

InstallShield 和安装软件制作技术

刘 艺



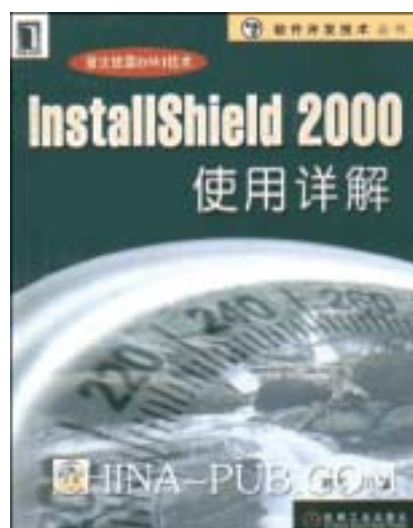
摘要：几乎所有的程序员在发布其软件产品时都面临过软件安装的问题。随着软件技术的日趋复杂以及软件产品对系统环境的依赖增加,如何使自己的软件产品能够一次性在用户的机器上安装成功并可以干净地卸载或自由增减功能一直是程序员烦恼的问题。

本文就安装软件制作技术以及该技术的最大供应商 InstallShield 公司的 InstallShield 工具进行了深入浅出的讨论。并重点介绍了 InstallShield 为适应微软 Windows Installer 标准而忍痛割爱推出的 ISWI。本文除了给出使用 ISWI 快速开发安装软件的应用实例外,还详细介绍了程序员最关心的如何制作国际版本安装软件和如何制作数据库应用程序安装软件这两种实用技术。

读者如果有兴趣深入了解 InstallShield, 请参阅本文作者主编的两本专著:

《InstallShield5 快速制作安装盘》(中国水利水电出版社 2000)

《InstallShield2000 使用详解》(机械工业出版社 2002)



关键词：软件安装 标准 安装机制 国际版本 数据库 对象 组件 功能部件
注册表 Windows Installer (WI) InstallShield Professional Version 6 (ISPro6)
InstallShield for Windows Installer (ISWI)

目 录

第 1 章	软件安装和安装软件制作.....	4
1.1	业界安装软件的工业标准是什么?	4
1.2	为什么选用 InstallShield 制作安装软件?	5
1.3	认识软件安装机制.....	5
第 2 章	Installshield 产品概述.....	11
2.1	传统和现实.....	11
2.2	ISPro6 的最新特点.....	12
2.3	ISWI 的最新特点.....	16
第 3 章	十五分钟快速搞定安装软件.....	22
3.1	启动工程向导.....	22
3.2	命名新工程.....	23
3.3	给出应用程序的有关信息.....	24
3.4	选择安装语言.....	25
3.5	创建功能部件.....	26
3.6	生成组件.....	27
3.7	关联组件和功能部件.....	28
3.8	把文件关联到组件.....	29
3.9	创建快捷键.....	31
3.10	配置注册表信息.....	32
3.11	定义用户界面.....	33
3.12	保存工程, 创建发布媒介, 测试安装程序.....	34
3.13	小结.....	35
第 4 章	国际版本安装软件制作技术.....	36
4.1	如何制作国际版本安装软件.....	36
4.2	国际版本安装软件制作实例.....	41
第 5 章	数据库应用程序安装软件制作技术.....	49
5.1	数据库应用程序及其安装要点.....	49
5.2	基于 BDE 的 Oracle 数据库应用程序安装软件制作.....	51
5.3	基于 ODBC 的 Access 数据库应用程序安装软件制作.....	56

第1章 软件安装和安装软件制作

计算机应用软件的发展可以大致分为 DOS 时代、Windows 时代和 Internet 时代，这三个时代的软件分别有着自己独特的特征，如：DOS 时代的单机命令行操作、Windows 时代的多用户图形界面和 Internet 时代跨平台、跨地区的网络应用。在这三个不同的时代，不仅软件自身发展经历了前所未有的巨大变化，而且用户安装软件的方式也在发生着深刻的变化。DOS 时代，我们只需将应用软件拷贝到计算机中即可通过命令行运行。但我们很难想象在 Windows 时代再继续使用 Xcopy 之类的复制命令来为用户安装软件。Windows 应用程序的许多诸如：注册组件、修改注册表、配置 ODBC 等复杂操作已经无法再交给用户自己完成。虽然在 Internet 时代，基于 Web Browser/Web Server 的应用盛行，让人们一度相信浏览器的使用免除了客户端软件安装的麻烦，但服务器端的软件安装却日趋复杂。在许多领域，对安装软件的要求甚至更为苛刻，比如要求跨平台跨操作系统安装、按需安装、即时激活、组件配置和网络部署等。于是寻找和使用功能强大的安装软件开发制作工具就成为每一个开发人员和软件商迫切关心的问题。那么怎样才能制作出更复杂、更创意的一流商业安装软件呢？

1.1 业界安装软件的工业标准是什么？

软件安装过程是用户对一个软件至关重要的第一印象，它可以戏剧化地影响到他们使用该软件的心情以及他们对该软件开发机构的看法。为得到一个好的第一印象，安装软件必须做到：

- 简单易用。
- 无操作问题和出错信息。
- 较好地处理应用程序中使用的第三方部件，如：ODBC。
- 一次性安装成功。
- 能干净地卸载清除所安装的软件。
- 允许用户根据不同需要灵活选择安装部件。

以上几条标准是业界对安装软件的基本要求，也是用户在安装使用软件时最关心的问题。

微软作为业界的龙头老大，为了进一步加强其在软件行业的垄断地位，在 Windows 2000 及其后续版本中内建了 Windows Installer 服务，并推出了符合 Windows 认证标志的安装软件标准。凡是符合该认证标志的软件产品都可以享用 Windows 2000 和 Windows 98 / me 的最新功能。如果要申请该认证标志，应用程序必须使用 Windows Installer 来安装和卸载。

以下列出的符合 Windows 认证标志安装软件的特征是从“微软 Windows 2000 应用程序规格说明：桌面应用程序”（“Application Specification for Microsoft Windows 2000: For Desktop Applications”）技术文档中提炼出来的与程序安装有关的要求：

- 必须将应用程序的缺省安装目标文件夹设定为\Program Files（该名称可能会因不同的语言文字及驱动器而不同）
- 必须以 Windows Installer 安装包来发布安装程序，并校验包结构的正确性。
- COM 服务器必须是一个组件的单独文件，并且是键文件（key file）。
- 安装程序必须能在完全图形化的用户界面下正确运行。即能使用基本的用户界面，也可进行无干涉安装。

- 如果用 CD-ROM 发布安装程序，则必须在该 CD-ROM 首次插入时支持自动运行安装程序的功能。
- 支持安装程序卸载功能，能出现在 Windows 控制面板的添加 / 删除程序中。
- 实现用于广告宣传的安装。
- 必须在注册表的 HKEY_CLASSES_ROOT 键下注册本地数据文件类型。
- 安装程序必须能够为应用程序创建合法的快捷方式。
- 安装程序必须能够检查操作系统的正确版本。
- 安装程序必须能够防止核心组件文件被旧版本覆盖。实际上，微软建议不要直接把任何东西安装到 Windows 的 system 文件夹中。

虽然微软的标准只是为微软的 Windows 平台服务的，但了解这一标准对于我们制作符合业界规范的安装软件有着重要的意义。

1.2 为什么选用 InstallShield 制作安装软件？

InstallShield 软件公司是美国一家专业从事安装软件开发工具生产的著名软件公司。该公司由 Viresh Bhatia 和 Rick Harold 于 1987 年创建。公司自创建之初就以独立软件供应商的角色致力于高可靠性软件开发工具的研发，并在软件及数字媒体打包和发布领域有突出的贡献，其软件安装标准和产品被各大商业软件供应商广泛使用。自 Windows 软件开发之初，InstallShield 软件公司就已经专注于通用安装软件制作工具的开发并解决 Windows 操作系统发展和变革所带来的挑战，以满足开发人员及最终用户的需要。其主打产品 InstallShield 为满足各种 Windows 平台和不同的安装需求提供完整的软件安装解决方案，并且已经形成人们所熟悉的一套软件安装的工业标准。InstallShield 的产品自 2000 版本开始还增加了对 Windows Installer 服务，可以开发完全符合 Windows 认证标志要求的安装软件。目前全球绝大多数主流商业软件都使用了 InstallShield 产品来开发安装软件，特别是该产品对国际化安装软件的支持使得制作本地语言和国际版本的安装程序不再是一个难题，特别是对东方国家计算机中使用的双字节字（如：中文、日文、韩文等）支持，为我们制作全中文的安装软件提供了优势。国内许多著名的商业软件（如金山公司的 WPS）都是用 InstallShield 制作的。

虽然许多应用程序开发系统都提供了专用的安装软件制作工具，如：Visual Basic 和 Power Builder 等。使用这些专用安装工具的唯一好处是可以比较容易地制作由本系统开发出的应用程序的安装软件，因为它们大都会自动找到并配置相关的动态链接库，操作也比较简单。遗憾的是，用它们制作出的安装软件功能单一，界面单调，并且带有原开发系统产商的印记，容易让别人一眼就看出你的应用程序是用什么工具开发的。这些都是专用安装软件制作工具与通用安装软件制作工具的不同之处。

随着 InstallShield 在安装软件制作领域的成功，许多公司纷纷采用 InstallShield 的技术为自己的开发工具提供安装软件制作支持，于是出现了许多 InstallShield 专用版本，如用于 Delphi 的 InstallShield Express For Borland Delphi，用于 VC 的 InstallShield Express For VC++ 等。

1.3 认识软件安装机制

1.3.1 InstallShield 的安装机制

InstallShield 软件安装工作的实质是把需要安装的文件传输到指定的系统中。应用文件自然会因自身的相似特点而归类，如：系统动态链接库（DLL）、可执行文件、帮助文件等。

根据文件相似的特点，我们可以把这些文件分成不同的文件组。

通常文件组由特点相似的文件组成，这些特点可以来自文件的不同目的和功能、是否可以共享、会不会存在加锁问题，是否压缩、使用何种操作系统或支撑平台、使用何种语言。作为安装软件的制作人，我们必须充分考虑安装文件的不同特点，并合理设计文件组。

划分好的不同文件组可以组织成独立的安装模块，这些安装模块我们称之为部件和子部件。部件和子部件为安装文件和相关软件打包提供了极大的灵活性。部件和子部件在以用户的眼光来选择安装时显得格外有用。

安装部件由不同的文件组组成，它作为独立的安装模块可以供用户进行选择性安装，以适合不同的使用需要。

如果你的安装软件是提供一套产品，就可以把这套产品中的每一个产品作为一个部件，供用户选择安装。比如，微软的 Office 套件，就把 Word、Excel、Powerpoint 等产品作为独立的安装部件由用户选择。每一部件又可以包含不同的子部件，这样可以提供用户更多的选择项。InstallShield 可以使你更自由的设计部件和子部件，以满足不同的安装配置需要。

为了减少用户自行选择安装部件的麻烦，我们可以向用户建议不同的安装方案，供用户选择。比如，有的便携机用户为节约硬盘空间希望按最小化方案进行安装，而有的用户则喜欢安装所有的图库和详细的帮助。于是根据不同类型用户的不同需要，设计具体的方案来组织安装部件，使之成为一种可选择的安装类型就显得特别重要。

组织安装部件时，可以把不同的安装部件按照实际的用户类型划分为几种安装类型，用以提供给用户更方便的安装选择，使得用户可以根据自己使用软件的实际需要选择安装类型，以节约时间和磁盘空间。

通过精心设计的不同安装类型，可以避免用户对各种复杂的安装要求和系统运行环境进行令人困惑的选择和配置。例如，你可以创建“服务器”、“客户机/管理员模式”和“客户机/一般用户模式”等安装类型，也可以创建“最小安装”、“典型安装”和“完全安装”等安装类型。这样用户就可以根据不同的安装对象，选定安装类型，而不必为那些部件安装在服务器，那些部件安装在客户机，以及那些用户选择那些部件而进行艰难、烦琐的选择。

InstallShield 可以让你把安装使用的文件、文件组、部件和安装类型作为模块自由设计。在制作安装软件的过程中，你必须先把独立的安装需要的文件分配到文件组中，然后把文件组连接到定义好的安装部件或子部件中，最后关联这些部件到不同的安装类型。这样，运行安装程序的时候，用户就可以选择不同的安装类型和想要安装的各个部件及子部件了。

安装中使用的文件、文件组、部件和安装类型作为模块，其相互关系如图 1-1 所示：

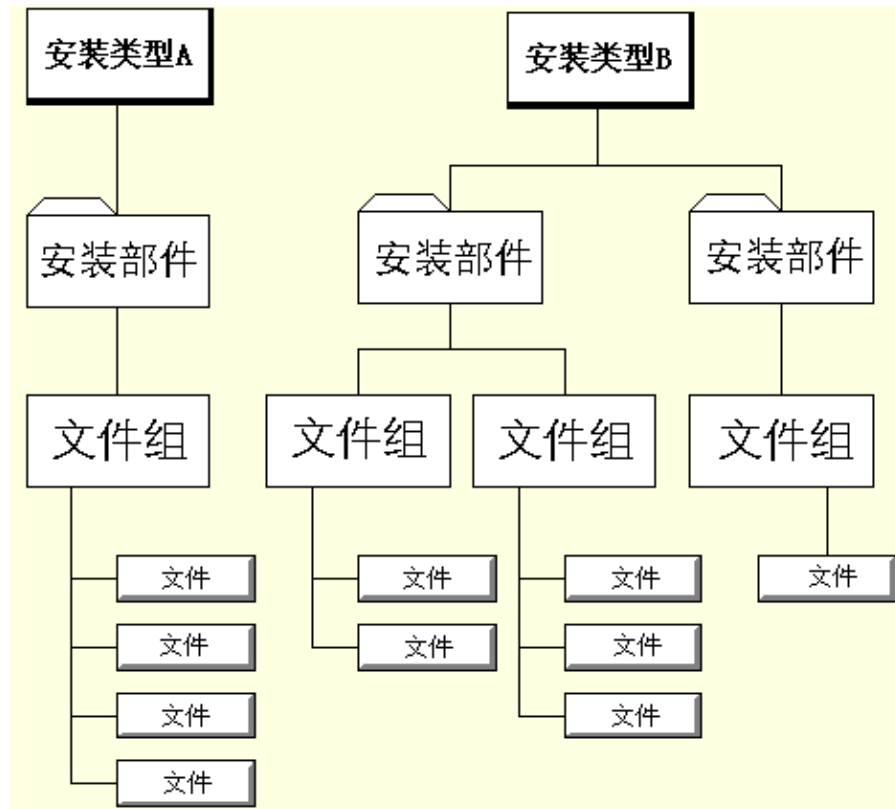


图 1-1 安装中使用的文件、文件组、部件和安装类型及相互关系

在图 1-1 中我们看到安装类型可简可繁，比如：安装类型 A 可能是一个最小安装类型，它只包含了一个安装部件，而安装类型 B 可能是一个完全安装类型，它只包含了两个安装部件，同时这两个安装部件又含有不同的文件组。由此可见安装使用的文件、文件组、部件和安装类型都可以灵活定义，然后通过互相间的关联设置来实现最终目的。

1.3.2 微软 Windows Installer 的安装机制

微软的 Windows Installer 是围绕着组件和功能部件的概念来组织安装的。

功能部件是用户可以选择进行独立安装的应用程序所有功能中的一部分。组件则是被安装的应用程序或产品的一部分。当安装程序从用户电脑安装或卸载一个组件时，总将它看成是一个有联系的部分。组件对于用户是不可见的，当用户选择安装某个功能部件，安装程序将决定选择哪一个组件来实现这个功能部件。

安装包的开发人员需要决定如何把他们的应用程序划分成功能部件和组件。功能部件的选择主要决定于用户对应用程序某些功能的偏爱。鉴于组件通常会被程序或产品共享 所以建议开发人员最好遵循一套标准的方法来选择组件。

(1) 组件

组件是被安装的程序或产品的一部分。组件的例子包括单一的文件、一组相关的文件、COM 对象、注册参数、注册键、快捷方式、资源、一个目录下的库以及诸如 MFC 或 DAO 那样的共享代码。

Installer 服务安装或卸载一个组件时，把它看成一个单独的联系部分。它通过在组件表

中组件 ID 这一列标明的组件 ID GUID 来跟踪每一个组件。两个共享同一个组件 ID 的组件被看成是同一个组件的不同实例，而不管它们的实际内容。对于任何组件，仅有一个实例会被安装到用户的电脑上。因此许多功能部件或应用程序可以共享一些组件。

因为组件通常是共享的，安装包的开发人员编写某个功能部件的组件或应用程序时，必须遵循严格的规定。这对于 Windows Installer 引用计数机制的正确操作是非常重要的。

这些规定主要是：每个组件必须被分别存储在单个文件夹中；文件、注册表条目、快捷方式或其他资源不能成为多个组件的成员。

(2) 功能部件

功能部件是一个用户可以了解并决定独立安装的应用程序全部功能中的一部分。举个例子，功能部件可以是一个拼写检查程序、一个辞典或者是一组剪贴板。功能部件可以存在父级和子级的继承关系，也就是说如果一个子功能部件被安装了，其相应的父功能部件也就自动被安装了。

(3) 安装过程

Windows Installer 安装过程分为两个阶段：获取和执行。如果一个安装没有成功，就会发生一个回退的过程。

- 获取——在获取阶段的一开始，应用程序或用户通知 Installer 安装某个功能部件或者应用程序。Installer 就顺着安装数据库中的队列表里所标明的动作一步步执行。这些动作访问安装数据库并生成了一个脚本，该脚本就是完成安装步骤的过程。
- 执行——在执行阶段里，Installer 把信息传送给具有相应权限的过程处理并运行脚本。
- 回退——如果安装不成功，Installer 将恢复计算机的初始状态。当 Installer 运行安装脚本时，它能自动生成了一个回退脚本。除了生成回退脚本，Installer 还保存了安装过程中它所删除的每个文件的副本。这些文件被保存在隐匿的系统目录中。一旦安装完成，回退脚本和这些保存的文件则被全部删除掉。

开发人员能通过编辑安装包和在应用程序代码中使用 Installer 函数把 Installer 组件管理能力结合到自己的应用程序中。下图 1-2 显示了应用程序对功能部件的请求安装过程。

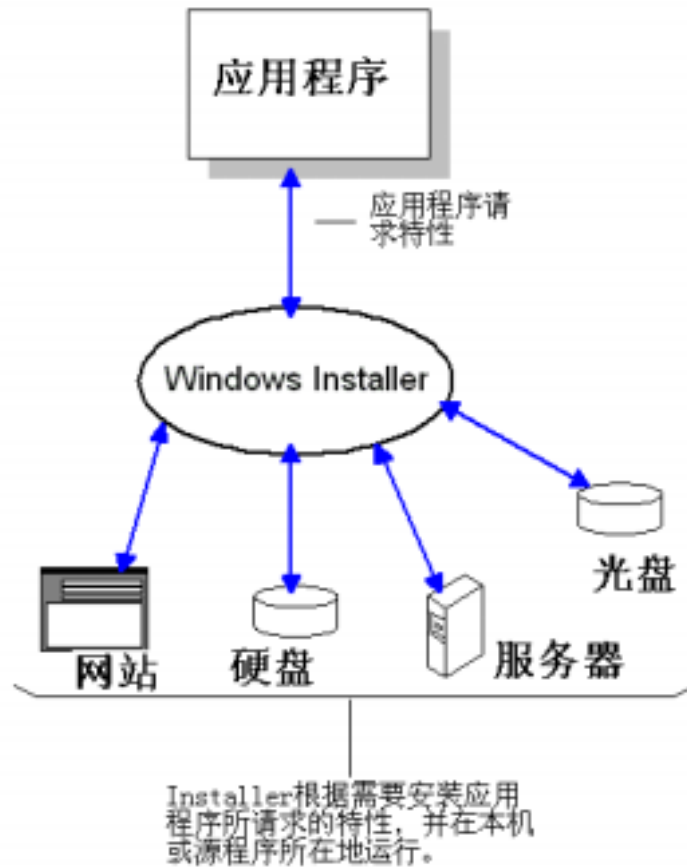


图 1-2 应用程序对功能部件的请求安装过程



图 1-3 ISWI 设计中的“安装文件—组件—功能部件”结构

在 InstallShield2000 及其后续版本中，新的 InstallShield for Windows Installer (ISWI) 使用了“安装文件—组件—功能部件”的结构取代了传统的“文件组—安装部件—安装类型”

结构，如图 1-3 所示。该结构适应了微软的 Windows Installer 服务的需要，其包容性更加广泛。其中安装文件包含了文件、注册表数据、快捷方式、高级设置（与 COM/COM+、ODBC、NT 服务等有关）；组件类似于原来的安装部件，但可以通过融合模块增加对第三方组件（技术）的支持；功能部件虽然也类似于原来的安装类型，但更突出了要安装的程序功能，该用户更多的安装选择。其中包括了传统安装中没有的应用程序广告功能（亦称即时激活功能），即可以不安装程序而只将应用程序的图标放置在桌面上，等到用户需要时，双击图标即可安装。图 1-4 中显示了一个用 ISWI 开发的安装软件在安装过程中对功能部件的选择情景。

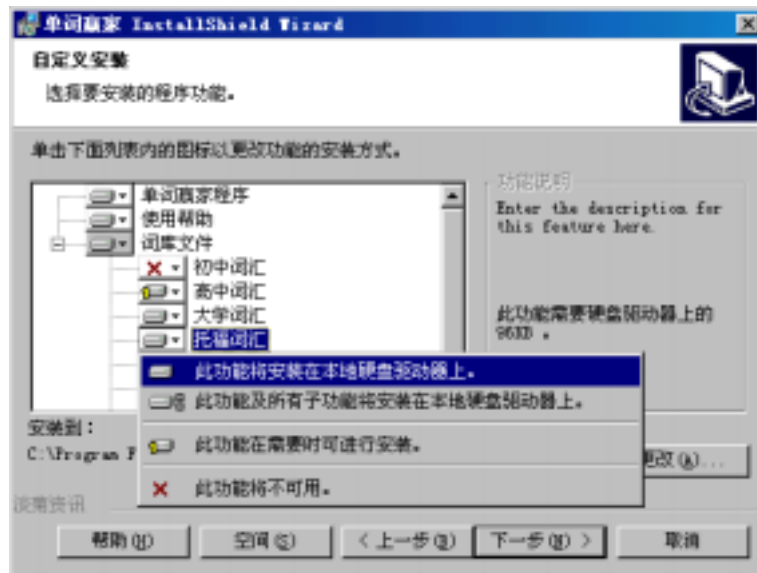


图 1-4 安装过程中对功能部件的选择情景

第2章 Installshield 产品概述

InstallShield 软件公司的主要产品为 InstallShield 和 DemoShield, 前者用于安装软件的开发, 后者则用于软件产品的演示。InstallShield 软件公司将这两种产品结合起来, 作为软件安装和演示的解决方案提供给用户, 使用户能够使用这两种产品实现商业软件无缝安装和对自己的产品进行演示宣传。

2.1 传统和现实

InstallShield 软件公司以开发 InstallShield 安装软件制作工具起家, 其特有的 InstallScript 脚本语言为安装软件的制作提供了广阔的空间。InstallScript 类似于 C 语言, 并有自己的编译器。InstallShield 在不断的发展中除了将 InstallScript 的功能不断加强外, 还增加和完善了大量的向导和函数, 以提高用户的安装软件制作效率。特别是为了支持像 ODBC、BDE、Access、RDO、DirectX 那样的第三方技术, InstallShield 从 InstallShield5.x 的模板到 InstallShield6.x 的对象, 一直在作用心良苦的改进。可以说 InstallShield 公司一直以其所拥有的标准和技术在同类公司中遥遥领先。

然而, InstallShield 公司仍然面临着来自微软的新挑战。微软自 Windows2000 操作系统开始引入了 Windows Installer (WI) 服务, 使得用户可以利用微软提供的 API 函数构建符合微软标准的安装软件, 并充分享用微软 Windows Installer 服务提供的诸多好处。这意味着微软对 Windows 平台上安装软件制作标准的新垄断。

为了跟上安装软件制作技术的自然进化, 满足用户的需求并迎接新的挑战, 1999 年 9 月, InstallShield 软件公司宣布 InstallShield2000 专业版 (简称 ISPro2K) 面世, 这是该公司首次将其软件安装开发工具划分采用自身传统标准的 InstallShield Professional 和采用微软 Windows Installer 标准的 InstallShield for Windows Installer 两个不同版本。

为了满足开发人员的高级安装要求为目标, ISPro2K 由两部分组成:

- InstallShield Professional Version 6——简称 ISPro6, 是对 InstallShield 5.x 的直接升级。
- InstallShield for Windows Installer——简称 ISWI, 它支持新的 Microsoft Windows Installer (WI) 服务。该服务是为 Windows2000 设计的一个关键部件, 通过向开发人员提供创建更智能化、更灵活、更易管理的安装软件的系统服务来实现总成本 (TCO) 的降低。

虽然 ISPro6 有许多突破性的特性, 但它还只是 InstallShield Professional 5.x 的升级版本。用户只能使用安装脚本语言创建安装程序包, InstallScript 安装脚本语言是一种强大的智能安装语言, 它简单易用, 菜单驱动的开发环境加上向导的帮助, 可以自动完成安装软件制作的许多步骤。使用 ISPro6 可以在一个集中的地方创建、组织、编译和测试安装程序包。

ISPro6 为了将其功能增加到 Windows 95/98 和 NT4 的系统平台中将创建自己的安装引擎。该引擎包括 InstallScript 脚本语言及其编译器, 以及和安装软件一起打包的运行时刻版本。当安装用户加载 InstallShield Professional 创建的 setup.exe 文件时, 实际上是 InstallShield 引擎在运行安装软件。

与 ISPro6 不同, 使用 InstallShield for Windows Installer 必须要有 Windows Installer 服务支持, Windows Installer 服务是 Windows 2000 及其后续版本的一个要素。如果操作系统使

用的是 NT4 或 Windows 95/98, Windows Installer 服务可以自动和安装程序一并被安装。

Windows Installer 包含了以下三大部分:

- 为软件开发者准备的一系列规则和 API 接口。
- Windows Installer 服务通过 API 处理命令, 这些服务程序包括一个 Client Install 和一个 Install Service, 它们可以作为 Windows NT 服务程序以提升的管理权限运行(这样的好处是需要管理权限的安装程序可以由一个标准用户来运行)。
- 新的文件格式定义的数据库包含了所有由 Windows Installer 服务处理的文件和设置。

有了 Windows Installer 服务, 操作系统可以跟踪所有安装的应用程序和为多个应用程序共享的管理组件。新的服务使安装过程更为轻松, 它使得用户对应用程序管理器有了更全面的了解。

微软认为增加 Windows Installer 服务的目的是为了让使用其操作系统的计算机用户减少 TCO。Windows Installer 服务的加入是 Windows 操作系统体系的一个巨大进步。然而, 这个新的服务程序并不是安装程序的开发环境, 开发者仍然需要一个像 ISWI 这样的环境以便:

- 不直接接触复杂的 Windows Installer 文件结构。
- 减少学习和编程花费的时间。
- 为安装程序生成没有错误的, 符合工业标准的界面。

Windows Installer 文件格式十分复杂, 想要了解其中的细节和手工处理数据库需要大量的时间。ISWI 有自动的向导和功能强大的特性, 例如: 全屏对话框编辑器、动态文件链接以及 SKU 管理器, 来生成基于 Windows Installer 的安装软件。使用 ISWI, 用户可以很容易地生成符合低 TCO 的 Windows Installer 功能部件, 并编译出符合“Windows 2000 Application Specification”要求的安装程序包。

关于 ISPro6 和 ISWI 的不同特点我们将在下面进一步讨论。

2.2 ISPro6 的最新特点

使用 ISPro6 可以制作出用户所熟悉的复杂的工业级安装软件, 以体现工作的一贯性。ISPro6 继承了 InstallShield 软件公司自最初研发 Windows 下安装工具的丰富经验。ISPro6 预测你的安装需求, 使你从安装结构的底层复杂性中隔离出来并帮助你避免出现问题。

ISPro6 还包含了许多突破性的特点, 增加了 Windows 环境下安装软件的易用性和高效性。以下是 ISPro6 的主要新增特点。

2.2.1 部件的安装和卸载

部件的安装和卸载是安装程序改进的一个关键特征。它使得开发人员可以让最终用户对软件的安装和卸载进行更多的控制。目前, 多数应用程序仅提供“典型安装”、“简洁安装”和“自定义安装”, 即使“自定义安装”对安装内容的控制也很有限。而卸载程序时, 也严格限制为卸载全部的应用程序。如果想创建一个超越以上限制的安装程序则需要进行大量的编程。

如图 2-1 所示, 部件的安装和卸载功能使得开发人员提供更多的安装选择变得轻而易举。同样应用程序也可容易地部分或全部卸载。用户也能对独立的部件进行重装, 并增加或修改其特性。

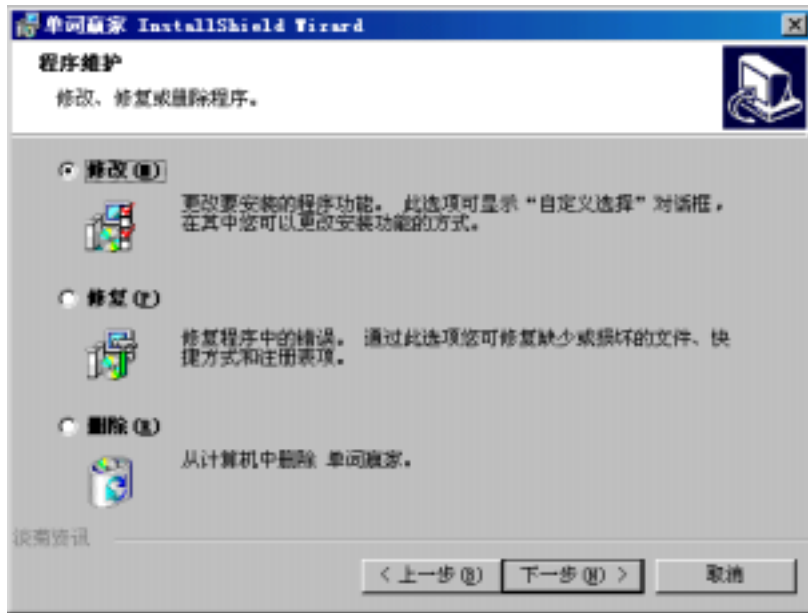


图 2-1 部件的安装和卸载功能提供了更多的安装选择

2.2.2 InstallShield 对象



图 2-2 使用 InstallShield 对象

现今，许多应用程序都要求对一些流行的第三方技术进行支持，比如 ODBC 数据库。举例说明，一个 Visual Basic 应付账应用程序安装在一个与公司网络上的 SQL server 数据库相连的工作站上。为了避免出错、失败和打搅系统管理员，在安装过程中，应付账应用程序的安装软件必须正确地完成数据源连接的设置。

InstallShield 以前的版本是通过使用预先写好的模板让开发人员放进他们的脚本，并按需要进行定制来实现的。尽管这样节省了编写代码的时间，但仍然需要谙悉第三方技术。

随着 InstallShield 对象的引入，开发人员现在可以与他们安装程序中所包含的第三方技术完全隔离。只要把需要的对象放进你的工程中，即可自动提供所需的支持。InstallShield

对象自动收集所需文件并且无需安装软件开发人员做任何编写脚本或设置文件组及安装部件的工作以支持第三方技术。

对于那个应付账应用程序来说，开发人员只需放入 ODBC 对象而无须知道 SQL server 支持文件和注册表的设置情况。当安装程序需要随着新版本的 SQL server 升级时，只要简单地替换 ODBC 对象即可。开发人员无需知道两个版本间的差异以及如何从安装脚本中去掉旧的代码。当应用程序卸载时，所有的文件和注册表中的设置一起清除干净。

如图 2-2 所示，ISPro6 包含了一组核心对象。它们是：ODBC、OLE、DAO、VB、MFC、DCOM、MDAC、RDO、Access 97、VB Runtime、DirectX 6 和 Configure NT Services。今后还将提供这些对象的升级版本以及另外的对象。

InstallShield 对象可以从 InstallShield 网站得到，InstallShield 将不停地更新并增加它们。这意味着如果开发人员需要更换或升级一个对象以及需要一个新对象时，可以立即下载它，而不依赖于 ISPro2K 的版本。开发人员可以容易地替换改对象并编译所维护的安装程序，比以前更快捷地向最终用户提供所需的服务。

2.2.3 基于事件的脚本模块

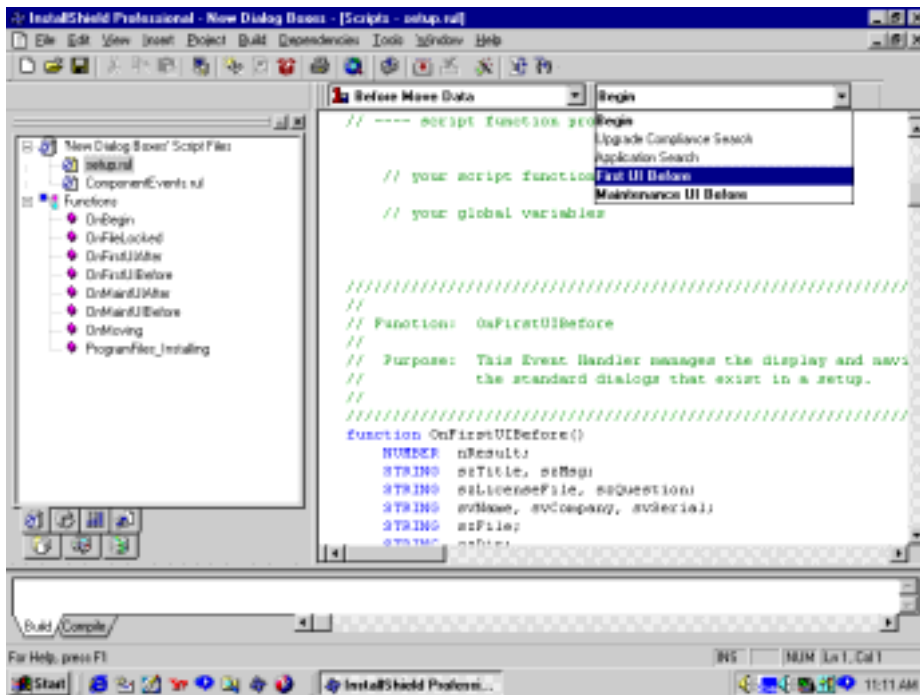


图 2-3 基于事件的脚本结构

如图 2-3 所示，ISPro6 采用了与 Visual Basic (VB) 类似的基于事件的脚本结构，这意味着大多数的安装程序不再需要 InstallShield Professional 早期版本必须的那么多脚本文件。与 VB 类似的结构也使得学习和使用脚本语言非常容易。ISPro6 包括事先定义好的对所有安装程序都适用的脚本框架。这个结构使得脚本建立在事件的基础之上，并提供了在更高层次上对脚本功能的把握。

2.2.4 Active Dependency Manager

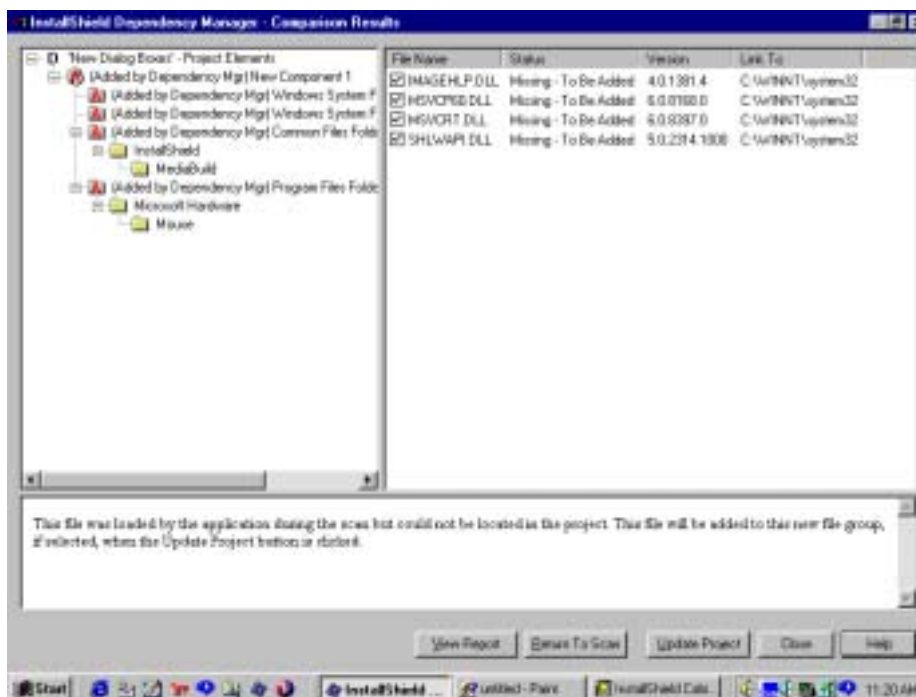


图 2-4 Active Dependency Manager

如图 2-4 示，有了 Active Dependency Manager (ADM) 的帮助，安装程序的开发者再也不会遗漏某个 DLL 文件了。ADM 能自动识别与安装工程有关的静态和动态 DLL 文件。这为跟踪安装程序中的所有 DLL 文件提供了完美的解决方案。

ADM 可以作为一个监控程序。开发者加载 ADM 程序，然后启动应用程序，确保访问所有的函数。ADM 能记录应用程序加载的所有 DLL 文件，并通过 ADM 独一无二的功能，启动一个对话框让开发者添加或跳过单个 DLL 文件。

2.2.5 导入注册参数

Windows 注册表非常复杂难于处理。当安装程序开发者手工添加注册参数时极易发生错误。ISPro6 通过自动导入参数解决了这个问题。从注册表编辑器 (REGEDIT) 中导出格式正确的注册表文件，并直接把这个文件导入安装工程。注册参数可以以标准“REG”格式在任何文本编辑器中创建、检查和修改。随后，这个文本文件被直接导入到工程中。

2.2.6 媒体编译器

开发人员在工程作过修改后不必再对整个工程进行编译。Build Media 功能只编译修改过的部分。当对工程仅作细微修改时，它可以大幅度减少编译的时间。当然你也可以随时选择对整个工程进行编译。

2.2.7 ISPro6 其他主要特征

■ 动态文件连接

ISPro6 提供了动态文件连接,使得开发者不需要跟踪工程中每个文件的变化。通过动态文件连接,在编译时只要事先定义了路径和通配符,所有源文件就可以自动被包含进工程。假如没有这项特性,一旦修改了文件名称或版本号,或者添加删除了文件,开发人员就不得不重新进行设定,跟踪每一个变化。动态文件连接可以保存修改次数并去除可能的错误。

■ 无拘无束的安装定制

ISPro6 提供了全部的“脚本”语句,包括超过 300 条预定义的与安装有关的函数。函数可以通过函数向导(Function Wizard)加入或者手工加入脚本。ISPro6 有一个类似 Visual C++ 的集成开发环境,帮助开发者便捷地修改和组织安装工程。集成开发环境是一个有语法着色功能的脚本编辑器,通过 InstallScript 的脚本语言可以完全控制安装的过程。ISPro6 还提供工程向导(Project Wizard)用来加速开发进度:使用向导只需九步即可创建好安装程序。Media Build Wizard 可以使安装工程方便的指向某个或所有的发布媒体、语言和平台。

■ 全面的发布选择

使用 ISPro6 可以将安装程序放到一张或多张 CD-ROM 上,或者自定义最大达到 4 GB 的容量,文件可以压缩或不压缩。现在 InstallFromTheWeb (IFTW)、PackageForTheWeb (PFTW)已经和 InstallShield Professional 6 集成在一起,可以在集成开发环境中调用这些 InstallShield 网络发布工具,还可以将工程信息导出(IFTW 单独出售)。ISPro6 中还包含了 InstallShield 日志文件阅读器(InstallShield Log File Viewer)和 InstallShield Cabinet 文件阅读器,前者用于浏览安装时生成的卸载日志(uninstall log),后者用于浏览打包在 CAB 文件中的文件组、组件和安装类型。

2.3 ISWI 的最新特点

同 ISPro6 类似,ISWI 也提供了方便易用的界面来创建强大而复杂的工业级安装软件。但是 ISWI 不同于 ISPro6 之处在于它使开发者能够充分利用微软的 Windows Installer 服务而不必考虑其复杂性。

微软推出的 Windows2000 包含了为减少 TCO 而设计的众多优秀特性。其中之一是 Windows Installer 服务,它使所有的 Windows2000 开发人员能够创建复杂的智能化安装软件。由于 Windows Installer 服务也支持 Windows95/98/NT4 和 Windows2000 操作系统,因此 Windows Installer 服务生成的安装程序亦可以在使用这些操作系统的机器上运行。

Windows Installer 服务的主要特征如下:

- ISPro6 可以进行功能部件级的安装或卸载(和 InstallShield 类似),这意味着可以只安装或卸载应用程序中的一个部件,或者是应用程序套件中的某个程序。
- 应用程序广告功能,可以不安装程序而只将应用程序的图标放置在桌面上,等到用户需要时,双击图标即可安装。
- 自动修复应用程序中受损组件的功能,可以不必卸载和再次安装程序就能修复各种错误。
- 安装失败后自动回退到原来状态的功能,可以使失败的安装不破坏原有的应用程序,以便更好地保护用户的系统。否则,残留的文件或参数可能导致其他应用程序出错。

Windows Installer 服务是 Windows 结构体系的一个重要突破,而不只是一个安装程序的

开发环境。Windows Installer 服务读取并处理其数据库中的一系列表。通过这些表并利用上述的功能，不必用到那些需要进行复杂而痛苦的手工处理的开发环境。

通过易用的界面、强大的功能和有用的向导，在开发者创建复杂的安装程序时，ISWI 能够对用于 Windows Installer 服务的表实现自动和高速处理。ISWI 使得开发人员不必熟知 Windows Installer 服务的所有细节，从而大大节约了开发时间。最重要的是，开发者使用 ISWI 可以创建基于 Windows Installer 服务的安装程序包，它就是用户早已熟悉的工业级安装界面。

下面先让我们快速了解一下 ISWI 的重要特性和向导：

2.3.1 对话框编辑器

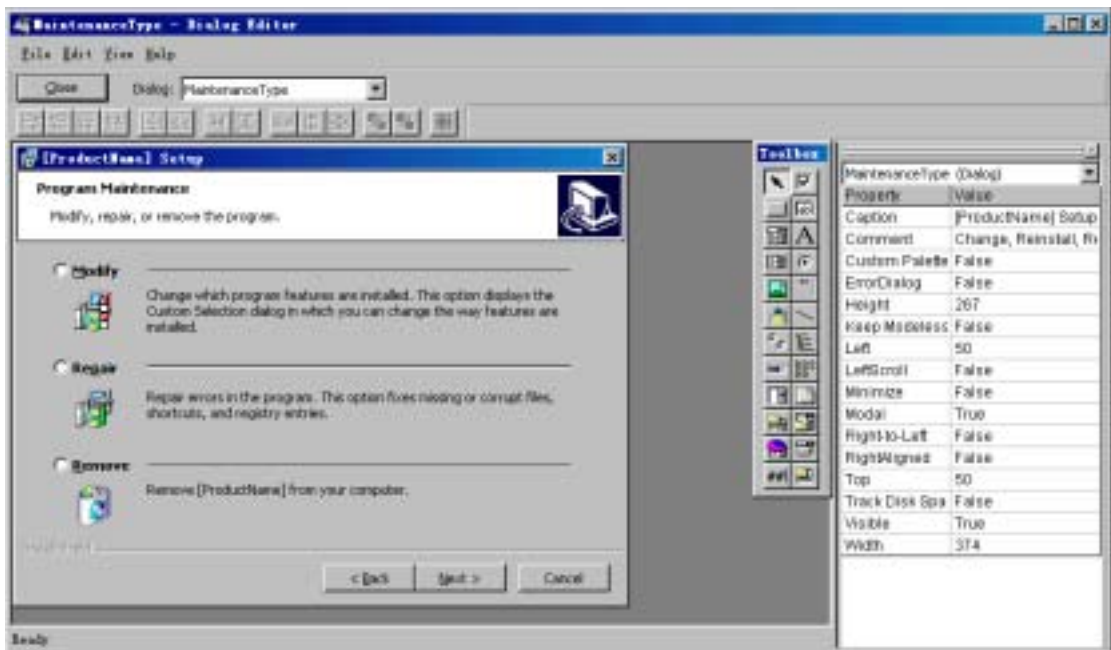


图 2-5 类似 VB 的对话框编辑器（Dialog Editor）

功能强大的全屏幕对话框编辑器（Dialog Editor）使得开发者创建和修改安装用户对话框非常容易。如图 2-5 所示，对话框编辑器使用了与 VB 相似的界面，包括创建窗口、工具箱和属性页，可以通过鼠标点击完成对单个属性进行添加、编辑或删除。

ISWI 内含超过 25 个预先设定的对话框，这些对话框也可以很容易地定制。开发者导出与对话框有关的信息，并把这些信息导入未来的工程中。这一功能使得在安装程序的多个应用程序间统一和更新安装界面变得十分容易。

2.3.2 国际版本的支持

与 InstallShield Professional 早期和现在的版本相似，国际版本的 ISWI 分为两部分出售：东方版和西方版。两部分加起来共有超过 20 种的语言支持。InstallShield 还为包括对话框、按钮和错误信息在内的最终用户界面提供了已经翻译好的文本字符串，他们是 ISWI 国际版本的重要组成部分。

使用 ISWI 可以很容易地对这些字符串进行编辑或扩充，还可以将字符串导出保存为文

本文件。这些文件可以自行编辑或者让人翻译。编辑后的文件能导入到一个已有的工程或新工程中去。比如，你已经编辑了一些字符串（新增的），并打算将应用程序进行国际化——比如：使用简体中文、日文、西班牙文和意大利文。方法很简单：首先用英语创建、编译和测试安装程序；一旦程序调试完毕，就把新文本字符串导出到一个文本文件里。把这个文件送给一个或多个翻译，让翻译将你的文本文件译为简体中文、日文、西班牙文和意大利文版本，而不必包括应用程序。然后把翻译好的文本文件导入到工程中。

开发者还能完全控制如何把语言版本交付给最终用户。安装程序可以：

- 只显示指定的语言。
- 自动读取当前 Windows 的语言环境，在安装时将其作为缺省语言。
- 允许安装用户选择语言。

2.3.3 SKU 管理器

SKU 管理器是一个省时省事的工具，它使开发者能够在一个工程中可以生成不同版本的应用程序——比如测试版、试用版、标准版和增强版。这一切只需在应用程序中为所有的功能部件设置一下“用户定义标记”（user-defined flags）。当为工程编译一个特别的发布版本时，只要将与选定标记相关联的功能部件作为一部分包含在其中即可。

SKU 管理器为开发者提供了非常多的版本控制，避免了不同版本间不必要的区别。

2.3.4 动态文件链接

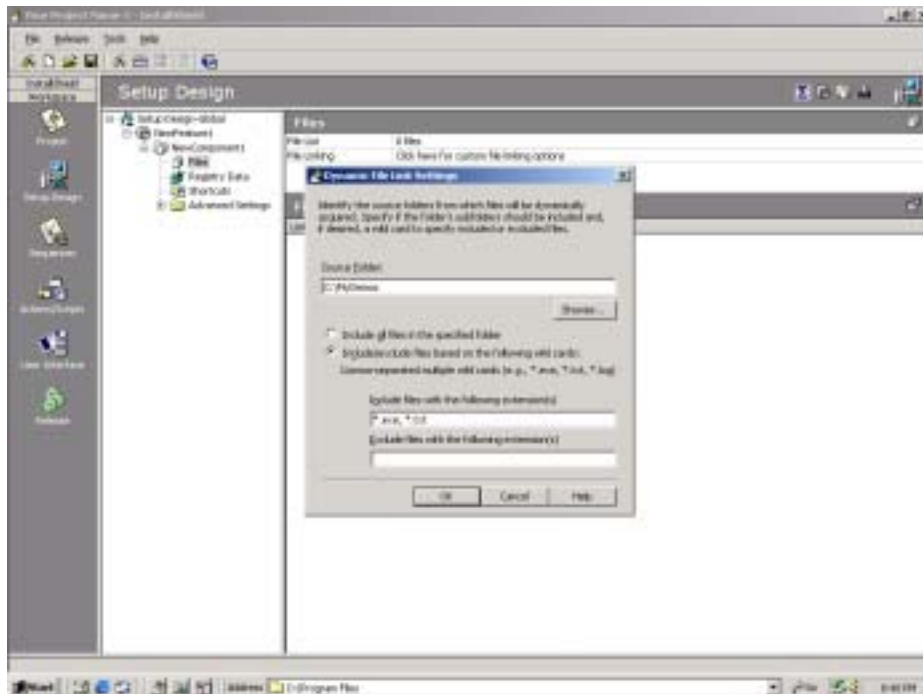


图 2-6 动态文件链接

如图 2-6 所示，和 InstallShield Professional 的早期版本相同，为了让开发者不必跟踪工程中每个文件变化，ISWI 也提供了动态文件链接。有了动态文件链接，只要事先定义了路径和通配符，所有的源文件在编译时会自动包含到工程中。假如没有这项特性，每当改变了

文件名或文件版本，或者添加删除文件时，开发者不得不重新调整每个变化后的设定。动态文件链接节省了时间，并消除了可能的错误。

2.3.5 融合模块的处理

融合模块的处理允许开发者对诸如运行时刻引擎或资源库那样的数据元素打包，以便在多个应用程序中使用。与 Windows Installer 服务安装包使用的标准 MSI 文件格式不同，融合模块的处理允许创建一个以 MSM 为后缀的子集。这些文件包含文件、注册文件、快捷方式、逻辑关系等，供以后的工程重复使用，以便节省大量的开发时间和增加应用程序间的一致性。

2.3.6 组件向导

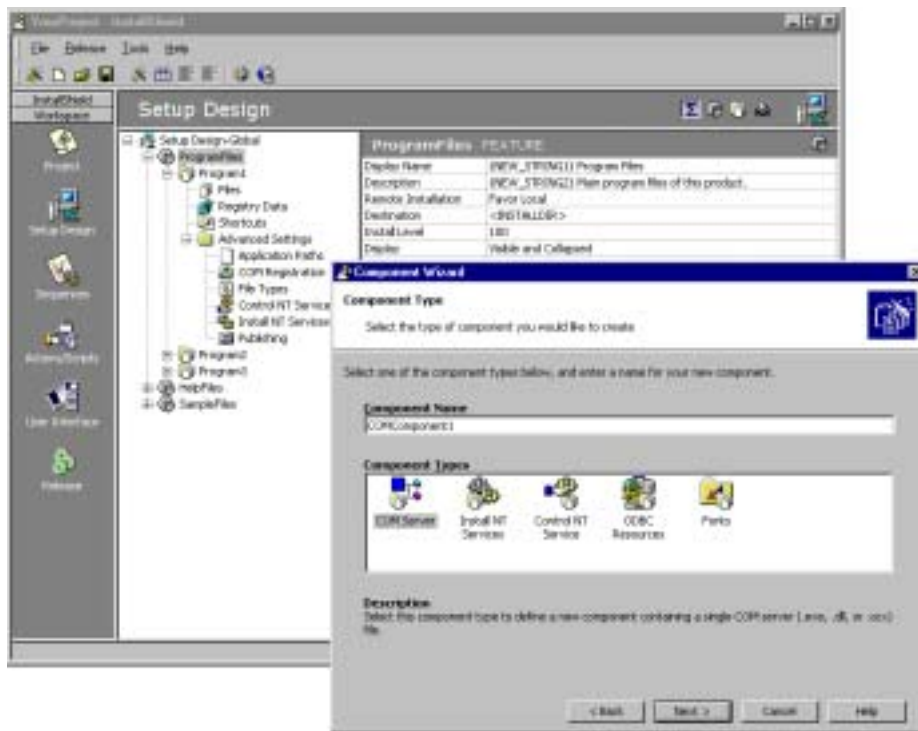


图 2-7 组件向导

组件是基于 Windows Installer 服务的安装软件中最基本的单元，用户使用组件向导可以快速创建一个组件。如图 2-7 所示，ISWI 组件向导支持包括 COM 服务器、ODBC 源、Windows2000 服务和控制以及字体在内的多种组件。

另外，组件可以从基于微软“最佳实践”方针的文件池中自动生成。“最佳实践”方针得到 InstallShield 软件公司的有力支持，它包括了像“每个组件只能有一个可携带的执行文件”以及“一个文件不能归属两个组件”那样的一些限制，以便开发者尽可能地利用 Windows Installer 服务的功能，使开发过程标准化。

2.3.7 工程向导

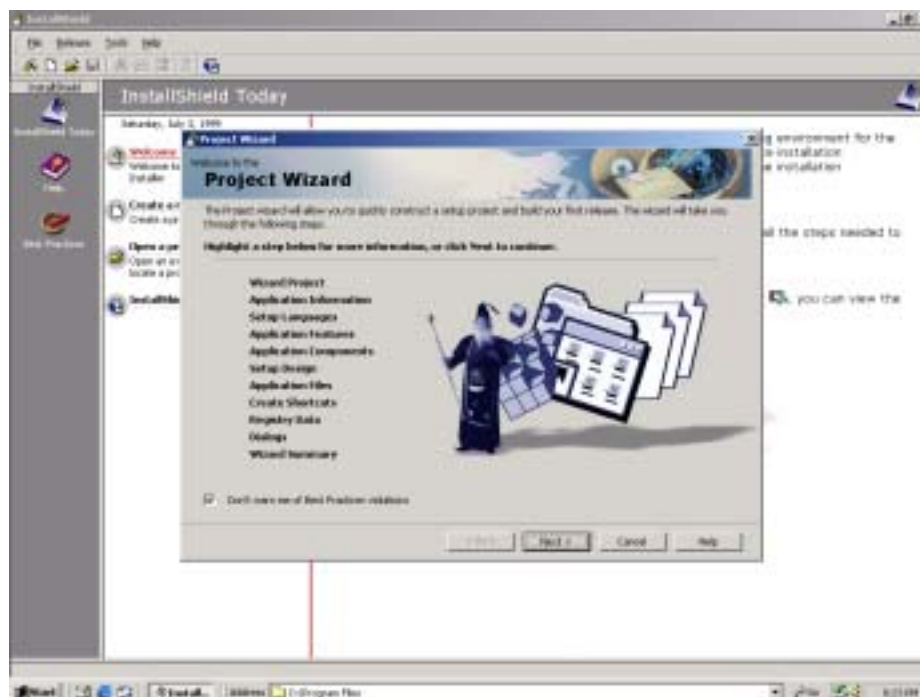


图 2-8 工程向导

工程向导是创建工程的得力工具，开发者只需指定应用程序、功能部件、组件、文件、注册文件和快捷方式并通过几个简单的步骤就能创建一个安装程序，如图 2-8 所示。这对初学者熟悉安装软件的结构和内容是再好不过的了。即使是有经验的开发者也愿意先用向导来创建新工程，然后再使用详细的视图来增强和定制安装程序的每一部分。

2.3.8 最佳实践向导

在创建工程时，“最佳实践向导”（Best Practice Wizard）在后台运行，并自动提示开发者最佳的操作方案。提示时，向导出现一个对话框，列出修改建议。开发者可以选择接受，也可以不接受建议。

2.3.9 发布向导

发布向导是编译安装软件的得力助手，如图 2-9 所示。开发者可以很容易地完成以下工作：

- 指定媒体类型(CD、DVD、自定义等)。
- 决定将应用程序或功能部件放在哪一张盘上（多张安装盘时）。
- 指定工程中要压缩的部分。
- 选择包含的语言。

基于语言标记的语言过滤功能，可以使不同的文件根据语言版本进行分类。也可以根据功能部件进行分类（见前述 SKU 管理器），以便生成不同的安装版本，例如测试板、试用板、

标准板和增强版。

发布向导还允许开发者在安装程序包中加入 Windows Installer 服务本身。当开发者不能肯定目标系统中是否有 Windows Installer 服务时，这样做极为有益。ISWI 可在 NT 和 95/98 上安装 Windows Installer 服务。

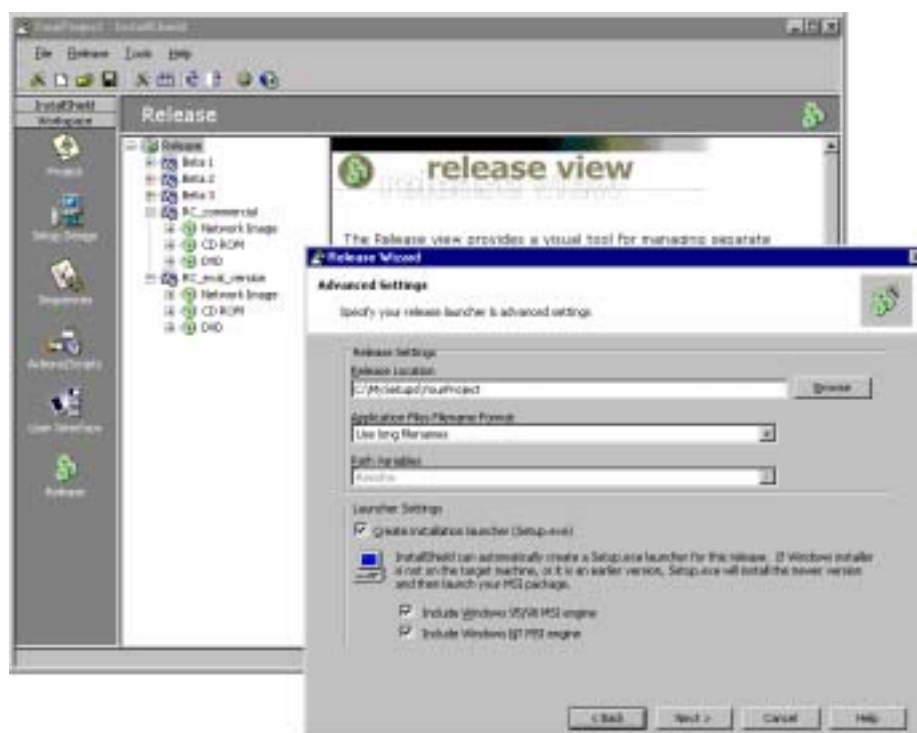


图 2-9 发布向导

第3章 十五分钟快速搞定安装软件

如果要制作普通的应用程序软件安装软件，实际上使用 InstallShield 要比我们想象的还要简单。由于 InstallShield for Windows Installer (ISWI) 在国内介绍的比较少，而且它又是 InstallShield 的发展趋势，所以我们在这里重点介绍 ISWI 的应用。


使用安装工程向导可以快速搞定安装软件。下面我们就通过一个例子来进行演示讲解。例子使用的程序是位于<InstallShield for Windows Installer 安装目录>\Samples\Othello\Data Files\下的 Othello game 文件，这个文件是在安装 ISWI 时提供的。我们将用它来生成一个安装程序。另外，一个名为 Othello.ism 的示范工程已经生成，在向导结束时可以马上看到该工程的大致面貌。这个工程位于<InstallShield for Windows Installer 安装目录 >\Samples\Othello\Project Files。

安装工程向导可提供有关应用程序的信息，并通过一系列简单易用的界面帮助我们设计一个新的安装工程。如果能巧妙的设置向导的某些值，那么，在退出向导前你就可以发布自己的第一个安装工程了。在这里我们还将向你提供各种建议，以便你将安装软件做得更为出色。

3.1 启动工程向导



图 3-1 工程向导的 Welcome 面板

单击工具栏左侧的 Project Wizard (工程向导) 按钮 ，启动向导。向导中的每个步骤依次列于欢迎界面中，如图 3-1 所示。将鼠标指向其中的任意一个，都可以获得详细的介绍。单击 Next 继续。界面也提供了打开 Setup Best Practice 的选项。此选项缺省值为关闭。为启动该功能，在界面底部的复选框中打勾。

对话框选项：

Ignore Best Practices violations

Windows Installer 的 Setup Best Practices 可以帮助生成简洁和可重复使用的代码，以便处理令人头痛的 DLL。作为缺省值，InstallShield 不会自动扫描安装工程，因此当第一次打开集成开发环境时，可以选定这项功能。

如果没有选择 Ignore Best Practices violations 功能，InstallShield 会监控工程向导中的程序。在 Application Files 面板上，对于不满足最佳实践要求的任何文件，会放置一个警告图标。点击 Details 按钮能查看到详细信息。

最佳实践提示是 InstallShield 集成开发环境里的一个选项，可在 Tools|Options 菜单项中进行设置。在工程向导中使用到的最佳实践监视功能，也能在集成开发环境视图下使用。

3.2 命名新工程

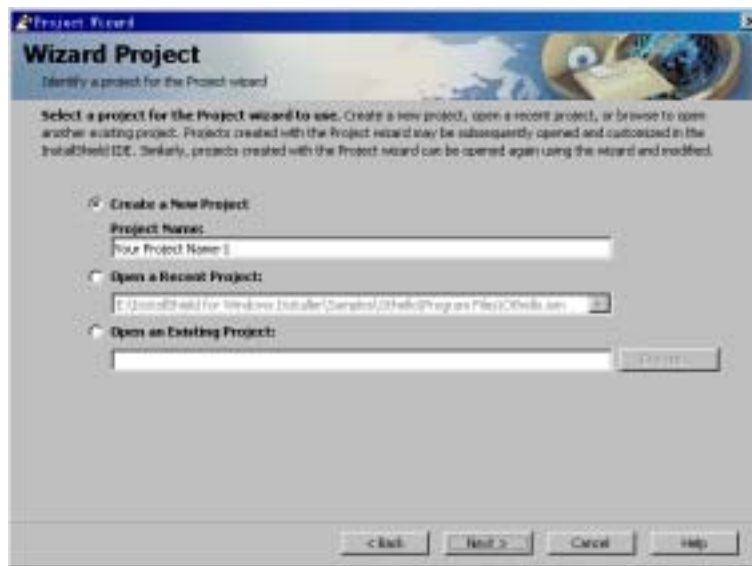


图 3-2 工程向导的 Wizard Project 面板

工程向导可以草拟一个新工程或打开现有的工程进行编辑。在这里你可以创建一个新的工程，或打开一个已有的工程进行编辑，如图 3-2 所示。单击 Create a New Project 并输入新工程的名称，比如“Othello”。单击 Next 继续。

对话框选项：

Create a New Project

为了命名新工程，InstallShield 提供了缺省名称：Your Project Name-n（n 为连续的数），也可以输入自己喜欢的名称。

为了不和源文件路径冲突，工程名不能包含以下字符：\: *?"<>|。

Open a Recent Project

通过选择此选项，可以在下拉列表中选择最近使用过的工程进行编辑。

Open an Existing Project

选择此选项，输入完整的路径或单击“浏览”（Browse）按钮来定位已存在的 InstallShield 工程（.ism）文件。

3.3 给出应用程序的有关信息

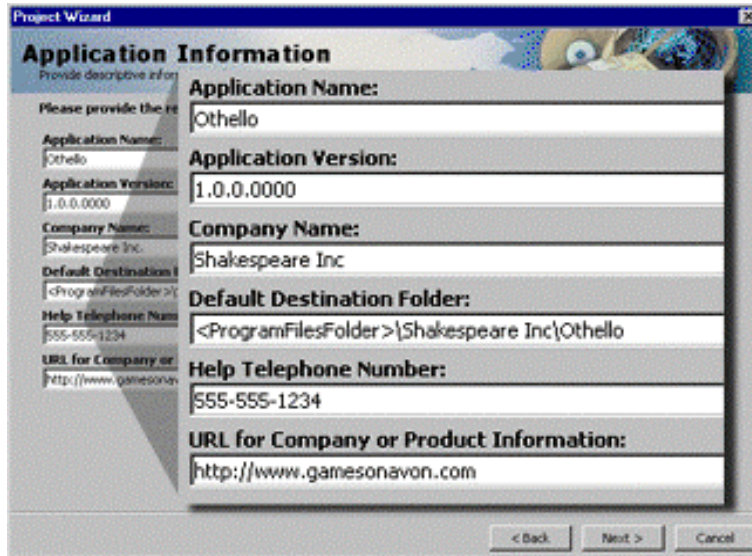


图 3-3 工程向导的 Application Information 面板

在这一步，要求你提供有关你的公司和应用程序的信息。这些信息主要用于注册时生成注册键，以及作为 Windows 2000 的 Add/Remove Programs 中的信息。请按如图 3-3 所示的示范填写。向导将根据你的公司和应用程序的名称来设置目标文件夹的值。单击 Next 继续。

注意，这里输入的信息将用来设置几个重要的工程属性和产品属性。

对话框选项：

Application Name

这里输入的值将在整个工程中使用，包括：

- 缺省的产品名
- 工程目录下的源文件夹名
- 编译程序包名(.msi 文件)
- 该值还用在安装用户面板的对话框中

提示：

为了不和源文件路径冲突，程序名不能包含以下字符：\: *?"<>|.

要注意的一点是，程序名将作为符合 Windows 标准的注册信息来进行注册。

该信息的值位于：

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\<产品代码>

用户可以使用“添加删除程序”（Add/Remove Programs）自行修改或删除产品。

注意，InstallShield 将程序名作为缺省目标文件夹值的一部分。

Application Version

输入应用程序的版本号。由于该值符合 Windows 的安装标准，因此当单击 support information link（支持信息链接）时，该值将出现在 Add/Remove Programs（添加删除程序）面板中。

Company Name

输入公司的名称。工程向导将用这个值来设置部分程序的缺省目标文件夹的路径。该值

将作为产品信息进行注册。当点击 support information link（支持信息链接）时，它将出现在 Add/Remove Programs（添加删除程序）面板中。

Default Destination Folder

产品所有的功能部件和组件都存放在 INSTALLDIR 中，这个缺省的目标文件夹为 INSTALLDIR 提供了初始值。（缺省目标文件夹将作为产品的 Destination Folder 属性）

为了替代硬编码的路径，应使用尖括号中的 Windows Installer folder property 属性。鉴于满足 Windows 安装标准的应用程序通常安装在\Program Files 文件夹中，因此向导建议将该文件夹作为缺省的目标文件夹，即<ProgramFilesFolder>。

Help Telephone Number

输入向用户提供技术支持的电话号码。当单击 support information link（支持信息链接）时，此号码将出现在 Add/Remove Programs（添加删除程序）面板中。

URL for Company or Product Information

将产品或公司的网址提供给用户，以使用户了解更多信息。在 product 属性中可以给帮助(Help)、更新(Updates)和信息(Information)分别设置 URL。注册时会用到该值。当单击 support information link（支持信息链接）时，该值将出现在 Add/Remove Programs（添加删除程序）面板中。

3.4 选择安装语言



图 3-4 工程向导的 Setup Languages 面板

安装程序可以使用多种语言。通过此面板可以选择安装程序运行的语言环境，如图 3-4 所示。缺省选项为英语，若要选择其它语言请选择相应的复选框。如果要增加对某种语言支持，Install Shield 可向你提供支持该语言的字符串表，和已翻译好的用户界面等资源，并允许你用该语言发布安装程序。当增加一种语言时，InstallShield 将向你提供一张列表，列出已翻译好的用户面板，并允许用这些语言发布安装程序。一旦生成了工程，就可以在工程的 Setup Languages 属性中对系统支持语言进行修改。还可以进入 Project 视图，在列表中修改缺省语言(Default language)，其在 Setup Languages 列表中用红旗标出。如果选择了工程向导里的 build your first release，那么，面板还能决定安装程序包使用的语言。

单击 Next 继续。

注意：

只有购买了Install Shield提供的相应配套软件后才能添加对应的语言。如果是使用中文等双字节语言文字的支持，请购买Install Shield的东方国际版。

对话框选项：

Setup Languages

单击一种语言将其添加到工程中。如果选择的语言目前无法使用，请浏览 InstallShield 的网页，购买相应的支持程序。

InstallShield Web site

浏览 InstallShield 的网页，可以查阅 InstallShield 支持的语言或 InstallShield 国际版本。

3.5 创建功能部件

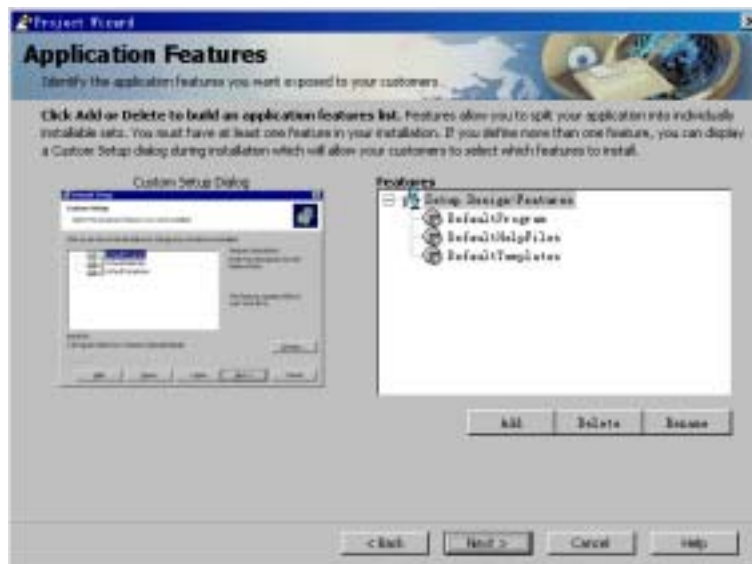



图 3-5 工程向导 Application Features 面板

如图 3-5 所示，设计安装程序的一个重要步骤是按功能部件组织产品——即让用户来选择安装哪些应用程序。你将根据安装功能部件来组织程序，以便最终用户选择安装要素。比如，可能希望为产品中的所有教程提供子功能部件和单独的功能部件。

在本例中我们只需要一个 Default Program 功能部件。为了删除向导所提供的其他缺省功能部件，选中这些功能部件并单击 Delete。单击 Next 继续。

对话框选项：

Add

如果最上层的一项  Setup Design Features 被选中，单击 Add 按钮生成一个新功能部件。假如选中的是功能部件则增加新的子功能部件。工程向导将以缺省的 New Feature n (n 为连续值) 命名新功能部件。可以立即修改名称，也可以稍后再改。亦可以右击功能部件树，选中 New Feature，单击 Add。

Delete

选中一个功能部件或子部件，单击 Delete 将其从树中删除。或者在功能部件上点击鼠

标右键，选择 Delete。

Rename

选择一个功能部件或子部件，点击 Rename 按钮重新命名。按 F2 键或在功能部件上点击鼠标右键，选择 Rename 也能重新命名。功能部件树中的各部件名称必须是唯一的。名称可以包括字母、数字、点(.)和下划线(_)，但必须以字母或下划线开头。在 Setup Design-Global 或 Setup Design-Features 视图中，可以为显示名指定序列号 (string ID)。

3.6 生成组件



图 3-6 工程向导的 Application Components 面板

组件是一种极为有用的工具，它能帮助你合理地安排类似的程序数据，如文件、注册项和快捷方式。每项功能部件至少应由一个相关组件组成。一个组件包括相关的程序资料，比如文件、注册表和快捷键。当最终用户安装了功能部件或子功能部件，相关数据将安装到目标系统中。

本例中我们只需两个组件：Program_Executables 和 Game_Pieces。如图 3-6 所示，为了删除向导提供的其他缺省部件，选中并单击 Delete。为了生成 Game Pieces 部件，单击 Add 并输入部件名称。重复上一步生成 Program_Executables 部件。选中各部件的 default self-registration and destination folder 属性。该属性使用缺省的自注册和目标文件夹。

单击 Next 继续。

提示：

由于需要将组件添加到工程里，因此假如本面板中未生成任何组件，向导将会直接跳到 Dialogs 面板。工程中应含有至少一个组件，这样才能使用 Setup Design、Application Files、Create Shortcuts 和 Registry Data 面板。

对话框选项：

Files are self-registering

如果组件中的所有文件都是自注册文件的话请选择此项。

注意：


调用自注册功能不符合最佳实践方案。作为补救方法，组件中要有单独的 EXE。

DLL,或 OCX文件,并用组件向导生成组件并提取文件的注册信息。

Component Destination/Standard Destination Folders

这里提供的值将决定组件的文件安装到系统的位置。输入到目标文件夹的内容将作为组件的 Destination Folder 属性。请使用标准目标文件夹列表尖括号中的 Windows Installer Folder 属性替代路径的硬编码。在选择新文件夹时请注意,它将覆盖 Component Destination 域中的所有内容。

Add

在组件上单击 Add 按钮生成一个新组件,并按字母顺序添加到树中。工程向导将以缺省的“New Component n”(n 为连续值)命名新组件。既可以马上修改其名称,也可以稍后再改。在顶端的  Setup Design-Components 上单击鼠标右键,选择“New Component”也可添加新组件。

Delete

选中一个组件,单击 Delete 按钮将其删除。在组件上单击鼠标右键选择 Delete 相当于单击 Delete。

Rename

选择一个组件,点击 Rename 按钮可重新命名该组件。按 F2 键或右击组件选择 Rename 也可重新命名。

3.7 关联组件和功能部件



图 3-7 工程向导的 Setup Design 面板

这一步是建立组件和功能部件间的关联。安装时,所有的组件必须与至少一个功能部件相关联。如图 3-7 所示,本面板将帮助建立组件与一个或多个功能部件或子部件间的关联。注意,只有 Application Components 面板里有多个组件时才会出现本面板。

虽然一个组件可能拥有多个功能部件,但本例中只需要一个功能部件。为了将组件和 Default_Program 功能部件相关联,选择 Setup Design 列表中的 Default_Program 功能部件,将组件列表全部选中并单击 Add。

单击“Next”继续。

对话框选项:

Add

要建立组件与功能部件间的关联, 可以:

- 在 Setup Design 列表中选择功能部件或子部件。
- 在 Components 列表选择一个组件 (或者按住 Shift 或 Ctrl 键单击 Additional components)
- 单击“Add”按钮。

一旦组件添加到功能部件后, 当在那个功能部件上单击时, 该组件将从 Components 列表中删除。

Remove

选中 Setup Design 列表中的某个组件, 单击 Remove 按钮, 可撤销它和功能部件或子部件间的关联。当某个组件和功能部件之间没有关联时, 它将出现在该功能部件可用的组件列表中。

3.8 把文件关联到组件

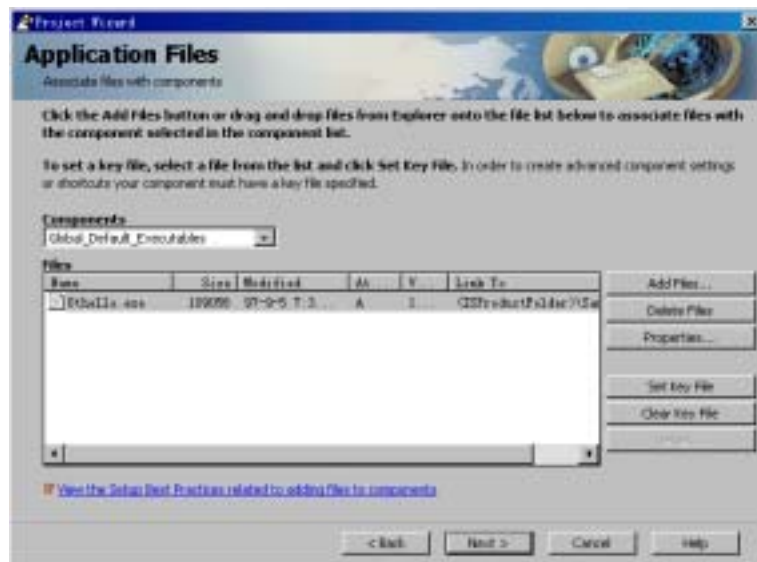


图 3-8 工程向导的 Application Files 面板

通过图 3-8 所示面板可以把程序文件安排到组件中, 并详细说明这些文件的安装信息。注意, 只有 Application Components 面板里有多个组件时才会出现此面板。

文件是与某个组件关联的主要数据。选中一个组件并单击 Add Files 按钮, 将其连接到你的程序文件。记住, 所有与组件关联的文件必须有相同的自注册、覆盖和目标文件夹等属性。

本示例中, 你将为 Othello 安装文件, 这些文件位于 <InstallShield for Windows Installer 安装目录>\Samples\Othello\Data Files 下。主执行程序属于 Program_Executables 组件。为了连接到这个文件, 首先在组件列表中选中 Program_Executables。再单击 Add Files, 浏览 Data Files 子文件夹。然后, 选择 Othello.exe 并单击 Open。部件列表中就出现了 Othello.exe。

接着将在以 Program_Executable 为目标的系统中创建 Othello.exe 的快捷键, 你必须为这个部件设置一个 key file (键文件)。选中 Othello.exe 并单击 Set Key File。

下一步, 为 Game_Pieces 组件中的所有.gif 文件建立链接。这些文件位于 Data Files 子

文件夹中。该组件不需要键文件。在左边的 Application Files 界面中单击可进行预览。单击 Next 继续。

注意：
Othello.exe 需要使用某些核心部件和旧系统中可能不存在的 Visual Basic 运行时刻文件。要使示例程序正确运行，你可能需要在安装程序中插入 Oleaut32.dll, Msvbvm50.dll, Comcat.dll, Olepro32.dll, AsyncFilt.dll, Ctl3d32.dll, 和 Stdole2.tlb 的融合模块，也可通过组件向导为这些文件生成组件。

对话框选项：

Components

首先是从列表中选择一个组件。此面板里的其他操作都和这个组件有关。重复执行本操作，可完成工程中文件与组件的关联。如果要为程序文件添加新组件，那么请单击 Back 按钮退回到 Application Components 面板。

Files

此处可以看到和组件相关联的文件的有关信息。可以通过以下三种方法向列表中添加文件：

- 右击文件列表，选择 Add
- 从 Windows 资源管理器（Windows Explorer）中拖放文件到列表中
- 单击 Add Files 按钮

如果在欢迎面板中选择了 Setup Best Practices，让工程向导扫描安装程序，那么当操作不符合最佳实践方案时，不符合的文件前将出现警告图标。单击 Details 按钮以了解为何此文件不能包含在组件中。

注意：
每个文件的路径是硬编码链接，如果文件被移动或重新命名，Link To 值将显示 *** File Not Found ***，编译安装程序包时也不能解析该文件链接。

Add Files

单击 Add Files，浏览要添加到组件中的文件。在 Resulting 对话框中可以按住 Shift 或 Ctrl 键来选择要添加到文件夹里的文件。记住，Application Components 面板中的选项对组件中的每个文件都起作用。右击文件列表选择 Add 相当于单击 Add Files 按钮。

Delete Files

选中一个或多个文件单击 Delete Files，将其从组件中删除。在文件列表中单击鼠标右键并选择 Delete 相当于单击 Delete Files 按钮。

Properties

选中一个文件单击 Properties 按钮，可以浏览其安装属性。

Set Key File

选中一个文件单击 Set Key File 按钮，将其设为键文件。

Clear Key File

如果组件里不再需要键文件，请单击 Clear Key File 按钮。由于组件里只能有一个键文件或键路径，因此有必要在重设键文件或键路径前先清除原有的键文件。如果没有这样做，向导在设置一个新的键文件前会提示清除当前的键文件或路径。

Details

此按钮只有在打开 Best Practices notification 并选择了带有警告图标的文件后才能使用。单击 Details 按钮可以了解为何最佳实践方案不包含此文件。

3.9 创建快捷键



图 3-9 工程向导——Create Shortcuts 面板

该面板用来设置组件安装或发布时所需的程序文件夹和快捷方式，如图 3-9 所示。快捷键与组件的键文件相联系，并于安装了组件的功能部件后在目标系统中生成。通过该面板你可以在目标系统的 Start|Programs 菜单中为 Othello.exe 文件创建快捷键。请依次执行以下步骤：

1. 选中部件下拉列表中的 Program_Executables。注意，这个部件的键文件提供了快捷键的目标值。
2. 在快捷键列表中选中 Programs Menu。
3. 单击鼠标右键选择 New Shortcut，输入 Othello 作为新快捷键名。
4. 由于未指明快捷键的任何值，组件的键文件“Othello.exe”将被用作快捷键的目标值和缺省图标。
5. 最后，单击 Next 继续。

对话框选项：

Components

首先是在列表中选择一个组件。面板中所有的快捷方式都将对该组件起作用。如果要为所有的文件生成快捷方式，请重复以上步骤。当要向工程中添加新组件时，请单击 Back 按钮退回到 Application Components 面板中。

Target/Standard Destination Folders

在安装时，Target 域的值将成为快捷方式的 Target 属性。可以使用 standard destination folders 下拉列表中的<Windows Installer Folder>属性替代指向目标文件夹的硬编码。选择新文件夹时要注意，它将覆盖 Target 域里现有的值。如果 Target 域为空，组件的键文件将被用作 Target 的快捷方式。只有选择了 Shortcuts Explorer 里的快捷方式后，这些选项才可用。

Icon

在这里输入有效的路径，或者浏览带有快捷方式图标的文件。必须指定一个可执行文件，因为 Windows Installer Service 无法辨认单独的.ico 文件。由于发布组件时 Windows Installer 需要自己单独的图标，因此 InstallShield 将从你指定的可执行文件中选用一个图标。当有多

个图标时请指定其中的一个。当从 Shortcuts 树上选择了一个快捷方式后，Icon 域和 Browse 按钮才可用。

Icon Index

当文件中有多个图标时请指定其中的一个图标索引。在可执行列表中用正整数标明图标索引。例如，0 或 1 表明文件里的第一个图标，2 是第二个，3 是第三个，等等。用负整数标明源文件的标识符。例如，“index-12”表明此图标的标识符为 12。

Shortcuts

InstallShield 的 Shortcut 浏览器中的这个子目录列出了组件里现有的快捷方式，以及程序文件夹中包含了快捷方式的标准文件夹。

New Shortcut

在 Shortcut 浏览器中选择一个文件夹，单击 New Shortcut 按钮设计快捷方式。工程向导将以缺省的“New Component n”（n 为连续值）命名新快捷方式。既可以立即修改名称，也可以稍后单击 Rename 修改。Shortcut 浏览器目前还不支持向 Windows 任务栏中添加快捷方式。在 Shortcut 浏览器上单击鼠标右键，选择 New Shortcut 也可添加新的快捷方式。

New Folder

在 Shortcut 浏览器里选择一个文件夹，单击 New Folder 生成一个新程序文件夹。当安装此组件时，该文件夹将被安装至目标系统中。工程向导将以缺省的 New Folder n（n 为连续值）命名新文件夹。可以马上修改名称，也可以稍后单击 Rename 修改。目前还不能向 Windows 任务栏里添加文件夹。在 Shortcut 浏览器上单击鼠标右键，选择 New Folder 也可添加新文件夹。

Delete

在 Shortcut 浏览器里选择一个文件夹或快捷方式，单击 Delete 可将其从安装工程中删除。在 Shortcut 浏览器的文件夹或快捷方式上单击鼠标右键，选择 Delete 也可将其删除。

3.10 配置注册表信息

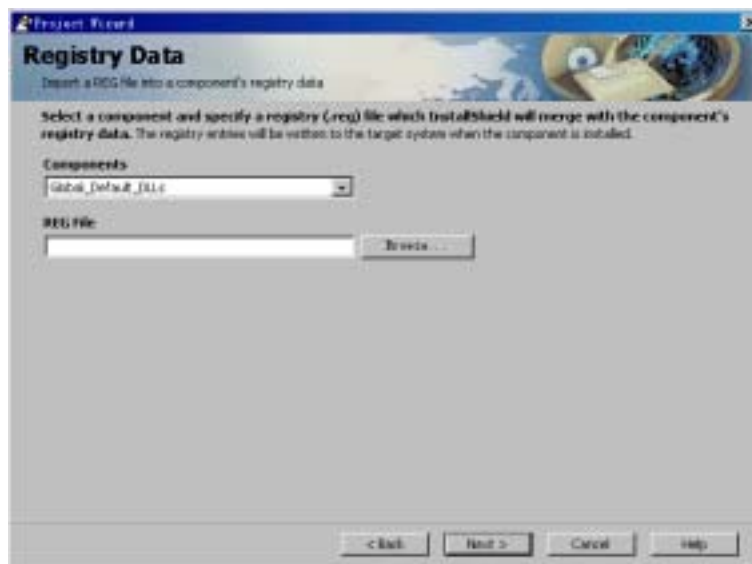


图 3-10 工程向导——Registry Data 面板

该面板用于向注册表添加注册信息，或向组件的数据包中添加 REG 文件，如图 3-10 所示。只有当 Application Components 面板中有多个组件时，才会出现本面板。由于此例非

常简单，不需要象别的程序那样修改注册表，因此在 Registry Data 页面上单击 Next，向导将继续创建你的安装程序。

对话框选项：

Components

首先从列表选择一个组件。导入的任何 REG 文件都将与选中的组件相关联。对工程里的每个组件都执行该步骤，使其都有与之关联的 REG 文件。当需要向工程中添加新组件时，请单击 Back 按钮退回到 Application Components 面板。

REG File

为导入到组件中的 REG 文件指定完整的路径，或对其浏览。向导可以为每个组件导入一个 REG 文件。也可以稍后再在 InstallShield 注册表浏览器里导入追加的 REG 文件。

此处指定的文件将不包含在安装工程里，因为它们的数据被合并后已没有参考价值。

3.11 定义用户界面

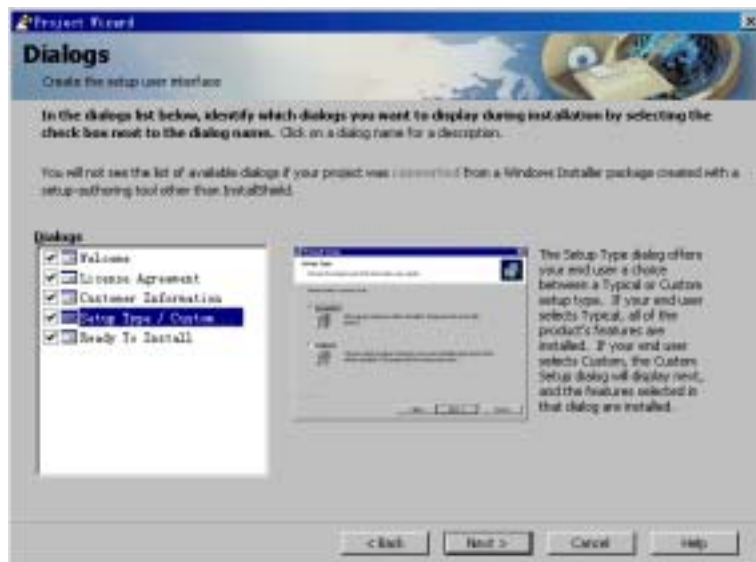


图 3-11 工程向导——Dialogs 面板

此面板提供用于安装队列的预定义安装对话框。只有在向导里草拟了一个工程后，才能看到这一系列对话框。如果只是对现有的工程进行编辑，将无法选择对话框。办法是必须进入 Sequences 视图或 User Interface 视图，再指定要显示的对话框。

此界面向允许你创建标准安装用户界面对话框。这些对话框将按列出的顺序显示。

注意：

只有最终用户选择了自定义安装，并且你在工程向导对话框或集成开发环境中选中了此选项，“自定义安装程序”对话框才会出现。

要深入了解这些对话框，单击列表中对对话框的名称，可以看到对话框的缩略图。最后检查所有将出现在最终用户面前的对话框。

接受向导中的缺省设置，我们就可以使用最常见的用户界面。单击 Next 继续。

对话框选项：

Welcome Panel

选择 Welcome Panel 选项，安装时将显示标准欢迎面板。

License Agreement

选择 License Agreement, 可要求用户在安装前阅读并接受产品的授权协议(EULA)。InstallShield 为 EULA 提供了位置标志符, 用来在对话框编辑器里指定文件, 以及编辑带滚动条的文本对话框中的 File Name 属性。

Customer Information

选择 Customer Information 可显示对话框来提醒用户进行注册, 并提供给当前用户或多用户 (需要多用户的系统, 比如 Windows 2000) 安装时的选择。

Setup Type/Custom Setup

选择 Setup Type/Custom Setup, 让用户选择 Complete Setup (完全安装——安装所有必需的功能部件) 和 Custom Setup (自定义安装——用户可以自由选择要安装的部件)。只有用户选择了 Custom Setup 后才显示 Custom Setup 对话框。

Ready to Install

选择 Ready to Install, 可以让用户确认以前所做的选择。

3.12 保存工程, 创建发布媒介, 测试安装程序

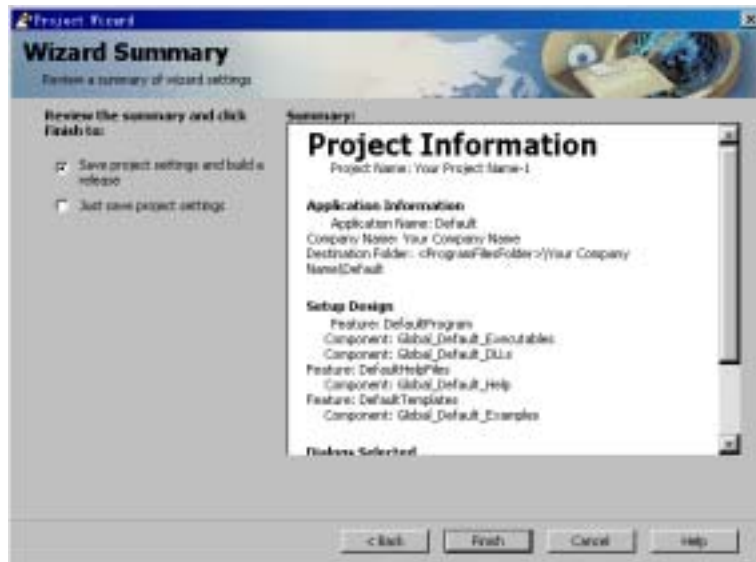


图 3-12 工程向导——Wizard Summary 面板

最后一个面板是 Wizard Summary, 让你回顾工程向导中所做的全部设置, 以便做最后的修改, 如图 3-12 所示。在这里, 你可以第一次保存自己的新安装程序 (其后缀为.ism)。选择其中的一个选项, 并单击 Finish, Othello.ism 将保存在缺省设置的<我的文档>\MySetups\Othello 下。

这里可以创建首个安装包发布媒介或者保存新工程。选中 Save project settings and build a release, 并单击 Finish。有关创建的情况, 以及任何错误或警告都将显示在 Output 窗口中。

此时新安装工程在 Project 视图中打开, 你可以做进一步修改。在工作区中点击快捷键, 以便熟悉集成开发环境, 并了解工程向导如何以你的信息为基础来设置所有参数。你也可以重新使用工程向导对你的工程进行修改。

进入 Release 视图并单击鼠标右键, 选中发布程序的名称 (缺省值为 My Release-1)。你的安装程序将在硬盘上直接运行。你的 MSI 包和数据文件位于<我的文档>\MySetups\Othello\Build Label 1\My Release-1\DiskImages\Disk1 中。

对话框选项:

Save project settings and build a release

选择此项并单击 Finish，认可总结，保存 InstallShield 工程 (.ism) 文件,并编译安装程序包 (.msi 文件)。Output 窗口里的编译统计则包含了编译中所有的错误和警告。此时新建的安装工程将自动打开 Project 视图，以便做进一步修改。可以单击工具栏的 Run Setup 按钮



来运行新安装程序。

Just save project settings

选择此选项，单击 Finish 认可总结，并保存 InstallShield 工程 (.ism) 文件。新建的安装工程将打开 Project 视图以便做进一步修改。

3.13 小结

利用 InstallShield for Windows Installer 的向导，我们已经成功地在十五分钟内做出了自己的第一个安装程序包。通过这个例子，我们了解了工程向导的功能特征，这是一个非常重要且简单易用的工具。同时我们还了解了有关安装工程的一些概念——创建功能部件、组件结构、快捷键和用户界面等等。为了进一步了解工程向导如何在集成开发环境中生成视图，以及如何对工程做进一步更改，可以单击工作区视图栏中的所有快捷键。

祝贺你已经建立了一个简单的安装工程！以次为基础，你可以继续深入学习 InstallShield 更多的高级开发技术。

第4章 国际版本安装软件制作技术

4.1 如何制作国际版本安装软件

国际版本安装软件是指能够使用多种语言文字安装界面,并安装相应语种版本应用程序的安装程序。使用国际版本安装软件是为了能够使得安装软件的使用及应用程序的发布尽量本地化,即满足当地用户的语言文字习惯。这对于开拓国际市场,推广软件产品有着积极的作用。然而,使用本地化的语言制作软件和安装软件在以前都不是一件容易的事情。这主要是因为计算机行业的标准和习惯都是建立在英语的基础之上。特别是像中文这样的双字节文字,在使用中经常会导致乱码。为此国内不少人对软件和安装程序进行汉化处理,结果费时费工且效果并不理想。然而 InstallShield 公司推出的国际化安装软件解决方案,为根本上解决这一难题提供了强有力的支持,使得我们不仅能够制作出中文内核中文版安装软件,还能同样开发出支持多种语言的国际版本安装软件。

制作国际版本安装软件首先要知道以下几点:

- 制作国际版本安装软件的目标是使软件产品本地化,以便世界上的一些地区能够接受和使用这一产品。
- 国际化的关键是资源和程序代码相互分离,地区和语言相互独立。
- 国际版本安装软件要求在设计上做到简单化和模块化
- 创建符合国际标准的安装软件意味着从一开始就要在安装软件制作规范中体现国际一体化的需求, InstallShield 使得这一开发长期有效,没有风险。
- 一些图标和位图在一些国家受到喜爱,在另一些国家可能成为禁忌。用 InstallShield 设计国际化安装软件可以解决这一文化冲突。
- 使用英语字符串通常要比其他语言简短,翻译后的字符串大约会增长 30-40%。这意味着有些存储区或显示区要调整大小。
- 要避免内部元素的位置和大小调整,因为它们经过翻译后可能会导致无法预料的变化。

为了获得 InstallShield 国际化安装软件制作支持,需要在安装 InstallShield for Windows Installer 基础上安装国际版本语言包程序。为此我们安装了 InstallShield Professional 2000 国际版本东方语言包,以便支持像中文这样的双字节字。如图 4-1、图 4-2 所示。



图 4-1 安装 InstallShield Professional 2000 国际版本东方语言包



图 4-2 选择安装 InstallShield Professional 2000 国际版本东方语言包的产品

有了 InstallShield2000 对国际化语言的支持，现在制作用于在全球发布的安装软件变得更加容易。我们既可以为某一地区制作单一语种的安装软件，也可以制作满足某些条件的多语种安装软件。总之，你可以使得安装软件的界面更加友好、使用更加简单。

提示：

如果不购买安装所需语言的国际版本语言包，可能无法实现对该语言安装软件的支持。在许多操作界面上，有关该语言的选择框或功能项将无法使用。有关 InstallShield2000 对国际化语言的支持，请查阅 installshield 相关网站 (<http://www.installshield.com/intliswi/lang/>)。

国际版本安装软件制作的关键在于定制与该语言代码相关的资源和安装文件。安装软件发布时，安装资源必须是与该语言一致的版本，包括：运行程序、帮助文件、图片、用户许可协议等。这些资源，应该在制作安装软件时准备好，并进行与该语言相关的必要设置。同时，还必须考虑到目标系统的语言环境，比如，目标操作系统对该语言的安装软件及所安装的应用程序是否支持，如果不支持则要考虑使用与语言环境无关的安装软件或需要安装的应用程序，InstallShield2000 在很多设置上支持语言无关选项 (Language Independent)。另外还可以在安装软件中准备多个语言版本的应用程序，并针对不同的目标系统的语言环境，根据条件选择合适的应用程序进行安装。

国际版本安装软件的制作包括以下几个主要方面：

一、指定安装软件要使用的语言版本（可以是一种，也可以是多种语言）

通过购买国际版本语言包就可以获得 InstallShield 支持的对应语言字符串表，这样使用该字符串表的各种安装信息和其他资源都能够自动翻译成该语言，即最终用户所看到的安装界面完全符合他们使用的本地语言。

通过向安装工程的安装语言属性中添加新的语种或使用安装向导中的安装语言设置可以指定安装软件要使用的语言版本。通常这些指定的语种中必须要有一种作为安装工程的缺省语言，系统初始化缺省语言是英语。虽然可以提供多种语言供安装软件发布时选用，但安装过程中只能使用一种选定的语言进行安装。

二、为每个支持安装的语言创建字符串表

字符串表为安装程序运行时使用的语言提供了可用的字符串资源，这些资源包括 Installshield 内建的和自定义的。使用国际版本语言包，可以获得 Installshield 已经翻译好的字符串资源。使用字符串表的好处是可以将安装程序设计代码和属性设置完全与不同语言的字符串分离开来，从而只需简单地更换一下语言代码、字符串标识，即可获得对不同语言的支持，无需修改任何程序代码。

在制作安装软件的任何过程细节中，我们必须使用字符串的标识 (ID)，该标识会自动引用字符串表中的值，所以完全不必担心不同语言版本对安装程序设计的影响。

存放字符串标识定义的地方是字符串表，它可以在工程视图找到。每个字符串标识都有一个唯一的值，你可以定义字符串的值和注释。每当在安装工程中包含一种语言，InstallShield 就为这种语言创建一个专用的字符串表，表中包括了所有现存的字符串标识。对于系统提供的字符串，一般都有对应的翻译好的值，只需将它们导入即可。当然有些值需要你自已翻译，还可以修改系统翻译好的值。

对于自己修改的或翻译的字符串表，可以导出保存，供自己或别人使用。要导出字符串表：

1. 在工程视图中的 String Tables 节点上按鼠标右键，在弹出菜单上选择 Export；
2. 找到所需复制字符串表的文本文件，点击 Open 按钮。

导出的字符串表是用制表符 (Tab 键) 分割的文本文件，同样格式的文件也可以导入字符串表。要导入字符串表：

1. 在工程视图中的 String Tables 节点上按鼠标右键，在弹出菜单上选择 Import；
2. 找到所需导入字符串表的文本文件，点击 Open 按钮。

字符串表编辑器此时会询问是否覆盖原来的字符串表。

三、为每个支持安装的语言修改用户对对话框（如果有必要的话）

当把新的语种增加到安装工程中时，InstallShield 提供翻译成该语种的标准对话框。但是，你仍然可以在对话框编辑器中修改这些对话框，以便更加适合所选择的语言。

要查看这些对话框，点击 Workspace 工具条中的 User Interface 按钮，展开树中所有节点，如图 4-3 所示。双击对话框名称，可以看到其所支持的语言。要更改对话框的界面，必须首先选择语言版本。



图 4-3 查看对话框所支持的语言

提示：

更改特定语言的对话框界面必须格外仔细。因为调整缺省语言对话框的大小或布局，其变化结果将被应用到所有其他特定语言的对话框中；但对特定语言的对话框的修改仅应用到该对话框的当前语言版本中，而对缺省语言版本的改动将不再影响这个改动过的特定语言版本。假如英语是缺省语言，你在这个英语版本的对话框中增加了文本框和按钮，于是英文和中文的对话框都共享了这一变化，比如都使用了共同的文本框长度和宽度。此时在中文版中，你可能发现文本框中文字符太大。如果你在中文版的对话框中进行修改，那么，中文版的对话框将不再与英文版共享同一属性，其所有资源将与英文版本分离。以后当你再更改对话框控件中的字符串标识时，需要在中文版和英文版中分别更改。如果你仅仅在英文版中调整文本框的大小，你则不需要同时维护两份资源，所以这也是最好的方法。

1、更改字符串

安装系统在对话框中使用的字符串来源于不同语言的字符串表。如果你为某个控件选择了使用本地化语言的字符串，显示的字符串值就是来源于该种语言的字符串表。一旦在该控件属性页中编辑字符串的值，你实际上编辑的是该字符串标识在其字符串表中的值。

2、更改文件资源

其他资源，如：位图、多媒体演示等都是以文件形式输入到安装打包中的。既然它们也是与不同的语言有关，所以需要为不同的语言提供不同版本的文件，并记住修改相应文件名称的值。

3、调整大小

由于同样的信息翻译成不同的语言可能造成文字长度上的不同。所以，在对话框中，一些控件的大小可能会因使用不同语言的字符串而有所差异。经常会出现像文本框需要因某些语言文字长度的增加而调整布局的情况。不过对特定语言对话框的调整不会影响到其他语言的对话框。

四、处理依赖于不同语种的组件

所有的新建组件缺省状态下都是与语种无关的。如果某个组件及其数据是针对特定语种的，则需要将这个组件标记为该特定语种。

以下步骤用于标记组件所依赖的语种：

1. 在 Setup Design 视图中选择组件，打开其属性页；
2. 点击 Languages 属性，查看属性页下面的帮助和语种列表；
3. 选择该组件适用的语种（可以多选）。

接下来就可以使用发布向导进行安装打包，向导会询问你需要包含哪些语种。如果你不选择要包含的语种，发布程序将与组件的特定语种无关，即同时打包那些依赖于和不依赖于特定语种的所有组件。但是，如果在向导的 Release Filtering 面板中选择语种，那么只有依赖于选中语种和与所有语种无关的组件会被打包。

例如，你可以创建一个支持多语种组件的安装工程，通过发布向导过滤不需要的语种组件，分别生成各个单一语种的专用安装软件。

另外，还有一种控制特定语种的组件安装到目标系统的方法是在组件的 Condition 属性中进行系统语言代码 SystemLanguageID 的判断。这样可以保证目标系统的语言环境满足组件语种的需要。

例如，对于日文的组件，在该组件的 Condition 属性加上 SystemLanguageID=1041，如果目标系统不是日文 Windows 操作系统，该组件则不会被安装。否则，盲目安装后可能导致日文无法正常显示。

五、选择发布安装软件的语言，以使用户能使用该语言完成安装

运行发布向导时，可以选择设定安装运行时的语言，以使用户能使用该语言完成安装。

可以为安装软件运行设定一种语言，也可以设定多种语言供用户选择。但后一种情况，一旦用户选定语言后，安装软件就以选定的那一种语言运行。

以上 5 个方面包含了国际化安装软件制作的关键步骤，具体的应用实例下一节将详细讨论。但是在制作一些双字节语言（中文、日文、韩文、希腊文等）的安装软件时，还要注意为这些语言包含代码页。如果在中文 Windows 系统中安装使用中文版安装软件当然不会有问题，然而当你无法预料用户的系统是否包含了所需的代码页时，必须考虑可能出现的问题，如乱码或文字无法显示等情况。

使用以下步骤可以安装代码页，以支持所需的语言：

对于 Windows NT 4.0

- 1、在光驱中插入 Windows NT 4.0 光盘；
- 2、进入 LanguagePack 目录；
- 3、右击需要安装的代码页，选择菜单命令 Install。

对于 Windows 2000

- 1、运行控制面板中的地区选项；

- 2、在 Language settings for the system 选项中选择要增加语言；
- 3、单击“OK”按钮安装该语言的代码页；
- 4、按照提示插入 Windows 2000 光盘完成安装。

4.2 国际版本安装软件制作实例

为了使读者易于理解和掌握国际版本安装软件制作过程和方法，下面给出了一个国际版本安装软件制作的实例。该实例使用了 Installshield 自带的一个名为 Othello 的现成例子。我们将为该应用装程序制作一个支持简体中文、日文和英文的国际版本安装程序。通过该实例，你将学会：

- 1、如何制作多语种的安装软件；
- 2、如何自动判断操作系统的语言环境，选择对应语言的安装组件；
- 3、如何在安装过程中让用户选择适合自己的语言环境；

下面将循序渐进，分步骤说明。

第一步：打开 Othello.ism

Othello.ism 是 Installshield 自带的一个工程模板。首先需要打开这个模板，它存放在 <InstallShield for Windows Installer 安装目录>\Samples\Othello\Program Files 中。可以用 Windows Explorer 打开。

第二步：为安装程序添加语种

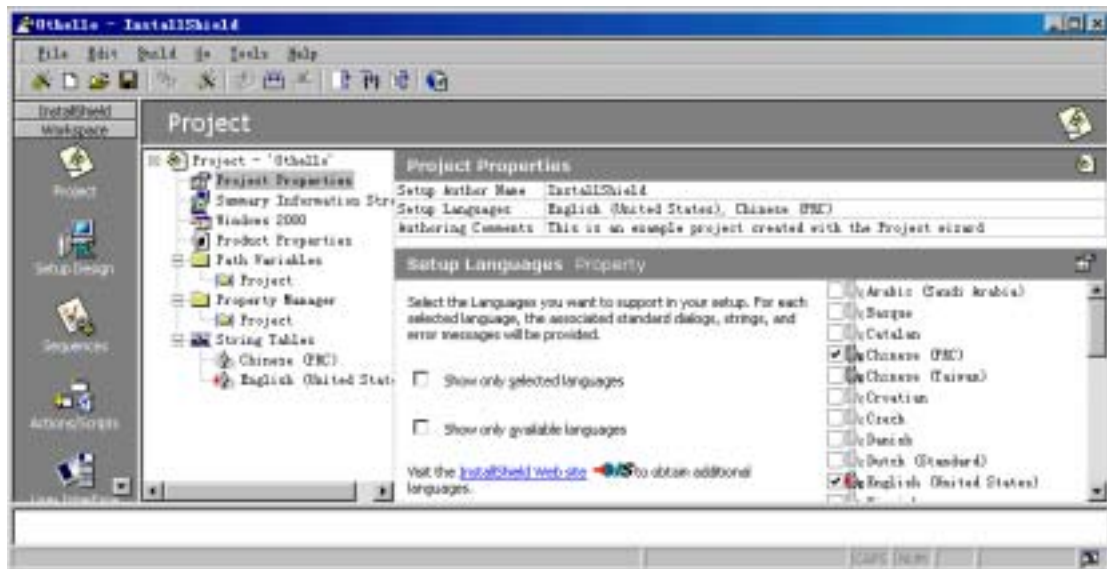


图 4-4 为安装程序添加语种

要想创建一个国际版本的安装程序，首先要选择合适的语种。要为你的安装程序添加语种，请进入 Project 视图下的 Project Properties。单击属性页中的 Setup Languages property，将显示可选择的语言类型。系统支持的语种后面会出现深灰色的图标。若是浅灰色的图标则表明该语言系统不支持。如果你要的语种未被列出，请查询 InstallShield 网站 <http://www.installshield.com/intliswi/lang/>。

为了完成此项功能，我们将为安装程序添加简体中文 (Chinese (PRC))。选中 appropriate language 左侧的复选框。与此同时，如图 4-4 所示，在右侧的 Project 视图中可以看到 Chinese (PRC) 和 English (United States)。

请注意，制作国际版本的安装软件时，Install Shield for Windows Installer 不会自动更新

已存在的安装工程中的字符串表。因此，你需要自己将某些语种导入列表。步骤如下：

1. 右击 Project 视图 String Tables 下的 Chinese (PRC)。从选项列表中选择 Import String Table;
2. 进入<InstallShield for Windows Installer 安装目录>\Languages，选中 CHS2052.txt 文件。
3. 将出现一个对话框，询问是否要覆盖当前字符串表的目录。单击“Yes All”。

第三步：自动生成字符串表的条目

下一步是为安装程序添加指定字符串。这里将要创建一个字符串表，它有三个不同属性：功能部件显示名称、快捷键显示名称和对快捷键的描述（此属性将作为下一步的一部分）。当在这些区域中输入文字时，将自动生成字符串表的对应条目。

1、功能部件显示名称

使用 Setup Design-Global 视图并单击名为 Default Program 的功能部件。双击 Display Name 属性，输入“程序文件”。如果在属性对话框的其他地方点击，你将发现已输入的文字前加上了{NEW_STRING1}前缀。单击屏幕底部的 String Table 页可查看缺省语种的完整字符串表，如图 4-5 所示。

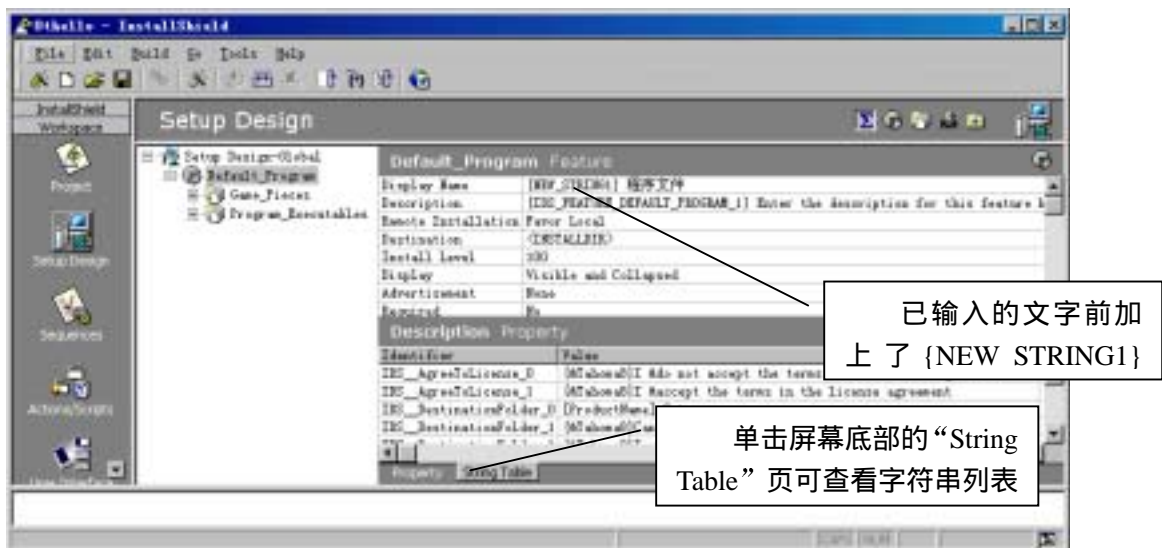


图 4-5 设置功能部件显示名称

2、快捷键显示名称

Program_Executables 组件含有 Othello.exe 文件的快捷键。选择该组件子节点下的 Shortcuts 图标。在 Shortcuts 视图中展开程序菜单的子目录并单击“Othello”快捷键。快捷键的属性将显示在右边。



图 4-6 设置快捷键显示名称

当前的 Display Name 属性为空。如果不输入一个显示名，快捷键的名称将与显示名一致。因此，既然快捷键名为“Othello”，那么显示名也为“Othello”。此名称未输入字符串表因此无法定位。但是如果为快捷键输入一个显示名，所有字符串表中都将出现一个新项目。双击 Display Name 属性并输入“奥赛罗”。如果在属性对话框的其他地方点击，你将发现输入的文字前加上了{NEW_STRING2}前缀，如图 4-6 所示。

第四步：自定义字符串表条目

由于多数字符串表将在安装程序中多处使用，在字符串表中存储这些字符串的每个实例显得没有必要。相反，你可以一次创建多次使用。

对于 Shortcut Description 属性，需要将新字符串直接输入字符串表。然后，将字符串关联到对快捷键的描述。

1. 首先单击 Shortcut Description 属性，然后单击屏幕底部的 String Table 标签。
2. 接着，为新字符串输入标识符。为与已创建的其他字符串保持连贯，将其命名为“NEW_STRING3”。
3. 在 Value 区域单击并输入下列语句：“奥赛罗快捷键的描述”
4. 最后右击新字符串表条目，从弹出菜单中选择 Select String。现在新字符串已作为 Shortcut Description 的值输入。如图 4-7 所示。



图 4-7 自定义字符串表条

第五步：创建指定语种的组件

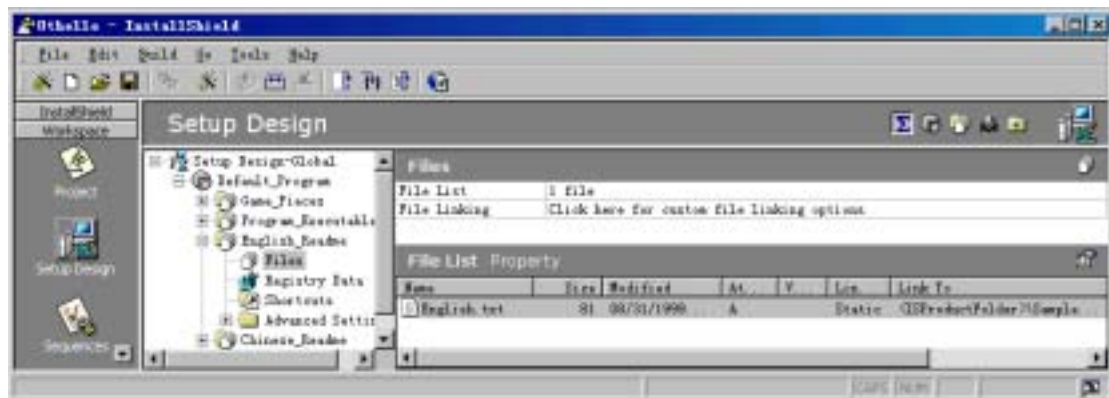


图 4-8 创建指定语种的组件

下一步要在安装程序中添加指定语种的文件和组件。比如，虽然许多程序文件不受语言

影响，但帮助文件和运行时刻字符串要依赖于该语言。为达到练习的目的，现在我们为安装程序添加三个新组件。每个组件包含一个带有所有支持语言版本的“Read me”文件，分别对应于英语、中文和日文。执行以下步骤添加组件：

1. 进入 Setup Design-Global 视图。
2. 右击名为 Default_Program 的功能部件并选择 New Component。
3. 输入“English_Readme”作为组件名。
4. 重复以上步骤两次，将这些组件分别命名为“Chinese_Readme”和“Japanese_Readme”。

准备好已经翻译过的 Readme 文件。假设这些文件位于<InstallShield for Windows Installer 安装目录>\Samples\Othello\Data Files\Readme。执行以下步骤向“English_Readme”组件中添加文件：

1. 选择“English_Readme”组件；
2. 展开该组件下的子目录并单击“Files shortcut”；
3. 右击文件列表并选择“Add”；
4. 导入“English.txt”文件并双击它。

最后结果如图 4-8 所示。对“Japanese”和“Chinese”组件重复以上步骤，分别添加“Japanese.txt”和“Chinese.txt”。

第六步：选择安装组件

既然已经创建了指定语种的组件，下一步需要让安装程序知道在那些情况下安装哪些组件。通过设置组件的 condition 属性，可以确定目标系统的缺省语言并安装所需文件。前述指定语种的组件均需如此设置。这样，一旦条件适合则安装此组件。

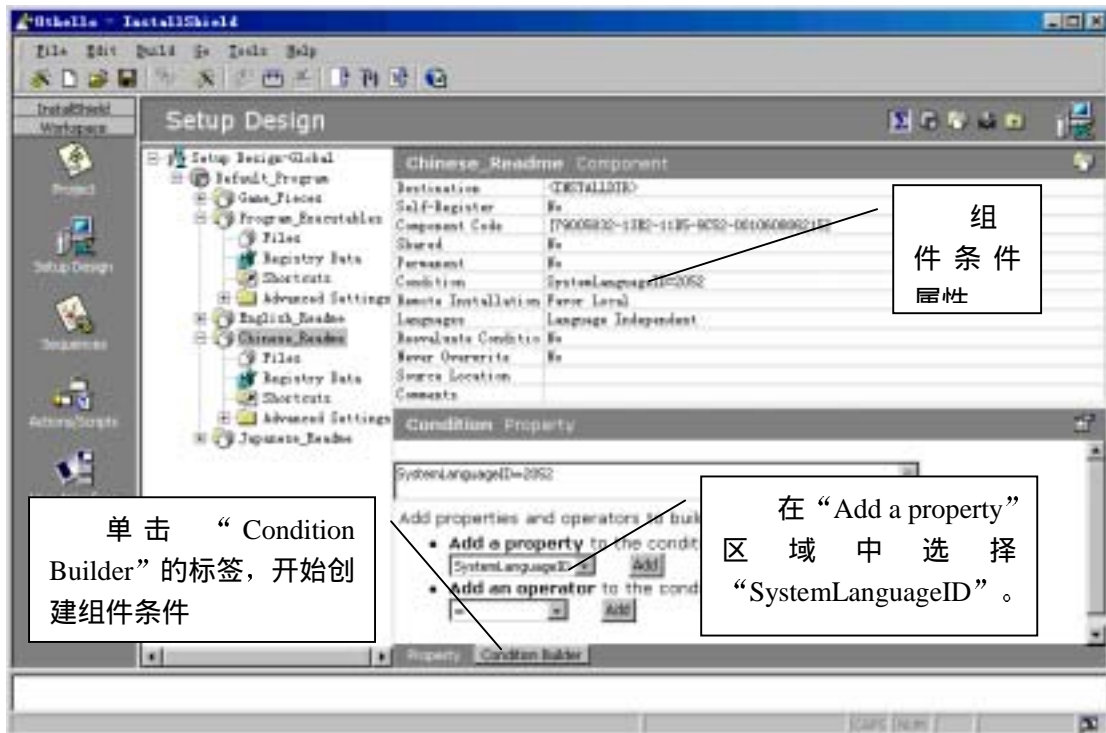


图 4-9 设置组件安装条件

点击 Chinese_Readme 组件并从右边的属性页中选择 Condition 属性。单击属性页下方的名为 Condition Builder 的标签，开始创建组件条件，如图 4-9 所示。

1. 在 Add a property 区域中选择 SystemLanguageID。按下“Add”将此属性添加到组件

条件中。

2. 在 Add an operator 区域中选择等号 (=)。按下“Add”将此运算符也添加到组件条件中。

在编辑对话框中，你将看到“SystemLanguageID =”，它表明你已经做好了选择。接下来，需要提供一个值，以便运行安装程序时进行检查。编辑中文组件时，请在等号后输入 2052。2052 是简体中文的标识符。由于这个组件只有在等式成立时才被安装（即：目标机器将把简体中文作为缺省语言），因此，如果某台机器使用的不是简体中文，该组件将不会安装。

按以上步骤为“Japanese_Readme”组件添加条件。不同的是，用 1041 来代替 2052，它是日语的标识符。

事实上，人们无法让安装软件和应用程序支持世界上所有的语种，也许你希望能够选择一种语言作为缺省语种。在本例中，缺省语种是英语。因此，使用“English_Readme”组件的条件将不同于日语和简体中文。

单击“English_Readme”组件，选择右边属性对话框中的 Condition 属性。单击属性单下方的 Condition Builder 标签，开始创建新的组件条件。最后的条件如下所示：

```
SystemLanguageID<>2052 AND SystemLanguageID<>1041
```

按此逻辑，如果目标机器使用的语言不是日语或简体中文，那么将会自动安装“English_Readme”组件。

第七步：本地化字符串

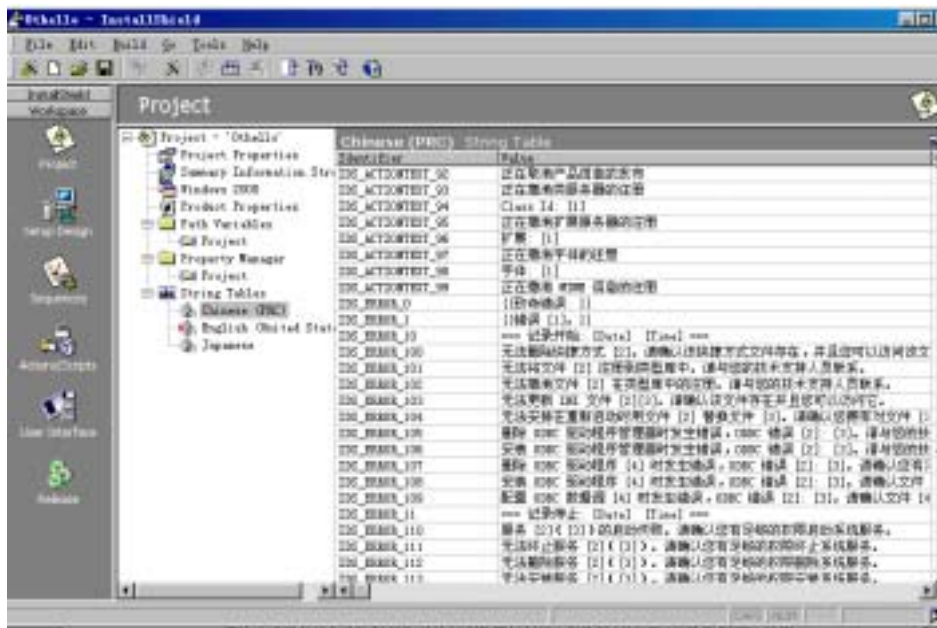


图 4-10 本地化字符串（简体中文）

在建立安装程序之前，必须定义作为功能部件名称、快捷键名称和对快捷键的描述而输入的英语字符串。幸运的是，Installshield 将为你做这些工作。你要做的只是在字符串表中输入正确的文字。

进入 Project 视图，在树型结构中选择 String Tables 下方的 Chinese (RPC)，如图 4-10 所示。此字符串表将显示在右边。如果将鼠标移到表的底部，你将发现通过向导建立的三个标识符。为相关联的字符串表输入以下文字：

Identifier	Value
------------	-------

NEW_STRING1	程序文件
NEW_STRING2	奥赛罗
NEW_STRING3	奥赛罗快捷键的描述

接着，单击日语的字符串表，操作同上。

理论上，你希望导出整个字符串表进行翻译。然而，在集成开发环境下编辑字符串表显得更轻而易举。Installshield 已经完成了大量的缺省字符串翻译工作。

第八步：编译国际化安装程序

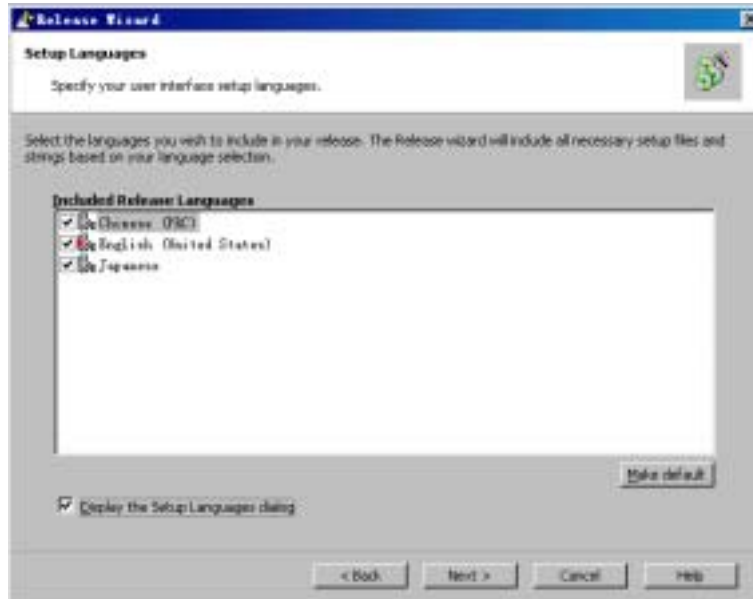


图 4-11 在 Setup Languages 面板上选择安装程序语言

现在安装程序已经完全国际化了，随时可以试运行。然而，在试运行之前，必须对它编译。单击工具栏中的“发布向导”按钮进行编译。在前两个界面中，确定编译标签和发布名称。除了 Setup Languages 面板之外，保持其他界面的缺省设置。此界面允许你选择安装程序语言。只有 Project properties 中指定的语言才会出现在候选列表中。你可以从英语，日语和简体中文中选择。如图 4-11 所示，选中每种语言后面的复选框即可。

确认你已选中了 Display the Language Selection dialog 选项。此对话框允许最终用户选择他们希望在运行安装程序中使用的语言。单击“Next”继续向导。

使用剩下的向导界面中的所有缺省设置。当完成向导时，单击 Build 编译你的.msi 文件。

第九步：运行安装程序

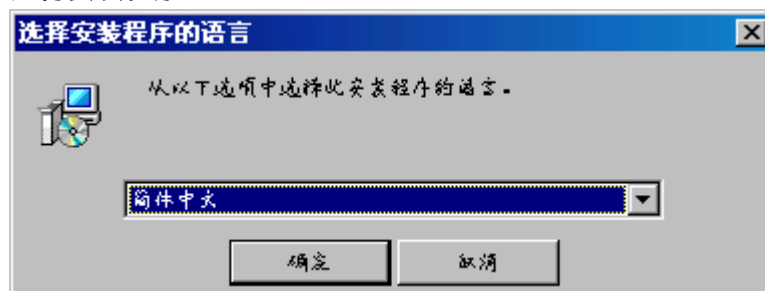



图 4-12 “选择安装程序的语言”对话框

单击工具条中的运行按钮,运行你的安装程序。出现的第一个对话框是“选择安装程序的语言”对话框,如图 4-12 所示。此对话框将永远以你设定的缺省语言显示。选择“简体中文”并单击“确定”。此后,每个对话框都将以简体中文显示。请注意,一旦以某种语言运行了一个安装程序,Windows Installer 将自动保存此信息,并总是以此语言运行安装程序,而不管你在 Language Selection 对话框中作过什么设置。

如果留心观察安装向导的话,你会注意到某些按钮的大小不太合适。你可以很容易地解决此问题,只需进入 Dialog editor 并重新调整大小即可。

如果进入 Custom Setup 界面(在简体中文中它被称为“自定义安装”),你将发现功能部件名称现在被改为“程序文件”。如图 4-13 所示,本地化的字符串表已包含在安装程序中了。

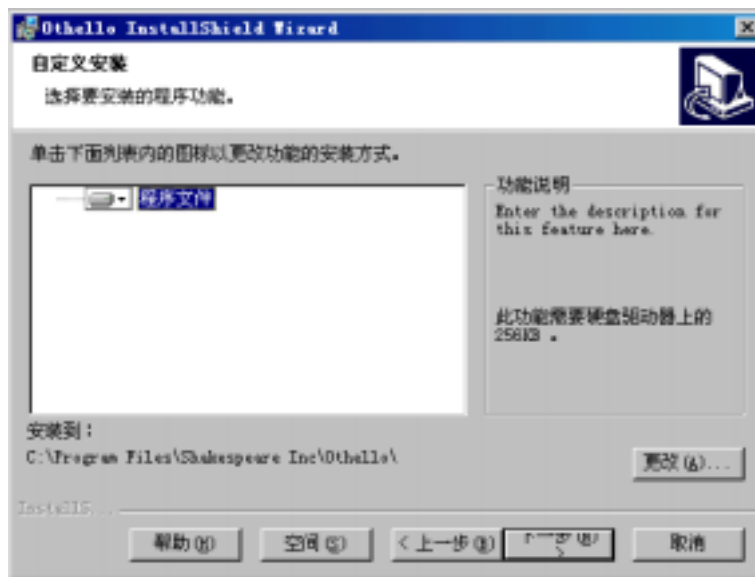


图 4-13 本地化的“自定义安装”界面

第十步：测试快捷键和 Readme 文件

打开“开始”菜单并选择“程序”。你将发现“Othello”的快捷键被显示为中文“奥赛罗”,如图 4-14 所示。在运行 Windows 2000 的机器上,你能够看到用简体中文显示的对快捷键的描述。

最后,进入“Othello”的安装目录。它应该位于<Program Files Folder>\Shakespeare Inc\Othello。你安装的“readme”文件被称为“Chinese_readme.txt”。



图 4-14 “开始”菜单“程序”中显示的快捷键为中文“奥赛罗”

现在，既然你已经了解了制作国际版本安装程序的全部步骤。你就有能力创建一个 Install Shield 所支持的任何语言的安装程序了。这里是要记住的主要步骤是：

- 为安装程序添加支持语言——需要在项目属性中为安装程序添加需要的语种。
- 创建指定语种的安装组件——如果应用程序拥有不同的语言版本，你将需要创建指定语种的组件，以便确定系统能够安装正确的版本。
- 翻译字符串表——虽然系统已经翻译了缺省的字符串表，你仍需添加一些自定义的字符串。这些字符串必须在你使用之前翻译好。
- 编译安装程序——不要忘记选择发布所用的语言。另外，还要包含进 Language Selection 对话框和 Setup.exe。所有这些选项都可通过 Release wizard 设置。

当然，这里只是一个整体介绍。InstallShield 提供了更多的好方法来制作国际版本安装程序，如使用基于语言的发布标记和数据筛选。想要进一步了解可以访问 InstallShield 网站或查阅有关资料。

第5章 数据库应用程序安装软件制作技术

利用前面所学的 InstallShield, 你已经能够开发出功能足够强大的安装程序。不过如果在完成安装以后, 你还需要用户自己去设置 ODBC 的话, 那将是你制作安装程序的一个最大败笔。更何况大多数用户对数据库设置一窍不通, 即使是很有经验的计算机维护人员, 在设置大型分布式系统的 ODBC 时 (如: Oracle, SQL Server 等), 常常也一筹莫展。所以制作能自动配置数据库连接的安装软件, 与其说是表现你非凡的开发技巧, 倒不如说是为了解决实际安装中的麻烦, 不然你会被用户的求助电话搞得焦头烂额。本章先介绍数据库应用程序及其安装要点, 然后讲解如何使用 ISPro2K 制作数据库应用程序安装软件的实例。

5.1 数据库应用程序及其安装要点

5.1.1 数据库应用程序的特点

一个数据库应用程序在逻辑上通常由两部分组成: 一是数据库访问链路, 二是用户界面, 这就是数据库应用程序的体系结构。

通常我们把实现数据访问链路的组件与实现用户界面的组件分开, 凡是数据访问组件最好放在数据模块上, 这样能够保证应用程序具有一致的用户界面。如果把设计好的数据模块和窗体加到对象库中, 在创建一个新的数据库应用程序时就不必什么都从头开始, 这样不但能够提高编程效率, 而且能够保证程序具有一致的风格。

数据库应用程序的体系结构取决于是使用本地数据库还是远程数据库, 取决于同时访问数据库的用户数以及数据库中需要存储哪些类型的信息。

在不同的数据库应用程序的结构体系中, 数据库的连接最常用的方式是通过 ODBC 数据源来实现的。ODBC 是开发与数据库有关的应用程序经常要用到的外部数据连接手段之一。利用 ODBC 能够编写客户应用程序使用来自许多不同数据源的宿主数据, 因此能够建立不依赖于数据的应用程序。ODBC 广泛地应用于多种不同的应用程序中, 它包括:

- 数据库存取实用程序——这些应用程序用来读、写和修改许多不同数据库中的数据。这种应用程序的一个较好的例子是 Visual Basic 的 VisData 程序。
- 纵向市场应用程序——在许多时候, 供应商都是用 ODBC 建立其应用程序, 以便应用程序能在多种不同的环境中使用, 从而使应用程序具有很大的灵活性和更好的市场销路。
- 广泛的应用程序——典型情况下, ODBC 用于建立客户机/服务器应用程序。然而有几个 ODBC 驱动程序可用于诸如 Access、dBASE、FoxPro 以及 Paradox 等等的本地数据库。驱动程序是用于本地访问还是远程访问, 这没有什么关系——函数调用方法是一样的。因此, 在建立从本地数据库应用程序方面, ODBC 功能是非常强大的。

ODBC 由以下四个组件构成:

- 应用程序——是象 Visual Basic 等语言编写的客户程序, 它调用 ODBC API 函数及 SQL 语句来和数据源通信。有时不直接用 API, 而是被另一个代码层隐式调用。例如: 使用带有 ODBC 的 Microsoft Jet 引擎就是这种情况。

- 驱动程序管理器——在逻辑数据源名和实际驱动程序之间起连接作用。根据数据源引用的驱动程序，驱动程序管理器装载并初始化该指定的 ODBC 驱动程序。例如，在打开名为 OracleDemo 的数据源时，驱动程序管理器在数据源入口查找数据源名并装载合适的 Oracle ODBC 驱动程序。
- 驱动程序——是一个含有供应商专门制作的 ODBC API 执行代码的动态链接库。驱动程序负责和数据源接口、处理 SQL 语句、返回结果对象的集合并执行错误检查。
- 数据源——数据源由多个部分组成，这些部分在 ODBC.INI 和系统注册表中定义为一个入口。

了解 ODBC 不仅有益于开发应用程序而且也关系到其对应的安装程序的制作。我们通过研究安装和配置的数据源，特别是 ODBC.INI 文件、ODBCINST.INI 文件及系统注册表，不难发现其中的参数设置与配置 ODBC 驱动程序可视化界面的联系，而这些界面通常是通过手工来进行设置的。

另外，对于使用 Delphi 的开发人员，Borland 公司提供了具有竞争力的 BDE 解决方案。BDE 的结构以及数据模块的使用使可伸缩性成为可能。不管是单层、两层还是多层，都可以把用户界面与数据访问链路分开其中，窗体主要用于实现用户界面，它的主要部件是数据控件。数据模块主要用于实现数据访问链路，换句话说，就是引入数据集。数据集与数据控件之间通过 TDataSource 组件连接。把用户界面与数据访问链路分开的好处是，当应用程序以后过渡到多层体系结构时，只有数据模块上的数据集组件需要修改，而用户界面不需要变动。

在企业级应用中，表与表之间的信息存在着比较复杂的关系，而且用户的数量不断增加，最好考虑多层的体系结构。与两层的应用程序相比。多层的应用程序多了一个中间层，中间层用于集中处理应用逻辑，这样，不同用途的客户程序可以使用相同的数据并且保证数据逻辑是一致的。同时，客户程序可以做得比较小巧，因为相当大的一部分工作由中间层去做了，这就是所谓的“瘦”客户。“瘦”客户更容易安装、配置和维护，因为它不需要包含数据库访问链路，不需要 ODBC 或 BDE。用多层的体系结构还有个好处是，可以把数据处理的任务分布在几个不同的系统中完成，用户无需直接和数据库打交道，与数据库的连接和配置是在数据库服务层完成的，所以在客户端应用程序安装时对数据库的连接和配置没有要求，但可能要求安装配置 COM Server、NT Services 等组件服务和控制模块。显然，层数越多，开发难度和费用就越大。

5.1.2 安装解决方案

在早期版本的 Installshield 或其他安装软件制作工具的开发中，通常需要了解 ODBC 等数据库接口驱动程序的复杂机制，手工编写脚本处理与注册表有关的问题。为了屏蔽数据库配置方面的复杂性，Installshield 公司一直在探索更好的解决方案。

在 Installshield5.x 中，Installshield 公司提供了 ODBC 模板用于 ODBC 3.0 的安装和数据源的设置。在 ISPro6.x 中，Installshield 公司采用了更简便的 ODBC 对象。通过 ODBC 对象向导可以将 ODBC 3.51 对象安装到 Windows 95/98/200 以及 Windows NT 4.0 (SP4) 或更高版本。最小安装 ODBC 包含安装 OLE DB。为了使 OLE DB 可以工作，目标计算机上必须安装 DCOM。ODBC 3.51 对象可以完成以下任务：

- 检查目标计算机操作系统是否满足对象运行要求。
- 检查用户是否有管理员权限。

- 安装 ODBC 3.51 对象运行文件。
- 配置驱动程序管理器、选择驱动程序、解释器以及数据源名称（使用 Odbc32.dll 输出的 SQL API 函数）。
- 对所有自注册文件进行注册。

在 ISWI 中，通常将 ODBC 数据源作为一个组件进行安装。简单的办法是使用组件向导（Component Wizard），这样驱动程序、转换器和数据源名（DSN）都将自动安装到位。

BDE 的解决方案与 ODBC 类似，我们将在下面的例子中详细解说，这里不再赘述。

5.2 基于 BDE 的 Oracle 数据库应用程序安装软件制作

下面我们剖析的实例是一个基于 BDE 的 Oracle 数据库应用程序安装软件，使用 ISPro6 制作。该数据库应用程序是一个装备管理系统，它是一个用 Delphi 开发的 C/S 构架的数据库应用程序。我们对安装程序的要求是：

- 完成客户端应用程序的安装。
- 自动安装配置 BDE，完成与 Oracle 数据库的连接设置。
- 安装完毕后，自动连接并初始化 Oracle 数据库。

由此可见，以上对安装软件制作的要求较高，基本上达到了最终用户对安装的傻瓜操作，实现了对客户端安装的零维护。

鉴于本实例的剖析重点在讨论数据库方面的安装技巧，所以我们不讨论如何创建安装工程、组织和关联安装文件。现在假设安装工程已经创建好，安装文件的设置也已完成。接下来我们就插入 DBE 对象。

5.2.1 插入 DBE 对象

BDE 5.1 对象是 Borland 数据库引擎。Borland 数据库引擎可以用作任意本地 SQL 驱动器，与大多数数据库服务器关联。BDE 5.1 对象可被安装到 Windows 95/98/2000 以及 WindowsNT4.0 或更高版本。同时需要本地硬盘上有 20M 的空间。

如图 5-1 所示在 Component 面板中按鼠标右键，选择 Insert Installshield Object...，出现如图 5-2 所示的 Installshield 对象库。在对象库中选择 BDE 对象，通过 BDE 向导就可以完成安装 BDE 的有关设置。

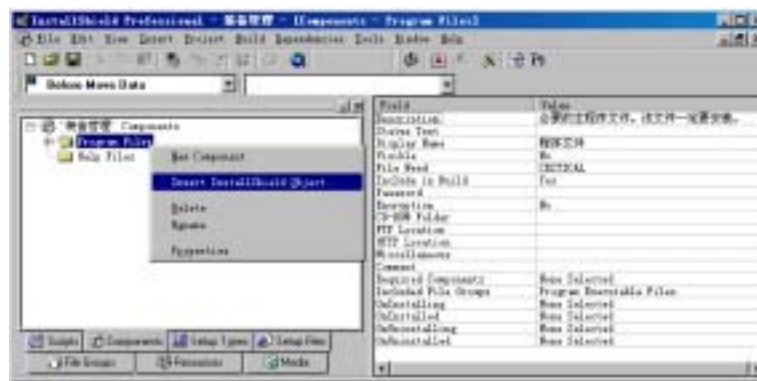


图 5-1 插入 Installshield 对象



图 5-2 在 Installshield 对象库中选择 BDE 对象

5.2.2 安装 BDE 的 Oracle 数据库驱动

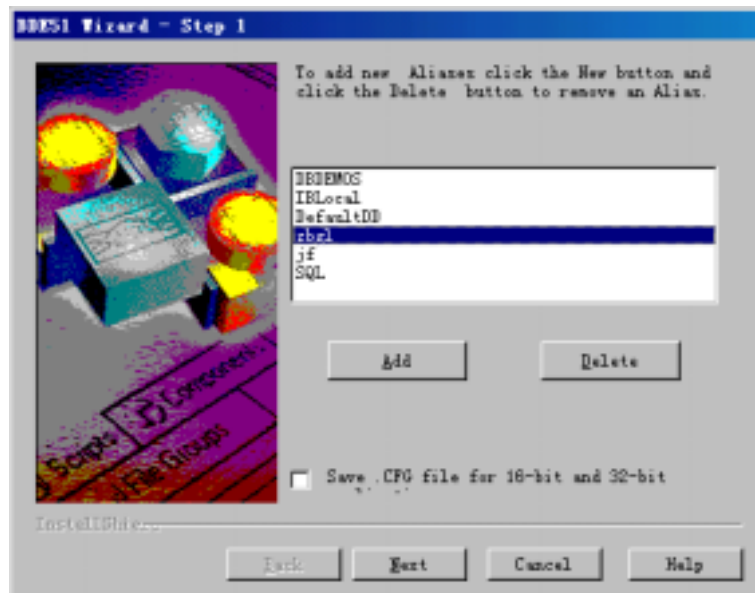


图 5-3 BDE 向导 step1 对话框

在 BDE 中，我们用到的数据库别名是 ZBGL，通常在该别名下需要设置 Oracle 数据库驱动，这些设置必须无误地安装到用户的机器上。在 BDE 向导 Step1 对话框中，如图 5-3 所示选择数据库别名 ZBGL，按 Next 按钮继续。如图 5-4 所示，BDE 向导 Step2 对话框用于设置驱动程序，这里我们选定驱动程序类型为 ORACLE，在 Optional Parameters 中可以进一步设置选项参数。这些设置随 BDE 一起安装。

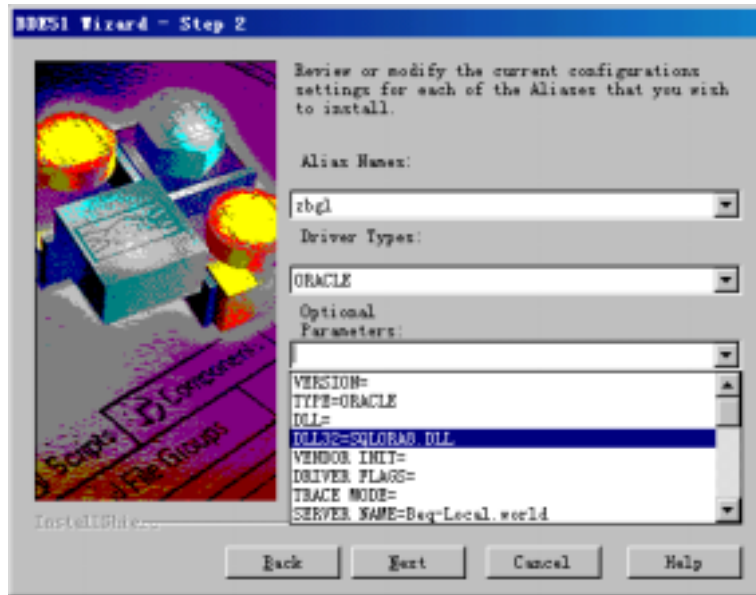


图 5-4 BDE 向导 step2 对话框

完成 BDE 向导之后，我们在 Components 面板中将看到如图 5-5 所示的“New BDE 5.1 1”对象已经位于“装备管理”部件的 Program Files 节点下。

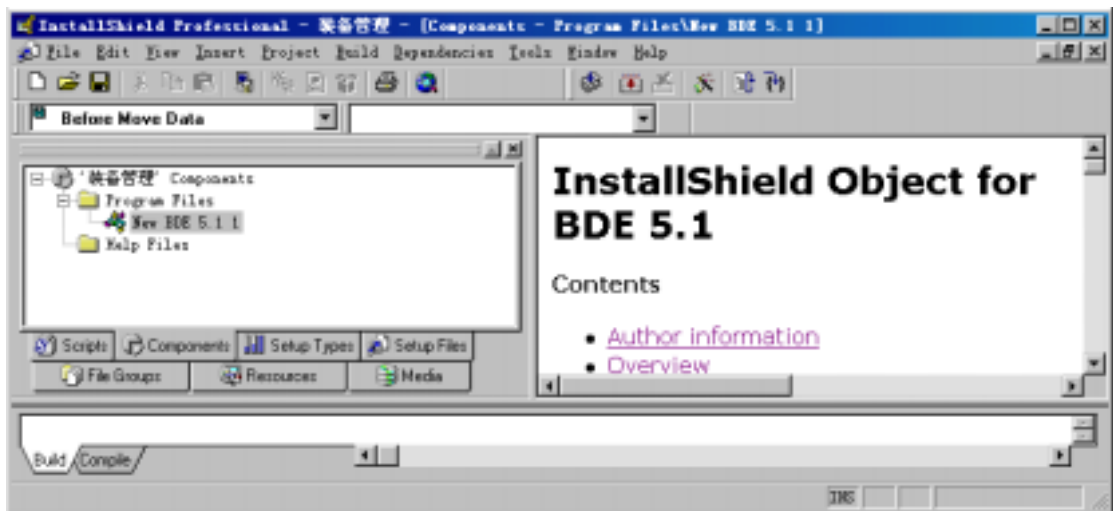


图 5-5 在 Components 面板中的“New BDE 5.1 1”对象

另外在 Resources 面板中，Installshield Objects 节点下也将出现“New BDE 5.1 1”对象，如图 5-6 所示。

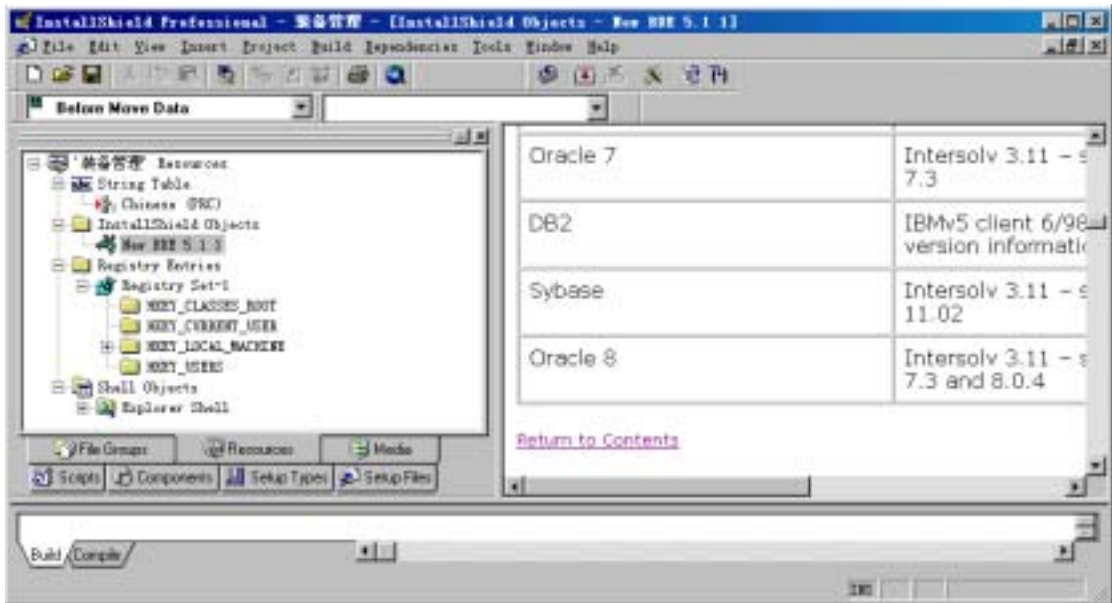


图 5-6 在 Resources 面板中的“New BDE 5.1 1”对象

如果你在插入 BDE 对象后的结果符合上面情况，则表示这部分工作已经顺利完成。这意味着安装软件运行后能够在目标系统是实现同样的 BDE 配置，保证数据源连接的正常进行。

5.2.3 调用数据库初始化程序

仅在客户端配置好 BDE 还不够，因为在运行数据库应用程序时还需要对数据库管理系统（RDBMS）进行初始化。为此我们设计了一个系统数据库初始化程序 inisql.exe，以便在安装完应用程序和配置好 BDE 后立即运行。该初始化程序的任务是：在 Oracle 中创建用户、表、存储过程、触发器等数据库对象。生成初始化数据。这样当用户首次运行应用程序时，不需要再进行任何复杂的设置维护工作，做到真正的“即插即用”。

为了在安装过程中调用数据库初始化程序，如图 5-7 所示我们在脚本的 OnMoved 事件中增加以下一行：

```
LaunchAppAndWait( SUPPORTDIR^"inisql.exe", "", WAIT);
```

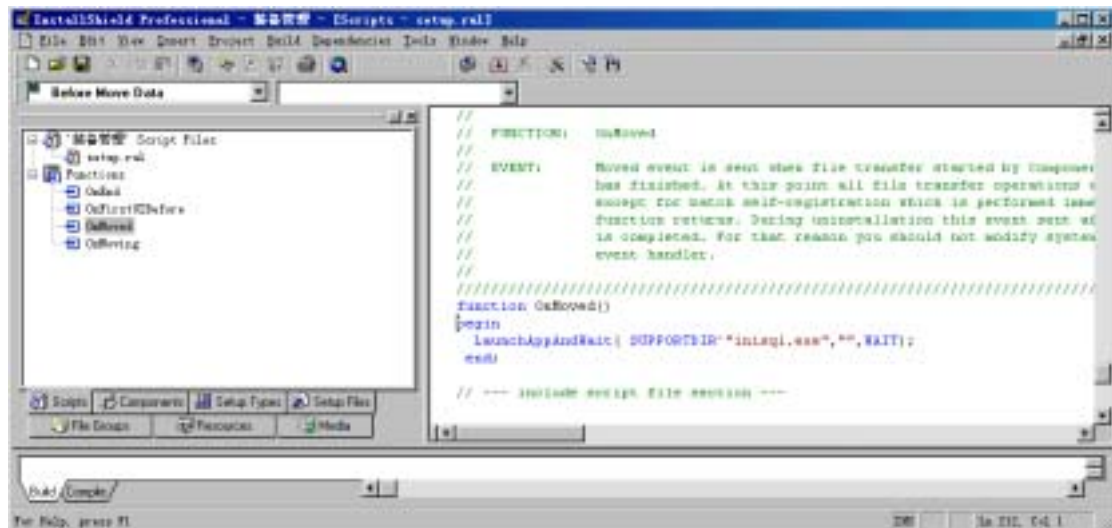


图 5-7 在 OnMoved 事件中调用数据库初始化程序 inisql.exe

其中, SUPPORTDIR 是指安装支持文件目录。为了能在该目录中找到 inisql.exe 文件, 我们将该文件拖放到 SetupFiles 面板的 Advanced Files\Disk1 中, 如图 5-8 所示。

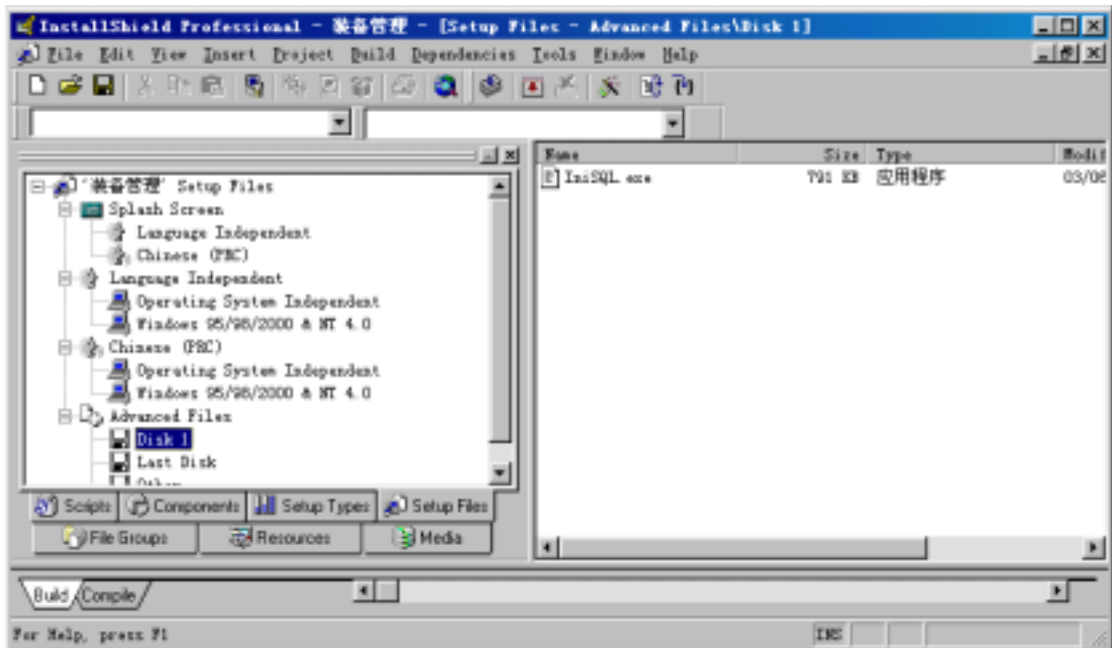


图 5-8 拖放到 SetupFiles 面板的 inisql.exe 文件

最后, 我们编译并测试该安装程序。安装程序在复制完安装文件后调用了数据库初始化程序, 运行结果如图 5-9 所示。如果该程序运行正常, 说明 BDE 安装正常且 Oracle 数据库的连接设置也正常。关于这个安装软件的安装工程源程序, 读者可以在随书附带的光盘上找到。



图 5-9 运行的数据库初始化程序 inisql.exe

5.3 基于 ODBC 的 Access 数据库应用程序安装软件制作

前面我们谈到的例子是用 ISPro6 实现的,但在 ISWI 中制作数据库应用程序安装软件也不复杂。下面的例子是用 ISWI 来制作一个多媒体资料库的安装程序,该程序使用的是 Access 数据库,所以如何在安装过程中配置 ODBC 数据库是安装程序制作的关键。

我们先用工程向导完成安装程序的大部分设计工作,如图 5-10、图 5-11 所示。



图 5-10 使用工程向导



图 5-11 主要的安装文件

运行完安装工程向导后,在 Setup Design 视图中如图所示右击鼠标,在弹出菜单中选择 Component Wizard...,如图 5-12 所示。

在 Component Wizard 的欢迎面板中选择 Select a type and define the component myself 选

项，可以进入如图 5-13 所示的对话框。在该对话框中出现的是单一类型的组件选项，选择 ODBC Resources 类型，我们可以进行 ODBC 相关的设置，但无法手工更改该组件。

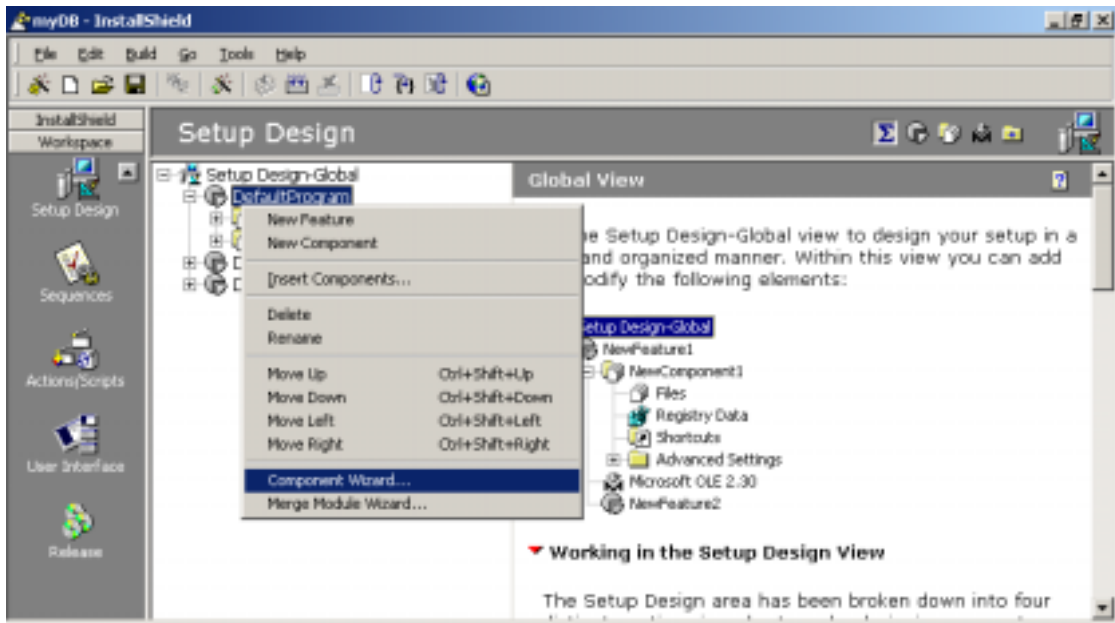


图 5-12 选择 Component Wizard

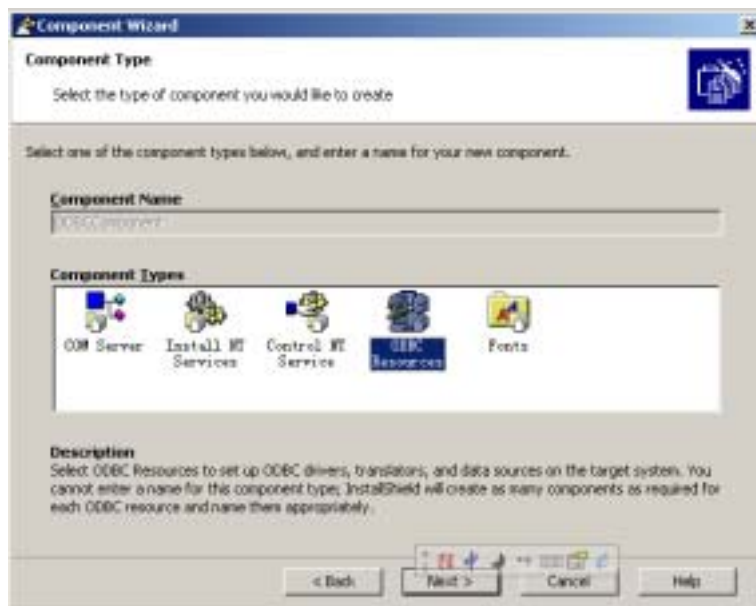


图 5-13 选择组件类型——ODBC

首先是设置 ODBC 驱动程序，如图 5-14 所示我们选择 MS Access 驱动程序。然后需要指定 ODBC 数据源，如图 5-15 所示我们指定 myDB 数据源，它指定了多媒体资料库 infoDB.mdb 的位置。这两步是关键，一定要设置正确。在结束向导之后，我们在 Setup Design 视图中将看到设置成功的 ODBC 源，如图 5-16 所示。至此，ODBC 类型组件的设置工作全部完成。

编译该安装工程。然后在另一用户的电脑上（该电脑未曾设置过 myDB 数据源）测试制作好的安装程序，多媒体资料库程序运行正常。在该用户电脑的控制面板中查看 ODBC

设置，完全符合要求。

通过以上的剖析，可以看出 ISPro6 和 ISWI 在处理数据库应用程序的安装软件制作上各有千秋。现在是你发挥想像力大显身手的时候了。

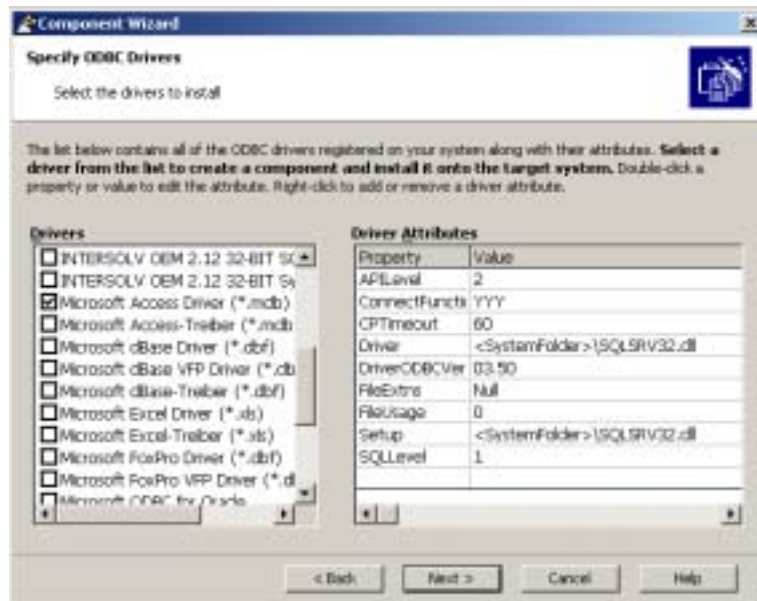


图 5-14 指定 ODBC 驱动程序

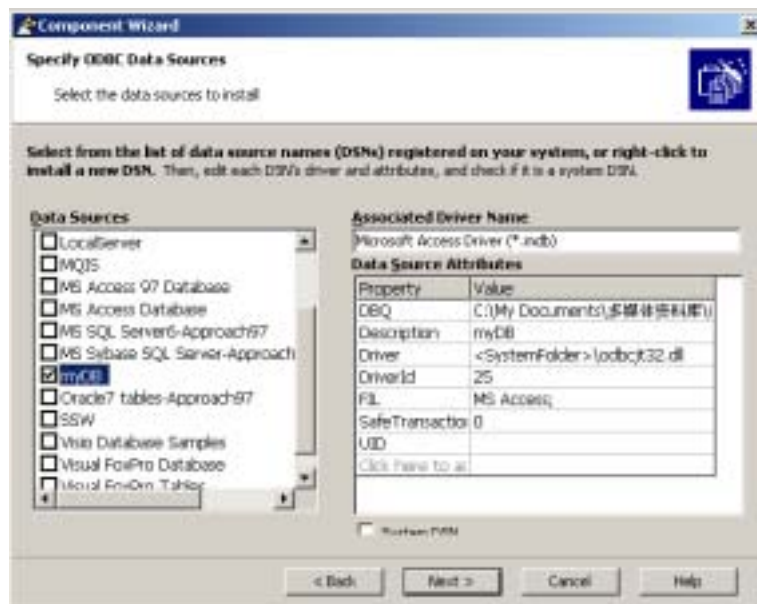


图 5-15 指定 ODBC 数据源

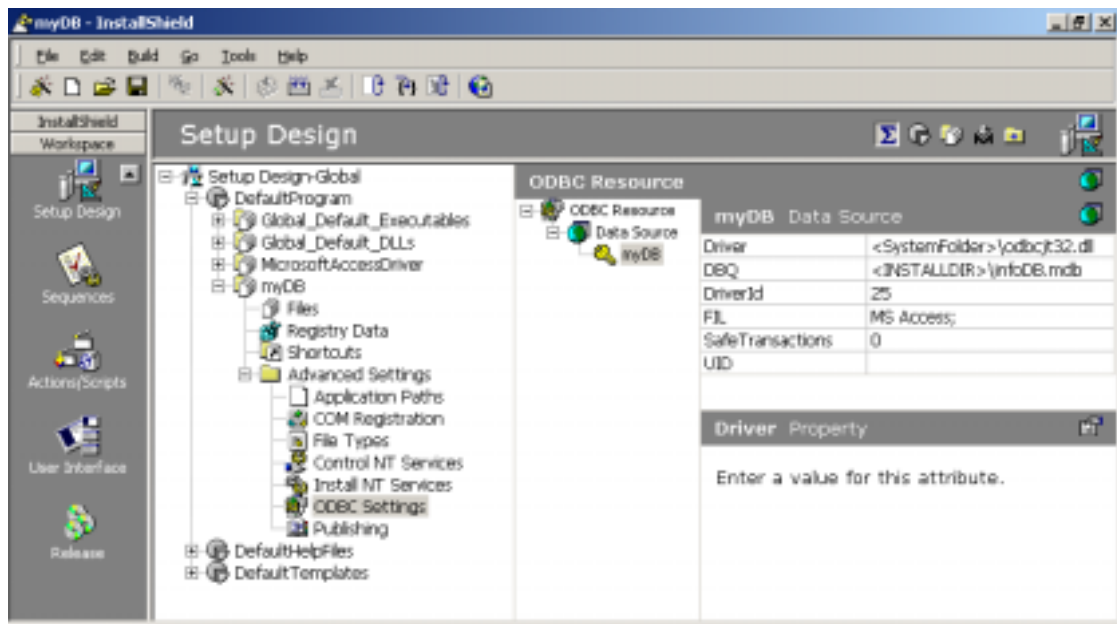


图 5-16 设置成功的 ODBC 源