**Полезные ссылки**

https://github.com/AndsonYe/MultipartEncoder/blob/master/parser\_test.cpp

http://www.atakansarioglu.com/easy-quick-start-cplusplus-rest-client-example-cpprest-tutorial/

https://mariusbancila.ro/blog/2017/11/19/revisited-full-fledged-client-server-example-with-c-rest-sdk-2-10/

https://github.com/hide1202/cpprest-sample/blob/master/src/ServiceHandler.cpp

**Сервер**

*Request string:*

Если надо распарсить через request string от клиента (например <http://localhost:12345/restdemo?increasingFlag=1>)

Далее можно получить данные из строки:

auto **req\_map** = *web*::*uri*::*split\_query*(**req**.*request\_uri*().*query*());

Затем можно перебрать элементы запроса:

for (const auto& **currParams** : **req\_map**)

{

*std*::*wcout* << "Request key = " << *std*::*wstring*(**currParams**.*first*.*c\_str*()) << " ; " <<

"Request value = " << *std*::*wstring*(**currParams**.*second*.*c\_str*()) << *std*::*endl*;

}

И соответственно обратиться к значению как **req\_map**[*U*("increasingFlag")]

Если захотим сравнить значение, тогда можно сделать так

if (**req\_map**[*U*("increasingFlag")] == *U*("1"))

*Json:*

Если клиент посылает json в теле запроса, тогда

void handle\_del(const *http\_request*& **req**)

{

//body

/\*

{

"removingFlag": true

}

\*/

*web*::*json*::*value* **temp**;

// extracts the request content into a json

**req**.*extract\_json*().

*then*([&**temp**](*pplx*::*task*<*web*::*json*::*value*> **task**)

{

**temp** = **task**.*get*();

}).*wait*();

if (**temp**.*at*(*U*("removingFlag")).*is\_boolean*())

{

bool **creationFlag** = **temp**.*at*(*U*("removingFlag")).*as\_bool*();

**nCounter** = 0;

*web*::*json*::*value* **answer**;

*utility*::*stringstream\_t* **ss**;

**ss** << **nCounter**;

**answer**[L"result"] = *web*::*json*::*value*::*string*(*utility*::*conversions*::*to\_utf16string*(**ss**.*str*()));

**answer**[L"result\_string"] = *web*::*json*::*value*::*string*(*U*("Successful"));

**req**.*reply*(*status\_codes*::*OK*, **answer**);

}

**req**.*reply*(*status\_codes*::*BadRequest*);

}

Пример функции POST

void handle\_post(*http\_request* **req**)

{

//body

/\*

{

"creationFlag": true

}

\*/

*web*::*json*::*value* **temp**;

// extracts the request content into a json

**req**.*extract\_json*().

*then*([&**temp**](*pplx*::*task*<*web*::*json*::*value*> **task**)

{

**temp** = **task**.*get*();

}).*wait*();

if (**temp**.*at*(*U*("creationFlag")).*is\_boolean*())

{

bool **creationFlag** = **temp**.*at*(*U*("creationFlag")).*as\_bool*();

**nCounter** = 0;

*web*::*json*::*value* **answer**;

*utility*::*stringstream\_t* **ss**;

**ss** << **nCounter**;

**answer**[L"result"] = *web*::*json*::*value*::*string*(*utility*::*conversions*::*to\_utf16string*(**ss**.*str*()));

**answer**[L"result\_string"] = *web*::*json*::*value*::*string*(*U*("Successful"));

**req**.*reply*(*status\_codes*::*OK*, **answer**);

}

/\*PS

Can be other records

auto answer = web::json::value::object();

answer[L"foo-list"][L"bar"] = web::json::value::string(utility::conversions::to\_utf16string("value1"));

answer[L"foo-list"][L"bob"] = web::json::value::string(utility::conversions::to\_utf16string("value2"));

answer[L"foo-list"][L"bobList"][0] = web::json::value::string(utility::conversions::to\_utf16string("bobValue1"));

answer[L"foo-list"][L"bobList"][1] = web::json::value::string(utility::conversions::to\_utf16string("bobValue1"));

answer[L"foo-list"][L"bobList"][2] = web::json::value::string(utility::conversions::to\_utf16string("bobValue1"));\*/

**req**.*reply*(*status\_codes*::*BadRequest*);

}

**Клиент**

*Request string:*

Чтобы собрать request string можно воспользоваться объектом *web*::*uri\_builder* **builder**;

*method* **mtd =** *methods*::*GET, methods*::*HEAD, methods*::*PUT.*

**client**.*request*(**mtd**, **builder**.*append\_path*(L"/restdemo").*append\_query*(L"increasingFlag", L"1").*to\_string*());

Пример для функции

void make\_request(*http\_client*& **client**, *method* **mtd**, *json*::*value* const& **jValue**)

{

make\_task\_request(**client**, **mtd**, **jValue**).*then*([](*http\_response* **response**)

{

if (**response**.*status\_code*() == *status\_codes*::*OK*)

{

return **response**.*extract\_json*();

}

return *pplx*::*task\_from\_result*(*json*::*value*());

}).*then*([](*pplx*::*task*<*json*::*value*> **previousTask**)

{

try

{

display\_json(**previousTask**.*get*(), L"R: ");

}

catch (*http\_exception* const& **e**)

{

*std*::*wcout* << **e**.*what*() << *std*::*endl*;

}

}).*wait*();

}