Model Models: Tools for Making Better Behaved Models

Qt World Summit, 2019

Presented by André Somers

Qt World Summit 2019 Berlin

The Qt, OpenGL and C++ Experts
Introduction

Model types

I guesstimate that:
- ≈ 90% of models are lists
  - Perhaps with different aspects of the items in different columns
  - Includes almost all QML-consumed models
- 9% are trees
- 0.9% are real spread-sheet like tables
- 0.1% really are hierarchical monsters

QAIM API requirements

QAbstractItemModel requires very detailed change notification:
- Before and after
  - For inserts, removals, reset
- On changed

You often don't have all that from your data source... So how do you deal with that?

Real live "solutions"

Observed:
- Simply doing a full model reset
- Partial solutions handling insert or removals, but doing emitting a blanket dataChanged on the whole model
- Messy, hand-crafted in-place code that still cuts a few corners...
- Lots of duplicating the above for code with multiple updatable models...

Demo: talks/modelModels/ex-basic-example

Ideal:
- Simple oneliner that can be re-used...
Towards a solution

Existing (std) algorithms are not suitable:
- No ‘notifications’
- So define our own
  - Avoiding being dependent on container type or data type → templates!

Basic assumptions:
- List-like models
- The model keeps a copy of the data internally in a random-access container
- The updated data is set as a single block
- Both are sorted (or can be sorted) in the same way on some non-changeable key

Basic algorithm

- Inputs:
  - lessThan
  - hasChanged
- Walk through input and exiting data at the same time
  - Using lessThan to see which one(s) to step
- Event-callbacks for needed inserts, removals, data updates
  - (and for equality too)
- Event-callbacks handle actual update and model-signaling

Basic algorithm (cont’d)

```cpp
// precondition: src and target are both ordered with respect to lessThan
template<ForwardIt FwdIt, Container TargetCollection,
         BinaryPredicate LessThan, BinaryPredicate HasChanged,
         EventHandler OnChanged, EventHandler OnInsert, EventHandler OnRemove, EventHandler OnEqual>
void updateCollection(const FwdIt srcBegin, const FwdIt srcEnd, TargetCollection& target,
                      LessThan lessThan, HasChanged itemHasChanged,
                      OnChanged onChanged, OnInsert onInsert, OnRemove onRemove, OnEqual onEqual) {
  auto srcIt = srcBegin;
  auto targetIt = std::begin(target);
  while (srcIt != srcEnd) {
    if (targetIt == std::end(target)) {
      //insert: src has still items left while target has no more items
      ---
    } else if (lessThan(*srcIt, *targetIt)) { //insert: src has one or more items that need to be inserted into target
      auto srcInsertEnd = std::next(srcIt);
      while (srcInsertEnd != srcEnd && lessThan(*srcInsertEnd, *targetIt)) {
        srcInsertEnd++;
      }
      targetIt = onInsert(srcIt, srcInsertEnd, targetIt);
      //targetIt now points to the item after the item(s) just inserted
      srcIt = srcInsertEnd;
    } else if (lessThan(*targetIt, *srcIt)) { //removal: target has items that are not in src (any more), so remove them
      ---
    } else { //same item, check for changes
      ---
    }
    } else {
      ---
    }
  }
}
```
Can we do better?

Much improved:
- Separate, readable blocks for comparing, detecting changes, inserting, removing and updating
- No loops (well, almost)
- Still a lot of code

Integrate in QAbstractModel

UpdateableModel template (simplified)

Towards a solution

UpdateableModel usage

Towards a solution

Resulting model update code

Towards a solution
**Features**

- Can take any forward-iterable data structure for source data
- Can deal with any storage container that:
  - Is forward-iterable
    - Supplies insert that takes an iterator range, a number of values (and a copy) or a single value. SFINAE is used to select most efficient option.
- Tries to emit as few signals as possible:
  - Inserts and removals in blocks if possible
  - dataChanged in blocks of columns
    - Subject to policy on how to merge rows
- Returns number of operations performed (inserts, removals & updates)

**Limitations**

- Only supports list-like models, no trees
  - Mostly because there are no standard data structures for trees
- Only one set of changed roles per row
- Data needs to be sorted by key

---

**Sorting**

QAIM does:

- Filtering
- Sorting

But how does it deal with changes

- in sort criteria (column or ascending/descending order)?
- in data affecting the sort order?

It uses the `layoutChanged` signal.

- No animations in QML.

[Demo: talks/modelModels/ex-sorting-qsfpml]
SortProxyModel reports changes in the order using row moves:

- for data changes
- for sort criteria changes
- drop-in replacement for QSortFilterProxyModel
  - for the sorting part of its API

Demo: talks/modelModels/ex-sorting-sortproxymodel

Conclusions

- We can solve the issue of changing models in a generic way
  - Just not quite as a one-liner
  - ... but close enough.
- QSortFilterProxyModel does not provide moves
  - We can implement our own though

Source code is available under a liberal license as part of the KDToolBox github repository.

Questions?

Thank you for your time!
Contact us:

- [http://www.kdab.com](http://www.kdab.com)
- info@kdab.com
- training@kdab.com
- andre.somers@kdab.com