第15章　Oracle Developer与开放式系统

开放式系统是指允许使用来自不同的制造商的工具的系统。 UNIX成为了原型的开放式系统：有许多不同的UNIX操作系统，可以互换地使用它们 (当然要合理地使用)。在20世纪80年代后期，随着库技术和工具集成体系结构的发展，工具集成开始在软件工具市场出现，例如Atherton Backplane，它允许程序员把多种软件工具集成到一个聚合的工作过程中。后来针对PC机和Macintosh机又出现了动态数据交换 (DDE)、发布 (Publish)和预定 (Subscribe)，允许系统设计人员在运行时与其他的工具进行通讯。

一方面，随着对象链接和嵌入 (OLE)接口技术的发展以及 ActiveX和组件对象模型 (COM)的成功实现，另一方面，随着 CORBA(公共对象请求代理体系结构)和网络化的Java(“可在任何地方运行”)的出现，开放式系统开始向建立软件体系结构方面发展，即向下至对象层开放所有方法。使用这样的体系结构，可以创建对象并在许多应用程序和服务器中共享使用它们。

本章介绍Oracle Developer提供的开放式系统环境中的一些方法。现在达到使应用程序脱离真实世界而只作为一个系统的阶段。如何能够使用所有的 Oracle Developer共同工作来开发应用程序呢？对Oracle Developer以外的对所有系统要做些什么呢？如何能创建一些东西作为一个系统公共的部分而不仅仅是孤立的呢？如何使应用程序在作用域中成为共用的，而不是要依赖于单一的环境或语言？

15.1　集成Oracle产品

作为迈向开放式系统的第一步， Oracle Developer可以使用另外一个工具。在第4章已经看到了这种使用方法，即几个混合使用表单、报表和图形的基本方案。还有其他可能的组合，例如在一个图形显示中显示一个报表，或从一个报表运行一个表单，以及其他可能的建立集成的方法。还可以在一个饼图图表的一部分单击鼠标，产生有关该主题的一个报表 (虽然像这样的下钻操动作的技术细节已超出了这本书的范围)。

本节内容给出如何集成Oracle Developer产品的基础的概述。对于PL/SQL程序包以及如何在应用程序中使用这个程序包的细节，可以参考每个单独的产品的手册。 Oracle Developer还集成了Oracle Express的数据开采工具和Oracle Developer系统，同时允许将Oracle Developer的安装和使用与系统的要求和设计结合起来。

Oracle Developr的Forms组件有最宽广的工具阵列，以集成其他的 Oracle Developer产品。要问的第一个问题是：将要运行的系统是一个客户机 /服务器系统还是带有 Oracle Developer Server的一个三层(Three-tier)系统。客户机/服务器系统把Reports Runtime或Graphics Runtime当作访问数据库服务器的一个本地程序来运行。把运行程序的可执行部分安装在每一个运行报表和图表的客户机上。三层 (Three-tier)系统使用运行于中间层服务器上的 Forms服务器或Graphic服务器来运行报表或图形显示。因此，应用程序运行在哪种体系结构上，涉及到对其他Oracle开发工具的集成。

如果希望报表和图形显示透明地运行，而不管是客户机 /服务器还是三层结构，那么应该

使用Run_Product内部过程来运行该程序。如果只是作为客户机/服务器型的程序运行，则应该使用Run_Product或是Run_Report_Object。Run_Report_Object是一个与Form Builder的报表对象一起工作的内置子程序，用于集成表单和报表。如果运行在三层结构上，可以使用这两个子程序，但还可以使用 Web.Show_Document内部过程，该内部过程允许把一个 URL提交给中间层的Application Server，并返回一个在Web浏览器上显示的页面。下面几节说明如何使用这三个内置子程序把组件集成表单应用系统中。

也可以使用图表项从表单中运行图形。第 4章的“在表单中创建图形显示”中给出了使用图表项建立图形显示的详细例子。

15.1.1 用Run_Product集成产品

Run_Product内部过程是在Oracle Developer Forms中使用两层方式运行其他Oracle产品的机构。

注意 在图形中，Run_Product过程是TOOLINT程序包的一部分，所以必须把程序包的名字加到过程名字上：TOOLINT.Run_Product。

下面是Run_Product内部过程的语法格式：

```
Run_Product(<product>, <module>, <communication mode>,
            <execution mode>, <location>, <list>, <display>);
```

表15-1给出这七个参数的可取值和含义。

表15-1 Run_product参数

参 数	可 取 值	描 述
<product >	FORMS、REPORTS、GRAPHICS、BOOK	要运行的Oracle Developer的组件：Forms、Reports、Graphics或Book
<module>	字符串	在运行产品时打开的模块或文档的名字
<communication mode>	SYNCHRONOUS、ASYNCHRONOUS	同步调用在退出所调用的产品以后将控制返回给Forms。异步调用立即将控制返回，并行运行所调用的产品
<execution mode>	BATCH、RUNTIME	调用一批产品，采用后台方式运行一个报表或显示，没有用户交互。运行时产品用前台方式调用运行产品，允许进行交互。只有Report和Graphics可以在批方式下运行
<location>	FILESYSTEM、DATABASE	这个参数通知本过程模块驻留的位置是在文件系统中还是在数据库中
<list>	Param-List对象	参数列表包含有传递给该产品的一系列参数，传递参数的名字或者参数对象的ID
<display>	字符串	这是图表项块的名字，其中将包含所请求的来自Graphics的显示

注意 当从表单向正被调用的产品传递一个记录组时，或者当希望在一个客户机的浏览器中查看HTML、HTML CSS或PDF的输出时，必须使用SYNCHRONOUS通信模式。当调用一个表单或者一个工作簿的文档时，必须使用RUNTIME执行方式。只有当产品是Oracle Developer的Graphics组件时，才使用display参数。

例如，当用户单击一个按钮时，可能想要运行一个报表。应该将下面的代码放到这个按

钮的When-Button-Pressed触发器中：

```
Run_Product(Reports, 'LedgerSummary', RUNTIME, FILESYSTEM,
            NULL, NULL);
```

提示 可以不用路径名来限定模块名。使用在中间层服务器上的注册变量REPORTS60_PATH，指明包含将要运行的报表的目录。因为不需要涉及模块，而仅仅涉及注册关键字，所以这样会很容易地实现将报表模块重新定位到其他目录上。

为了在三层结构中用 Run_Product来运行一个报表，必须要设置注册变量 REPORTS60_PATH(包含报表的目录列表)、FORMS60_OUTPUT(将要保留报表输出的目录)、FORMS60_MAPPING(FORMS60_OUTPUT目录的虚拟目录，例如 /report_output/)和FORMS60_REPFORMAT(根据是否要进行 Adobe Acrobat输出或者 HTML输出决定取“ PDF ”还是“ HTML ”)。

第10章的“ Run_Prodnct ”节中也包含有一些例子和补充解释，说明如何来使用Run_Product子程序的参数列表的性能。

15.1.2 用 Run_Report_Object 集成报表

内置子程序Run_Report_Object使用表单上的报表对象的设置信息，这个报表对象是在Form Builder中创建的。

首先创建这个报表对象并且设置好它的特性。然后在一个菜单项或一个表单触发器中的代码中使用这个内置子程序。下面的教程在一个报表对象中建立 LedgerSummary报表，然后编制When-Button-Pressed触发器的代码，来运行这个报表。

1) 在Form Builder Object Navigator中，选择Reports节点，并单击Add工具按钮。

Form Builder显示一个对话框，或是使用 Report Builder建立一个新的报表，或者是输入一个报表的名字。

2) 输入名字“ LedgerSummary.RDF ”，或者使用“ Browse(浏览) ”按钮浏览找到Ledger-Summary.RDF 文件，然后单击OK(见图15-1)。

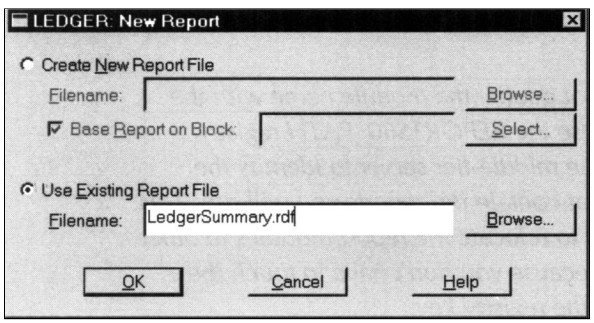

图15-1 输入文件名

3) 单击新的对象，改变新对象的名字，输入“ LedgerSummaryReport ”，再按回车键。

4) 双击该对象，显示Property Palette。

5) 设置Execution Mode属性为Runtime、Communication Mode属性为Synchronous、Report Destination Type为File、Report Destination Name为LedgerSummary.html、Report Destination Format为HTML、Report Server为repsvr.world(Reports Server的TNS名字)(见图15-2)。

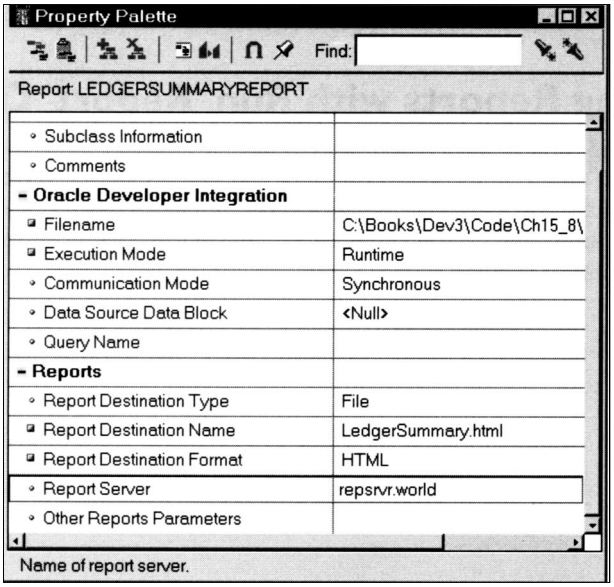

图15-2 Property Palette

提示 可以用一个命令行字符串形式在Other Report Parameters特性中提供任何的报表参数，例如用“PERSON_NAME=‘Adah Talbot’”来设置Person_Name参数。

6) 在画布上创建一个按钮项，并增加一个When-Button-Pressed触发器。

与Run_Report_Object、Report_Object_Status和Copy_Report_Object_Output过程一起使用这个报表对象，来设置和处理在服务器上或本地运行的一个报表。

```
DECLARE
  vReportID REPORT_OBJECT;
  vReportName VARCHAR2(100);
  vReportStatus VARCHAR2(20);
BEGIN
  vReportID := Find_Report_Object('LedgerSummaryReport');
  vReportName := Run_Report_Object(vReportID);
  vReportStatus := Report_Object_Status(vReportName);
  IF vReportStatus = 'FINISHED' THEN
    Message('Report ready.');
    Copy_Report_Object_Output(vReportName,
            'C:\TempReports\Ledger.htm');
  ELSIF vReportStatus IN ('RUNNING', 'OPENING_REPORT', 'ENQUEUED')
    THEN Message('Report running, please check for output later.');
  ELSE
    Message('Error when running report: '||vReportStatus||'.');
  END IF;
END;
```

Report_Object_Status函数可以返回这些值中的任何一个：finished、running、canceled、opening_report、enqueued、invalid_job、terminated_with_error、crashed。这些函数允许对报表进行处理，这些报表是在Reports Server上运行，而不是在当前进程中运行。可以等待报表的输出，可以报告状态并继续执行(如在例子的代码中所示)，可以定期地检查有关状态，使用报表的状态来告知用户何时可以使用报表。

15.1.3 使用Web.Show_Document集成Reports组件和Graphics组件

使用Web.Show_Document允许表单发送一个URL给一个Application Server。这个URL可以是任意的Web页的地址或者是要显示文件的地址，但是它也可以是一个 Cartridge或CGI命令字符串，该命令串通知服务器执行 Report或Graphics的Runtime。

例如，如果在名为Ledger的表单上有一个用来运行名为 LedgerSummary报表，并在Web浏览器中显示它的按钮，那么这个按钮的 When-Button-Pressed触发器包含这样的代码：

```
Web.Show_Document(
  'http://www.talbotsfarm.com/reports/Reports60Cartridge?
  server=repsvr&report=LedgerSummary.rdf&userid=talbot@orcl&
  DESTYPE=CACHE&DESFORMAT=HTML', '_self');
```

引号中的字符串在程序单元中应该是在一个文本行上，在这里多行显示是由于长度的限制。这个命令的CGI描述与上面的描述非常相似：

```
Web.Show_Document('http://www.talbotsfarm.com/cgi-bin/rwcgi60.exe?
  server=repsvr&report=LedgerSummary.rdf&userid=talbot@orcl&
  DESTYPE=CACHE&DESFORMAT=HTML''_self');
```

CGI版本用CGI的虚拟目录代替报表的 cartridge的虚拟目录，使用的是 rwgi60.exe而不是cartridge的名字(Reports60Cartridge)。Web.Show_Document的第二个参数告知Forms Runtime在哪显示报表的输出：

- _SELF：在与表单相同的框架或窗口中装载报表。
- _PARENT：在表单的父窗口或框架中装载报表。如果该表单是最高层， _PARENT与_SELF相同。
- _TOP：把报表装载到包含该表单的窗口上，并替换表单的显示内容。
- _BLANK：打开一个新的、最高层的无名窗口，并将报表装入其中。

使用Web.Show_Document显示一个非常相似的图形：

```
Web.Show_Document(
  'http://www.talbotsfarm.com/graphics/Graphics60Cartridge?
  &module=AmountTrend.ogd&userid=talbot@orcl', '_self');
```

对一个图形的显示，要提供 MODULE和已经在 Application Server上定义了的 Oracle Graphics cartridge的名字。

15.2 通过ODBC使用其他数据库管理器

ODBC标准是一个用于数据库管理器的实际的标准应用程序编程接口 (API)。API接口使得所编写的程序可以从一种DBMS移植到另一个DBMS上，而不用重新编程，甚至不用重新编译代码。为了在一个非 Oracle的其他的 DBMS上使用 Oracle Developer，需要了解 Oracle Developer如何能够通过ODBC来访问那种DBMS系统。

Oracle开放客户机适配器(OCA)是一个ODBC的接口，这个接口允许在Oracle Developer中使用ODBC的调用代替 Oracle的调用。通过使用专门的数据库注册字符串，来访问这个软件层：

```
user_name/password@odbc:data_source
```

User_name和password是数据库的标准的名字和口令， data_source是数据库的ODBC的名字。data_source的前缀词‘ odbc ’告诉Oracle Developer要使用OCA接口。也可以用一个星号

作为data_source，在这种情况下，ODBC将用机器上已注册的数据源进行提示。

使用ODBC管理程序或者其他的数据库制造商提供的程序来设置一个数据源。有关设置的详细内容，查阅数据库管理程序文档或ODBC驱动程序文档。

注意 OCA接口支持各种ODBC兼容的数据源。Oracle带有几个具体的ODBC驱动程序和数据库的文档，但是OCA接口支持各种兼容的驱动程序。

Oracle提供了创建数据字典视图的脚本，这些数据字典视图与在 Oracle RDBMS中的类似。运行这些脚本允许使用 Oracle Developer中的多种工具访问数据字典。可以创建自己的系统脚本，而不是Oracle所提供的那些脚本。只是复制一个脚本，然后根据所发现的系统具体的不同对它进行修改。

如果使用一个ODBC驱动程序通过OCA连接到目标数据库，有几个 Oracle Developer的功能将不能工作：

- 不能在数据库中存储模块。
- 不能访问Object Navigator的数据库对象组中的列的触发器的有关信息。
- 只能查看存储在数据库对象组中的过程，不能编辑它们。
- 不能在数据库对象组之间对过程进行拖动和放入。
- Oracle Developer不能使用约束信息产生主从关系或者强制主关键字。
- 如果在SQL语句中出现优化程序的提示，OCA不予理睬。

Oracle Developer有两种可用的工具，分别提供使用特定数据源的最佳使用方法。 OCA PL/SQL库函数包含有几个用于确定目标数据库是什么以及它提供哪些功能的子程序。EXEC_SQL库函数允许发出使用专用的语法说明访问服务器的 SQL语句。通常，PL/SQL代码必须遵循标准的Oracle PL/SQL的规定以及访问数据库服务器的特定语法。可以在ORACLE.INI文件中或者在系统注册表中，对OCA_DEBUG_SQL=TRUE和OCA_DEBUG_ERROR=TRUE进行设置，以便对将被发送到数据源的SQL语句进行调试。

Oracle Developer自动地根据情况对它的对象的特性进行调整，在需要时对它们进行修改，以求产生一致的结果。有一些确定的特性必须特别小心：

- 必须把与主关键字列相对应的项的 Primary Key特性设置为Yes，因为OCA不能从数据源得到这个信息。
- 如果得到不能更新主关键字列的有关错误信息，可能还必须把数据块特性 Update Changed Column设置为Yes。
- 如果试图一次要提取多于数据源能支持的记录，可能还得调整 Records Fetched这个数据块特性。
- 必须把数据源以混合形式存储的对象名用引号引起来。多数数据源对大小写不进行区分，但是有一些数据源，像Sybase，把“ Ledger ”和“ ledger ”区别对待。如果使用的是这些数据源，而且对象名字是用小写形式，那么在 SQL语句中必须用双引号(“)把名字引起来。

OCA提供了一个功能强大的工具，用于在非Oracle的数据源中的数据上打开应用程序。

15.3 将组件集成到Oracle Developer表单

组件是一个软件系统，可以与其他组件结合成为一个完整的运转的软件系统。组件一直

是被认为包含在一个表单或其他的对象中，但是随着面向对象的体系结构的广泛应用，这个概念具有了新的重要地位。 Oracle Developer支持几种主要的组件标准。主要的两种标准是JavaBeans和ActiveX。下面两节分别地说明如何与Oracle Developer的Forms应用系统一起重新使用Java和ActiveX组件。

15.3.1 将Java组件集成到表单中

Java是一种编程语言，由Sun微系统公司不间断的和有争议的过程中发展起来，并形成标准。Java标准化的一个方向是JavaBean —— 一种组件说明，它允许在Java的小应用程序(运行在Web浏览器上的代码)、servlet(运行在Application Server上的代码)和application(运行在它自身系统上的代码)把Java类的系统当作组件使用。 Java从C++程序设计语言中沿用了许多语法结构，进行许多改进和扩充。 JavaBean说明使方法标准化，使它的系统的特性对客户机有效。

注意 如果想在应用程序中使用JavaBean创建自己的JavaBean类或者Java类，可能要求具备一定的Java的编程技能。例如，为了在一个bean区域中使用一个JavaBean，必须在Java中编写一个wrapper类，该类对由带有Developer服务器的Oracle提供的VBean类进行扩充(从它继承)。所有这些Java代码放入在应用服务器上的虚拟目录/codebase/中(有关这些虚拟目录的详细情况参看第14章)。由于讲授Java技能和方法超出了本介绍主题的范围，并且因为篇幅所限，这一节不会介绍集成到JavaBean需要编写的实际的Java代码。查阅Oracle Developer的帮助文件，了解可以使用的一些例子代码，仔细查阅随 Oracle Developer系统销售的Oracle Developer演示的例子。

可以从第三方销售商处购买 JavaBean，或者可以使用 Oracle的Developer或是其他Java Developer编写自己的JavaBean。

可以通过两种方法来使用JavaBean：

- 链接：可以在一个数据块中创建专门的 Bean Area项，使用Implementation Class特性链接到JavaBean，使用When-Custom-Item-Event触发器来处理在Bean Area中发生的事件。
- 定制：通过编写实现IView接口的JavaBean，可以用JavaBean来替代具体的控件(复选框、列表项、控制按钮、单选按钮组和文本项)，既可以直接地替换，也可以用扩展的VBean替代或者是一种其他的Oracle JavaBean类如VButton进行替换。

要把一个JavaBean控制链接到应用程序，在Form Builder中按下列步骤进行：

1) 打开将要显示控件的画布，单击Bean Area工具，拖画出将显示这个控件的一个矩形区域。

2) 编写扩充VBean类的一个包装类，例如像 VCalendar(一个日历bean)。把Java和类文件放到在应用服务器上的虚拟目录/codebase/中，确保表单在运行时能够对它进行访问。

这个包装类提供一些特定的方法， Forms小应用程序调用这些方法来设置特性和获得鼠标事件。它还注册了表单能够识别的一些特殊事件。

3) 导航定位到在 Object Navigator中的Bean Area项上，双击该项以显示它的 Property Palette。将Implementation Class属性改为该包装类的名字。使用完全限定名字，包括整个软件包的名字，例如“ oracle.forms ”(见图15-3)。

图15-3 输入合法名字

4) 编写一个事件捕获过程并将它存储到一个程序库中。把这个程序库附加到使用JavaBean的表单上。这个过程从系统变量CUSTOM_ITEM_EVENT_PARAMETERS获取参数，从系统变量CUSTOM_ITEM_EVENT获取事件的消息。程序代码与下列内容相似：

```
PROCEDURE EventTrap IS
  vBean ITEM;
  vBeanParamList PARAMLIST;
  vParamType NUMBER;
  vEvent VARCHAR2(20);
BEGIN
  vBeanParamList := Get_Parameter_List(:SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS);
  vEvent := :SYSTEM.CUSTOM_ITEM_EVENT;
  vBean := Find_Item('CONTROL.CALENDAR'); -- gets the CALENDAR bean item from the
CONTROL block
-- Process the supported events registered through the wrapper Java class
END;
```

提示 可以在应用程序的程序单元部分编写这个过程。但是，当考虑可重用组件时，把该过程包装到可以在其他应用程序中重用的一个程序包中是一个好方法。或许想要有与JavaBean的集合相对应的一个库，这样，当使用一个bean时附加上这个程序库，同时事件捕获过程处于就绪和等待状态。

5) 从这个Bean Area项的一个When-Custom-Item-Event触发器中调用这个事件捕获过程：

```
EventTrap;
```

在PL/SQL代码中使用Set_Custom_Item_Property内部过程可以设置JavaBean对象中的属性，需要给出Bean Area项的名字，属性的名字（检查Bean文档）和属性的值。值可以是VARCHAR2、INT或BOOLEAN。

```
Set_Custom_Item_Property('Control.Calendar', 'enabled', TRUE);
```

开发一个标准的表单控件的定制版本，只需要替换在上述处理步骤中的 Bean Area控件。例如在一个文本项中，把Implementation Class属性指定为置换的JavaBean的名字，并且为这个文本项增加一个When-Custom-Item-Event触发器。

15.3.2 把ActiveX组件集成到Forms中

ActiveX是Microsoft专用的一种技术套件，它能够使Windows应用程序对各种各样系统所开发的不同方面使用标准的接口。每个ActiveX控件都是一个提供定义明确的接口（程序和图形接口）并能发送事件到客户端（例如应用程序或网络上的其他应用程序）的组件。ActiveX控件以前被当作是OCX控件，它是建立在组件对象模型（COM）标准上的，组件对象模型（COM）标准也是Microsoft的一项技术。

注意 只能把ActiveX控件用于在Microsoft Windows系统下运行的应用程序中。

通过在数据块中定义一个ActiveX控件的方法，把ActiveX控件链接到应用程序上。Oracle Developer导入该控件的方法和事件，创建与该控件进行接口的代码骨架，然后在已产生的过程内部编写代码。当该控件激活一个事件时，调用与该事件对应的过程。

注意 使用On-Dispatch-Event触发器，可以进行一些复杂的程序设计，允许在事件过程

是缺省限定过程的情况下进行工作(也就是说，不能调用导航的非限制的表单过程)。参看帮助文件，可了解有关这种高级ActiveX编程技术的更多信息。

要在应用程序中包含一个ActiveX组件，在Form Builder中执行下列步骤：

1) 选择“ Program| Import OLE Library Interfaces ”菜单项。Form Builder显示“ Import OLE Library Interfaces ”对话框(见图15-4)。

2) 滚动已登记的ActiveX组件，直到发现要查找的ActiveX组件(本教程使用OracleSpreadTable组件)。对话框中显示所有可用于这个组件的方法和事件 (见图15-5)。

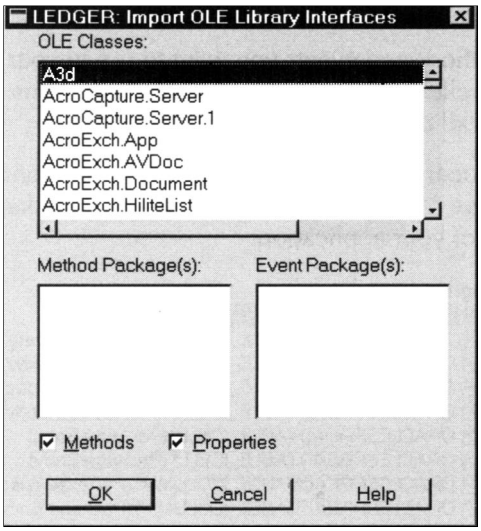

图15-4 Import OLE Library Interfaces对话框

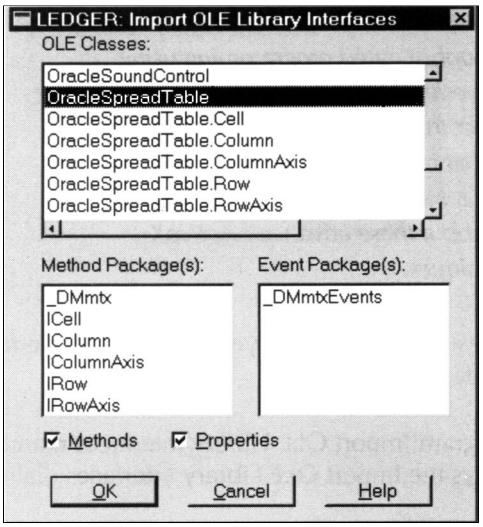

图15-5 对话框显示所有的方法和事件

3) 使用标准的选择多个项目的方法 (按住CTRL键，单击鼠标)，选择要在应用程序中使用的方法和事件。当选择了所有的方法和事件以后，单击OK。

然后Oracle Developer为所选择的方法和事件创建过程的骨架。在应用程序的 Program Unit(程序单元)段产生一系列的程序包 (见图15-6)。

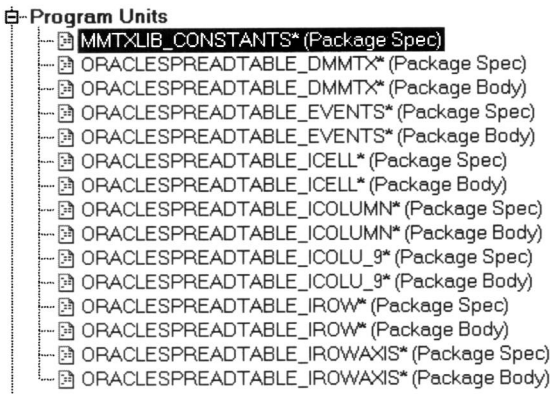

图15-6 Program Units段

4) 显示将要显示控件的画布。单击 ActiveX控件的工具，在要显示控件的画布上画一个矩形(见图15-7)。

5) 双击画布上的矩形控件，显示该控件的 Property Palette。单击“ … ”按钮，显示 OLE Classes的对话框。这是一个LOV，所以可以使用标准的LOV模式匹配想要查找的控件 (当在导入事件和方法时，在选择列表对话框中能找到相同的控件)(见图15-8)。

6) 选择控件名字，然后单击OK(见图15-9)。

7) 在画布的控件上单击右键，在弹出菜单中选择“ Insert Object(插入对象) ”。Form Builder显示Insert Object对话框。确保Create Control单选按钮被选定，然后单击OK(见图15-10)。

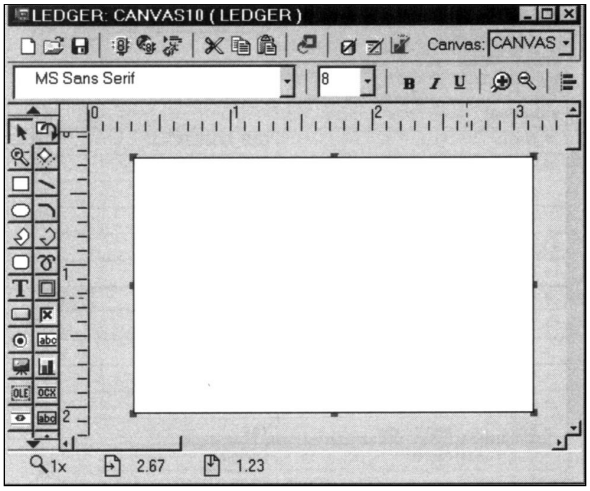

图15-7 在画布上画一个矩形

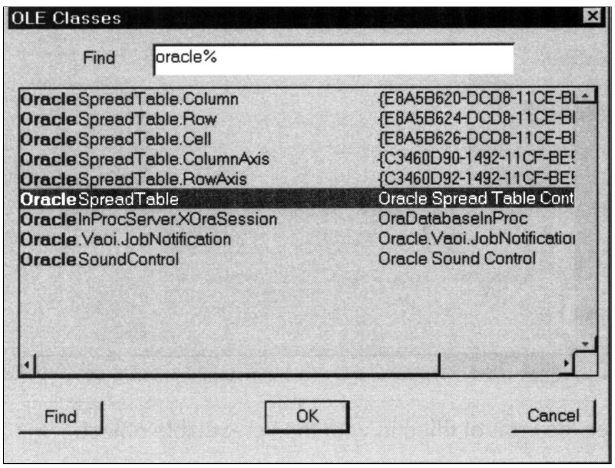

图15-8 OLE Classes对话框

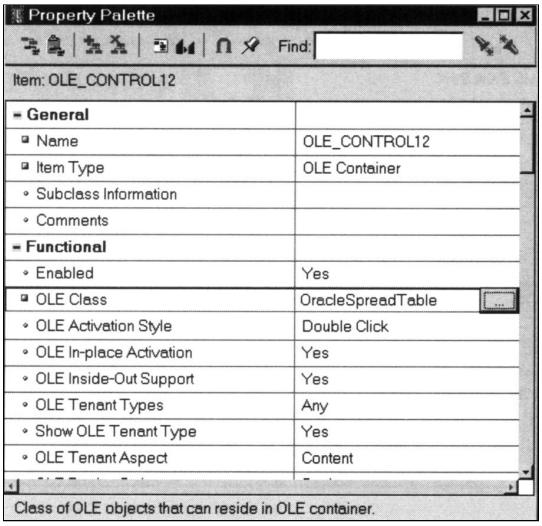

图15-9 选择控件名

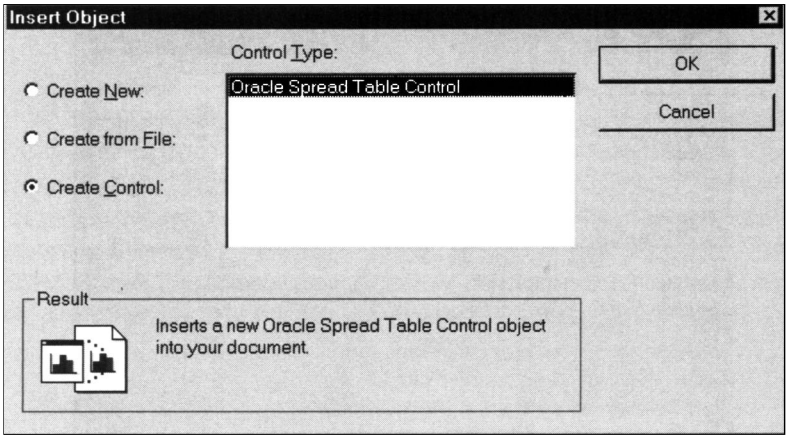

图15-10 在Insert Object对话框中选择Create Control单选按钮

然后会看到控件已被用Spreadtable对象填充(见图15-11)。

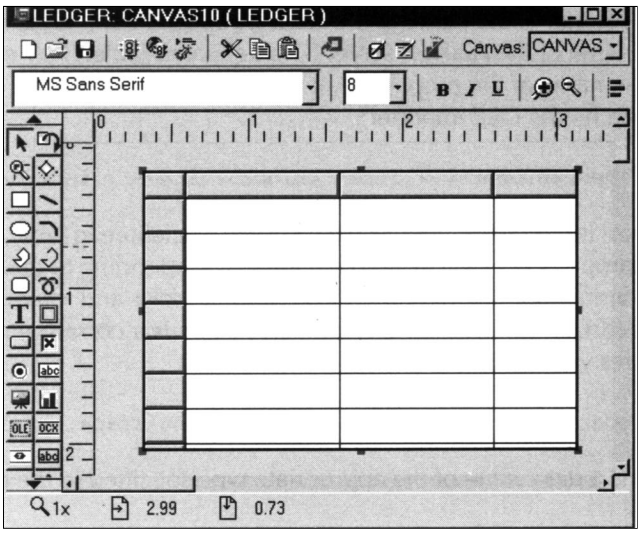

图15-11 填充Spreadtable对象的控件

提示 一旦已经“占据”了控件，就能够查阅有关该控件的任何文档，弄清楚每个事件和方法要做什么以及为了响应一个事件需要做些什么。通过控件的Property Palette，使用Control Help属性，可以访问内部文档。一旦已经设置了控件的名字，只需要单击“…”按钮即可。使用Control Properties属性同样可以设置控件的任何属性。

接下来需要进入到程序单元中的每个程序包，用实现已导入界面的代码来替换程序包中的NULL部分。也可以使用导入方法编写其他的子程序和程序包，做所想要做的各种事情。可以在PL/SQL中通过变量赋值语法，可以引用组件的属性：

```
vVar := :Item('<block>.<item>').ocx.<server>.<property>
```

其中<block>.<item>是数据块中的ActiveX控件项的名字，<server>是ActiveX服务器的名字(例如，OracleSpreadTable.1)，<property>是属性的名字。也可以使用由OLE导入程序生成的代码：

```
vVar := <package>.<property>(:Item('<block>.<item>').interface);
```

其中<package>是已生成的包的名字(例如OracleSpreadTable_lColumn)，<property>是对应于该属性的包函数(例如对列对象的位置特性)，<block>和<item>与前边一样，是数据块和ActiveX控件项的名字。有一个对应的用来设置值的过程语法：

```
<package>.<property>(:Item('<block>.<item>').interface, <value>);
```

其中<value>是在过程的参数表中说明的相应类型的数据值。

15.4 修改键值

这章最后的话题涉及到Oracle Developer Forms界面的一个小的但很重要的部分。Forms用户通过键盘上的键执行许多操作。可以控制这些键的赋值实现要做的任何事情。如果正在使用Developer Forms Server，在文本编辑器中编辑键资源文件。对于客户机 /服务器的表单，Oracle Terminal允许修改用于执行命令的键，既可以在生成器中，也可以在 Runtime程序中。

15.4.1 编辑Web键值

在安装Developer Server时，Oracle Installer增加安装一个称为FMRWeb.RES的资源文件：

```
#  FMRWEB.RES is the key definition file for webforms. The syntax is:
#
#    JFN : JMN : URKS : FFN : URFD   (whitespace ignored)
#
#      JFN = Java function number
#      JMN = Java modifiers number
#     URKS = User-readable key sequence (double-quoted)
#      FFN = Forms function number
#     URFD = User-readable function description (double-quoted)
#
#  JAVA FUNCTION NUMBER
#         33 = PageUp
#         34 = PageDown
#         35 = End
#         36 = Home
#         37 = LeftArrow
#         38 = UpArrow
#         39 = RightArrow
#         40 = DownArrow
#    65 - 90 = Ctrl+A thru Ctrl+Z (These will always have the control
#              modifier explicitly included, as well as any other
#              modifiers that might be used.)
#  112 - 123 = F1 thru F12
#          9 = Tab (Ctrl+I, without the control modifier)
#         10 = Return (Ctrl+J, without the control modifier)
#
#  JAVA MODIFIERS NUMBER
#  Equal to the sum of the values for the modifier keys:
#    0 = None
#    1 = Shift
#    2 = Control
#    4 = Meta
#    8 = Alt
#
#  FORMS FUNCTION NUMBER
```

```
#  The Forms function numbers match the function numbers found in a
#  typical Forms key binding file.
#
#  USER-READABLE STRINGS
#  The double-quoted strings appear when users click [Show Keys], and
#  are used for this purpose only. These strings can be translated as
#  needed. Note that the strings do not affect what actually happens
#  when end users press a particular key sequence.
#
9    : 0 : "Tab"             : 1  : "Next Field"
9    : 1 : "Shift+Tab"       : 2  : "Previous Field"
116  : 0 : "F5"              : 3  : "Clear Field"
38   : 0 : "Up"              : 6  : "Up"
40   : 0 : "Down"            : 7  : "Down"
33   : 0 : "PageUp"          : 12 : "Scroll Up"
34   : 0 : "PageDown"        : 13 : "Scroll Down"
69   : 2 : "Ctrl+E"          : 22 : "Edit"
10   : 0 : "Return"          : 27 : "Return"
76   : 2 : "Ctrl+L"          : 29 : "List of Values"
115  : 0 : "F4"              : 32 : "Exit"
75   : 2 : "Ctrl+K"          : 35 : "Show Keys"
83   : 2 : "Ctrl+S"          : 36 : "Commit"
118  : 1 : "Shift+F7"        : 61 : "Next Primary Key"
117  : 0 : "F6"              : 62 : "Clear Record"
38   : 2 : "Ctrl+Up"         : 63 : "Delete Record"
117  : 1 : "Shift+F6"        : 64 : "Duplicate Record"
40   : 2 : "Ctrl+Down"       : 65 : "Insert Record"
119  : 1 : "Shift+F8"        : 66 : "Next Set of Records"
1005 : 0 : "Down"            : 67 : "Next Record"
1004 : 0 : "Up"              : 68 : "Previous Record"
118  : 0 : "F7"              : 69 : "Clear Block"
66   : 2 : "Ctrl+B"          : 70 : "Block Menu"
34   : 1 : "Shift+PageDown"  : 71 : "Next Block"
33   : 1 : "Shift+PageUp"    : 72 : "Previous Block"
116  : 1 : "Shift+F5"        : 73 : "Duplicate Field"
119  : 0 : "F8"              : 74 : "Clear Form"
122  : 0 : "F11"             : 76 : "Enter Query"
122  : 2 : "Ctrl+F11"        : 77 : "Execute Query"
69   : 3 : "Shift+Ctrl+E"    : 78 : "Display Error"
80   : 2 : "Ctrl+P"          : 79 : "Print"
123  : 0 : "F12"             : 80 : "Count Query"
85   : 2 : "Ctrl+U"          : 81 : "Update Record"
121  : 3 : "Shift+Ctrl+F10"  : 82 : "Function 0"
112  : 3 : "Shift+Ctrl+F1"   : 83 : "Function 1"
113  : 3 : "Shift+Ctrl+F2"   : 84 : "Function 2"
114  : 3 : "Shift+Ctrl+F3"   : 85 : "Function 3"
115  : 3 : "Shift+Ctrl+F4"   : 86 : "Function 4"
116  : 3 : "Shift+Ctrl+F5"   : 87 : "Function 5"
117  : 3 : "Shift+Ctrl+F6"   : 88 : "Function 6"
118  : 3 : "Shift+Ctrl+F7"   : 89 : "Function 7"
119  : 3 : "Shift+Ctrl+F8"   : 90 : "Function 8"
120  : 3 : "Shift+Ctrl+F9"   : 91 : "Function 9"
```

使用任何一种编辑器，例如 Notepad，可以编辑这些代码和键的描述，给任何键分配任何功能。可以对这个文件中的任何一个键重新映射，改变第一列 (Java功能代号)和第二列(Java的限定符代号)中的代码，这两列合起来建立完整的键的设置。在靠近这个文件头的上边代码中

有详细的定义。不要改变 Forms功能代号(第4列)。可以通过对两个字符串的改变来修改在键的帮助屏幕上出现的内容，第一个字符串是键的描述，第二个字符串是功能描述。

15.4.2 用Oracle Terminal指定键

作为一个客户机/服务器的应用程序，运行的Oracle Developer Forms组件从一个资源文件中获取它的键的指定情况，例如从 Windows中的FMRUSW.RES文件。Oracle Terminal程序允许对那个文件的部分内容进行编辑，定义一个键的赋值。可以把键与应用程序的功能或键的触发器结合在一起，这样允许修改表单应用程序的键盘接口，以适应实际需要。也可以修改Form Builder的接口，以适应自己的任务。

注意 如果是通过一个Web浏览器运行应用程序，则不能够用Oracle Terminal改变键的赋值。

1) 在运行Oracle Terminal时，它给出一个文件提示。找到 Forms的资源文件 (通常是fmrusw.res)，单击OK。在Windows上，该文件是在FORMS60目录中。如果试图使用一些其他的资源文件，将会发现这些文件不具备用于 Oracle Terminal的正确的结构。然后Terminal显示主终端窗口(见图15-12)。

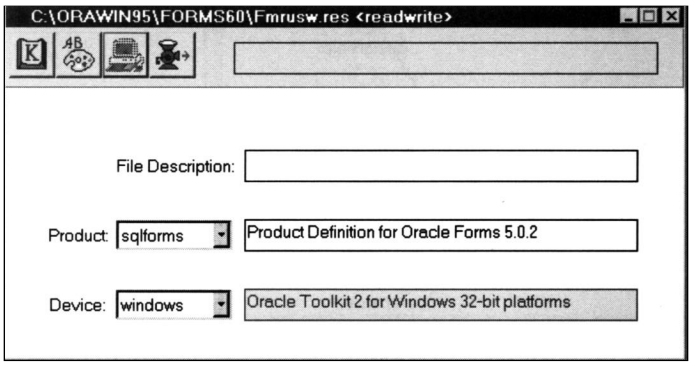

图15-12 主终端窗口

如果文件定义的产品和设备不止一个，这个窗口允许设置产品和设备，这对当前产品并不适用。

2) 单击Key Binding工具，利用Key Binding Editor(键赋值编辑器)对键的值进行编辑。

Key Binding Editor显示键分类树(见图15-13)。每一种分类都包含一个键赋值集合，与一种特定的操作类型相关联。中间级分类 (runform和design)具备可应用于该产品中的所有情况的操作，例如Show Keys、Help、Cancel和Exit。叶分类具备应用于特定情况的操作。

3) 为了看到键的赋值，双击分类的圆圈。

编辑器显示一个赋值表 (图15-14显示的是“ normal ”分类节点)。这个表给出了“ 操作 ”，如[Next Block]或[Scroll Up]，以及对其进行赋值的键，如CONTROL+D或SHIFT+F9。对一个操作可以有多行，指示可以通过任何一组键进行该操作。对一个操作的每一个键在表中要占据一个独立的行。

4) 要修改一个赋值，查找到该操作，用键入的内容覆盖该键。

可以用Row按钮(Duplicate Row、Insert Row、Delete Row)对行进行操作。也可以输入键，

把游标放入键的区域中，单击 Macro Mode按钮。然后可以输入该键，按下真正的键，而不是拼写出单词如 ”Control ”和 ”Shift ”。按照结果对话框的指导，完成宏的录制。

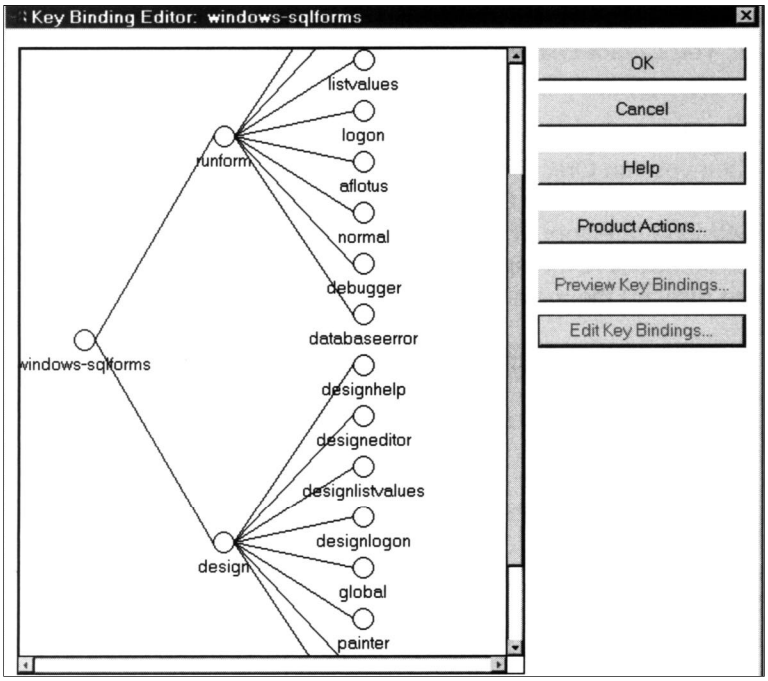

图15-13 Key Binding Editor

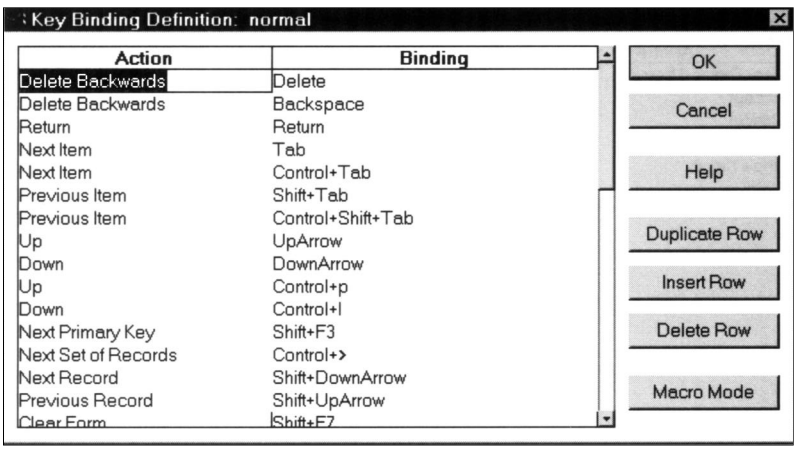

图15-14 Key Binding Definition Editor

设置键的触发器是一个更复杂的过程，因为需要定义操作。要在键赋值编辑器中的最高层windows-sqlforms上进行这项工作。

5) 双击windows-sqlforms节点，看到一个空的操作表。

6) 单击Insert Row按钮插入一行。

7) 插入对应于键触发器的操作 (例如，KEY-F3)。然后如前所述，输入该键的赋值。

8) 现在单击Product Actions按钮，调用Product Action Editor，它的图像与Key Binding Editor非常相似。

9) 双击sqlforms分类，看到Product Action Definition窗口(图15-15)。

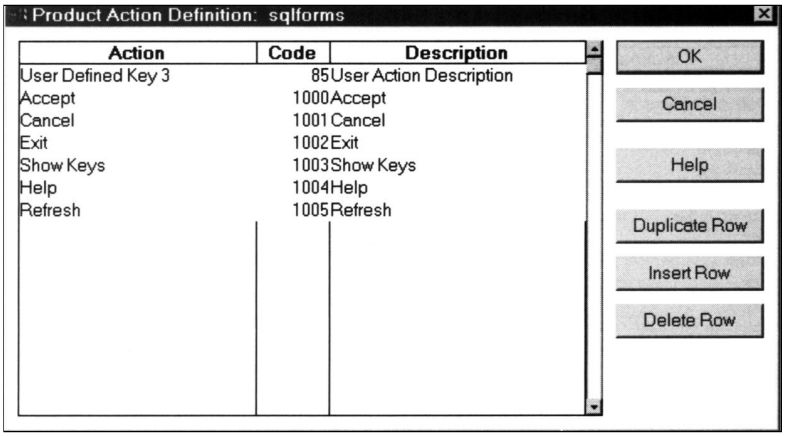

图15-15 Product Action Definition窗口

10) 输入操作(例如，[User Defined Key 3])、它对应的数值编码(从82到91对应‘0’键到‘9’键，在这种情况中编码(85)对应‘3’键)、以及显示在键赋值列表中的描述。单击OK。

11) 当完成定义赋值时，单击OK，释放键赋值编辑器，返回主窗口。

12) 保存键的赋值。

13) 单击Generate工具，生成与新的赋值对应的新的资源文件。

14) 再次保存，完成该过程。

15.5 用Oracle Translation Builder使应用程序国际化

由于因特网的出现，世界正变得越来越小。作为一个经验法则，如果从开始对应用程序就提出了语言可移植方面的要求，将会处于主动。进入90年代，世界已经变得非常地小。如果开始就知道必须使应用程序支持挪威语、俄语、阿拉伯语或者是汉语，就有可能设计出能够适应这些语言的应用程序，而不是去改变语言的要求。

为了便于把Oracle Developer的应用程序翻译成不同的语言，需要了解三个方面的内容：

- 本国语言支持(NLS)。
- 版面设计。
- 字符串转译。

15.5.1 Oracle中的本国语言支持

Oracle提供一种扩充功能，称为本国语言支持(NLS)，用于处理语言方面的差异。在《Oracle 8i数据库管理员手册》中可以了解这个功能的详细信息。如果将要在多种字符集和语言环境中工作，应该先从阅读Oracle文档中有关NLS的内容着手，因为应该清楚地知道Oracle能干些什么。

NLS提供对这些语言方面的支持：

- 存储在数据库中的、不同语言的文本和字符集。
- 来自Oracle和运行时程序的不同语言的消息。
- 对不同的语言相应格式的默认掩码。

■ 在SQL表达式和ORDER BY子句中使用关系操作符的不同字母表的排序和整理顺序。

■ 在表单、报表和图形中被显示的项的双向格式化。

在这种情况下，所谓“支持”，意味着Oracle透明地处理各种事情。作为一个应用程序开发人员，除了告诉Oracle希望使用的是哪种语言与地域以外，不需要做任何事情。在系统注册表中用NLS_LANG环境变量说明语言种类。

可以使用ALTER SESSION语句(Oracle SQL对ANSI标准的一种扩充)修改NLS的任何参数。如果愿意的话，可以练习一下控件对语言支持的敏感程度。本书建议对此少花力气，既是因为NLS功能并不符合已形成的ANSI标准，又是因为对于这种类型特性通常最好接受它的缺省值。不然，可能要不断地开足马力进行测试、调整和研究，才能得到正好是利比亚的阿拉伯语的方法，或者可能根据流程继续进行…

另一个根据流程继续进行的地方是使用缺省的格式掩码。所有的 Oracle Developer的产品都使用格式掩码来格式化数值型和日期型数据。本书中的各种例子建议用设置格式掩码的方法来说明日期型、数值型的特定值和货币值。如果将来提供的是基于语言的应用程序，可以不采用这个建议。使用缺省的格式掩码， NLS将自动地使用与在NLS_LANG中说明的语言合适的掩码。如果使用的是显式格式，则将失去自动处理功能。然后必须对所包括的每个事情进行编码，众所周知，这是很乏味的工作。

有几个特殊的可以使用的格式掩码字符， Oracle使用NLS能正确地对它进行翻译：

字符	含义
C	国际货币符号
L	本地货币符号
D	数字中的小数点
G	数字中的千位分隔符

例如，L999G999D99是$999,999.99的一种国际上通用的写法。 NLS也会把不同的日期组件翻译成本地的写法，例如月份名或者星期几。参看 Oracle Developer中Forms、Reports、Graphs等组件的联机文档，了解有关格式掩码的详细内容。

有一种特殊情况必须注意： PL/SQL表达式。如果在PL/SQL中因某种原因使用日期常量，应把它包含在一个To_Date函数中，以确保PL/SQL能够正确地处理它。例如，这里是一个依赖语言的表达式：

```
:Ledger.ActionDate := '1/1/01';
```

它依赖于语言，因为PL/SQL使用标准“美国式”格式掩码 “dd-MON-yy” 对该字符串进行转换，而不去管当前使用的语言。应该使用这样的代码去代替：

```
:Ledger.ActionDate := To_Date('01/01/01', 'DD/MM/YY');
```

15.5.2 布局设计

布局设计中关键的问题是基于窗口系统的可移植性问题：必须在不同时刻调整对象的不同的大小。对于一个基于窗口的系统，因为缺省字体的作用，按钮的大小是不同的。在一个系统中某些按钮是双线边框，而在另一个系统中又没有。

对于不同的语言，有几个布局问题：

■ 文本字符串的大小随被转换的字符串的长度而变化。

■ 格式化过的字符串(格式掩码区)的大小随语言而变化。
■ 界面组件“ widgets ”的大小随语言而变化。界面组件是显示在屏幕上包含有文本信息的各种图形对象，比如使用各种语言的单选按钮。
■ 标签的位置随它们所标注的区域而变化，均依赖于语言。

对所有这些情况， Oracle建议使用一个经验值，即预留 30%的扩展位置。如果能这样做，可以开发出这种语言占据大部分位置的原始的应用程序。

同样地，可以能设计出具有语言可移植性的屏幕、报表和图形的布局。尽量少使用样板，这种静态的资料将会在需要扩充区域时，增加要求转换的文本的每一位而得到。不要与图形一起混用模板，那样将会妨碍进行扩展。重新编排区域，这样不同的语言将显得是合理的，即使在所开发的应用程序中语言的阅读方向是反读的。

15.5.3 字符串的转换

开发的任何应用程序都将有大量的字符串。为了使字符串的转换减少到最低程度，应该在模板中尽可能少地使用字符串。例如，对需要成为国际上通用的应用程序，不要把大量的要使用的文本指导信息放到模板上。可以把指导信息放到数据库中，并在一个只读的区域中显示它们。

在Oracle Developer中有两种类型的字符串值得担心：

■ 模板文本，包含有标签和标题。
■ 在PL/SQL过程中的字符串常量和变量。

1. 模板文本

模板文本是在表单、报表或图形中的一种文本，这些文本是通过一个编辑器输入的，而不是从数据库填写的。设计者将这个文本用不同的格式保存到应用程序文件中 (或者是数据库中，如果数据库存在)。

模板文本的转换是一个查找该文本并将它翻译为所希望出现的语言的过程。至少有两三个方法来研究这个问题。首先，可以制作出对各种语言的应用程序文件 (表单、报表、图形)。其次，可以用显示区域来代替模板，并从数据库进行填写，用专门的 PL/SQL代码变换出合适的语言和数据的位置。本书建议使用前面的方案。维护应用程序的数据，和维护那些不得不增加到应用程序中的相对复杂的代码相比起来，要简单的多。

再次说明，虽然一味地减少文本是很困难的，但是限制这样文本的数量对国际化的应用程序而言仍是一个很好的策略。

2. Oracle Translation Builder

Oracle Translation Builder的工作是把在一个Oracle Developer的应用程序从一种基本语言翻译成为多种目标语言。它通过在数据库服务器上建立的一系列的数据库表进行这项工作。

要使用Oracle Translation Builder，首先必须把它的表安装到数据库中 (参见附录B有关安装数据库表的细节)。一旦已经安装了数据库，从任务条上启动 Translation Builder。

1) 选择Start| Oracle Developer 6.0|Translation Builder。这个菜单项启动该Builder，Builder显示一个初始化屏幕，提示选择是在一个工程中工作，还是运行 Quick Tour。选择在一个工程中工作(见图15-16)。

2) 第一个任务是建立一个标准的连接，从主菜单上选择File |New |DB Connection。图15-17

给出建立了Talbot连接的Navigator。

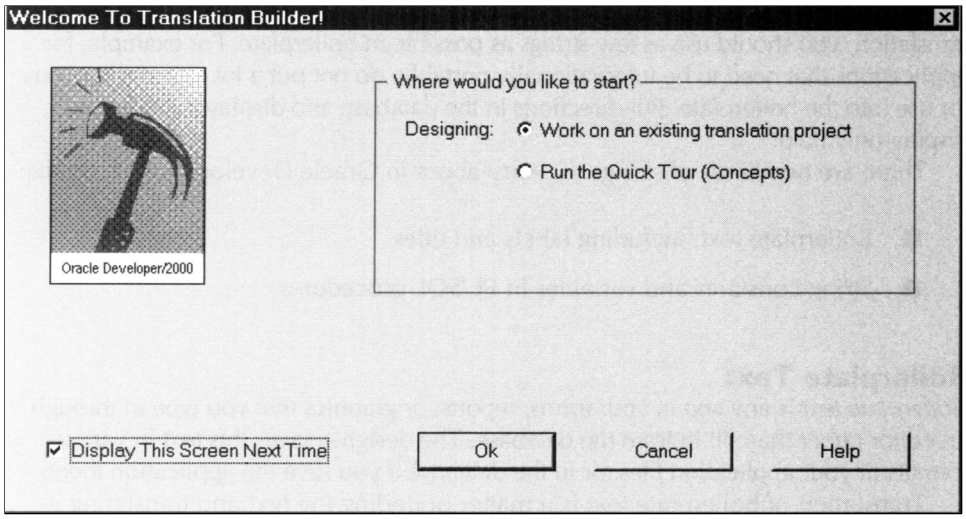

图15-16 选择在一个工程中工作

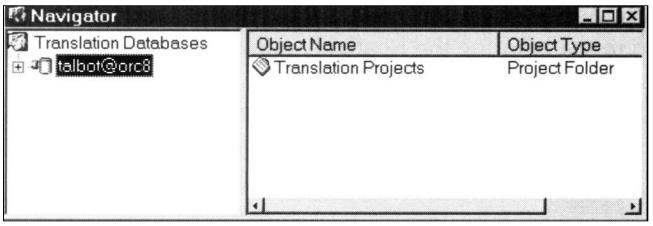

图15-17 Navigator窗口

3) 现在需要创建一个工程文件。这个工程文件对应于要捆绑到一个单个的应用程序中的一系列模块，或者是对应于其他的希望作为一个系统看待的 Forms模块组。单击 File |New |Project，在数据库连接之下创建一个工程。出现 New Project(新工程)对话框(见图15-18)。

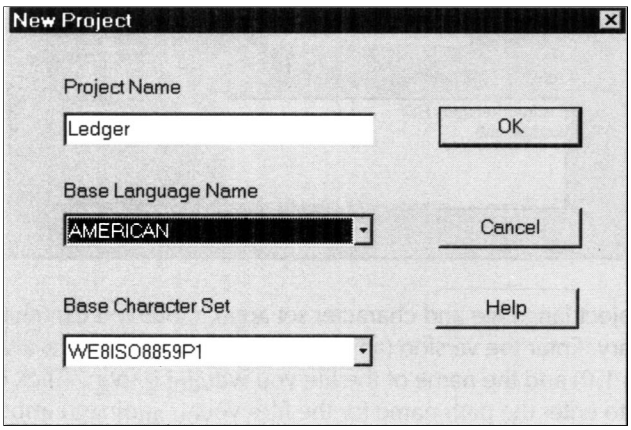

图15-18 New Project对话框

4) 要创建一个工程，必须指定一个名字和基本语言以及字符集。对于后两项，最佳的选择是NLS_LANG环境变量的当前设置值， NLS_LANG环境变量可以从与所有其他的 Oracle变

量一起的System Registry中找到。例如，在本例中， Talbot使用美国语言和WE8ISO8859P1字符集。单击OK，将会看到新的工程出现在工程文件夹中 (见图15-19)。

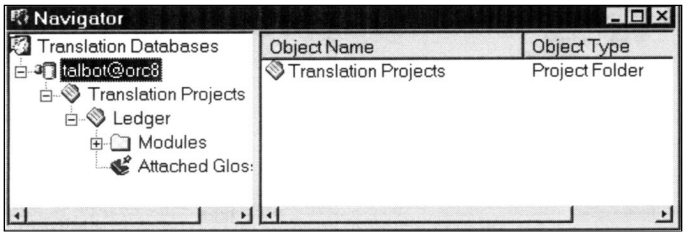

图15-19 工程文件夹中的新工程

5) 创建转换的第一个步骤是把基本版本引入到工程中。打开该工程，然后单击 Module |Import Strings菜单项或者 Import Strings 工具，将会看到这个Import Strings对话框(见图15-20)。

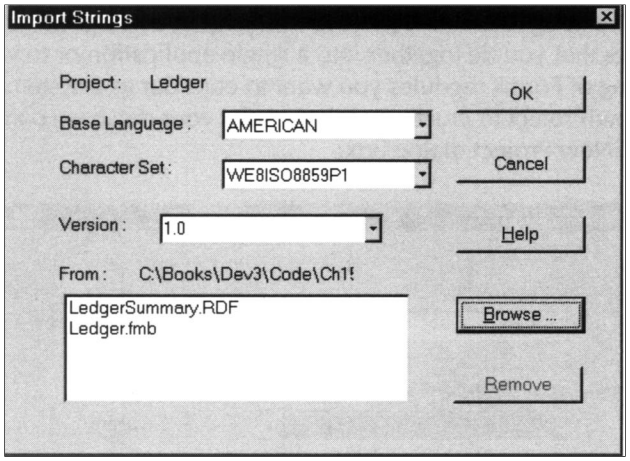

图15-20 Import Strings对话框

6) 出现工程的语言和字符集，但是如果需要可以修改它们。输入版本 (任何字符串，但是最佳的选择是一个版本号比如 1.0)和希望引入的文件的名字。单击 Browse按钮，输入将要引入的文件的路径名，并从出现的 Open File对话框中选择文件 (这里是Ledger的表单模块和LedgerSummary的报表模块)。会看到已选择的文件列表和那些文件的路径。单击 OK结束。Navigator(导航器)现在就包含有新的模块，如图 15-21所示，并可以开始进行转换。

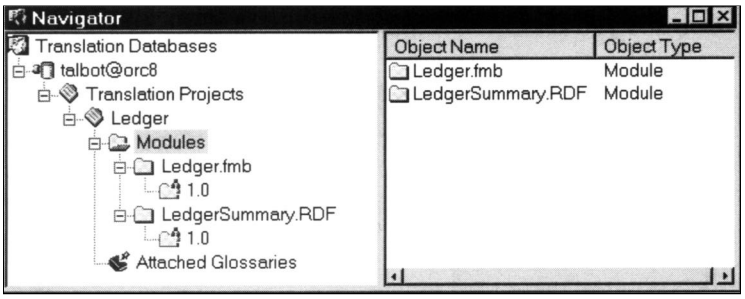

图15-21 包含有新模块的Navigator(导航器)

7) 打开一个模块的版本文件夹。用菜单项 Module |Add 或Add Translation工具条来增加

一个转换模块。显示Add Translation对话框(见图15-22)。

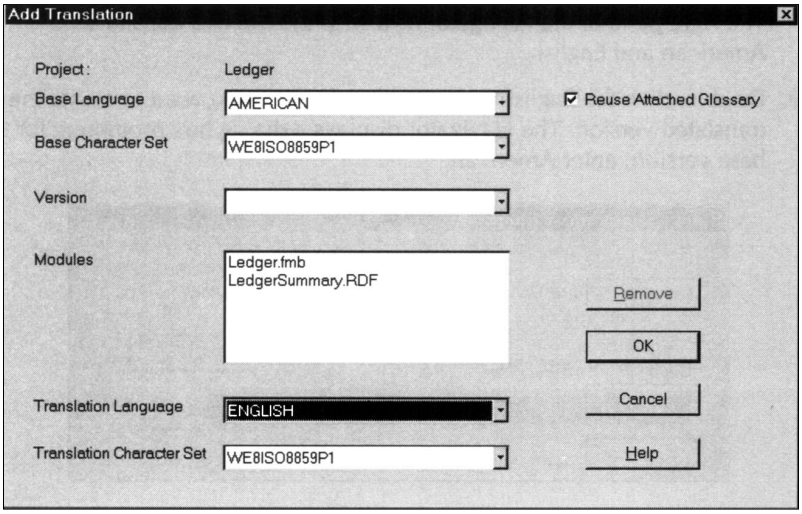

图15-22 Add Translation对话框

8) 在本例中， Talbot进行一个英式英语的转换。改变语言，由美语变成了英语。拉下Translation Language列表框并选择English(英语)。字符集保持不变。单击 OK创建新的转换，它出现在Navigator(导航器)中的版本下面(见图15-23)。

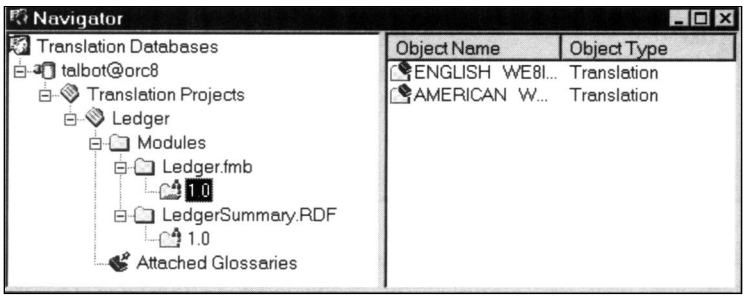

图15-23 Navigator下新的转换

注意 查阅有关Oracle的NLS特性的文档资料，了解语言和字符集的一个完整的列表。

现在在Navigator右边的显示这两个转换器：美语和英语。

9) 双击英语转换器，指定要创建该转换的版本。 Navigator(导航器)显示一个对话框，提示输入基本版本，输入美语(见图15-24)。

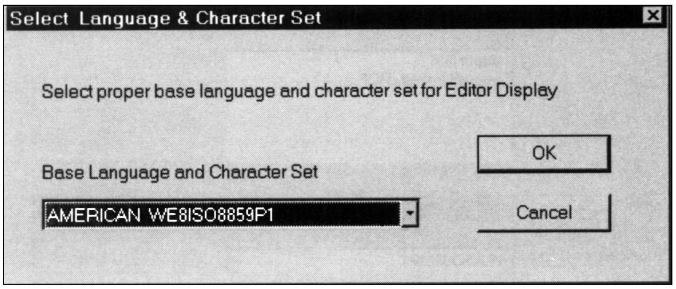

图15-24 在对话框中输入美语

现在会看到Translation Editor(转换编辑器)，有多排的字符串列表(见图15-25)。

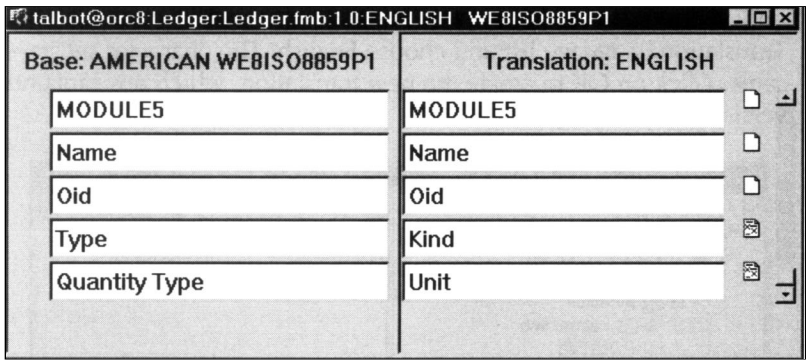

图15-25 多排字符串列表

10) 改变需要转换的字符串。对改变的字符串，会看到已转换图标显示在字符串旁边。在本例中，Talbot已经把Type转换为Kind，把Quantity Type转换为Unit，而这些词语将非常容易地被英国用户认识。当完成了编辑时，单击 Save工具，然后在Editor中的标题条中用关闭窗口控制按钮关闭该Editor。

接下来，进行最后一步 —— 把转换集成到原来的模块中。

11) 在Navigator中选择Ledger节点。单击Module |Export Strings菜单项或者是 Export Strings工具条。会看到Export Strings对话框，类似于Import Strings对话框(见图15-26)。

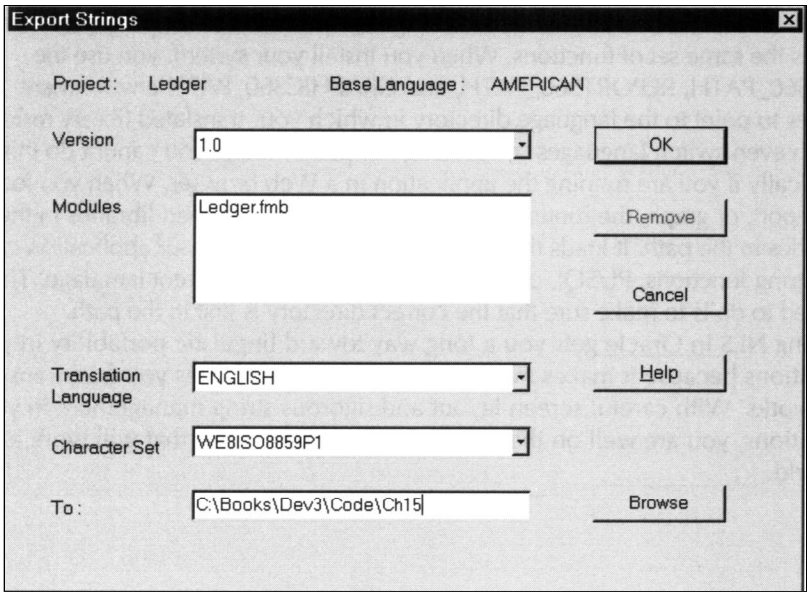

图15-26 Export Strings对话框

12) 用Remove按钮选择并删除不希望导出的任意模块。在本例中， Talbot已经删除了LedgerSummary报表，因为它没有被转换过。

13) 从下拉列表中选择导出的版本。输入要导出的转换的语言和字符集 (英语，被转换的语言)。单击OK。

转换生成器把转换结果写到模块文件中，现在模块文件包含有两种字符串的集合。当运行于合适的NLS_LANG设置时，看到的是转换后的字符串，而不是基本的字符串。

3. PL/SQL字符串

在许多时候，PL/SQL代码将通过警告或其他机制显示错误、警告或劝告信息。经常是通过自动地并置所生成的字符串来构造这些信息。如果希望制作一个国际通用的应用程序，不得不采取一些方法保证这些字符串以正确的语言出现。可以把它们进行编码，动态地检查当前的语言和回答的正确性，但是有一些比较好的方法可以达到同样的效果。

最有效的方法是把字符串集中到一个 PL/SQL库中进行处理。可以进行简单地编码，把信息与要查找的信息编号固定，或者可以开发专用函数，进行参数化的输入并产生一个合适的信息。为每一种希望支持的语言建立一个库，每个库包含同样的函数设置。当安装系统时，使用FORMS60_PATH、REPORTS60_PATH和GRAPHICS60_PATH环境变量，指出被转换库所驻留的语言目录。甚至可以通过重置这个路径对语言进行转换。可是如果在一个 Web浏览器中运行应用程序，则不能动态地这样做。当装载一个表单、报表或图形时，运行中的程序在路径的目录中查找附加的库。它装载所发现的第一个版本。当应用程序调用一个字符串函数时，PL/SQL调用为当前语言所定义的函数。这样，需要做的所有事情就是确信正确的目录是路径中的第一个目录。

使用Oracle中的NLS能够获得一个在应用程序中面向语言可移植的长远的方法，因为它使得许多字符集问题变为透明的：只是进行操作。在应用程序中严谨的屏幕布局和严格的字符串管理，将会使你的面对国际环境的应用程序取得成功。