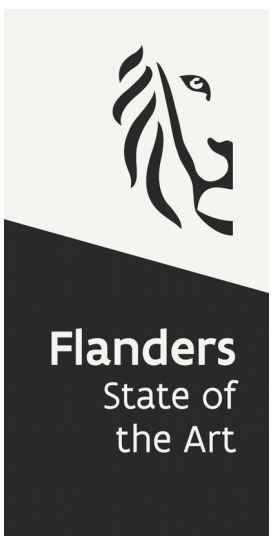
Migrating a C++03 library to C++11 case study: Wt 4



Roel Standaert
September 25, 2017

What is Wt?

- First released in 2006
- Widget-based web framework:
  - Inspired by desktop GUI frameworks
  - Abstracts away underlying web technologies (HTML, JavaScript, Ajax, WebSockets,…), graceful degradation depending on available technologies
  - Qt in particular
  - Use: embedded and as frontend for C++-based applications

Wt (3) application

```
#include <Wt/WApplication>
#include <Wt/WContainerWidget>
#include <Wt/WServer>

Wt::WApplication *createApplication(const Wt::WEnvironment &env)
{
  auto app = new Wt::WApplication(env);
  Wt::WContainerWidget *root = app->root();

  // ...

  return app;
}

int main(int argc, char *argv[])
{
  return Wt::WRun(argc, argv, &createApplication);
}
```

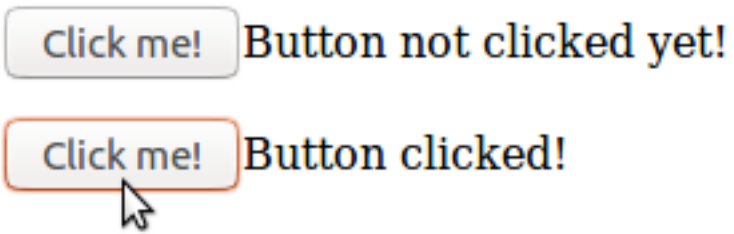
C++

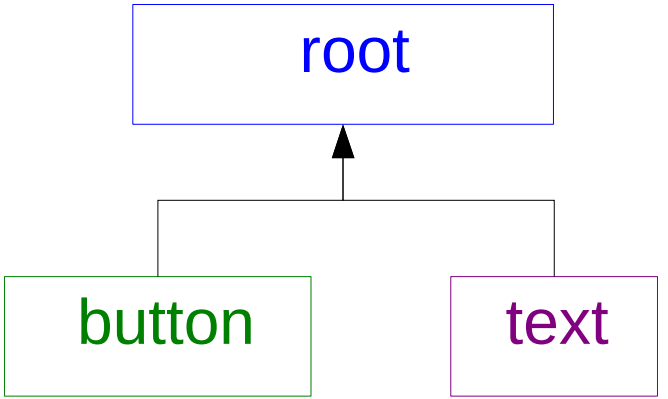
```
Wt::WContainerWidget *root = app->root();

Wt::WPushButton *button =
    new Wt::WPushButton("Click me!");
root->addWidget(button);

Wt::WText *text =
    new Wt::WText("Button not clicked yet!");
root->addWidget(text);

button->clicked().connect(std::bind([text]{
  text->setText("Button clicked!");
}));
```





HTML

```
<div …>
    <button …>
      Click me!
    </button>
    <span …>
      Button not clicked yet!
    </span>
</div>
```

Wt 4 goals

- Make using Wt fun!
  - More clear and safe (especially memory model)
  - Faster compilation
  - Familiar: aligned with modern C++ practices
- Fix problems in API
- But:
  - Keep it consistent
  - Breaking changes should break at compile time

Problem #1: non-obvious memory model

https://stackoverflow.com/questions/46251809/how-does-wt-c-call-delete-for-allocated-objects

"How does Wt C++ call delete for allocated objects"

"When looking at any given Wt C++ example, there are a lot of new calls but how do these even get deleted? […]"

Wt 3: creation and adding of a new widget

Method 1: construct, and explicit addWidget call:

```
Wt::WContainerWidget *container = …

Wt::WPushButton *button = new Wt::WPushButton("Click me!");
// button may leak, not exception safe
container->addWidget(button);
// ownership transferred: not obvious!
```

Method 2: passing container to constructor

```
Wt::WContainerWidget *container = …

Wt::WPushButton *button =
  new Wt::WPushButton("Click me!", container);
// need to be careful in Wt implementation:
// Effective C++ item #9: Never call virtual functions during construction
//                        or destruction!
```

No transfer of ownership?

```
Wt::WContainerWidget *container = …

auto button = std::make_unique<Wt::WPushButton>("Click me!");
container->addWidget(button.get());
// save button until it is no longer needed
```

Pros

- ✔ exception: doesn't leak
- ✔ clear

Cons

- ✗ what if button is deleted?
  - Remove from widget tree: button disappears!
  - Let pointer dangle: undefined behaviour
- ✗ cumbersome
- ✗ Wt 3 code: compiles but leaks!

shared_ptr?

```
Wt::WContainerWidget *container = …

auto button = std::make_shared<Wt::WPushButton>("Click me!");
container->addWidget(button);

button->clicked().connect([button]{
  button->setText("I was clicked!");
});
```

Pros

- ✔ exception: doesn't leak

Cons

- ✗ ownership less clear
- ✗ watch out for memory leaks
- ✗ thread safety unnecessary

Wt 4: same ownership, with unique_ptr

```
Wt::WContainerWidget *container = …

auto button = std::make_unique<Wt::WPushButton>("Click me!");
container->addWidget(std::move(button));
// button == nullptr

auto text = std::make_unique<Wt::WText>("Button not clicked yet!");
container->addWidget(std::move(text));
// text == nullptr

button->clicked().connect([t=text.get()]{
  t->setText("Button clicked!");
});
```

Pros

- ✔ exception: doesn't leak
- ✔ clear

Cons

- ✘ a bit more verbose
- ✘ widget is null after move

Catching use-after-move

```
clang-tidy -checks=misc-use-after-move use-after-move.cpp -- -std=c++14 ...
```

```
1 warning generated.
/home/roel/use-after-move.cpp:22:5: warning: 'button' used after it was moved [
misc-use-after-move]
    button->clicked().connect([t=text.get()]{
    ^
/home/roel/use-after-move.cpp:17:26: note: move occurred here
    container->addWidget(std::move(button));
                         ^
```

```
run-clang-tidy.py -checks=-*,misc-use-after-move
```

addWidget returning Widget*

```
virtual void addWidget(std::unique_ptr<Wt::WWidget> widget);

template<typename Widget>
  Widget *addWidget(std::unique_ptr<Widget> widget)
{
  Widget *result = widget.get();
  addWidget(std::unique_ptr<Wt::WWidget>(std::move(widget)));
  return result;
}
```

addWidget returning Widget*

```
Wt::WContainerWidget *container = ...

Wt::WPushButton *button = container->addWidget(
                 std::make_unique<Wt::WPushButton>("Click me!"));
Wt::WText *text = container->addWidget(
              std::make_unique<Wt::WText>("Button not clicked yet!");

button->clicked().connect([text]{
  text->setText("Button clicked!");
});
```

observing_ptr

- What if the widget emitting the signal outlives widgets used in the slot?
- Wt::Core::observing_ptr
- Every widget is observable (Wt::Core::observable)
- Simple std::vector<Wt::Core::observing_ptr*> implementation
- Optional: usually unnecessary

Replacing boost

- A lot is now part of the STL
- Boost headers slowed compilation
- On Windows: using Boost in public API required distributing binary builds with Boost.

Replacing boost

Wt 3	Wt 4
Boost.Signals2	Own implementation*
Boost.Any	Wt::cpp17::any (included implementation: github.com/thelink2012/any, std::any or std::experimental::any)
Boost.Thread	Mostly STL <thread>
Boost.Regex (Perl-compatible!)	STL <regex> (ECMAScript!)
Boost.Random	STL <random>
Boost.Asio	Boost.Asio *or* standalone
Boost.Date_Time	STL <chrono> + Howard Hinnant's date library

*based on https://testbit.eu/cpp11-signal-system-performance/, but heavily modified

Wt 3: Boost.Signals2 stack trace

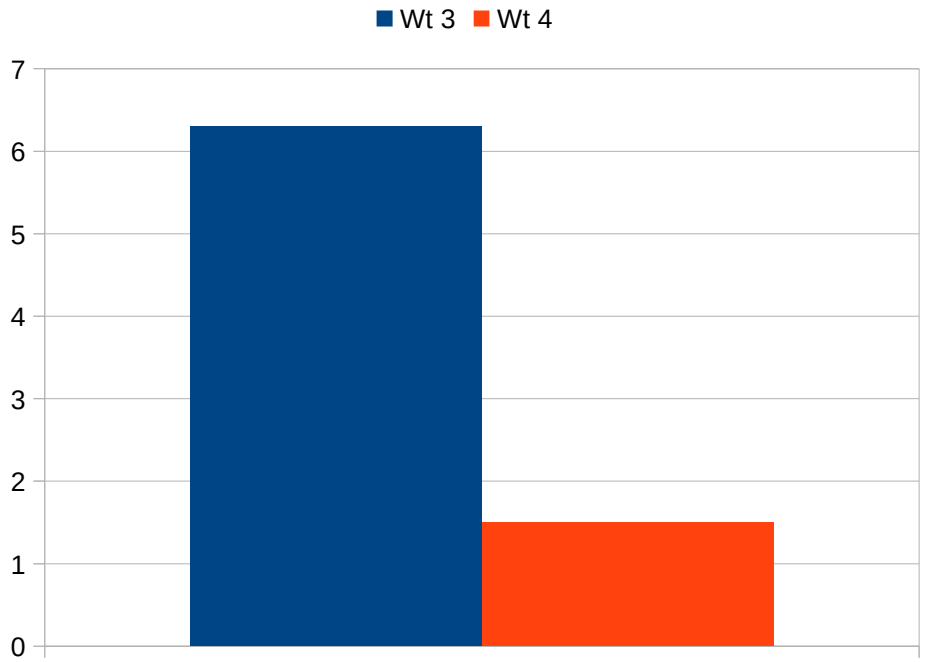
```
#0  std::_Bind<main(int, char**)::<lambda(const Wt::WEnvironment&)>::<lambda()>()>::operator()<Wt::WMouseEvent, void> ...
#1  boost::detail::function::void_function_obj_invoker1<std::_Bind<main(int, char**)::<lambda(const Wt::WEnvironment&)>:: ...
#2  0x00007ffff6bb5215 in boost::function1<void, Wt::WMouseEvent>::operator() (this=0x7fffd400e748, a0=...) at ...
#3  0x00007ffff6bb4708 in boost::signals2::detail::call_with_tuple_args<boost::signals2::detail::void_type>::m_invoke ...
#4  0x00007ffff6bb34f0 in boost::signals2::detail::call_with_tuple_args<boost::signals2::detail::void_type>::operator() ...
#5  0x00007ffff6bb1a8b in boost::signals2::detail::variadic_slot_invoker<boost::signals2::detail::void_type, ...
#6  0x00007ffff6baf6ac in boost::signals2::detail::slot_call_iterator_t<boost::signals2::detail::variadic_slot_invoker ...
#7  0x00007ffff6badb4c in boost::iterators::iterator_core_access::dereference<boost::signals2::detail:: ...
#8  0x00007ffff6bac472 in boost::iterators::detail::iterator_facade_base<boost::signals2::detail::slot_call_iterator_t …
#9  0x00007ffff6baaaa5 in boost::signals2::optional_last_value<void>::operator()<boost::signals2::detail:: ...
#10 0x00007ffff6ba8c7c in boost::signals2::detail::combiner_invoker<void>::operator()<boost::signals2:: ...
#11 0x00007ffff6ba6fd1 in boost::signals2::detail::signal_impl<void (Wt::WMouseEvent), boost::signals2:: ...
#12 0x00007ffff6f0d0bc in boost::signals2::signal<void (Wt::WMouseEvent), boost::signals2::optional_last_value<void>, ...
#13 0x00007ffff6f0b80d in Wt::EventSignal<Wt::WMouseEvent>::processDynamic (this=0x7fffd400d2f0, jse=...) at ...
#14 0x00007ffff72376b2 in Wt::WebSession::processSignal (this=0x7fffd4000940, s=0x7fffd400d2f0, se="", request=..., ...
#15 0x00007ffff72373bf in Wt::WebSession::notifySignal (this=0x7fffd4000940, e=...) at ../../src/web/WebSession.C:3011
#16 0x00007ffff7231ce5 in Wt::WebSession::notify (this=0x7fffd4000940, event=...) at ../../src/web/WebSession.C:2559
#17 0x00007ffff6ab3332 in Wt::WApplication::notify (this=0x7fffd4005460, e=...) at ../../src/Wt/WApplication.C:1510
#18 0x00007ffff7229f79 in Wt::WebSession::handleRequest (this=0x7fffd4000940, handler=...) at ...
#19 0x00007ffff72097ff in Wt::WebController::handleRequest (this=0x660810, request=0x7fffcc001ba0) at ...
```

Wt 4 stack trace

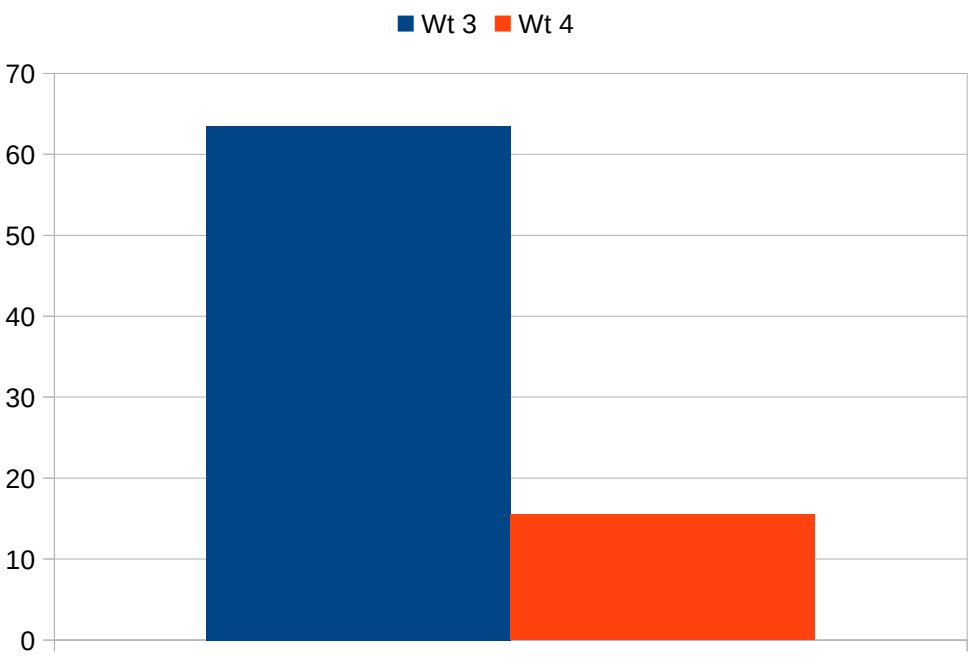
```
#0  <lambda(const Wt::WEnvironment&)>::<lambda()>::operator()(void) const (__closure=0x7fffcc008330) at ...
#1  0x000000000040dca5 in std::_Function_handler<void(), main(int, char**)::<lambda(const Wt::WEnvironment&)>:: ...
#2  0x000000000040fc0a in std::function<void ()>::operator()() const (this=0x7fffcc008330) ...
#3  0x000000000040f961 in Wt::Signals::Impl::ConnectHelper<0, Wt::WMouseEvent>::connect(Wt::Signals::Signal ...
#4  0x00000000004101ee in std::_Function_handler<void (Wt::WMouseEvent), Wt::Signals::Impl::ConnectHelper<0, ...
#5  0x00007ffff75dbec1 in std::function<void (Wt::WMouseEvent)>::operator()(Wt::WMouseEvent) const ...
#6  0x00007ffff75dc67c in Wt::Signals::Impl::ProtoSignal<Wt::WMouseEvent>::emit (this=<optimized out>, args#0=...) ...
#7  0x00007ffff77a7113 in Wt::EventSignal<Wt::WMouseEvent>::processDynamic (this=0x7fffcc008f80, jse=...) ...
#8  0x00007ffff7968dff in Wt::WebSession::processSignal (this=this@entry=0x7fffdc001bf0, s=0x7fffcc008f80, se="", ...
#10 0x00007ffff797728e in Wt::WebSession::notify (this=0x7fffdc001bf0, event=...) at ../../src/web/WebSession.C:2543
#11 0x00007ffff7578cca in Wt::WApplication::notify (this=<optimized out>, e=...) at ../../src/Wt/WApplication.C:1480
#12 0x00007ffff796f728 in Wt::WebSession::handleRequest (this=0x7fffdc001bf0, handler=...) at ../../src/web/WebSession.C:1692
#13 0x00007ffff795c71f in Wt::WebController::handleRequest (this=0x663fd0, request=<optimized out>) at ...
```

Eliminating Boost includes

Preprocessed file size (MiB)

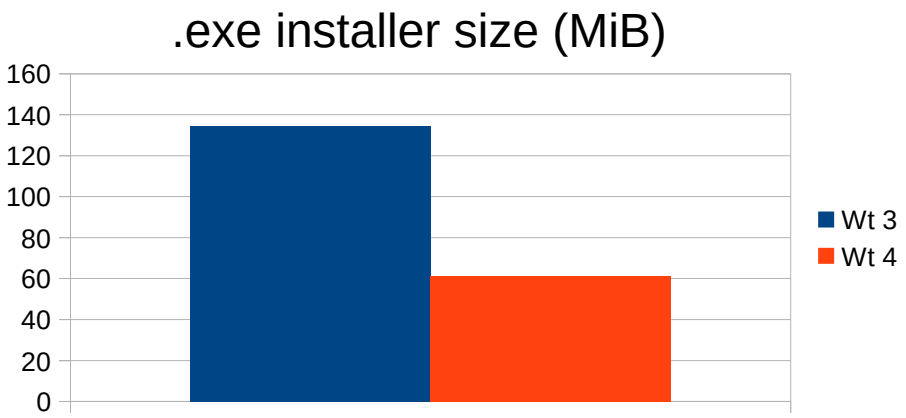


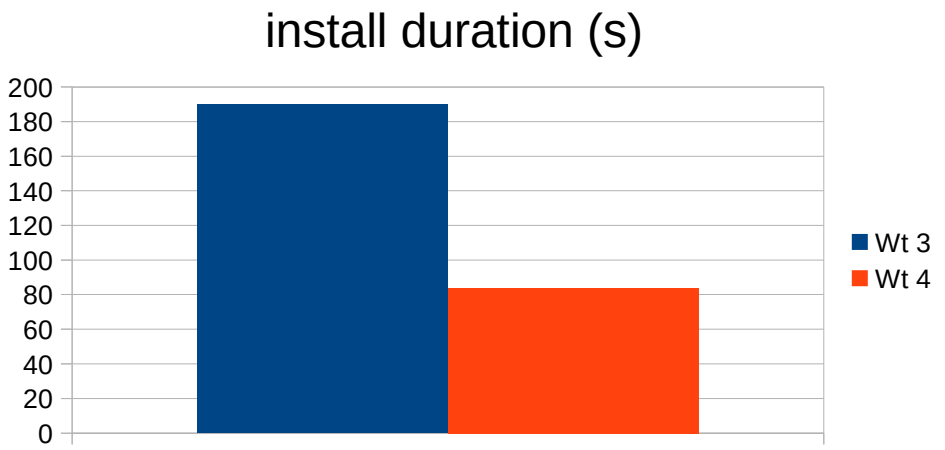
Compilation time (ms, hot cache)

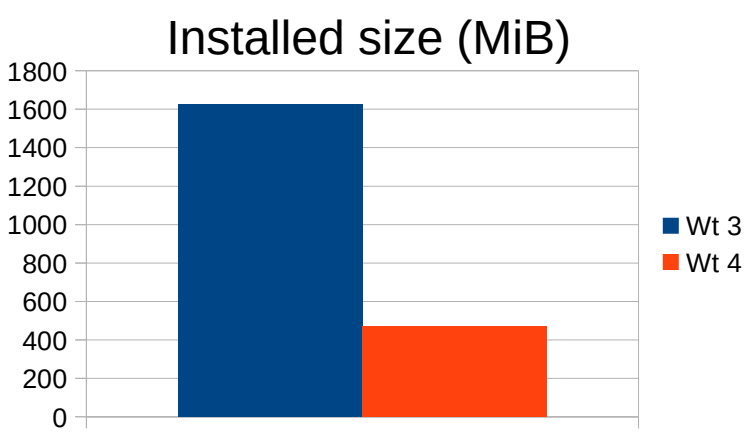


GCC 5.4.0, average of 1000 compiles, -O3 -std=c++14, without linking on running example

Windows installer size reduction







Conclusions

- Explicit ownership is nice, not straightforward to adapt our ownership model to it
- Additions to STL are a great opportunity to shed some (compilation) weight
- Wt 4 released last week: verdict still out on user response

Get Wt 4 at https://www.webtoolkit.eu

Questions?