C++ Montréal

Gabriel
Aubut-Lussier

Druide

Développement web avec

C++

Web Development

2017-03-22

Librairies Libraries

- Beast
- CppRestSDK (Casablanca)
- Proxygen

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓
TLS	TLS	✓	✓	✓
HTTP 2.0	HTTP 2.0	✓		
WebSockets	WebSockets		½	✓
JSON	JSON		✓	

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓
TLS	TLS	✓	✓	✓
HTTP 2.0	HTTP 2.0	✓		
WebSockets	WebSockets		½	✓
JSON	JSON		✓	
Bien documenté	Well documented		✓	✓
License	Licence	BSD	MIT	Boost
Linux	Linux	✓	✓	✓
macOS	macOS		✓	✓
Windows	Windows		✓	✓

⬇️ Client

⬇️ CppRestSDK

```
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
using namespace utility;                    // Common utilities like string conversions
using namespace web;                         // Common features like URIs.
using namespace web::http;                   // Common HTTP functionality
using namespace web::http::client;           // HTTP client features
using namespace concurrency::streams;        // Asynchronous streams

int main(int argc, char* argv[]) {
   auto fileStream = std::make_shared<ostream>();
   pplx::task<void> requestTask = fstream::open_ostream(U(« results.html"))
   .then([=](ostream outFile) {
      *fileStream = outFile;
      http_client client(U("http://isocpp.org/"));
      return client.request(methods::GET);
   })
   .then([=](http_response response) {
      printf("Received response status code:%u\n", response.status_code());
      return response.body().read_to_end(fileStream->streambuf());
   })
   // Close the file stream.
   .then([=](size_t) {
      return fileStream->close();
   });
   try {
      requestTask.wait();
   } catch (const std::exception &e) {
      printf("Error exception:%s\n", e.what());
   }
}
```

Proxygen (1/2)

```
#include <iostream>
#include <folly/SocketAddress.h>
#include <folly/io/async/EventBase.h>
#include <proxygen/lib/http/HTTPConnector.h>
#include <proxygen/lib/http/HTTPMessage.h>
#include <proxygen/lib/http/codec/HTTP2Codec.h>
#include <proxygen/lib/http/session/HTTPTransaction.h>
#include <proxygen/lib/http/session/HTTPUpstreamSession.h>
#include <proxygen/lib/utils/URL.h>
using namespace folly;
using namespace proxygen;
using namespace std;
class CurlClient : public HTTPConnector::Callback, public HTTPTransactionHandler {
public:
  CurlClient(EventBase* evb, const URL& url) : evb_(evb), url_(url) {}
  ~CurlClient() override {}

  // HTTPConnector methods
  void connectSuccess(HTTPUpstreamSession* session) override;
  void connectError(const AsyncSocketException& ex) override {LOG(ERROR) << "Coudln't connect to " << url_.getHostAndPort() << ":" << ex.what();}

  // HTTPTransactionHandler methods
  void setTransaction(HTTPTransaction* txn) noexcept override {}
  void detachTransaction() noexcept override {}
  void onHeadersComplete(unique_ptr<HTTPMessage> msg) noexcept override;
  void onBody(unique_ptr<IOBuf> chain) noexcept override;
  void onTrailers(unique_ptr<HTTPHeaders> trailers) noexcept override {LOG(INFO) << "Discarding trailers";}
  void onEOM() noexcept override {LOG(INFO) << "Got EOM";}
  void onUpgrade(UpgradeProtocol protocol) noexcept override {LOG(INFO) << "Discarding upgrade protocol";}
  void onError(const HTTPException& error) noexcept override {LOG(ERROR) << "An error occurred: " << error.what();}
  void onEgressPaused() noexcept override {LOG(INFO) << "Egress paused";}
  void onEgressResumed() noexcept override {LOG(INFO) << "Egress resumed";}

protected:
  EventBase* evb_{nullptr};
  URL url_;
};
```

⬇️ Proxygen (2/2)

```
void CurlClient::connectSuccess(HTTPUpstreamSession* session) {
  HTTPTransaction* txn = session->newTransaction(this);
  HTTPMessage request;
  request.setMethod(HTTPMethod::GET);
  request.setHTTPVersion(1, 1);
  request.setURL(url_.makeRelativeURL());
  HTTP2Codec::requestUpgrade(request);
  auto& headers = request.getHeaders();
  headers.add(HTTP_HEADER_USER_AGENT, "proxygen_curl");
  headers.add(HTTP_HEADER_HOST, url_.getHostAndPort());
  headers.add("Accept", "*/*");
  txn->sendHeaders(request);
  txn->sendEOM();
  session->closeWhenIdle();
}
void CurlClient::onHeadersComplete(unique_ptr<HTTPMessage> msg) noexcept {
  cout << msg->getStatusCode() << " " << msg->getStatusMessage() << endl;
  msg->getHeaders().forEach([&](const string& header, const string& val) { cout << header << ": " << val << endl; });
}
void CurlClient::onBody(unique_ptr<IOBuf> chain) noexcept {
  if (chain) {
    const IOBuf* p = chain.get();
    do { cout.write((const char*)p->data(), p->length()); p = p->next(); } while (p != chain.get());
  }
}
int main(int argc, char* argv[]) {
  EventBase evb;
  URL url("http://isocpp.org");
  CurlClient curlClient(&evb, url);
  SocketAddress addr(url.getHost(), url.getPort(), true);
  HHWheelTimer::UniquePtr timer{HHWheelTimer::newTimer(&evb, chrono::milliseconds(HHWheelTimer::DEFAULT_TICK_INTERVAL),
AsyncTimeout::InternalEnum::NORMAL, chrono::milliseconds(5000))};
  HTTPConnector connector(&curlClient, timer.get());
  connector.connect(&evb, addr);
  evb.loop();
}
```

⬇️ Beast

```
#include <beast/http.hpp>
#include <boost/asio.hpp>
#include <boost/lexical_cast.hpp>
#include <iostream>
#include <string>

int main()
{
    std::string const host = "isocpp.org";
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver r{ios};
    boost::asio::ip::tcp::socket sock{ios};
    boost::asio::connect(sock, r.resolve(boost::asio::ip::tcp::resolver::query{host, "http"}));

    beast::http::request<beast::http::empty_body> req;
    req.method = "GET";
    req.url = "/";
    req.version = 11;
    req.fields.replace("Host", host + ":" + boost::lexical_cast<std::string>(sock.remote_endpoint().port()));
    req.fields.replace("User-Agent", "Beast");
    beast::http::prepare(req);
    beast::http::write(sock, req);

    beast::streambuf sb;
    beast::http::response<beast::http::streambuf_body> resp;
    beast::http::read(sock, sb, resp);
    std::cout << resp;
}
```

Serveur ⬆️ Server

DISCLAIMER : The code in the next few slides is sample code from every concerned project thinned down to the essentials and adapted for the presentation. Please refer to the project licences :

CppRestSDK : https://github.com/Microsoft/cpprestsdk/blob/master/license.txt (MIT)
Proxygen : https://github.com/facebook/proxygen/blob/master/LICENSE (BSD 3-clauses)
Beast : https://github.com/vinniefalco/Beast/blob/master/LICENSE_1_0.txt (Boost)

⬆️ CppRestSDK

```
#include <cpprest/http_listener.h>
#include <cpprest/uri.h>
#include <thread>
#include <chrono>
using namespace web::http::experimental::listener;
using namespace web::http;
using namespace web;
using namespace std;

void handle_get(http_request request) {
  request.reply(status_codes::OK);
}

int main() {
  http_listener listener{http::uri("http://localhost:8080/")};
  listener.support(methods::GET, handle_get);
  listener.open().wait();
  while(true) {
    this_thread::sleep_for(chrono::milliseconds(2000));
  }
  listener.close();
}
```

⬆️ Proxygen (1/2)

```
#include <folly/Memory.h>
#include <folly/io/async/EventBaseManager.h>
#include <proxygen/httpserver/HTTPServer.h>
#include <proxygen/httpserver/RequestHandler.h>
#include <proxygen/httpserver/RequestHandlerFactory.h>
#include <proxygen/httpserver/ResponseBuilder.h>
#include <unistd.h>
using namespace proxygen;
using namespace folly;

class EchoHandler : public RequestHandler {
 public:
  explicit EchoHandler() {}
  void onRequest(std::unique_ptr<HTTPMessage> headers) noexcept override {}
  void onBody(std::unique_ptr<IOBuf> body) noexcept override {}
  void onEOM() noexcept override;
  void onUpgrade(UpgradeProtocol proto) noexcept override {}
  void requestComplete() noexcept override;
  void onError(ProxygenError err) noexcept override;
};

class EchoHandlerFactory : public RequestHandlerFactory {
 public:
  RequestHandler* onRequest(RequestHandler*, HTTPMessage*) noexcept override {
    return new EchoHandler();
  }
  void onServerStart(EventBase* evb) noexcept {}
  void onServerStop() noexcept {}
};
void EchoHandler::onEOM() noexcept {
  ResponseBuilder(downstream_)
    .status(200, "OK")
    .sendWithEOM();
}

void EchoHandler::requestComplete() noexcept {
  delete this;
}

void EchoHandler::onError(ProxygenError err) noexcept {
  delete this;
}
```

⬆️ Proxygen (2/2)

```
int main() {
  std::vector<HTTPServer::IPConfig> IPs = {
    {SocketAddress("127.0.0.1", 11000, true /*allowNameLookup*/), HTTPServer::Protocol::HTTP},
    {SocketAddress("127.0.0.1", 11001, true), HTTPServer::Protocol::SPDY},
    {SocketAddress("127.0.0.1", 11002, true), HTTPServer::Protocol::HTTP2},
  };

  HTTPServerOptions options;
  options.threads = static_cast<size_t>(sysconf(_SC_NPROCESSORS_ONLN));
  options.shutdownOn = {SIGINT, SIGTERM};
  options.handlerFactories = RequestHandlerChain().addThen<EchoHandlerFactory>().build();
  options.h2cEnabled = true;

  HTTPServer server(std::move(options));
  server.bind(IPs);

  // Start HTTPServer mainloop in a separate thread
  std::thread t([&] () {
    server.start();
  });

  t.join();
}
```

⬆️ Beast (1/4)

```
#include <beast/http.hpp>
#include <beast/core/handler_helpers.hpp>
#include <beast/core/handler_ptr.hpp>
#include <beast/core/placeholders.hpp>
#include <beast/core/streambuf.hpp>
#include <beast/test/sig_wait.hpp>
#include <boost/asio.hpp>
#include <cstddef>
#include <cstdio>
#include <iostream>
#include <memory>
#include <thread>
#include <utility>
namespace beast { namespace http {
class http_async_server {
    using endpoint_type = boost::asio::ip::tcp::endpoint;
    using address_type = boost::asio::ip::address;
    using socket_type = boost::asio::ip::tcp::socket;

    using req_type = request<string_body>;
    using resp_type = response<empty_body>;

    boost::asio::io_service ios_;
    boost::asio::ip::tcp::acceptor acceptor_;
    socket_type sock_;
    std::vector<std::thread> thread_;
public:
    http_async_server(endpoint_type const& ep)
        : acceptor_(ios_), sock_(ios_) {
        acceptor_.open(ep.protocol());
        acceptor_.bind(ep);
        acceptor_.listen(boost::asio::socket_base::max_connections);
        acceptor_.async_accept(sock_,
            std::bind(&http_async_server::on_accept, this, beast::asio::placeholders::error));
        size_t threads = sysconf(_SC_NPROCESSORS_ONLN);
        thread_.reserve(threads);
        for(std::size_t i = 0; i < threads; ++i)
            thread_.emplace_back([&] { ios_.run(); });
    }
```

⬆️ Beast (2/4)

```
    ~http_async_server() {
        error_code ec;
        ios_.dispatch([&]{ acceptor_.close(ec); });
        for(auto& t : thread_)
            t.join();
    }
private:
    template<class Stream, class Handler, bool isRequest, class Body, class Fields>
    class write_op {
        struct data {
            bool cont;
            Stream& s;
            message<isRequest, Body, Fields> m;

            data(Handler& handler, Stream& s_, message<isRequest, Body, Fields>&& m_)
                : cont(beast_asio_helpers::is_continuation(handler)), s(s_), m(std::move(m_))
            {}
        };
        handler_ptr<data, Handler> d_;
    public:
        write_op(write_op&&) = default;
        write_op(write_op const&) = default;
        template<class DeducedHandler, class... Args>
        write_op(DeducedHandler&& h, Stream& s, Args&&... args) : d_(std::forward<DeducedHandler>(h), s, std::forward<Args>(args)...) {
            (*this)(error_code{}, false);
        }
        void operator()(error_code ec, bool again = true) {
            auto& d = *d_;
            d.cont = d.cont || again;
            if(! again) {
                beast::http::async_write(d.s, d.m, std::move(*this));
                return;
            }
            d_.invoke(ec);
        }
        friend void* asio_handler_allocate(
            std::size_t size, write_op* op) {
            return beast_asio_helpers::allocate(size, op->d_.handler());
        }

        friend void asio_handler_deallocate(void* p, std::size_t size, write_op* op) {
            return beast_asio_helpers::deallocate(p, size, op->d_.handler());
        }
```

⬆️ Beast (3/4)

```
    friend bool asio_handler_is_continuation(write_op* op) {
        return op->d_->cont;
    }
    template<class Function>
    friend void asio_handler_invoke(Function&& f, write_op* op) {
        return beast_asio_helpers::invoke(f, op->d_.handler());
    }
};
template<class Stream, bool isRequest, class Body, class Fields, class DeducedHandler>
static void async_write(Stream& stream, message<isRequest, Body, Fields>&& msg, DeducedHandler&& handler) {
    write_op<Stream, typename std::decay<DeducedHandler>::type, isRequest, Body, Fields>{
         std::forward<DeducedHandler>(handler), stream, std::move(msg)
    };
}
class peer : public std::enable_shared_from_this<peer> {
    int id_;
    streambuf sb_;
    socket_type sock_;
    http_async_server& server_;
    boost::asio::io_service::strand strand_;
    req_type r
public:
    peer(peer&&) = default;
    peer(peer const&) = default;
    peer& operator=(peer&&) = delete;
    peer& operator=(peer const&) = delete;
    peer(socket_type&& sock, http_async_server& server) : sock_(std::move(sock)), server_(server), strand_(sock_.get_io_service()) {
        static int n = 0;
        id_ = ++n;
    }
    void fail(error_code ec, std::string what) {
        if(ec != boost::asio::error::operation_aborted)
            cout << "#" << id_ << " " << what << ": " << ec.message() << '\n';
    }
    void run() {
        do_read();
    }
    void do_read() {
        async_read(sock_, sb_, req_, strand_.wrap(std::bind(&peer::on_read, shared_from_this(), asio::placeholders::error)));
    }
```

⬆️ Beast (4/4)

```
        void on_read(error_code const& ec) {
            if(ec)
                return fail(ec, "read");
            resp_type res;
            res.status = 200;
            res.reason = "OK";
            res.version = req_.version;
            res.fields.insert("Server", "http_async_server");
            prepare(res);
            async_write(sock_, std::move(res), std::bind(&peer::on_write, shared_from_this(), asio::placeholders::error));
        }
        void on_write(error_code ec) {
            if(ec)
                fail(ec, "write");
            do_read();
        }
    };
    void fail(error_code ec, std::string what) {
        cout << what << ": " << ec.message() << '\n';
    }
    void on_accept(error_code ec) {
        if(! acceptor_.is_open())
            return;
        if(ec)
            return fail(ec, "accept");
        socket_type sock(std::move(sock_));
        acceptor_.async_accept(sock_, std::bind(&http_async_server::on_accept, this, asio::placeholders::error));
        std::make_shared<peer>(std::move(sock), *this)->run();
    }
};
} } // http // beast

int main()
{
    using namespace beast::http;
    using endpoint_type = boost::asio::ip::tcp::endpoint;
    using address_type = boost::asio::ip::address;
    endpoint_type ep{address_type::from_string("127.0.0.1"), 8080};
    http_async_server server(ep);
    beast::test::sig_wait();
}
```

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓
TLS	TLS	✓	✓	✓
HTTP 2.0	HTTP 2.0	✓		
WebSockets	WebSockets		½	✓
JSON	JSON		✓	
Bien documenté	Well documented		✓	✓
License	Licence	BSD	MIT	Boost
Linux	Linux	✓	✓	✓
macOS	macOS		✓	✓
Windows	Windows		✓	✓

⁉️ All libraries are very efficient. Comparative measurements aren't provided in this talk.

⁉️ Toutes les librairies sont très efficaces. Un comparatif n'est pas fourni avec cette présentation.

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓
TLS	TLS	✓	✓	✓
HTTP 2.0	HTTP 2.0	✓		
WebSockets	WebSockets		½	✓
JSON	JSON		✓	
Bien documenté	Well documented		✓	✓
License	Licence	BSD	MIT	Boost
Linux	Linux	✓	✓	✓
macOS	macOS		✓	✓
Windows	Windows		✓	✓
Facile d'utilisation	Ease of use		✓	

⁉️ All libraries are very efficient. Comparative measurements aren't provided in this talk.

⁉️ Toutes les librairies sont très efficaces. Un comparatif n'est pas fourni avec cette présentation.

Comparatif Comparison

		Proxygen	CppRestSDK	Beast
Soutien corporatif	Corporate support	Facebook	Microsoft	Ripple
Stable	Production ready	✓	Bêta	Bêta
Client HTTP	HTTP Client	✓	✓	✓
Serveur HTTP	HTTP Server	✓	✓	✓
TLS	TLS	✓	✓	✓
HTTP 2.0	HTTP 2.0	✓		
WebSockets	WebSockets		½	✓
JSON	JSON		✓	
Bien documenté	Well documented		✓	✓
License	Licence	BSD	MIT	Boost
Linux	Linux	✓	✓	✓
macOS	macOS		✓	✓
Windows	Windows		✓	✓
Facile d'utilisation	Ease of use		✓	
Performant	Performance	⁉	⁉	⁉

⁉ All libraries are very efficient. Comparative measurements aren't provided in this talk.

⁉ Toutes les librairies sont très efficaces. Un comparatif n'est pas fourni avec cette présentation.

JSON

- CppRestSDK
- Nlohmann
- RapidJSON

⬇️⬆️ CppRestSDK

json::value : parse(…), serialize(…), is_null(), is_boolean(), is_number(), is_integer(), is_double(), is_string(), is_array(), is_object()

Object : has_field(…), operator[], as_object()

Array : at(…), operator[], as_array()

Number : as_number(), as_integer(), as_double()

String : as_string()

Bool : as_bool()

⬇️⬆️ Nlohmann/JSON

json : parse(…), dump(…), operator>>, operator<<, type(), is_null(), is_boolean(), is_number(), is_string(), is_array(), is_object(), _json

Object : find(…), begin(), end(), rbegin(), rend(), operator[…], operator=, initializer_list, push_back(…), emplace_back(…), operator +=

Implicit conversions with STL containers (vector, deque, list, forward_list, array, set, unordered_set, multiset, unordered_multiset, map, unordered_map, multimap, unordered_multimap)

⬇️ RapidJSON

Document, **Value**, **Value::ConstMemberIterator**

Document : ParseStream(…), ParseInSitu(…), Parse(…), HasParseError()

Value : GetType(), IsNull(), IsBool(), IsNumber(), IsString(), IsArray(), IsObject()

Object : MemberBegin(), MemberEnd(), FindMember(…), HasMember(…)

Array : Size(), operator[], Begin(), End()

Number : GetInt(), GetUInt(), GetInt64(), GetUInt64(), GetDouble(), GetFloat()

String : GetString(), GetStringLength()

Bool : GetBool()

⬆️ RapidJSON

Document, **Document::AllocatorType**, **Value, StringBuffer, Writer**

Document : GetAllocator(), Accept(Writer<StringBuffer>)

Value : SetNull(), SetBool(…), SetInt(…), SetUint(…), SetInt64(…), SetUint64(…), SetDouble(…), SetFloat(…), SetString(…), SetArray(), SetObject(), Move()

Object : AddMember(…), RemoveMember(…), EraseMember(…), RemoveAllMembers()

Array : Reserve(…), PushBack(…), Erase(…)

Comparatif Comparison

		RapidJSON	Nlohmann	CppRestSDK
Stable	Production ready	✓	✓	✓
RFC~~4627~~7159 JSON	RFC~~4627~~7159 JSON	✓	✓	✓
RFC6901 JSON Pointer	RFC6901 JSON Pointer	✓	✓	

Comparatif Comparison

		RapidJSON	Nlohmann	CppRestSDK
Stable	Production ready	✓	✓	✓
RFC~~4627~~7159 JSON	RFC~~4627~~7159 JSON	✓	✓	✓
RFC6901 JSON Pointer	RFC6901 JSON Pointer	✓	✓	
Unicode	Unicode	✓	✓	✓
SAX	SAX	✓		
CBOR, MessagePack	CBOR, MessagePack		✓	
SIMD	SIMD	✓		

Comparatif Comparison

		RapidJSON	Nlohmann	CppRestSDK
Stable	Production ready	✓	✓	✓
RFC~~4627~~7159 JSON	RFC~~4627~~7159 JSON	✓	✓	✓
RFC6901 JSON Pointer	RFC6901 JSON Pointer	✓	✓	
Unicode	Unicode	✓	✓	✓
SAX	SAX	✓		
CBOR, MessagePack	CBOR, MessagePack		✓	
SIMD	SIMD	✓		
Bien documenté	Well documented	✓	✓	
License	Licence	MIT	MIT	MIT
Linux	Linux	✓	✓	✓
macOS	macOS	✓	✓	✓
Windows	Windows	✓	✓	✓

Comparatif Comparison

		RapidJSON	Nlohmann	CppRestSDK
Stable	Production ready	✓	✓	✓
RFC~~4627~~7159 JSON	RFC~~4627~~7159 JSON	✓	✓	✓
RFC6901 JSON Pointer	RFC6901 JSON Pointer	✓	✓	
Unicode	Unicode	✓	✓	✓
SAX	SAX	✓		
CBOR, MessagePack	CBOR, MessagePack		✓	
SIMD	SIMD	✓		
Bien documenté	Well documented	✓	✓	
License	Licence	MIT	MIT	MIT
Linux	Linux	✓	✓	✓
macOS	macOS	✓	✓	✓
Windows	Windows	✓	✓	✓
Facile d’utilisation	Ease of use		✓	✓

Comparatif Comparison

		RapidJSON	Nlohmann	CppRestSDK
Stable	Production ready	✓	✓	✓
RFC~~4627~~7159 JSON	RFC~~4627~~7159 JSON	✓	✓	✓
RFC6901 JSON Pointer	RFC6901 JSON Pointer	✓	✓	
Unicode	Unicode	✓	✓	✓
SAX	SAX	✓		
CBOR, MessagePack	CBOR, MessagePack		✓	
SIMD	SIMD	✓		
Bien documenté	Well documented	✓	✓	
License	Licence	MIT	MIT	MIT
Linux	Linux	✓	✓	✓
macOS	macOS	✓	✓	✓
Windows	Windows	✓	✓	✓
Facile d'utilisation	Ease of use		✓	✓
Performant	Performance	✓		

Tests unitaires

- Codes d’erreur pour les paramètres manquants
- Paramètres optionnels
- Structure attendue des réponses et type des données

Unit testing

- Error codes for missing mandatory parameters
- Optional parameters
- Expected response structure and field types

RapidJSON et GoogleTest

```
TEST(RestAPITests, TestReponseJSON)
{
  auto json = /* httpRequest */;
  auto itField1 = json.FindMember("field1");
  auto itEnd = json.MemberEnd();
  ASSERT_NE(itField1, itEnd);
  const auto& field1 = itField1->value;
  ASSERT_TRUE(field1.IsString());
  EXPECT_GT(field1.GetStringLength(), 0);

  auto itField2 = json.FindMember("field2");
  ASSERT_NE(itField2, itEnd);
  const auto& field2 = itField2->value;
  ASSERT_TRUE(field2.IsArray());
  EXPECT_GT(field2.Size(), 0);
  for (int i = 0, n = field2.Size(); i < n; ++i) {
    const auto& element = field2[i];
    ASSERT_TRUE(element.IsNumber());
    EXPECT_GT(element.GetInt(), 10);
  }
}
```

Performance

- E/S asynchrones, par événements
- Programmation réseau sans copies
- Exécuteur avec bassin de fils d'exécution

- Asynchronous, event-based IO
- Zero-copy network programming
- Thread pool based executors

Kernel

- File Descriptor limit (ulimit -n; /etc/security/limits.conf)
- net.ipv4.ip_local_port_range Ephemeral ports
- net.ipv4.tcp_tw_reuse
- net.ipv4.tcp_max_syn_backlog

WA

WEBASSEMBLY

Historique History

Dates	Historique	History
2011-10	JSConf.eu 2011: Présentation sur Emscripten par Alon Zakai (kripken)	JSConf.eu 2011: Emscripten talk by Alon Zakai (kripken)
2012-11	Emscripten v1.0.1	Emscripten v1.0.1
2013-03	asm.js	asm.js
2013-07	Premières optimisations pour asm.js dans le navigateur Firefox	Firefox as first web browser to implement asm.js-specific optimizations

Historique ## History

Dates	Historique	History
2011-10	JSConf.eu 2011: Présentation sur Emscripten par Alon Zakai (kripken)	JSConf.eu 2011: Emscripten talk by Alon Zakai (kripken)
2012-11	Emscripten v1.0.1	Emscripten v1.0.1
2013-03	asm.js	asm.js
2013-07	Premières optimisations pour asm.js dans le navigateur Firefox	Firefox as first web browser to implement asm.js-specific optimizations
2015-04	Création d'un groupe ouvert sur WebAssembly au W3C	W3C WebAssembly Community Group started
2015-06	Première annonce publique du groupe ouvert sur WebAssembly	The first public announcement of the WebAssembly Community Group
2016-03	Définition du coeur de fonctionnalités minimales à implémenter	Definition of core feature with multiple interoperable implementations
2016-10	Annonce des premiers navigateurs offrant un aperçu de la technologie	Browser Preview announced with multiple implementations

Historique History

Dates	Historique	History
2011-10	JSConf.eu 2011: Présentation sur Emscripten par Alon Zakai (kripken)	JSConf.eu 2011: Emscripten talk by Alon Zakai (kripken)
2012-11	Emscripten v1.0.1	Emscripten v1.0.1
2013-03	asm.js	asm.js
2013-07	Premières optimisations pour asm.js dans le navigateur Firefox	Firefox as first web browser to implement asm.js-specific optimizations
2015-04	Création d'un groupe ouvert sur WebAssembly au W3C	W3C WebAssembly Community Group started
2015-06	Première annonce publique du groupe ouvert sur WebAssembly	The first public announcement of the WebAssembly Community Group
2016-03	Définition du coeur de fonctionnalités minimales à implémenter	Definition of core feature with multiple interoperable implementations
2016-10	Annonce des premiers navigateurs offrant un aperçu de la technologie	Browser Preview announced with multiple implementations
2017-02	Sélection du logo officiel	Official logo chosen
2017-03	Consensus multi-navigateur et fin de la période démonstration	Cross-browser consensus and end of Browser Preview

```
#include <boost/icl/interval_set.hpp>
#include <emscripten/bind.h>
#include <sstream>
using namespace emscripten;
class Intervals {
public:
	using IntervalType = boost::icl::interval_set<int>;
public:
	void AddInterval(int start, int end);
	void RemoveInterval(int start, int end);
	std::string AsString() const;
private:
	boost::icl::interval_set<int> intervals;
};
void Intervals::AddInterval(int start, int end) {
	assert(start < end);
	intervals.add(IntervalType::segment_type{start, end});
}
void Intervals::RemoveInterval(int start, int end) {
	assert(start < end);
	intervals.subtract(IntervalType::segment_type{start, end});
}
std::string Intervals::AsString() const {
	std::stringstream result;
	for (const auto& segment : intervals)
		result << '[' << segment.lower() << ',' << ' ' << segment.upper() << ')' << ',' << ' ';
	return result.str();
}
EMSCRIPTEN_BINDINGS(TestMod) {
	class_<Intervals>("Intervals")
		.constructor<>()
		.function("AddInterval", &Intervals::AddInterval)
		.function("RemoveInterval", &Intervals::RemoveInterval)
		.function("AsString", &Intervals::AsString);
}
```

```
#include <boost/icl/interval_set.hpp>
#include <emscripten/bind.h>
#include <sstream>
using namespace emscripten;
class Intervals {
public:
	using IntervalType = boost::icl::interval_set<int>;
public:
	void AddInterval(int start, int end);
	void RemoveInterval(int start, int end);
	std::string AsString() const;
private:
	boost::icl::interval_set<int> intervals;
};
```

```
em++ --bind -I/usr/include/
boost_1_63_0 -std=c++14 intervals.cpp
-s WASM=1 -o Intervals.html
```

```
	std::stringstream result;
	for (const auto& segment : intervals)
		result << '[' << segment.lower() << ',' << ' ' << segment.upper() << ')' << ',' << ' ';
	return result.str();
}
EMSCRIPTEN_BINDINGS(TestMod) {
	class_<Intervals>("Intervals")
		.constructor<>()
		.function("AddInterval", &Intervals::AddInterval)
		.function("RemoveInterval", &Intervals::RemoveInterval)
		.function("AsString", &Intervals::AsString);
}
```

Démo

Ressources Resources

- https://github.com/vinniefalco/Beast
- https://github.com/Microsoft/cpprestsdk
- https://github.com/facebook/proxygen
- https://github.com/nlohmann/json
- https://github.com/miloyip/rapidjson
- https://github.com/miloyip/nativejson-benchmark
- https://github.com/juj/emsdk

Merci !

Gabriel Aubut-Lussier
gaubut@druide.com
@Gab_AL_