Exposing Data to QML

Ulf Hermann

Sept 7th, 2025

Contents

Model/View/Delegate Pattern

- Most views have many data items
- Qt provides
  - Models
  - Views
  - Delegates
- Delegates
  - Receive proxy objects for their model data
  - Bind to model data
  - Can update model data

```qml
ListView {
    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: Text {
        id: delegate
        text: `${model.greeting}, World, ` +
              `${model.value} times!`
        MouseArea {
            anchors.fill: parent
            onClicked: ++model.value
        }
    }
}
```

QML

Why?

- Separation of concerns
- Allows cross product of
  - Multiple models
  - Multiple delegates
  - Multiple views

```qml
// View1.qml
ListView {
    model: findTheModel()
    delegate: findTheDelegate()
}

// Model1.qml
ListModel {
    ListElement { greeting: "Hello"; value: 1 }
    ListElement { greeting: "Goodbye"; value: 3 }
    ListElement { greeting: "Cheers"; value: 4 }
}

// Delegate1.qml
Text {
    // Can use any "greeting" model's data ...
    text: `${model.greeting}, World, ` +
          `${model.value} times!`
    MouseArea {
        anchors.fill: parent
        onClicked: ++model.value
    }
}
```

QML

Default delegates

- TableViewDelegate
- TreeViewDelegate
- Just work (tm)
  - Use default item model roles
  - Allow editing
  - Follow default theme
- ... but let's talke about **custom** delegates

```qml
TableView {
    required model
    delegate: TableViewDelegate {}
}

TreeView {
    required model
    delegate: TreeViewDelegate {}
}
```

QML

Problem: Mismatched interfaces

- Delegate expects data from model
- View cannot verify correctness
- Solution: Required Properties

```qml
// View1.qml
ListView {
    model: findTheModel()
    delegate: findTheDelegate()
}

// Model2.qml
ListModel {
    ListElement { fruit: "apple"; quality: 1 }
    ListElement { fruit: "orange"; quality: 3 }
    ListElement { fruit: "pear"; quality: 4 }
}

// Delegate1.qml
Text {
    // --> Fails silently! <--
    text: `${model.greeting}, World, ` +
          `${model.value} times!`
    MouseArea {
        anchors.fill: parent
        onClicked: ++model.value
    }
}
```

QML

Required Properties

- Declare requirements of delegates
  - Clean error when model doesn't provide
  - Requirements visible ahead of time
- Facilitate separation of concerns
  - Easier re-use of models and delegates

```qml
ListView {
    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: Text {
        id: delegate
        required property string greeting
        required property int value
        text: `${greeting}, World, ${value} times!`
    }
}
```

QML

Required Properties before Qt 6.10

- Declare requirements of delegates
  - Clean error when model doesn't provide
  - Requirements visible ahead of time
- Facilitate separation of concerns
  - Easier re-use of models and delegates
- Do **not** allow updating the model
  - **before** Qt 6.10

```qml
ListView {
    id: listView

    // Doesn't update!
    header: Text { text: listView.model.get(0).value }

    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: MouseArea {
        width: 100; height: 100
        required property int value

        // WRONG! Changes delegate, not model
        onClicked: ++value
    }
}
```

QML

Required Properties from Qt 6.10

- Declare requirements of delegates
  - Clean error when model doesn't provide
  - Requirements visible ahead of time
- Facilitate separation of concerns
  - Easier re-use of models and delegates
- **Do** allow updating the model
  - **starting with** Qt 6.10
  - Use **delegateModelAccess** on the view
  - Available for **all views** shipped with Qt

```qml
ListView {
    id: listView

    // Causes view to set up two-way bindings
    delegateModelAccess: DelegateModel.ReadWrite

    // Updates when model changed by delegate!
    header: Text { text: listView.model.get(0).value }

    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: MouseArea {
        width: 100; height: 100
        required property int value

        // Writes through delegate to model
        onClicked: ++value
    }
}
```

QML

Required Properties from Qt 6.10

- Declare requirements of delegates
  - Clean error when model doesn't provide
  - Requirements visible ahead of time
- Facilitate separation of concerns
  - Easier re-use of models and delegates
- **Do** allow updating the model
  - **starting with** Qt 6.10
  - Use **delegateModelAccess** on the view
  - Available for **all views** shipped with Qt
  - Also works for short form 'required'

```qml
ListView {
    id: listView

    // Causes view to set up two-way bindings
    delegateModelAccess: DelegateModel.ReadWrite

    // Updates when model changed by delegate!
    header: Text { text: listView.model.get(0).value }

    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: SpinBox {
        // Writes through delegate to model
        required value
    }
}
```

QML

Editable Models: Delegate <-> View

- Model provides value
  - Might change spontaneously
- Control populated with model value
  - Needs update on model change
- Control changes value
  - Needs propagation to model

```qml
ListView {
    delegateModelAccess: DelegateModel.ReadWrite
    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: Row {
        id: delegate
        required property int value
        required property string greeting

        SpinBox {
            // * Initialize from 'value' property
            // * Update SpinBox when model changes
            // * Update model when SpinBox edited
            // * Avoid binding loops
            // => How to do this??
        }

        Text { text: delegate.greeting }
    }
}
```

QML

Editable Models: Delegate <-> View

- Common solution: implement in C++
- Separate signal for 'input happened'
  - avoids binding loops
- Change value in C++, not QML
  - avoids breaking bindings
- Requires binding **and** signal handler
- Done for QtQuick controls
- Not ideal for user controls

```qml
ListView {
    delegateModelAccess: DelegateModel.ReadWrite
    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: Row {
        id: delegate
        required property int value
        required property string greeting

        SpinBox {
            // Input does not break this binding!
            value: delegate.value

            // Extra signal, only for input!
            onValueModified: delegate.value = value
        }

        Text { text: delegate.greeting }
    }
}
```

QML

Synchronizer

- New solution: **Synchronizer**
- From Qt 6.10
- Works with any property
  - does not break bindings
  - update property from C++ or QML
- Can synchronize >=2 properties
- Tech Preview for now
- Requires only **one** element

```qml
ListView {
    delegateModelAccess: DelegateModel.ReadWrite
    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: Row {
        id: delegate
        required property int value
        required property string greeting

        SpinBox {
            Synchronizer on value {
                property alias source: delegate.value
            }
        }

        Text { text: delegate.greeting }
    }
}
```

QML

Editable Models: Model <-> View

- Any **QAbstractItemModel** can be edited from delegates
- including **ListModel**
  - which is a custom parser
  - which doesn't scale

```qml
ListView {
    id: listView
    delegateModelAccess: DelegateModel.ReadWrite

    // Updates when model is edited
    header: Text { text: listView.model.get(0).value }

    model: ListModel {
        ListElement { greeting: "Hello"; value: 1 }
        ListElement { greeting: "Goodbye"; value: 3 }
        ListElement { greeting: "Cheers"; value: 4 }
    }

    delegate: SpinBox {
        required value
    }
}
```

QML

Editable Models: Model <-> View

- For **lists**, Qt's views are **black holes**
- ingest source data
- apply magic internally
- produce internal model
- editing affects **only internal** model
  - **not** source data

```qml
ListView {
    id: listView
    delegateModelAccess: DelegateModel.ReadWrite

    // Doesn't update when model is edited!
    header: Text { text: listView.model[0].value }

    model: [
        { greeting: "Hello", value: 1 },
        { greeting: "Goodbye", value: 3 },
        { greeting: "Cheers", value: 4 }
    ]

    delegate: SpinBox {
        required value
    }
}
```

QML

QRangeModel

- since Qt 6.10
- Can be initialized from a QList
  - Or any sequential container
- Can **update** the source data
  - if passed as **pointer**
    - which QML doesn't do
- Is a QAbstractItemModel
  - Can be used for any QML view
- Lacks QML itegration so far

Not Pretty (yet)

```cpp
class RangeModelProvider : public QObject
{
    Q_OBJECT
    QML_ELEMENT
    Q_PROPERTY(QList<QVariantMap> source
            READ source
            WRITE setSource
            NOTIFY sourceChanged)
    Q_PROPERTY(QStringList roleNames
            READ roleNames
            WRITE setRoleNames
            NOTIFY roleNamesChanged)
    Q_PROPERTY(QRangeModel *model
            READ model
            NOTIFY modelChanged)
    [...]
};
```

C++

```qml
ListView {
    delegateModelAccess: DelegateModel.ReadWrite
    RangeModelProvider {
        id: provider
        roleNames: ["name", "value"]
        source: [
            {name: "a", value: 1},
            {name: "b", value: 2}
        ]
    }
    model: provider.model

    // Updates when model changed by delegate!
    header: Text { text: provider.source[0].value }
    delegate: MouseArea {
        width: 100; height: 100
        required property int value

        // Writes through delegate to model
        onClicked: ++value
    }
}
```

QML

SortFilterProxyModel

- Adaptation of QSortFilterProxyModel
- From Qt 6.10
- **Tech Preview**
- Will be improved

Future work

- Finish SortFilterProxyModel
- Shorthand syntax for Synchronizer
- QML keyword for sticky bindings
- Properly integrate QRangeModel
- 'Transactions' for Synchronizer