C++ QT

Matthias Colin

Historique

- C : 1970s
- C++ : 1980s, normé en 1998, 2003, **2011**, 2014, 2017, 2020, 2023
- Qt : 1995, GUI, simplification de la gestion des données
  - en C++
  - bindings dans d'autres langages Python, Java, …
  - Qt 4 (2005)
  - Qt 5 (2012) et Qt 6 : QWidgets, UI
  - Qt 6 (2020)
  - QtQuick (QML)
- Python : 1989, fonctionnel, POO
  - en C/C++
  - version 3.12 et feu 2.7
  - librairies Web, Gui, Scipy, GIS, ML, ...

Bindings Python

- Commun
  - QtQuick + Qt5 ou Qt6
- PyQt
  - https://www.riverbankcomputing.com/software/pyqt/
- Qt for Python, PySide
  - https://www.qt.io/qt-for-python

QML

- UI Mobile (téléphone, tablette, ..)
- Desktop
- Langage:
  - Possible sans C++: QML avec JavaScript
  - Dialogue avec C++ ou Python (Qt for Python/PySide ou PyQt)
  - Composants: QtQuick components

Qt, QtWidgets, UI

- GUI
  - QtWidgets
  - .UI file describing GUI
- Non graphic:
  - Model (data), OOP
  - Network
  - Multithreading
  - Database
  - Tests
  - …

Qt Objects

- QObject: parent class of many QT classes
- QString: text
- Integers: qint8, qint16, qint32, qint64, quint8, quint16, quint32, quint64
  - Ex: qint8 -128 to 127 ; quint8 0 to 255
- Floats: qreal (IEEE754)
- Temporal data: QDate, QDateTime, QTime
- QVariant: open union type

Containers

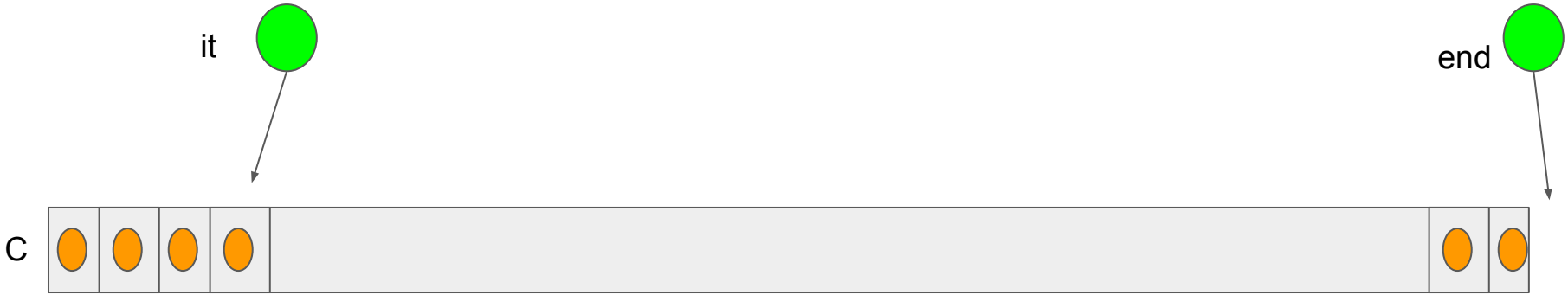
- QT Containers: QVector, QList, QHash, QMap, QQueue, QSet
  - QStringList

  https://doc.qt.io/qt-6/containers.html

- C++ standard containers: std::vector, std::list, …
- Mécanisme d’iterateur

Iterator / Iterable

- Article: https://en.cppreference.com/w/cpp/iterator
- Indépendance du parcours avec la structure (external object)
- Parcours exhaustif des éléments un à un
- Opérations
  - démarrer un parcours : créer un iterator positionné au début de la structure : it = c.begin()
  - passer au suivant (next) : ++it (opt: --it, it += 3, ..)
  - lire l'élément courant : element = *it
  - savoir si le parcours est fini : c.end() with ==, !=



Display, Debug, Output

QT

- QDebug
- QDataStream
- QTextStream

C++ standard

- std::ostream (console, file, string)

QT Meta model

- Class with Q_DECLARE_METATYPE + qRegisterMetaType
  - can be used as a parameter in signal/slot
  - can be used in a variant
- Class inherits from QObject + Q_OBJECT
  - idem META_TYPE (auto registered)
  - signal/slots + QObject::connect
  - Q_PROPERTY
  - copy constructor and operator are deactivated
- QML: metatype + Q_GADGET + Q_PROPERTY

Threads vs Process

- Process: aucun partage de ressource par défaut (environnement, RAM)
  - Partage à créer avec:
    - communication par socket
    - mémoire partagée
  - QProcess
- Threads: appartiennent au même processus
  - Partage environnement
  - Partage RAM (Instructions, Données globales, **Heap**)
  - Chacun a son propre stack (appel de fonction, var locales, …)
  - https://doc.qt.io/qt-6/threads-technologies.html
    - QThread (low level)
    - QThreadPool and Runnable
    - QtConcurrent and Future (Promise)

SQL Database

- Relational Databases using SQL
- Standard SQL
- Editors:
  - Oracle Database
  - MySQL / MariaDB
  - PostgreSQL
  - Microsoft SQL Server
  - SQLite (File ou Memory)
  - IBM DB2
- Driver Qt SQL
- QSqlTableModel for GUI

QTest

Test frameworks: QTest, google test, boost, …

QTest:

- https://doc.qt.io/qt-6/qttestlib-tutorial1-example.html
-

Network

https://doc.qt.io/qt-6/topics-network-connectivity.html

https://doc.qt.io/qt-6/examples-network.html

Sockets: local, TCP, UDP, …

Connectivity: Bluetooth, NFC, ...

Files

QFile

https://doc.qt.io/qt-6/qfile.html

QtWidgets

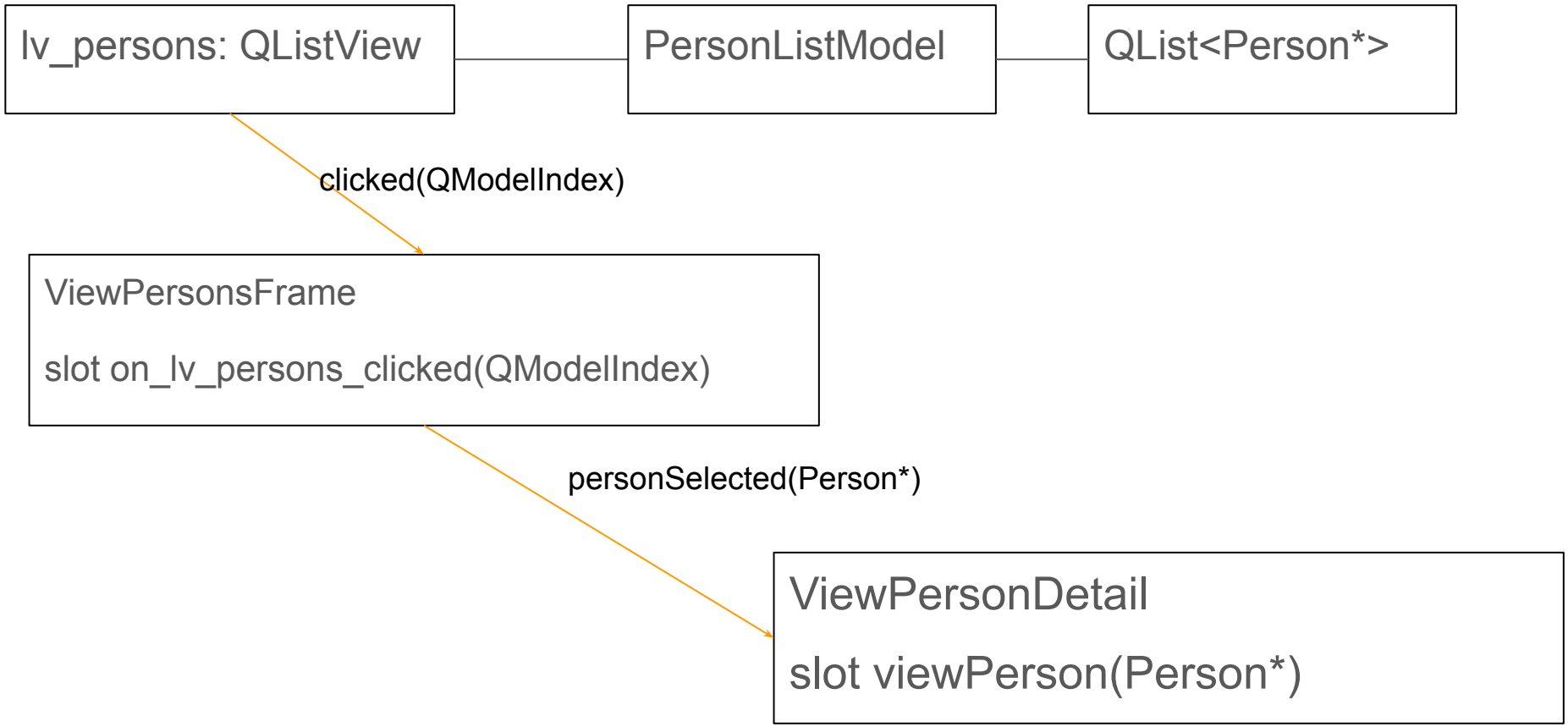
- Simple model (1 value)
  - QLabel (text)
  - QLineEdit (text)
  - QSpinBox (value)
  - QDateEdit (date)
  - QSlider
- Complex model (n values)
  - Widget
    - QComboBox
    - ListView, TreeView, TableView
    - ListWidget, TreeWidget, TableWidget
  - Model: QAbstractItemModel
    - QAbstractListModel, QStringListModel
    - QAbstractTableModel, .

https://doc.qt.io/qt-5/model-view-programming.html

LayoutManager

- https://doc.qt.io/qt-5/layout.html
- Exemples
  - QFormLayout
  - QGridLayout
  - QHBoxLayout
  - QVBoxLayout
  - QStackedLayout

Signal/Slot


lv_persons: QListView
PersonListModel
QList<Person*>
clicked(QModelIndex)
ViewPersonsFrame
slot on_lv_persons_clicked(QModelIndex)
personSelected(Person*)
ViewPersonDetail
slot viewPerson(Person*)

Signal/slots

https://wiki.qt.io/New_Signal_Slot_Syntax

Custom UI component

2 possibilities:

- custom.cpp,.h,.ui reusable (not visible)
- custom widget for qt designer

  https://doc.qt.io/qt-6/designer-creating-custom-widgets.html

Dialogs

- QDialog
  - QMessageBox : simple dialogs
  - QFileDialog : choose a file/directory

QML: types de données basiques

- https://doc.qt.io/qt-6.2/qtqml-typesystem-basictypes.html
- nombre :
  - int 12
  - real, double 3.14
- booléen : bool true, false
- texte : string "Toulouse"
- énumération : enumeration définie par Qml
- liste : list [1,2,3]
- localisation ressource : url
- générique : var
- temporel : date (jour + heure)
- géométriques: point (x,y), size (width, height), rect (x, y, width, height)

QtQml types

- https://doc.qt.io/qt-5/qtqml-qmlmodule.html
- Component : composant Qml interne
- QtObject
- Number, String, Date : formatage de données
- Locale
- Binding
- Timer

QtQuick basics items

- https://doc.qt.io/qt-6.2/qml-qtquick-item.html
- Item : général
  - Text, TextInput, TextEdit : texte
  - Rectangle : couleur de fond, bordure
  - Image : image 2D seule
  - Canvas : dessin 2D, image
  - Shape
  - Grid, Row, Column, Flow
  - GridLayout, RowLayout, ColumnLayout, StackLayout
  - Repeater
  - MouseArea, MultiPointTouchArea
  - Loader
- Window : fenêtre simple, composant racine

QtQuick Controls

- https://doc.qt.io/qt-5/qtquick-controls2-qmlmodule.html
- Label
- TextField, TextArea
- CheckBox, RadioButton, ComboBox
- Slider, SpinBox, Tumbler
- Button, RoundButton, Switch
- Frame, Page, StackView, TabView, SplitView, ScrollView, SwipeView
- MenuBar, TabBar, Popup, Drawer
- ApplicationWindow

QtQuick Model View

- ListView, GridView
  - ListModel, XmlListModel : modèle Qml
  - QAbstractListModel, QAbstractItemModel : modèle C++, python, …
- TableView
  - TableModel, ListModel, XmlListModel : modèle Qml
  - QAbstractTableModel, QAbstractItemModel : modèle C++, python, …
- TreeView
- property delegate : dessin de chaque item

https://doc.qt.io/qt-6/model-view-programming.html

https://doc.qt.io/qt-6/modelview.html

Positionnement

- Fixe
- Anchors
- Positioner
  - Row, Column, Grid,
- Layout
  - RowLayout, ColumnLayout, GridLayout, StackLayout

Modèle QAbtractItemModel

- Base de QAbstractListModel, QAbstractTableModel, YourModel
- méthodes à implémenter :
  - rowCount [, columnCount]
  - [headerData]
  - data [, setData]
  - [flags]
  - [roleNames]
- tout changement doit être encadré
  - beginInsertRows / endInsertRows, …
  - dataChanged

Qt Role	QML Role Name
Qt::DisplayRole	display
Qt::DecorationRole	decoration
Qt::EditRole	edit
Qt::ToolTipRole	toolTip
Qt::StatusTipRole	statusTip
Qt::WhatsThisRole	whatsThis

TreeModel

- No base class
- Example TreeModel + TreeItem
  - https://doc.qt.io/qt-6/qtwidgets-itemviews-simpletreemodel-example.html
  - https://code.qt.io/cgit/qt/qtbase.git/tree/examples/widgets/itemviews/simpletreemodel?h=6.6
- Need 2 extra methods:
  - item
  - parent
- UI:
  - QTreeView (QWidget .ui)
  - TreeView
    - QT5.5+ but only qtquickcontrols 1 (not 2)
    - available as extension (market) for QT 5.15 and QT 6.0->6.2
    - QT6.3+ available again
    - TreeViewDelegate (can be customized)

Internationalization

- python code
  - https://wiki.qt.io/PySide_Internationalization
  - pyside2-lupdate, pygettext, xgettext
- QML, C++ Qt
  - https://doc.qt.io/qt-5/internationalization.html
  - https://doc.qt.io/qt-5/linguist-manager.html
  - lupdate
- QtLinguist
  - https://doc.qt.io/qt-5/linguist-translators.html
- lrelease : transforme .ts en .qm

Serveur Tcp/Ip

- https://doc.qt.io/qt-5/qtcpserver.html
-

Testing

- tests unitaires
  - chaque fonction, avec paramètre significatif
  - mocking : test brique
- tests fonctionnels
  - enchaînements, scenarios
- tests end2end
- TDD : Test Driven Development
- BDD : Behaviour Driven Development

ProxyModel

https://doc.qt.io/qt-6/qsortfilterproxymodel.html

QT5 -> QT6

- QT
  - simplifications: QList fusionne avec QVector
  - new macros: QML_VALUE_TYPE (with Q_GADGET)
  - qmake => cmake (en cours)
- QML
  - plus de version nécessaire des modules QML (last version 2.15 from QT 5.15)
  - TreeView réapparaît en 6.3
  - réorganisation des modules
- Optimisations

https://www.qt.io/product/qt6/technical-specifications

Performance, Profiling

https://wiki.qt.io/Profiling_and_Memory_Checking_Tools

- environment variables: PATH (+ unix/linux: LD_LIBRARY_PATH)
- use same compiler (same version) => cross compilation