Write Event-based programs again sequentially or how to Clean Code in asynchronous programs.

Helge Betzinger
CTO
pcvisit Software AG
Agenda

- What is the problem and how to escape?
- coasync4cpp let you program TODAY without callbacks!
- Where to go from here?
- No more Callbacks!
What is the problem and how to escape?

A typical requirement for an application these days...

If the user clicks the button, then replace the image within his clipboard by a URL with a copy of this image within the cloud.
What is the problem and how to escape?

A typical requirement for an application these days...

If the user clicks the button, then replace the image within his clipboard by a URL with a copy of this image within the cloud.

please wait for the next slide clicking won’t make it come any faster
What is the problem and how to escape?

A typical requirement for an application these days...

If the user clicks the button, then replace the image within his clipboard by a URL with a copy of this image within the cloud.

The UI must stay responsive all the time.
Async becoming the norm!
Example: Concurrent waiting with signals

```cpp
void MainView::uploadImageFromFile(const QString &filePath)
{
    QJsonObject object;
    // configure object ...
    EnginioReply *reply = mModel->append(object);
    connect(reply, &EnginioReply::finished, this, &MainView::beginUpload);
}

void MainView::beginUpload(EnginioReply *reply) {
    reply->deleteLater();
    // use result/reply here ..
}
```

1) Manage the control flow of the application
2) Manage resources of the infrastructure
3) Business logic related code
Example: Concurrent waiting with futures

C++11

```cpp
File saveCliprdToDisk();

std::future<File> f = std::async(saveCliprdToDisk);

f.get(); // this blocks, until saveCliprdToDisk is done!
// even the destructor of std::future blocks!
```
Example: Concurrent waiting with boost

C++ standard proposal N3558, Boost.Thread 1.55.0

```cpp
boost::future<File> f = boost::async(saveCliprdToDisk);

f.then([] (boost::future<File> savedF) {
    // use result.get() here ...
    uploadImage(savedF.get()).then(
        [=] (future<Reply> uploadedFile) {
            requestUrl(uploadedFile.get()).then(
                ...);
        });
});
```
And what about Clean Code?
Developer Days 2014

And what about Clean Code?
... how to escape?

Document number: N3721
Date: 2013-08-30
Reply-to: Niklas Gustafsson <niklas.gustafsson@microsoft.com>
Artur Laksberg <arturl@microsoft.com>
Herb Sutter <hsutter@microsoft.com>
Sana Mithani <sanam@microsoft.com>

Improvements to std::future<T> and Related APIs
coasync4cpp let you do asynchronous programming without callbacks TODAY!
std::future<File> f = std::async(saveCliprdToDisk);

File f = f.get(); // this blocks, until saveCliprdToDisk is done!
Coasync4cpp - How it works

```cpp
std::future<File> f = std::async(saveCliprdToDisk);

File f = f.get(); // this blocks, until saveCliprdToDisk is done!
```

```cpp
File f = Task(boost::async(saveCliprdToDisk));
```

```cpp
File f = await Task(boost::async(saveCliprdToDisk));
```

```cpp
File f = await boost::async(saveCliprdToDisk);
```
Task<...> Wrap around a awaitable to make code simpler Allows to use Task/await within a routine
await

Unwraps value of a given awaitable without blocking your thread
Understanding async Tasks

```cpp
bindAsTask(void Button_Click()) {
    QUrl url = await clip2UrlAsync();
    url2clip(url);
}

Task<QUrl> clip2UrlAsync () {
    ... return Task<QUrl>();
}
```

Message Pump with TaskDispatcher

Click

Url available..
Example using await

```cpp
Button.connect( bindAsTask( &MainView::convertIntoUrl, this ) );

File saveCliprdToDisk();
QNetworkReply * uploadImage( File );
QNetworkReply * requestUrl( QNetworkReply * );
void put2clipboard( QUrl );

void convertIntoUrl() {
    File tmpFile = await boost::async( saveCliprdToDisk() );
    QNetworkReply * uploadedFile = await uploadImage( tmpFile );
    QNetworkReply * fileUrl = await( requestUrl, uploadedFile );
    put2clipboard( fileUrl->result() );
}
```
Example using Task

```cpp
Button.connect( bindAsTask( &MainView::convertIntoUrl, this ) );

Task<File> saveCliprdToDiskAsync();
Task<QNetworkReply * > uploadImageAsync( File );
Task<QUrl> requestUrlAsync(QNetworkReply * );
void put2clipboard(QUrl);

void convertIntoUrl() {
    auto tmpFile = saveCliprdToDiskAsync();
    auto uploadedFile = uploadImageAsync( tmpFile );
    auto fileUrl = requestUrlAsync( uploadedFile );
    put2clipboard(fileUrl);
}
Task Factories creates from methods.

Awaitables awaits.

Task Dispatcher empowers.
**make_task**

Creates an `Task<R>` from anything, that is callable

Starts the method immediately

**bindAsTask**

Creates an `std::function< Task<R> (...)>` from anything, that is callable

Start the method later, with invocation of the function object
taskify

auto taskify( method, placeholders::CALLBACK, Args...) -> Task< std::tuple< P... >> ;

Starts the method immediately
Transforms the callback into an awaitable Task

Returns a Task with a std::tuple, containing the parameters of the CALLBACK.
method can be anything, that is callable
CALLBACK must be a function object.
placeholders::EXCEPTION also supported
Task<...>
boost::future<R>

Operation is already running

await directly
Store and await later
Create a Task from it and get result or await later
Helper: TaskDispatcher

TaskDispatcher4StdThread
TaskDispatcher4QtThread
ThreadWithTasks

Creates an dispatcher for Tasks within current thread or creates a new thread with a dispatcher in it

Prerequisite to get Task<> working within a particallary thread!
Summary Usage

1. Instantiate suitable TaskDispatcher in your thread

2. Call async method as Task, using a Task Factory

3. Use await/Task with any Awaitable within this method
Example using Task

Button.connect( bindAsTask( &MainView::convertIntoUrl, this ));

Task<File> saveCliprdToDiskAsync();
Task<QNetworkReply * > uploadImageAsync( File );
Task<QUrl> requestUrlAsync(QNetworkReply * );
void put2clipboard(QUrl);

void convertIntoUrl() {
    auto tmpFile = saveCliprdToDiskAsync();
    auto uploadedFile = uploadImageAsync( tmpFile );
    auto fileUrl = requestUrlAsync( uploadedFile );
    put2clipboard(fileUrl);
}
coasync4cpp makes consuming async APIs simple
Where to go from here?
Where to go from here?

Play around with testcoasync4cpp and testcoasync4qt to understand

https://github.com/helgebetzinger/coasync4cpp
coasync4cpp

coasync4cpp

coroutine; threading

boost

depends on

Googletest

Google C++ Testing Framework

depends on

testcoasync4cpp

testcoasync4qt

utilize

test

test
What can you expect from version 0.10?

Simple integration with legacy code

https://github.com/helgebetzinger/coasync4cpp
What can you expect from version 0.10?

More Awaitables
- QFuture
- QNetworkReply
- EnginioReply

More Task Factories
- taskifyQtSignal

More Msg-Dispatchers

https://github.com/helgebetzinger/coasync4cpp
What can you expect from version 0.10?

Extended build support

clang, cmake

https://github.com/helgebetzinger/coasync4cpp
Watch the project and stay tuned

Comment and report issues and requirements

Contribute added features or fixed bugs

coasync4cpp@pcvisit.com
https://github.com/helgebetzinger/coasync4cpp
No more callbacks!

Questions?

coasync4cpp@pcvisit.com
https://github.com/helgebetzinger/coasync4cpp
What is the problem and how to escape?

coasync4cpp@pcvisit.com
https://github.com/helgebetzinger/coasync4cpp
Best Practices for App-developers
Future themes

Using it with legacy code
Extension Points (Awaitables, TaskDispatcher)
Best Practices
Interplay between sync and async code
Async API
Exception
Subscribe the project on github
Comment on feature request or bugs (instead of voting ;-)

Exceptions
Cannot await top level
Maximize parallelism for I/O bound work
Library methods should not lie
If your async void method has side effects, return Task<void> anyway
Convert Signals into Tasks
Take care of your locks!
Is it CPU Bound or I/O Bound?
Task Factories

make_task
taskify
bindAsTask

Creates an Task from anything, that is callable, a callback, event or signal.

Starts the method immediately or later

Adds an separate stack to your routine
make_task

“makes your method asynchronous” lets you put awaits and Tasks in it
bind2current
bind2thread
Example: Concurrent waiting with QFutureWatcher

```cpp
File saveCliprdToDisk();

QFuture<File> qfuture = QtConcurrent::run(saveCliprdToDisk);

auto watcher = new QFutureWatcher<File>();

QObject::connect( watcher, &QFutureWatcherBase::finished,
                  [=] {
                    // use watcher->result() here ...
                    watcher->deleteLater();
                  });

watcher->setFuture(qfuture);
```
Task<R>
boost::future<R>

Operation is already running

await directly
Store and await later
Create a Task from it and get result or await later
Awaitables

QFuture*
QNetworkReply* (impl. using taskifyQtSignal)
EnginioReply* (impl. using taskifyQtSignal)

Operation is already running

await directly
Store and await later
Create a Task from it and get result or await later
Task Factories

**taskifyQtSignal**

```cpp
auto taskifyQtSignal( R(Args...), obj ) -> Task< std::tuple< Args... >>;
```

*Starts an task immediatelly or later explicit*

Returns a Task with a `std::tuple`, containing the parameters of the signal.
Requirements design coasync4cpp library

- Solve the problem!
- Applicable on Legacy Code / Brownfield Code
- Preferably Compatible with upcoming C++ Developments C++1xx
- don’t hide the interfaces of used future implementation to prevent lock out of existing tools around them
- Enhancements points for smooth integration with other libraries, as Qt
What is the problem and how to escape?

cōasync4cpp@pcvisit.com
https://github.com/helgebetzinger/coasync4cpp