

<<Qt中的C++技术>>

图书基本信息

书名：<<Qt中的C++技术>>

13位ISBN编号：9787121171598

10位ISBN编号：7121171597

出版时间：2012-7

出版时间：电子工业出版社

作者：张波

页数：285

字数：444000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<Qt中的C++技术>>

内容概要

《Qt中的C++技术》剖析了开源开发框架Qt中的C++技术，给读者提供一个优秀的案例，以学习C++语言以及面向对象设计技术。该书讨论了以下内容：类模板特化技术；分析比较了C++标准库、Qt对字符串、数据输入/输出的处理思路；隐式共享与d-pointer技术；函子及其在QTL (Qt Template Library) 中的应用，QTL是如何使用模板特化技术优化QList性能的；如何在C++程序中嵌入汇编代码，实现一个原子操作，以很小的开销实现线程间通信；信号与槽机制；Graphics/View框架等。

软件学院或者计算机学院的学生，可将《Qt中的C++技术》作为课程“ C++程序设计 ” 或者 “ 面向对象软件设计 ” 的参考书；上述课程的教师，可将《Qt中的C++技术》的内容融入他们的主讲或者试验环节，作为相关实训课程的教材；软件行业的开发者，可将《Qt中的C++技术》作为深入学习C++设计与编程技术的案例教材。

<<Qt中的C++技术>>

书籍目录

第1章 案例的选择与评估

- 1.1 案例的初步选择
- 1.2 案例的定量评估
- 1.3 其他案例
- 1.4 基本约定
- 1.5 关于类图的约定

第2章 qt概述

- 2.1 qt版权
- 2.2 qt库的编译
- 2.3 开发环境的设置
- 2.4 主控台的输入与输出
- 2.5 qt风格的编程规范
- 2.6 与qt及c++相关的文献资源

第3章 类模板特化

- 3.1 类模板特化
- 3.2 traits技术
- 3.3 类型分类 (type classification) 技术
- 3.4 降低代码膨胀

第4章 标准库及qt对字符串的处理

- 4.1 字符及其编码
- 4.2 标准库的类模板basic_string
- 4.3 qt的类qstring

第5章 国际化与区域文化

- 5.1 区域文化
- 5.2 facet
- 5.3 类locale的实现
- 5.4 类模板facet的实现
- 5.5 派生新的facet类

第6章 c++的iostream

- 6.1 c语言的scanf/printf函数组
- 6.2 iostream的总体结构
- 6.3 字符特征的描述
- 6.4 模板特化后的总体结构
- 6.5 文件流
- 6.6 字符串流
- 6.7 流缓冲区
- 6.8 二进制文件的处理
- 6.9 用户自定义类型的输入和输出

第7章 qt的流

- 7.1 文件系统及底层文件操作
- 7.2 类qtextstream
- 7.3 类qdatastream
- 7.4 类qlocale
- 7.5 iostream和qt流类的比较

第8章 隐式共享与d-pointer技术

<<Qt中的C++技术>>

- 8.1 隐式共享
- 8.2 d-pointer在隐式共享中的应用
- 8.3 二进制代码兼容
- 8.4 d-pointer模式的实现
- 8.5 qobject中的d-pointer
- 第9章 qt容器与迭代器
- 9.1 qtl概述
- 9.2 qtl容器和qdatastream的无缝连接
- 9.3 类型分类技术在qlist中的应用
- 9.4 函子的应用——相关词词典
- 第10章 多线程与可重入
- 10.1 创建一个线程
- 10.2 线程间同步
- 10.3 线程安全与可重入
- 10.4 多线程环境下的singleton模式
- 第11章 信号与槽 (signals and slots)
- 11.1 对象树 (qobject tree)
- 11.2 信号与槽机制
- 11.3 信号与槽的应用例子
- 第12章 graphics/view框架
- 12.1 qt图形系统介绍
- 12.2 graphics/view框架
- 12.3 例子——相撞的老鼠
- 第13章 model/view框架
- 13.1 model/view框架总体架构
- 13.2 模型 (models)
- 13.3 视图 (views)
- 13.4 选择操作
- 13.5 委托 (delegates)
- 13.6 代理模型 (proxy models)
- 13.7 便利视图类
- 第14章 qt中的命令模式
- 14.1 qt的undo framework
- 14.2 使用undo framework的一个例子
- 第15章 qt中的抽象工厂模式
- 15.1 抽象工厂模式简介
- 15.2 qtextcodec及其子类的定义
- 15.3 界面风格
- 第16章 qt中的观察者模式
- 16.1 事件处理机制
- 16.2 事件滤波器
- 16.3 一个简单的例子——图像浏览器
- 16.4 一个有趣的例子——鼠标手势
- 第17章 qt的元对象系统
- 17.1 c++ rtti (run-time type information)
- 17.2 qt的元对象系统
- 第18章 智能指针

<<Qt中的C++技术>>

18.1 qpointer

18.2 qshareddatapointer

参考文献

索引

代码目录

第1章 案例的选择与评估

第2章 qt概述

代码段2 1, 使用qt进行主控台输入与输出, 取自z:\examples\qt_console\main.cpp

第3章 类模板特化

代码段3 1, 类模板stack, 引自z:\examples\template_specialization

代码段3 2, 完全特化的类模板stack, 引自examples\template_specialization

代码段3 3, 部分特化的类模板stack, 引自examples\template_specialization

代码段3

4, 使用traits技术封装float及double类型的特征, 取自z:\examples\float_traits\main.cpp

代码段3 5, 类型分类技术, 取自z:\examples\type_classification\main.cpp

代码段3 6, 应用类模板特化降低代码膨胀, 摘自examples\reduce_code_bloat

代码段3 7, 应用类模板特化降低代码膨胀, 摘自z:\examples\reduce_code_bloat

第4章 标准库及qt对字符串的处理

代码段4 1, 类模板basic_string的构造函数

代码段4

2, 类模板basic_string常用构造函数的使用, 取自z:\examples\basic_string_demo\main.cpp

代码段4 3, basic_string对象和字符串的比较

代码段4 4, 字符串的不同存放方式, 摘自z:\examples\qstring_demo\main.cpp

代码段4 5, qstring的字符编码转换功能, 摘自z:\examples\qstring_merit\main.cpp

第5章 国际化与区域文化

代码段5

1, 类模板time_get的使用方法, 取自z:\examples\locale_time_get\main.cpp

代码段5 2, 类模板time_put的成员函数put的用法, 取自z:\examples\locale_time_put\main.cpp

代码段5 3, 类模板codecvt的成员函数in的功能, 取自z:\examples\locale_codecvt\main.cpp

代码段5 4, 类locale以及facet的实现框架, 取自vs 2010安装目录crt\src\locale

代码段5 5, 类locale::id的作用

代码段5 6, 创建新的facet子类, 取自z:\examples\locale_unit\main.cpp

第6章 c++的iostream

代码段6

1, 应用类模板char_traits实现大小写不敏感的字符串类ci_string, 取自z:\examples\ci_string\ci_string\main.cpp

代码段6 2, 对文件流进行读取以及写入操作, 取自z:\examples\fstream_demo\main.cpp

代码段6 3, 具有多种格式设置的流, 取自z:\examples\share_streambuf

代码段6 4, 流缓冲区的复制, z:\examples\copy_streambuf\main.cpp

代码段6 5, 二进制文件的读取, 摘自z:\examples\process_binary\main.cpp

代码段6 6, 直接操作二进制文件对应的流缓冲区, 摘自z:\examples\process_binary\main.cpp

代码段6 7, 用户自定义类型的输入和输出, 摘自z:\examples\overloaded_io\main.cpp

<<Qt中的C++技术>>

第7章 qt的流

代码段7

1, 使用qfileinfo以及qdir获取各驱动器下的子目录名, 取自z:\examples\qdir_demo\main.cpp

代码段7 2, 使用qfile操作一个文件, 取自z:\examples\qfile_read\main.cpp

代码段7 3, 使用qtextcodec转换编码方案, 摘自z:\examples\

qtextstream_demo\main.cpp

代码段7 4, 使用qdatastream输出二进制数据, 摘自z:\examples\

qdatastream_usage\main.cpp

代码段7 5, 使用qdatastream读取二进制数据, 摘自z:\examples\

qdatastream_usage\main.cpp

代码段7

6, 重载运算符以使qdatastream支持新的数据类型, 取自z:\examples\qdatastream_demo\main.cpp

代码段7 7, qlocale的使用, 摘自z:\examples\qlocale_set\main.cpp

第8章 隐式共享与d-pointer技术

代码段8 1, 采用隐式共享技术的qstring::tocasefolded(), 取自src\

corelib\tools\qstring.cpp

代码段8 2, qstring的复制构造函数, 摘自src\corelib\tools\qstring.h

代码段8 3, 类matrix的传统定义方式, 摘自z:\examples\d_pointer\matrix\main.cpp

代码段8

4, 应用d-pointer模式的类matrix, 摘自z:\examples\d_pointer\matrix_with_d_pointer\main.cpp

代码段8 5, qt 4.5版本中类qlocale的定义, 摘自s:\corelib\tools\qlocale.h

代码段8 6, qt 4.5版本中类qlocaleprivate的定义, 摘自

s:\corelib\tools\qlocale_p.h

代码段8 7, 修改成员函数toString()加入qt库的版本信息, 摘自

s:\corelib\tools\qlocale.cpp

代码段8 8, 使用类qlocale的qt应用程序, 摘自z:\examples\d_

pointer\test_qlocale\main.cpp

代码段8 9, 实现d-pointer模式的一个例子

代码段8 10, qt中与d-pointer模式相关的宏, 摘自src\corelib\global\qglobal.h

代码段8 11, 宏q_declare_private展开后的结果

代码段8 12, qobject及qobjectdata的定义, 摘自s:\corelib\kernel\qobject.h

代码段8 13, 类qobject及qobjectprivate对d-pointer的使用

代码段8 14, QWidget继承了qobject的d-pointer模式

第9章 qt容器与迭代器

代码段9 1, stl风格以及java风格的迭代器, 摘自z:\examples\

qlist_change_value\main.cpp

代码段9 2, foreach的使用格式, 取自z:\examples\foreach_demo\main.cpp

代码段9 3, 类模板qlless, 摘自src\corelib\tools\qalgorithms.h

代码段9 4, 使用qdatastream保存/读取qmap对象, 摘自

z:\examples\english_pron\main.cpp

代码段9

5, qlistdata的成员函数remove(), 摘自src\corelib\tools\qlistdata.cpp

代码段9 6, qtypeinfo的定义, 摘自src\corelib\global\qglobal.h

代码段9 7, qlist的数据结构, 摘自src\corelib\tools\qlist.h

代码段9 8, 向qlist中添加元素, 摘自src\corelib\tools\qlist.h

代码段9 9, 使用qsort对容器排序, 摘自z:\examples\qtl_related_words\main.cpp

代码段9

<<Qt中的C++技术>>

10, 函子indirectcompare, 摘自z:\examples\qtl_thesaurus\main.cpp

第10章 多线程与可重入

代码段10 1, 在qt中创建多线程, 摘自z:\examples\simple_thread\main.cpp

代码段10 2, 互斥体qmutex的使用, 取自z:\examples\qmutex\main.cpp

代码段10 3, 使用互斥体的一个简单方法

代码段10 4, 用信号量来管理循环缓冲区, 摘自z:\examples\qsemaphore\main.cpp

代码段10 5, 使用条件量管理循环缓冲区, 摘自z:\examples\qwaitcondition\main.cpp

代码段10 6, 以传统方式实现singleton模式

代码段10 7, 在堆中创建全局对象

代码段10 8, 简化的成员函数instance()

代码段10 9, 通过静态局部对象来定义singleton对象

代码段10 10, 静态局部对象的初始化

代码段10 11, c++的判断/赋值操作无法锁定共享资源

代码段10 12, qbasicatomicpointer

的定义, 摘自s:\corelib\thread\qbasicatomic.h

代码段10 13, qbasicatomicpointer

在windows/intel平台上的实现, 摘自s:\corelib\arch\qatomic_windows.h

代码段10 14, 类模板qglobalstatic的定义, 摘自s:\corelib\global\qglobal.h

代码段10

15, 类模板qglobalstaticdeleter的定义, 摘自s:\corelib\global\qglobal.h

代码段10 16, 宏q_global_static的定义, 摘自s:\corelib\global\qglobal.h

代码段10 17, 单线程环境下宏q_global_static的定义, 取自

s:\corelib\global\qglobal.h

代码段10 18, 直接返回一个指向singleton对象的指针

代码段10

19, 使用一个已经析构的singleton对象, 摘自z:\examples\use_destructed_singleton\main.cpp

第11章 信号与槽 (signals and slots)

代码段11 1, QObject对象的定义顺序, 摘自z:\examples\

QObject_destruction_order\main.cpp

代码段11 2, 信号与槽的定义, 摘自z:\examples\signals_slots_

demo\signals_slots_declare.h

代码段11 3, 信号与槽的绑定, 摘自z:\examples\signals_slots_demo\main.cpp

代码段11 4, 类finddialog的定义, 摘自

z:\examples\find_dialog\find_dialog.h

代码段11 5, 类finddialog的构造函数, 取自

z:\examples\find_dialog\find_dialog.cpp

代码段11 6, 类finddialog的构造函数 (续), 摘自

z:\examples\find_dialog\find_dialog.cpp

代码段11 7, 类finddialog的槽函数及析构函数, 摘自

z:\examples\find_dialog\find_dialog.cpp

第12章 graphics/view框架

代码段12 1, 类mouse的定义, 取自z:\examples\collidingmice\mouse.h

代码段12 2, 类mouse的部分成员函数, 取自z:\examples\collidingmice\mouse.cpp

代码段12 3, 类mouse的成员函数advance(), 取自

z:\examples\collidingmice\mouse.cpp

代码段12 4, 项目collidingmice的主函数, 取自

z:\examples\collidingmice\main.cpp

<<Qt中的C++技术>>

第13章 model/view框架

代码段13

1, 类treemodel的声明, 取自z:\examples\mvc\binary_tree\treemodel.h

代码段13

2, 类treemodel的实现, 取自z:\examples\mvc\binary_tree\treemodel.cpp

代码段13 3, 类treemodel的实现(续), 取自

z:\examples\mvc\binary_tree\treemodel.cpp

代码段13 4, 满二叉树例子的主函数, 取自z:\examples\mvc\binary_tree\main.cpp

代码段13 5, 能够处理更多角色的模型类, 取自

z:\examples\mvc\binary_tree_more_role\treemodel.cpp

代码段13 6, 显示自身发生变化的数据项, 取自

z:\examples\mvc\binary_tree_changing_data\treemodel.cpp

代码段13 7, 更改数据集的标头, 取自z:\examples\mvc\
binary_tree_header\treemodel.cpp

代码段13 8, 编辑满二叉树的叶节点, 取自z:\examples\mvc\binary_
tree_editable\treemodel.cpp

代码段13

9, 重载qabstractlistmodel的虚函数以显示、编辑一个列表, 取

自z:\examples\mvc\qabstractlistmodel_demo\listmodel.cpp

代码段13

10, 使用qstandarditemmodel处理列表, 取自z:\examples\mvc\qstandarditemmodel_demo\main.cpp

代码段13

11, 使用qstandarditemmodel处理表格, 取自z:\examples\mvc\qstandarditemmodel_demo\main.cpp

代码段13

12, 使用qstandarditemmodel处理树, 取自z:\examples\mvc\qstandarditemmodel_demo\main.cpp

代码段13

13, 类qstringlistmodel的使用, 取自z:\examples\mvc\qstringlistmodel_demo\main.cpp

代码段13

14, 便利模型类qfilesystemmodel的用法, 取自z:\examples\mvc\file_system\main.cpp

代码段13

15, 用qcolumnview对象显示本地文件系统, 取自z:\examples\mvc\qcolumnview_demo\main.cpp

代码段13

16, 类mainwindow的声明, 取自z:\examples\mvc\selection_monitoring\mainwindow.h

代码段13

17, 类mainwindow实现, 取自z:\examples\mvc\selection_monitoring\mainwindow.cpp

代码段13

18, 同步两个视图对象中的选择信息, 取自z:\examples\mvc\sync_selection\main.cpp。

代码段13 19, 例子spinbox的主函数, 取自z:\examples\mvc\
spinboxdelegate\main.cpp

代码段13 20, 类spinboxdelegate的实现, 取自z:\examples\mvc\
spinboxdelegate\delegate.cpp

代码段13 21, 代理模型索引的创建, 取自z:\examples\mvc\revertproxymodel\revertproxymodel.cpp

代码段13 22, 代理模型revertproxymodel的其他2个接口函数, 取

代码段13

自z:\examples\mvc\revertproxymodel\revertproxymodel.h

代码段13

代码段13 22, 代理模型revertproxymodel的其他2个接口函数, 取
自z:\examples\mvc\revertproxymodel\revertproxymodel.h

<<Qt中的C++技术>>

代码段13

23, 代理模型revertproxymodel的接口函数parent(), 取自z:\examples\mvc\revertproxymodel\revertproxymodel.cpp
代码段13 24, 接口函数data()的实现, 取自q:\src\gui\itemviews\qabstractproxymodel.cpp

代码段13

25, 创建源模型, 取自z:\examples\mvc\basicsortfiltermodel\main.cpp

代码段13 26, 类window的构造函数, 取自z:\examples\mvc\basicsortfiltermodel>window.cpp

代码段13 27, 令代理模型指向源模型, 取自z:\examples\mvc\basicsortfiltermodel>window.cpp

代码段13

28, 代理模型对源模型数据项的过滤、排序, 取自z:\examples\mvc\basicsortfiltermodel>window.cpp

代码段13

29, 向qmainwindow中添加数据项, 取自z:\examples\mvc\mainwindow_demo\main.cpp

代码段13 30, 新闻的表示, z:\examples\mvc\item_roles\newsdialog.cpp

代码段13

31, 设置mainwindowitem所表示数据项中的数据子项, 取自z:\examples\mvc\item_roles\newsdialog.cpp

代码段13 32, gdp数据的表示, 取自z:\examples\mvc\

mainwindow_demo\main.cpp

代码段13 33, 类mainwindow的使用, 取自z:\examples\mvc\

mainwindow_demo\main.cpp

代码段13 34, 书籍目录的表示, 取自z:\examples\mvc\

mainwindow_demo\main.cpp

代码段13

35, 构建mainwindow中的树状模型, 取自z:\examples\mvc/mainwindow_demo\main.cpp

第14章 qt中的命令模式

代码段14 1, qundocommand的部分定义

代码段14 2, 类qundocommand部分成员函数的实现, 摘自s:\gui\util\qundostack.cpp

代码段14 3, qundostack基本功能部分的定义, 取自s:\gui\util\qundostack.h

代码段14 4, qundostackprivate的部分定义, 取自s:\gui\util\qundostack_p.h

代码段14 5, 类movecommand的定义, 取自

z:\examples\undoframework\commands.h

代码段14 6, 类movecommand的实现, 取自

z:\examples\undoframework\commands.cpp

第15章 qt中的抽象工厂模式

代码段15 1, 依据全局变量创建不同风格的界面元素

代码段15 2, 抽象工厂模式的使用

代码段15

3, latin1到unicode的转换, 取自z:\examples\factory_pattern\main.cpp

代码段15 4, 类qtextcodec的部分定义, 取自s:\corelib\codecs\qtextcodec.h

代码段15 5, qtextcodec部分成员函数的实现, 取自

s:\corelib\codecs\qtextcodec.cpp

代码段15 6, 类qlatin1codec的定义与实现

代码段15

7, 类widgetgallery的成员函数changestyle(), 取自z:\examples\styles\widgetgallery.cpp

<<Qt中的C++技术>>

代码段15 8, 圆角矩形绘制路径的绘制, 取自z:\examples\
styles\norwegianwoodstyle.cpp

代码段15 9, 基本元素的绘制, 取自z:\examples\styles\norwegianwoodstyle.cpp

代码段15 10, 基本元素的绘制(续), 取自z:\examples\styles\
norwegianwoodstyle.cpp

代码段15

11, norwegianwoodstyle的成员函数drawcontrol, 取自z:\examples\styles\norwegianwoodstyle.cpp

代码段15 12, 更改控件属性的成员函数polish, 取自z:\examples\
styles\norwegianwoodstyle.cpp

代码段15

13, 更改控件尺寸的成员函数pixelmetric, 取自z:\examples\styles\norwegianwoodstyle.cpp

代码段15 14, 设置与风格相关的一些属性, 取自z:\examples\
styles\norwegianwoodstyle.cpp

代码段15

15, 更改应用程序调色板的成员函数polish, 取自z:\examples\styles\norwegianwoodstyle.cpp

代码段15

16, 设置填充图像的私有成员函数settexture, 取自z:\examples\styles\norwegianwoodstyle.cpp

第16章 qt中的观察者模式

代码段16 1, 设置qscrollarea为另一控件的观察者, 取自

q:\src\gui\widgets\qscrollarea.cpp

代码段16 2, 将一个qscrollarea对象设置为一个qlabel对象的观察者, 取自

z:\examples\imageviewer\imageviewer.cpp

代码段16 3, 类imageviewer中更改图像显示比例的函数, 取自

z:\examples\imageviewer\imageviewer.cpp

代码段16 4, 鼠标手势及其回调函数的定义, 取自

z:\examples\mouse_gesture\mousegesturerecognizer.h

代码段16 5, 类mousegesturerecognizer的定义, 取自

z:\examples\mouse_gesture\mousegesturerecognizer.h

代码段16 6, 对鼠标轨迹进行识别的核心算法, 取自

z:\examples\mouse_gesture\mousegesturerecognizer.cpp

代码段16 7, 类mousegesture的定义, 取自z:\examples\mouse_

gesture\mousegesture.h

代码段16 8, 类gesturecallbacktosignal 的定义, 取自

z:\examples\mouse_gesture\mousegesturefilter.cpp

代码段16 9, 类mousegesturefilter的定义

代码段16

10, 鼠标手势的添加与删除, 取自z:\examples\mouse_gesture\mousegesturefilter.cpp

代码段16

11, 事件滤波器及相关函数, 取自z:\examples\mouse_gesture\mousegesturefilter.cpp

代码段16 12, 类mainwindow的定义, 取自z:\examples\mouse_\
gesture/mainwindow.h

代码段16 13, 鼠标手势例子的主函数, 取自z:\examples\mouse_gesture\main.cpp

第17章 qt的元对象系统

代码段17 1, 类typeid_info的声明

代码段17

2, typeid的操作数可以为基本类型、非多态类及多态类, 取自z:\examples\typeid\main.cpp

代码段17

<<Qt中的C++技术>>

3, 判断一个qobject派生类的对象是否“具有”某个类型, 取自z:\examples\qmetaobject_demo\main.cpp
代码段17

4, 获取qobject派生类对象的类型信息, 取自z:\examples\qmetaobject_demo1\main.cpp

代码段175, 获取qobject派生类对象的数据, 取自z:\examples\introspect_qobject\main.cpp

代码段176, qvariant的使用, 取自z:\examples\qvariant\main.cpp

代码段177, qvariant支持二进制输入/输出, 摘自z:\examples\qvariant\main.cpp

第18章 智能指针

代码段181, qpointer的功能, 取自z:\examples\qpointer_demo\main.cpp

代码段182, 对一个对象施加delete运算符, 取自z:\examples\delete_object\main.cpp

代码段183, 具有淡入显示效果的类faderwidget

代码段184, 使用qpointer来判断一个qfaderwidget控件是否存在

代码段185, qwidget对qpointer的使用, 取自s:\gui\kernel\qwidget.cpp

代码段18

6, 遍历qmultihash中具有相同关键字的元素, 取自z:\examples\qmultihash_demo\main.cpp

代码段187, 使用信号量qreadwritelock锁定某个资源以进行写入操作

代码段188, 使用信号量qwritelock锁定某个资源以进行写入操作

代码段189, 与qpointer相关的类型与函数, 取自s:\corelib\kernel\qobject.cpp

代码段1810, qpointer的定义, 摘自s:\corelib\kernel\qpointer.h

代码段1811, qmetaobject中的相关代码, 取自s:\corelib\kernel\qobject.cpp

代码段1812, qobject析构函数中与qpointer相关的代码, 摘自s:\corelib\kernel\qobject.cpp

代码段1813, 使用类模板qshareddatapointer

实现隐式共享, 取自z:\examples\qshareddatapointer_demo\main.cpp

代码段1814, 关于常量型成员函数的约定, 取自z:\examples\select_const\main.cpp

代码段1815, 类qshareddata

的定义, 取自q:\src\corelib\tools\qshareddata.h

代码段1816, 类模板qshareddatapointer的定义(待续), 取自q:\src\corelib\tools\qshareddata.h

代码段1817, 类模板qshareddatapointer的定义(待续), 取自q:\src\corelib\tools\qshareddata.h

代码段1818, 类模板qshareddatapointer的定义(续), 取自q:\src\corelib\tools\qshareddata.h

参考文献

索引

<<Qt中的C++技术>>

章节摘录

Qt概述 Qt (发单词“Cute”的音) 是一个跨平台的C++开发框架, 它包含一个功能丰富的C++类库以及一套简便易用的集成开发工具。

Qt所支持的平台不但包括Linux, Windows以及MacOSX等主流桌面操作系统, 还包括诸如Symbian, Maemo以及MeeGo这样的嵌入式操作系。

使用Qt编写的C++程序具有良好的跨平台特性, 程序员几乎无须更改源代码, 所编写的应用程序即可运行在各种操作系统中, 这能大幅度缩短开发周期、降低开发成本。

Qt的C++类库是完全面向对象的, 经过精心的设计, 该类库不但功能强大, 而且方便易用。

这些优点使得Qt被Adobe, Boeing, Google, IBM, Motorola, NASA, Skype等大型机构以及众多的中小公司采用。

1.Qt的历史 回顾Qt二十余年的发展历史, 我们可以学习是哪些因素促成了Qt的成功。

Qt的创始人是HaavardNord和EirikChambe-Eng, 二人后来分别成为Trolltech公司的首席执行官和总裁。

1988年, 受一个瑞典公司的委托Haavard开始开发一个C++图形库。

1990年夏季, 二人共同开发一个处理超声波图像的数据库系统时, 需要一个能够运行在UNIX, Macintosh以及Windows上的跨平台C++图形库。

一天, 两人在公园长椅上享受阳光浴时, Haavard说: “我们需要一个面向对象的图形显示系统”, 之后的讨论促成了Qt的诞生。

这是Qt成功的首要因素: 源于实际需求。

1991年-1993年, 二人设计并实现了Qt库的图形核心库, 一组控件以及“信号与槽”机制。

1994年, 二人创建了后来的Trolltech公司, 并与1995年5月公开发布了Qt0.90版。

自发布之日起, Qt就提供了商业授权和开源软件授权两种方式。

发布之后的10个月中, 没有任何人购买Qt的商业授权。

直到1996年3月, 欧洲航天局终于购买了Qt的10份商业授权, Qt才得以逐步壮大。

从这一阶段的历史, 我们可以看出Qt成功的另外两个因素: 开发团队精良的技术(比如提出并实现了“信号与槽”机制); 欧洲人对知识产权的尊重(Qt创始人能够放心地发布Qt的源代码, 而欧洲航天局在能够看到Qt所有源代码的条件下仍然购买Qt的商业授权)。

1997年, KDE项目的组织者MatthiasEttrich决定使用Qt构建KDE, 使Qt实际成为Linux上开发C++图形程序的标准库。

2001年, Qt3.0发布, 它的源代码已经超过50万行。

2005年, Qt4.0发布, 包含500多个类, 9000个函数。

2008年, Nokia收购了Trolltech公司, 将Qt作为该公司移动设备的主要开发平台。

.....

<<Qt中的C++技术>>

编辑推荐

《Qt中的C++技术》共18章。

第1章讲述为什么会从众多的开源C++项目中选择Qt。

读者可以借鉴其中的方法选择其他C++案例，或者在学习其他编程语言时，使用其中的方法选择对应的案例。

而且，读者还可以使用其中的工具CppDepend剖析其他软件的结构与质量。

这一章还介绍了本书对术语、UML类图方面的约定。

在阅读后续章节前，读者应该首先阅读这一章。

本书不但剖析Qt的源代码，有的章节还涉及修改Qt的源代码，此时需要重新编译整个Qt库。

第2章简要介绍Qt，并讲述如何在Visual Studio 2010开发环境下安装、编译Qt库。

Qt库多处用到了类模板特化技术。

考虑到一般的C++教科书不会详细讲解这个话题，故第3章阐述该技术的概念和基本应用，第6章及第9章用到了该技术。

<<Qt中的C++技术>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>