# 第1章 Duke实例一览

## 1.1 从J2EE谈起

谈到Duke在线银行实例，我们得先从J2EE谈起。谈起J2EE，我们最好先看看这4个字所代表的含义。第一个J代表Java。Java在IT工业界享誉甚久，无须过多阐述。简而言之，Java是一种划时代的计算机语言，并且远远超出一般计算机语言的范畴，是一个完整的计算机运行环境。第二个数字2代表Java的第二版，也是Java趋于成熟的一个里程碑。后面的两个E代表企业级版本（Enterprise Edition）。它有别于Sun的另外两个重要的规范版本，一个是J2SE，一个是J2ME。前者含有S的是Java的标准版，也是Java的基础版，我们通常所说的Java即指J2SE而言；后者含有M的是Java的微型版，也是Java用于手机、掌上型电脑等各种手执无线设备的一套规范。这三套标准规范，以J2SE为基础，向上J2EE涵盖了企业级的应用，向下J2ME涵盖了所有手执无线设备的应用。从上到下，可谓囊括了几乎所有主要的IT应用。J2SE已有不少专书谈及，J2ME也需要专书讨论，它们均不在本书范围之内。本书是针对J2EE，即Java的企业级版本而展开的。

那什么是“企业级”？它与标准Java又有什么区别呢？

我们知道，标准版Java包括了一整套现代面向对象的语言规范。它主要是基于两种重要的面向对象的语言，C++和SmallTalk发展而来。C++是在Java推出之前在工业界最为流行的面向对象语言，而SmallTalk虽不如C++那样被广泛应用于IT工业界，但它不像C++被称为“杂交的”面向对象语言，SmallTalk是一个“纯正的”面向对象语言。如果说Java只是另一种新的面向对象语言，它决不会在当今IT工业界产生如此强烈的影响。除了面向对象的所有特征以外，Java还包含了一整套的核心类库和可任意扩展的专项类库以及完整的开发工具和虚拟机，所有这些组成了Java一个完整的开发和运行环境。其中尤为值得一提的是虚拟机概念，它可称得上是一个划时代的概念，它通过二次编译实现了WORA（Write Once，Run Anywhere）。在弄清Java的标准版的重要特征之后，我们再来谈谈什么是Java的企业版。

首先，Java的企业版包括了标准版的全部内容，换句话说，它就是Java标准的延伸和扩充。如果没有J2SE的开发和运行环境，就无法想像J2EE企业级的开发和运行，所以说Java标准版是基础，但Java的企业版并不是标准版的简单延伸和扩充，否则我们只可以称它为一个新的标准版“J3SE”。Java的企业版是一个大的跨越，它针对整个企业范围规范了一个新的体系结构，即一个分布式、多层次的体系结构。在这个新的架构平台上，孕育着一个又一个新的现代软件技术。

在谈及分布式、多层次体系结构之前，让我们简单回顾一下计算机体系结构的进化历程。我们知道，早期的计算机体系结构是以中央控制型主机和终端机组成的集中式体系结构，所有的中央处理器、存储单元均放在主机上，终端机只是一个简单的输入装置和显示器。因为所有的系统软件均放在主机上，任何应用软件均可以直接外加在系统软件上，所以这种集中式的体

系结构对软件系统的要求也就相对简单得多。随着 PC 机的发展并逐步取代无智能的终端机，PC 机自身已能完成一定的工作，这样，一部分软件功能就渐渐从中央控制的主机中分离出来。随着 PC 硬件功能的不断增强，促使软件的应用也越来越广泛，对软件的需求也越来越大。从主机上分离出来并移植到 PC 机上的软件，大部分都是与用户有直接界面的应用软件。这部分应用软件使 PC 机形成了有丰富的用户接口的客户软件。并且，随着小型机、PC 机和工作站的功能渐渐接近原来的中、大型主机，另一部分软件功能也渐渐从中央主机中分离出来，移植到 PC 机和工作站上，从而形成了一系列客户端软件和服务器，如常用的文件服务器、数据库服务器、网络服务器以及相应的客户端软件等。众多的客户机通过局部网络与一个或多个服务器联接，由服务器给它们提供相应的服务，这样，就从“中央集权”式的体系结构分为前台和后台的二级模式，也称为客户机－服务器的请求响应体系结构。前端 PC 机作为客户机，主要提供客户界面和做一部分简单处理，再将客户请求送交后端服务器进行处理；后端服务器响应前端客户机的请求进行处理后，将结果返回给前端客户机。这是 PC 机硬件功能的增强带来计算机体系结构变化的一个方面。

另一方面是网络的发展。从单一主机加终端机的星型网络演变到多个客户机加服务器的局域网，再发展到跨地区的广域网，特别是网络与网络相联，形成一个网中套网，网外有网的全球型因特网，使得计算机体系结构从客户机－服务器的二层体系结构朝着分布式、多层次的方向发展。硬件和网络像两只高速运转的车轮，载着软件向前飞快发展。软件从“散兵游勇”式的单一应用，进展到覆盖一个办公室、一个部门进而到整个企业，跨地区、跨国界的应用。

J2EE 就是在这个大环境下应运而生，形成的第一个面向企业级的分布式、多层次的软件体系结构规范，并很快被工业界广泛采用，成为企业应用开发的标准。J2EE 以 Java 技术为基础，包含了一整套的服务、应用程序接口和协议标准以及众多的支持技术。随着越来越多的第三方软件厂商对 J2EE 的支持，J2EE 已经被广泛用来开发各种企业应用系统和支撑企业应用系统的应用服务器，像著名的 BEA 公司的 WebLogic 和 IBM 公司的 WebSphere，就是基于 J2EE 技术标准构造的两个用途最广的应用服务器。

## 1.2 Duke 在线银行实例

开始学 J2EE 时，总会让人有一种茫然无从着手之感。概括来看，原因有二：一是 J2EE 体系宏大，一大堆令人听之生畏的名词和概念纷至沓来，诸如，企业级平台、组件构架、分布式分层次体系结构、框架模式，还有中间件、组件、容器、应用服务器等，着实使不少初学者眼花缭乱，望而生畏；二是缺少完整和详细的实例。一般的书，大都以讲解概念为主，间或穿插一些不相连贯的小例子，很难找到一本从头到尾、从里到外来讲解一个完整的实例的书，就像有一位老师能面对面、手把手地教你一样。当然，这样一本书难度较大。为什么这样说呢？因为首先要有一个实例，这个实例不能太大，也不能太小；不能太繁，也不能太简。太大则一本书无法容下，太小则无法包容众多的方面，太繁则不适合初学者，太简则不足以讲解清楚。第二，要讲解实例，必须要有公开的全部实例的源代码。要作者自己去闭门造车写全部源代码，还包括测试、安装、部署等，需要有丰富的 IT 工业界实际经验且不谈，光是花费大量的时间和精力就让作者难以完成，更不谈写出的实例是否适合初学者的需求。IT 工业界实际应用的案例虽然完整，但大都是专用的，且规模很大，运行环境复杂，加上版权、保密等因素，实际

上无法利用。

Sun 作为 J2EE 的创始者，以其独到的眼光，在其最新《J2EE 学习指南》中为初学者提供了这样一个完整的实例——Duke 在线银行实例。这是一个大小适中、功能齐全、结构严谨、分布式、多层次的企业应用实例。说它是“实例”，毫不为过，其核心包括了企业级应用的基础框架和各个主要层面。在这个基础框架上，不仅可以构建各种不同的分布式、分层次的应用模式，也可以任意扩充各种专用功能，可以说是“麻雀虽小，五脏俱全”。

从技术层面来说，Duke 应用实例，既涵盖了前端的应用客户和 Web 客户，又详细叙述了 Web 组件、企业 Bean 组件的中间件技术，也包含了应用服务器、容器、数据库、生成和部署工具等。同时，《J2EE 学习指南》也列举了如何编译、生成、打包、部署、运行该实例的具体执行步骤。如果简单划分的话，前端的应用客户和 Web 客户属于客户层，后端的数据库属于应用数据层，其余的组件、容器、应用服务器都属中间层，形成了一个典型的分布式、三层体系结构。如果在中间层内再进一步划分，则可演变为多层模式，比如最接近客户端的可划分为显示层，通过显示层可调用中间的事务处理层、服务层，事务处理层又可调用中间的服务层。针对不同的事务，可有相应的事务处理和服务，不同的层次相互独立，又可相互调用。它们可以在一台机器上，使用一个应用服务器，也可以在不同地域，使用多个应用服务器。事务处理器和服务层可调用数据访问层，从不同的数据源中存取数据，常用的数据源有：数据库、文件系统、消息队列等。

从事务层面来说，Duke 实例包含了一般在线银行的最基本的事务功能，如账户列表、查找并显示历史数据、转账、存取等。另外，它还包括了顾客账户管理程序。当然，Duke 实例的图形用户界面远不能与实际的商业在线银行应用系统相比，因为 Duke 实例的重心在中间件、应用服务器端，对图形用户界面感兴趣的读者来说，这里正好给他们留有一个可施展的空间。另外，有兴趣的读者还可以基于此例，扩充更多的功能。

## 1.3　Duke 的运行环境

为使读者真正把握和透彻地了解 Duke 实例，我们分下列 3 个方面进行讲解。第一方面是搭建 Duke 在线银行实例的运行环境。要让一个实例有实际价值，必须让它能运行起来，而要让一个实例运行起来，则首先必须设置它的运行环境。本节将带领读者从零开始，一起动手，从无到有地来搭建 Duke 在线银行实例的运行环境。第二方面是通过实际运行的用户界面来了解事务管理及其功能。这方面 Duke 实例相对比较简单，因其重点突出在企业级的中间件技术上。下一节，我们将向读者展示一个实际运行的 Duke 实例全景。第三方面是在实际开发和运行环境中学习源代码。这就是本书其余章节的任务了。

### 1.3.1　Duke 的生存空间

为了建立 Duke 的执行环境并创建、部署和运行 Duke，必须配备下列软件：

- J2SE JDK
- J2EE JDK
- Ant

- Struts
- J2EE Tutorial

J2SE JDK 可以从 Sun 网站的下列网址下载：

http：//java.sun.com/j2se/download.html#sdk

J2EE JDK 可以从 Sun 同一网站的下列网址下载：

http：//java.sun.com/j2ee/download.html#sdk

安装这两个 JDK 非常容易，只要双击.exe 文件就可以完成，但必须先安装 J2SE JDK，然后再安装 J2EE JDK。

Ant 是一个基于 Java 的便携式建造工具，但它远比其他建造工具为好，我们后面还会详细谈到。Ant 是在 Apache 软件基金会的 Jakarta 项目主持下产生的。Ant 可以从下列站点下载：

http：//jakarta.apache.org/builds/jakarta-ant

Struts 是一个在开发 Web 应用中非常有名和实用的“模型－视图－控制器（Model-View-Controller）”框架，它也简称为 MVC。它主要使用了 Java Servlet 和 JSP 技术。如同 Ant 一样，Struts 也是在 Jakarta 项目主持下产生的，它可以从下列站点下载：

http：//jakarta.apache.org/builds/jakarta-struts

当我们下载并安装好两种 JDK，Ant 和 Struts 后，就可以从以下的网址下载《J2EE 学习指南》：

http：//java.sun.com/j2ee/download.html#tutorial

安装学习指南只需要解压下载的学习指南包就行了。图 1.1 显示了学习指南安装后的目录结构。

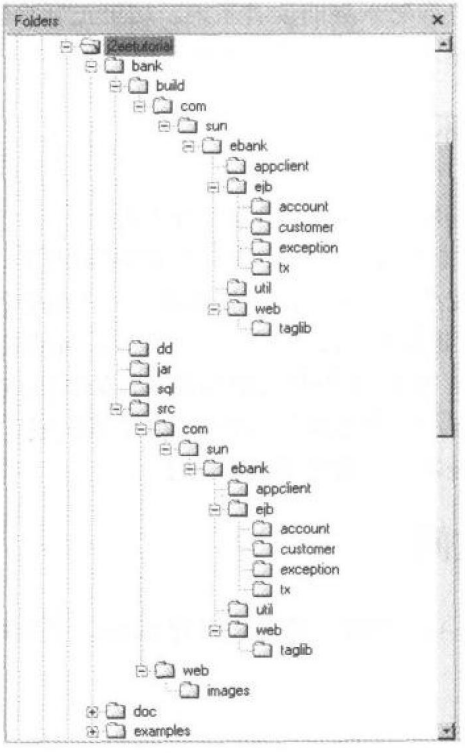


图 1.1　J2EE 学习指南的目录结构

在《J2EE 学习指南》目录下有 3 个子目录，其中 doc 目录下包括了学习指南的所有文档；examples 目录下包括了《J2EE 学习指南》所有的单个实例；我们最关心的、包含Duke 银行实例所有应用的文档在 bank 目录下，包括 ant 的一个 build.xml 文件和 5 个子目录。其中 build.xml 是一个非常重要的文件，我们以后会详细解释。bank 的 build 子目录储备了所有构造后的文件，我们可以看到 build 目录下的所有子目录与 src 目录下的是相同的，其主要文件是.class 文件。“dd”是 deployment descriptors 的缩写，表示部署描述文件。该子目录包含了对 Duke 系统、客户端、Web 以及 EJB 应用的部署说明。jar 子目录包含了所有的.jar、.war 以及.ear 文件。sql 子目录提供了进行操作数据库的 SQL 语句。src 子目录包含了 Duke 银行应用所有的源代码。因为这个小节的目的在于设置 Duke 的运行环境，所以这里不讨论所有的细节，我们将在后面的章节逐步阐述。

## 1.3.2　包装 Duke

安装运行环境的第一步就是编译所有的 Duke 组件，然后将它们打包。我们可以按以下步骤进行。

- 编译企业 Bean 组件

打开另一个命令提示窗口，并转到安装学习指南后所产生的子目录 j2eetutorial\bank 下，然后键入下列命令行：

```
ant compile-ejb
```

这个命令是要使用建造工具 Ant 编译所有 Duke 的企业 Beans 组件。在该命令执行后，我们转到 j2eetutorial\bank\build\com\sun\ebank\ejb 子目录下，可以找到所有生成的EJB class 文件。

下面的所有命令都可以使用同一个命令提示窗口，并且都在同一个目录 j2eetutorial\bank 下执行 Ant 的命令。有关 Ant，我们还会在第 2 章的第 2.4 节中谈到。

- 包装企业 Bean 组件

当企业 beans 的编译完成以后，下一步就可以键入如下命令行来将企业 beans 打包：

```
ant package-ejb
```

上述命令将我们在前面的步骤中得到的 EJB class 文件再加上部署描述文件，分别封装到以下的 EJB JAR 文件中：

account-ejb.jar　　tx-ejb.jar

customer-ejb.jar

这些生成的文件都保存在 j2eetutorial\bank\jar 子目录下。

- 编译 Web 组件

我们可以键入以下的命令行来编译 Web 客户程序：

```
ant compile-web
```

在命令执行后，所有的与 Web 的组件有关的.class 文件都会放在 j2eetutorial\bank\build\com\sun\ebank\web 目录中。

- 包装 Web 组件

完成编译 Web 组件后，我们可以将 Web 组件打包。由于 Web 组件使用 struts 标记库，所以，我们首先需要在目录 jakarta-struts-1.0.2\lib 下将 struts-logic.tld 和 struts.jar 文件拷贝到目录



此为试读，需要完整PDF请访问：www.ertongbook.com

j2eetutorial\bank\jar 下，然后键入如下命令：

```
ant package-web
```

在执行完该命令后，所有的 servlet 类、JSP 文件、JavaBean 组件类、标签库以及 Web 应用部署描述文件都被封装到 web-client.war 文件中，该文件也被放到目录 j2eetutorial\bank\jar 下。

- 编译 J2EE 的应用客户程序

下一步我们可以键入如下的命令来编译 J2EE 的应用客户程序：

```
ant compile-ac
```

在该命令执行后，所有的与应用客户相关的.class 文件将被放到以下目录中：

```
j2eetutorial\bank\build\com\sun\ebank\appclient
```

- 包装 J2EE 应用客户程序

完成编译 J2EE 应用客户程序后，我们可以键入如下命令将 J2EE 应用客户程序打包：

```
ant package-ac
```

该命令将生成一个 app-client.jar 文件，并将该文件放到同一目录 j2eetutorial\bank\jar 下。接着键入如下命令：

```
ant setruntime-ac
```

该命令将 j2eetutorial\bank\dd 目录下的一个运行部署描述文件 runtime-ac.xml 添加到文件 app-client.jar 中。

- 包装企业应用文档（Enterprise Archive）

最后一个步骤是创建 Duke 银行应用实例的企业应用文档。这一文件将把上述步骤产生的企业 bean 组件包、Web 组件包、J2EE 应用客户程序包再加上部署描述文件都封装在一起。

这一步骤，我们可以键入如下的命令来执行：

```
ant assmeble-app
```

该命令执行后，在同一目录 j2eetutorial\bank\jar 下将生成最后的 DukesBankApp.ear 文件。

像封装 J2EE 应用客户程序一样，我们要将 j2eetutorial\bank\dd 下的一个运行部署描述文件 runtime-app.xml 添加到 DukesBankApp.ear 文件中，键入如下命令来执行这一添加任务：

```
ant setruntime-app
```

这样，我们就完成了 Duke 所有的编译和包装的步骤。

### 1.3.3 部署 Duke

在 Duke 应用实例的编译和打包完成后，下面我们将要部署 Duke 银行应用系统。所谓部署，就是将我们在前面最后一步生成的 Duke 企业应用文档装载配置到 J2EE 应用服务器上。在部署前，让我们首先来做 3 件事。打开一个命令提示窗口，并转到 J2EE SDK 的 bin 子目录下，然后在命令提示符下分别运行 3 个批处理文件，其具体运行步骤如下。

1. 运行 j2ee.bat 文件来启动 J2EE 应用服务器，如图 1.2 所示键入以下命令：

```
j2ee  - verbose
```

2. 在 J2EE 应用服务器完全启动后，打开另一个命令提示窗口，并转到 J2EE SDK 的 bin 子目录下运行 deploytool.bat 文件来启动部署工具，只需键入如下命令：

deploytool

即可。图 1.3 显示了 deploytool 启动后运行的画面。

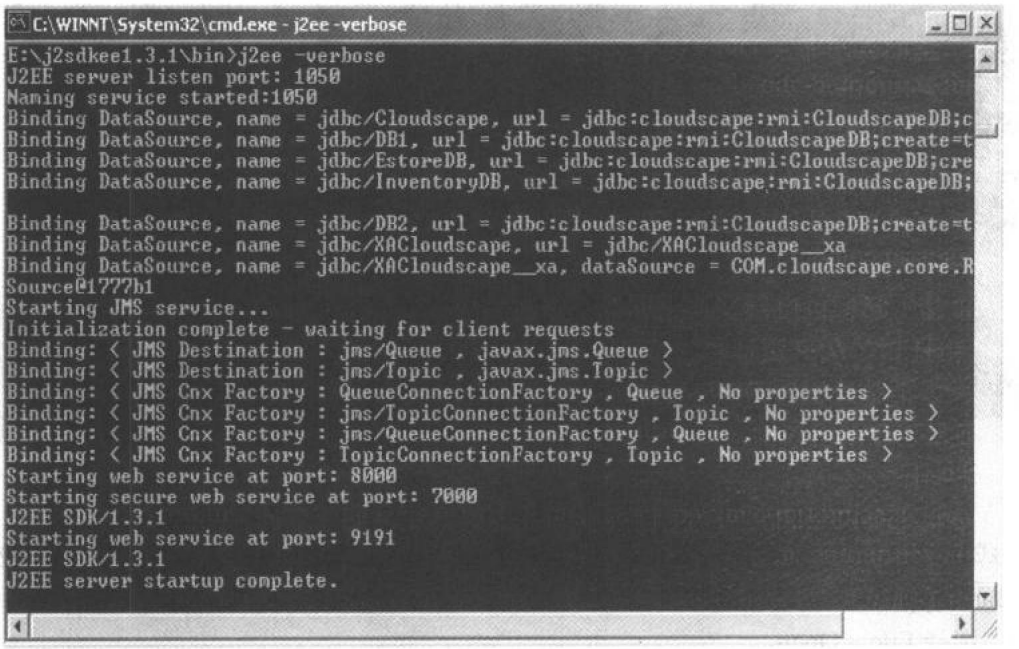


图 1.2　J2EE 应用服务器启动工作完成

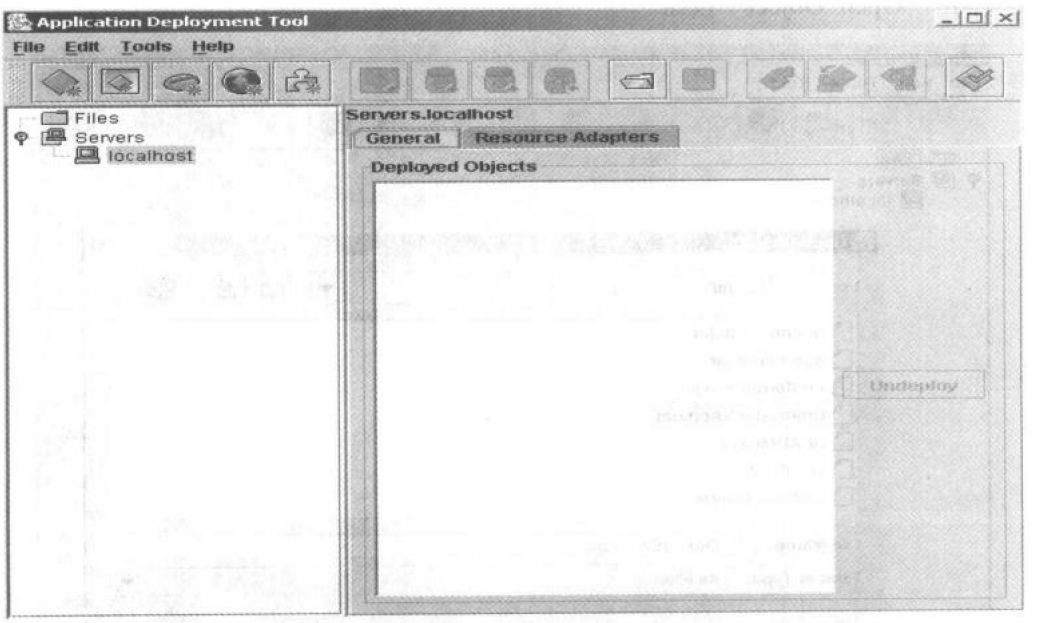


图 1.3　部署工具已启动并且正在执行

3. 在同一子目录下运行 cloudscape.bat 来启动 cloudscape 数据库服务器。按图 1.4 所示键入以下命令：

Cloudscape　- start

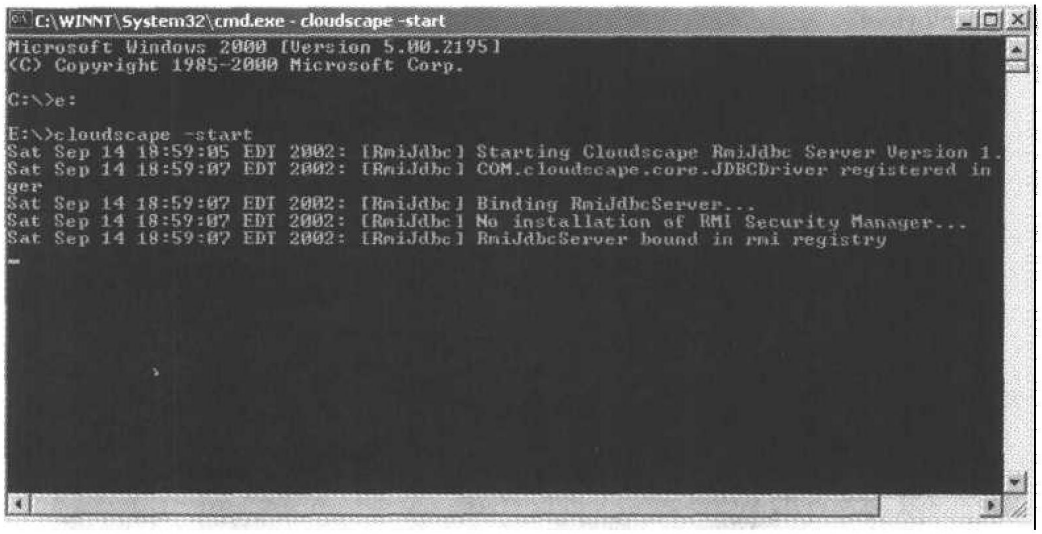


图 1.4　Cloudscape 数据库服务器正在执行

完成这 3 步后，接着我们就可以进行部署了。先在 deploytool 工具中打开企业应用文档 DukesBankApp.ear，其具体操作步骤如下：

- 选择 File→Open。
- 转到子目录 j2eetutorial\bank\jar 下选择 DukesBankApp.ear，如图 1.5 所示。
- 单击“Open Object”按钮。

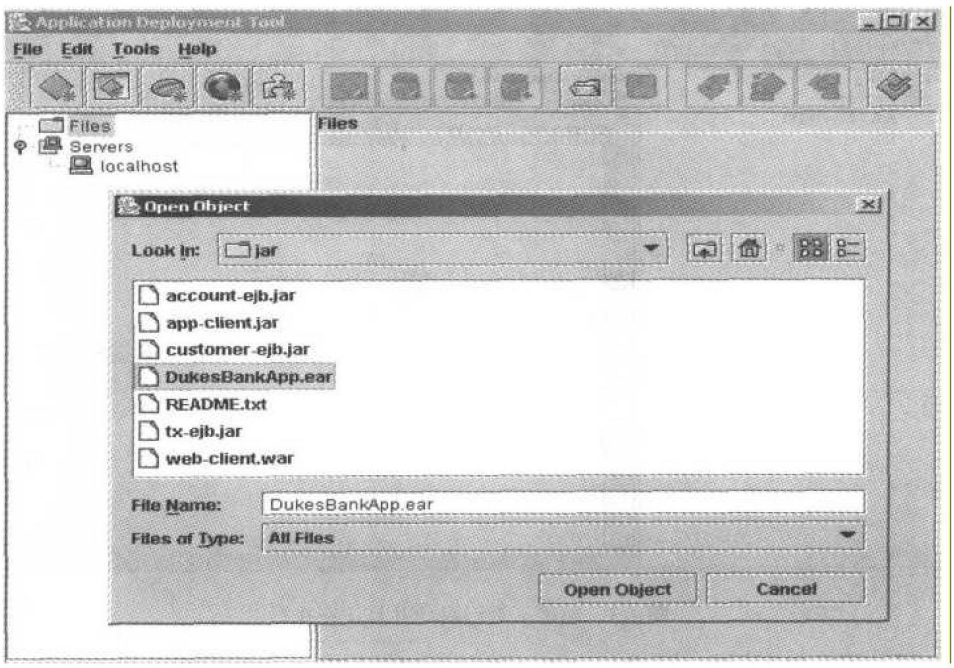


图 1.5　在部署工具中打开对象

现在我们可以看到如图 1.6 所示的窗口。这样，我们就可以在 deploytool 中开始部署该企业的应用文档，其具体操作步骤如下。

- 在 deploytool 左边窗口的组件树中，选中 DukesBankApp 应用。
- 选择 Tool→Deploy，如图 1.7 所示。

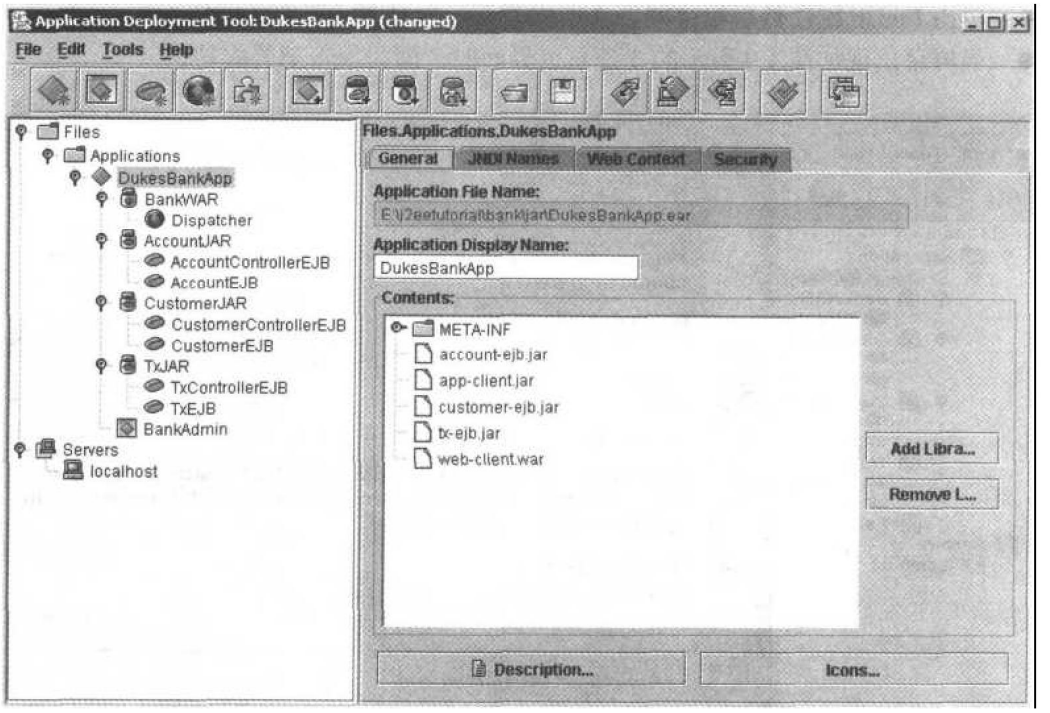


图 1.6　应用中的档案文件和组件

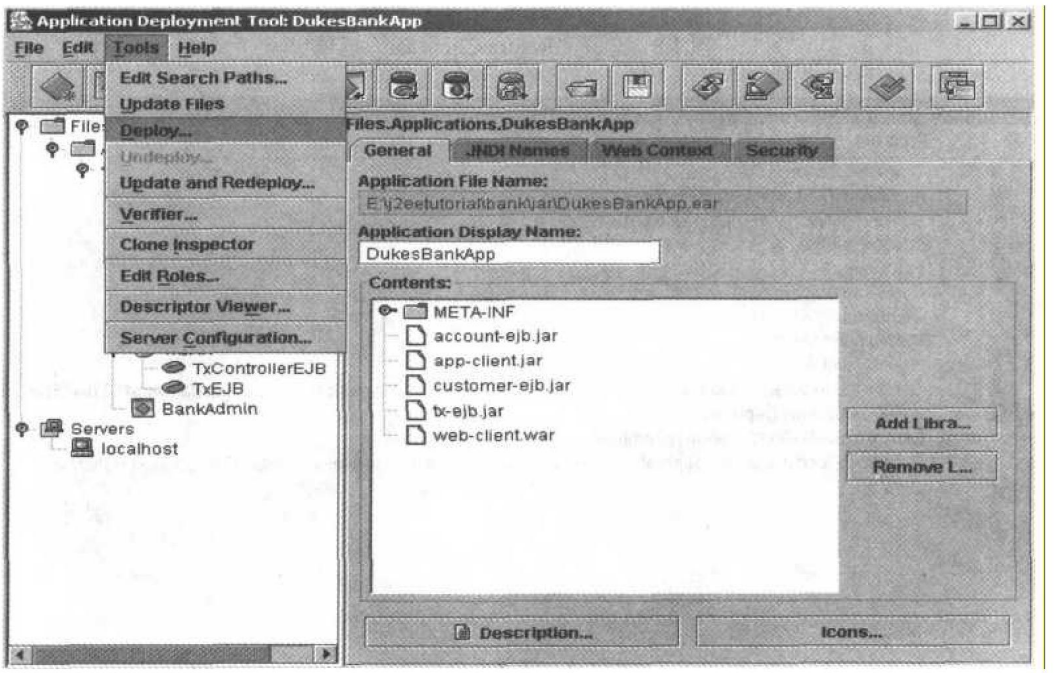


图 1.7　选择部署选项

- 选择标记为“Return Client Jar”的选择框，如图 1.8 所示。
- 客户程序默认的 jar 文件名为应用的名字加上 Client.jar，即 DukesBankAppClient.jar。
- 单击 Finish 键，然后等待部署进展屏幕显示自动完成所有的部署工作。
- 当屏幕出现如图 1.9 所示的 3 条彩色界面时，即为报告部署成功。

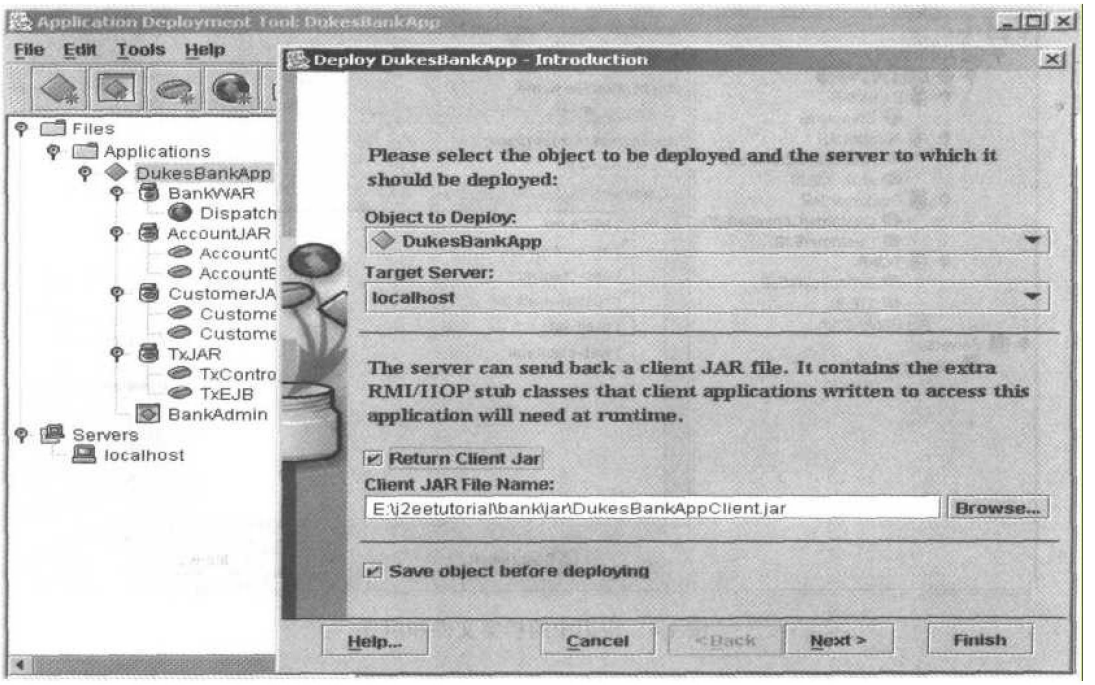


图 1.8　选择“Return Client Jar”选项

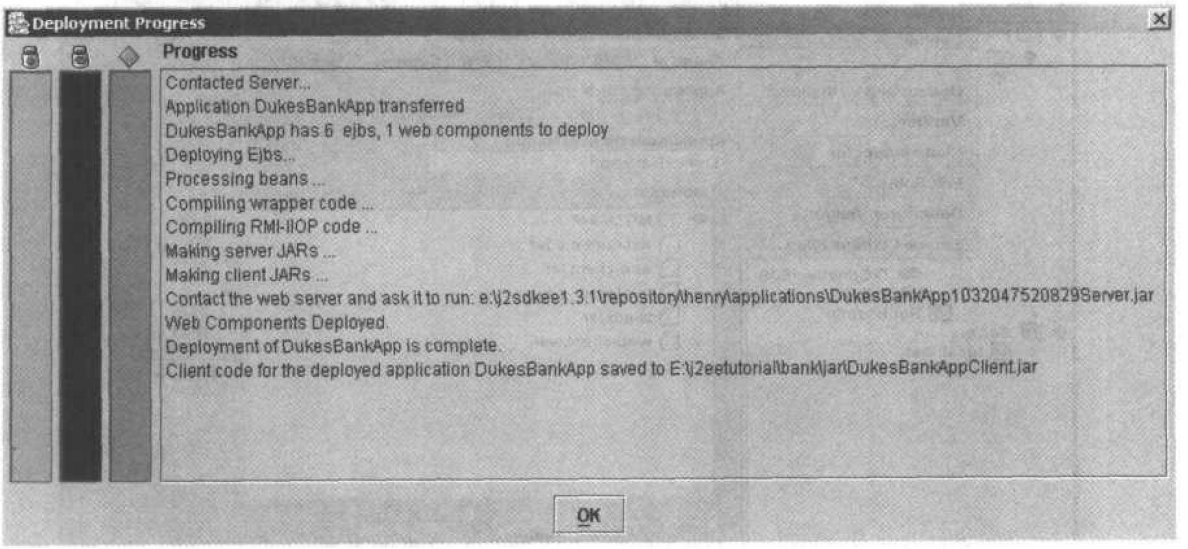


图 1.9　部署完全成功

## 1.4　Duke 全景浏览

上一节，我们已经设置好了 Duke 的运行环境，现在该是主角登场的时候了。俗话说，“一图值千字”。在我们走进 Duke 实例之前，先让 Duke 运行起来，使我们透过浏览 Duke 图形用户界面来获得一个直观清晰的全景。首先暂不理会诸多的概念，也不深入探究繁杂的细节，让我们简单地从用户角度来看看 Duke 提供的图形用户界面和所能实现的管理功能。

### 1.4.1　设置 Duke 的安全机制

为了安全地运行 J2EE 应用程序和 Web 客户程序，我们必须向缺省安全机制添加一些组和用户。为了创建客户组和管理者组，我们用 deploytool 将一个名为 200 的用户增加到客户组中，然后再将这个用户添加到管理者组。其具体操作步骤如下：

1. 选择 Tools（工具）菜单下的 Server Configuration（配置服务器）项。
2. 在这个树型列表中，选择 Users node（用户结点）项。
3. 确认在 Realm 组合框中选择的是 Default（缺省）项。
4. 单击 Add User（添加用户）项。
5. 单击 Edit Groups（编辑组）项。
6. 单击 Add（添加）按钮。
7. 输入“Customer（顾客名）”。
8. 单击 Add（添加）按钮。
9. 输入“Admin（管理者名）”。
10. 单击 OK 按钮。
11. 在 User Name（用户名）中输入“200”，在 Password（密码）中输入“j2ee”。
12. 从可用组列表中选择 Customer group（客户组）。
13. 单击 Add（添加）按钮。
14. 单击 Apply（应用）按钮。
15. 在 User Name（用户名）中输入“admin”，在 Password（密码）中输入“j2ee”。
16. 从可用组列表中选择 Admin group（管理者组）。
17. 单击 Add（添加）按钮。
18. 单击 OK（确定）按钮。

将这些组和用户加入安全机制后，就该给各组分配安全角色了。下面的步骤是把 BankAdmin 分配给 Admin 组，并把 BankCustomer 分配给 Customer 组。

1. 在 deploytool 中，选择 DukesBankApp 项。
2. 在 Security（安全）属性页，从角色名列表中选择 BankAdmin。
3. 单击 Add（添加）按钮。
4. 在用户组对话框中，从组名列表中选择 Admin 组。
5. 单击 OK 按钮。
6. 在 Security（安全）属性页，从角色名列表中选择 BankCustomer。
7. 单击 Add（添加）按钮。



此为试读，需要完整PDF请访问：www.ertongbook.com

8. 在用户组对话框中，从组名列表中选择 Customer 组。
9. 单击 OK 按钮。
10. 从主菜单中选择 File（文件）菜单中的 Save（保存）项。

图 1.10 是为一个组分配安全角色的安全属性页面。

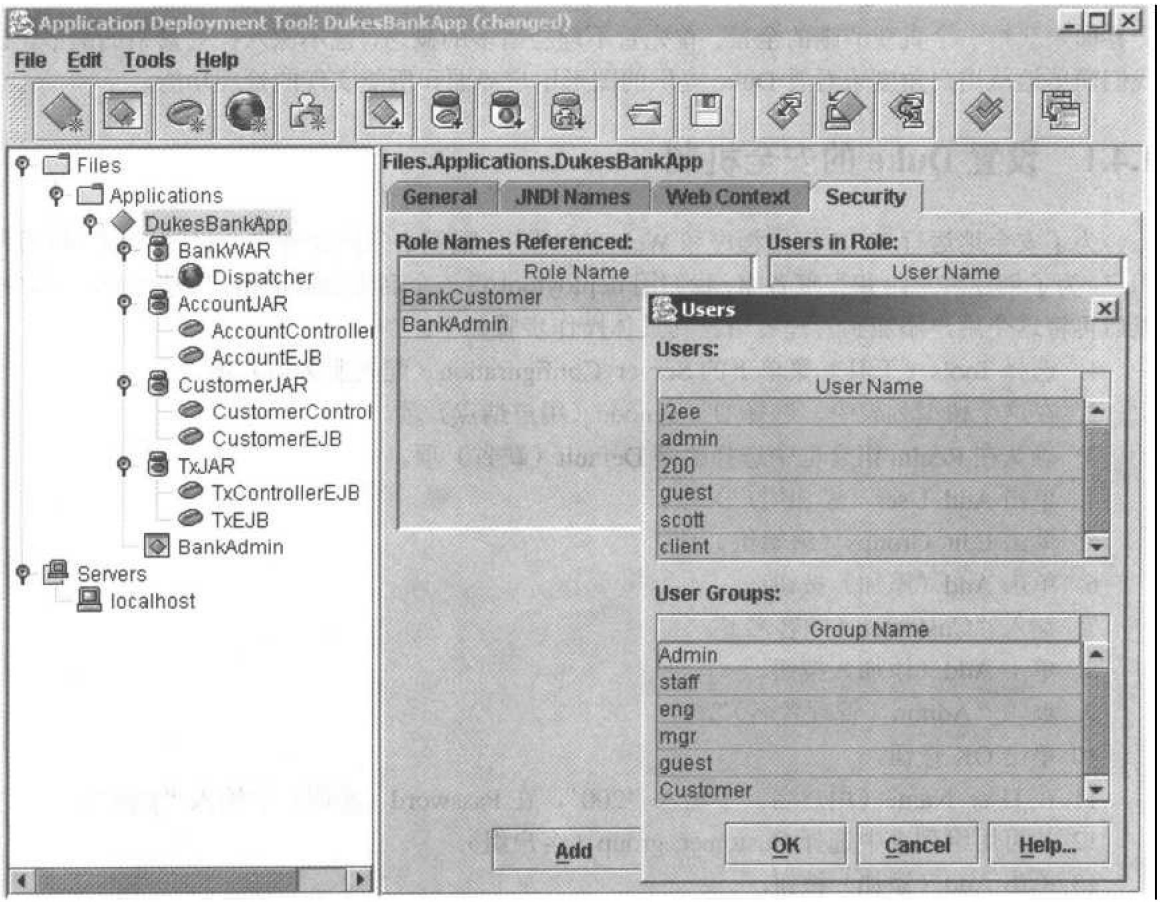


图 1.10　配置工具的安全属性页面

## 1.4.2　创建 Duke 数据库

现在运行 Duke 可以说是万事俱备，只欠东风了。我们要做的是提供数据给 Duke 银行，因此，必须创建一些数据表格并且将数据写入相应的表格中，这样企业 Bean 就可以从数据库中读出或向数据库写入它们需要的信息。有关 Duke 数据库的细节我们将在第 2 章中详细讨论。为了创建和装入数据库表格，可以在 Window 命令提示符窗口下进入 j2eetutorial\bank 目录，然后键入如下命令：

- ant db-create-table
- ant db-insert

其结果如图 1.11 所示。

```
db-create-table:
     [java] ij version 4.0 (c) 1997-2001 Informix Software, Inc.
     [java] WARNING 01J01: Database 'CloudscapeDB' not created, connection
     [java] CLOUDSCAPEDB* -     jdbc:cloudscape:CloudscapeDB;create=true
     [java] * = current connection
     [java] ij> -- create tables for online banking app
     [java] -- also seeds next_id tables w. initial values
     [java]
     [java] DROP TABLE customer_account_xref;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE tx;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE customer;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE account;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE next_tx_id;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE next_customer_id;
     [java] 0 rows inserted/updated/deleted
     [java] ij> DROP TABLE next_account_id;
     [java] 0 rows inserted/updated/deleted
     [java] ij> CREATE TABLE account
     [java]    (account_id VARCHAR(8)
     [java]        CONSTRAINT pk_account PRIMARY KEY,
     [java]     type VARCHAR(24),
     [java]     description VARCHAR(30),
     [java]     balance NUMERIC(10,2),
     [java]     credit_line NUMERIC(10,2),
     [java]     begin_balance NUMERIC(10,2),
     [java]     begin_balance_time_stamp TIMESTAMP);
     [java] 0 rows inserted/updated/deleted
     [java] ij> CREATE TABLE customer
     [java]    (customer_id VARCHAR(8)
     [java]        CONSTRAINT pk_customer PRIMARY KEY,
     [java]     last_name VARCHAR(30),
     [java]     first_name VARCHAR(30),
     [java]     middle_initial VARCHAR(1),
     [java]     street VARCHAR(40),
     [java]     city VARCHAR(40),
     [java]     state VARCHAR(2),
     [java]     zip VARCHAR(5),
     [java]     phone VARCHAR(16),
     [java]     email VARCHAR(30));
     [java] 0 rows inserted/updated/deleted
     [java] ij> CREATE TABLE tx
     [java]    (tx_id VARCHAR(8)
     [java]        CONSTRAINT pk_tx PRIMARY KEY,
     [java]     account_id VARCHAR(8),
     [java]     time_stamp TIMESTAMP,
     [java]     amount NUMERIC(10,2),
     [java]     balance NUMERIC(10,2),
     [java]     description VARCHAR(30));
     [java] 0 rows inserted/updated/deleted
     [java] ij> CREATE TABLE customer_account_xref
     [java]    (customer_id VARCHAR(8),
     [java]     account_id VARCHAR(8));
     [java] 0 rows inserted/updated/deleted
```

图 1.11　创建 Duke 数据库的表格

## 1.4.3　从 Web 客户浏览

在从 Web 客户端浏览 Duke 之前，我们首先应确认 J2EE 服务器处于运行状态。现在假定 J2EE 服务器和 Web 浏览器运行在同一机器上，打开 Web 浏览器，在 URL 地址栏中键入包含 Duke 银行应用程序的主页地址：

http://localhost:8000/bank/main

如果 J2EE 服务器运行在另一台机器上，则将 localhost 替换成那台机器的 IP 地址。Duke 银行应用程序将显示如图 1.12 所示的登录页面。在 Customer ID 域中键入“200”，作为顾客登录身份号，并在 Password 域中键入“j2ee”作为密码号，最后单击 Submit（提交）按钮。

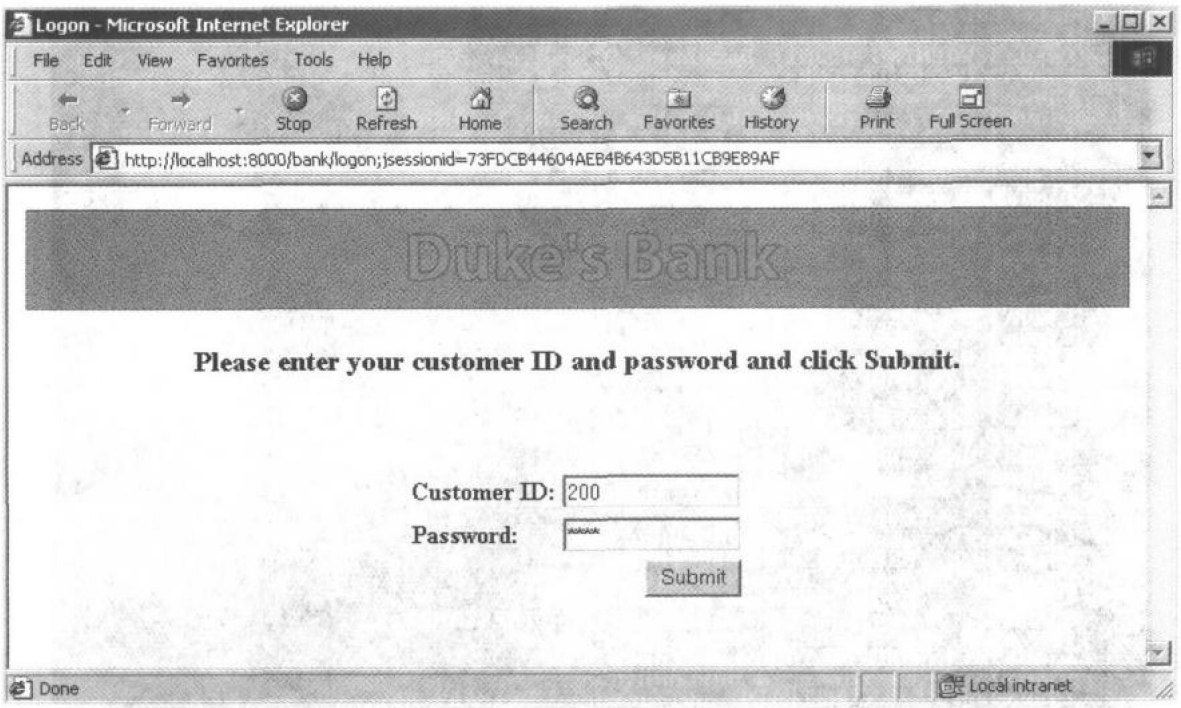


图 1.12　Duke 的 Web 客户程序登录页面

一旦登录成功，将显示如图 1.13 所示的欢迎页面。在欢迎页面的上方有 4 个可以选择的功能项：账户列表（Account List）、转账（Transfer Funds）、自动提款机（ATM）和退出。

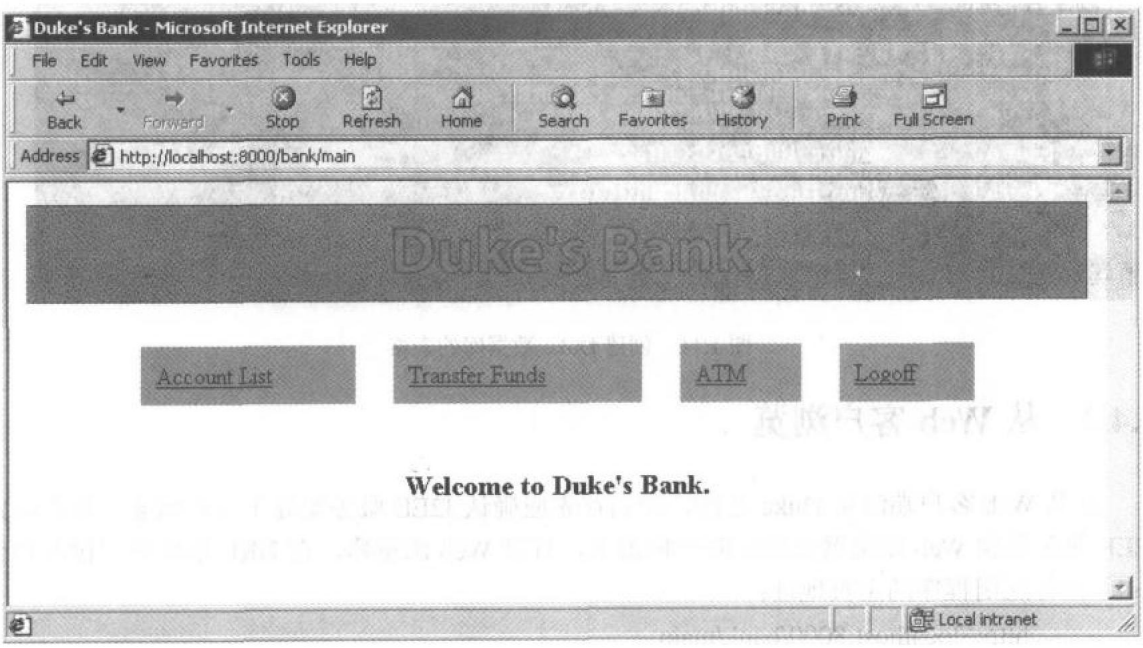


图 1.13　Duke 的欢迎页面

我们首先选择账户列表，将看到如图 1.14 所示的界面，上面列出了该客户的所有账户。每个账户都有其账号和收支状况，对于 Visa 账户还可显示其信用额度。

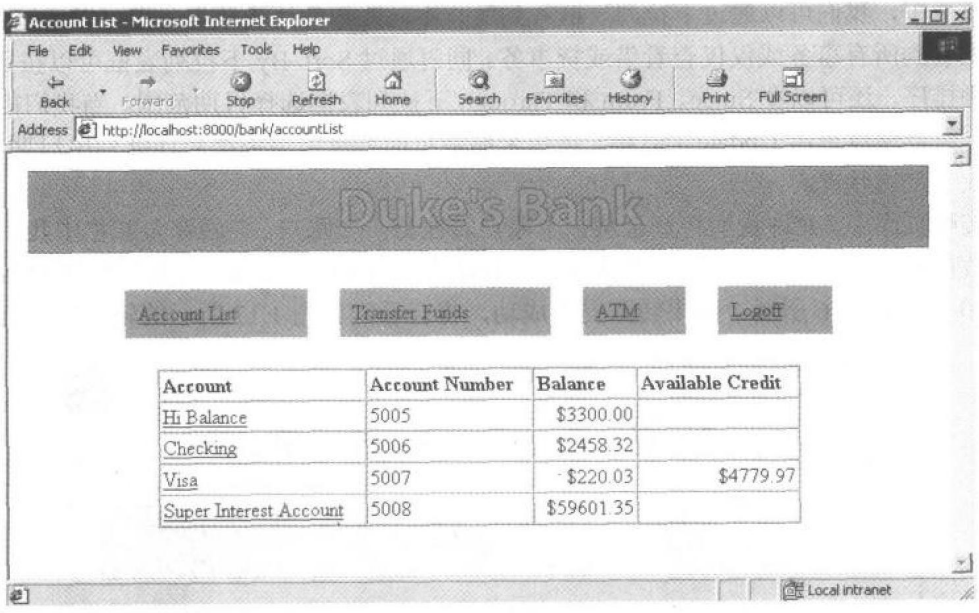


图 1.14　Duke 的账号列表页

我们可以通过选择账户链接，来查看账户的历史记录页，比如，单击 Checking 链接将显示如图 1.15 所示的 Checking 账户的历史记录页。

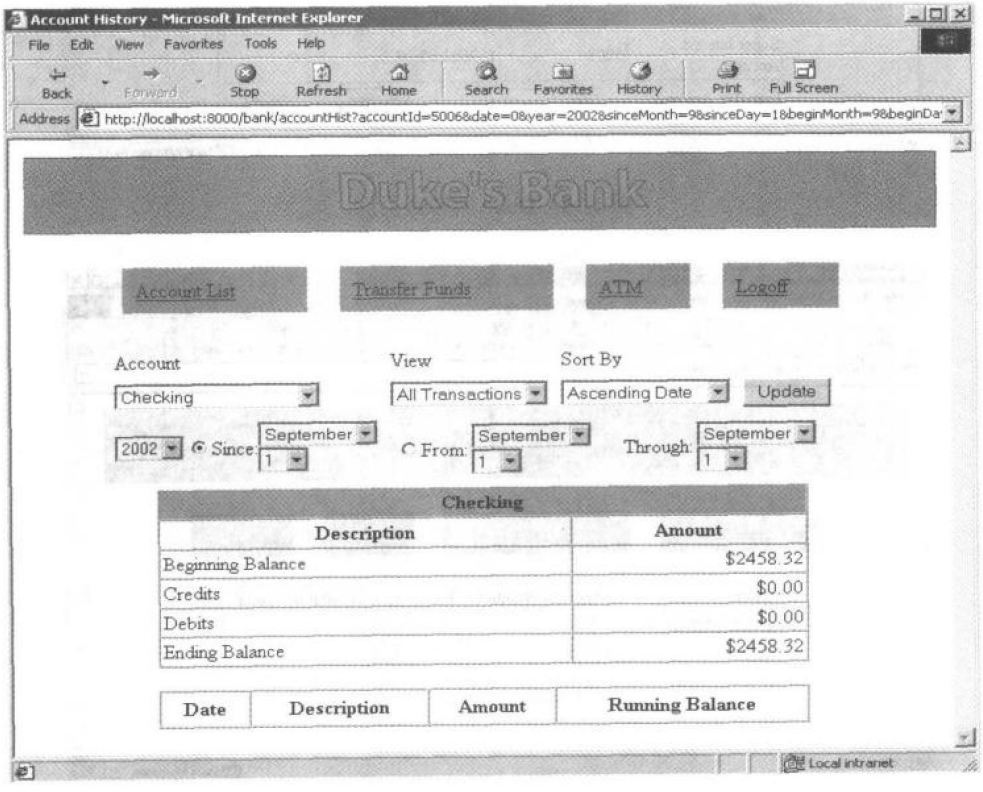


图 1.15　Duke 账号的历史记录页

在该页中，我们可以通过下拉列表框看到所有不同账户的历史数据，也可以通过 View 下拉列表框看到所有事务或仅仅查看借或贷事务。同时通过 Sort By 下拉列表框可以按日期、种类和金额排序，还可以在 Since、From 和 Through 下拉列表框选择日期范围。当我们需要改变要求时，可以通过单击 Update（更新）按钮来刷新页面。每笔事务都会详细列出日期、种类、金额和收支平衡状况。

在转账页面中，允许我们将资金从一个账户转往另一个账户。在转账金额栏中我们可以输入任意数量的金额。例如，我们可以从超级利息账户（Super Interest）向支票账户（Checking）转入$500，如图 1.16 所示。如果转账提交成功，将会看到如图 1.17 所示的转账确认页面。

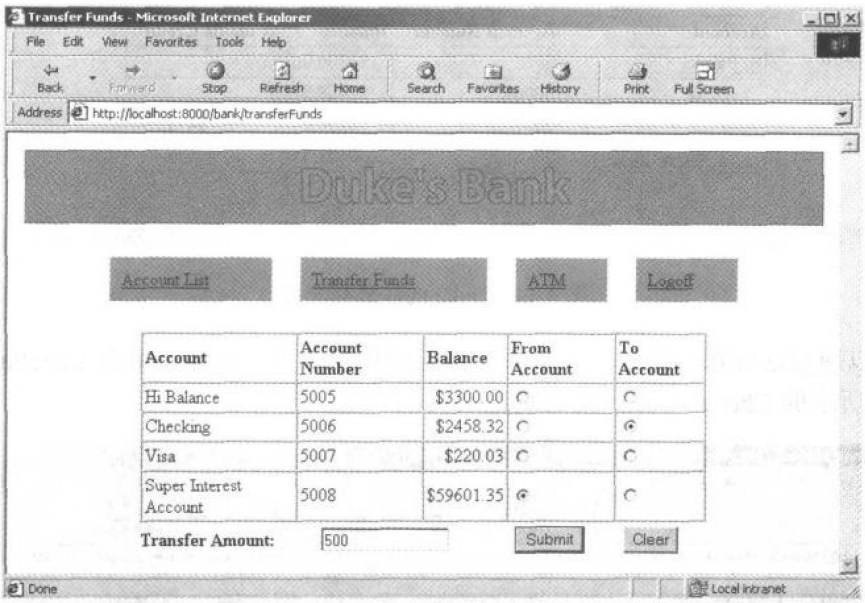


图 1.16　Duke 的转账页面

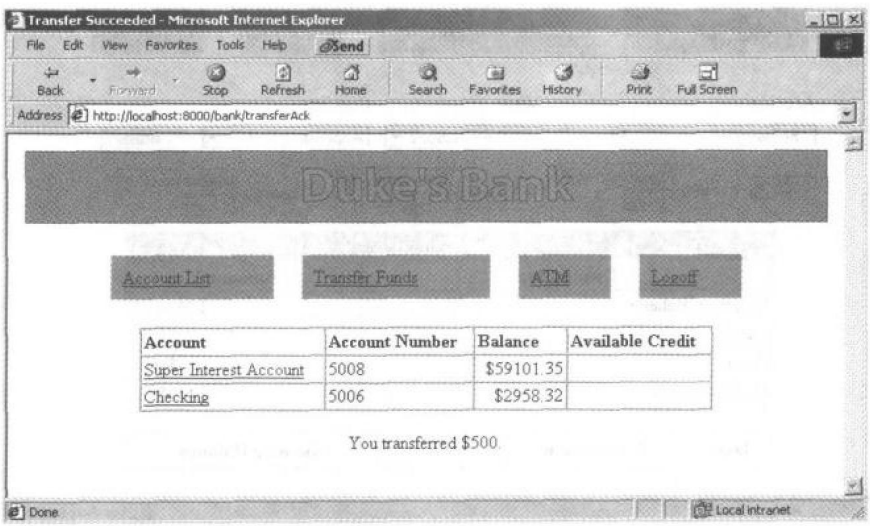


图 1.17　Duke 的转账确认页面

ATM 自动提款机页面有两个功能项，可以在该客户的任意一个账户提款或存款，如图 1.18 所示。从 Hi Balance 账户提出款项$200 的结果如图 1.19 所示。

退出页面只是简单地退出该应用程序，如果还想使用该程序则必须重新登录。

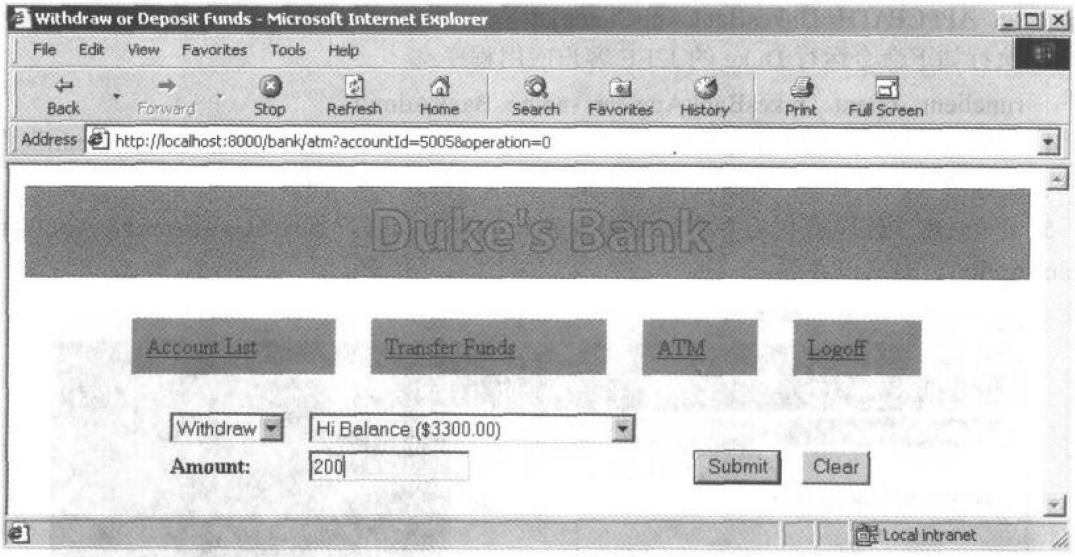


图 1.18　Duke 的 ATM（自动提款机）页面

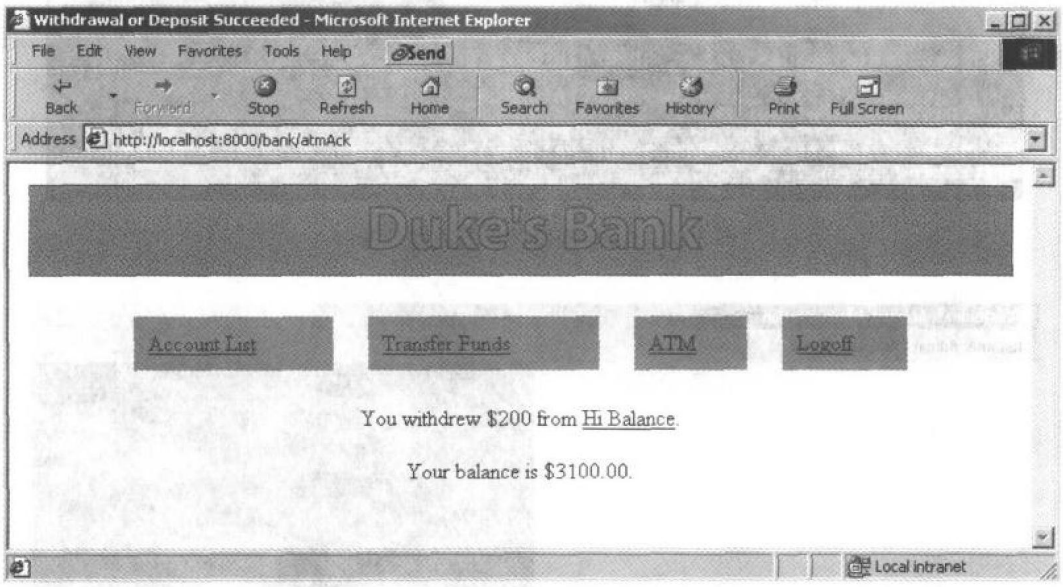


图 1.19　ATM（自动提款机）确认页面

## 1.4.4　从应用客户浏览

Duke 的 Web 客户端向用户提供了访问银行业务功能的接口，而 Duke 应用客户端则向用户提供了访问银行管理功能的接口。要从应用客户端浏览 Duke 银行，我们必须按如下方式运



此为试读，需要完整PDF请访问：www.ertongbook.com