Inhaltsverzeichnis

Einleitung **17**

Vorbereitungen **21**

1 Grundbegriffe, Werkzeuge und erster Code **23**

1.1 Das erste Qt-Programm 23

1.1.1 Kompilieren eines Qt-Projekts 25

1.2 Layout, Objekthierarchien und Speicherverwaltung 27

1.2.1 Widgets automatisch anordnen 27

1.2.2 Speicherverwaltung in Objekthierarchien 29

1.2.3 Weitere Layout-Typen 30

1.3 Signale und Slots 32

1.3.1 Der einfachste Fall: Slot reagiert auf Signal 32

1.3.2 Signale mit Zusatzinformationen und deren Verarbeitung . 34

1.4 Basisklassen in Qt 37

1.4.1 Von QObject abgeleitete Klassen 37

1.4.2 QString und andere Klassen, die nicht auf QObject zurückgehen 37

1.4.3 Die Qt-Ableitungshierarchie 39

1.5 Das Qt-Paket im Überblick 39

1.5.1 Programm-Bibliotheken 40

1.5.2 Werkzeuge und Hilfsprogramme 44

1.5.3 Beispiele und Demos 54

1.6 Umgang mit der Dokumentation 54

Copyright (C) Open Source Press

2 Das Handwerkszeug zum Erstellen von Dialogen 57
2.1 Was unterscheidet Dialoge von Widgets? . . . 58
2.1.1 Ableiten von QObject . . . 60
2.1.2 Komplexere Layouts . . . 61
2.1.3 Benutzbarkeit verbessern . . . 64
2.1.4 Slots implementieren . . . 66
2.2 Trennung von GUI und Verarbeitungslogik . . . 69
2.2.1 Alternatives Design . . . 69
2.2.2 Signale deklarieren und aussenden . . . 71
2.2.3 Benutzen eigener Signale . . . 75

3 GUI-Gestaltung mit dem Qt Designer 77
3.1 Dialoge „per Mausklick" . . . 77
3.1.1 Layouten im Designer . . . 79
3.1.2 Der Property-Editor . . . 80
3.1.3 Die Vorschau . . . 83
3.1.4 Signal-Slot-Verbindungen . . . 83
3.1.5 Tab-Reihenfolge . . . 84
3.1.6 Tastenkürzel und Buddies . . . 85
3.2 Designer-Dateien in Qt-Projekte einbinden . . . 86
3.2.1 Designer-generierte Klassen als Hilfsklasse nutzen . . . 87
3.2.2 Designer-generierte Widgets stets verfügbar halten . . . 89
3.2.3 Multiple Vererbung . . . 90
3.3 Automatische Signal-Slot-Verbindungen . . . 92
3.4 Abgeleitete Klassen im Designer einfügen . . . 94
3.5 Der Ressourceneditor . . . 94

4 Eine GUI-Applikation mit Hauptfenster 97
4.1 Anatomie eines Hauptfensters . . . 97
4.2 Von QMainWindow ableiten . . . 99
4.3 Hauptfenster mit dem Designer entwerfen . . . 102
4.3.1 Menüleisten hinzufügen . . . 103
4.3.2 Aktionen in der Werkzeugleiste wiederverwerten . . . 104

Copyright (C) Open Source Press

4.3.3 Hauptfenster-Entwurf und Quellcode vereinen 107
4.4 Die Statusleiste ausreizen 115
4.4.1 Temporäre Meldungen 116
4.4.2 Normale Meldungen 117
4.4.3 Permanente Meldungen 118
4.5 Werkzeugleisten 121
4.6 Wie funktionieren Aktionen? 123
4.6.1 QAction manuell instanzieren 123
4.6.2 Anwählbare Aktionen 125
4.6.3 Gruppierte Aktionen 125
4.7 Dockfenster 127
4.7.1 Positionierung 129
4.7.2 Dockfenster für einen Editor 130
4.8 Einstellungen speichern 133
4.8.1 Fallbeispiel CuteEdit 136

5 Layouts 139
5.1 Manuelles Layout 139
5.2 Automatisches Layout 141
5.2.1 Horizontales und vertikales Layout 142
5.2.2 Raster-Layout 146
5.2.3 Verschachtelte Layouts 147
5.3 Splitter 148
5.3.1 Verhalten bei Größenänderung 148
5.3.2 Splitter-Positionen speichern und Widgetgrößen bestimmen 149
5.3.3 Relative Größen definieren 150
5.3.4 Handles anpassen 151
5.3.5 Layout bei von rechts nach links geschriebenen Sprachen . 154
5.4 Stapellayouts 155
5.4.1 Die Alternative: Stapelwidgets 155
5.4.2 Einsatzbereiche für Stapellayouts und -widgets 156

Copyright © Open Source Press

6 Dialoge **159**
6.1 Modale Dialoge . . . 159
6.2 Nichtmodale Dialoge . . . 161
6.2.1 Mögliche Usability-Probleme . . . 161
6.3 Semimodale Dialoge . . . 162
6.4 Überfrachtete Dialoge vermeiden . . . 162
6.5 Vorgefertigte Dialoge in Qt . . . 164
6.5.1 Nachrichtendialoge . . . 164
6.5.2 Einmalig sichtbare Fehlermeldungen . . . 172
6.5.3 Dateiauswahldialoge . . . 173
6.5.4 Eingabedialoge . . . 177
6.5.5 Fontauswahldialog . . . 181
6.5.6 Farbauswahl- und Druckdialog . . . 182

7 Events, Drag&Drop und Zwischenablage **183**
7.1 Event-Loop und Event-Handler . . . 183
7.2 Events behandeln . . . 184
7.2.1 Spezialisierte Event-Handler verwenden . . . 184
7.2.2 Den allgemeinen Event-Handler verwenden . . . 187
7.3 Eventfilter nutzen . . . 188
7.4 Drag&Drop . . . 192
7.4.1 MIME-Typen . . . 192
7.4.2 Die Drag-Seite . . . 194
7.4.3 Die Drop-Seite . . . 196
7.5 Die Zwischenablage . . . 199

8 Datenvisualierung mit Interview **205**
8.1 Grundlegende Konzepte . . . 206
8.1.1 Die Ansichtsklassen . . . 207
8.1.2 Die Modellklassen . . . 208
8.2 Darstellen von Verzeichnishierarchien . . . 210
8.2.1 View-Klassen im Designer benutzen . . . 211
8.2.2 Die Funktionalität des Dateiauswahldialogs implementieren 213

Copyright (C) Open Source Press

8.3 Das Stringlisten-Modell 218
8.4 Eigene Modelle implementieren 219
8.4.1 Ein Adressbuchmodell 219
8.4.2 Eigene Modelle beschreibbar machen 224
8.5 Daten mit Proxymodellen sortieren und filtern 228
8.5.1 Anpassungen an der Benutzerschnittstelle 229
8.6 Einträge durch Checkboxen anwählbar machen 231
8.7 Eigene Proxymodelle entwerfen 234
8.8 Drag&Drop in Modellen implementieren 238
8.9 Eigene Delegates 243
8.10 Ohne eigene Datenquelle: Das Standard-Modell 246
8.11 Elementbasierte Ansichten ohne Modellzugriff 248
8.11.1 Items 249
8.11.2 Die Listenansicht 250
8.11.3 Die Baumansicht 251
8.11.4 Die Tabellenansicht 252
8.11.5 Items klonen 253

9 Das QtSql-Modul 255

9.1 Aufbau des QtSql-Moduls 255
9.2 Den passenden Treiber wählen 256
9.3 Verbindung aufnehmen 258
9.4 Anfragen stellen 259
9.5 Transaktionen 262
9.6 Eingebettete Datenbanken 262
9.7 SQL-Modell-Klassen mit Interview verwenden 263
9.7.1 SQL-Tabellen ohne Fremdschlüssel in Tabellen- und Baumansichten darstellen 263
9.7.2 Fremdschlüssel-Relationen auflösen 264
9.7.3 Abfrageergebnisse darstellen 265
9.7.4 Editierstrategien 266
9.7.5 Fehler der Tabellenmodelle 268

Copyright © Open Source Press

10 Die Grafikbibliothek Arthur **269**
10.1 Farben . . . 269
10.1.1 Der RGB-Farbraum . . . 270
10.1.2 Weitere Farbräume . . . 271
10.1.3 Farbauswahldialog . . . 273
10.2 Zeichnen mit Qt . . . 274
10.3 Geometrische Hilfsklassen . . . 277
10.4 Zeichnen auf Widgets . . . 279
10.4.1 Flimmerfrei auf den Bildschirm . . . 281
10.5 Praktischer Umgang mit **QPainter** . . . 282
10.5.1 Tortendiagramm zeichnen . . . 283
10.5.2 Widgetgröße festlegen . . . 288
10.5.3 Die Diagramm-Applikation . . . 289
10.6 Transformationen des Koordinatensystems . . . 289
10.6.1 Transformationen in der Praxis . . . 292
10.7 **QImage** . . . 295
10.7.1 Speicherformate, Transparenz und Farbpaletten . . . 296
10.7.2 Pixel zeilenweise auslesen . . . 297
10.8 SVG-Unterstützung . . . 299
10.9 Drucken mit **QPrinter** . . . 301
10.9.1 Exkurs: Screenshots erstellen . . . 303
10.9.2 Ausdrucken einer Bilddatei . . . 304
10.9.3 PDFs generieren . . . 305
10.9.4 Die Testanwendung . . . 305
10.10 Komplexe Grafiken . . . 306
10.10.1 Clipping . . . 306
10.10.2 Painterpfade . . . 308
10.10.3 Kompositionsmodi . . . 310

11 Ein-/Ausgabeschnittstellen **317**
11.1 Die **QIODevice**-Klassenhierarchie . . . 317
11.1.1 Abgeleitete Klassen . . . 318
11.1.2 I/O-Devices öffnen . . . 319

Copyright (C) Open Source Press

11.2 Zugriff auf lokale Dateien 320
11.3 Objekte serialisieren 322
11.3.1 Serialisierungsoperatoren definieren 325
11.3.2 Serialisierte Daten in einer Datei speichern und aus ihr auslesen 326
11.4 Prozesse starten und kontrollieren 328
11.4.1 Synchroner Gebrauch von **QProcess** 328
11.4.2 Asynchroner Gebrauch von **QProcess** 331
11.5 Kommunikation im Netzwerk 333
11.5.1 Namensauflösung mit **QHostInfo** 333
11.5.2 **QTcpServer** und **QTcpSocket** nutzen 334

12 Threading mit QThread **339**
12.1 Threads verwenden 340
12.2 Threads synchronisieren 343
12.2.1 Das Consumer/Producer-Pattern 344
12.3 Thread-gebundene Datenstrukturen 347
12.4 Signale und Slots zwischen Threads nutzen 349
12.5 Eigene Event-Loops für Threads 352
12.5.1 Kommunikation über Events ohne Thread-eigene Event-Loop 354

13 XML-Behandlung mit QtXml **355**
13.1 Die SAX2-API 356
13.1.1 Funktionsweise 356
13.1.2 Default-Handler für das Lesen von RSS-Feeds reimplementieren 357
13.1.3 Exkurs: Den RSS-Reader mit GUI und Netzwerkfähigkeit ausstatten 363
13.2 Die DOM-API 369
13.2.1 XML-Dateien einlesen und verarbeiten 370
13.2.2 Elemente gezielt suchen 373
13.2.3 Den DOM-Baum manipulieren 374
13.2.4 DOM-Baum als XML ausgeben 375

Copyright © Open Source Press

14 Internationalisierung **377**

14.1 Anwendungen in andere Sprachen übersetzen 378

14.1.1 Die Anwendung vorbereiten 378

14.1.2 Übersetzungsquellen mit Linguist bearbeiten 379

14.1.3 Übersetzungen im Programm verwenden 380

14.1.4 Hinweise für die Übersetzung geben 382

14.1.5 Übersetzungskontext angeben 382

14.1.6 Zeichenketten außerhalb von Qt-Klassen internationalisieren 383

Anhang **385**

A Hilfe bei der Fehlersuche **387**

A.1 Debugging-Funktionen 387

A.1.1 Einfache Debug-Ausgaben 388

A.1.2 Fehler und Warnungen 389

A.1.3 Ausgabe der Debugging-Funktionen anpassen 390

A.2 Methoden zur Fehlerbehebung 392

A.2.1 Annahmen prüfen 392

A.2.2 Zeiger überprüfen 393

A.2.3 Häufige Linker-Fehler 394

B Tulip: Container und Algorithmen **395**

B.1 Iteratoren 396

B.1.1 STL-artige Iteratoren 397

B.1.2 Java-artige Iteratoren 398

B.2 Listen 400

B.2.1 Einfache Liste (QList) 402

B.2.2 Verkettete Liste (QLinkedList) 402

B.2.3 Vektor (QVector) 403

B.3 Stapel und Schlangen 405

B.3.1 Stapel (QStack) 405

B.3.2 Warteschlange (QQueue) 406

B.4 Assoziative Arrays 406

Copyright (C) Open Source Press

B.4.1 Zuordnungstabelle (QMap) 406
B.4.2 Mehrere gleiche Schlüssel zulassen (QMultiMap) 409
B.4.3 Hashtabellen mit QHash 410
B.4.4 Hash-basierte Mengen mit QSet 413
B.5 Algorithmen 414
B.5.1 Das foreach-Schlüsselwort 414
B.5.2 Sortieren 415
B.5.3 Suche in unsortierten Containern 416
B.5.4 Containerbereiche kopieren 417
B.5.5 Binärsuche in sortierten Containern 418
B.5.6 Vorkommen gleicher Elementen zählen 420
B.5.7 Zeiger in Listen löschen 420
B.5.8 Datenstrukturbereiche auf elementweise Gleichheit prüfen 421
B.5.9 Datenstrukturen auffüllen 422
B.5.10 Werte tauschen 422
B.5.11 Minima, Maxima und Grenzen 423
B.5.12 Betrag bestimmen 423
B.6 Qt-eigene Typdefinitionen 424
B.6.1 Ganzzahl-Typen 424
B.6.2 Fließkommazahlen 425
B.6.3 Kurzformen für gängige Typen 425

Index **427**

Copyright (C) Open Source Press