第 9 章　应用程序的调试与发布

教学提示：上一章我们学习了用户对象和用户事件的概念及用户对象的创建和基本操作方法。本章我们将学习应用程序的调试方法与操作步骤以及应用程序的发布与发布磁盘的制作方法。

教学目标：通过本章学习，我们要掌握应用程序的调试方法与操作；掌握应用程序的错误处理方法；掌握应用程序的发布方法；掌握编译应用程序的方法；掌握发布磁盘的制作方法。

9.1　调试器介绍

在介绍调试器之前，首先要对程序的调试有所了解。

没有任何一个人能够一次写出一个没有错误的程序，对于应用程序开发来说，整个的开发的过程也是一个不断排除错误的过程。应用程序经过设计、开发、调试和试运行，排除了明显的与潜在的错误并达到用户所要求的功能后，把应用程序编译与可执行文件，这样用户就可以脱离开发环境运行应用程序了。可以说，调试是应用程序开发过程中必不可少的环节，而且往往调试所花费的时间可能是程序设计的好几倍。所以调试效率的高低也是衡量一个软件开发平台优劣的重要指标。

为了保证应用程序功能的完整性、正确性，任何一个应用程序在其投入使用之前都要进行测试。通过测试，可以发现应用程序中的各种错误，进而进行修正与弥补，使之完善以保证应用程序的健壮性。

应用程序完整的测试通常包括过程如下。

(1) 单元测试：测试应用程序的每一个逻辑模块，使每个模块都能实现各自的功能。

(2) 系统测试：把各种测试完毕的模块集中起来，然后测试整个应用程序的功能是否达到了预期的目标，实现既定的功能。

(3) 集成测试:测试应用程序能否与系统中的其他部件协调工作。这些部件包括硬件(例如计算机、网络、交换机等)和软件(例如数据库、操作系统等)。

(4) 负荷测试：测试应用程序在大负荷的情况下，能否按预定的要求处理数据，完成用户交给的任务。

(5) 用户测试：把应用程序交给用户去测试，在实际使用中去检查应用程序能否满足用户的要求。

应用程序的调试要通过调试器完成。PowerBuilder 提供的调试器(Debug Painter)是 PowerBuilder 内置的调试工具，是查找、定位、排除程序错误的功能强大的集成调试环境工具。利用调试器工具，开发人员可以即时查看应用程序执行到某条语句时变量及对象的属性值并进行修改。不仅可以完成普通的单步跟踪、断点设置、变量查看，而且可以实现

设置断点、改变下一条要执行的语句等功能。

9.1.1　打开调试器

方法 1：选择【Run】|【Debug Target】命令，其中“Target”是当前处于活动状态的目标的名称；或者选择【Select and debug】按钮。

方法 2：在 PowerBar 工具条上单击【Debug Target】按钮或者【Select and debug】按钮。

以上两种方法都可以进入调试器画板。

9.1.2　调试器界面简介

调试器画板界面，如图 9.1 所示。

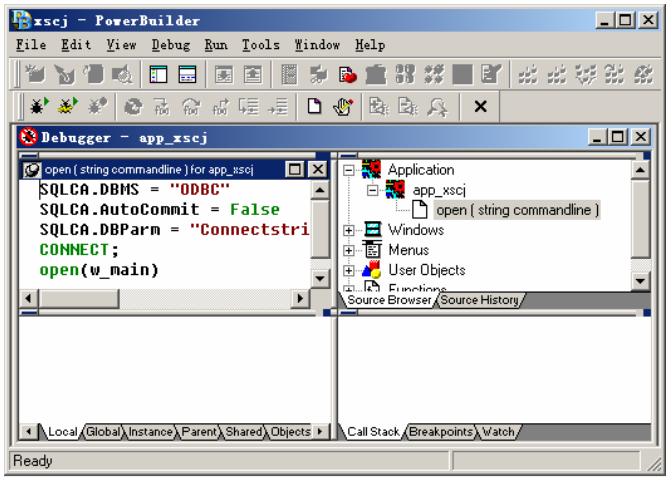

图 9.1　调试器画板

Debug 画板由 4 个窗口构成。

1. 脚本显示视窗

在 Debug 画板的左上角是 Source(脚本显示视窗)。Source 视窗中显示的是用户选中的事件的脚本，用户如果在 Source Browser(对象显示视窗)中选择一个对象的某个事件，那么就会在 Source 视窗中显示出它对应的脚本。如图 9.1 所示在 Source 视窗中显示的是 xscj 数据库的 Open 事件的脚本。

但需要注意的是，在 Source 视窗中只能浏览、设置断点，但不能修改脚本。

2. 对象显示视窗

在 Debug 画板的右上角是脚本显示视窗(Source Browser 视窗和 Source History 视窗)。它们用来显示应用程序中的对象。

在 PowerBuilder 中能够编写脚本的对象共有以下 5 种：

- Application
- Windows

- Menus
- User Objects
- Functions

在这个视窗中有两个选项，一个是 Source Browser，另一个是 Source History。其中 Source Browser 是显示当前打开的事件，Source History 是显示所有曾经被打开过的事件。

在 Source Browser 视窗中，单击对象左边的“＋”号或双击对象名便可展开对象，在展开的对象名下，列出当前应用程序中所有这种对象类型的实例对象。如图 9.2 所示，在 Application 对象下只有一个 app_xscj 实例对象，再展开这个实例对象，则可看到在这个实例对象下只有一个 Open 事件。

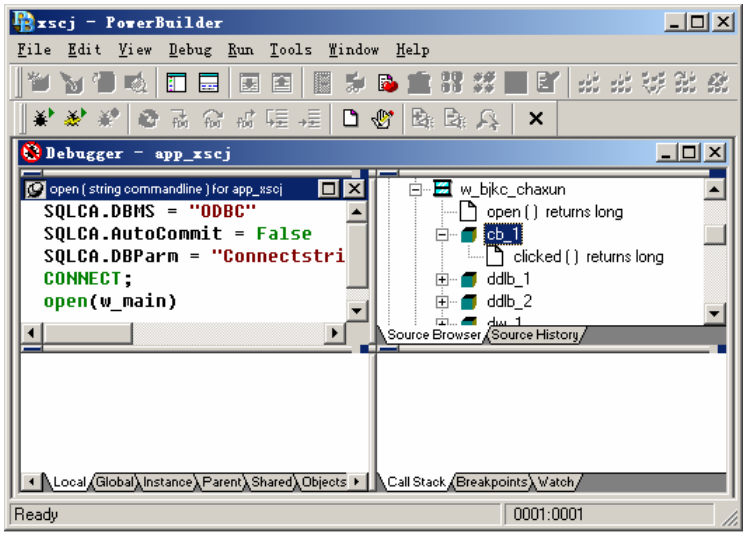

图 9.2　显示对象及事件

如果展开窗口对象，那么在窗口对象的实例下不仅列出当前窗口对象所有已编写了脚本的事件，还会列出当前窗口对象中的所有控件。展开这些控件，可看到在控件下面列出了当前控件所有已编写了脚本的事件。

3. 变量显示视窗

在 Debug 画板的左下角是变量显示视窗，用以显示变量信息。包括 Local、Global、Instance、Parent、Shared 和 Objects In Memory 等几个视窗。不同的视窗显示不同类型变量的变量信息。一般来说，这些视窗只显示对应变量类型的变量信息，如 Local 视窗只显示 Local 类型变量的信息。

4. 信息显示视窗

在 Debug 画板的右下角是信息显示视窗，用来显示调试过程中的信息。主要包括 Call Stack、Breakpoints 和 Watch 3 个视窗。

1) Call Stack 视窗

调用 Call Stack 视窗，即堆栈视窗，显示一个函数和事件的调用顺序的列表。通过此列表，可以知道当前正在执行哪一个事件或函数，并且这个事件或函数是由另外哪一个事

件或函数引起的。实际上也就是了解整个调用的层次关系，从而对整个程序的执行进行深入了解，那么在调试过程中可以更准确、方便地找出在执行中出现的问题。

2) Breakpoints 视窗

Breakpoints 视窗即断点视窗，主要作用是用来显示应用程序中当前已经设置的所有断点。由每个断点旁边的指示器显示断点的状态，如为实心则为活动断点，空心则为非活动断点。我们还可以通过双击断点进行断点的编辑。

3) Watch 视窗

Watch 视窗实际上可以看做是 Variables 视窗的一个变形，显示了需要经常查看的变量列表，并且可以定义表达式，在 Watch 视窗中查看该表达式的值。

5. Debug 画板的工具栏

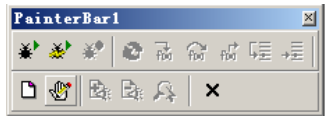

图 9.3　Debug 画板工具栏

如图 9.3 所示，Debug 画板的工具栏共有 15 个按钮，它们的功能如下。

- Start：开始在调试状态执行应用程序。它和【Run】按钮的区别在于【Run】按钮执行了应用程序，而【Start】按钮在执行应用程序的同时也打开了调试器。
- Start Remote：调试远程应用程序组件。
- Stop Debugging：停止当前程序的调试与执行，返回到 Debug 画板。
- Continue：如果在调试时在程序中设置了断点，那么在调试状态下执行应用程序到该断点时，会在断点处暂时停止执行。如果用户想继续执行应用程序，只要单击此按钮即可。
- Step In：单步执行或单步跟踪。也就是每按一下这个按钮，程序就执行一条指令或一条语句。要注意，如果当前语句是用户自定义的函数或调用的事件名，按此按钮会进入函数或事件内部来单步执行，返回的结果是一条语句的结果。
- Step Over：与 Step In 不同的是，当用户执行到用户自定义的函数或调用的事件时，单击这个按钮，系统会把函数或事件的所有代码当作一条语句来处理，而不会进入函数或事件的内容进行处理。返回的结果是整个函数或事件的结果，而不是一条语句的结果。
- Step Out：当用户不小心已经进入了一个复杂的函数或事件的内部，但又不想继续详细跟踪函数或事件内部具体操作的情况，可按此按钮，这时系统会把剩下的语句当作一条语句执行，跳过复杂的单步跟踪。
- Run to Cursor：执行到光标处，即从断点开始处执行，一直到当前光标所在的语句。
- Set Next Statement：设置下一条要执行的语句，这样可以灵活地进行程序执行过程的控制，不必按照原有的流程规定的顺序。主要作用是调试某些不经常出现的条件。
- Select Script：选择要在 Source 视窗中显示的程序代码。用于显示需要显示的事件或函数的代码。
- Edit Stop：编辑当前应用程序中的断点。

- Add Watch：增加一个变量到 Watch 视窗。
- Remove Watch：从 Watch 视窗中把一个变量删除。
- Quick Watch：快速在 Watch 视窗中增加一个观察变量。
- Close：关闭 Debug 画板。

9.1.3　设置调试器功能界面

前面我们介绍调试器功能界面时，都是使用调试器本身提供的默认的界面。在有些情况下，需要设置调试器的功能界面，使之更符合使用习惯。即把一些常用的视窗放在一定的位置，把一些不用的视窗去掉。通过设置调试器的功能界面，便可实现这个目的并提高工作效率。

修改调试功能界面的具体操作如下。

(1) 如图 9.1 所示，一般使用的都是默认的界面。首先在默认界面的基础上选择需要的视窗，去掉不用的视窗并进行重新排列，如图 9.4 所示。

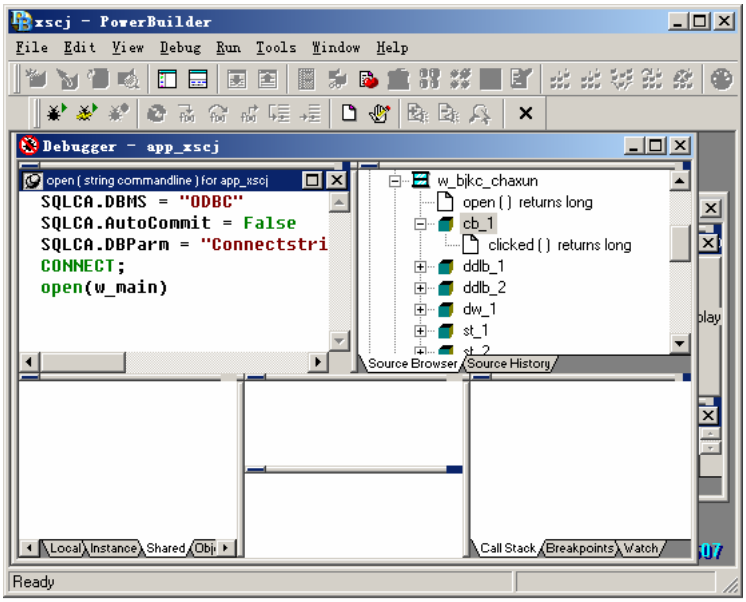

图 9.4　重新设定界面

(2) 如图 9.5 所示，选择【View】|【Layouts】|【Manage】命令，弹出如图 9.6 所示对话框。

(3) 在【Layout】对话框中单击右上方【New Layout】按钮，在操作区域中输入新界面的名称“aaa”，然后关闭对话框。这时，已经设置了一个新的功能界面“aaa”，如图 9.7 所示。

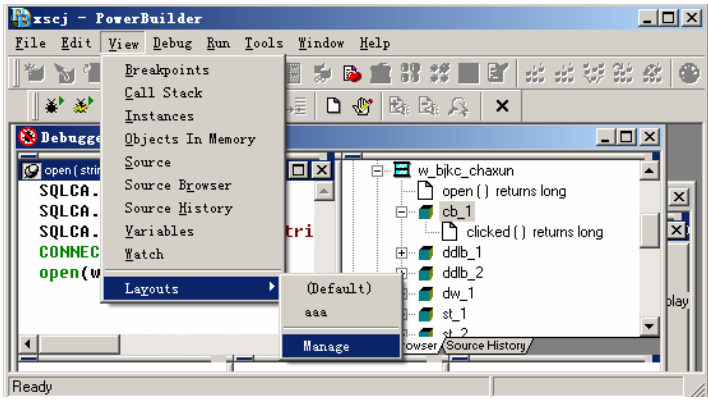

图 9.5　界面设置

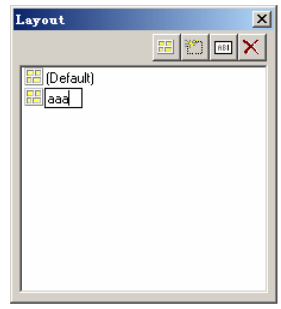

图 9.6　新建界面

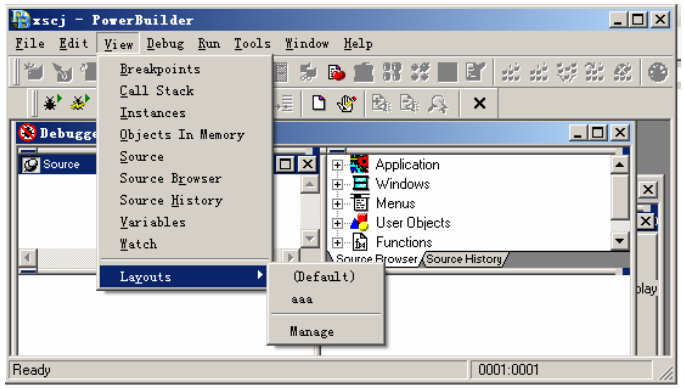

图 9.7　设置界面结果

9.2　脚本调试方法

9.2.1　打开要调试的脚本

用前面介绍的方法在 Source Browser 视窗中展开要调试的对象，双击要调试的事件或函数，在左上方的 Source 视窗中会显示出要调试事件或函数的脚本，如图 9.1 所示。

9.2.2　设置断点

可以根据需要设置不同类型的断点。

1. 设置简单的位置断点

所谓简单的位置断点，也就是没有触发条件的位置断点。特点是一旦设置了位置断点，当调试画板关闭后，这些断点位置还保存在系统中，除非标记了断点的行已经不存在了。

设置方法是双击 Source 视窗中某个可执行代码行的任意位置，此时系统就会在此行代码处设置一个断点，在该代码行前可以看到一个红点，如图 9.8 所示。如果再次双击此代码行，则会删除该断点。也可以通过在该断点上右击，在弹出的快捷菜单上选择【Disable

Breakpoint】菜单项(如图 9.9 所示)的方法，在 Source 视窗中禁止某个断点(但并不删除该断点)，如图 9.10 所示。

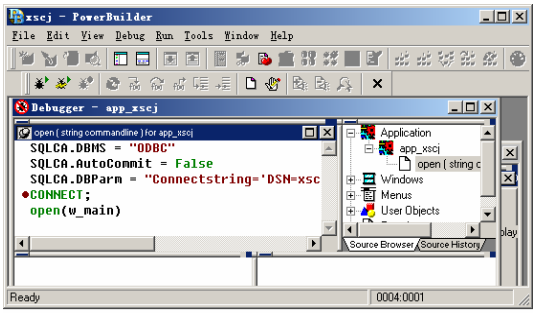

图 9.8　设置断点

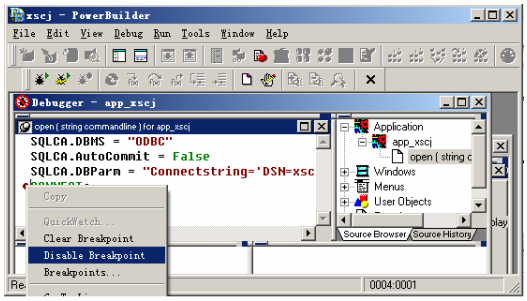

图 9.9　禁止断点

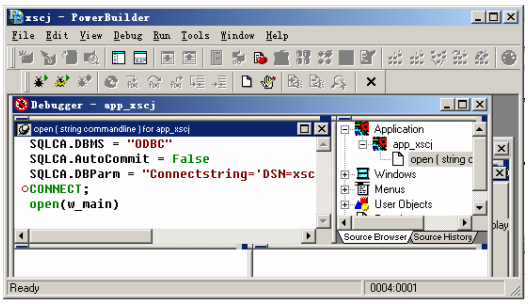

图 9.10　断点禁止显示结果

通过这种方法设置的断点非常简单，它们不需要触发条件，只要程序运行到此处，程序就会停止运行。

2. 设置复杂的位置断点

复杂的位置断点具有触发条件。当程序运行到设置断点的代码行时，如果满足触发条件，则程序会停止；否则程序将继续执行。设置具有触发条件的断点，选择【Edit】|【Breakpoints】菜单项，在弹出的对话框中进行设置，如图 9.11 所示。

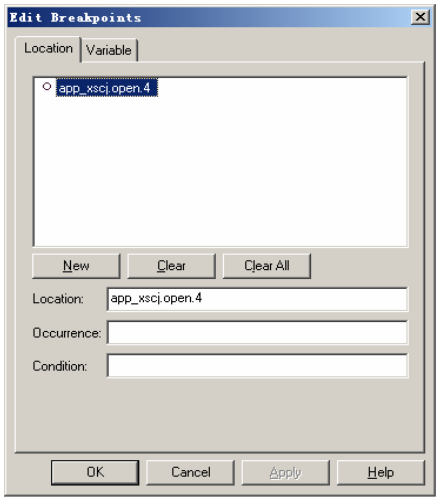

图 9.11　设置复杂的位置断点

在对话框中有两个标签页：一个是【Location】标签页，另一个是【Variable】标签页。分别用来设置位置断点和变量断点。

在【Location】标签页中，上面区域用来显示当前应用程序中设置的所有断点，可以对断点进行新建、清除，清除所有断点。以下显示的是设置断点的属性。

(1) Location：用来设置断点位置。

(2) Occurrence：用来设置该断点所在代码行执行的次数。这里要注意，一定要输入一个正整数。当应用程序执行该断点所在代码行到达设定次数后，应用程序在该代码行进入

中断模式，进入调试状态。通过这种设置，我们可以灵活地控制该代码行的执行次数，以便于调试。

(3) Condition：用来确定一个布尔表达式。当执行到这个断点所在的代码行时，系统将计算该表达式的值。如果该表达式的值为 True，则程序停止执行进入调试状态，否则程序将继续执行。要注意，Occurrence 和 Condition 是“与”的关系，也就是只有这两个条件都满足时，应用程序才会在该断点处停止。

3. 设置变量断点

所谓变量断点，就是指当设定的某个变量的内容发生变化时，便可作为断点的触发条件来中止应用程序的执行。设置变量断点要在图 9.12 中进行。单击【New】按钮，然后在下面的 Variable 编辑框中输入要引用的变量名，单击【OK】按钮后就可设置一个变量断点。但要注意，如果引用的变量名不存在，系统就会提示出错。

4. 编辑设置好的断点

设置好的断点也可以在图 9.11 和图 9.12 中进行编辑。

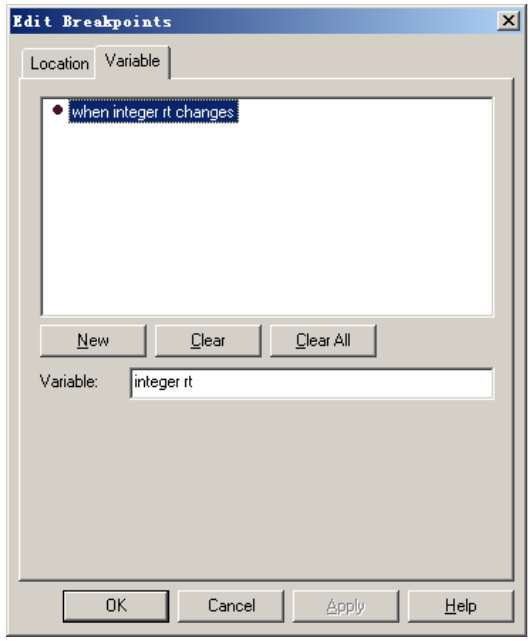

图 9.12　变量断点设置

9.2.3　调试运行脚本

设置好了断点后，就可以运行脚本进行调试。单击调试器画板工具栏上的【Start】按钮，应用程序就可以调试运行了。当程序执行到第一个断点时，系统就进入中断模式。此时开发人员可以根据自己的需要，既可以让程序运行到下一个断点，也可以单步运行，逐行跟踪。下面介绍几种跟踪方法。

1. 运行到下一个断点

调试过程中，如果某断点处没有问题或要程序执行到下一断点，都可以单击调试器画板工具栏上的【Continue】按钮或直接按快捷键【Ctrl+C】。

2. 单步执行

在前面已经介绍了在调试器画板工具栏上的【Step IN】、【Step Over】和【Step Out】按钮 3 种单步执行的方式。在需要单步执行进行跟踪的时候，就可以根据具体需要选择单步执行的方式。

3. 执行到光标处

可以从断点处，继续执行到光标当前所在位置。

4. Set Next Statement 调试方法

这种调试方法的特点是：可以自由地指定将要执行的下一条语句的位置；并且应用程序执行的方式是直接跳到这条语句上的，它不会去执行原来的执行点与新确定的执行点之间的任何一条语句。我们可以通过鼠标或光标定位到要执行的下一条语句，然后单击【Set Next Statement】按钮即可。

9.2.4　添加观察变量

若要向观察变量视窗中添加观察变量，应首先在变量视窗中找到要查看其取值的变量。然后将该变量从变量视窗中拖到观察变量视窗中，或右击该变量从快捷菜单中选择【Add Watch】菜单项进行添加。

还可以在观察变量视窗中右击，从弹出的快捷菜单上选择【Insert】菜单项，在出现的如图 9.13 所示对话框中输入所需表达式。

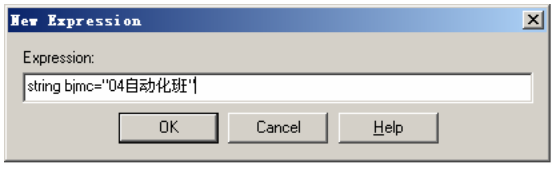

图 9.13　观察变量设置

对于添加到观察变量视窗中的变量，随时都可以将之删除。只要选中要删除的变量，右击并从快捷菜单中选择【Clear】菜单项来删除该变量；或选择【Clear All】菜单项将所有变量删除。

9.3　错 误 处 理

任何应用程序都不可能设计得非常完美，在运行使用过程中都会出现这样或那样的一些错误。当然出现的错误也有所不同，可能是程序本身的问题，也可能是用户在操作时动

作不规范造成的。但不管是什么样的错误，对于用户来说都是不希望的，但对于应用程序来说却也是不可避免的。我们在设计应用程序时就应该提前考虑到这个问题，而且应该在应用程序设计时采取一定的措施来避免或弥补。那么，对于应用程序来说，应该能够处理用户的一些非法操作和异常情况，所以在应用程序中必须有错误处理程序。当程序出现错误时，应用程序本身能够自己处理而不会导致非法关闭甚至丢失数据。这种功能或考虑也可称为“容错”。错误处理程序工作的过程如图 9.14 所示。当程序出错时程序判断错误类型，然后调用相应的错误处理程序进行处理，处理完毕后返回程序继续执行下面的代码。

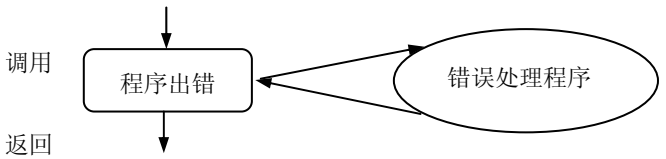

图 9.14　错误处理程序调用过程示意图

9.3.1　预见性错误处理程序

1. 基本概念

预见性错误：指能够预见的错误。

如程序在运行中，用户可能在操作中出现的一些非法操作。例如在数值型字段内输入字符型数据，这种操作对于应用程序来说，输入的数据肯定是应用程序不能处理的，也就是一种错误。但这种错误是可以想象的。

预见性错误处理程序：指用户能够预见错误产生的情况，为了不让用户在操作时出现这种错误而为程序添加的错误处理程序代码。

预见性错误处理程序主要处理用户能够预见的在运行中可能出现的错误，而对于一些出乎意料的错误无法进行处理，只能采用后面将要介绍的 Try 错误处理语句。

2. 实例演示

下面以一个实例来介绍预见性错误处理程序的设计。

创建一个简单的年龄转换程序，输入出生年份，则计算出相应的年龄。如果在输入时有误，算出年龄小于 0 或大于 200 或输入为空时都会有相应的提示，以避免因输入不规范而产生常识性的错误。

如在图 9.15 中输入出生年份为空；图 9.16 中输入年份太小；图 9.17 中输入年份太大。这三种都是输入不规范的错误，但都是可以预见的，可以预先对其进行处理。在图 9.18 中输入内容正确，所以没有错误提示，表示是正确的。

下面是【转换】按钮的程序代码：

```
integer age,year
year=integer(sle_1.text)
st_7.text=""
if year=0 then
st_7.text="您输入的年份为空，请重新输入！"
else
```

```
age=2005-year
if age<0 or age>200 then
  st_7.text="您输入的年份有误，请重新输入！"
else
st_5.text=string(age)
end if
end if
```

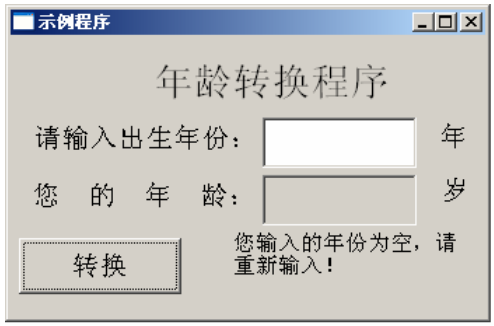

图 9.15　输入年份为空

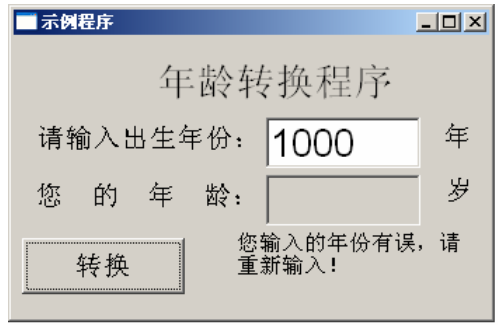

图 9.16　输入年份太小

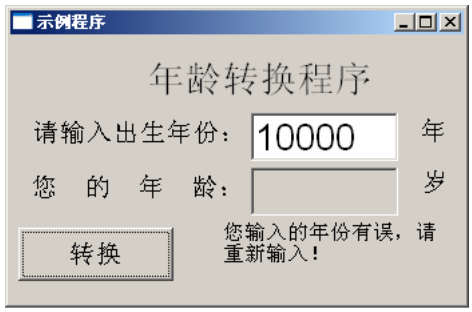

图 9.17　输入年份太大

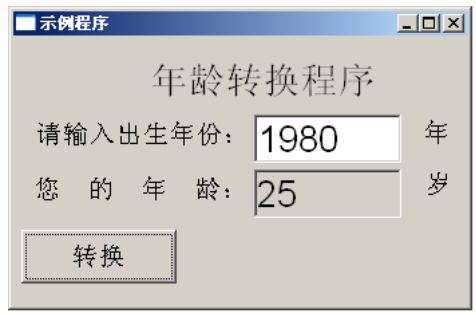

图 9.18　输入年份正确

9.3.2　使用 Try…End 语句

对于一些不可预见的错误就不可能提前对可能出现的错误进行处理，只有通过系统提供的 Try...End try 语句进行处理。

Try...End try 语句的基本结构如下：

```
TRY
   trystatements
 CATCH ( ThrowableType1 exIdentifier1 )
   catchstatements1
 CATCH ( ThrowableType2 exIdentifier2 )
   catchstatements2
 ...
 CATCH ( ThrowableTypeN exIdentifierN )
   catchstatementsN
 FINALLY
   cleanupstatements
END TRY
```

- Try(尝试)：Try 块是代码的主要部分。它是该结构其他部分的基础。
- Catch(捕捉)：只有当错误发生在 Try 块里的时候，这一块里的代码才会被执行。Catch 块可以处理所有的或者只处理特定的错误。此外，多个 Catch 块也能够被用来处理多个异常类型。
- Finally(最后)：这一块里的代码每次都会执行而不管是什么错误。
- Trystatements：表示可能造成错误的程序块。
- ThrowableType：表示可能出现的错误类型。
- Exidentifier：表示可能出现的错误代号。
- Catchstatements：表示解决出现的错误的程序块。

9.4 发布应用程序

当完成对应用程序的调试和测试后，就可以发布应用程序。发布的主要作用是将应用程序及相关的文件进行打包，并生成可执行程序，也就是把在 PowerBuilder 环境下运行的程序交付给用户，使用户在脱离了开发环境的情况下使用所开发的应用程序。所以应用程序的发布是十分重要的。

9.4.1 编译应用程序

首先我们要了解一下编译的基础知识。

1. 可执行文件

在应用程序提供给用户之前，必须要先把应用程序编译成为可执行文件，以便能够脱离 PowerBuilder 开发环境运行。在编译生成可执行文件时，PowerBuilder 提供了两种格式：伪代码和机器代码。

伪代码：是一种解释性语言，是 PowerBuilder 编译生成的一种中间代码。

机器代码：是 PowerBuilder 编译生成的二进制文件。

两种格式的可执行文件的主要区别是：

(1) 速度：机器代码的运行速度比伪代码要快，但生成时间以伪代码较短。

(2) 文件大小：伪代码生成的文件大小比机器代码生成的文件要小。

(3) 可移植性：伪代码在 PowerBuilder 支持的每个操作系统平台上都是兼容的，便于程序的移植。机器代码是与操作平台相关的，程序移植时要重新生成可执行文件。

2. 动态库文件

在生成可执行文件时，如果文件过大，一般都会把一些相关的对象放到一个或多个动态库文件中去，这样可以把应用程序划分成几个模块，缩短应用程序加载的时间；对象只在需要时才载入，节约了系统的资源。

动态库有如下优点。

(1) 可重用性：动态库可以被多个应用程序使用。

(2) 可维护性：程序更新修改后，只要修改有改动的动态库，而无需重新发行整个应

用程序。

(3) 模块化：容易划分应用程序，方便应用程序管理。

PowerBuilder 创建动态库的方式取决于编译格式：采用机器代码生成的动态库是 DLL (Dynamic Link Library，动态链接库)文件；采用伪代码格式生成的动态库是 PBD(PowerBuilder Dynamic Library，PB 动态库)文件。

3. 资源文件

几乎所有的应用程序中除了包含窗口、菜单等对象外，还会使用到诸如图标、光标指针、图形等其他资源。这些资源也应和其他文件一样和应用程序一起提供给用户。

要在编译时把这些资源一起打包，封装到动态库文件中去；或者如不封装则要给用户提供详细正确的文件路径，以避免出现错误。

9.4.2　生成可执行文件

当应用程序调试完毕后，就可以生成可执行文件。

生成可执行文件有以下两种方法。

方法 1：将应用程序中所有的对象都打包成一个可执行文件。

方法 2：把应用程序中的对象分别生成一个可执行文件和一个或多个动态库，这些动态库中包含了程序运行时所需要的对象。

第一种方法是最简单的，但它只对较小的项目来说是非常有效的。如果应用程序比较大，把所有的对象都打包在一个可执行文件中，那么这个可执行文件是非常大的。大的可执行文件的致命缺点是难于管理、执行效率低下。

从这里可以看出，一个较大的项目生成可执行文件较为合理的方法是如 9.4.1 小节中所介绍的，一个要发布的应用程序应由三个部分构成：一个可执行文件，一个或多个动态库，一些资源文件。

一个可执行文件的生成是在 Project 画板中进行的。

1. Project 画板

1) 进入 Project 画板

首先单击工具栏上【New】按钮或选择【File】|【New】命令，弹出【New】对话框。如图 9.19 所示。

在【PB Object】标签页下选择【Application】图标进入 Project 画板，如图 9.20 所示。下面对对话框中的选项进行简单的介绍。

- Executable File Name：指定可执行文件名。如希望改变可执行文件存储的目录可单击浏览按钮进行选择改变。
- Resource File Name：指定资源文件名。
- Project Build Options：指定创建可执行文件的方式。在这个选项下有一个复选框 Prompt For Overwrite：指定覆盖文件时是否提示选择。一个下拉列表框 Rebuild：两个选项即 Full 和 InCremental。

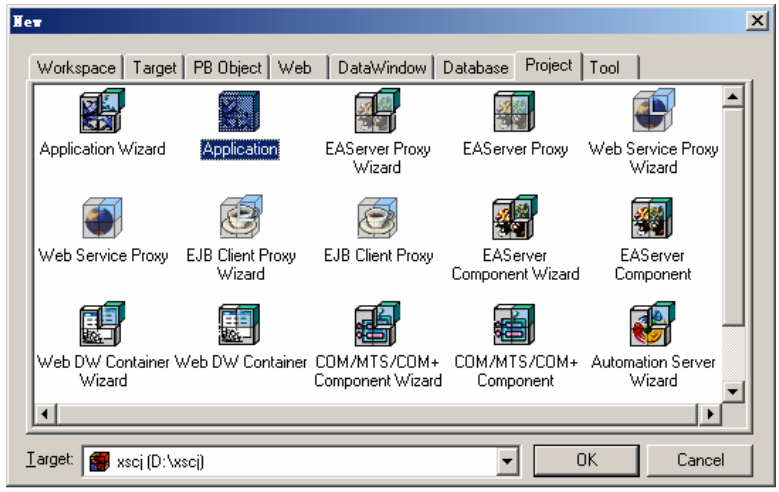

图 9.19　Project 画板

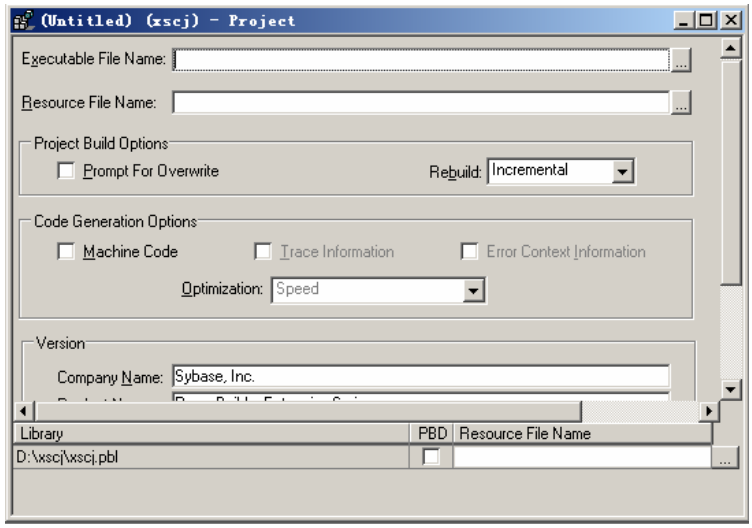

图 9.20　应用程序设置

Full：重新生成全部应用。

Incremental：仅生成新增部分。

若选择后者，则 PowerBuilder 将只生成改变的对象和被其他对象引用的对象。

- Code Generation Options：生成代码的格式。

Machine Code：生成机器代码。

Trace Information：指定是否生成跟踪文件。

Error Context Information：指定是否显示错误信息。

Optimization：选择编译时优化的级别。Speed：速度优化；No Optimization：无优化。

- Version：生成可执行文件的版本号等信息。

2) 创建资源文件

资源文件是一个扩展名为 PBR 的文本文件，是由开发人员建立的。如果应用程序没有

在此文件中被显示引用对象时，编译生成的可执行文件是不会把它们包含进来的。所以如果有引用的对象就应该在资源文件中把这些对象列出来，从而保证生成的可执行文件能正常运行。创建资源文件可以用文本编辑器把需要列出的资源在此文件中列出，然后保存为 PBR 类型的文件就可以了，如图 9.21 所示。

图 9.21　创建资源文件

3) 创建动态库

前面介绍过我们可以把应用程序中使用的对象存储在一个或多个动态库中，在执行时，动态地调用那些没有封装在 EXE 文件中的对象。创建动态库的方法有以下两种。

方法 1：在 Project 画板中创建。

(1) 进入 Project 画板。

(2) 在图 9.20 中，最下方列出了当前应用程序所包含的 PBL 库，可以选中相应的 PBL 库后面的 PBD 复选框，即可以生成动态库。生成的动态库的名称与 PBL 库文件同名，只是后缀不同，为 PBD。如果生成动态库还需要资源文件，就要在后面的【Resource File Name】编辑框中给出资源文件。

(3) 最后选择【Design】|【Deploy Project】菜单项。这样在创建可执行文件的同时，就可以创建动态库了。

方法 2：在 Library 画板中创建。

(1) 单击工具栏上的【Library】按钮，进入 Library 画板，如图 9.22 所示。

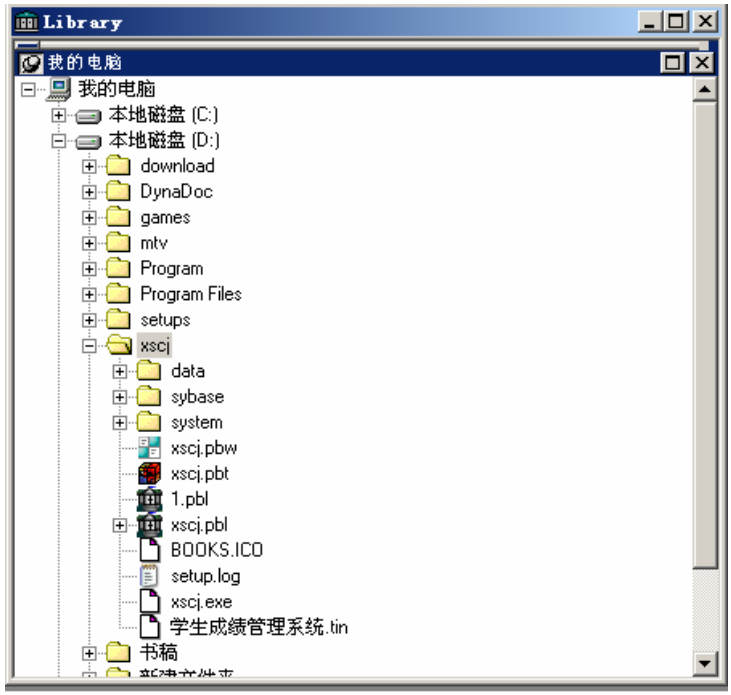

图 9.22　PBL 库

(2) 在其中选择要创建动态库的 PBL 文件。

(3) 在该 PBL 文件上右击，在快捷菜单上选择【Build Run Time Library】命令，弹出对话框，如图 9.23 所示。

(4) 指定各个选项，单击【OK】按钮即完成。

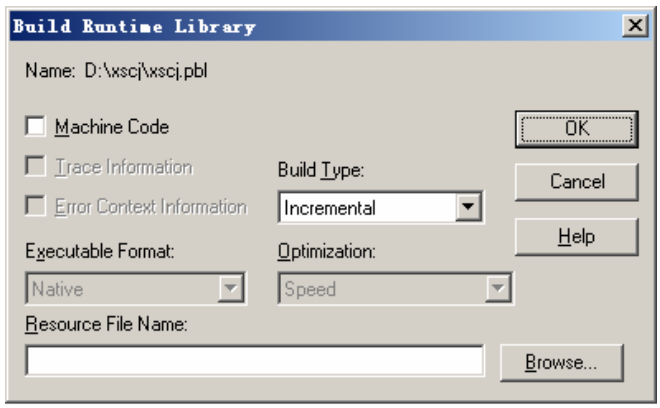

图 9.23　创建动态库

2. 生成可执行文件的步骤

可执行文件的生成步骤如下。

(1) 打开要创建可执行文件的应用程序，然后正常关闭。要注意在创建可执行文件时不可以打开其他画板。

(2) 单击工具栏上的【New】按钮，打开对话框，如图 9.14 所示。

(3) 选择【Project】标签页，选择【Application Wizard】图标，单击【OK】按钮，弹出对话框，如图 9.24 所示。

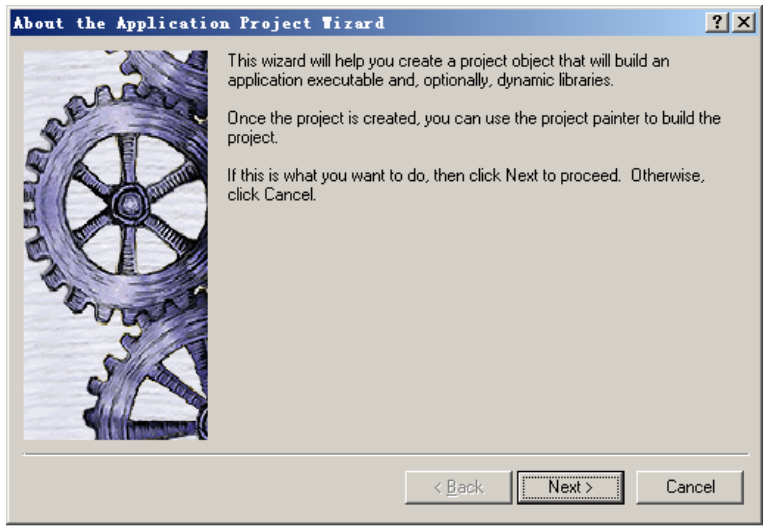

图 9.24　可执行文件建立

(4) 按照提示，单击【Next】按钮。弹出对话框，如图 9.25 所示。

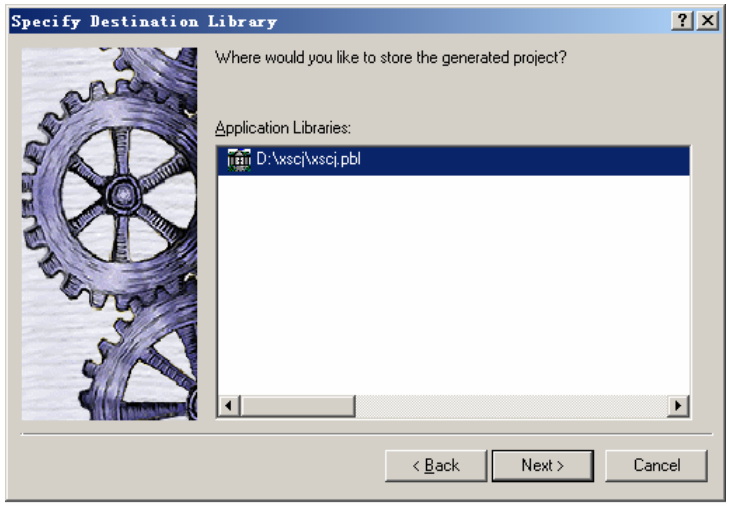

图 9.25　指定目标库

(5) 在这个对话框中显示出当前应用程序中所有的 PBL 库文件，选择需要的库文件来指定所创建的 Project 对象存放的库。单击【Next】按钮，弹出对话框，如图 9.26 所示。

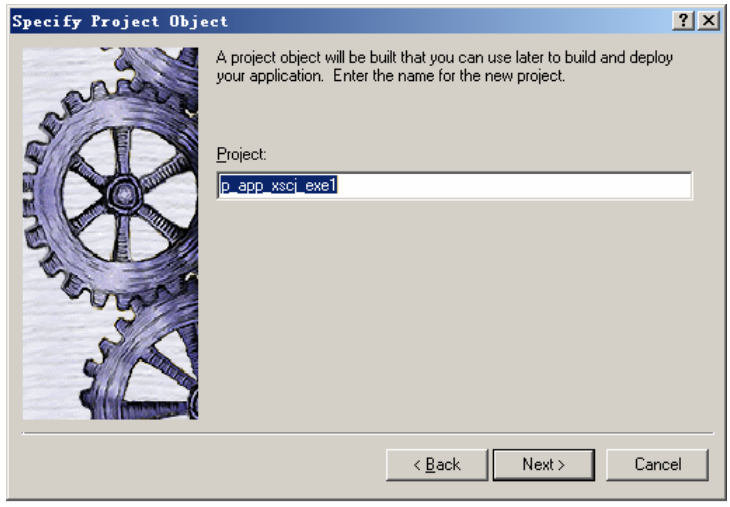

图 9.26　指定对象

(6) 在这个对话框中输入要生成的 Project 对象的名称，单击【Next】按钮。弹出对话框，如图 9.27 所示。

(7) 在【Executable File Name】编辑框中输入生成的可执行文件的路径和名称。如果需要使用资源文件，还要在【Optional Resource File(pbr)】编辑框中输入资源文件路径和名称。然后单击【Next】按钮，弹出对话框，如图 9.28 所示。

(8) 在这个对话框中可参考前面所介绍的内容进行各个选项的设置。然后再单击【Next】按钮，弹出对话框，如图 9.29 所示。

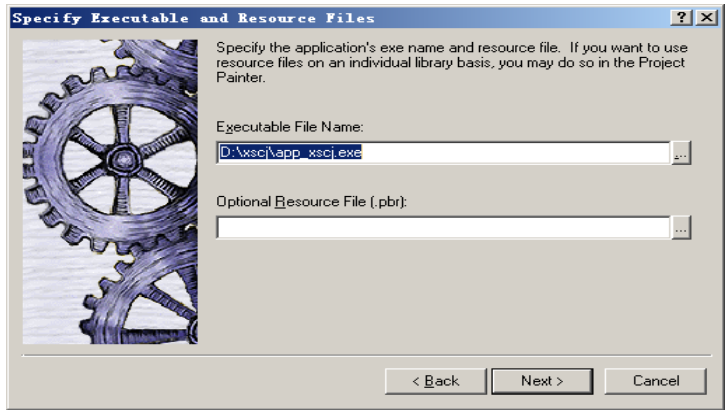

图 9.27　指定可执行文件名

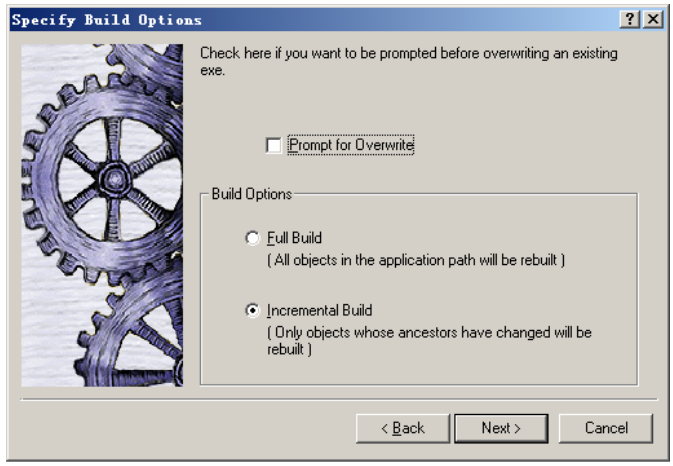

图 9.28　指定建立设置

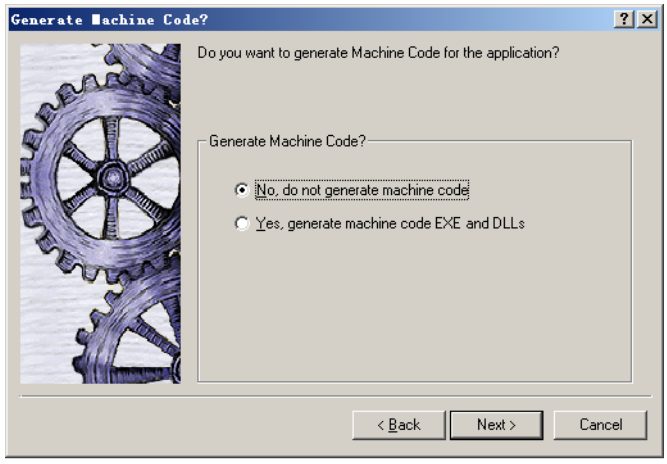

图 9.29　设置编码类型

(9) 在这个对话框中，询问是否要生成机器代码，选中【Yes】按钮或【No】按钮后，单击【Next】按钮，弹出对话框，如图 9.30 所示。

(10) 如果用户要生成动态库，则选中复选框，否则不选。单击【Next】按钮，弹出对话框，如图 9.31 所示。

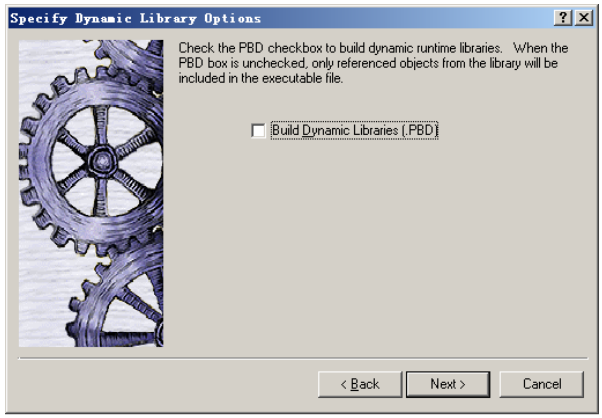

图 9.30　动态库设置

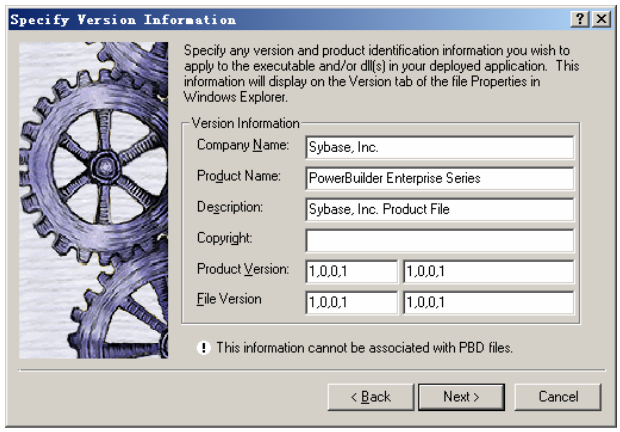

图 9.31　版本设置

(11) 在这个对话框中设置版本信息，单击【Next】按钮，弹出对话框，如图 9.32 所示。

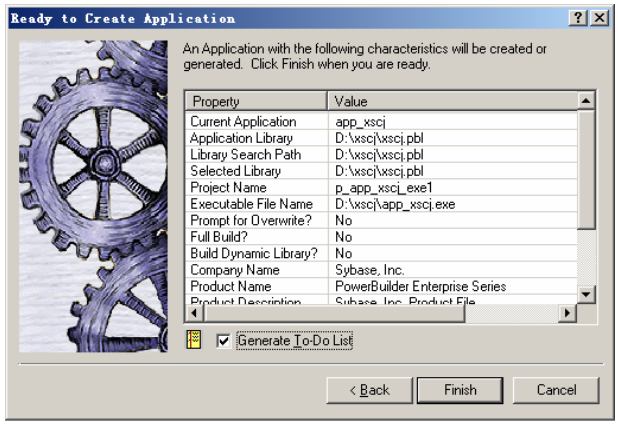

图 9.32　准备创建可执行文件

(12) 这个对话框中显示出在创建可执行文件过程中的所有参数，如果不满意可返回重新设定。如果满意就可以单击【Finish】按钮，进入 Project 画板，如图 9.33 所示。

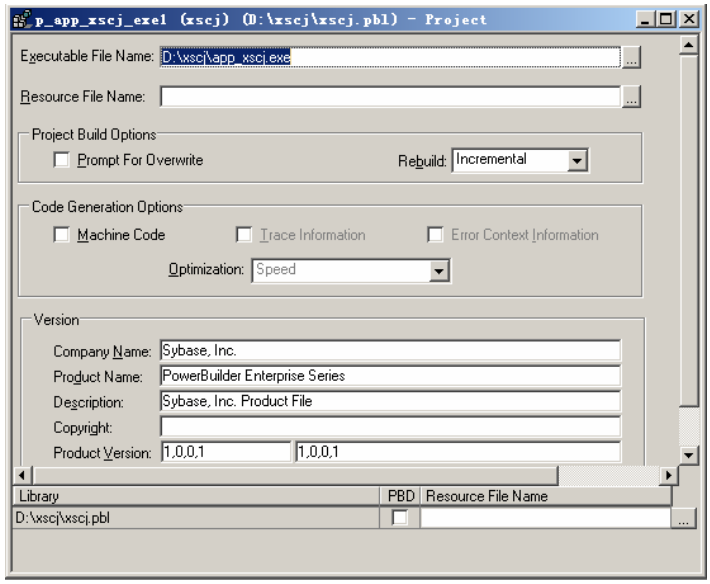

图 9.33　显示信息

(13) 如果对 Project 画板中显示的内容不满意，还可以在此进行修改。如果正确地设置了所有的选项，就可以单击工具栏上的【Save】按钮进行保存了。

(14) 创建好 Project 对象后，就可以利用它为应用程序生成可执行文件了。在 Project 画板中单击工具栏上的【Deploy】按钮，就可以立即创建应用程序的可执行文件了。如果创建不成功，系统会提示错误信息。

(15) 发布应用程序。通过前面的操作生成了应用程序的可执行文件，但此文件不是一个可以独立运行的文件。因为一个应用程序往往都需要与之配合及与之相关的其他文件的支持。所以如果要正常地使用应用程序，就必须把所需要的相关文件如动态库等告诉用户它们所在的位置，或给用户配置好文件路径，这样才能真正使应用程序发挥其作用。

9.4.3　创建发布磁盘

在上一小节中提到，应用程序的可执行文件创建完毕后，并不能真正交付用户使用。因为可执行文件要和其他很多文件配合才能正确运行，实现其实际功能，所以要告诉用户所有这些文件所在的位置，甚至于要为用户做好配置工作。但这些工作太过繁琐，而且在很多情况下是不切合实际的，因为用户不知道怎样去配置软件，也不能为所有使用应用程序的用户提供上门服务。所以可以把先期工作做好，做到这点就可以使用专用的工具为用户制作发布磁盘。

1. 制作安装盘的准备工作

制作安装盘要使用专用的工具软件。在这里介绍一种简单易用的安装盘制作工具软件：CreateInstall 2003。该软件具有以下特性。

- 可以选择包括简体中文在内的多种语言作为安装界面的语言，而不用改变程序源代码。
- 可以选择安装界面的颜色、字体和标题。
- 可以把安装程序分割成多个文件，每个文件大小可由用户自定。
- 可以在制作安装程序过程中把源文件有选择地进行压缩。
- 可以自动生成卸载程序。

制作安装程序首先应准备好应用程序的支持文件(以本书实例为例)。

(1) 要发布的程序的可执行文件以及动态库、资源文件等放在 D:\xscj 目录下。

(2) PowerBuilder 运行库以及 ODBC 接口包括内容见表 9-1。

表 9-1　ODBC 配置

文 件 名 称	说　明
PBVM90.DLL	PowerBuilder 虚拟机
LIBJCC.DLL	PowerBuilder 分发文件
PBDWE90.DLL	数据窗口引擎
PBRTC90.DLL	Rich Text
PBFNT90.INI	字体映像
PBLAB90.INI	预定义的数据窗口标签表现风格
PBTRA90.DLL	数据库链接
PBODB90.DLL	PB 的 ODBC 接口
PBODB90.INI	PB 的 ODBC 接口配置文件

以上文件可以在 PowerBuilder 安装目录下的 Sybase\Shared\PowerBuilder 获得。把它们统一复制到应用程序所在的位置 D:\xscj\Sysbase\Shared\PowerBuilder 目录下。

(3) Microsoft ODBC 驱动程序和 DLL 包括：

```
DS16GT.DLL,DS32GT.DLL,ODBC32.DLL,ODBC32GT.DLL,ODBCAD32.EXE,ODBCCP32.C
PL,ODBCCP32.DLL,ODBCINST.CNT,ODBCINST.HLP,ODBCINT.DLL,ODBCTRAC.DLL
```

以上文件可以在 Windows 系统目录 System32 获得。我们把它们统一复制到应用程序所在的位置 D:\xscj\System 目录下。

(4) Adaptive Server Anywhere 的 ODBC 数据库驱动程序及其支持文件：

```
PBBAS15.DLL,PBFLT15.DLL,PBUTL15.DLL,PBTRN15.DLL,IVPB.LIC,PBDRV15.CNT,
PBDRV15.HLP
```

以上文件可以在 PowerBuilder 安装目录下的 Sysbase\Shared\MerantODBC 获得。我们把它们统一复制到应用程序所在的位置 D:\xscj\Sysbase\Shared\MerantODBC 目录下。

(5) Adaptive Server Anywhere 运行系统文件包括：

```
DBODBC8.DLL,DBBACKUP.EXE,DBCON8.DLL,DBISQLC.EXE,DBLGEN8.DLL,DBLIB8.DL
L,DBODTR8.DLL,DBUNLOAD.EXE,DBVALID.EXE,DBENG8.EXE,DBCTRS8.DLL,DBWTSP8
.DLL
```

以上文件可以在 PowerBuilder 安装目录下的 Sybase\SQL Anywhere8\win32 获得。我们

把它们统一复制到应用程序所在的位置 D:\xscj\Sysbase\ SQL Anywhere8\win32 目录下。

2. 制作安装盘

(1) 安装好 CreateInstall 2003 以后运行，显示如图 9.34 所示。

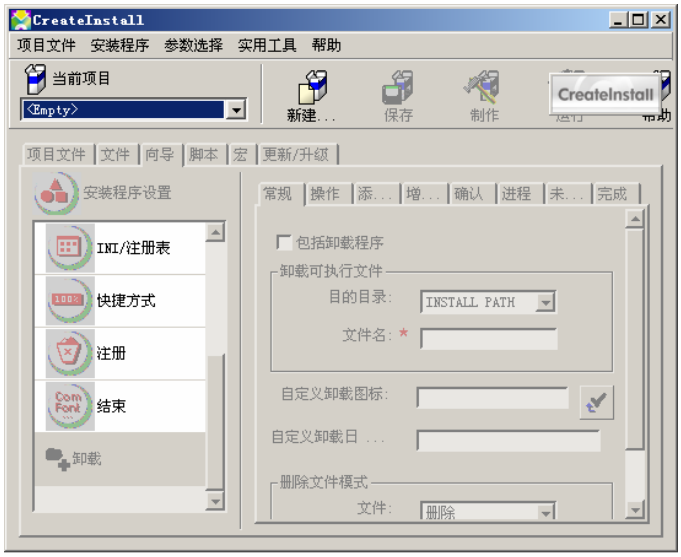

图 9.34　CreateInstall 运行界面

(2) 首先新建一个项目文件，单击工具栏【新建】按钮。弹出对话框，如图 9.35 所示。在新建项目名称中输入“学生成绩管理系统”。单击【确定】按钮。

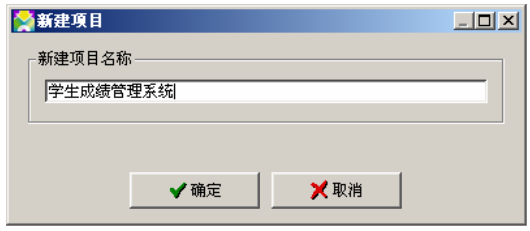

图 9.35　新建项目

(3) 在图 9.36 中的下方有 6 个标签页。首先对【项目文件】标签页进行设置。

- 输出目录：指定生成的安装文件所存入的位置。
- 安装程序的文件名：指定生成的安装文件的文件名。
- 安装程序图标：可指定生成的安装文件的图标。
- 分卷：设定安装文件是否分割成多个文件，空间是多大。

(4) 然后在【文件】标签页中进行设置，如图 9.37 所示。

在这里主要是添加安装所需要的文件。单击绿色“＋”号，弹出对话框，如图 9.38 所示。

图 9.36　项目文件标签页设置

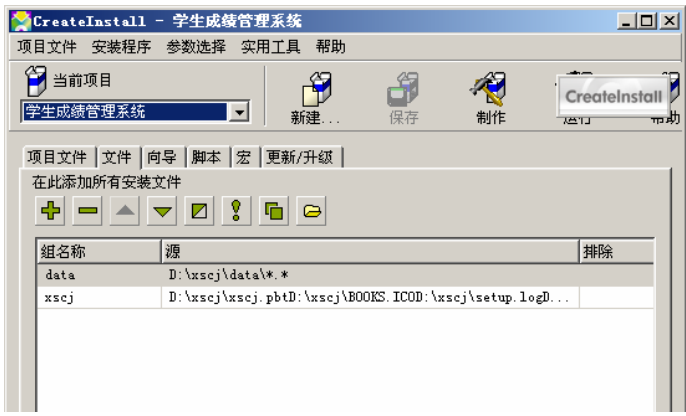

图 9.37　文件标签页设置

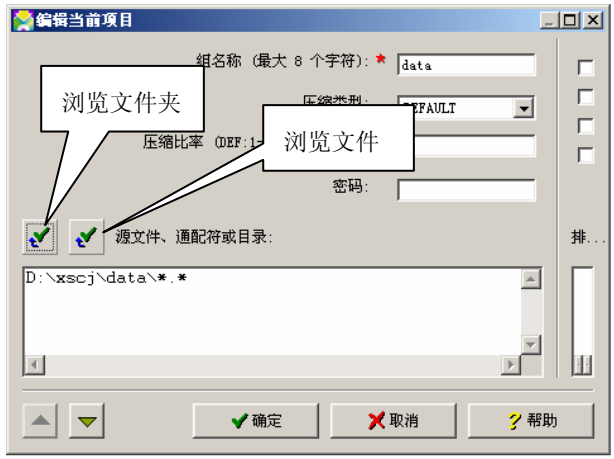

图 9.38　建立组

在组名称处输入文件组名“data”，在下方的【浏览文件夹】按钮上单击，打开如图 9.39 所示对话框。选择源程序存放目录“xscj”下的【data】文件夹，这样就可以把“data”文件夹下所有的文件定义成为一个“data”文件组。然后在图 9.34 中单击【确定】按钮。同样在图 9.37 上再新建一个文件组“xscj”，在图 9.38 中输入组名称为“xscj”，单击【浏览文件】按钮，选择源程序存放目录“xscj”，这样就可以把“xscj”目录下所有的文件定义成为一个文件组。通过以上操作就设置好安装文件中的主要文件了。

图 9.39　浏览文件夹

(5) 在向导标签页中进行设置，如图 9.40 所示。

图 9.40　向导标签页设置

向导标签页主要是引导对安装程序界面进行设置，可以依次进行设置操作。

- 在常规选项中主要是设定安装程序时应用程序的名称及默认的语言，如图9.41所示。

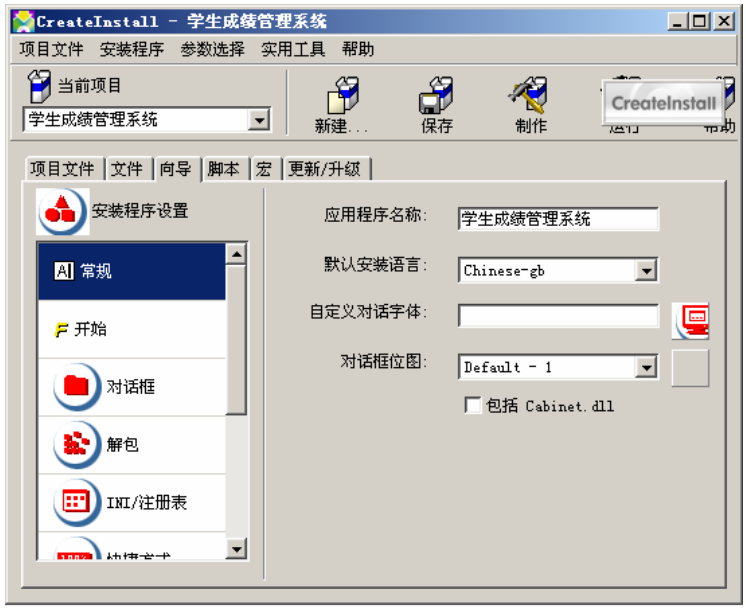

图 9.41　常规项设置

- 在开始选项中主要是设置安装程序运行时显示的背景图形及文字内容，如图 9.42 所示。

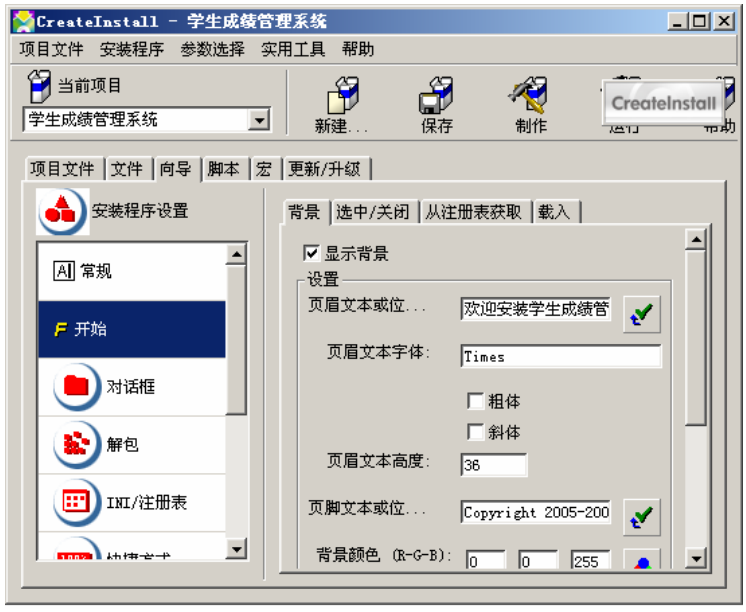

图 9.42　开始选项设置

- 在对话框选项中主要设置安装程序运行过程中出现的对话框形式和内容，如图 9.43 所示。

图 9.43　对话框设置

- 在解包选项中是对需要解包的安装文件、附加文件及文件/目录操作进行设置，如图 9.44 所示。在这里要把安装文件及前面我们提前做好准备的支持文件加入到安装程序中去。

图 9.44　解包设置

- INI/注册表选项是对注册表进行设置，涉及 ODBC 的安装与数据源的配置，所以至关重要，如图 9.45 所示。

图 9.45　INI/注册表设置

INI/注册表设置步骤如下。

(1) 选择注册表项，单击绿色“＋”号，弹出对话框，如图 9.46 所示。

(2) 在根键处输入“HKEY_LOCAL_MACHINE”；在子键处输入“software\ODBC\ODBC Drivers”；在值名称处输入“Adaptive Server Anywhere8.0”；在值类型处选择“字串类型”；在值数据处输入“installed”。

(3) 其他操作按向导提示操作即可。

通过以上操作就可制作出自己风格的安装程序。

图 9.46　注册表设置

9.5 小　　结

PowerBuilder 为我们提供了强大的软件调试工具，即 Debug 画板。在 Debug 画板中可以充分利用其强大的功能进行应用程序的断点设置、单步执行、单步跟踪等调试功能。使得我们的应用程序在交付用户使用之前，尽可能地做到完善。

PowerBuilder 提供的可执行文件的创建工具可以让我们把开发出的应用程序编译、生成可执行文件，结合其他的一些配置，最终使应用程序能够完全脱离开发环境独立运行。

9.6 实　　训

实训目的

通过本节实训，主要学习：

(1) 应用程序的调试方法。

(2) 断点的设置方法。

(3) 可执行文件的创建。

(4) 安装盘的制作。

实训内容

按照前面的案例为学生成绩管理系统制作安装程序。

实训步骤

1. 制作安装盘的准备工作

制作安装盘要使用专用的工具软件。在这里介绍一种简单、易用的安装盘制作工具软件：CreateInstall 2003。该软件具有以下特性。

(1) 可以选择包括简体中文在内的多种语言作为安装界面的语言，而不用改变程序源代码。

(2) 可以选择安装界面的颜色、字体和标题。

(3) 可以把安装程序分割成多个文件，每个文件大小可由用户自定。

(4) 可以在制作安装程序过程中把源文件有选择地进行压缩。

(5) 可以自动生成卸载程序。

制作安装程序首先应准备好应用程序的支持文件(以本书实例为例)：

(1) 要发布的程序的可执行文件以及动态库、资源文件等放在 D:\xscj 目录下。

(2) PowerBuilder 运行库以及 ODBC 接口包括内容见表 9-1。

以上文件统一复制到应用程序所在的位置 D:\xscj\Sysbase\Shared\PowerBuilder 目录下。

(3) Microsoft ODBC 驱动程序和 DLL

包括：

```
DS16GT.DLL,DS32GT.DLL,ODBC32.DLL,ODBC32GT.DLL,ODBCAD32.EXE,ODBCCP32.C
PL,ODBCCP32.DLL,ODBCINST.CNT,ODBCINST.HLP,ODBCINT.DLL,ODBCTRAC.DLL
```

以上文件统一复制到应用程序所在的位置 D:\xscj\System 目录下。

(4) Adaptive Server Anywhere 的 ODBC 数据库驱动程序及其支持文件如下。

```
PBBAS15.DLL,PBFLT15.DLL,PBUTL15.DLL,PBTRN15.DLL,IVPB.LIC,PBDRV15.CNT,
PBDRV15.HLP
```

以上文件统一复制到应用程序所在的位置 D:\xscj\Sysbase\Shared\MerantODBC 目录下

(5) Adaptive Server Anywhere 运行系统文件

包括：

```
DBODBC8.DLL,DBBACKUP.EXE,DBCON8.DLL,DBISQLC.EXE,DBLGEN8.DLL,DBLIB8.DL
L,DBODTR8.DLL,DBUNLOAD.EXE,DBVALID.EXE,DBENG8.EXE,DBCTRS8.DLL,DBWTSP8
.DLL
```

以上文件统一复制到应用程序所在的位置 D:\xscj\Sysbase\ SQL Anywhere8\win32 目录下。

2. 制作安装盘

(1) 安装好 CreateInstall 2003 以后运行，显示如图 9.34 所示。

(2) 首先新建一个项目文件，单击工具栏【新建】按钮。弹出对话框，如图 9.35 所示。在新建项目名称中输入“学生成绩管理系统”。单击【确定】按钮。

(3) 在图 9.36 中的下方有 6 个标签页。首先对【项目文件】标签页进行如下设置。

①【输出目录】下拉框中：选择 C:\setups\my project。

② 安装程序的文件名：选择 setup.exe。

③ 安装程序图标：默认图标。

④ 分卷：默认大小。

其他：均为默认值。

(4) 然后在【文件】标签页中进行设置，如图 9.37 所示。

在这里主要是添加安装所需要的文件。单击绿色“＋”号，弹出对话框如图 9.38 所示。

在组名称处输入文件组名“data”，在下方的【浏览文件夹】按钮上单击，打开如图 9.39 所示对话框。选择源程序存放目录“xscj”下的【data】文件夹，这样就可以把“data”文件夹下所有的文件定义成为一个“data”文件组。然后在图 9.34 中单击【确定】按钮。同样在图 9.37 上再新建一个文件组“xscj”，在图 9.38 中输入组名称为“xscj”，单击【浏览文件】按钮，选择源程序存放目录“xscj”，这样就可以把“xscj”目录下所有的文件定义成为一个文件组。通过以上操作就设置好安装文件中的主要文件了。

(5) 在向导标签页中进行设置，如图 9.40 所示。

向导标签页主要是引导对安装程序界面进行设置，可以依次进行设置操作。

- 在常规选项中主要是设定安装程序时应用程序的名称及默认的语言，如图 9.41 所示。
- 在开始选项中主要是设置安装程序运行时显示的背景图形及文字内容，如图 9.42 所示。

- 在对话框选项中主要设置安装程序运行过程中出现的对话框形式和内容，如图 9.43 所示。
- 解包选项中是设置对需要解包的安装文件、附加文件及文件/目录操作进行设置，如图 9.44 所示。在这里要把安装文件及前面提前做好准备的支持文件加入到安装程序中去。
- INI/注册表选项是对注册表进行设置，涉及 ODBC 的安装与数据源的配置，所以至关重要，如图 9.45 所示。

INI/注册表设置步骤如下。

(1) 选择注册表项，单击绿色“＋”号，弹出对话框，如图 9.46 所示。

(2) 在根键处输入“HKEY_LOCAL_MACHINE”;在子键处输入“software\ODBC\ODBC Drivers”；在值名称处输入“Adaptive Server Anywhere8.0”；在值类型处选择“字串类型”；在值数据处输入“installed”。

(3) 其他操作按向导提示操作即可。

9.7 习　　题

1．简述 PowerBuilder 编译生成的伪代码与机器代码的区别。

2．简述可执行文件的创建过程。

3．简述制作安装盘的作用及意义。