

# The-Qt4-Book

Jasmin B., Mark S.<sup>①</sup> | 德山书生<sup>②</sup>

版本： 0.01

---

① 作者全名：Jasmin Blanchette, Mark Summerfield。本文代码来自[这个网站](#)。原中文翻译：闫锋欣，曾泉人，张志强；原中文审校：周莉娜，赵延兵。

② 编者：德山书生，湖南常德人氏。我负责整理排版工作，关于版权我只有一句话，希望我的工作能够帮助更多的人的学习和研究。本项目 Github 网站在[这里](#)。有意见请反馈。编者邮箱：[a358003542@gmail.com](mailto:a358003542@gmail.com)。

## 前言

为什么会是 Qt? 为什么像我这样的程序员会选择 Qt? 这个问题的答案显而易见: Qt 单一源程序的兼容性、丰富的特性、C++ 方面的性能、源代码的可用性、它的文档、高质量的技术支持, 以及在奇趣科技公司那些精美的营销材料中所涉及的其他优势等。这些答案看起来确实都不错, 但是遗漏了最为重要的一点: Qt 的成功缘于程序员们对它的喜欢。

那么, 是什么让程序员喜欢某种技术而放弃另外一种呢? 就我而言, 我认为软件工程师们喜欢某种技术, 是因为他们觉得这种技术是合适的, 但是这也会让他们讨厌所有那些他们觉得不合适的其他技术。除此之外, 我们还能解释下面的这些情况吗? 例如, 一些最出众的程序员需要在帮助之下才能编写出一个录像机程序, 或者又比如, 似乎大多数工程师在操作本公司的电话系统时总会遇到麻烦。我虽然善于记住随机数字和指令的序列, 但是如果将其比作用于控制我的应答系统所需要的条件来说, 则可能一条也不具备。在奇趣科技公司, 我们的电话系统要求在拨打其他人的分机号码前, 一定要按住“\*”键 2 秒后才允许开始拨号。如果忘记了这样做而是直接拨打分机号码, 那么就不得不再重新拨一遍全部的号码。为什么是“\*”键而不是“#”键、“1”键或者“5”键? 或者为什么不是 20 个电话键盘中的其他任何一个呢? 又为什么是 2 秒, 而不是 1 秒、3 秒或者 1.5 秒呢? 问题到底出在哪里? 我发现电话很气人, 所以我尽可能不去使用它。没有人喜欢总是去做一些不得不做的随机事情, 特别是当这些随机事情显然只出现在同样随机的情况下的时候, 真希望自己从来都没有听到过它。

编程很像我们正在使用的电话系统, 并且要比它还糟糕。而这正是 Qt 所要解决的问题。Qt 与众不同。一方面, Qt 很有意义; 另一方面, Qt 颇具趣味性。Qt 可以让您把精力集中在您的任务上。当 Qt 的首席体系结构设计师面对一个问题的时候, 他们不是寻求一个好的、快速的或者最简便的解决方案, 而是在寻求一个恰当的解决方案, 然后将其记录在案。应当承认, 他们犯下了一些错误, 并且还要承认的是, 他们的一些设计决策没有通过时间的检验, 但是他们确实做出了很多正确的设计, 并且那些错误的设计应当而且也是能够进行改正的。看一看最初设计用于构建 Windows 95 和 UNIX Motif 之间的桥梁系统, 到后来演变为跨越 Windows Vista、Mac OS X

和 GNU/Linux 以及那些诸如移动电话等小型设备在内的统一的现代桌面系统，这些事实就足以证明这一点。

早在 Qt 大受欢迎并且被广泛使用很久以前，正是 Qt 的开发人员为寻求恰当的解决方案所做出的贡献才使 Qt 变得与众不同。其贡献之大，至今仍然影响着每一个对 Qt 进行开发和维护的人。对我们而言，研发 Qt 是一种使命和殊荣。能够使您的职业生涯和开源生活变得更为轻松和更加有趣，这让我们倍感自豪。

人们乐于使用 Qt 的诸多原因之一是它的在线帮助文档，但是该帮助文档的主要目的是集中介绍个别的类，而很少讲述应当如何构建现实世界中那些复杂的应用程序。这本好书填补了这一缺憾，它展示了 Qt 所提供的东西，如何使用“Qt 的方式”进行 Qt 编程，以及如何充分地利用 Qt。本书将指导 C++、Java 或者 C# 程序员进行 Qt 编程，并且提供了丰富详实的资料来使他们成长为老练的 Qt 程序员。这本书包含了很多很好的例子、建议和说明——并且，该书也是我们对那些新加入公司的程序员们进行培训的入门教材。

如今，已有大量的商业或者免费的 Qt 应用程序可以购买或者下载，其中的一些专门用于特殊的高端市场，其他一些则面向大众市场。看到如此多的应用程序都是基于 Qt 构建而成的，这使我们充满了自豪感，并且还激励我们要让 Qt 变得更好。相信在这本书的帮助下，将会前所未有的出现更多的、质量更高的 Qt 应用程序。

Matthias Ettrich

德国，柏林

2007 年 11 月

## 0.1 序言

Qt 使用“一次编写，随处编译”的方式为开发跨平台的图形用户界面应用程序提供了一个完整的 C++ 应用程序开发框架。Qt 允许程序开发人员使用应用程序的单一源码树来构建可以运行在不同平台下的应用程序的不同版本；这些平台包括从 Windows 98 到 Vista、MacOS X、Linux、Solaris、HP-UX 以及其他很多基于 X11 的 Unix。许多 Qt 库和工具也都是 Qt/Embedded Linux 的组成部分。Qt/Embedded Linux 是一个可以在嵌入式 Linux 上提供窗口系统的产品。

本书的目标就是教您如何使用 Qt4 来编写图形用户界面程序。本书从“Hello Qt”开始，然后很快地转移到更高级的话题中，如自定义窗口部件的创建和拖放功能的提

供等。通过本书的[互联网站点](#)，您可以下载到一些作为本书文字补充材料的示例程序。附录 A 说明了如何下载和安装这些软件，其中包括一个用于 Windows 的 C++ 免费编译器。

本书分为四部分。第一部分涵盖了在使用 Qt 编写图形用户界面应用程序时所必需的全部基本概念和练习。仅掌握这一部分中所蕴含的知识就足以写出实用的图形用户界面应用程序。第二部分进一步深入介绍了 Qt 的一些重要主题，第三部分则提供了更为专业和高级的材料。您可以按任意顺序阅读第二部分和第三部分中的章节，但这是建立在您对第一部分中的内容非常熟悉的基础之上的。第四部分包括数个附录，附录 B 说明了如何构建 Qt 应用程序，附录 C 则介绍了 Qt Jambi，它是 Java 版的 Qt。

本书的第一版建立在 Qt 3 版本的基础上，尽管已通过全书修订来反映那些很好的 Qt4 编程技术，但本书还是根据 Qt4 的模型，视图结构、新的插件框架、使用 Qt/Embedded Linux 进行嵌入式编程等内容而引入了一些新的章节和一个新的附录。作为第二版，本书充分利用了 Qt 4.2 和 Qt 4.3 中引入的新特性对其进行了彻底更新，并包含“自定义外观”和“应用程序脚本”两个新的章以及两个新的附录。原有的“图形”一章已经拆分为“二维”和“三维”两章，在它们中间，涵盖了新的图形视图类和 QPainter 的 OpenGL 后端实现。此外，在数据库、XML 和嵌入式编程等几章中，还添加了许多新内容。

与本书的前两版一样，这一版的重点放在如何进行 Qt 编程的说明和许多真实例子的提供上，而不是对丰富的 Qt 在线文档的简单拼凑和总结。因为本书纯粹讲授的是 Qt 4 编程中的原理和实践知识，因而读者能够轻松学会将要出现在 Qt 4.4、Qt 4.5 以及 Qt 4.x 等后续版本中的 15 个 Qt 新模块。如果您正在使用的 Qt 版本恰好是这些后续版本中的一个，那么当然要阅读一下参考文档中的“*What's New in Qt 4.x*”一章，以便可以对那些可用的新特性有一个总体把握。

在写作本书的时候，是假定您已经具备了 C++、Java 或者 C# 的基本知识。本书中的例子代码使用的是 C++ 中的一个子集，从而避免了很多在 Qt 编程中极少使用的 C++ 特性。在某些不可避免而必须使用 C++ 高级结构的地方，会在使用时对其做出必要的解释。如果您对 Java 或者 C# 已经非常熟悉但是对 C++ 还知之不多甚至一无所知，那么建议您先阅读附录 D。附录 D 提供了对 C++ 较为充分的介绍，从而能够让您具有使用本书所必备的 C++ 知识。对于 C++ 中的面向对象编程更为全面的介绍，建议您阅读由 P. J. Deitel 和 H. M. Deitel 编著的“*C++ How to Program*”(Prentice Hall, 2007)，以及由 Stanley B. Lippman, Josée Lajoie 和 Barbara E. Moo 编著的“*C++ Primer*”(Addison-Wesley, 2005) 这两本书。

## 0.2 Qt 简史

Qt 框架首度为公众可用是在 1995 年 5 月。它最初由 Haavard Nord（奇趣科技公司的 CEO）和 Eirik Chambe-Eng（公司总裁）开发而成。Haavard 和 Eirik 在位于挪威特隆赫姆的挪威科技学院相识，在那里，他们都获得了计算机科学的硕士学位。

Haavard 对 C++ 图形用户界面开发的兴趣始于 1988 年，当时一家瑞典公司委托他开发一套 C++ 图形用户界面框架。几年后，在 1990 年的夏天，Haavard 和 Eirik 因为一个超声波图像方面的 C++ 数据库应用程序而在一起工作。这个系统需要一个能够在 UNIX、Macintosh 和 Windows 上都能运行的图形用户界面。在那个夏天中的某天，Haavard 和 Eirik 一起出去散步，享受阳光，当他们坐在公园的一条长椅上时，Haavard 说：“我们需要一个面向对象的显示系统。”由此引发的讨论，为他们即将创建的面向对象的、跨平台的图形用户界面框架奠定了智力基础。

1991 年，Haavard 和 Eirik 开始一起合作设计、编写最终成为 Qt 的那些类。在随后的一年中，Eirik 提出了“信号和槽”的设想——一个简单并且有效的强大的图形用户界面编程规范，而现在，它已经可以被多个工具包实现。Haavard 实践了这一想法，并且据此创建了一个手写代码的实现系统。到 1993 年，Haavard 和 Eirik 已经开发出了 Qt 的第一套图形内核程序，并且能够利用它实现他们自己的一些窗口部件。同年末，为了创建“世界上最好的 C++ 图形用户界面框架”，Haavard 提议一起进军商业领域。

1994 年成为两位年轻程序员不幸的一年，他们没有客户，没有资金，只有一个未完成的产品，但是他们希望能够闯进一个稳定的市场。幸运的是，他们的妻子都有工作并且愿意为他们的丈夫提供支持。在这两年里，Haavard 和 Eirik 认为，他们需要继续开发产品并且从中赚得收益。

之所以选择字母“Q”作为类的前缀，是因为该字母在 Haavard 的 Emacs 字体中看起来非常漂亮。随后添加的字母“t”代表“工具包” (toolkit)，这是从“Xt”——一个 X 工具包的命名方式中获得的灵感。公司于 1994 年 3 月 4 日成立，最初的名字是“Quasar Technologies”，随后更名为“Troll Tech”，而公司今天的名字则是“Trolltech”。

1995 年 4 月，通过 Haavard 就读过的大学的一位教授的联系，挪威的 Metis 公司与他们签订了一份基于 Qt 进行软件开发的合同。大约在同一时间，公司雇佣了 Arnt Gulbrandsen，在公司工作的 6 年时间里，他设计并实现了一套独具特色的文

档系统，并且对 Qt 的代码也做出了不少贡献。

1995 年 5 月 20 日，Qt 0.90 被上传到 [sunsite.unc.edu](http://sunsite.unc.edu)。6 天后，在 [comp.os.linux.announce](http://comp.os.linux.announce) 上发布。这是 Qt 的第一个公开发行人版本。Qt 既可以用于 Windows 上的程序开发，又可以用于 UNIX 上的程序开发，而且在这两种平台上，都提供了相同的应用程序编程接口。从第一天起，Qt 就提供了两个版本的软件许可协议：一个是进行商业开发所需的商业许可协议版，另一个则是适用于开源开发的自由软件许可协议版。Metis 的合同确保了公司的发展，然而，在随后长达 10 个月的时间内，再没有任何人购买 Qt 的商业许可协议。

1996 年 3 月，欧洲航天局 (European Space Agency) 购买了 10 份 Qt 的商业许可协议，它成了第二位 Qt 客户。凭着坚定的信念，Eirik 和 Haavard 又雇佣了另外一名开发人员。Qt 0.97 在同年 5 月底正式发布，随后在 1996 年 9 月 24 日，Qt1.0 正式面世。到了这一年的年底，Qt 的版本已经发展到了 1.1，共有来自 8 个不同国家的客户购买了 18 份 Qt 的商业许可协议。也就是在这一年，在 Matthias Ettrich 的带领下，创立了 KDE 项目。

Qt 1.2 于 1997 年 4 月发布。Matthias Ettrich 利用 Qt 建立 KDE 的决定，使 Qt 成为 Linux 环境下开发 C++ 图形用户界面的事实标准。Qt 1.3 于 1997 年 9 月发布。

Matthias 在 1998 年加入公司，并且在当年 9 月，发布了 Qt 1 系列的最后一个版本——V 1.40。1999 年 6 月，Qt 2.0 发布，该版本拥有一个新的开源许可协议——Q 公共许可协议 (QPL, Q Public License)，它与开源的定义一致。1999 年 8 月，Qt 赢得了 LinuxWorld 的最佳库/工具奖。大约在这个时候，Trolltech Pty Ltd (澳大利亚) 成立了。

2000 年，公司发布了 Qt/Embedded Linux，它用于 Linux 嵌入式设备。Qt/Em-bedded Linux 提供了自己的窗口系统，并且可以作为 X11 的轻量级替代产品。现在，Qt/X11 和 Qt/Embedded Linux 除了提供商业许可协议之外，还提供了广为使用的 GNU 通用公共许可协议 (GPL: General Public License)。2000 年底，成立了 Trolltech Inc. (美国)，并发布了 Qtopia 的第一版，它是一个用于移动电话和掌上电脑 (PDA) 的环境平台。Qt/Embedded Linux 在 2001 年和 2002 年两次获得了 LinuxWorld 的“Best Embedded Linux Solution”奖，Qtopia Phone 也在 2004 年获得了同样的荣誉。

2001 年，Qt 3.0 发布。现在，Qt 已经可用于 Windows、Mac OS X、UNIX 和 Linux (桌面和嵌入式) 平台。Qt 3 提供了 42 个新类和超过 500 000 行的代

码。Qt 3 是自 Qt 2 以来前进历程中最为重要的一步，它主要在诸多方面进行了众多改良，包括本地化和统一字符编码标准的支持、全新的文本查看和编辑窗口部件，以及一个类似于 Perl 正则表达式的类等。2002 年，Qt 3 赢得了 *Software Development Times* 的“Jolt Productivity Award”<sup>①</sup>。

2005 年夏，Qt 4.0 发布，它大约有 500 个类和 9000 多个函数，Qt 4 比以往的任何一个版本都要全面和丰富，并且它已经裂变成多个函数库，从而使开发人员可以根据自己的需要只连接所需要的 Qt 部分。相对于以前的所有 Qt 版本，Qt 4 的进步是巨大的，它不仅彻底地对高效易用的模板容器、高级的模型/视图功能、快速而灵活的二维绘图框架和强大的统一字符编码标准的文本查看和编辑类进行了大量改进，就更不必说对那些贯穿整个 Qt 类中的成千上万个小的改良了。现如今，Qt 4 具有如此广泛的特性，以至于 Qt 已经超越了作为图形用户界面工具包的界限，逐渐成长为一个成熟的应用程序开发框架。Qt 4 也是第一个能够在其所有可支持的平台上既可用于商业开发又可用于开源开发的 Qt 版本。

同样在 2005 年，公司在北京开设了一家办事处，以便为中国及其销售区域内的用户提供服务 and 培训，并且为 Qt/Embedded Linux 和 Qtopia 提供技术支持。

通过获取一些非官方的语言绑定件 (language bmdings)，非 C++ 程序员也已早就开始使用 Qt，特别是用于 Python 程序员的 PyQt 语言绑定件。2007 年，公司发布了用于 C# 程序员的非官方语言绑定件 Qyoto。同一年，Qt Jambi 投放市场，它是一个官方支持的 Java 版 Qt 应用程序编程接口。附录 C 提供了对 Qt Jambi 的介绍。

自奇趣科技公司诞生以来，Qt 的声望经久不衰，而且至今依旧持续高涨。取得这样的成绩不但说明了 Qt 的质量，而且也说明了人们都喜欢使用它。在过去的 10 年中，Qt 已经从一个只被少数专业人士所熟悉的“秘密”产品，发展了到如今遍及全世界拥有数以千计的客户和数以万计的开源开发人员的产品。

---

<sup>①</sup> Jolt 大奖素有“软件业界的奥斯卡”之美誉，共设通用类图书、技术类图书、语言和开发环境、框架库和组件、开发者网站等十余个分类，每个分类设有一个“震撼奖”(Jolt Award)和三个“生产力奖”(Productivity Award)。一项技术产品只有在获得了 Jolt 奖之后才能真正成为行业的主流，一本技术书籍只有在获得了 Jolt 奖之后才能真正奠定其作为经典的地位。虽然 Jolt 奖项并不起决定作用，但它代表了某种技术趋势与潮流——译者注。

## 0.3 编者的话

书名修改不是想标新立异，实在是 **github** 和本地文档编译方便，不支持空格。



# 目 录

前言	<b>i</b>
0.1 序言 . . . . .	ii
0.2 Qt 简史 . . . . .	iv
0.3 编者的话 . . . . .	vii
目录	<b>viii</b>
<b>I Qt 基础</b>	<b>1</b>
<b>1 Qt 入门</b>	<b>2</b>
1.1 Hello Qt . . . . .	2

## I Qt 基础

# 1 Qt 入门

这一章介绍了如何把基本的 C++ 知识与 Qt 所提供的功能组合起来创建一些简单的图形用户界面（Graphical User Interface, GUI）应用程序。在这一章中，还引入了 Qt 中的两个重要概念：一个是“信号和槽”，另外一个“布局”。第 2 章还将对它们做进一步的阐述，而第 3 章将着手创建一个具有真正意义的应用程序。

如果你已经熟知 Java 或 C#，但对 C++ 的编程经验还有些欠缺的话，那么在开始阅读本书之前，可能需要先阅读附录 D，它对 C++ 做了简要介绍。

## 1.1 Hello Qt

我们先从一个非常简单的 Qt 程序开始。首先一行一行地研究这个程序，然后将会看到如何编译并运行它。

```
1  #include <QApplication>
2  #include <QLabel>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication app(argc, argv);
7      QLabel *label = new QLabel("Hello Qt!");
8      label->show();
9      return app.exec();
10 }
```

第 1 行和第 2 行包含了类 QApplication 和 QLabel 的定义。对于每个 Qt 类，都有一个与该类同名（且大写）的头文件，在这个头文件中包括了对该类的定义。

第 5 行创建了一个 `QApplication` 对象，用来管理整个应用程序所用到的资源。这个 `QApplication` 构造函数需要两个参数，分别是 `argc` 和 `argv`，因为 Qt 支持它自己的一些命令行参数。

第 6 行创建了一个显示 “Hello Qt!” 的 `QLabel` 窗口部件 (widget)。在 Qt 和 UNIX 的术语 (terminology) 中，窗口部件就是用户界面中的一个可视化元素。该词起源于 “window gadget” (窗口配件) 这两个词，它相当于 Windows 系统术语中的 “控件” (control) 和 “容器” (container)。按钮、菜单、滚动条和框架都是窗口部件。窗口部件也可以包含其他窗口部件，例如，应用程序的窗口通常就是一个包含了一个 `QMenuBar`、一些 `QToolBar`、一个 `QStatusBar` 以及一些其他窗口部件的窗口部件。绝大多数应用程序都会使用一个 `QMainWindow` 或者一个 `QDialog` 来作为它的窗口，但 Qt 是如此灵活，以至于任意窗口部件都可以用作窗口。在本例中，就是用窗口部件 `QLabel` 作为应用程序的窗口的。

第 7 行使 `QLabel` 标签 (label) 可见。在创建窗口部件的时候，标签通常都是隐藏的，这就允许我们可以先对其进行设置然后再显示它们，从而避免了窗口部件的闪烁现象。

第 8 行将应用程序的控制权传递给 Qt。此时，程序会进入事件循环状态，这是一种等待模式，程序会等候用户的动作，例如鼠标单击和按键等操作。用户的动作会让可以产生响应的程序生成一些事件 (event，也称为 “消息”)，这里的响应通常就是执行一个或者多个函数。例如，当用户单击窗口部件时，就会产生一个 “鼠标按下” 事件和一个 “鼠标松开” 事件。在这方面，图形用户界面应用程序和常规的批处理程序完全不同，后者通常可以在没有人为干预的情况下自行处理输入、生成结果和终止。

为简单起见，我们没有过多关注在 `main()` 函数末尾处对 `QLabel` 对象的 `delete` 操作调用。在如此短小的程序内，这样一点内存泄漏 (memory leak) 问题无关大局，因为在程序结束时，这部分内存是可以由操作系统重新回收的。

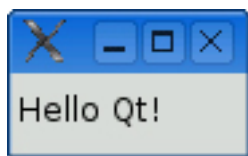


图 1-1: Linux 上的 Hello 程序