Qt keretrendszer

Qt nyelvi könyvtárai,

a Qt programok fordítása,

és bevezetés az eseménykezelésbe

A Qt egy platformfüggetlen alkalmazás-fejlesztési keretrendszer, amellyel nemcsak asztali, de akár mobil vagy beágyazott alkalmazások is fejleszthetők.

- Lehetőséget ad az eseményvezérelt grafikus felhasználói felületű alkalmazások fejlesztésére, adatbázisok kezelésére, multimédiás szoftverek készítéséhez, 3D grafika alkalmazására, hálózati és webes kommunikáció megvalósítására.
- Rendelkezik nyílt forráskódú (LGPL) és kereskedelmi verzióval is.
- Fejlesztő nyelve elsősorban a C++, de más nyelvekre (pl. Python) is elérhető, sőt rendelkezik saját leíró nyelvvel (Qt Quick) is.
- A Qt 5.x keretrendszer, valamint QtCreator fejlesztőkörnyezet (benne a QtDesigner) elérhető a https://www.qt.io/ oldalról.
  - Linux esetén: **`apt-get install qt-sdk`**

Segédanyag

- Qt : https://doc.qt.io
- Jasmin Blanchette, Mark Summerfield: C++ GUI programming with Qt4, Prentice Hall, 2006. ISBN 0-13-187249-4
- Lee Zhi Eng: Hands-On GUI Programming with C++ and Qt5 – Build stunning cross-platform applications and widgets with the most powerful GUI framework,
  Packt Publishing, 2016. ISBN 978-1-78646-712-6
- Lee Zhi Eng: Qt5 C++ GUI Programming Cookbool – Design and build a functional, appealing, and user-friendly graphical user interface,
  Packt Publishing, 2018. ISBN 978-1-78328-027-8
- Guillaume Lazar, Robin Penea: Mastering Qt5 – Master application development by writing succinct, robust, and reusable code with Qt5
  Packt Publishing, 2018. ISBN 978-1-78839-782-7

Fejlesztés nyelve és környezete

- A fejlesztés C++/Qt nyelven.
  - Elérhető C++ a teljes utasításkészlete, nyelvi könyvtárai.
  - A standard C++ nyelv kiterjesztését Qt-s makrók biztosítják, amelyeket a *Meta Object Compiler* (*MOC*) fordít le ISO C++ kódra.
  - Speciális osztályokat biztosít, amelyek között számos C++ nyelven megszokott osztálynak Qt-s megfelelője (QString, QQueue, stb.) is megtalálható.
- Az alapértelmezett fejlesztőeszköz a *QtCreator*, de más környezetekben is lehetőség van a Qt fejlesztésre (pl. *Code::Blocks*, *Visual Studio*).
- Külön tervezőprogram (*QtDesigner*) áll rendelkezésre a grafikus felület létrehozásához, amelynek kimenete a felület XML nyelvű leírása.

”Hello Qt” alkalmazás


```
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[]) {
    QApplication app(argc, argv); // alkalmazás példányosítása
    QLabel myLabel("Hello Qt!");  // címke a felirattal
    myLabel.show();               // címke megjelenítése

    return app.exec();            // alkalmazás futtatása
}
```

létrehozza az események kezelését végző környezetet

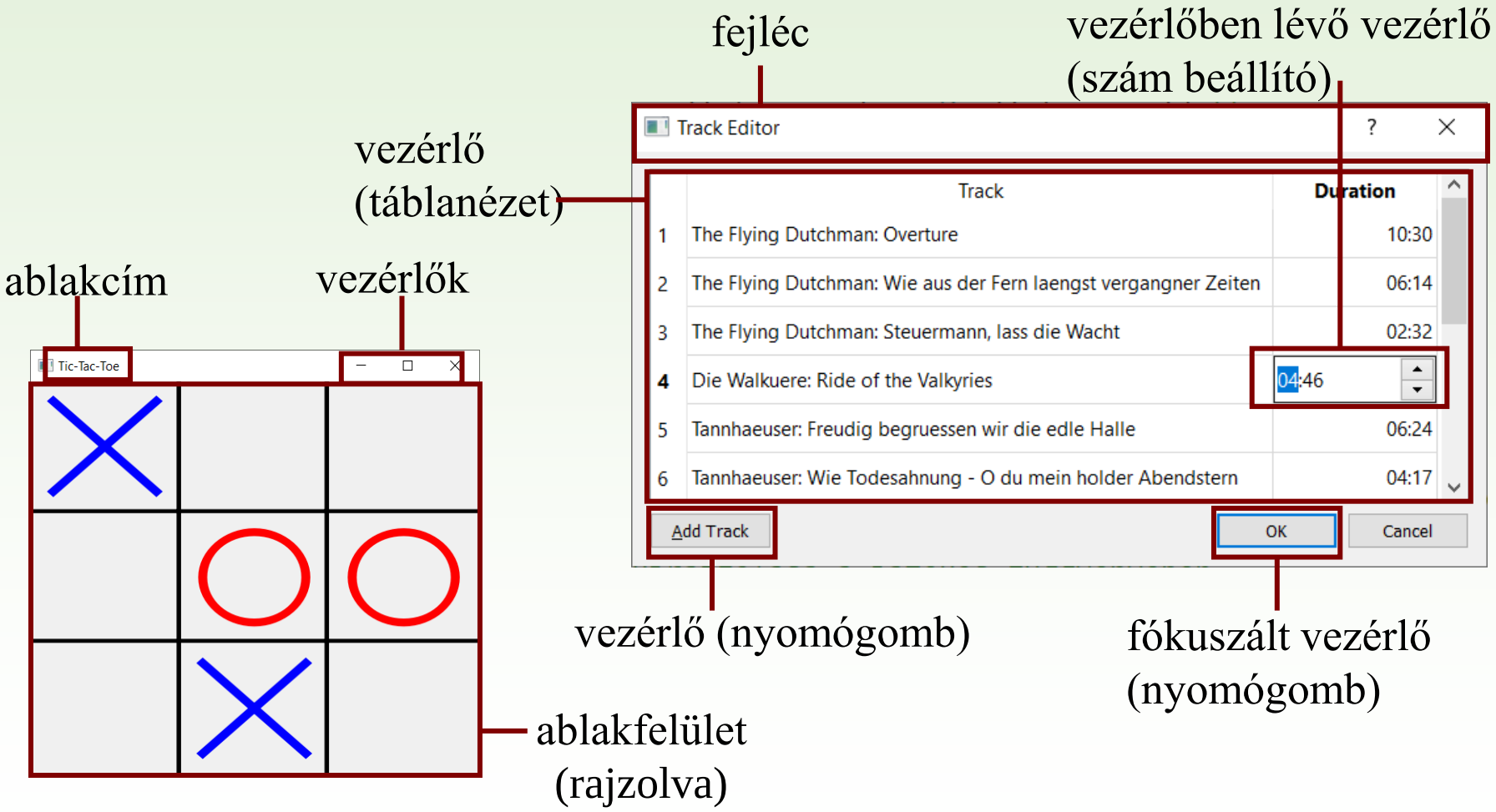
megjeleníti egy önálló ablakban a címkét

elindítja az alkalmazást

Eseményvezérelt grafikus felületű alkalmazás

- Eseményvezérelt grafikus felületű alkalmazásnak nevezzük azt a programot, amely 2D-s interaktív felhasználói felületen (*GUI*, *Graphical User Interface*) keresztül kommunikál a felhasználóval, de más forrásból származó (pl.: egy időzítő, vagy egy másik alkalmazás) jelzésre is reagál.
  - A működését a felhasználói interakcióra történő várakozás jellemzi, és a programfutásba történő beavatkozáshoz egy konzolablaknál lényegesen változatosabb interakciós lehetőséget biztosít.
  - A felület grafikus megjelenítéssel ellátott ún. vezérlőkből (*control/widget*) áll (pl.: nyomógombok, listák, menük, stb.).
  - Egy grafikus vezérlő megjelenhet önállóan – ablakként (*form/window*) –, vagy egy másik vezérlő részeként. Több ablak esetén mindig van egy aktív ablak, és egy aktív vezérlő (ezen van a fókusz).

Példák GUI alkalmazásokra



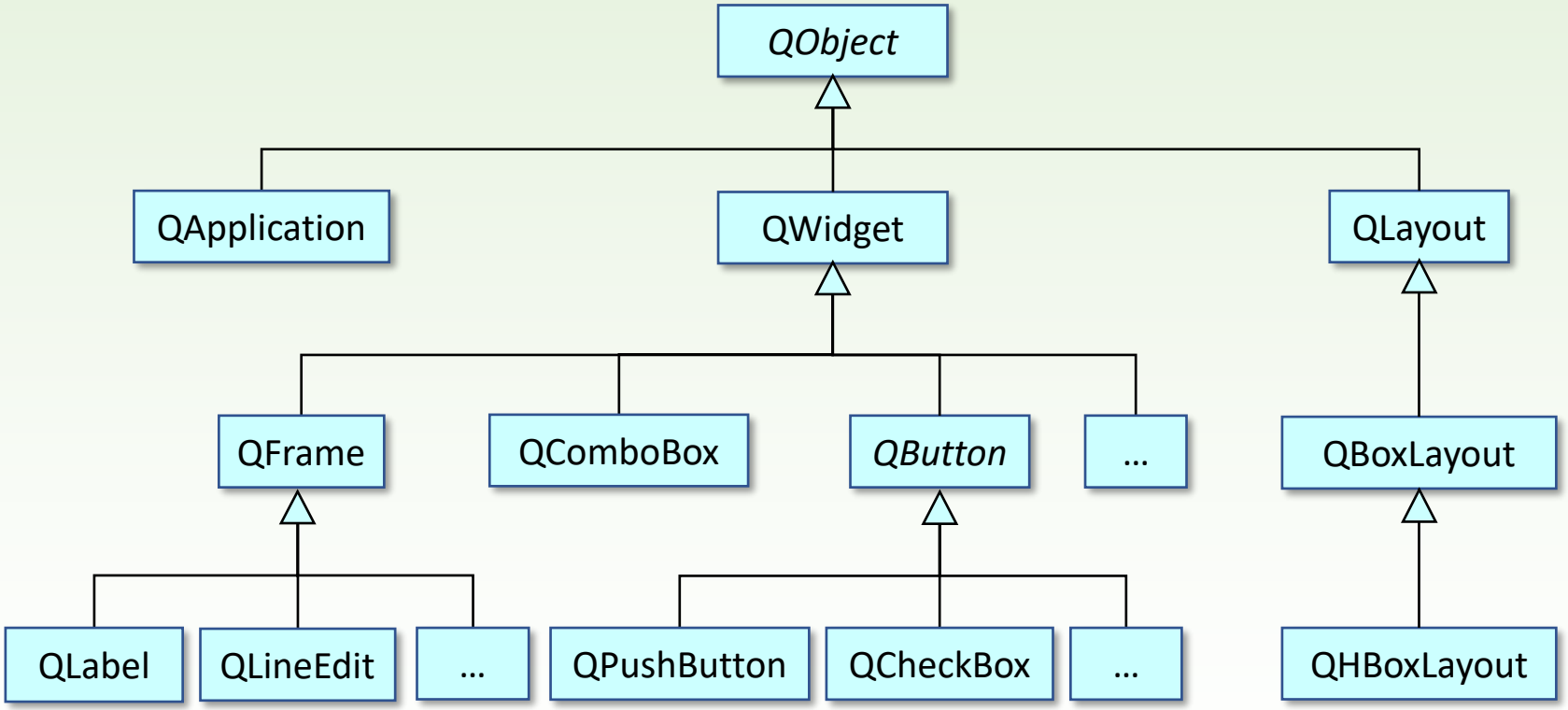
Objektumorientált grafikus felület

- Minden vezérlő egy objektum, amelyet az osztálya jellemez.
- Mivel a különféle vezérlők sok hasonló tulajdonsággal bírnak, így osztályaik öröklődési hierarchiába szervezhetők.
  - A Qt keretrendszer nyelvi könyvtárában találjuk az általános vezérlő osztályt, amelyből nevezetes vezérlők osztályai származnak.
  - Ha egyedi vezérlőre van szükségünk, akkor annak osztályát nekünk kell származtatással definiálni.
- Egy adott alkalmazás vezérlői elhelyezkedési hierarchiába rendeződnek aszerint, hogy egy vezérlő önállóan, azaz *ablak*ként jelenik meg, vagy egy másik vezérlő részeként.

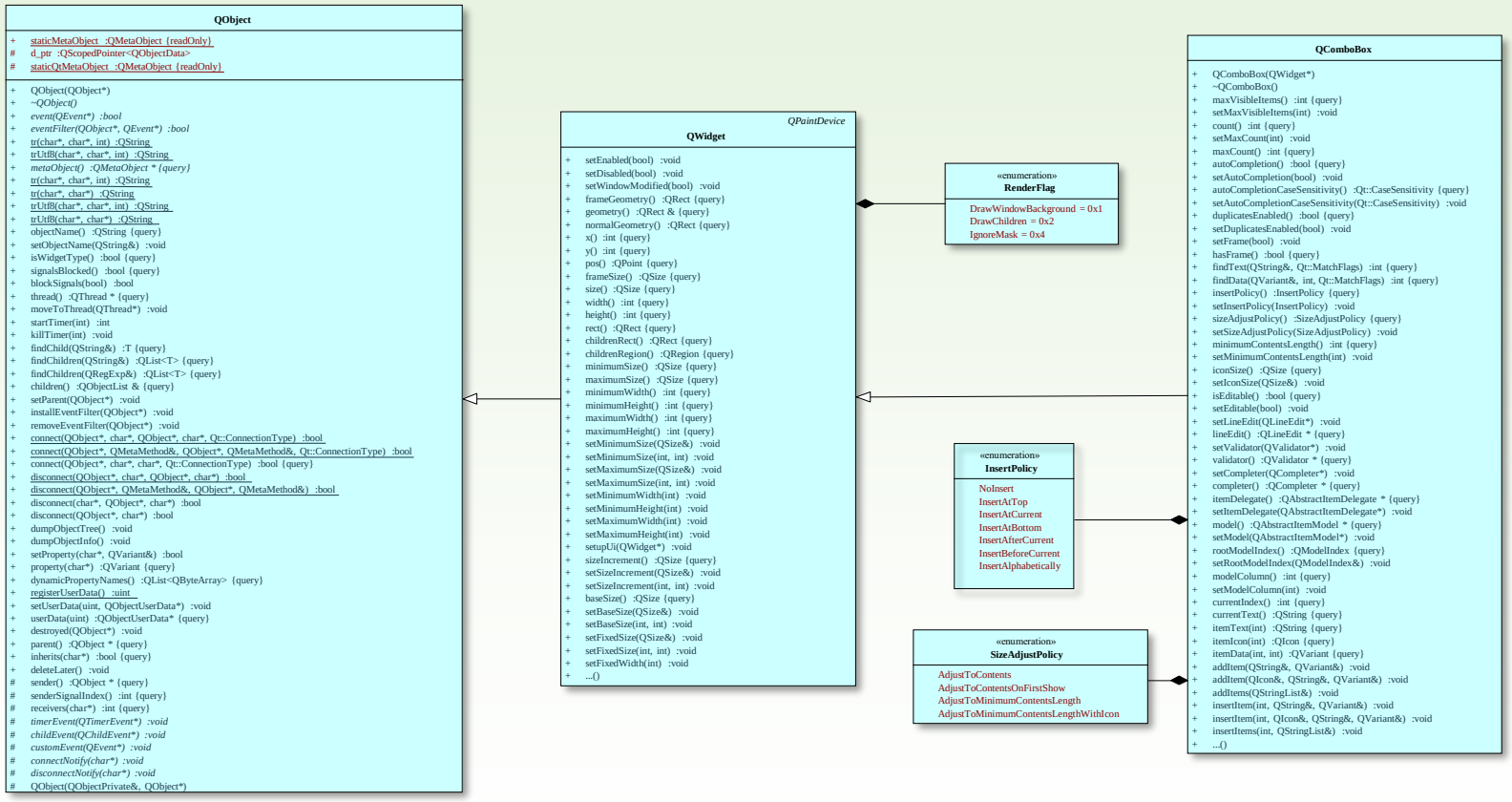
Qt osztálykönyvtár

- A Qt nyelvi könyvtár osztályai mind a **`QObject`** ősosztály leszármazottjai. A **`QObject`** példányok nem másolhatók, ezért jórész mutatók és referenciák segítségével kezeljük őket
- Számos osztállyal rendelkezik:
  - grafikus megjelenéssel rendelkező nevezetes vezérlők osztályai, amelyek az általános **`QWidget`** osztály leszármazpottai
  - adatszerkezetek osztályai (**`QVector`**, **`QStack`**, **`QLinkedList`**, …)
  - fájlok és fájlrendszerek kezelésének osztályai (**`QFile`**, **`QTextStream`**, **`QDir`**, …)
  - adatbázis kezelés osztályai (**`QSqlDataBase`**, **`QSqlQuery`**, **`QSqlResult`**, **`QSqlQueryModel`**, **`QSqlTableModel`**)
  - párhuzamosság és aszinkron végrehajtást támogató osztályok (**`QThread`**, **`QSemaphore`**, **`QFuture`**, …)

Vezérlők osztályhierarchiája



Vezérlők osztályai



Modulok

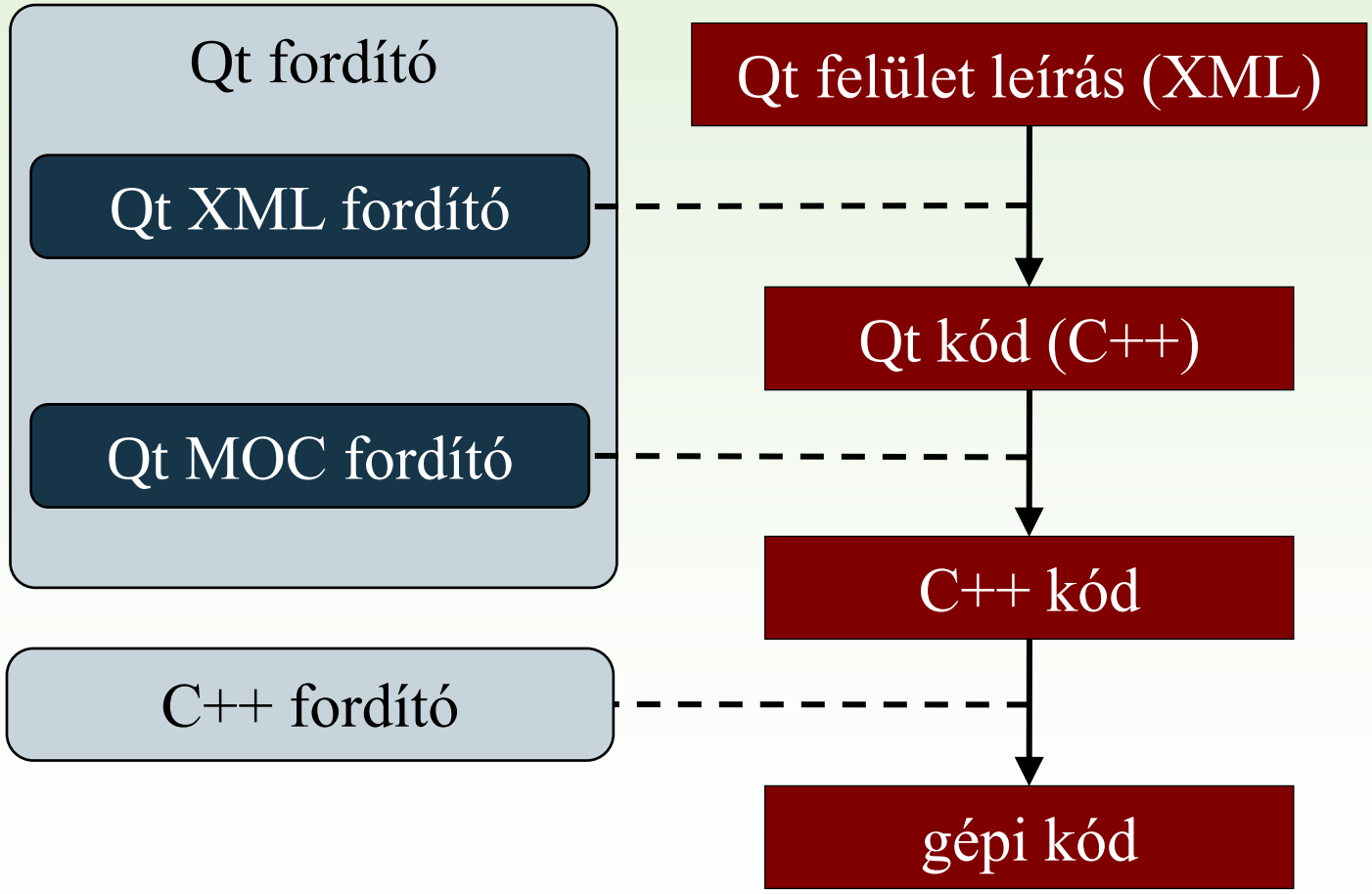
- A Qt keretrendszer felépítése modularizált. A legfontosabb modulok:
  - központi modul (**`QtCore`**)
  - grafikus felület (**`QtGui`**), grafikus vezérlők (**`QtWidgets`**)
  - adatbázis-kezelés (**`QtSQL`**)
  - hálózat-kezelés (**`QtNetwork`**)
- Egy modul tartalmát egyszerre (pl.: **`#include <QtGui>`**) is, de akár osztályonként is (pl.: **`#include <QLabel>`**) betölthetjük az aktuális fájlba.

Fordítás

- A fordítás projektszinten történik, az ehhez szükséges információk a *projektfájlban* (**`.pro`**) tárolódnak, amely tartalmazza
  - a felhasznált modulokat, kapcsolókat
    pl.: **`QT += core gui widgets sql network`**
  - a forrás- (.cpp) és fejállományok (.h), formok (ui) és erőforrások (.png, .txt, …) listáját
  - sablon leírást (app)
- A fordítás a QtCreatorban egy kattintással, de akár közvetlenül (parancssorban) is elvégezhető:

```
qmake -project # könyvtár nevével azonos projektfájl létrehozása
qmake          # fordítófájl (makefile) előállítása a projektfájlból
make           # projekt fájlnak megfelelő fordítás és
               # szerkesztés végrehajtása
```

Fordítás lépései



Vezérlés

- Egy eseményvezérelt alkalmazás vezérlését egy ún. alkalmazás objektum (a Qt-ben a **`QApplication`** osztály példánya) látja el.
  - Beállítja az alkalmazás szintű tulajdonságokat (megjelenés, elérési útvonal).
  - Felügyeli a program futása során létrejött vezérlő objektumokat, (pl.: nyilvántartja azok elhelyezkedési hierarchiáját).
  - A programfutás során bekövetkező különféle történéseket, akciókat (pl. billentyűzet-, vagy egérhasználat, egy objektum üzenete, stb.) eseményként (speciális objektum) értelmezi.
  - Gondoskodik arról, hogy az események eljussanak azokhoz az objektumokhoz (vezérlőkhöz), amelyek megfelelő tevékenységek végrehajtásával reagálhatnak az eseményre.

Akció – Esemény – Kezelés

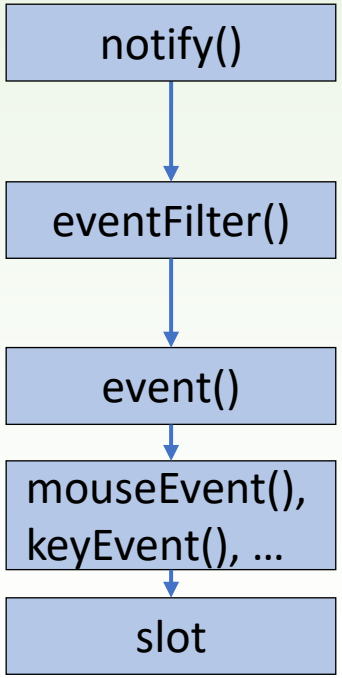
- Az akció az adott alkalmazás aktuális tevékenységétől függetlenül bekövetkező történés, amelyet kezdeményezhet a felhasználó (billentyűzet, egér, érintőképernyő, stb.), az alkalmazás egyik objektuma (pl. időzítő tikkelése), vagy egy másik alkalmazás.
- Az esemény (*event*) az alkalmazásban objektumként megjelenő akció.
  - Egy billentyű leütés akciójából a lenyomás és a felengedés eseménye lesz.
  - Az egér jobb fülével történő kattintás akcióból egy „egérgomb-lenyomás” és egy „egérgomb-felengedés” esemény születik.
- Egy eseményre az alkalmazás valamelyik vezérlő objektuma reagálhat úgy, hogy végrehajt valamilyen tevékenységet.
  - Ugyanazt az esemény akár több vezérlő objektum is megkapja, és többen is reagálhatnak rá.
  - Ugyanazt a reakciót több különböző esemény is kiválthatja (pl. egy fókuszban levő nyomógomb feletti egér-kattintás, vagy az <enter> leütése ugyanazon gomb megnyomását okozza).

Eseménykezelés formái

- Eseményvizsgáló metódusok: Egy vezérlő olyan metódusai, amelyeket az alkalmazás hív meg úgy, hogy átadja az adott eseményt. Ez a technika úgy tehető rugalmassá, ha
  - Származtatás: egy létező vezérlő osztályából származtatunk újat, és abban felüldefiniáljuk az eseményvizsgáló metódusokat.
    pl.: a **`keyPressEvent()`** metódust lehet (ha kell) ilyen módon felülírni.
  - Függőség befecskendezés: átadunk a vezérlőnek egy olyan objektumot, amelyiknek egy adott metódusa kezeli eseményt.
    pl.: az **`installEventFilter()`** metódussal rendelhetünk futási időben egy ún. *monitoring* objektumot, amelynek **`eventFilter()`** metódusa kezeli a vezérlő eseményeit.
- Signal-Slot kapcsolat: egy vezérlő ún. eseményvizsgáló metódusai az esemény hatására egy szignált váltanak ki (*emit*), amelyre futási időben iratkozhat fel a szignált kiváltó esemény lekezelő metódusa (vagy metódusai).

Eseménykezelés szintjei

- Egy esemény feldolgozása az alkalmazás-objektum irányítása mellett elosztott módon történik a megfelelő sorrendben hívott eseményvizsgáló metódusok által.
  - Először az alkalmazás objektum **`notify()`** metódusa (amely felüldefiniálható) fut le. Ez juttatja el az eseményt a megfelelő (fókuszban levő vagy egér mutató alatti) vezérlőhöz.
  - A vezérlőkhöz opcionálisan hozzárendelt figyelő (*monitoring*) objektum (ugyanaz tartozhat több vezérlőhöz is) végezhet eseményfeldolgozást az **`eventFilter()`** metódusával.
  - Egy vezérlő a hozzá eljutó eseményt az **`event()`** metódusával (amely felüldefiniálható) dolgozza fel, és az esemény fajtájától függően lefut a **`mouseXxxxEvent()`**, vagy a **`keyPressEvent()`**, vagy **`paintEvent()`** eseménykezelő is.
  - Ha ezek valamelyike szignált is kivált, akkor a szignálhoz rendelt tevékenységek (slot) végrehajtására is sor kerül.



Események és szignálok

- Az esemény a **`QEvent`** osztályból származtatott objektum.
  - A több, mint száz különféle esemény típust *enum* értékek azonosítják, amelyeket a **`QEvent::type()`** add meg.
    - Pl: **`QEvent::KeyPress`**, **`QEvent::MouseButtonPress`**
  - Egy eseménynek lehetnek paraméterei (*arguments*).
    - Pl. egérfül lenyomásakor: melyik fület nyomták le, mi az egérmutató pozíciója.
- Egy szignálra úgy gondoljunk, mint egy törzs nélküli void-os függvényre, amelyet épp emiatt nem „meghívunk”, hanem „kiváltunk”, és ha van hozzárendelve eseménykezelő tevékenység, akkor majd az hajtódik végre, különben nem történik semmi (a szignál lekezeletlen marad).
- Fejlesztés során
  - új vezérlő elkészítésekor annak eseményeire koncentrálunk
  - egy vezérlő felhasználásakor annak szignáljaival foglalkozunk

Szignálok és slot-ok

- Egy szignálnak lehetnek aktuális paraméterei, amelyet a szignálhoz társított eseménykezelő formális paraméterváltozói vesznek át. Ezeknek sorrendben, típusban illeszkedni kell, de darabszámban nem.
- Egy szignálhoz több tevékenység is társítható, több különböző szignálhoz ugyanaz a tevékenység is, sőt egy szignálhoz egy másik szignál is.
- A társítást végző **`connect`** bármelyik **`QObject`**-ből leszármazott osztály metódusaként jelen van, de statikus metódus hivatkozással is elérhető. Négy paramétere van:
  - a szignált kiváltó küldő objektum (*sender*)
  - a szignált (**`SIGNAL`**)
  - a fogadó objektum (*receiver*)
  - a fogadó metódusaként definiált eseménykezelő (**`SLOT`**).
- Egy szignálhoz rendelt eseménykezelő metódusban mindig lekérdezhető a szignált kiváltó vezérlő objektum: a szignál küldője (*sender*).

”Quit” alkalmazás

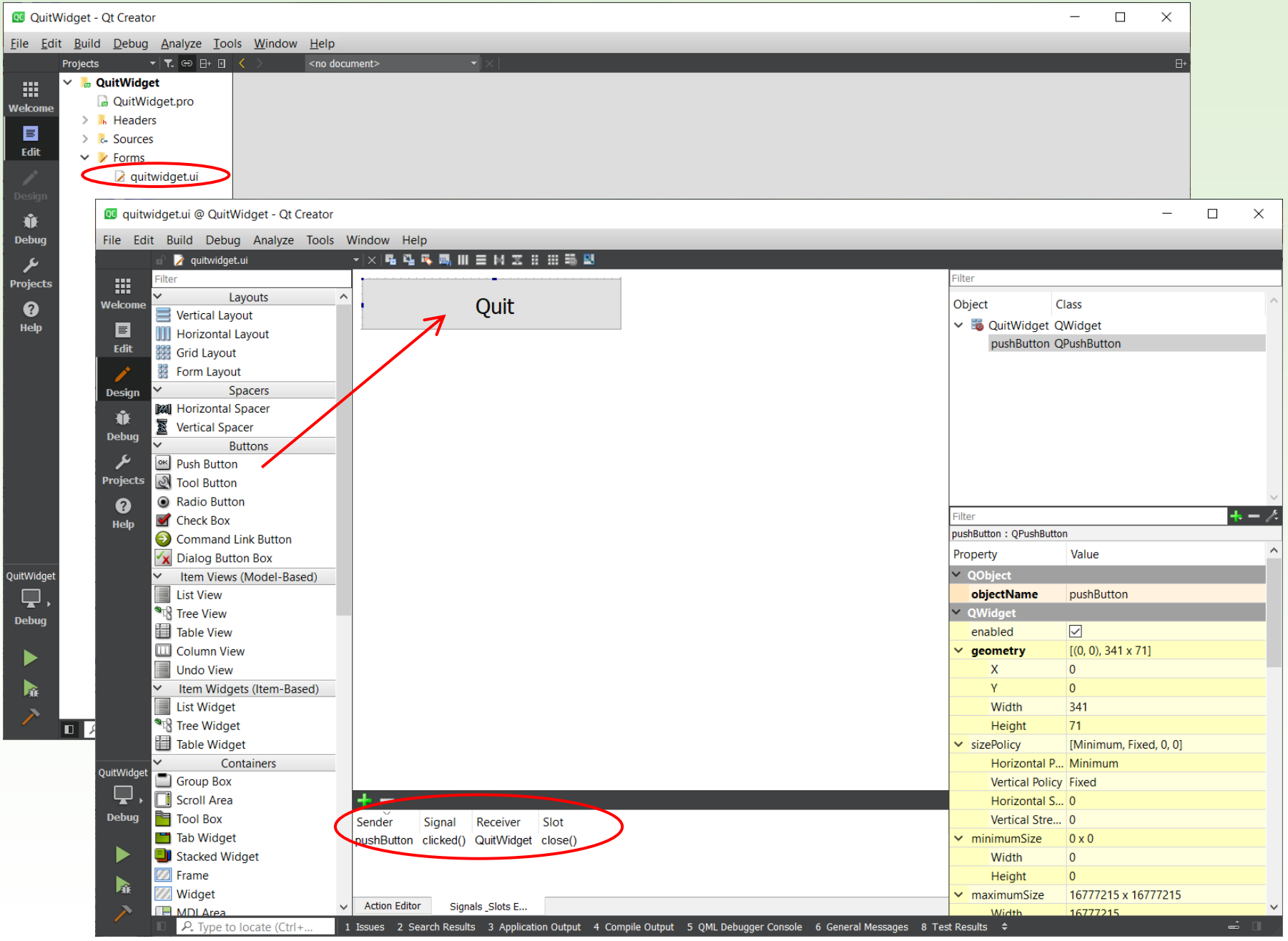

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char *argv[]) {
    QApplication app(argc, argv); // alkalmazás

    QPushButton quit;           // nyomógomb létrehozása
    quit.setText("Quit");       // ez konstruktorban is beállítható
    quit.resize(350, 50);       // méret beállítása
    quit.setFont(QFont("Times", 18, QFont::Bold)); // betűtípus
    quit.setWindowTitle("Quit application");       // ablak címe
    quit.setToolTip("You can exit");       // előugró szöveg

    QObject::connect(&quit, SIGNAL(clicked()), &app, SLOT(quit()));

    quit.show();          // nyomógomb megjelenítése
    return app.exec(); // alkalmazás indítása
}
```

Lehetne a quit objektum close() metódusa is

Felülettervező

A *felülettervező* (*Qt Designer*) lehetőséget ad a felület gyors elkészítésére

- Az elkészített felületterv XML-ben mentődik (**`<ablaknév>.ui`**), majd abból egy Qt osztály készül (**`ui_<ablaknév>.h`**)
- Az így generált osztály tulajdonságait egy saját, **`Qwidget`**-ből származó osztály részévé úgy tehetjük, hogy
  - vagy annak másik őse a *Qt Designer*-rel generált osztály lesz,
  - vagy adattagként hivatkozunk (objektum befecskendezés) a generált osztály egy példányára, illetve annak vezérlőire
- A generált osztály vezérlőinek kialakítását a generált osztály **`setupUi(QWidget* parent)`** metódusának hívásával kell elvégezni.


QuitWidget - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
<no document>
QuitWidget
QuitWidget.pro
Headers
Sources
Forms
quitwidget.ui
Welcome
Edit
Design
Debug
Projects
Help
quitwidget.ui @ QuitWidget - Qt Creator
File Edit Build Debug Analyze Tools Window Help
quitwidget.ui
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Quit
Sender Signal Receiver Slot
pushButton clicked() QuitWidget close()
Action Editor
Signals _Slots E...
Object Class
QuitWidget QWidget
pushButton QPushButton
pushButton : QPushButton
Property Value
QObject
objectName pushButton
QWidget
enabled
geometry [(0, 0), 341 x 71]
X 0
Y 0
Width 341
Height 71
sizePolicy [Minimum, Fixed, 0, 0]
Horizontal P... Minimum
Vertical Policy Fixed
Horizontal S... 0
Vertical Stre... 0
minimumSize 0 x 0
Width 0
Height 0
maximumSize 16777215 x 16777215
Type to locate (Ctrl+...
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 6 General Messages 8 Test Results

Generált kód

```
namespace Ui { class QuitWidget; }

class QuitWidget : public QWidget
{
    Q_OBJECT
public:
    explicit QuitWidget(QWidget *parent = nullptr);
    ~QuitWidget();
private:
    Ui::QuitWidget *ui;
};
```

```
#include "quitwidget.h"
#include "ui_quitwidget.h"

QuitWidget::QuitWidget(QWidget *parent) :
    QWidget(parent), ui(new Ui::QuitWidget)
{
    ui->setupUi(this);
}
QuitWidget::~QuitWidget()
{
    delete ui;
}
```

innentől használhatók a tervezővel definiált vezérlők: `ui->`pushButton