Developer
Days
2014

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

Helge Betzinger
CTO
pcvisit Software AG

Agenda

Developer Days 2014

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

pcvisit

What is the problem and how to escape?

A typical requirement for a application these days…

If the user clicks the button, than 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 a application these days...

If the user clicks the button, than 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

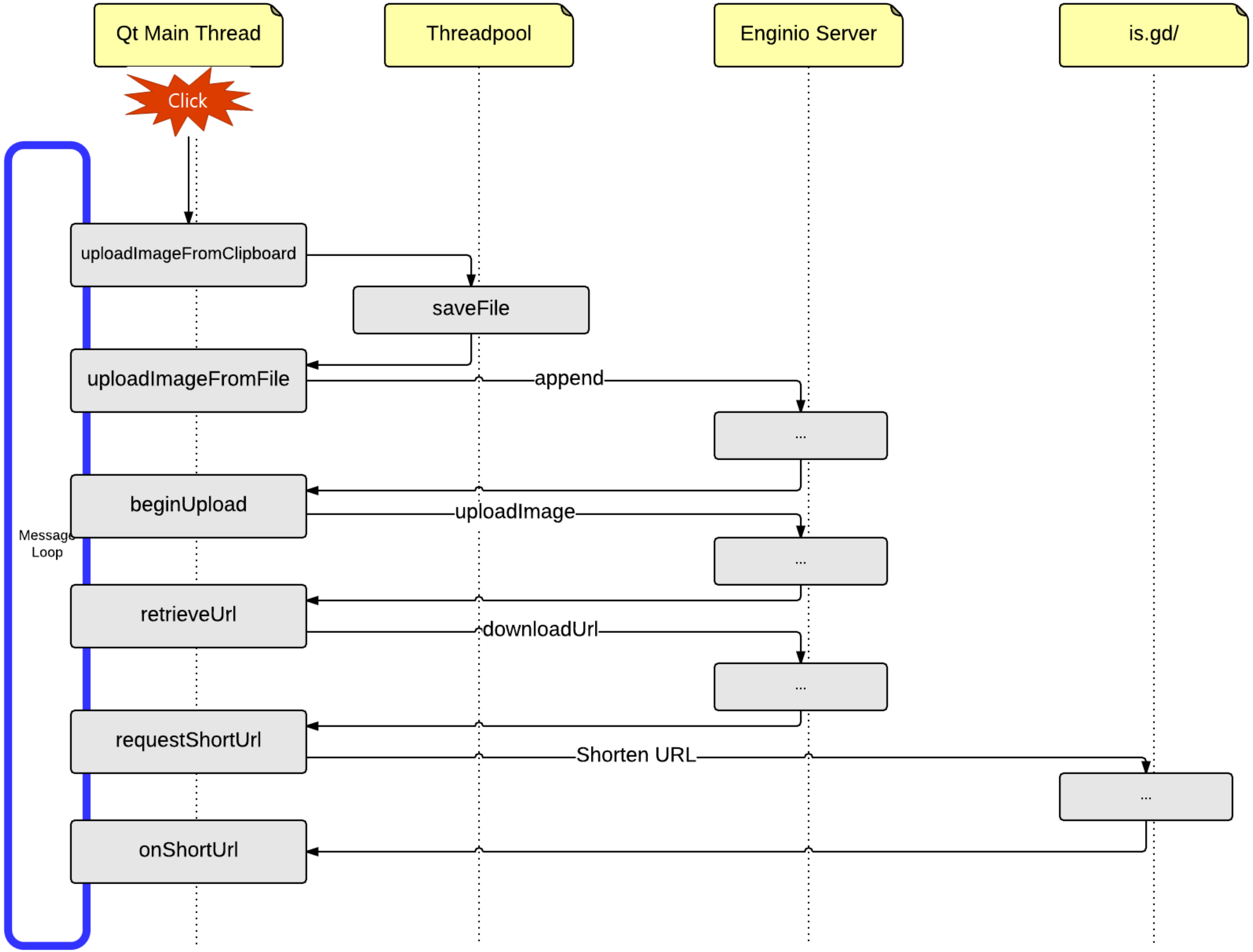
What is the problem and how to escape?

A typical requirement for a application these days...

If the user clicks the button, than 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!


Qt Main Thread
Threadpool
Enginio Server
is.gd/
Click
Message Loop
uploadImageFromClipboard
saveFile
uploadImageFromFile
append
...
beginUpload
uploadImage
...
retrieveUrl
downloadUrl
...
requestShortUrl
Shorten URL
...
onShortUrl

Example: Concurrent waiting with signals

```
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

```
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

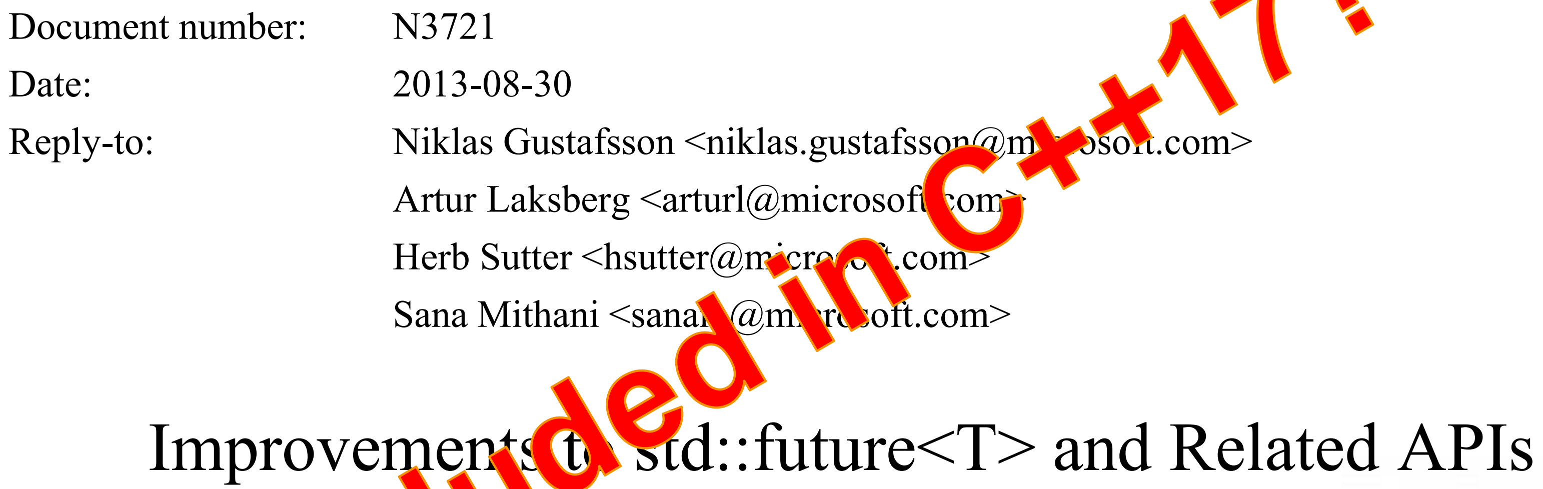
C++ standard proposal N3558, Boost.Thread 1.55.0

```
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?

"THY WILL BE DONE."

… 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

included in C++17?

pcvisit

coasync4cpp
let you do asynchronous programming without callbacks
TODAY!

Coasync4cpp - How it works

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

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

Coasync4cpp - How it works

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

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

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

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

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

Overview coasync4cpp

Task<...>

Wrap around a awaitable to make code simpler

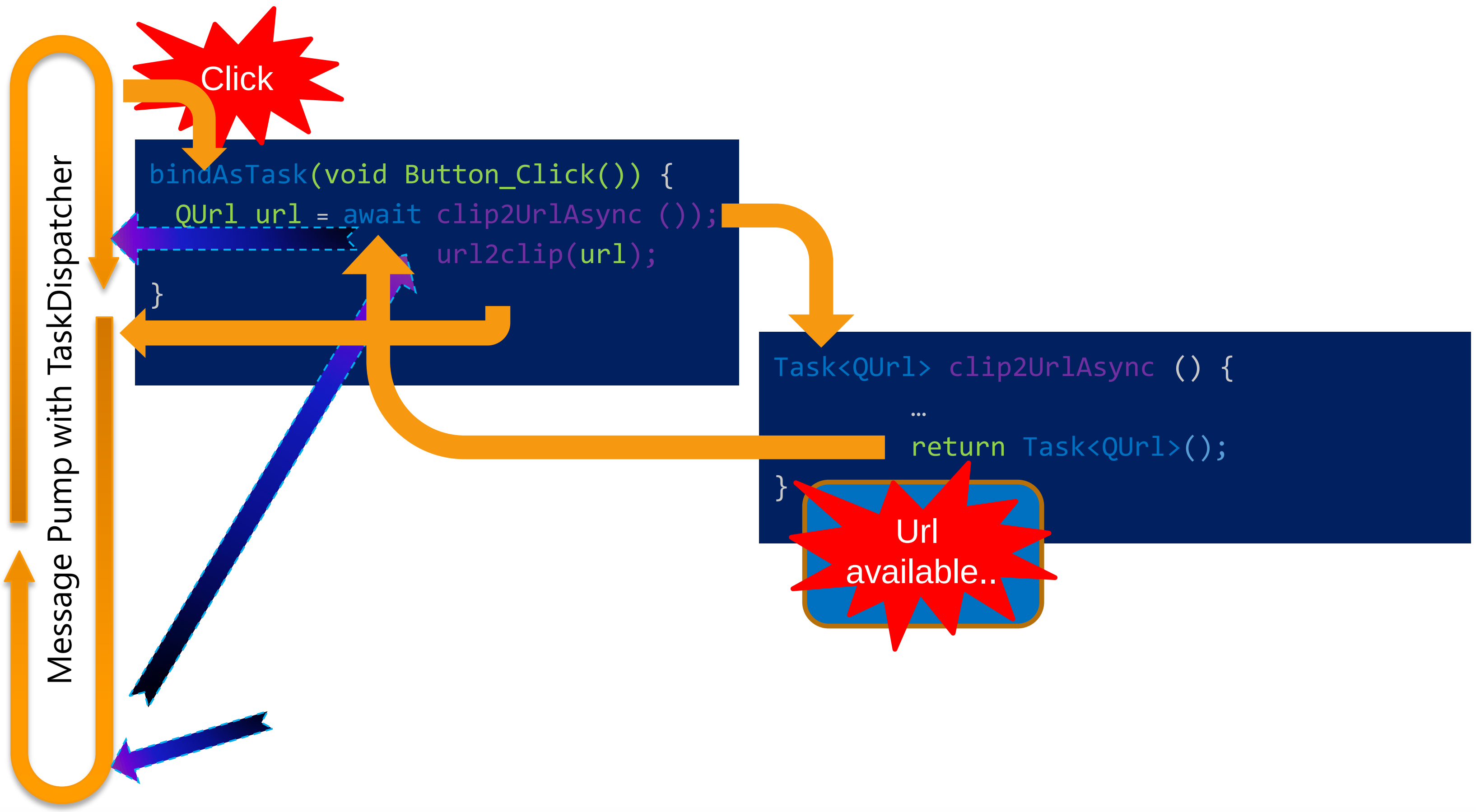
Allows to use Task/await within a routine

Overview coasync4cpp

await

Unwraps value of a given awaitable without blocking your thread

Understanding async Tasks



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

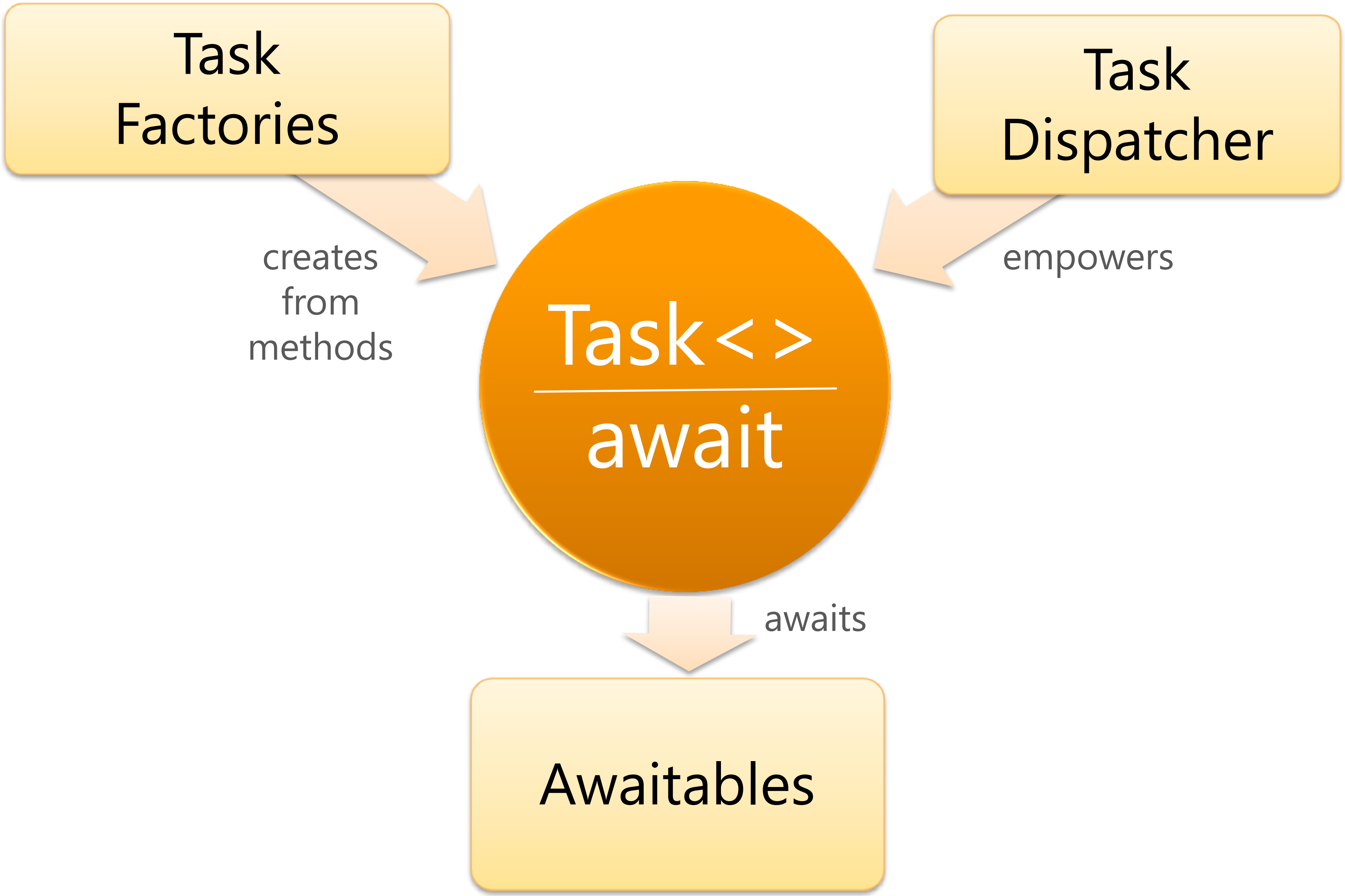
```
Task<QUrl> clip2UrlAsync () {
        …
        return Task<QUrl>();
}
```

Example using await

```
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

```
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
Task Dispatcher
empowers
Task< >
await
awaits
Awaitables

Task Factories

Developer Days 2014

make_task

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

Starts the method immediatelly

bindAsTask

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

Start the method later, with invocation of the function object

pcvisit

taskify

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

Starts the method immediatelly
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

Awaitables

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. Instanciate 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

Developer Days 2014

```
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);
}
```

pcvisit

coasync4cpp
makes consuming async
APIs simple

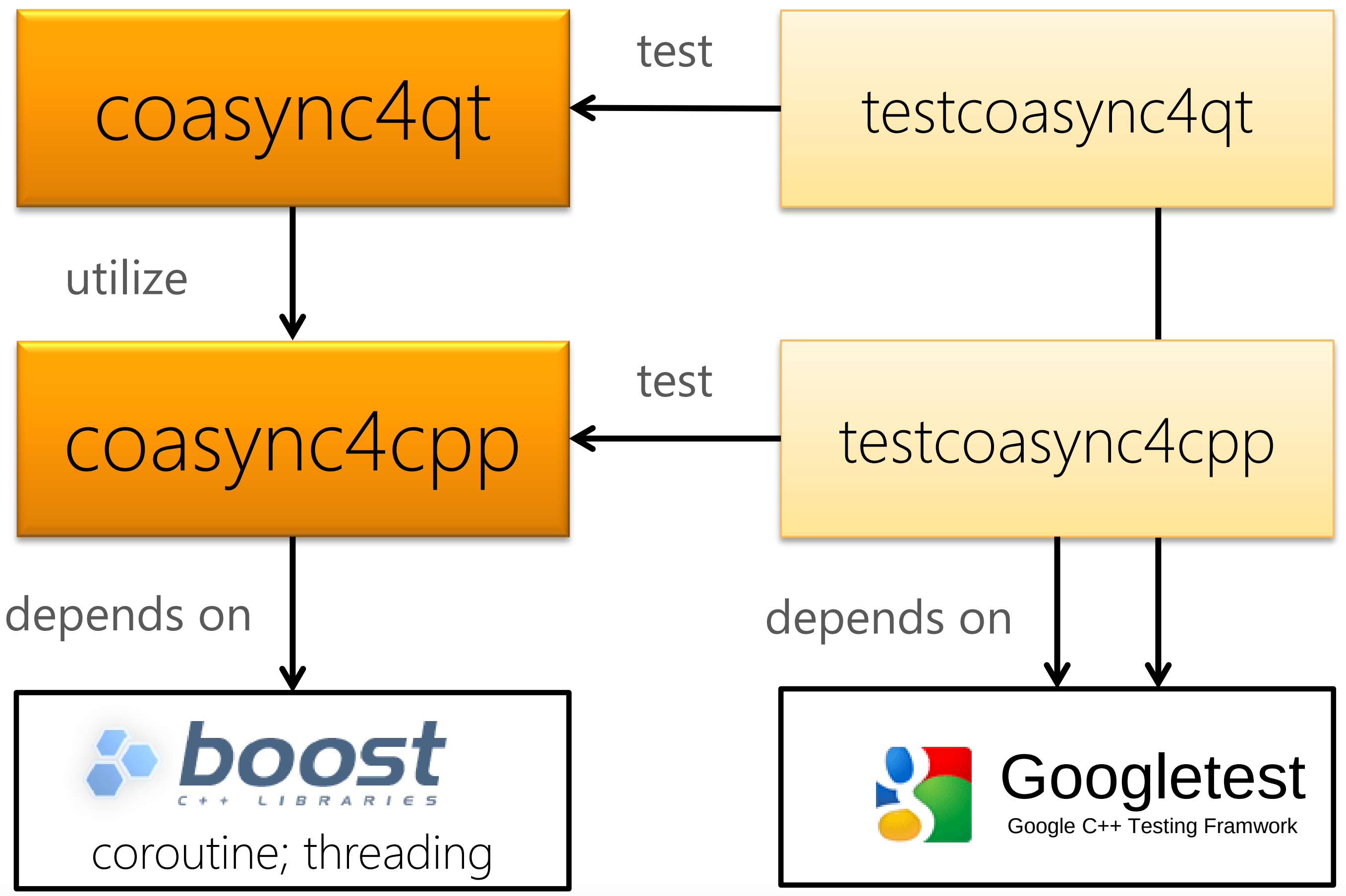
Where to go from here?

Where to go from here?

Play around with testcoasync4cpp and testcoasync4qt to understand

https://github.com/helgebetzinger/coasync4cpp

coasync4cpp



https://github.com/helgebetzinger/coasync4cpp

What can you expect from version 0.10?

Simple integration with legacy code

What can you expect from version 0.10?

More Awaitables

More Task Factories

QFuture*

QNetworkReply*

EnginioReply*

taskifyQtSignal

More Msg-Dispatchers

What can you expect from version 0.10?

Extended build support

clang, cmake

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

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

Best Practices for App-developers

Future themes

Developer Days 2014

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 ;-)

pcvisit

Exceptions

Exceptions

Best Practices

Cannot await top level

Best Practices

Maximize parallelism for I/O bound work

Library methods should not lie

Best Practices

If your async void method has side effects, return Task<void> anyway

Convert Signals into Tasks

Best Practices

Take care
of your locks!

Is it CPU Bound or I/O Bound?

Archive

Task Factories

make_task
taskify
bindAsTask

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

Starts the method immediatelly or later

Adds an separate stack to your routine

Overview coasync4cpp

Developer Days 2014

make_task

“makes your method asynchronous”
lets you put awaits and Tasks in it

pcvisit

Helpers

bind2current

bind2thread

Example: Concurrent waiting with QFutureWatcher

```
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);
```

Awaitables

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

```
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.

coasync4cpp

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

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