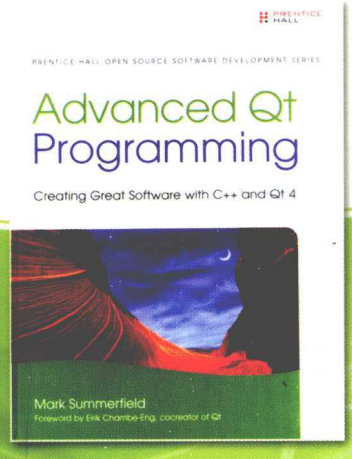
PEARSON

Qt 高级编程

Advanced Qt Programming
Creating Great Software with C++ and Qt 4



[英] Mark Summerfield 著

白建平 王军锋 闫锋欣 白 净 译

高 波 审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Qt 高级编程

Advanced Qt Programming

Creating Great Software with C++ and Qt 4

[英] Mark Summerfield 著

白建平 王军锋 闫锋欣 白 净 译

吴 迪 戚 彬 高 波 审校

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

译 者 序

Qt 是跨平台的应用程序和用户接口(UI)开发框架,由集成开发工具、跨平台类库和集成开发环境(IDE)组成,可轻松实现应用程序的"一次编写,随处编译"。目前,Qt 主要由诺基亚的 Qt Development Frameworks(Qt 开发框架组)负责开发和维护,用户涵盖全球 60 多个国家的 4400 多家厂商,如 Google、Adobe、IBM、华硕、CNTV、中国移动等,基于 Qt 的知名应用程序有 KDE、KOffice、Skype、Google Earth 等。

自 2008 年 6 月诺基亚购并奇趣科技后,Qt 在嵌入式移动平台上的发展大大提速。Qt 4.6 增加了 Symbian、Maemo 平台的支持,旋即发布的 QtMobility 开发包可提供各移动支持平台下的联系人、导航、网络连接等 API;Qt 4.7 引入了快速开发脚本语言 QML,为普通开发人员快速开发 Qt 应用程序提供了可能。而今集成了 Qt Creator 开发环境的 Nokia Qt SDK,加强了基于 Qt 开发 Symbian、Maemo/MeeGo/WinCE 应用程序的易用性,进一步为非专业人士投身移动平台应用程序开发降低了门槛。

在翻译本书的过程中,深感国内 Qt 技术力量的薄弱。在互联网如此发达的今天,除 Qt 中文论坛(www. qtcn. org)、CuteQt 博客(www. cuteqt. com)和 CSDN 的 Qt 技术社区(qt. csdn. net)等少数网站外,大多处于停滞状态。Qt 中文论坛建立于 2005 年,面向广大初、中级 Qt 开发人员,是目前最为活跃的 Qt 综合技术中文讨论区;CuteQt 博客紧跟 Qt 的前沿技术领域和最新的 Qt 开发平台,由许多一线的 Qt 资深工程师负责维护和运行;CSDN 的 Qt 技术社区提供了许多权威资料。正是这些技术社区在不遗余力地积极推动着 Qt 技术在国内的发展。

由本书译者参与的《C++ GUI Qt 4 编程》(第二版)是第一本对 Qt 4 技术进行全面、系统介绍的中文权威译著。Jasmin Blanchette 和 Mark Summerfield 是该书的作者,也是 Qt 在线帮助文档的创建者和 Qt 开发人员。在书中,他们用许多示例程序和技术案例全面地介绍了 Qt 框架,使该书成为官方推荐的一本学习书籍,并作为诺基亚员工的 Qt 入门培训教材。正是如此,该书中文版自出版以来的两年内,已累计重印 6 次。

本书是《C++ GUI Qt 4 编程》(第二版)出版两年后 Mark Summerfield 的又一本学习 Qt 的里程碑级图书。在这本书中,不仅涵盖了《C++ GUI Qt 4 编程》(第二版)中部分过于高深和未能包含其中的内容,还有许多针对 Qt 技术底层细节的探讨,大多还没有在任何书籍中涉及过。因此,本书是近两年和未来一段时间内 Qt 高级技术的概括和预览,可帮助 Qt 编程人员切实提高他们使用 Qt 成就事业的能力。

在本书的翻译和审校过程中,我们坚持使用了"两译三审"的严苛做法,力求保证译稿质量,减少误译和纰漏。对于书中涉及的 Qt 和计算机技术词汇用语,则尽量与《C++ GUI Qt 4 编程》(第二版)一书的译法保持一致,避免读者产生困惑。同时,对于某些技术细节还向不少 Qt 一线开发工程师和原书作者 Mark Summerfield 做了求证。同时,结合英文原书的勘误信息,译者已将本书的相关代码更新至 2011 年 3 月。

本书的翻译和审校工作具体分工是:Qt 中文论坛管理员白建平(XChinux)负责本书的第 3 章、第 4 章、第 5 章和第 6 章翻译工作;西南科技大学的王军锋负责第 1 章、第 2 章、第 9 章和第 10 章的翻译;CuteQt 博客管理员 Shiroki 负责第 11 章和第 12 章的翻译;西北农林科技大学的闫锋欣负

责第7章、第8章、第13章以及书中剩余的前言、简介、精选参考书目和结束语等部分的翻译和质量控制工作。解放军装甲兵工程学院的吴迪(wd007)、高波和山东理工大学的戚彬负责了全书的终审和统稿。此外,我们还邀请了西安欧亚学院的周莉娜、赵延兵和韩二伟作为本书的外部审稿人。参与本书文字校对工作的还有朱加平、齐亮、王宁、赵拓和范文等人。

感谢电子工业出版社的编辑。他们对于计算机当前技术趋势的把握和战略眼光,为Qt系列图书的选题引进提供了许多方便,为译者提供了充足的工作时间,使译稿质量得到了最大限度的保证。正是他们对于图书选题的理解,才能为大家的Qt学习之路提供如此多的选择。

书中所用到的示例程序的源代码可从原书站点www.qtrac.eu/(英文)下载,也可直接从www.qtcn.org/advqt(中文)下载。

由于本版书中概念和术语数目繁多,并且许多概念和术语目前尚无公认的中文译法,加之译者水平所限,时间仓促,译稿中难免存在曲解或误解作者原意的地方,恳请读者谅解。读者也可以登录www.qtcn.org/advqt参与讨论,我们也会在此及时更新本书的勘误信息。

译　者

2010年10月

序　言

回想起 1991 年，我与 Haavard Nord 一起坐在挪威特隆赫姆公园中的长椅上。在我和 Haavard Nord 完成非服兵役服务的那段时间内，我俩一起为当地的一家医院开发一个超声波图像的存储和分析软件。这家医院使用了各种类型的计算机，因此，医院希望这个软件系统可以在 UNIX、Mac 和 Windows 平台上工作。这是一个巨大的挑战，我们调查市场，希望可以找到一些可以使用的类库。但对于找到的那些类库的质量，令我们担心不已。就在公园的那个长椅上，我们决定迎接这个挑战，提出我们自己的解决方案。

那个时候，我们都年轻、充满斗志且有些天真。讨厌浪费时间查找如何使用那些非直觉型的工具和库，我们决定改进这些工具和库。希望能稍微改善一下世界曾有的软件开发工作。我们的目标就是要让软件开发人员的生活变得更轻松些。众所周知，要实现这一目标，就要专注于软件开发中有趣的那一面，要富有创造性、编写出良好的代码。这样，我们就创造了第一个简单的 Qt 版本，并在数年后组建了 Trolltech（奇趣）公司。

至少，我们完成了一部分既定目标。自从 1995 年第一次发行 Qt 以来，它已取得了巨大的成功。

2008 年，Nokia 收购了 Trolltech，2009 年 4 月，对我而言，是时候要继续前进了。在公司里度过了 15 年又 27 天后，我不再是其中的一员了。

产品在优秀的人手中，团队的激情和繁重的工作一如既往。在 Nokia 的奇趣科技（Trolltech）依旧在努力确保 Qt 成为一流的、众望所归的框架。Lars Knoll（kHTML、鼎鼎大名的 WebKit 的鼻祖）今天正率领着近 150 名 Qt 专业工程师。作为授权协议的选择，Nokia 还新增了 LGPL 授权，这使 Qt 可被更多的开发人员所使用。

2009 年秋，我以荣誉嘉宾的身份受 Nokia 之邀参加了德国慕尼黑的 Qt 开发人员日（Qt Developer Days）活动。这个用户参与的会议（也在美国举办）是 Qt 爱好者的盛会，其规模正在逐年扩大。能够倾听来自全欧洲 Qt 用户间的探讨声，是一种很棒的感觉。我与许多开发人员进行了交流，他们都告诉我，在他们的软件开发工作中，Qt 的确与众不同。这让身为程序员的我感觉良好。

Qt 作为一个很棒的工具和类库，这仅仅是它取得成功的部分原因。用户还需要良好的文档、一些教程和书籍。毕竟，让开发人员生活更轻松才是目标。

这是我所深信不疑的，回溯到 2003 年也是如此。当时，我是 Trolltech 的总裁，负责文档的 Mark Summerfield 走进了我的办公室。他希望和 Jasmin Blanchette 一起写一本关于 Qt 的书。一部非常好的书，要由一个对产品知识无限熟悉且能够清晰、直观说明事物的人来撰写。当时，还有谁能够比 Qt 的文档负责人、同时也是最好的 Qt 开发人员的人，更适合这项工作呢？

最终的结果就诞生了一部关于 Qt 的伟大图书，并随后进行了更新和扩展。

Mark 现在又完成了另一项重要的工程。

在 Qt 编程人员的武器库中，就差一部有关 Qt 高级编程的书籍。我非常高兴，Mark 已经写完了这样的一部书。他是一名非常好的撰写技术书籍的作家，拥有撰写 Qt 编程书籍所应具有的全

部背景知识，是最具权威的人选。他专注于细节和清晰直观表达自己的能力总是令我印象深刻。也就是说，您一定会满意的！

在您手中（或通过屏幕阅读），您正紧握着一次扩充自己知识的极好机会，它可以让你用 Qt 做出许多超酷的东西。

编程快乐！

Eirik Chambe-Eng
于法国，南阿尔卑斯山
2009 年 12 月 24 日

前　言

一段时间以来，我一直想写一本 Qt 书籍，一本能够涵盖《C++ GUI Qt 4 编程》一书中过于高深内容的书籍，尽管对一些读者来说，该书本身已经够有挑战性了。还有一些我打算涉及的专题材料（并非是比较难的）而是它们并没能包含进第一本关于 Qt 编程的书中。此外，从 Qt 庞大的规模上来看，也没有哪一本书能够对 Qt 所有的内容进行毫无偏颇的描述。毫无疑问，这为新技术文稿的撰写留下了空间。

这本书所做的就是从许多模块和各个方面的类中选择了一些内容，并展示该如何使用它们。这些所选择的主题都是我自己感兴趣的，同时好像也正是它们在 Qt 爱好者邮件群 qt-interest 中引起了许多讨论。这些主题中的一些主题还没有在任何其他书籍中涉及过，而另外一些主题则较为熟悉，比如，模型/视图编程。无论如何，我将尽量提供比其他可借鉴材料更为全面的内容。

因此，这本书的目的就是帮助 Qt 编程人员加深和拓宽他们的知识，提高他们使用 Qt 成就事业的能力。“高级”方面通常更多地是指能做到什么，而不是实现方法的手段。这是因为，正如常说的那样，Qt 让我们尽可能远离不相关的细节和潜在的复杂事物，提供易于使用的应用程序接口（API），从而只需简单、直接地使用就可以获得极好的效果。例如，我们将会看到：在不知道任何播放器工作原理的情况下，创建一个音乐播放器的过程；而所需要了解的仅仅是 Qt 所提供的那些高级 API。另一方面，即使对于高级 QtConcurrent 模块的用法，它所涵盖的对多线程的必要知识也都很有挑战性。

这本书假设读者都具有基本的 C++ 编程能力，并且至少知道如何来创建基本的 Qt 应用程序——例如，已经读过一部好的 Qt 4 书籍，并有一定的工程实践经验。本书还认为，读者应该熟悉 Qt 的参考文档，至少能够使用它查询到感兴趣的类的 API。此外，一些章节会假设读者已经知道相关主题的基本知识——例如，第 1 章会假设读者已经知道一些 JavaScript 和 Web 编程的知识，在多线程的那些章节里，作者会假设读者能够理解线程的基本知识和 Qt 的线程类。所有这些假设都意味着，这本书将能够免于介绍那些 Qt 程序开发人员已经熟知的许多细节和类，比如布局的使用、动作的创建、信号和槽的连接等，从而可以让本书完全专注于那些读者不是很熟悉的知识。

当然，没有哪部单卷本书籍可以真正毫无偏颇地描述那 700 多个 Qt 公共类——在 Qt 4.6 中，几乎有 800 个，以及 100 多万字的 Qt 文档，所以本书也不会试图去那样做。相反，这本书为如何使用 Qt 最具强大功能的那些特征提供了一些说明和示例，用来补充参考文档而不是对它的重复。

本书在章节设计上，已尽可能做到内容完整，因而也就没有必要按照章节顺序自始至终地进行阅读。为了实现这一点，对于不同章节中要用到的那些特定技术，仅会在一个地方进行说明，而在其他地方则会使用交叉引用的方式给出。即使如此，如果你打算随机阅读一些零星章节，建议至少先对整本书做一个粗略的浏览，因为一些章节会专注于某个特定主题，而它又是其他主题必不可少的材料。同样，我将尽可能多地介绍那些完全来自 Qt 的 API 的小细节，以使本书的内容更为丰富，并在上下文中尽可能多地介绍那些特性，因而通篇会出现一些有用的信息。

与我之前那些书一样，本书中引用的代码段都是些“活代码”，也就是说，这些代码都是直接从例子的源文件中自动抽取并直接嵌入到送给出版商的 PDF 文件中的——因而就不会有剪切、粘贴方面的错误，而且可保证代码能够正常工作。这些例子可以从 www.qtrac.eu/aqpbook.html 获得，基于 GPL（GNU General Public License，GNU 通用公共授权第 3 版）进行授权。本书将给出多达 25

个例子，分布在150多个.hpp和.cpp文件中，累计超过20 000行代码。尽管全部最为重要的代码段都在书中进行了引用和解释，但还有大量的细节无法在这本书内进行阐释，因此，建议下载这些示例并至少阅读一下那些你所特别感兴趣的例子的源代码。除了这些例子，本书还提供了一些包含有常用功能的模块。所有这些都用AQP命名空间来确保其重用性，开头的一些章节会将它们引进来，然后会在整本书中一直使用。

所有例子（除了最后一章中用到了Qt 4.6特性的那些例子）都用Qt 4.5和Qt 4.6在Linux、Mac OS X和Windows平台上进行了测试。使用Qt 4.5建立的那些应用程序将可以在Qt 4.6下不做修改而直接运行，对后续的其他Qt 4.*x*版本也可以运行，因为Qt在各个次要发行版中维持向后兼容。然而，对于这两个Qt版本之间的那些不同之处，本书会说明和解释与Qt 4.6相关的方法，而源代码部分会使用#if QT_VERSION，以便可以用特定的版本或者最好的习惯来编译代码。一些例子或许可用于先前的Qt 4.*x*版本，特别是Qt 4.4，且一些例子或许可以向后移植（backport）到更早的Qt版本——然而，这本书仅仅完全关注于Qt 4.5和Qt 4.6，所以不会明确涉及到向后移植的问题。

本书给出了最好的Qt 4.6实践，尽管Qt 4.6比Qt 4.5包含更多的新特征，但对代码来说却并没有太多不同。一个细微差别之处在于：Qt 4.6有“退出”（quit）动作的快捷方式而Qt 4.5没有；源代码中，对于Qt 4.6会使用其快捷方式，而对于Qt 4.5，则会用#if QT_VERSION表示与之功能相当的代码。更为重要的不同之处在于，Qt 4.6引入了QGraphicsObject类，而且还在它与几何形状变化通信时改变了那些图形项（graphics item）的行为。我们会在某些地方说明这些不同之处，并在书中的代码段中给出Qt 4.6的方法，但是在源代码中，用#if QT_VERSION来说明如何用Qt 4.6和Qt 4.5及其早期版本来完成同样的事情，并为两者选择最好的方法。在本书的最后一章，作为之前给出例子的转换，用三个例子中的两个来说明与Qt 4.6相关的那些特性，以及对Qt 4.6动画和状态机框架的应用。通过修改之前的例子，就更容易看出如何从传统的Qt方法过渡到新的框架下。

Qt的下一个版本，Qt 4.7将重点关注于稳定性、速度以及除Qt Quick之外的新技术（可提供一种使用类JavaScript语言创建GUI声明的方法），我们希望引入比之前发行版更少的新特性。尽管现在仍然有巨大的精力投入到Qt中，其范围也在不断扩大，但本书应当作为学习和使用Qt 4.*x*系列方面重要技术的一个有用资源，特别是对Qt 4.5、Qt 4.6和若干年后就要来临的那些后续版本来说。

致谢

我第一个要感谢的是我的朋友Trenton Schulz，Nokia公司Qt开发框架组（Qt Development Frameworks，之前的Trolltech公司）中的一名前任软件工程师，他目前是挪威计算中心（Norwegian Computing Center）的一名研究学者。事实证明，Trenton是一名可靠的、富有远见的和挑战性的审稿人，他阅读仔细、标准严格，他提出的一些建议对改进本书相当有帮助。

接下来要感谢的是另外一位朋友，Jasmin Blanchette，他以前也是Qt开发框架中的一名软件工程师，与我一起合著了*C++ GUI Programming with Qt* 4[①]一书，目前正在慕尼黑工业大学攻读博士

① 本书的中文译名是《C++ GUI Qt 4编程》，已由电子工业出版社于2008年8月出版发行，详情请参阅http://www.phei.com.cn/bookshop/bookinfo.asp?bookcode=TP070380%20&booktype=main或www.china-pub.com/42122——译者注。

学位。我们两个在前一段时间对这本书就形成了一致意见，而仅仅是因为工作的压力让他成为了一名出色的而且是苛刻的审稿人，而不是合著者。

我还要感谢很多那些工作（或任职）于 Qt 开发框架的人，他们阅读了该书的一些部分并提供了有益的反馈信息，还要感谢那些回答了技术问题的人，还有同时做了以上两件事的人。这些人包括：Andreas Aardal Hanssen（对图形/视图那几章给出了特别优秀的反馈和建议，并为我列出了离屏渲染方面的补充材料）、Andy Shaw、Bjørn Erik Nilsen、David Boddie、Henrik Hartz、Kavindra Devi Palaraja、Rainer Schmid（目前在 Froglogic）、Simon Hausmann、Thierry Bastian 和 Volker Hilsheimer。

意大利软件公司（www. develer. com）是一家很好的一个软件公司，为我提供了免费主机，让我能够在漫长的写作过程中安心完成这本书。他们的一些开发人员给了我有用的反馈，特别是早期章节中的一些例子。我特别感谢 Gianni Valdambrini、Giovanni Bajo、Lorenzo Mancini（为我创建了资料库）和 Tommaso Massimi。

特别感谢初稿读者 Alexey Smirnov，他指出了一些错误，并鼓励我在一些网络示例中加入对网络代理的支持。

我还要感谢 Froglogic 的创始人，Reginald Stadlbauer 和 Harri Porten——他们提供给我的兼职顾问的工作，以帮我找到了写作这本书的时间，也同时向我介绍了一些编程技术，它们对我来说都是一些全新的想法。他们还把我变成了他们的 GUI 应用程序测试工具——Squish 的超级爱好者。

我的朋友 Ben Thompson 也应得到许多感谢，他帮我回忆起一些已经忘却的、可靠的数学概念，并且尤其要感谢他的耐心，一遍遍地向我解释这些数学概念直到我能够理解为止。

若没有 Qt，这本书（以及其他一些书）就不会成为现实。因此，我非常感谢 Qt 的创始人 Eirik Chambe-Eng 和 Haavard Nord，尤其要感谢 Eirik，他允许我在 Trolltech 的时候，把撰写我的第一本书作为日常工作，并且他还花费时间和精力来为这本书写了序言。

要特别感谢我的编辑，Debra Williams Cauley，相当独到地建议我优先撰写这本书，并且，她的支持和帮助让工作取得了实质性的进展。还要感谢 Jennifer Lindner 在书籍的结构和其他反馈方面做出的有效努力。还要感谢 Audrey Doyle，她对本书的出版管理工作认真负责，还要感谢审校读者 Barbara Wood，做出了那么好的审校工作。

我还要感谢我的妻子 Andrea，她与我一起经历了撰写过程中的坎坎坷坷，感谢她不朽的爱和无尽的支持！

目　录

第1章　混合桌面/Internet应用程序

- Internet相关窗口部件
- Webkit的使用

目前，无处不在的“云计算”，依靠网络驱动的手机，体积小巧的上网笔记本计算机和智能笔记本产品（更不用说Google Doc的文件存储系统了），再加上基于网络的零部署成本的应用程序，这一切都使桌面应用程序成为即将灭绝的恐龙——但它们对此却熟视无睹。

在我们抛弃C++和Qt而转向网络程序，体验JavaScript和HTML所带来的微妙乐趣之前，回顾一下桌面应用程序所能带来的优势还是很有必要的。

- 可用性——在特定的负有关键任务的区域之外，我们相信很少（通常是因为不方便）会出现Internet不可用的情况，诸如因网络故障、ISP错误等。此时，那些基于网络的应用程序将毫无用处[①]。
- 资源获取途径——桌面应用程序可以不受任何限制地获取用户计算机上的所有资源，但基于网络的应用程序却会因为安全限制而不能发挥全部功能。
- 观感（look and feel）——桌面应用程序除了自身的菜单栏和工具条之外，没有多余的（让人迷惑不解的）浏览器菜单栏和工具条。它拥有自己的快捷键，并且不会与浏览器的快捷键相冲突。它的观感永远是在程序产生时设定好的，而基于Internet的应用程序的观感却会因为浏览器的改变而有所不同。
- 自定义窗口部件——桌面应用程序可以提供给用户一些具有常用功能的控件，并且，它的性能是网络应用程序无法比拟的。

在最为理想的情况下，我们能够拥有桌面应用程序所带来的全部优点，并且在网络可用时能够享受Internet所带来的好处。感谢Qt 4.4引入的QtWebKit模块，它能够实现我们的这一梦想。QtWebKit可以创建在线（online）和离线（offline）都能使用的混合桌面/互联网应用程序。

与基于Internet的应用程序相比，Qt的劣势主要在于其部署方面——它只能在用户的计算机上运行。如果部署消耗或带宽利用率必须为最小值时，可以采取一些办法来应对这种情况，例如，将大部分程序功能放在一些可以独立更新的小插件里。又或者，使用为JavaScript（ECMAScript语言）开发的QtScript模块或如果想使用不同的脚本语言的话，可以使用第三方的脚本模块——利用程序脚本来提供应用程序的大部分功能，再根据需求更新或添加独立脚本即可。还可以将尽可能多的功能放在服务器和网页脚本中，这样就可以节省很多为升级客户端而消耗的时间。

本章的重点在于Qt所提供的支持混合应用程序的关键技术。在1.1节中，我们将使用Qt 4.4中引入的便捷类QNetworkAccessManager来创建与互联网相关的窗口部件；在1.2节中，将使用QtWebKit模块，首先开发一个通用网络浏览器窗口部件——这借助于QtWebkit模块所提供的功能可以非常容易地实现它。然后，将使用刚才开发的通用浏览器组件，例如创建一个特定网站应用程序——利用QtWebKit读取在后台下载网页的DOM（Document Object Model，文档对象模型），这

---

① 例如，可以参阅opencloudcomputing.info/trends/cloud-computing-downtime或云计算事件数据库（Cloud Computing Incidents Database，CCID）。

样就可以从中为后续的开发工作提取信息。我们将看到如何把 Qt 窗口部件,甚至是自定义的窗口部件嵌入到网页中去,来提供一些使用标准 HTML 窗口部件无法实现的功能。

1.1 Internet 相关窗口部件

对 Internet 相关窗口部件(Internet-aware widget)的定义是:一种自动从 Internet 检索数据信息的窗口部件,它的构建可以是一次性事件,也可以是一种有规律性时间间隔的事件。

最简单地创建一个 Internet 相关窗口部件的方法是使用 QNetworkAccessManager 对象,构造出它的窗口部件子类。QNetworkAccessManager 的这些对象可以完成 HTTP(包括 HTTPS)的 HEAD、POST、GET 和 PUT 请求,并处理相关的 cookie(使用 QNetworkCookieJar)和身份验证(使用 QAuthenticator)。

我们将在本节介绍一个样例,它使用 QNetworkAccessManager 定时从 Internet 读取数据,还会介绍另外一个使用 QNetworkAccessManager 响应图片下载请求的样例。这两个示例完全可以说明如何使用 QNetworkAccessManager。图 1.1 给出了 QNetworkAccessManager 和外部网站之间的关系示意图。值得注意的是,QNetworkAccessManager 是 Qt 中 QtNetwork 模块的一部分,任何使用到该模块的应用程序都要在其.pro 文件中加一行 QT += network。

本节中的例子是一个任务栏托盘图标程序。此程序主要用来显示那些常用控制操作(如音量控制)的图标,或用来提供内存的使用状态和当前的时间等信息。此处,我们将开发一个 Weather Tray Icon 应用程序(weathertrayicon)。它能从美国国家气象服务机构(U. S. National Weather Service, www. weather. gov)检索当前某个机场的天气信息(图标和数据),然后将相应的图标显示出来。

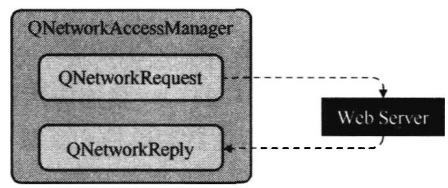


图 1.1 与外部站点通信的 QNetworkAccessManager

图 1.2 中左边的截图给出的是 Weather Tray Icon 应用程序和一个提示工具——其图标在提示工具的右下角;右边的截图显示的是该应用程序的上下文菜单,程序会每隔一个小时下载一次气象数据及相应的天气图标,还会根据下载的内容更新选定机场的天气情况。

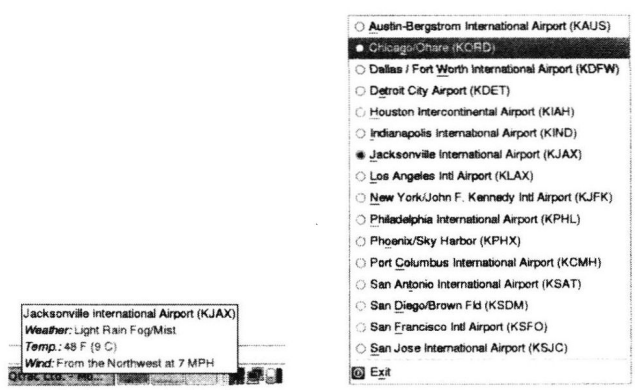


图 1.2 Weather Tray Icon 应用程序及其上下文菜单

像这样的任务栏托盘图标应用程序可以在所有支持 Qt 的桌面平台上运行。图 1.2 就是在基于 Linux 的 Fedora 系统运行 GNOME 桌面时的截图。在 Windows 和 Mac OS X 平台下,工具提示(tooltip)仅仅显示成纯文本,因为这两个平台不支持 Qt 工具提示的富文本格式(HTML)。当然,在 Mac OS X 下,图标仍然会像我们期待的那样显示在菜单栏上。

在本书中的大部分示例中，我们通常不会给出 main( ) 函数，因为它非常简单，并且经过了标准化。但在这个示例中，它与标准的 main( ) 函数稍微有点不同，所以这里将给出 Weather Tray Icon 应用程序的 main( ) 函数。

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    app.setApplicationName(app.translate("main",
                                         "Weather Tray Icon"));
    app.setOrganizationName("Qtrac Ltd.");
    app.setOrganizationDomain("qtrac.eu");
    app.setQuitOnLastWindowClosed(false);
    if (int error = enableNetworkProxying())
        return error;

    WeatherTrayIcon weatherTrayIcon;
    weatherTrayIcon.show();
    return app.exec();
}
```

该函数以标准的 Qt 方式建立了一个 QApplication 对象。设定了应用程序的名称，这会在后面用到（如用于对话框的标题）可以通过 QApplication::applicationName( ) 来获取，也可以用组织的名称和定义域来设置它，这就意味着，我们可以在任何想创建 QSettings 对象的时候创建它，而无须给定任何参数。

这个函数有两个不同寻常的地方。第一，在最后的窗口已经关闭时，必须告诉 Qt 不要关闭应用程序。这是因为，正常情况下，一个托盘图标应用程序是没有窗口的（仅保留一个托盘图标），它所使用的任何窗口都是临时的（如一个工具提示或情景菜单），关闭它们不会导致程序的终止。

第二个方面是，自定义的 enableNetworkProxying( ) 函数的调用。我们在“网络代理的支持”的阴影部分中讨论过这个函数。如果它返回一个非零的错误代码，那么就意味着有错误出现，需要终止程序。

网络代理统一支持

对于能够直接连接到（如使用宽带调制解调器或路由器）Internet 的设备来说，本章所列举的这些例子都能够正常运行。然而，对处于带有防火墙网络的那些内部设备来说，尤其是企业网络，这些示例可能无法连接到 Internet。大部分带有防火墙的网络都有某种能够提供网络连接的代理服务器。Qt 为这些代理提供了支持，所以我们已经为 browserwindow、nyrbviewer、rsspanel 和 weathertrayicon 等示例提供了代理支持，这可以通过在 main( ) 函数内部调用自定义的 enableNetworkProxying( ) 函数来实现。

enableNetworkProxying( ) 函数利用 AQP::OptionParser（由本书在 option_parser 中并且使用 AQP 命名空间的示例提供）解析用来建立代理的命令行参数。启用代理的应用程序支持的命令行选项有：

-h	--help	显示帮助信息并终止
-H	--host = STRING	主机名，如 www.example.com
-P	--password = STRING	密码
-p	--port = INTERGER	端口号，如 1080
-t	--type = STRING	代理类型（http，socks5；默认为 socks5）
-u	--username = STRING	用户名

只有在指定了主机名后才能在 enableNetworkProxying( ) 函数中建立代理。以下是其代码，解析器的类型是 AQP::OptionParser。

```
if (parser.hasValue("host")) {
    QNetworkProxy proxy;
    proxy.setType(parser.string("type") == "socks5"
            ? QNetworkProxy::Socks5Proxy
            : QNetworkProxy::HttpProxy);
    proxy.setHostName(parser.string("host"));
    if (parser.hasValue("port"))
        proxy.setPort(parser.integer("port"));
    if (parser.hasValue("username"))
        proxy.setUser(parser.string("username"));
    if (parser.hasValue("password"))
        proxy.setPassword(parser.string("password"));
    QNetworkProxy::setApplicationProxy(proxy);
}
```

如果已经指定了主机，那么就可以利用已给出的主机、默认或给定的代理类型和用户拥有的其他信息来建立代理。在这个示例中，为整个程序设定了全局代理。也可以使用 QAbstractSocket::setProxy( ) 分别为每一个 socket 建立代理。

程序会提供多种格式的天气数据，但这里选择使用 XML 格式。此格式非常简单，实际上只包含了成对出现的键值列表，键为一个标签名，其值是位于开始和关闭标签之间的文本，例如：

```
<weather>Fair</weather>
<temperature_string>49 F (9 C)</temperature_string>
<temp_f>49</temp_f>
<temp_c>9</temp_c>
<wind_string>From the Northeast at 5 MPH</wind_string>
<visibility_mi>9.00</visibility_mi>
<icon_url_base>http://weather.gov/weather/images/fcicons/</icon_url_base>
<icon_url_name>nskc.jpg</icon_url_name>
```

当程序第一次启动时，它将机场设定为用户上一次设定，或者是当程序第一次运行时默认指定的某个值。然后，利用 QNetworkAccessManager 取得天气数据。气象数据包含的两个元素为：URL 和一个表示机场当前天气状况图标的文件名。应用程序使用第二个 QNetworkAccessManager 来获取图标，并把它作为任务栏托盘里的程序图标。实际上，程序会将图标隐藏起来以节省带宽，稍后我们将会看到这种情况。

```
class WeatherTrayIcon : public QSystemTrayIcon
{
    Q_OBJECT

public:
    explicit WeatherTrayIcon();

private slots:
    void requestXml();
    void readXml(QNetworkReply *reply);
    void readIcon(QNetworkReply *reply);
    void setAirport(QAction *action);

private:
    ...
    QMenu menu;
```

```
    QNetworkAccessManager *networkXmlAccess;
    QNetworkAccessManager *networkIconAccess;
    QString airport;
    QCache<QUrl, QIcon> iconCache;
    int retryDelaySec;
};
```

稍后会给出并解释所有的方法,包括那些没有给出的私有方法,但现在我们要先对该类的一些私有成员进行说明。airport 字符串的值为当前选定的机场,如“Chicago/Ohare(KORD)”。iconCache 拥有 QUrl 键和指向 QIcon 的指针。其余的成员变量我们会在讨论那些用到它们的函数时再给出说明。

QCache 类使用“成本”(cost)模式来缓存各个项。可以缓存的项的最大值默认为 100(成本项的总数一般等于或小于最大设定值)。默认情况下,每个项的成本值都是 1,因此,当在没有修改该最大值或自行设定我们自己的项的成本值时,能够缓存高达 100 个项。在增加一个新项时,如果该项的成本超过了最大的成本值,那么就会移除一个或多个最近访问的项,直到项的成本总数小于或等于最大值。

QCache 在后台利用 QHash 通过键值进行快速轮询。然而,出乎意料的是,QHash 不能将 QUrl 存储为键值,因为 Qt 没有提供 qHash(QUrl)函数①。这可以通过添加一行代码来做到。

```
inline uint qHash(const QUrl &url) { return qHash(url.toString()); }
```

在这里,我们仅把任务传递给内置的 qHash(QString)函数。

我们现在已经为查看这些方法做好了准备,就先从构造函数开始。

```
WeatherTrayIcon::WeatherTrayIcon()
    : QSystemTrayIcon(), retryDelaySec(1)
{
    setIcon(QIcon(":/rss.png"));
    createContextMenu();

    networkXmlAccess = new QNetworkAccessManager(this);
    networkIconAccess = new QNetworkAccessManager(this);
    connect(networkXmlAccess, SIGNAL(finished(QNetworkReply*)),
            this, SLOT(readXml(QNetworkReply*)));
    connect(networkIconAccess, SIGNAL(finished(QNetworkReply*)),
            this, SLOT(readIcon(QNetworkReply*)));

    QTimer::singleShot(0, this, SLOT(requestXml()));
```

我们给定程序一个初始图标,在等待第一个天气图标下载时,可以先暂时使用它。然后创建一个带有动作的上下文菜单,用它来改变机场和终止程序。

绝大多数的构造函数都专门用于通过创建两个 QNetworkAccessManager 来建立 Internet 访问路径。其中一个用来获取天气数据,另一个用来获取当前天气情况相关的图标。它们分别使用单独的网络路径管理器,以便使其独立工作,对于这两条路径,仅创建一个单独的信号-槽连接,因为我们感兴趣的是每个下载是什么时候完成的。

最后,我们使用一个单计时器来调用 requestXml()槽。这种方法使用 networkXmlAccess 网络路径管理器来取得当前机场的天气数据。

我们已经简单、直接地调用了 requestXml(),但考虑到编程风格,还是建议尽量少用在各个构造函数中构造一个对象的“create”方法,也尽量少用单触发器来调用那些构造后初始化方法(post-construction initializing method)。这样做可确保在调用初始化方法时,对象已经完全构造好了。这意味着初始化方法可以访问任何的成员变量或方法,但在对象的构造过程中这样做则不一定安全。

---

① Qt 4.7 已经计划提供一个 qHash(QUrl)函数。

在介绍 requestXml( )槽之前,先简要看一下上下文菜单是如何建立的,以弄明白用户可以设置机场的哪些详细信息必须下载下来。

```
void WeatherTrayIcon::createContextMenu()
{
    QStringList airports;
    airports << "Austin-Bergstrom International Airport (KAUS)"
             ...
             << "San Jose International Airport (KSJC)";
    QSettings settings;
    airport = settings.value("airport", QVariant(airports.at(0)))
                      .toString();

    QActionGroup *group = new QActionGroup(this);
    foreach (const QString &anAirport, airports) {
        QAction *action = menu.addAction(anAirport);
        group->addAction(action);
        action->setCheckable(true);
        action->setChecked(anAirport == airport);
        action->setData(anAirport);
    }
    connect(group, SIGNAL(triggered(QAction*)),
            this, SLOT(setAirport(QAction*)));
    menu.addSeparator();
    menu.addAction(QIcon(":/exit.png"), tr("E&xit"), qApp,
                   SLOT(quit()));
    AQP::accelerateMenu(&menu);
    setContextMenu(&menu);
}
```

这里通过硬编码(hard-coded)列表给出了机场的名字,但它们本可以简单地从文件或资源文件中读取出来(如果想列出所有美国的机场,可以简单地将其分组,如可以通过将州作为在顶部菜单项,机场作为子菜单项的方法)。使用 QSettings 设定当前机场,在初次运行程序时,默认为列表中的第一个机场。

为每一个机场创建一个 QAction,并为当前机场选择一个相匹配的动作。把所有的机场动作添加到一个 QActionGroup 中。默认情况下,QActionGroup 拥有设置为 true 的唯一属性。这就保证了它的每个动作都拥有的是单选按钮而不是复选按钮,每次也只能选中一个机场。

还向程序添加了一个退出(exit)动作,并为其指定了一个专有的键盘加速键。然后,调用 AQP::accelerateMenu( )为尽可能多的机场提供键盘加速键,并将之前建立的菜单设定为应用程序的上下文菜单。如果是在 Mac OS X 平台上编写程序,正常情况下,调用 AQP::accelerateMenu( )是无效的,因为该平台不支持加速键。关于自动设定键盘加速键的更多内容,请参阅"键盘加速键"的阴影部分。

要把每个动作连接到 setAirport( )槽,并以某种方式参数化每个槽的调用,以便让槽知道选择的是哪个机场。一种简单的办法是在槽内部调用 QObject::sender( ),以查看出是哪个动作调用了它,然后提取该动作的文本来判别出所选择的机场。另外一种方法是使用 QSignalMapper。但在这种情况下,还有一种更为容易的解决方案——连接 QActionGroup,而不是每一个机场的 QAction。QActionGroup::triggered( )信号会把相关的 QAction 作为它的参数。

```
void WeatherTrayIcon::requestXml()
{
    QString airportId = airport.right(6);
    if (airportId.startsWith("(") && airportId.endsWith(")")) {
        QString url = QString("http://www.weather.gov/xml/"
                "current_obs/%1.xml").arg(airportId.mid(1, 4));
        networkXmlAccess->get(QNetworkRequest(QUrl(url)));
    }
}
```