Borland/Inprise核心技术丛书

# Kylix应用程序设计

（美） Cary Jensen
Loy Anderson 著
辰卓工作室 译



机械工业出版社
China Machine Press

本书介绍Borland公司推出的Linux平台RAD工具——Kylix，主要内容包括Kylix应用程序开发、数据库应用程序、Kylix先进特性精选和Internet应用程序开发等。

此外，本书还包含了两个附录，分别介绍如何下载和安装书中使用的示例代码以及如何访问本书的支持站点。

本书适合于所有对Kylix开发感兴趣的技术人员。



**本书版权登记号：图字：01-2000-1859**

**图书在版编目（CIP）数据**

Kylix应用程序设计 /（美）詹森(Jensen, C.)，（美）安德森(Anderson, L.)，著；辰卓工作室译. －北京：机械工业出版社，2002.2

（Borland/Inprise核心技术丛书）

书名原文：Building Kylix Applications

ISBN 7-111-09754-8

Ⅰ.K… Ⅱ.① 詹… ② 安… ③ 辰… Ⅲ.Linux操作系统-软件工具，Kylix-程序设计 Ⅳ.TP311.56

中国版本图书馆CIP数据核字（2001）第097304号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：瞿静华

北京忠信诚胶印厂印刷·新华书店北京发行所发行

2002年2月第1版第1次印刷

787mm×1092mm 1/16·31.75印张

印数：0 001-4 000册

定价：58.00元(附光盘)

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

# 译 者 序

2001年1月31日，Borland公司正式发布了Kylix，它是世界上第一种运行在Linux平台上的完整的RAD（快速应用程序开发）工具，其重要意义不亚于Borland推出的世界上第一个IDE（集成开发环境）——Turbo Pascal。正是以极为出色的Pascal编译器为基础的Delphi为Borland在全球赢得了众多开发者的支持，并成为Win32平台最主要的开发工具之一。Kylix是Delphi在Linux平台上的具体实现，它将极大地提高Linux平台上GUI应用程序（尤其是商业和企业的数据库应用）的开发效率，数以万计的技术熟练的Win32开发人员将能够轻松转向Linux，难以计数的Win32商业应用将可能被移植到Linux平台上，Linux在服务器市场取得重要份额之后，将向桌面计算市场发起新的凌厉攻势。

Kylix，这个名称来自于古希腊的一种器皿，是Borland公司一项庞大计划的代码名称，这项计划的主要目的就是将Win32平台上的Delphi和C++Builder完整地移植到Linux平台上，现在该项目已经取得重大成果——Kylix的正式发布。

Aberdeen集团研究主任BillClaybrook说："Linux要成为主流操作系统，需要专业的应用程序。Linux开发者需要在一个简单化、标准化的环境中以更快的速度构建这些应用程序，Kylix是惟一满足这种需要的开发工具。它建立在现有技术基础上，保持了与Windows同样的开发模式，从而大大地推动了Linux走入市场的进程。"

本书正是一本介绍如何使用Kylix进行应用程序设计的优秀书籍。在翻译本书的过程中，我们发现本书在结构安排上和内容的易理解性方面上都具有很大的优势。本书结构清晰，读者可以由浅入深地逐步进入Kylix世界。本书既可以作为一本教材来通读，亦可以作为一本参考书，在你进行Kylix开发遇到问题时，可以通过参阅本书的内容来寻求解决的方法。书中对当前流行的很多技术都有详细的讲解，比如数据库开发技术以及Web开发技术，而且代码详尽，例图丰富。相信通过对本书的学习，你将很快掌握Kylix的使用。

在术语的使用方面，我们参考了国内现有的Kylix及Delphi方面的书籍，力求避免混淆。

本书由胡季红、邓谆谆、姜玉琴、白建军组织翻译，参与翻译的有王玉峰、张江涛、刘志华和易会战。辰卓工作室全体人员参加了本书的翻译、校对、录入和排版工作。全书最后由肖国尊统稿。由于Kylix发布的时间较短，而且限于经验和水平，书中难免会存在一些不足和错误之处。敬请广大读者批评指正。

译 者

2001年9月20日

# 前　言

Kylix是近年来发布的最令人期待的开发工具之一。自从Borland于1999年第一次在Philadelphia召开的Borland研讨会上示范一个Linux编译器示例以来，关于Linux下的Delphi的传闻就广泛散播。在1999年9月25日，Borland做出了正式声明，表示他们正在创建一个基于Delphi产品的、针对Linux的RAD（rapid application development，快速应用程序开发）工具。这个软件的名称为Kylix。

Kylix的开发是一项非常宏大的事业，要求Borland公司耗费大量的资源。幸运的是，在2001年3月，当Borland最终推出Kylix后，漫长的等待终于结束了。

有下面一些原因使得Kylix非常重要。首先，Kylix是第一个针对Linux操作系统的RAD工具，这使得在Linux下构建经过编译的应用程序的速度大大提高。此外，Kylix对数据库应用程序以及Web服务器扩展都提供了强大的支持。这些特性使很多熟练的程序员能很容易地进入Linux开发，这是因为他们可以重用在创建Windows应用程序时已经做过的工作。最后，Kylix代表第一个新型的开发工具，这种开发工具有一个抽象层，将源代码与编译产品隔离开来。这个抽象层由TrollTech的Qt（发音为“cute”）C库提供，使用它可以将一个单独的文件编译为可在多个操作系统下运行。

也许多年以后才可以确定，但这是一个很好的机会——Kylix将极大地改变计算领域。尽管Linux在服务器配置方面已经是一个成功的平台，但是在桌面应用程序方面，它远远落后于微软的Windows操作系统。但是有了Kylix之后，不仅可以建立强大的Linux服务器，还能轻松地装配图形最终用户应用程序。换句话说，Kylix可能会使得Linux成为可行的工作站操作系统。

## 关于本书

本书的写作目的是为了使读者能够快速掌握Kylix。每章的内容都循序渐进，提供了创建各种应用程序的详细信息。

本书共分为四部分。第一部分是创建Kylix应用程序时使用的基本工具的概述，包括Component Palette（组件面板）、Object Repository（对象仓库）、代码编辑器以及调试器。

第二部分深入探讨了数据库应用程序的创建。包括对dbExpress的详细讨论，以及可能用到的其他访问数据的机制。同时，包含了使用Kylix的数据识别部件来创建并执行查询、读、写数据，显示主细（master-detail）关系以及创建定制的数据视图。这一部分还包括dbExpress内部的概述，并提供了创建定制dbExpress驱动程序的信息。

第三部分包含了Kylix的大量高级主题。例如，在第13章中，你将学习到如何创建并管理多线程的执行。第14章将讲述创建和使用共享对象（so）库。这一部分还包括创建部件和使用接口的简介。

第四部分将所有知识综合在一起，讨论了Internet以及Web相关的技术。你将学到如何创建动态Web站点，包括使用数据库服务器中的数据来创建Web页面。这一部分的最后两章提供了Internet Direct的概述，这是一个组件集合，可以用来在分布式网络计算环境下创建服务器以及客户端。

本书还包括两个附录，附录A列出了一些网址，从这些网址可以下载大量的示例代码。附录B提供了本书的支持站点的URL。通过访问这些站点，你可以经常获得Kylix的最新信息，以及本书的实时信息（包括必要的修订）。

尽管我们不愿提及，但是仍要说明的是：最初发布的Kylix有一些错误。这是很正常的，从来没有哪个软件是没有缺陷的。然而，Kylix中的这些错误使得本书中使用的一些工程不能够正常工作。例如，数据库应用程序中合计（aggregate）字段的例子，如果用最初发布的Kylix来创建，则不能够正确计算它的合计值。对每个包含进来的不能得到预期结果的工程，我们都加入了一个警告信息来描述这个问题，这样你就不会认为错误的产生是因为你自己做错了。

你也许在想，为什么要保留这样的例子呢？原因很简单：它们揭示了你希望知道的重要的技巧。此外，我们期望这些问题会被Borland通过Kylix补丁或者在以后的Kylix版本中加以解决。因此，我们包含进了创建这些工程的步骤，希望它们在你正在使用的Kylix上能够工作。

## 本书适合的读者

无论你是想马上开始创建Linux应用程序，或者只是想更好地理解如何使用Kylix来创建Linux应用程序，这本书都将适合你。

如果你不了解Kylix，而且没有使用过Delphi，那么你将会发现本书中所提供的解释和描述将会使你在这些方面得到迅速提高。如果你现在是一个Delphi开发人员，我们在书中包含了能够帮助你更好地理解这两个强大开发环境之间的关系的信息。

## 关于代码

几乎在本书的每一章中都给出了在Kylix中创建工程的逐步指导。你可以按照提供的步骤来构建这些工程，也可以从示例代码下载中获取完整的工程。类似地，在某些章中，我们会提到完整的工程而不是手工输入的内容。这是因为由于工程的复杂性，使得键入代码成为非常长或者很复杂的任务。这些工程也可以在示例代码下载中找到。如下所示的参考将告诉你正在讨论的工程的示例代码在哪个目录下可以找到。参见附录A中关于下载和安装示例工程的说明。

**Web参考** 你可以在sample路径下找到这个工程的代码。

由于是逐步地进行指导，因此前面的章节中出现的内容非常详细。在后面的章节里，我们假定你对一些基本的Kylix任务已相当熟悉，就不再仔细重复每个步骤，这样，我们能够迅速跳到问题的核心。

## 从哪里入手

有些书的写作目的是作为参考手册，你可以迅速查询你所感兴趣的特定主题。有些书则叙述性很强，需要一页一页地顺序读下去。这本书是以上两种作用的综合。我们特别希望本书能够更有魅力，能够吸引你愿意从头至尾地读完每一章。同时，我们还将这本书的内容组织进行了结构化，这样你就可以在短时间内迅速找到你所关心的特定信息。

我们还意识到，对于读者而言，并不会对每一章都有兴趣。例如，如果你已经是一个熟练的Delphi开发人员，那么你可能不需要读第1章和第2章的细节内容。我们在每一章的简介中指出了所包含的主题，而你可以根据自己的经验水平，决定是否可以快速浏览某些主题。

我们想说的是，在写这本书的过程中，我们度过了美好的时光。Kylix是令人惊奇的，我们认为你也会像我们这样，在使用它的过程中发现乐趣。

# 目　录


译者序
前言

## 第一部分　Kylix应用程序开发

第1章　Kylix简介 ……1
1.1　Kylix概述 ……1
1.1.1　使用组件方式构建应用程序 ……2
1.1.2　Kylix执行代码生成 ……3
1.1.3　Kylix编程环境是事件驱动的 ……5
1.1.4　面向对象的软件开发 ……5
1.1.5　理解Kylix的开放工具API ……5
1.1.6　快速应用程序开发 ……6
1.2　Delphi开发者眼中的Kylix ……6
1.2.1　Kylix和Linux ……7
1.2.2　CLX概述 ……8
1.2.3　一些缺陷 ……10
第2章　创建应用程序 ……13
2.1　创建一个基本的应用程序 ……13
2.1.1　创建一个新的工程 ……13
2.1.2　放置并配置组件 ……16
2.1.3　书写事件处理程序 ……18
2.1.4　添加菜单 ……20
2.1.5　使用附加的窗体和对话框 ……22
2.1.6　在命令提示符下运行完成的工程 ……30
2.2　Kylix工程中的文件 ……31
2.2.1　工程源文件 ……31
2.2.2　单元 ……31
2.2.3　窗体文件 ……32
2.2.4　被编译单元 ……33
2.2.5　可执行文件 ……33
2.2.6　被修改文件 ……33
2.2.7　工程选项文件 ……33
2.2.8　工程编译器设置文件 ……33
第3章　Kylix的RAD框架 ……34
3.1　使用可视窗体继承 ……34
3.1.1　改变继承对象的属性 ……37
3.1.2　在继承对象中覆盖事件处理程序 ……37
3.1.3　源于当前工程 ……39
3.1.4　定义共享的Object Repository ……39
3.2　设计动作列表 ……39
3.3　使用帧 ……41
3.3.1　创建帧 ……42
3.3.2　使用帧 ……43
3.3.3　覆盖包含组件的属性 ……44
3.3.4　包含的对象事件处理程序 ……45
3.3.5　覆盖包含的对象的事件处理程序 ……47
3.3.6　帧和资源 ……48
3.3.7　简化帧的使用 ……49
3.3.8　将帧转化为实际的组件 ……51
3.4　使用数据模块 ……51
3.4.1　使用数据模块来共享组件 ……52
3.4.2　数据模块的局限性 ……53
第4章　使用并配置编辑器 ……55
4.1　编辑器键映射 ……55
4.2　选择编辑器按键 ……57
4.2.1　键宏记录 ……57
4.2.2　块缩进与不缩进 ……58
4.2.3　使用书签 ……59
4.2.4　将To-Do列表项作为书签来使用 ……59
4.2.5　类导航 ……60
4.2.6　代码浏览 ……60
4.2.7　类完成 ……62
4.2.8　递增查找 ……64
4.2.9　查找匹配分隔符 ……64
4.2.10　列操作 ……65
4.3　Code Insight ……65

4.3.1 代码完成 ……65
4.3.2 代码参数 ……66
4.3.3 工具提示表达式求值 ……67
4.3.4 工具提示符号洞察 ……67
4.3.5 代码模板 ……67
4.4 编辑器键绑定 ……70
4.4.1 声明键绑定类 ……71
4.4.2 实现键绑定类 ……72
4.4.3 声明并实现Register过程 ……75
4.4.4 创建并安装新的设计时包 ……76
第5章 调试Kylix应用程序 ……78
5.1 集成调试器 ……78
5.1.1 工具提示表达式求值 ……78
5.1.2 调试窗口 ……79
5.1.3 Run菜单 ……84
5.2 禁用调试器 ……84
5.2.1 指示调试器忽略引发的异常 ……86
5.2.2 指示Kylix忽略特定异常 ……86
5.3 断点概述 ……87
5.3.1 源断点 ……87
5.3.2 其他断点类型 ……91
5.3.3 在Kylix会话之间持续断点 ……93

**第二部分 数据库应用程序**

第6章 数据库应用程序 ……95
6.1 理解数据库 ……96
6.1.1 数据库与表 ……96
6.1.2 数据库与SQL ……97
6.1.3 其他与数据库相关的概念 ……97
6.1.4 本书中使用的数据库 ……98
6.1.5 创建数据库与表 ……98
6.2 利用Kylix开发数据库概述 ……99
6.3 数据库应用程序中使用的组件 ……106
6.3.1 数据识别控件 ……106
6.3.2 数据访问组件 ……108
6.3.3 dbExpress组件 ……109
6.4 dbExpress概述 ……110
6.5 创建简单的数据库应用程序 ……116
第7章 使用数据识别控件 ……120
7.1 数据识别控件的常规配置 ……120
7.1.1 Tab顺序与可视控件 ……121
7.1.2 修改Tab顺序 ……122
7.2 配置Kylix的数据识别控件 ……123
7.2.1 准备一个启用数据的组件模板 ……123
7.2.2 使用DBGrid ……124
7.2.3 使用DBNavigator ……142
7.2.4 使用DBText ……143
7.2.5 控制DBEdit ……144
7.2.6 使用DBMemo ……146
7.2.7 使用DBListBox ……147
7.2.8 使用DBComboBox ……149
7.2.9 使用DBCheckBox ……150
7.2.10 使用DBRadioGroup ……150
7.2.11 使用DBLookupListBox与DBLookup ComboBox ……151
第8章 使用TField ……154
8.1 字段概述 ……154
8.2 理解永久性字段 ……159
8.2.1 创建永久性字段 ……160
8.2.2 配置永久性字段 ……160
8.3 在程序运行时使用字段 ……167
8.3.1 读写数据集的字段 ……167
8.3.2 编写OnValidate事件处理程序 ……171
8.3.3 字段的访问、性能与维护问题 ……172
8.4 创建新的永久性字段 ……175
8.4.1 创建计算字段 ……176
8.4.2 创建查找字段 ……177
8.4.3 定义合计字段 ……178
第9章 使用单向数据集 ……182
9.1 单向数据集 ……182
9.2 使用单向数据集 ……183
9.2.1 定义SQL连接 ……183
9.2.2 返回记录集的单向数据集 ……184
9.2.3 不返回结果集的单向数据集 ……186
9.3 使用参数化查询 ……187
9.3.1 创建主细视图 ……189
9.3.2 通过链接查询创建主细视图 ……192

9.3.3 准备单向查询 ……193
9.4 执行存储过程 ……194
9.4.1 返回各个值的存储过程 ……195
9.4.2 返回数据集的存储过程 ……196
第10章 使用内存数据集 ……199
10.1 内存数据集 ……199
10.2 创建内存表与索引 ……200
10.3 保存数据与取消修改 ……206
10.4 按索引排序 ……207
10.5 使用范围 ……212
10.6 过滤 ……214
10.6.1 根据属性进行过滤 ……214
10.6.2 过滤器选项 ……216
10.6.3 使用OnFilterRecord事件处理程序 ……216
10.6.4 使用过滤器进行导航 ……217
10.7 搜索数据 ……218
10.7.1 使用FindKey与FindNearest ……218
10.7.2 使用GotoKey与GotoNearest ……219
10.7.3 使用Locate与Lookup ……219
10.8 记录级别的有效性检查 ……224
第11章 高级数据库技术 ……226
11.1 应用更新的艺术 ……226
11.1.1 利用内存数据集编辑数据 ……226
11.1.2 基于记录状态进行过滤 ……230
11.1.3 判定记录状态 ……233
11.1.4 从Change日志中删除修改 ……234
11.1.5 刷新记录 ……237
11.2 自定义更新过程 ……238
11.2.1 通过DataSetProvider属性控制更新 …239
11.2.2 向更新过程添加代码 ……242
11.2.3 处理更新错误 ……246
11.3 实用技术 ……248
11.3.1 使用SQL监视器 ……248
11.3.2 克隆游标 ……249
11.3.3 以每次保存一条记录的方式保存数据 ……249
第12章 编写dbExpress驱动程序 ……253
12.1 理解数据库提供商客户端 ……253
12.1.1 初始化环境 ……254
12.1.2 连接数据库服务器 ……254
12.1.3 初始化语句句柄 ……254
12.1.4 准备SQL语句 ……255
12.1.5 传递运行时参数 ……255
12.1.6 执行SQL语句 ……256
12.1.7 绑定记录缓冲区 ……256
12.1.8 读取记录 ……257
12.1.9 释放句柄并且断开连接 ……257
12.2 dbExpress核心实现 ……258
12.2.1 SQLDriver类 ……258
12.2.2 SQLConnection类 ……261
12.2.3 SQLCommand类 ……267
12.2.4 SQLCursor类 ……286
12.2.5 SQLMetaData类 ……292
12.3 dbExpress接口源文件 ……292

## 第三部分 Kylix先进特性精选

第13章 多线程应用程序 ……293
13.1 多线程的优点 ……295
13.2 创建多线程应用程序 ……296
13.3 线程同步 ……307
13.3.1 使用Synchronize ……307
13.3.2 使用临界区 ……308
13.3.3 等待线程 ……310
13.3.4 使用事件对象 ……313
13.3.5 锁对象 ……315
13.4 其他线程技术 ……315
13.4.1 线程局部变量 ……315
13.4.2 调试线程 ……316
13.4.3 对数据库的多线程访问 ……317
第14章 共享对象库 ……319
14.1 共享对象库概述 ……319
14.2 创建一个共享对象库的例子 ……320
14.2.1 将函数书写到Export ……321
14.2.2 控制共享对象库名称 ……323
14.3 加载共享对象库存例程 ……326

14.4 创建共享对象库导入单元 ……328
14.5 动态加载共享对象库 ……330
14.5.1 声明变量 ……331
14.5.2 动态加载共享对象库 ……332
14.5.3 获取函数或者过程的地址 ……332
14.5.4 释放共享对象库 ……332
14.6 调试共享对象库 ……335
14.6.1 使用宿主应用程序进行调试 ……335
14.6.2 使用工程组进行调试 ……336
14.7 共享对象库初始化及退出 ……337
14.7.1 定义初始化代码 ……337
14.7.2 定义退出过程 ……337
第15章 构建组件 ……338
15.1 对象概述 ……338
15.1.1 从记录转向类 ……338
15.1.2 封装与成员可见性 ……341
15.1.3 定义运行时接口 ……342
15.1.4 继承与多态 ……345
15.2 组件创建简介 ……347
15.3 一个简单组件示例：定义新的属性缺省值 ……348
15.3.1 使用Component向导 ……349
15.3.2 覆盖方法 ……350
15.3.3 实现覆盖构造函数 ……351
15.3.4 测试新的组件 ……352
15.3.5 安装组件 ……354
15.4 创建设计时包 ……355
15.5 有关属性的一个例子 ……357
15.5.1 定义成员域 ……357
15.5.2 定义属性 ……358
15.5.3 定义方法 ……358
15.5.4 覆盖现有方法 ……359
15.5.5 实现覆盖方法 ……360
15.5.6 创建事件属性 ……361
15.5.7 在完成组件时需要注意的细节问题 ……362
15.6 相关主题：提高属性可见性 ……369
第16章 使用接口 ……372
16.1 接口概述 ……373
16.2 为何使用接口 ……373
16.3 声明接口 ……375
16.4 实现接口 ……377
16.5 接口和方法解析 ……384
16.6 通过授权实现接口 ……384
16.7 接口实例：数据泵 ……387

## 第四部分 Internet应用程序开发

第17章 Internet技术概述 ……393
17.1 协议、技术与专业术语 ……393
17.2 RFC ……394
17.3 IP地址 ……394
17.3.1 域名 ……394
17.3.2 TCP/IP与UDP ……395
17.3.3 套接字与端口 ……395
17.3.4 SGML ……396
17.3.5 HTML ……396
17.3.6 FTP ……397
17.3.7 HTTP ……397
17.3.8 MIME ……397
17.3.9 万维网 ……397
17.3.10 Web服务器 ……398
17.3.11 Web浏览器 ……398
17.3.12 Apache ……398
17.3.13 CGI与DSO ……399
17.3.14 SSI ……399
17.4 Web服务器扩展概述 ……400
17.5 Web交互简介 ……400
17.5.1 URL的组成 ……401
17.5.2 请求类型 ……403
17.6 使用HTML ……403
17.7 使用HTML将数据提交给Web服务器扩展 ……404
17.7.1 图片标记 ……405
17.7.2 锚标记 ……405
17.7.3 HTML表单 ……406
17.8 编译Apache服务器来使用DSO ……411
第18章 使用Web Broker编写Web服务器

扩展 ……414
18.1 创建简单的CGI Web服务器扩展 ……414
18.2 安装、使用CGI服务器 ……418
18.2.1 添加LD_LIBRARY_PATH环境变量 ……418
18.2.2 将CGI应用程序写到ScriptAlias目录中 ……420
18.2.3 从浏览器执行CGI应用程序 ……421
18.3 创建、配置一个简单的Apache DSO …422
18.3.1 创建DSO工程 ……422
18.3.2 使用生成器 ……423
18.4 安装、使用DSO库 ……427
18.4.1 将DSO库写到Apache目录中……427
18.4.2 将DSO添加到httpd.conf中……427
18.4.3 关闭和启动Apache ……428
18.4.4 从一个浏览器执行DSO库 ……429
第19章 高级Web Broker主题 ……431
19.1 从HTML表单中获取数据 ……431
19.2 创建基于Web的数据库应用程序 ……435
19.2.1 Web服务器扩展与并发性 ……435
19.2.2 数据识别提供者 ……436
19.2.3 在TableProducer中格式化单元 ……440
19.3 Cookie与WebRequest对象 ……441
19.3.1 获取和设置cookie ……442
19.3.2 使用cookie和重定向 ……443
19.4 WebRequest的内容 ……446
19.5 调试Web服务器扩展 ……449
19.5.1 将CGI工程转换成DSO工程 ……449
19.5.2 调试DSO工程 ……450
第20章 Internet Direct概述 ……453
20.1 什么是Internet Direct ……453
20.2 Internet Direct组件 ……455
20.2.1 Indy客户端组件 ……455
20.2.2 Indy服务器组件 ……457
20.2.3 Indy Miscellaneous组件 ……459
20.2.4 下载更新的Internet Direct组件 ……461
20.3 使用Internet Direct组件 ……461
20.3.1 Internet Direct客户端是如何工作的 ……461
20.3.2 使用TIdAntiFreeze ……462
20.3.3 Internet Direct服务器的工作方式 …463
20.3.4 使用线程管理器 ……464
20.4 Internet Direct许可证 ……464
20.4.1 Indy修改的BSD许可证 ……464
20.4.2 Indy MPL许可证 ……465
20.4.3 在Kylix应用程序中遵守Indy许可证 ……465
20.5 技术支持 ……465
第21章 使用Internet Direct ……467
21.1 客户端和服务器的创建顺序 ……467
21.2 一个简单的服务器示例 ……468
21.2.1 线程和IdTCPServer ……470
21.2.2 阻塞调用与并发 ……470
21.2.3 OnExecute和异常 ……471
21.3 数据库服务器示例 ……472
21.3.1 创建数据库服务器 ……472
21.3.2 创建数据库客户 ……474
21.4 在客户端处理异常 ……476
21.4.1 从客户端检测连接中断 ……476
21.4.2 使用多线程客户端测试服务器 …477
21.5 使用TIdSMTP发送邮件 ……480
21.5.1 创建消息 ……480
21.5.2 创建TIdSMTP客户……481
21.5.3 在线程中创建客户端 ……482
21.5.4 初始化客户端线程 ……484
21.5.5 从一个线程中更新用户界面 ……484
21.6 ZIP编码查找服务器和客户端 ……486
21.6.1 定义ZIP编码协议 ……486
21.6.2 ZIP编码服务器 ……487
21.6.3 ZIP编码客户端 ……489
21.7 创建一个控制台服务器 ……491
21.7.1 创建控制台服务器的例子 ……491
21.7.2 测试纯文本控制台服务器 ……493

## 附 录

附录A 安装示例代码 ……495
附录B 本书的Web支持站点 ……496

# 第一部分 Kylix应用程序开发

**学习目标：**

- 通过组件面板来放置组件，并使用属性来配置组件。
- 将事件处理程序与组件的事件属性相关联。
- 建立一个基本的Kylix工程。
- 在工程中创建并使用帧。
- 理解ActionList组件的作用。
- 能够在工程中放置一个活跃断点。
- 使用Kylix的在线帮助获得关于编辑器按键的更多信息。
- 理解自定义键绑定的目的。

## 第1章 Kylix简介

**本章包含的内容：**

- Kylix概述
- Delphi开发者眼中的Kylix

Kylix是用来创建Linux应用程序的快速应用软件开发（rapid application development，RAD）环境，它具有基于组件、代码生成、事件驱动、面向对象、界面开放、支持数据库等特点。这些特点也许正是你对它感兴趣的原因。那么，这些特点都有什么含义，我们为何要关心这些特点呢？本章的目的就是为你提供Kylix的概貌，例如，什么是Kylix，它为什么这么重要。

此外，本章还有另外一个目的。某些Kylix开发者可能对Borland公司基于Object Pascal开发环境的基于微软Windows的版本——Delphi比较熟悉。如果你属于这一类开发者，那么本章的第二部分将会适合你阅读。在第二部分中，你能够了解到Kylix和Delphi的类似之处，以及两者的差别所在，这样你就能够迅速掌握Kylix。

### 1.1 Kylix概述

概括地说，使用Kylix可以迅速且容易地为Linux操作系统创建软件。你可能要开发两类软件，它们分别是独立的应用程序和共享对象库。独立的应用程序包括那些显示用户界面并在X服务器下运行的应用程序，这里的X服务器包括GNOME（GNU network model environment，GNU网络模型环境）以及KDE（K development environment，K开发环境）图形用户界面，还有那些可以在命令提示符下运行的控制台应用程序。共享对象（so）库是被其他应用程序载入并执行的二进制可执行文件，不能够通过在命令提示符下键入命令来直接运行共享对象库。

Kylix生成的可执行文件使用x86指令集的ELF格式（executable and linking format，可执行及

链接格式）。换句话说，这些二进制可执行文件在与Intel兼容的CPU下的Linux版本中可以运行。

能够编译真正Linux ELF格式文件的方法并不是首次出现。通过使用公开源码的GNU（另一个类似UNIX的操作系统）C编译器，你就可以编译Linux ELF格式文件。这个编译器（还有其他编译器）一直都与多数Linux安装在一起。在Kylix下的真正变化在于生成这些文件的方式与C编译器不同。从这个意义上讲，Kylix是一个革命性的产品，是Linux下第一个迷人的应用程序。

下面几节将对Kylix开发的主要特点进行讨论。

### 1.1.1 使用组件方式构建应用程序

Kylix是一个基于组件的开发环境。基于组件的软件开发可谓是近十年来软件编程方面最重要的进步，它使得应用程序开发的速度显著加快，而且开发的应用程序更加易于维护。

基于组件的软件开发的特点是存在大量的设计时（design-time）配置，该配置涉及到组件的放置位置以及通过属性对组件进行的配置。从面向对象的意义上讲，组件是预定义的对象，它们通过一个简单的矩形或者一个精致的图形来以可视的方式表示。图1-1显示了一个窗体（Kylix设计器中一个包含可视及非可视组件的窗体），如同在Kylix设计器（designer）中所显示的那样。

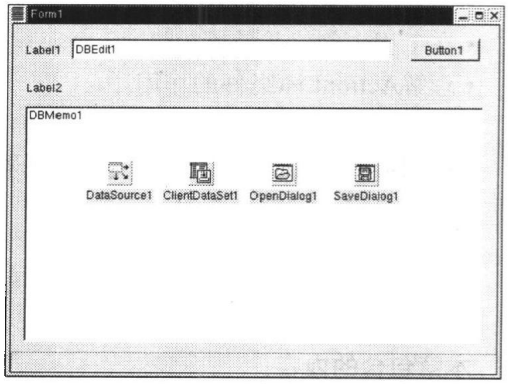


图 1-1

**注意**　不是所有基于组件的开发工具都是面向对象的。然而，Kylix既是基于组件的，又是面向对象的。

图1-1中的窗体包含了很多个组件。注意，它们中有些被显示为包含图形的小矩形，有些则显示为用户界面元素，比如按钮、标签以及文本框。该窗体中的非可视组件也显示了一个标题来指示自身。然而，组件显示标题并不是缺省的设置。为了启用Kylix中的组件标题，必须在Environment Options对话框中的Preferences页中选中Show component captions复选框。可以通过在Kylix的主菜单中选中Tools|Environment Options来显示该对话框，然后选择Preferences标签。

在设计时的可选组件主要位于Kylix的Component Palette（组件面板）上，在缺省情况下，Component Palette显示在Kylix的主窗体中（也可以将它从主窗体中拖出来，显示为浮动工具栏）。使用Kylix还可以将其他组件添加到Component Palette中，包括那些自己生成的组件以及从第三方开发商那里购买的组件。图1-2显示了停靠在主菜单下面的Component Palette，这也正是它的缺省位置。

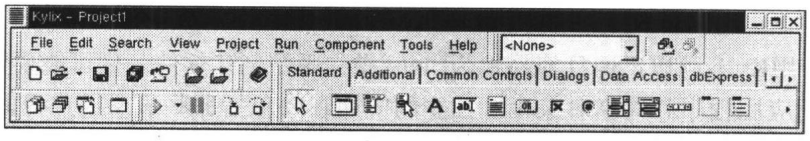


图 1-2

Component Palette由许多页组成，每页中包含着相关的组件集合。例如，很多同基本用户界面相关的组件出现在Standard页中，而dbExpress页则包含一些非可视化组件，你可以通过这些组件连接到远程数据库管理服务器（remote database management server，RDBMS）上（如Oracle、

MySQL或者InterBase），并从数据库中获取数据。

一旦将组件放置到窗体中，你就可以通过设置它的一个或者多个设计时属性来配置它。当你选中窗体中的一个组件时，它的所有属性就会显示在Object Inspector（对象检查器）中，如图1-3所示(使用对象检查器配置对象的设计时属性)。除了从窗体上选取组件外，还可以使用Object Inspector的Object Selector（对象选择器），Object Selector是一个列出了当前窗体上所有对象的组合框。

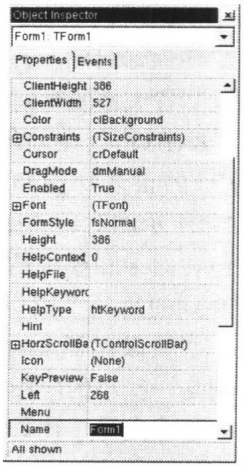


图 1-3

有些组件一旦放置到窗体中后，就不再需要进行更多的配置。例如，一旦将组件面板中Common Controls页中的状态栏放置到窗体后，就不需要再去对它进行配置了。当然，这是一个例外。多数开发者会改变Name属性的值，给组件一个能够更好地反映该组件作用的名称。

这对代码中将会引用到的组件尤为重要。赋予这些组件有意义的名称，可以使代码有更好的易读性。

Object Inspector是一个由两个标签页组成的对话框，如图1-3所示。这两页分别是Properties和Events。两者都列出了被选中组件的设计时属性。其中，Events页列出了事件属性，Properties页列出了所有其他的属性。

可以使用Properties页中的属性来改变选定组件的外观以及行为。例如，设置一个按钮的Caption属性可以定义显示在按钮上的文本。同样，可以通过设置组合框的Style属性来指示用户是将内容键入到组合框中，还是被限制为只能在组合框的下拉菜单中进行选择。

尽管基于组件的开发环境允许你通过使用Properties页中的属性来控制应用程序的外观和行为，然而你仍然需要书写一些在运行时执行的代码。这些代码与Object Inspector的Events页中的事件属性相关联，这些内容将在本章稍后部分进行详细讨论。基于组件开发的方便之处在于它减少了代码的书写量。这样，可以更快、更简便地创建应用程序。

基于组件开发的第二个主要优点在于易于维护。这在很大程度由于用户必须书写代码数量的减少。换句话说，通过书写较少的代码，使得只有较少的代码需要修改和调试。基于组件的应用程序易于维护的另一个原因在于组件所使用的自归档（self-documenting）特性。很多代码使用点号来引用组件以及它们的方法和属性。点号包括了方法、属性或者域的全值引用。点号是用来把对象和类引用同方法、属性、域分隔开来的符号。例如，Button1.Caption，其中Button1是TButton的一个实例，Caption是它的属性。由于方法和属性名称使用自然语言，因此你的代码很接近于描述你正在实现的效果。例如，考虑如下代码段：

```
if CustomerDataSet.IsEmpty then
  EmptyDataSetMenuItem.Enabled := False;
```

如果你知道CustomerDataSet是一个组件，它指向数据库中的一个数据表，而EmptyDataSetMenuItem是一个菜单项，那么即使对Object Pascal不是十分了解，也能知道这段代码将会在数据集中没有数据的时候禁用该菜单项。易读、易理解的代码易于维护和调试。

### 1.1.2 Kylix执行代码生成

当你开始在Kylix中创建一个应用程序时，首先需要创建一个新的工程。可以通过在Kylix的主菜单中选择File|New Application来创建工程。Kylix则会相应地生成很多文件，包括一些Object

Pascal源文件，它们提供了应用程序的基础。尽管你可以手工输入这些文件的代码，但这样做将会花费许多时间，而且还要对所输入的代码进行测试。而Kylix生成的代码则是无错的，同时还可以节省大量的时间。

Kylix不仅在创建工程时生成代码。实际上，Kylix的很多功能和特性都会生成代码。例如，图1-4所示的对象仓库（Object Repository）是一个包含向导和模板的工具（在对象仓库的New页中看到的每一个图标均对应一个向导）。模板就是一些被保存下来的、易于重用的代码段，而向导则是一些小的应用程序，它们能够生成代码。

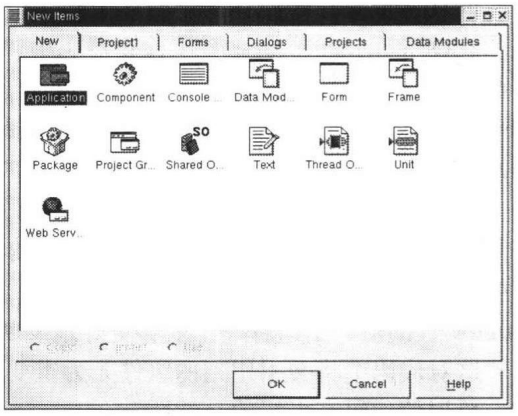


图 1-4

某些向导（比如Thread Object向导）可以将生成的代码添加到当前工程中去。某些向导，包括Web Server Application向导，则可以生成一个完整的工程。你可以使用已经生成的代码作为起点来增强你的应用程序。换句话说，你只需将附加代码添到向导生成的代码中去。然而，向导总可以为你节省大量时间，使得你能够将精力集中在增加某些改进上面，而不是将时间耗费在书写支持代码上面。

Kylix执行的另一个主要的代码生成动作同分配给事件属性的方法相关。如果你查看选定组件的Object Inspector中的Events页，并且双击位于你希望指派方法到其上的事件属性右侧的栏，那么Kylix将会生成一个新的方法，增加一个该方法的前向声明到窗体的类声明当中去，并在窗体的implementation部分增加用于实现该方法的实现块。它还为事件属性指定生成的方法的名称。你所需要做的就是在对应的事件发生时增加你所希望执行的代码。增加事件处理程序将在第2章进行详细讨论。图1-5显示了当设计器中的一个Button组件被选中时在Object Inspector中Events页中显示的内容。

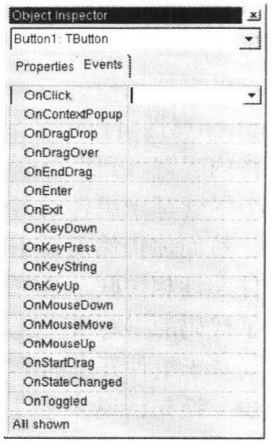


图 1-5

另一个代码生成的途径与称为“类完成”的编辑器特性相关联。简单地说，如果在类中声明了一个方法或者属性，然后按下CTRL-SHIFT-C，那么Kylix将会生成对应的实现块，这大大节省了使用手工输入的时间。类完成将在第4章进行详细讨论。

代码完成，做为Code Insight（代码洞察）的一个特性，也提供了一些生成代码的便利特性。当启用了代码完成（缺省），而且在类或者对象的限定词之后键入了点号时，在一个短的暂停之后，代码完成将显示该限定词可能的方法、属性或者域的列表。可以从这个列表中进行选择，然后按下ENTER使得Kylix插入这个方法、属性或者域的名称引用。代码完成也将在第4章进行讨论。

除了可以生成代码之外，Kylix还动态管理它所生成的代码。例如，当你创建了最初的工程之后，它生成了大量的文件，这些文件中包括单元、窗体文件以及工程源文件。这些文件将在第2章进行详细讨论。另外，工程源文件在它的uses子句中包括对生成的单元以及对应窗体的引用。当你保存单元或者改变窗体时，Kylix更新uses子句来反映你所做的任何改变。

Kylix还维护生成的事件处理程序的名称。特别是，这些事件处理程序被赋予一个很简单的名称（比如ButtonClick、SaveClick等等）。如果使用Object Inspector来改变事件处理程序的名称，那么Kylix将修改事件处理程序出现处的代码，这里所指的出现处既包括该事件处理程序的声明，又包括它的实现。

### 1.1.3 Kylix编程环境是事件驱动的

事件驱动编程意味着你的代码中的很大部分是针对某些事件的发生而执行的。典型情况下，事件是可视组件的一次用户交互，或者组件状态的一次改变。事件驱动编程是包含图形用户界面（Graphical user interface，GUI）的应用程序的主要特性。

在很多其他图形开发环境中，你必须手工处理所有事件。明确地说，必须初始化一个在应用程序的整个生命周期中运行的事件环（event loop），在事件发生时，从很多个对应该事件的方法中选择执行分支来处理该事件。

Kylix使书写与事件相对应的代码变得非常容易。明确地说，Kylix为你初始化了一个事件环，并关注对应特定事件的分支。这个分支的过程由两种机制实现。第一个（也是让创建新的组件的开发者最感兴趣的）是封装于Kylix组件内部的消息处理机制（message-handling mechanism）。很多Kylix开发者，包括那些创建新的组件的人，都不需要同消息处理机制打交道。

事件的另外一个分支机制是跟事件属性相关的。明确地说，几乎组件面板上的每个对象在选中时，都会在Component Palette的Events页中显示一个或者多个事件属性。为了执行针对特定事件的代码，开发者所需要做的只是将一个已定义好的方法指派到组件的对应事件属性上。如同在前面所描述的那样，在Kylix中完成这个过程很容易，因为它使用代码生成来创建合适的方法，并将它指派给事件属性。

事件驱动编程极大地简化了创建应用程序的任务，尤其是那些提供用户界面的应用程序。你不再需要跟踪用户的按键或者鼠标状况，Kylix的组件可以去做这些事情。你所需要做的工作就是书写一些在特定事件发生时应当执行的代码，而且Kylix会保证你的代码在合适的时候被触发。

### 1.1.4 面向对象的软件开发

在Kylix中使用的语言是Object Pascal，它是Pascal语言的面向对象版本。面向对象语言鼓励代码重用，基于组件的面向对象语言使得代码重用更加容易。

面向对象语言最重要的特点之一，就是允许你创建自己的对象，甚至可以完全使用自己创建的对象。自己创建的对象可以是基于Kylix中的已有对象，也可以是几乎完全由自己创建的（几乎完全，但不可能是完全，至少你所定义的新的对象必须继承TObject，它是Kylix对象层次中位于最高层次的对象）。

如果你不是很了解面向对象编程，没有关系，Kylix并不强迫你定义自己的对象，因为它会为你定义一些。例如，你创建的每个窗体都被定义为Kylix的一个窗体的后代。可以说面向对象语言是强有力的，也就是说如果你愿意，你就可以拥有这个能力。要想了解关于面向对象编程以及定义类的更多详细信息，则参见第15章。

### 1.1.5 理解Kylix的开放工具API

Kylix有一个值得一提的特点，那就是它的开发环境是可以配置的。例如，你可以创建自己的组件并将它们安装到Component Palette中（详细过程见第15章）。

但Kylix的特点不只限于此。它的IDE的很多特征可以被你或者第三方开发商扩展。例如，可

以书写代码并将它同你增加到Kylix的主菜单中的全新的菜单项进行关联。类似地，你可以书写自己的向导并将它安装到对象仓库（Object Repository）中。你甚至可以书写自己的键绑定（定制按键或者组合键）并将它们增加到编辑器中。所有这些之所以成为可能，是因为存在Kylix开放工具API（application programming interface，应用编程接口）。

简单说来，Kylix引入了许多类和接口，你可以实现它们，并将它们注册到Kylix中来增加自己的特性（类在第15章讨论，接口在第16章有详细描述）。当然，使用开放工具API是Kylix开发新手很少使用的，但是知道它的存在相当重要，因为你可能会需要它。在第4章有一个使用Kylix的开放工具API的例子，通过那个例子，你可以学习到如何创建自己的键绑定来将新的按键增加到编辑器中去。

### 1.1.6 快速应用程序开发

快速应用程序开发（rapid application development, RAD）是指使用开发环境的功能或者框架（framework）来快速创建应用程序。Kylix是一个RAD环境，而且它是Linux下第一个RAD环境。

当然，RAD是一个相对的概念。RAD并不代表任何应用程序都能够被快速地创建，而是说RAD环境提供的工具能够使你比在没有这些工具的情况下可以更快地创建应用程序。例如，使用Kylix能够在几分钟内创建一个小的、具有某些功能的应用程序（如果有合适的组件，一个小的应用程序完全可以在几分钟内完成）。但是这必须是一个相当小的应用程序。相比之下，同样复杂度的应用程序在其他开发环境下可能需要一两天（至少几个小时）的时间来完成。

但是期望任何应用程序可以在几分钟内完成是不现实的。一个复杂的Kylix应用程序可能需要几个月，甚至几年的时间来开发。但总地说来，Kylix提供的功能和框架使你能够以快于其他可选方法的速度来开发这些应用程序。

## 1.2 Delphi开发者眼中的Kylix

Kylix经常会被称为是Linux下的Delphi，这是相当正确的描述。Kylix和Delphi有着相同的代码基础，它们都给开发者提供一个熟悉的创建应用程序的环境，无论在Windows下还是在Linux下。

实际上，Kylix和Delphi的相似之处主要有两方面。首先，用来创建应用程序的IDE是类似的。大多数Delphi开发者在最初看到Kylix时，会感到相当亲切。如果你对Delphi 5的IDE很熟悉，观察图1-6所显示的Kylix IDE，你会发觉：惟一使你感觉不是在Delphi下的证据是Linux的图形桌面（本例中是GNOME，运行在GNOME桌面下的Kylix几乎与Delphi 5的IDE一样）。

另一个相似之处在于创建应用程序时使用的组件库和运行时库（runtime library）。例如，Kylix和Delphi 6（最新版本的Delphi）的运行时库几乎是一致的。换句话说，使用Kylix书写的使用了运行时库的程序和符号（常量、变量等等）的Object Pascal大多可以被Kylix和Delphi编译通过。当使用Kylix编译时，编译结果可以在Linux下执行，当使用Delphi编译时，编译结果可以在MS Windows下执行。

组件库也存在这种兼容性，但是有一定的限制。Delphi 6支持两个组件库：可视组件库（visual component library，VCL）以及跨平台组件库（component library cross-platform，CLX，发音为“clicks”）。VCL严重依赖于Windows API以及Windows消息队列。因此，Kylix不支持VCL应用程序。