开始向我们的 ViewCustomersServlet servlet 增加口令保护。回去到 Administration GUI（如果你测试和命名 servlet 用了同样的浏览器，你将需要重新登录进 WebSphere 管理）。从树里选择”Security”节点。你会看到 Security tree 有四个节点：

Users

Groups

Access Control Lists

Resources

我们将在我们的练习里用所有四个 tabs。

2．选择“ Users”。一个表会打开，显示 WebSphere 已经在它的内部数据库里有的用户的列表。一个缺省的 WebSphere 安装会有两个用户，” jeeves”和”admin”。我们将增加一个新的用户叫”sales”。按”Add”。一个对话框显示出来，问用户的名字和口令，要你确认口令。在用户名域输入“ sales”，在口令和确认口令域都输入”bluefish”。然后按对话框里的“ 确定 ” 按钮。用户”sales”将被加进表。在一个实在的应用里，我们显然不用一个可在字典里找到的口令——你可以用字母和数字的随机组合。

3．我们将增加一个用户组到 WebSphere。我们要允许销售人员和管理人员都可以列出客户——销售人员会看客户列表，管理员会需要在数据库崩溃时维护客户列表。我们要创建一个新的用户组名字为“ customer-access”包含所有这些用户。按”Group”。显示器有三个列表。最上边的是 ”Defined Groups”(现在忽略”realm”下拉菜单——让它保持”DefaulgRealm”设置)。左边的按钮是增加和删除组。按“ Add ” 按钮。一个对话框显示出来，向你要要增加的组的名字。输入”customer_access”按“ Add”按钮。”customer_access”组将被加进列表。我们现在已经准备好用底部的两个列表。左边的是成员列表，现在还空着。在右边的列表(“non-memebers”)里选择”admin”用户，按”<Add”按钮。这个用户会被加到用户组。对”sales”用户作同样的事。我们现在完成了创建用户组的工作。

第三部分：创建 ACL 和增加资源

1．我们现在准备好向 WebSphere 的 ACL 增加新的 ACL 了。按”Access Control List”.显示的面板和 Groups 的面板相似，最上边的下拉菜单显示 realms(再一次，让它保持‘ defaultRealm”)，最上边的列表显示 ACL’s 的列表。应该有一个 ACL，简单的叫做 ”defaultAcl” 。我们将创建一个新的 ACL 叫做”customer_access_ACL”。按左边的 Add 按钮。一个对话框会显示出来，要求要加入的新的 ACL 的名字。在文字域输入”customer_access_ACL”，按”Add”。一个新的 ACL 叫做”customer_access_ACL”的 ACL 将被加进列表。

2．这里我们已经准备好向用户组指派权限。在 ACL 列表下面有一个树形图包含三个分枝：Users,groups,computers。所有这些分枝应该都是空的(没有儿子)。按中间的分枝(Groups)，然后按这个视图左边的”Add”按钮。这个将带出来一个对话框，让你向不同的用户组指派权限。

3．现在对话框出现了，下一步是在我们的 ACL 里向新的用户组指派权限。在”Assign Permission for:”对话框的上面有两个 radio button。这两个选择是”files and folders”和”servlets”。你也许以为我们要”servlets”，但是相反我们要选择”files and folders”。”servlets “ 选项是指一个运行的 servlet 拥有的权限，例如打开 sockets，读文件等。我们要设置访问一个 servlet 的权限；这是 HTTP 协议下的文件访问。另一组 radio buttons 让你选择你是否要为”Users”,”Groups”,或”Host”设置权限。选择 ” Groups”。因为只有一个组，它(customer_access)应该在第二组 radio buttons 下面的”principals”列表里被选中。在 ”principals”列表下面

是一组 check boxes,标着”GET”,”PUT”,”POST”,”DELETE”。check”GET”和“ POST”check boxes，按 check boxes 下面的“ 确定 ”按钮。这给这个用户组对这个 ACL 保护的文件，文件夹和 servlets 使用 ”GET“ 和 ” POST “ 方法的权限。

4．我们现在要结束我们的练习。按”Resouces”。在”realm”下拉菜单下面你将看到一个空的表。按表左边的”Add”按钮。这个会带出来一个对话框，让我们增加资源到我们创建的 customer_access_ACL 保护的资源列表里。

5．当”Protect A Resource”对话框出现，改变 ACL 下拉菜单为”customer_access_ACL”。然后在”Resource to Protect”下面，选择”Servlet” radio button，从下拉菜单选择”ViewCustomers”servlet，然后按 “ 确定 ” 释放对话框。你会在 Resource 表里看到资源”ViewCustomersServlet”和”customer_access_ACL”ACL 之间有一个联系。我们现在结束了编辑我们的 ACL 和资源，准备测试了。

6．在一个浏览器，输入 URLhttp://localhost/JKToys/html/ViewCustomersInputPage.html。”View Customers” 按钮出现，按它。一个 challenge dialog 应该出现，向你要名字和口令。如果对话框没有出现，重新看看你在这以前的步骤。如果你不能找到错误，向你的实验辅导员请求帮助。首先输入一个无效的用户名和口令。你应该接着收到 challenge dialog。输入”sales”和”bluefish”给名字和口令。现在，客户的表应该象在这个练习的开头一样显示出来。如果 challenge dialog 又出现了，重新输入名字和口令。challenge dialog 通常会给你三次机会输入正确的名字/口令对，如果都失败，浏览器会显示错误 “ 401 Unauthorized”。

在这个来年练习里，我们创建用户和用户组，创建一个 ACL 和指派一个 servlet 资源到 ACL。我们已经展示怎样给一个 servlet 增加口令保护。