The Role of new Languages in the Future of the Qt Ecosystem.

Dr. Cristián Maureira-Fredes
maureira.dev
@cmaureir

What is the best programming language?

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.10%
2	2		C++	9.18%	-0.86%
3	3		C	9.03%	-0.15%
4	4		Java	8.59%	-0.58%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	↑	Visual Basic	2.33%	+0.15%
8	9	↑	Go	2.11%	+0.08%
9	25	⇈	Perl	2.08%	+1.17%
10	12	↑	Delphi/Object Pascal	1.82%	+0.19%

Source TIOBE index August 2025

Most Popular technologies (Professionals) - Source StackOverflow Survey 2025

Most Popular technologies (Learning) - Source StackOverflow Survey 2025

Most Popular technologies (All) - Source StackOverflow Survey 2025


Qt

Languages in the Qt Project

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

1995

1995

2010

1995

2010

2011*

1995

2010

2011*

2025

?

The approach so far:

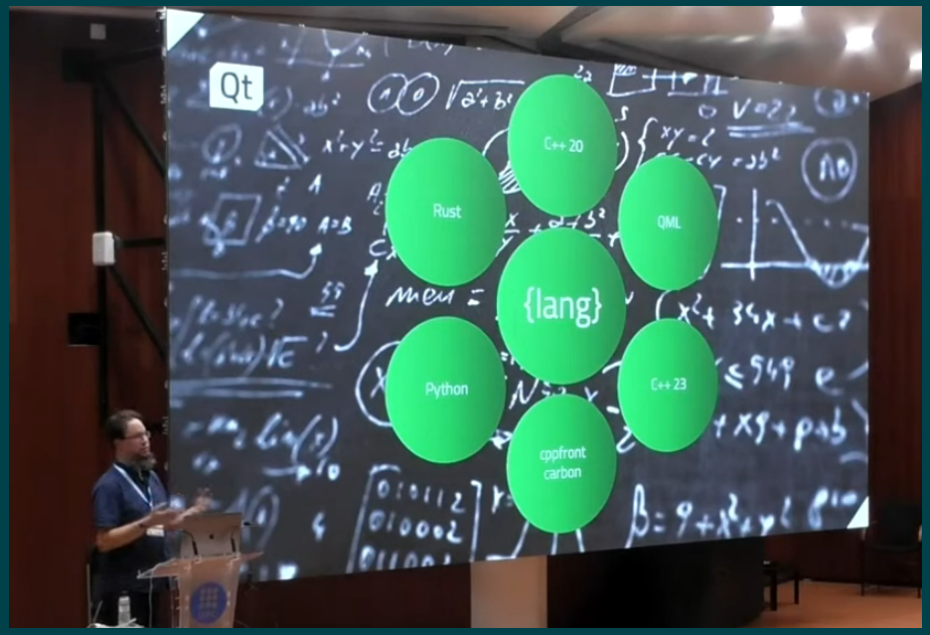
The approach so far:

Bindings

Name	Language	Latest version	Last commit	License	Company
PySide	Python	6.9.2	-	LGPLv3/Commercial	The Qt Company
PyQt	Python	6.9.1	-	GPLv3/Commercial	Riverbank Computing
CXX-Qt	Rust	6.x*	-	MIT/Apache 2.0	KDAB
qmetaobject-rs	Rust	6.5.0	2025.01.18	MIT	Woboq
QtJambi	Java/Kotlin	6.9.2	2025.08.27	GPLv3/LGPLv2	Omix Visualization
QML.jl	Julia	6.5.2	2023.08.20	MIT	JuliaGraphs
NodeGui	Node.js	6.6.0	2025.06.30	MIT	NodeGui
nimqt	Nim	6.4.2	2024.11.5	GPL2	-
QML-zig	Zig	6.4.2	2025.05.09	Apache 2.0	-

(updated on Sep 2nd, 2025)

Other 24 projects (some inactive) in wiki.qt.io/Language_Bindings



Keynote: Building the Future of Qt, Together

Volker Hilsheimer
(Akademy 2022)

Snapshot from https://youtu.be/kiw4fQH9vb8

What has been the motivation?

What has been the motivation?

- The need of GUI frameworks

What has been the motivation?

- The need of GUI frameworks
- C++ flexibility for bindings

What has been the motivation?

- The need of GUI frameworks
- C++ flexibility for bindings
- Language showcase

What has been the motivation?

- The need of GUI frameworks
- C++ flexibility for bindings
- Language showcase
- Qt's popularity ⭐

What other languages bring to the table

What Python and Rust bring to the table

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Python's syntax

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPushButton>

class MainWindow : public QMainWindow
{
    Q_OBJECT
    public:
        MainWindow(QWidget *parent = nullptr);
    private slots:
        void handleButton();
    private:
        QPushButton *m_button;
};

#endif // MAINWINDOW_H
```

```
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent)
{
    m_button = new QPushButton("My Button", this);
    connect(m_button, SIGNAL(clicked()), this, SLOT(handleButton()));
}

void MainWindow::handleButton()
{
    m_button->setText("Ready");
}
```

```
#include
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow mainWindow;
    mainWindow.show();
    return app.exec(d);
}
```

Python's syntax

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPushButton>

class MainWindow : public QMainWindow
{
    Q_OBJECT
    public:
        MainWindow(QWidget *parent = nullptr);
    private slots:
        void handleButton();
    private:
        QPushButton *m_button;
};

#endif // MAINWINDOW_H
```

```
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    m_button = new QPushButton("My Button", this);
    connect(m_button, SIGNAL(clicked()), this, SLOT(handleButton()));
}

void MainWindow::handleButton()
{
    m_button->setText("Ready");
}
```

```
#include
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow mainWindow;
    mainWindow.show();
    return app.exec(d);
}
```

```
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.button = QPushButton("My Button", self)
        self.button.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.button.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

Python's heterogeneity

Python's heterogeneity

Web Development

FastAPI

Python's heterogeneity

Web Development

Data Science

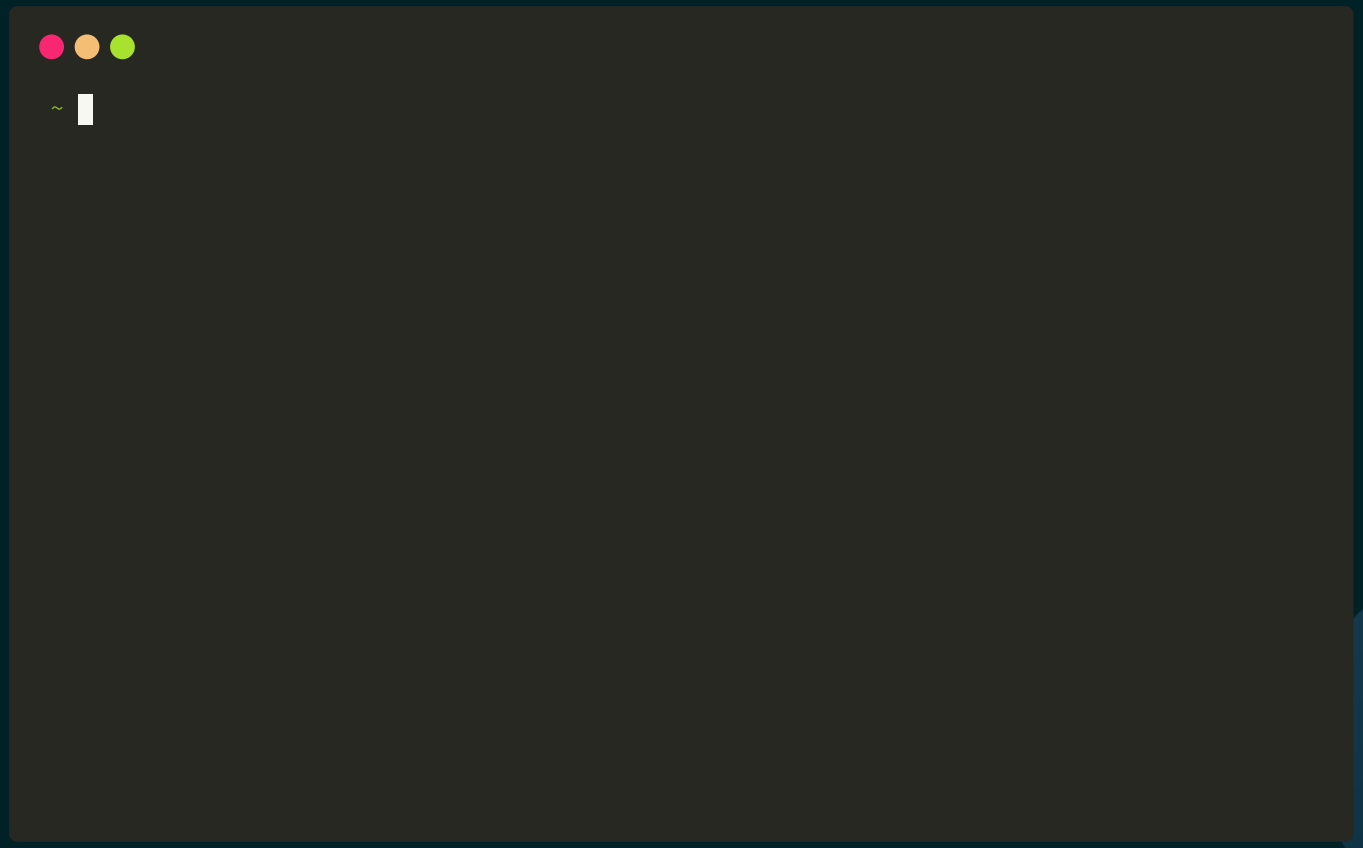
Python's heterogeneity

Web Development

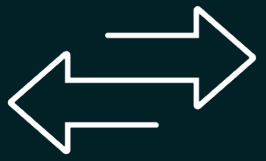
Data Science

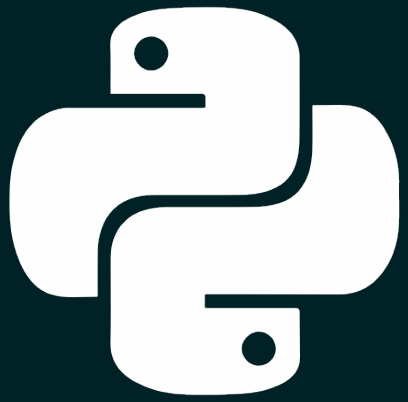
Embedded systems

Python: package distribution




C++





More Qt for Python

- Wiki page pyside.org
- 📚 Official docs doc.qt.io/qtforpython
- 🤝 Community at wiki.qt.io/Qt_for_Python#Community
- 📺 Talks at doc.qt.io/qtforpython-6/videos.html

Rust and C++ (1/2)

```cpp
#include <iostream>

using namespace std;

int main() {
    int nums[] = {1,2,3,4,5};
    int five = nums[1] + nums[2];
    cout << "Result: " << five";
}
```

```rust
fn main() {
    let arr: [i32; 5] = [1, 2, 3, 4, 5];
    let five = arr[1] + arr[2];
    println!("Result: {}", five);
}
```

Rust and with C++ (2/2)

```cpp
class Person {
  public:
    Person(string n, int a);
    void talk();
    string name;
    int age;
};
Person::Person(string n, int a){
    name = n;
    age = a;
}
void Person::talk(){
    cout << name << " says Hello!\n";
}

int main() {
    Person maria("Maria", 22);
    maria.talk();
}
```

```rust
// declare a struct
struct Person {
    name: String,
    age: i32
}
impl Person {
    fn speak(&self){
        println!("{} says Hello!",
          self.name);
    }
}

fn main() {
    let maria = Person{
      name: String::from("Maria"),
      age: 5
    };
    maria.speak();
}
```

CXX-Qt Rust ⇄ C++ bindings

```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

CXX-Qt Rust ⇄ C++ bindings

```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

- CXX - safe FFI between Rust and C++

Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

CXX-Qt Rust ⇄ C++ bindings

```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

- CXX - safe FFI between Rust and C++
- Idiomatic C++ and Rust

Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

CXX-Qt Rust ⇄ C++ bindings

```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

- CXX - safe FFI between Rust and C++
- Idiomatic C++ and Rust
- Macros → Bindings

Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

CXX-Qt Rust ⇄ C++ bindings

```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

- CXX - safe FFI between Rust and C++
- Idiomatic C++ and Rust
- Macros → Bindings
- Rust threads → update Qt state

Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

CXX-Qt Rust ⇄ C++ bindings

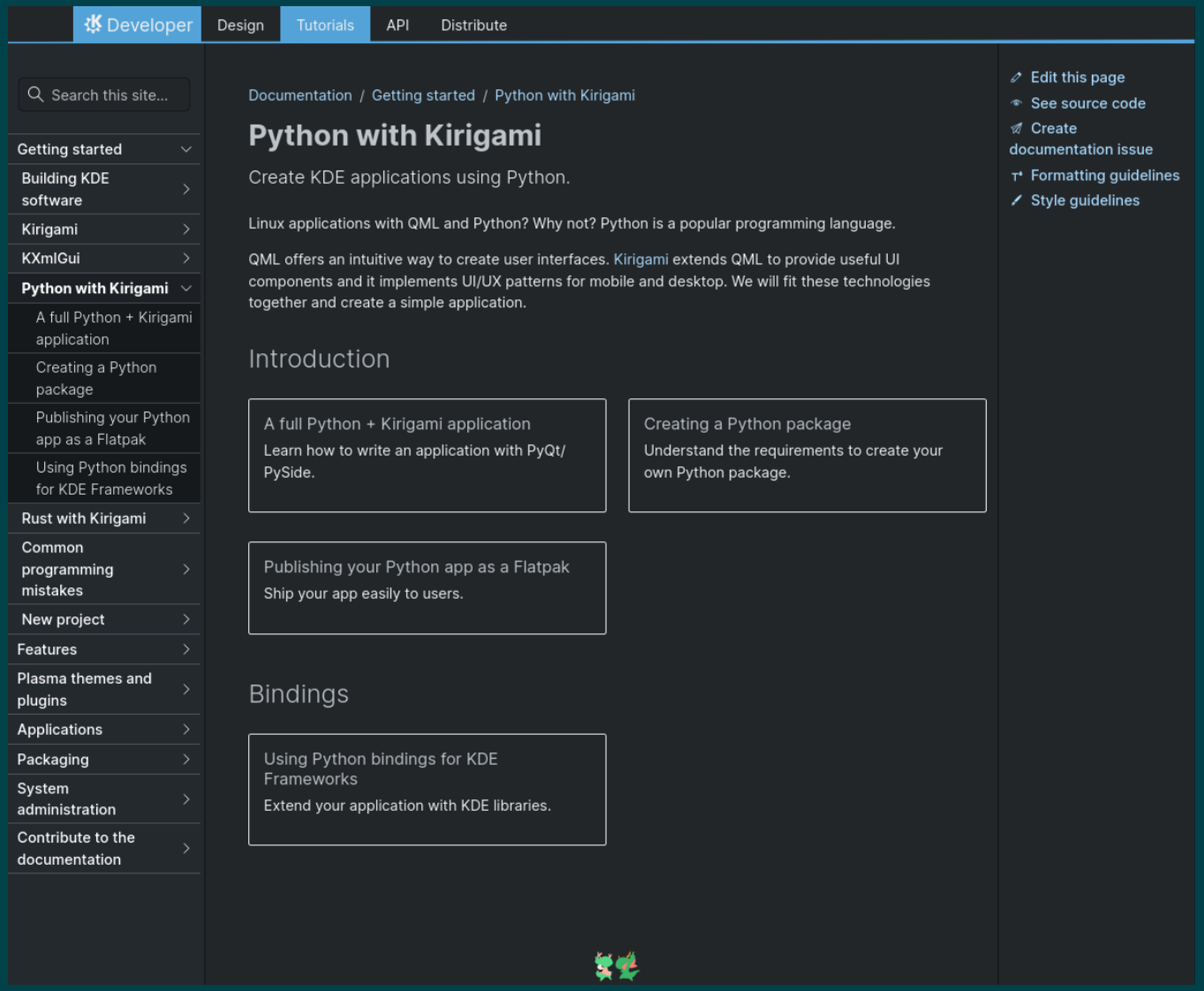
```rust
#[cxx_qt::bridge]
pub mod qobject {
  unsafe extern "C++" {
    include!("cxx-qt-lib/qstring.h");
    type QString = cxx_qt_lib::QString;
  }
  unsafe extern "RustQt" {
    #[qobject]
    #[qml_element]
    #[qproperty(i32, number)]
    #[qproperty(QString, string)]
    type MyObject = super::MyObjectRust;
  }
  unsafe extern "RustQt" {
    #[qinvokable]
    fn increment_number(self: Pin<&mut MyObject>);
    #[qinvokable]
    fn say_hi(self: &MyObject,
              string: &QString,
              number: i32);
  }
}
```

- CXX - safe FFI between Rust and C++
- Idiomatic C++ and Rust
- Macros → Bindings
- Rust threads → update Qt state
- Still in development

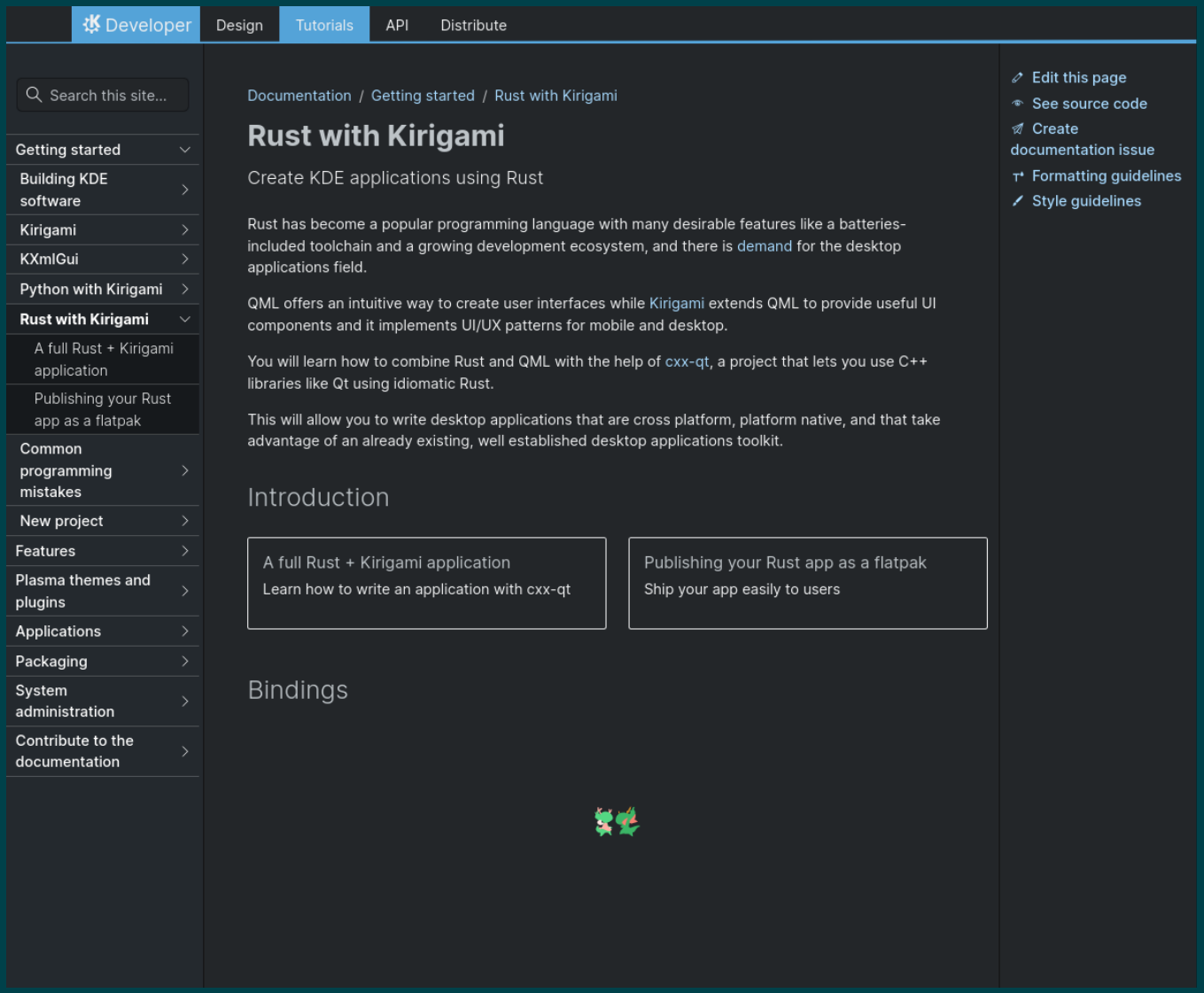
Snippet from: github.com/KDAB/cxx-qt/tree/main/examples/qml_minimal

This worked well for Qt developers

...also for KDE developers



Source: develop.kde.org/docs/getting-started/python/



Source: develop.kde.org/docs/getting-started/rust/

Session at the end of the day

What about people that's unaware of Qt?

Python's case (from before)

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPushButton>

class MainWindow : public QMainWindow
{
    Q_OBJECT
    public:
        MainWindow(QWidget *parent = nullptr);
    private slots:
        void handleButton();
    private:
        QPushButton *m_button;
};

#endif // MAINWINDOW_H
```

```
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent)
{
    m_button = new QPushButton("My Button", this);
    connect(m_button, SIGNAL(clicked()), this,
      SLOT(handleButton()));
}

void MainWindow::handleButton()
{
    m_button->setText("Ready");
}
```

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow mainWindow;
    mainWindow.show();
    return app.exec(d);
}
```

```
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class
- Understand Qt-API

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class
- Understand Qt-API
- Connecting a button with a Slot

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class
- Understand Qt-API
- Connecting a button with a Slot
- Decorating a method as a Slot

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class
- Understand Qt-API
- Connecting a button with a Slot
- Decorating a method as a Slot
- Event loop

Python's case

```python
import sys
from PySide6.QtCore import Slot
from PySide6.QtWidgets import (
  QApplication, QMainWindow, QPushButton
)

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.b = QPushButton("My Button", self)
        self.b.clicked.connect(self.handle_button)

    @Slot()
    def handle_button(self):
        self.b.setText("Ready")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    sys.exit(app.exec())
```

- Inheriting from a Qt class
- Understand Qt-API
- Connecting a button with a Slot
- Decorating a method as a Slot
- Event loop
- Writing a Qt application in Python from scratch

This might feel

not-natural and extra work

We force languages to handle Qt/C++ details

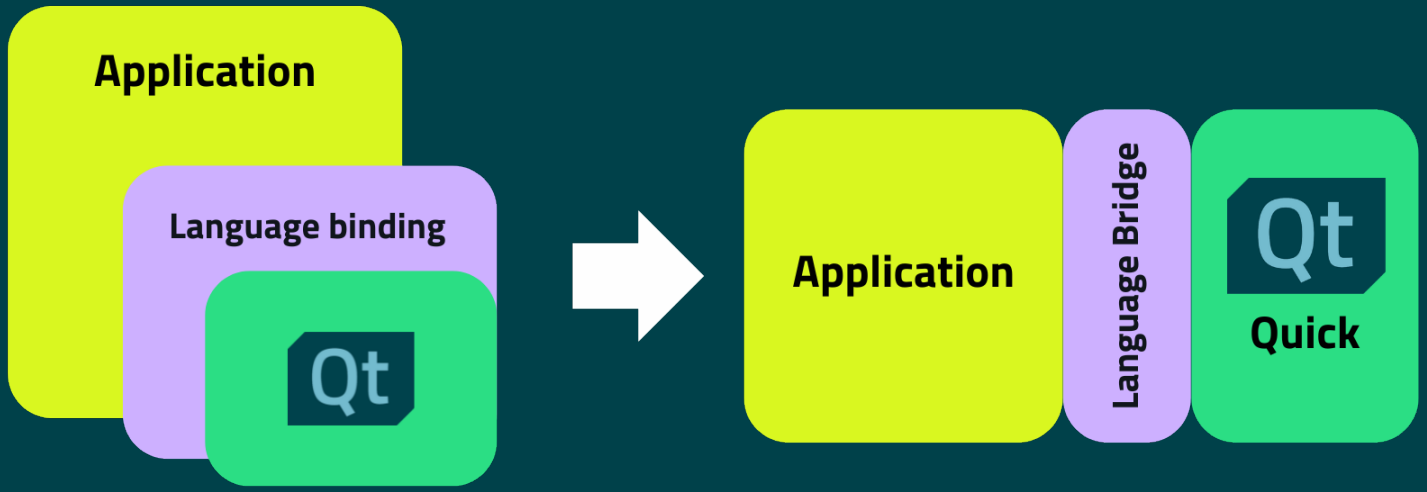
How can we include other languages without breaking the language paradigm?

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

The development team

- Ahmed El Khazari
- Juha Vuolle
- Karsten Lamm-Heimrich
- Lena Biliaieva
- Magdalena Stojek
- Matthias Rauter
- Miguel Costa
- Piotr Wiercinski
- Shyamnath Premnadh
- Yuri Knigavko
- Vladimir Minenko (PM)
- Cristián Maureira-Fredes (TL)
- + More

The new approach

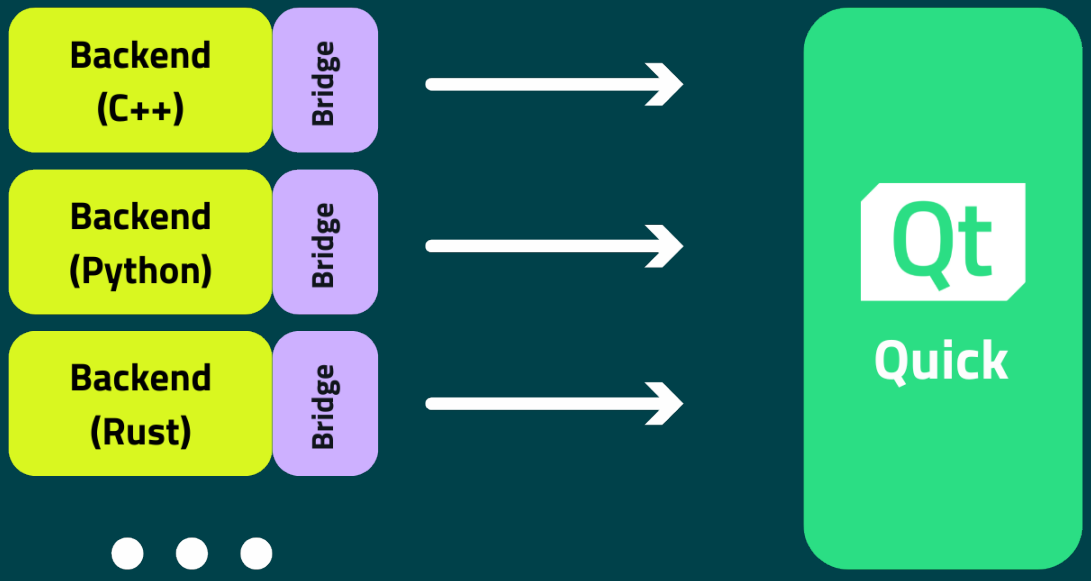


Split Backend from UI:

Reduce Qt footprint from backend

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Language independence



Internal process

- `QMetaObjectBuilder` -> Runtime generation of QML-types
  - Methods, Signals, Slots, Properties, etc.

Internal process

- `QMetaObjectBuilder` -> Runtime generation of QML-types
  - Methods, Signals, Slots, Properties, etc.
- `QQmlPrivate:qmlregister` for instantiable types (Still in progress)

Implementation "detail"

Implementation "detail"

Decorators, Macros, Observable types, etc

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Package distribution

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Package distribution

Wheels (Python), Crates (Rust), NuGet (C#), ...

Development environment

Development environment

Different languages, different IDEs.

⚠️ Please note

⚠️ Please note

The target users are non-Qt developers.

⚠️ Please note

The target users are non-Qt developers.

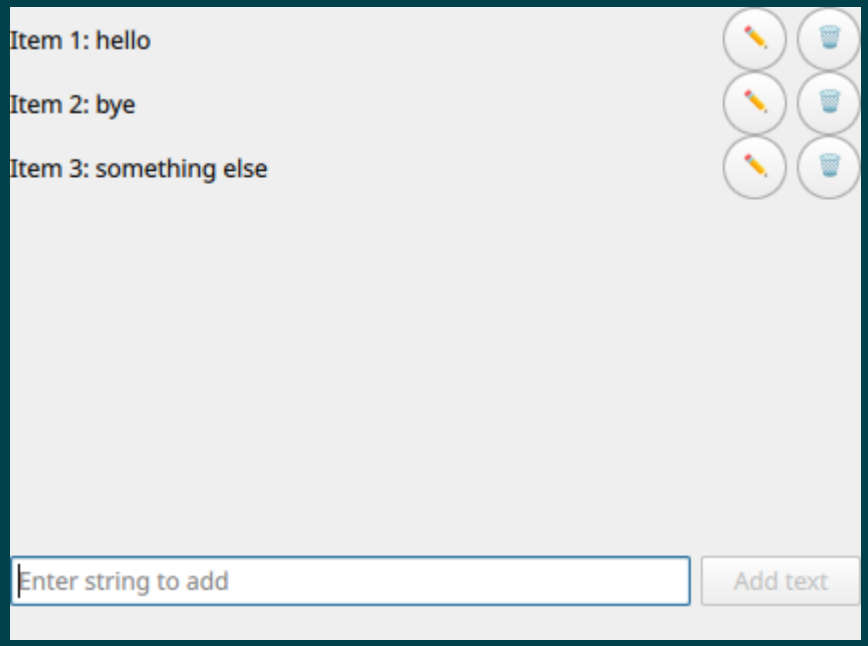
Not everything in Qt will be possible in Qt bridges.

A sneak peak of one application

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Base QtQuick application

```
pragma ComponentBehavior: Bound

import QtQuick
import QtQuick.Controls
import QtQuick.Layouts
import QtQml.Models
import minimalapp

ApplicationWindow {
    id: root

    Timer {
        id: clearStatusTimer
        interval: 1500
        onTriggered: statusDisplay.text = ""
    }

    Connections {
        target: Backend
        function onDuplicateFound(duplicate) {
            statusDisplay.text = `List already contains entry ${duplicate}!`
            clearStatusTimer.restart();
        }
    }

    width: 640
    height: 480
    visible: true
    title: qsTr("Minimal QML app")
    ColumnLayout {
        id: mainLayout
        anchors.fill: parent
        ListView {
            id: lv
            model: Backend.list
            delegate: Control {
                id: ld
                required property var model
                required property int index
                width: lv.width
```

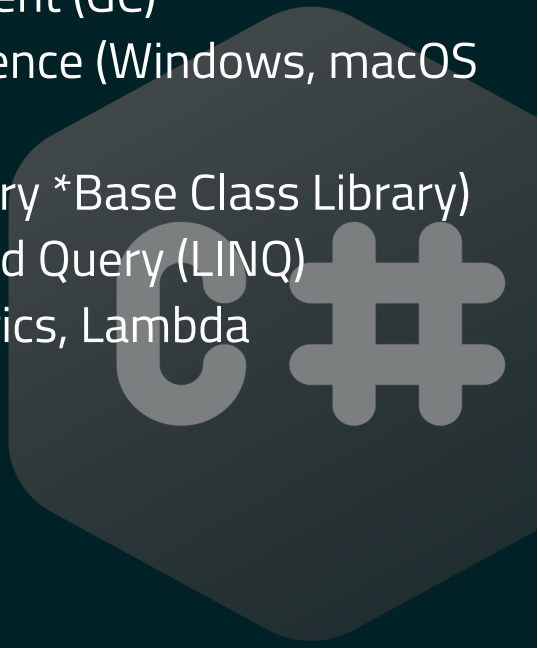



C#

Why C# (1/2)

```
using System;

class Program
{
  static int AddNumbers(int num1, int num2)
  {
    return num1 + num2;
  }

  static void Main(string[] args)
  {
    int result = AddNumbers(40, 2);
    Console.WriteLine($"The sum is {result}");
  }
}
```

- OOP
- Type safety and strong typing
- Memory management (GC)
- Platform Independence (Windows, macOS and Linux)
- Rich Standard Library *Base Class Library)
- Language Integrated Query (LINQ)
- Async/await, Generics, Lambda expressions, etc



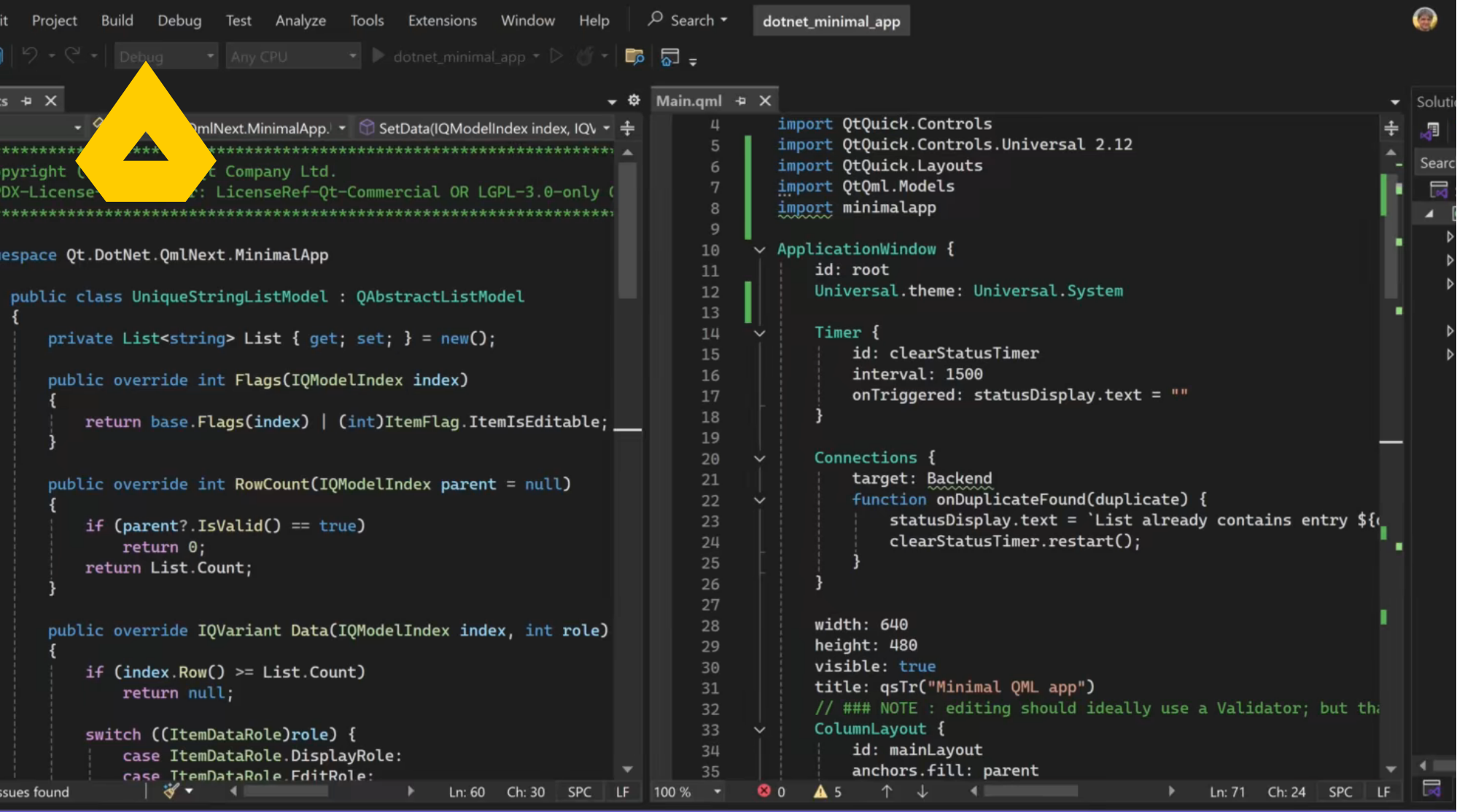
Why C#? (2/2)

- Default language for Windows development
- Previous effort code.qt.io/cgit/qt-labs/qtdotnet
- .NET ecosystem
- Tooling

Qt Bridges (C#) - Minimal application

```cs
// Backend.cs
using Qt.Quick;

namespace Qt.DotNet.QmlNext.MinimalApp
{
  [QmlElement(Singleton = true)]
  public class Backend
  {
    public UniqueStringListModel List { get; } = new();

    public Backend()
    {
        List.DuplicateFound += OnDuplicateFound;
    }

    public bool AddString(string newValue)
    {
        return List.AddString(newValue);
    }

    public delegate void DuplicateFoundHandler(object sender,
      DuplicateFoundArgs args);
    public event DuplicateFoundHandler DuplicateFound;

    private void OnDuplicateFound(object sender,
      DuplicateFoundArgs args)
    {
        DuplicateFound?.Invoke(this, args);
    }
  }

  public class DuplicateFoundArgs : EventArgs
  {
    public string Value { get; set; }
    public override string ToString()
    {
        return Value;
    }
  }
}
```

```cs
// Main.cs
using Qt.Quick;

namespace Qt.DotNet.QmlNext.MinimalApp
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Qml.LoadFromModule("Main");
            Qml.WaitForExit();
        }
    }
}
```

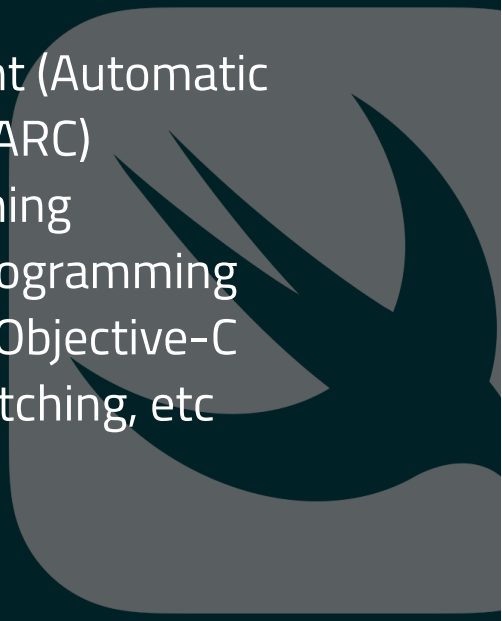

Project Build Debug Test Analyze Tools Extensions Window Help
Search
dotnet_minimal_app
Debug
Any CPU
dotnet_minimal_app
SetData(IQModelIndex index, IQV
Company Ltd.
LicenseRef-Qt-Commercial OR LGPL-3.0-only
espace Qt.DotNet.QmlNext.MinimalApp
public class UniqueStringListModel : QAbstractListModel
{
private List<string> List { get; set; } = new();
public override int Flags(IQModelIndex index)
{
return base.Flags(index) | (int)ItemFlag.ItemIsEditable;
}
public override int RowCount(IQModelIndex parent = null)
{
if (parent?.IsValid() == true)
return 0;
return List.Count;
}
public override IQVariant Data(IQModelIndex index, int role)
{
if (index.Row() >= List.Count)
return null;
switch ((ItemDataRole)role) {
case ItemDataRole.DisplayRole:
case ItemDataRole.EditRole:
Ln: 60 Ch: 30 SPC LF
Main.qml
import QtQuick.Controls
import QtQuick.Controls.Universal 2.12
import QtQuick.Layouts
import QtQml.Models
import minimalapp
ApplicationWindow {
id: root
Universal.theme: Universal.System
Timer {
id: clearStatusTimer
interval: 1500
onTriggered: statusDisplay.text = ""
}
Connections {
target: Backend
function onDuplicateFound(duplicate) {
statusDisplay.text = `List already contains entry ${
clearStatusTimer.restart();
}
}
width: 640
height: 480
visible: true
title: qsTr("Minimal QML app")
// ### NOTE : editing should ideally use a Validator; but th
ColumnLayout {
id: mainLayout
anchors.fill: parent
100 % 0 5 Ln: 71 Ch: 24 SPC LF
XX object CMakeFiles\appminimalapp.dir\main.cpp.obj
XX object CMakeFiles\appminimalapp.dir\backend.cpp.obj
XX object CMakeFiles\appminimalapp.dir\appminimalapp_qmltyperegistrations.cpp.obj
X executable appminimalapp.exe
Add to Source Control

Swift

Why Swift? (1/2)

```swift
import Foundation

func addNumbers(_ num1: Int, _ num2: Int) -> Int {
  return num1 + num2
}

func main() {
  let result = addNumbers(40, 2)

  print("The sum is: \(result)")
}

main()
```

- Safety and Performance
- Modern syntax
- Type Safety and type inference
- Optionals
- Memory management (Automatic Reference Counting, ARC)
- Functional programming
- Protocol-Oriented Programming
- Interoperability with Objective-C
- Generics, Pattern matching, etc



Why Swift? (2/2)

- It's the default language for macOS development
- Previous effort on binding it for Qt Quick apps
- Efforts to bring it to Linux and Windows

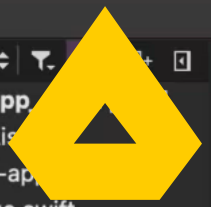
Qt Bridges (Swift) - Minimal application

```swift
// main.swift (1/2)
@QtBridgeable
public class ListModel {
  @Published var list: [String] = ["test string"]
  @Published var duplicateStringFound : String = ""

  public func editString(index: Int, newString: String) {
    if list.contains(newString) {
        postDuplicateNotification(with: newString)
    } else {
        list[index] = newString
    }
  }

  public func addString(newString: String) {
    if list.contains(newString) {
        postDuplicateNotification(with: newString)
    } else {
        list.append(newString)
    }
  }

  public func removeString(index: Int) {
    list.remove(at: index)
  }

  private func postDuplicateNotification(with duplicateStrin
    duplicateStringFound = duplicateString
  }
```

```swift
// main.swift (2/2)
var app = QApp()

var model = ListModel()
app.addInitialProperty(name: "listModel", model: model)

let qmlPath = Bundle.module.url(forResource: "main",
                withExtension: "qml")!.path
app.setRootQml(path: qmlPath)

app.run(argc: CommandLine.argc,
        argv: CommandLine.unsafeArgv)
```

File Edit View Build Debug Analyze Tools Window Help

model.swift @ minimal_app_swift [dev] - Qt Creator



```
minimal_app
CMakeLis
minimal-ap
bridge.swift
model.swift
minimal_app_swift
Header Files
Source Files
main.qml
<Other Locations>
CMake Modules
```

model.swift — Unix (LF)

```
4  import Combine
5  import Foundation
6  import CxxStdlib
7
8  public class SwiftModel: ObservableObject {
9      @Published var list: [String] = ["test string"]
10
11     // use properties for this instead?
12     static let duplicateFoundNotification = Notification.Name("duplicateFound")
13
14     public init() {}
15
16     public func getList() -> [String] {
17         return list
18     }
19
20     public func editString(index: Int, newString: String) {
21         if list.contains(newString) {
22             postDuplicateNotification(with: newString)
23         } else {
24             list[index] = newString
25         }
26     }
27
28     public func addString(newString: String) {
29         if list.contains(newString) {
30             postDuplicateNotification(with: newString)
31         } else {
32             list.append(newString)
33         }
34     }
35
36     public func removeString(index: Int) {
37         list.remove(at: index)
```

Compile Output — Filter

```
11:15:29: Running steps for project minimal_app_swift...
11:15:29: Starting: "/opt/homebrew/Cellar/cmake/3.31.6/bin/cmake" --build /Users/meow/Desktop/qt/minimal_app/swift/build-minimal
ssssss-Debug --target all
ninja: no work to do.
11:15:29: The process "/opt/homebrew/Cellar/cmake/3.31.6/bin/cmake" exited normally.
11:15:29: Elapsed time: 00:00.
```

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Type to locate (⌘K) 1 Iss... 2 Search Res... 3 Application Out... 4 Compile Out... 5 Termi... 6 QML Debugger Cons... 7 General Messa... 9 Test Res...


Java

Why Java? (1/2)

```java
public class Program {

  public static int addNumbers(int num1, int num2) {
    return num1 + num2;
  }

  public static void main(String[] args) {
    int result = addNumbers(40, 2);

    System.out.println("The sum is " + result);
  }
}
```

- Platform Independence
- OOP
- Automatic Memory Management (GC)
- Strong Type System
- Security (management and bytecode verification)
- Multithreading, Rich Standard Library, Large ecosystem

Why Java? (2/2)

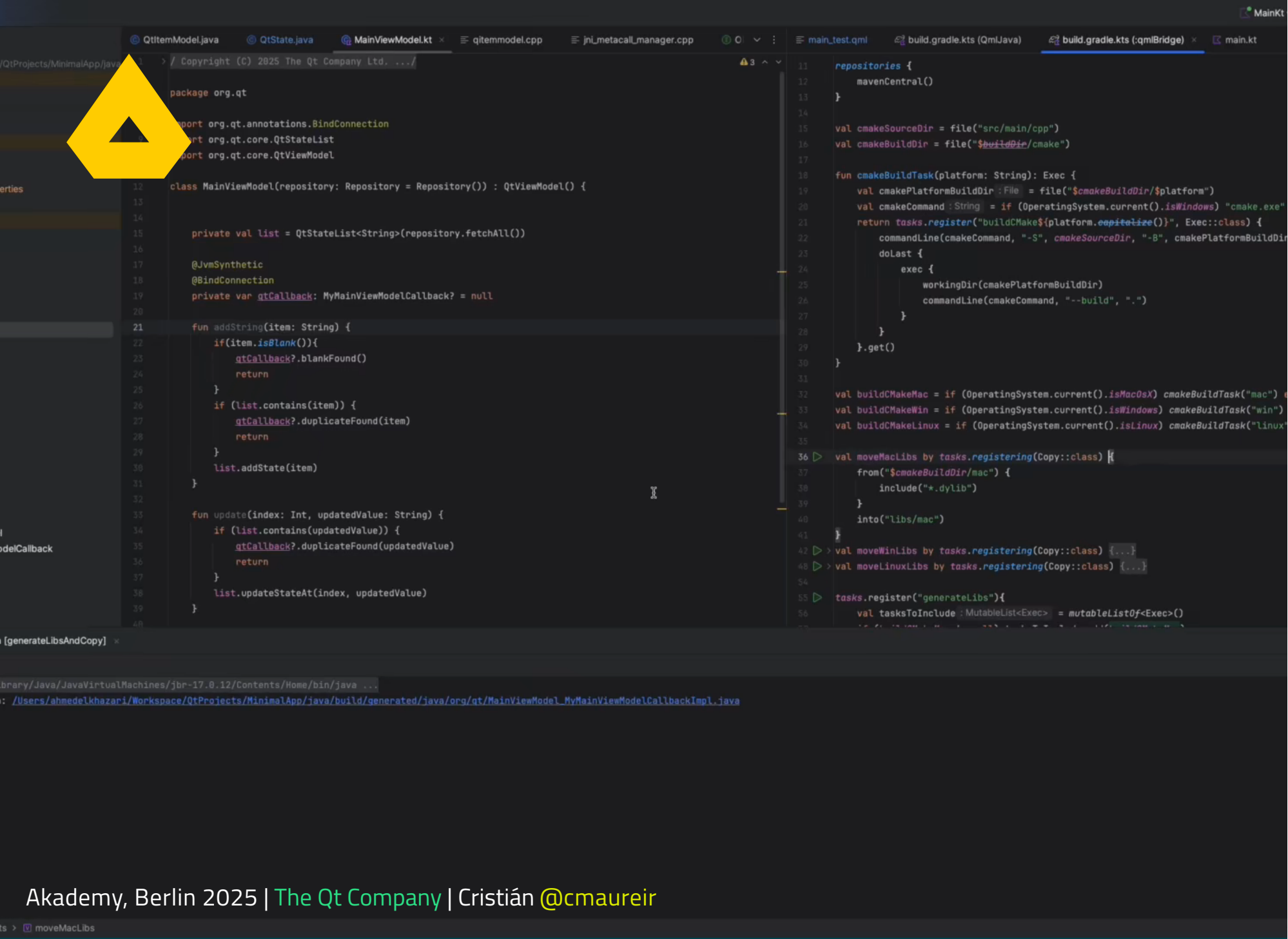
- Already in the ecosystem (Android)
- Automotive experience
- Code generator (Qt Jenny)
- Active 3rd party bindings (Qt Jambi)

Qt Bridges (Java) - Minimal application

```java
// Backend.java
package org.qt.bridge;

import org.qt.bridge.core.QtObject;
import org.qt.bridge.core.QtState;
import org.qt.bridge.core.QtStateModelList;

@QMLRegistrable(name = "Backend")
public class ViewModel extends QtObject {

  @BindConnection
  QCallback qtCallback;

  private final Repository db = new Repository();
  private final QtState<Integer> size = new QtState<>(this,0);
  private final QtStateModelList<String> list = new \
        QtStateModelList<>(db.fetchAll());
  {
    list.observe(item -> size.update(item.size()));
  }

  public void addString(String item) {
    if (item.isBlank()) {
      if (qtCallback != null) {
        qtCallback.blankFound();
      }
      return;
    }
    if (list.contains(item)) {
      if (qtCallback != null) {
```

```java
// QCallback
package org.qt.bridge;

public interface QCallback {
    void duplicateFound(String item);
    void erased();
    void blankFound();

}
```

```java
// Main.java
package org.qt.bridge;

import org.qt.bridge.app.QtQuickApplication;

public class Main {

    public static void main(String[] args) {
        final QtQuickApplication app = new QtQuickApplication(args);
        app.registerQmlSingleton(new ViewModel());
        app.execute();
    }
}
```



Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Interoperability with Kotlin for free*

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Why Python? (1/2)

```python
def add_numbers(a: int, b: int) -> int:
  return a + b


def main():
  result = add_numbers(40, 2)
  print(f"The sum is {result}")


if __name__ == "__main__":
  main()
```

- OOP
- Simple and readable syntax
- Dynamic typing
- High level language
- Extensive standard library
- Huge ecosystem
- Versatility, rapid development and educational value

Why Python? (2/2)

- Previous experience with Qt for Python (PySide)
- Python remains the most popular language
- Interoperability with many technologies

Qt Bridges (Python) - Minimal application

```python
// main.py (1/2)
import sys
from PySide6.QtGui import QGuiApplication
from PySide6.QtQml import QQmlApplicationEngine
from QtBridges import AutoQmlBridge, updateQML

class StringModel:
  def __init__(self):
    self._items = ["Apple", "Banana"]

  @updateQML
  def add_string(self, value: str):
    if value in self._items:
      print(f"Duplicate found: {value}")
      return False
    self._items.append(value)
    return True

  def delete_string(self, value: str):
    if value in self._items:
      self._items.remove(value)
      print(f"Deleted item: {value}")
      print(self._items)
      return True
    return False

  def set_item(self, index: int, value: str):
```

```python
// main.py (2/2)
if __name__ == "__main__":
    app = QGuiApplication(sys.argv)
    engine = QQmlApplicationEngine()

    # Register with QML
    string_model = StringModel()
    AutoQmlBridge(string_model, data_type="List")

    engine.addImportPath(sys.path[0])
    engine.loadFromModule(".", "Main")

    if not engine.rootObjects():
        sys.exit(-1)

    sys.exit(app.exec())
```

View Go Run Terminal Help

python

main.py

QtBridges > examples > minimal_app > main.py

```python
from PySide6.QtQml import QQmlApplicationEngine
from QtBridges import AutoQmlBridge, updateQML


class StringModel:
    def __init__(self):
        self._items = ["Apple", "Banana"]

    @updateQML
    def add_string(self, value: str):
        if value in self._items:
            print(f"Duplicate found: {value}")
            return False
        self._items.append(value)
        return True

    def delete_string(self, value: str):
        if value in self._items:
            self._items.remove(value)
            print(f"Deleted item: {value}")
            print(self._items)
            return True
        return False

    def set_item(self, index: int, value: str):
        if 0 <= index < len(self._items):
            self._items[index] = value
            print(f"Item at index {index} set to {value}")
            print(self._items)
            return True
        return False

    @property
    def items(self) -> list[str]:
        return self._items
```

nal_app

bridge_p.h
bridge.cpp
ule.cpp
rator_p.h
rator.cpp

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

fish - minimal_app

```
(env) ➜ minimal_app git:(97464a2) ✗ python main.py
(env) ➜ minimal_app git:(97464a2) ✗
```

Ln 6, Col 1 Spaces: 4 UTF-8 LF {} Py

Why Rust? (1/2)

```rust
fn add_numbers(num1: i32, num2: i32) -> i32 {
    num1 + num2
}

fn main() {
  let result = add_numbers(40, 2);

  println!("The sum is: {}", result);
}
```

- Memory safety without GC
- Ownership system
- Zero-cost abstractions
- Concurrency safety
- Performance
- Strong type system
- Cargo
- Community and ecosystem
- Functional programming, pattern matching, trait system, etc

Why Rust? (2/2)

- Previous experience (qtmetaobject-rs, CXX-Qt, etc)
- Language popularity
- Ecosystem
- Tooling

Qt Bridges (Rust) - Minimal application

```rust
// backend.rs
use qt_gen::qobject_impl;
use qt_type_lib::{QVariant, QStringList,
                  QModelIndex, QSignalBlocker};

#[derive(Default)]
pub struct Backend {
}

#[qobject_impl(Base = QStringListModel)]
impl Backend {
  #[overridden]
  fn set_data(&mut self, index: &QModelIndex,
      value: &QVariant, role: i32) -> bool {
    if let Ok(value_str) = String::try_from(value) {
      if self.string_list().contains(&value_str) {
        self.duplicate_found(&value_str);
        return false;
      }
    }

    self.base_set_data(index, value, role)
  }

  #[qslot]
  fn add_string(&mut self, value: &str) {
    if self.string_list().contains(&value) {
```

```rust
// main.rs
use bridge::QObjectHolder;
use qt_type_lib::{QGuiApplication, QQmlApplicationEngine, QVariantMap};

mod backend;
use backend::Backend;

fn main() {

  let mut backend = QObjectHolder::new(Backend::default());

  let _app = QGuiApplication::new();
  let mut qml_engine = QQmlApplicationEngine::new();

  let mut properties = QVariantMap::default();
  properties.insert("backend", &backend.as_qobject_mut().into());
  qml_engine.pin_mut().set_initial_properties(&properties);

  let main_qml = include_bytes!("Main.qml");
  qml_engine.pin_mut().load_data(main_qml);

  QGuiApplication::exec();
}
```

minimal_qml [Administrator]

backend.rs

app > src > backend.rs > ...

```rust
use qobject_impl;

#[qobject_impl]

pend {
    m_data: Vec<String>,
}

#[qobject_impl]
impl Backend {

    #[qslot(cpp_name = "addString")]
    fn add_string(&mut self, value: &str) {
        if self.contains_string(value) {
            self.duplicate_found(value);
        }
        else {
            self.m_data.push(value.to_string());
            self.list_changed();
        }
    }

    #[qslot(cpp_name = "removeString")]
    fn remove_string(&mut self, idx: i32) {
        let idx = idx as usize;
        if idx >= self.m_data.len() {
            return;
        }

        self.m_data.remove(idx);
        self.list_changed();
    }

    #[qslot(cpp_name = "setString")]
    fn set_string(&mut self, idx: i32, value: &str) {
        let idx = idx as usize;
        if idx >= self.m_data.len() || self.m_data[idx] == value {
            return;
        }

        if self.contains_string(value) {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS F:\dev\repos\minimal_qml> cargo run --release
   Compiling minimalapp_rust v0.1.0 (F:\dev\repos\minimal_qml\app)
   Compiling bridge v0.1.0 (F:\dev\repos\minimal_qml\bridge)
    Building [=======================> ] 66/68: bridge
```


Ln 2, Col 1 Space

Note

Note

> The previous snippets are only for one specific example. More examples are being worked on, and the API will probably change.

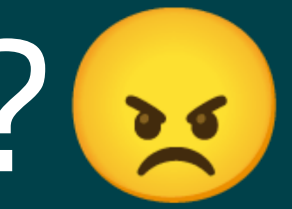
Brief FAQ

(based on common questions)

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

What about <lang>?

What about <lang>?



> One of the goals is to enable people to add more bridges


GO


php


julia


TS


C


Lua


elixir

What about <feature>?🤔

What about <feature>? 🤔

> It can be that some features will not be considered.
> Remember the scope!

What if I need more? 😢

What if I need more? 😢

> If Qt Bridges is not enough, developers will be pointed to any current binding or C++.

So no more bindings? 💀

So no more bindings? 

- The development of binding is still on-going!

Why it's only QtQuick?😒

Why it's only QtQuick? 😔

> QtQuick is a more modern and attractive way of creating UI.

Will you support everything in QML?🤨

Will you support everything in QML? 🤨

> Not at the moment, it depends on the adoption and scope.

Can I combine all the languages? 🧪

Can I combine all the languages? 🧪

- Not for TP, but backend interchangeable

Do you want to brainwash C++ developers? 😵‍💫

Do you want to brainwash C++ developers? 😵

- ~~Yes~~ No

Will this be Open Source?👐

Will this be Open Source? 👐

Of course, the main goal is to expand our community.

When will this be released?

When will this be released?

> Our initial plan is to release by the end of the year (TP)

To summarize...

What does it mean to have more languages with Qt Bridges?

Qt Bridges means more communities

Qt Bridges means new ideas

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

Qt Bridges means new expertise

Qt Bridges means a better Qt ecosystem

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir


Qt
WORLD
DOMINATION

Let's bring Qt everywhere

Akademy, Berlin 2025 | The Qt Company | Cristián @cmaureir

The Role of new Languages in the Future of the Qt Ecosystem.

Dr. Cristián Maureira-Fredes
@cmaureir