1

- *Hello Qt*
- *Pravljenje veza*
- *Korišćenje priručne dokumentacije*

Početak rada

U ovom poglavlju naučićete kako da kombinujete osnovni C++ sa funkcionalnošću koju pruža Qt kako biste napravili nekoliko malih GUI aplikacija. U ovom poglavlju izložićemo dve ključne ideje Qt-a: „signali i slotovi" i projekat. U poglavlju 2, temu ćemo obraditi detaljnije, a u poglavlju 3 ćemo početi sa pravljenjem stvarne aplikacije.

Hello Qt

Evo vrlo jednostavnog Qt programa:

```
 1  #include <qapplication.h>
 2  #include <qlabel.h>

 3  int main(int argc, char *argv[])
 4  {
 5       QApplication app(argc, argv);
 6       QLabel *label = new QLabel("Hello Qt!",0);
 7       app.setMainWidget(Label);
 8       label->show();
 9       return app.exec();
10  }
```

Kôd ćemo prvo proučiti red po red, potom ćemo vas naučiti kako da ga kompajlirate i izvršite.

Redovi 1 i 2 sadrže definicije klasa `Qapplication` i `QLabel`.

U redu 5 stvara se objekat klase `QApplication` koji će upravljati resursima cele aplikacije. Konstruktor klase `QApplication` zahteva argumente `argc` i `argv` zato što Qt podržava nekoliko sopstvenih argumenta komandne linije.

U redu 6 stvara se objekat klase `QLabel`, softverska naprava koja će prikazati tekst „Hello Qt!". U terminologiji Qt-a, *softverska naprava* je vizuelni element na korisničkom interfejsu. Softverske naprave mogu da sadrže druge softverske naprave, na primer, prozor aplikacije obično je softverska naprava koja sadrži softverske

naprave `QMenuBar`, `QToolBar`, `QStatusBar` i još neke softverske naprave. Argument 0 u konstruktoru klase QLabel (nula pokazivač) govori da je softverska naprava u sopstvenom prozoru, a ne unutar drugog prozora.

U redu 7 pravi se labela glavne aplikacijske softverske naprave. Kada korisnik zatvori glavnu softversku napravu (na primer, pritiskom na X u naslovnoj liniji prozora), rad programa će se završiti. Bez glavne softverske naprave, program bi bio aktivan u pozadini čak i kada korisnik zatvori prozor.

U redu 8 labela će postati vidljiva. Prilikom izrade, softverske naprave su uvek nevidljive, tako da ih možemo podesiti pre prikazivanja, i na taj način izbeći treperenje.

U redu 9 kontrola aplikacije prebacuje se na Qt. U ovom trenutku, program ulazi u neku vrstu režima pripravnosti, u kome čeka akcije korisnika preko miša ili tastature.

Akcije korisnika stvaraju *događaje* na koje program može da odgovori, obično izvršavanjem jedne ili više funkcija. Po ovome, GUI aplikacije se drastično razlikuju od konvencionalnih komandnih programa, koji obično obrađuju ulazne parametre, stvaraju rezultate i prekidaju izvršavanje bez ljudske intervencije.

Slika 1.1. Hello na Windowsu XP

Vreme je da program testirate na vašem računaru. Prvo, morate da instalirate Qt 3.2 (ili noviju verziju Qt-a 3), proces koji je objašnjen u dodatku A. Od ovog trenutka, pretpostavljaćemo da ste ispravno instalirali kopiju Qt-a 3.2 i da je Qt-ov direktorijum `bin` smešten u promenljivoj okruženja `PATH`. (U Windowsu, ovo će automatski uraditi instalacioni program Qt-a, tako da ne morate da brinete o tome.)

Biće vam neophodan izvorni kôd programa Hello koji se nalazi u datoteci `hello.cpp` smeštenoj u direktorijum `hello`. Datoteku `hello.cpp` možete sami popuniti ili je kopirati sa CD-a koji ste dobili uz ovu knjigu, a nalazi se u direktorijumu `\examples\chap01\hello\hello.cpp`.

Iz komandne linije, pređite u direktorijum `hello` i potom napišite

```
qmake –project
```

da biste napravili datoteku projekta nezavisnu od platforme (`hello.pro`) i potom napišite

```
qmake hello.pro
```

da biste od datoteke projekta napravili datoteku za izgradnju za specifičnu platformu. Pokrenite naredbu `make` da biste izgradili program, a program pokrenite pisanjem naredbe `hello` na Windowsu, `./hello` na Unixu i `open hello.app` na Mac OS X. Ukoliko koristite Microsoftov Visual C++, umesto naredbe `make` morate da

izvršite naredbu `nmake`. Ili, možete napraviti datoteku projekta Visual Studija od datoteke `hello.pro` ukoliko napišete

```
qmake –tp vc hello.pro
```

i da potom program izgradite u Visual Studiju.

Slika 1.2. Labela sa osnovnim formatiranjem u HTML-u

Sad se možemo malo zabaviti: osvežićemo labelu korišćenjem jednostavnog HTML stila za formatiranje. Ovo možemo uraditi zamenom koda

```
QLabel *label = new QLabel("Hello Qt!",0);
```

kodom

```
QLabel *label = new QLabel("<h2><i>Hello</i> "
                           "<font color+red>Qt!</font></h2>",0);
```

i ponovo izgraditi aplikaciju.

Pravljenje veza

Primerom koji sledi ilustrovaćemo kako da odgovorite na akcije korisnika. Aplikacija se sastoji od dugmeta na koje korisnik može pritisnuti mišem kako bi izašao iz aplikacije. Izvorni kôd je vrlo sličan programu Hello, osim što kao glavnu softversku napravu koristimo klasu `QPushButton` umesto klase `QLabel`, i što povezujemo akciju korisnika (pritisak dugmeta) sa delom koda.

Izvorni kôd ove aplikacije nalazi se na CD-u u datoteci `\examples\chap01\quit\quit.cpp`.

Slika 1.3. Aplikacija Quit

```
 1  #include <qapplication.h>
 2  #include <qpushbutton.h>

 3  int main(int argc, char *argv[])
 4  {
 5      QApplication app(argc, argv);
 6      QPushButton *button = new QPushButton("Quit",0);
 7      QObject::connect (button, SIGNAL(clicked()),
 8                        &app, SLOT(quit()));
 9      app.setMainWidget(button);
10      button->show();
```

```
11      return app.exec();
12  }
```

Softverska naprava Qt-a *emituje signale* kako bi utvrdila da li je korisnik izvršio akciju ili da li se promenilo stanje*. Na primer, klasa `QPushButton` emituje signal `clicked()` kada korisnik izabere dugme. Signal može biti povezan sa funkcijom (koja se u tom kontekstu naziva *slot*), tako da se prilikom emitovanja signala slot automatski izvršava. U našem primeru, signal dugmeta `clicked()` povezujemo sa slotom `quit()` objekta `QApplication`. Makroi `SIGNAL()` i `SLOT()` deo su sintakse; biće detaljnije objašnjeni u narednom poglavlju.

Sada ćemo napraviti aplikaciju. Pretpostavljamo da ste napravili direktorijum `quit` u kome je `quit.cpp`. Izvršite `qmake` u direktorijumu `quit` da biste generisali projektnu datoteku, zatim je izvršite ponovo da biste generisali datoteku za izgradnju:

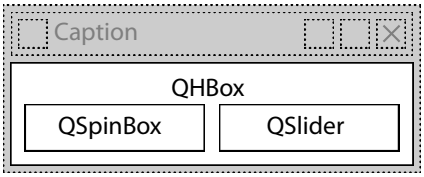
```
qmake -project
qmake quit.pro
```

Sada podignite i izvršite aplikaciju. Ukoliko pritisnete `Quit`, ili pritisnete razmaknicu (što je isto kao da ste pritisnuli dugme), prekinućete izvršavanje aplikacije.

U narednom primeru pokazaćemo na koji način da koristite signale i slotove kako biste sinhronizovali dve softverske naprave. Aplikacija korisnika pita za godine starosti, koje korisnik može da upiše upotrebom brojača ili klizača.

Slika 1.4. Aplikacija Age

Aplikacija se sastoji od tri softverske naprave: `QSpinBox`, `QSlider` i `QHBox` (maketa horizontalnog okvira). Glavna softverska naprava aplikacije je `QHBox`. Softverske naprave `QSpinBox` i `QSlider` prikazuju se unutar softverske naprave `QHBox`, oni su *deca* softverske naprave `QHBox`.



Slika 1.5. Softverske naprave aplikacije Age

```
1  #include <qapplication.h>
2  #include <qhbox.h>
3  #include <qslider.h>
4  #include <qspinbox.h>
```

* Qt-ovi signali nisu u vezi sa Unixovim signalima. U ovoj knjizi, razmatraćemo samo Qt-ove signale.

```
 5  int main(int argc, char *argv[])
 6  {
 7      QApplication app(argc, argv);

 8      QHBox *hbox = new QHBox(0);
 9      hbox->setCaption("Enter Your Age");
10      hbox->setMargin(6);
11      hbox->setSpacing(6);

12      QSpinBox *spinBox = new QSpinBox(hbox);
13      QSlider *slider = new QSlider(Qt::Horizontal, hbox);
14      spinBox ->setRange(0,130);
15      slider->setRange(0,130);

16      QObject::connect(spinBox, SIGNAL(valueChanged(int)),
17                       slider, SLOT(setValue(int)));
18      QObject::connect(slider, SIGNAL(valueChanged(int)),
19                       spinBox, SLOT(setValue(int)));
20      spinBox->setValue(35);

21      app.setMainWidget(hbox);
22      hbox->show();

23      return app.exec();
24  }
```

U redovima od 8 do 11 podešava se softverska naprava `QHBox`*. Pozvaćemo funkciju `setCaption()` da bismo postavili tekst koji će biti prikazan u naslovnoj liniji prozora. Potom ćemo postaviti prazan prostor (6 piksela) oko i između dece softverskih naprava.

U redovima 12 i 13 prave se softverske naprave `QSpinBox` i `QSlider` kojima je roditelj softverska naprava `QHBox`.

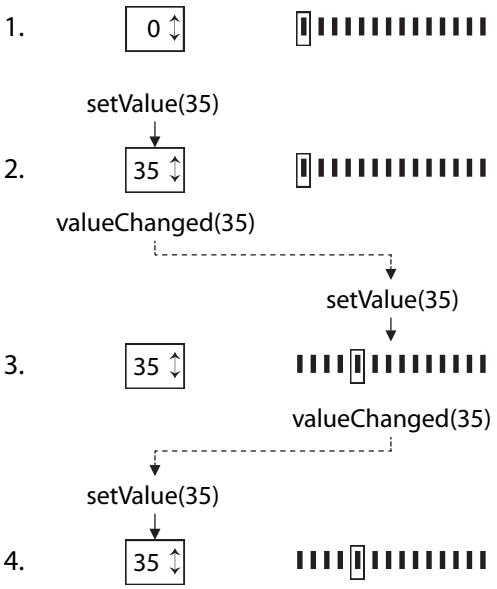
Iako nismo eksplicitno podesili poziciju ili veličinu bilo koje softverske naprave, softverske naprave `QSpingBox` i `QSlider` pojaviće se lepo raspoređene jedna pored druge unutar softverske naprave `QHBox`. Ovako su raspoređene zato što softverska naprava `QHBox` automatski dodeljuje razumne pozicije i veličine svojoj deci na osnovu njihovih potreba. Qt pruža mnoge klase kao što je `QHBox` kako bi nas oslobodio posla fiksnog kodiranja pozicije na ekranu unutar aplikacije.

U redovima 14 i 15 određuju se važeći opsezi brojača i klizača. (Sa sigurnošću možemo pretpostaviti da korisnik može biti star najviše 130 godina.) Dva poziva funkcije `connect()` prikazane u redovima 16 i 19 obezbeđuju da su brojač i klizač sinhronizovani tako da uvek prikazuju iste vrednosti. Svaki put kada se promeni vrednost jedne softverske naprave, emituje se signal `valueChagned(int)`, a slot `setValue(int)` druge softverske naprave poziva se sa novom vrednošću.

U redu 20, vrednost brojača postavlja se na 35. Kada se ovo dogodi, softverska naprava `QSpinBox` emituje signal `valueChanged(int)` sa argumentom tipa `int` čija je vrednost 35. Ovaj argument prosleđuje se slotu `setValue(int)` softverske

* Ukoliko prilikom kompajliranja od konstruktora QHBox dobijete grešku, znači da koristite stariju verziju Qt-a. Potrudite se da koristite Qt 3.2.0 ili noviju verziju Qt-a 3.

naprave `QSlider`, koji vrednost klizača postavlja na 35. Klizač potom emituje signal `valueChanged(int)`, jer se njegova vrednost promenila, okidajući slot brojača `setValue(int)`. Međutim, u ovom trenutku, slot `setValue(int)` ne emituje nikakav signal, jer je vrednost klizača već postavljena na 35. Na ovaj način onemogućena je beskonačna rekurzija. Na slici 1.6 dat je pregled ove situacije.



Slika 1.6. Promenom jedne vrednosti promeniće se obe.

U redu 22 prikazujemo softversku napravu `QHBox` i njegovo dvoje dece softverskih naprava.

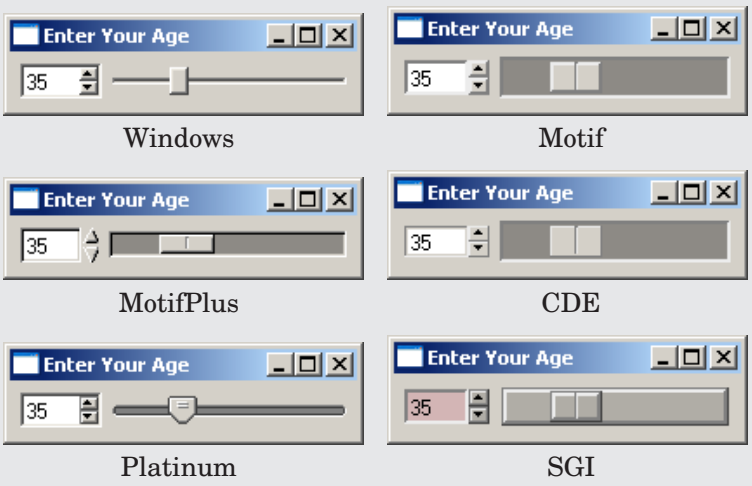
Qt-ov pristup pravljenju korisničkih interfejsa lako se razume i veoma je fleksibilan. Najčešći šablon koji koriste Qt programeri je da naprave primerke softverskih naprava i da, po potrebi, podese njihova svojstva. Programeri dodaju softverske naprave postojećim uređenjima GUI-ja, koje se potom automatski staraju o veličini i poziciji. Ponašanjem korisničkog interfejsa upravlja se povezivanjem softverskih naprava pomoću Qt-ovog mehanizma signala i slotova.

Korišćenje priručne dokumentacije

Qt-ova priručna dokumentacija je neophodan alat svakog Qt razvijaoca, budući da su njome obuhvaćene sve Qt-ove klase i funkcije. (Qt 3.2 sadrži preko 400 javnih klasa i preko 6 000 funkcija.) U ovoj knjizi koristićemo mnoge klase i funkcije Qt-a, ali ih nećemo sve pomenuti, niti ćemo pružiti sve detalje o klasama koje pomenemo. Da biste izvukli najveću korist od Qt-a, treba da se upoznate sa Qt-ovom referentnom dokumentacijom.

Stilovi softverskih naprava

Snimci ekrana koje smo do sada videli načinjeni su na Windowsu XP, ali aplikacije pisane na Qt-u izgledaju srodno na svakoj podržanoj platformi. Qt ovo postiže emuliranjem izgleda platforme, a ne „navlačenjem" određene platforme ili kompleta softverskih naprava iz platformskog kompleta alatki.

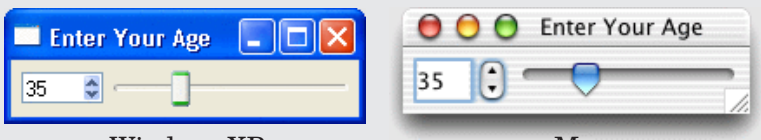


Slika 1.7. Stilovi koji su svuda dostupni.

Korisnici Qt aplikacija mogu da izbegnu podrazumevani stil korišćenjem naredbe komandne linije `–style`. Na primer, da biste na Unixu pokrenuli aplikaciju Age sa stilom Platinum, jednostavno upišite

```
./age –style=Platinum.
```

na komandnoj liniji.

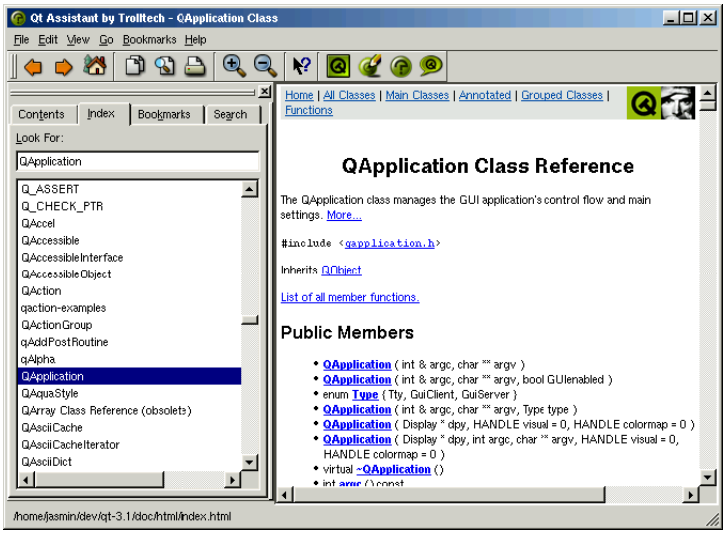


Slika 1.8. Stilovi specifični za platformu

Za razliku od ostalih stilova, stilovi za Windows XP i Mac OS X dostupni su samo na maternjim platformama, jer se oslanjaju na platformski mehanizam tema.

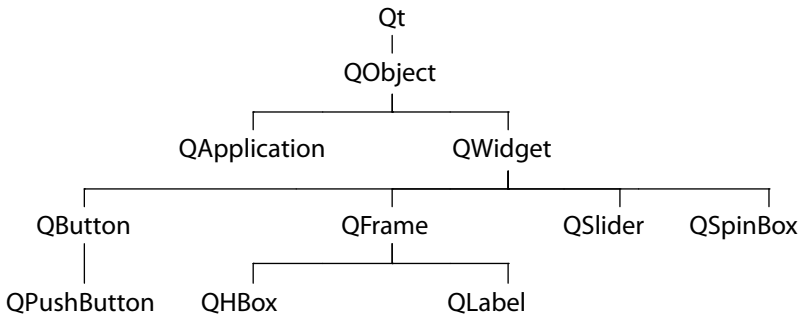
Dokumentacija je na raspolaganju u HTML formatu u Qt-ovom direktorijumu `doc\html` i možete je čitati u bilo kom pretraživaču. Možete koristiti aplikaciju *Qt Assistant*, Qt-ov pretraživač za pomoć, čija moćna svojstva pretrage i indeksiranja omogućuju bržu i lakšu upotrebu, za razliku od veb pretraživača. Da biste pokrenuli aplikaciju Qt Assistant, u Windowsu iz menija Start izaberite stavke

Qt 3.2.x | Qt Assistant, na Unixu u komandnoj liniji upišite `assistant` ili na Mac OS X Finderu dvaput pritisnite mišem na stavku `assistant`.



Slika 1.9. Qt-ova dokumentacija u programu *Qt Assistant*

Veze u delu „API Reference" na početnoj stranici omogućuju različite načine kretanja kroz klase Qt-a. Na stranici „All Classes" navedene su sve klase Qt-ovog API interfejsa. Stranica „Main Classes" sadrži listu najčešće korišćenih klasa u Qt-u. Za vežbu, mogli biste da potražite klase i funkcije koje smo koristili u ovom poglavlju. Zapamtite da su nasleđene funkcije dokumentovane u osnovnoj klasi, na primer, klasa `QPushButton` nema svoju funkciju `show()`, ali je nasleđuje od svog pretka, klase `QWidget`. Na slici 1.10 prikazane su međusobne veze klasa koje smo do sada prikazali.



Slika 1.10. Stablo nasleđivanja Qt-ovih klasa koje smo do sada prikazali.

Priručna dokumentacija tekuće verzije Qt-a i nekih ranijih verzija dostupna je na adresi `http://doc.trolltech.com/`. Na ovoj lokaciji nalaze se odabrani članci iz biltena *Qt Quarterly*, koji se šalje svim Qt programerima sa komercijalnim licencama.