

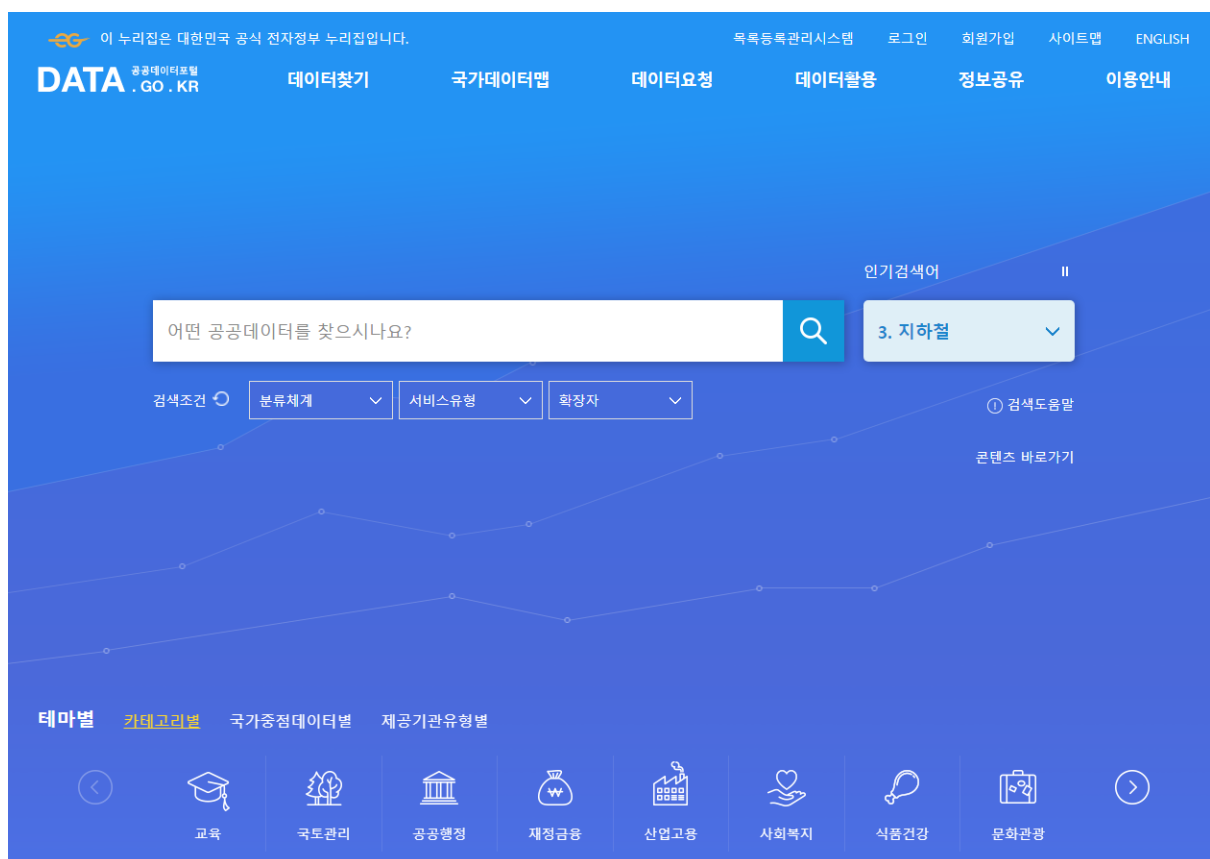
(9기 스타트캠프) C++ 사용한 REST API

🕒 Created	@2023년 1월 4일 오후 2:06
🕒 Last Edited Time	@2023년 1월 10일 오전 11:09
📄 Type	
📄 Status	
👤 Created By	
👤 Last Edited By	
👥 Stakeholders	
📅 날짜	

강사님들께서는, API 키 발급에 시간이 꽤 소요되므로, 아침에 미리 학생들 지시해 신청해두시기 바랍니다.

공공 API 키 발급

공공데이터포털(<https://data.go.kr/>)에 접속. 회원 가입하고 로그인하자.



검색어: 보건복지부 코로나19 확진자 성별 연령별 현황

전체(33,387건)

파일데이터(29,603건)

오픈 API(3,784건)

표준데이터셋3개(0건)

정확도순

10개씩

정렬

오픈 API (3,784건)

보건의료

국가행정기관

국가중점

미리보기

XML

JSON

보건복지부_코로나19 확진자 성별 연령별 현황

보건복지부 코로나19 확진자 성별 연령별 현황에 대한 데이터로 코로나19 감염현황을 성별 및 연령별로 조회하는 기능을 제공합니다.

제공기관 보건복지부 수정일 2022-05-04 조회수 5453 활용신청 401 키워드 보건의료,코로나19,확진자

활용신청

오픈 API 탭 클릭 후, 활용신청 버튼 클릭.

신청 후, 10분 정도 소요된다.

개발계정

<div>신청 0건 ></div> <div>신청중인 단계</div> <div> <div>· 보류</div> <div>0건</div> </div> <div> <div>· 반려</div> <div>0건</div> </div>	<div>활용 1건 ></div> <div>승인되어 활용중인 단계</div> <div> <div>· 변경신청</div> <div>0건</div> </div>	<div>중지0건 ></div> <div>중지신청하여 운영이 중지된 단계</div>
--	--	---

상세검색 열기

총1건

폐기된 목록 포함 보기

OFF

보건의료

보건복지부

활용신청

[승인] 보건복지부_코로나19 확진자 성별 연령별 현황

신청일 2023-01-04 만료예정일 2025-01-04

1

다음과 같이 신청에서 활용으로 넘어가면, 발급 받은 키를 확인할 수 있다.

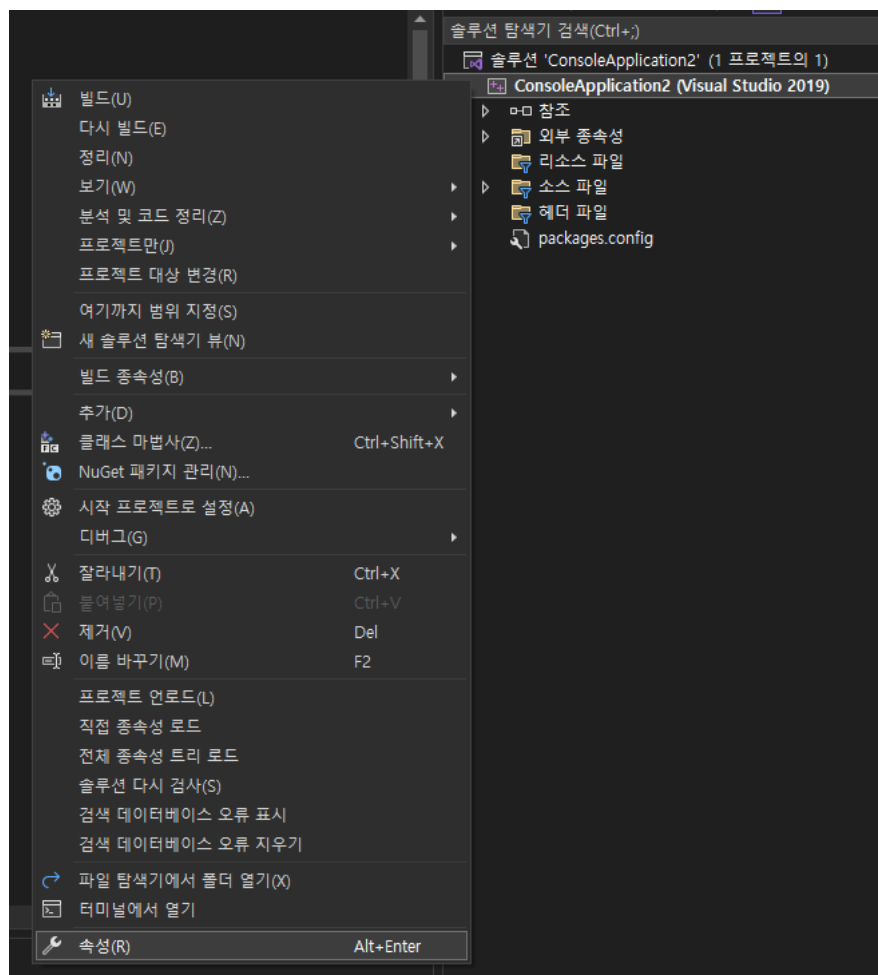
서비스정보

참고문서	OpenAPI 활용가이드(코로나19 확진자 성별 연령별 현황).hwp
데이터포맷	JSON+XML
End Point	http://apis.data.go.kr/1352000/ODMS_COVID_05
API 환경 또는 API 호출 조건에 따라 인증키가 적용되는 방식이 다를 수 있습니다. 포털에서 제공되는 Encoding/Decoding 된 인증키를 적용하면서 구동되는 키를 사용하시기 바랍니다. * 향후 포털에서 더 명확한 정보를 제공하기 위해 노력하겠습니다.	
일반 인증키 (Encoding)	Ws5k5995In2gtc%2FI2DqX3NvqnEAuZ6WhbEoRqm56kxdzbt4J19ezLit6Ahj3dAC3eNSjqgLGz8McdNmcPp1XQ%3D%3D
일반 인증키 (Decoding)	Ws5k5995In2gtc/I2DqX3NvqnEAuZ6WhbEoRqm56kxdzbt4J19ezLit6Ahj3dAC3eNSjqgLGz8McdNmcPp1XQ==

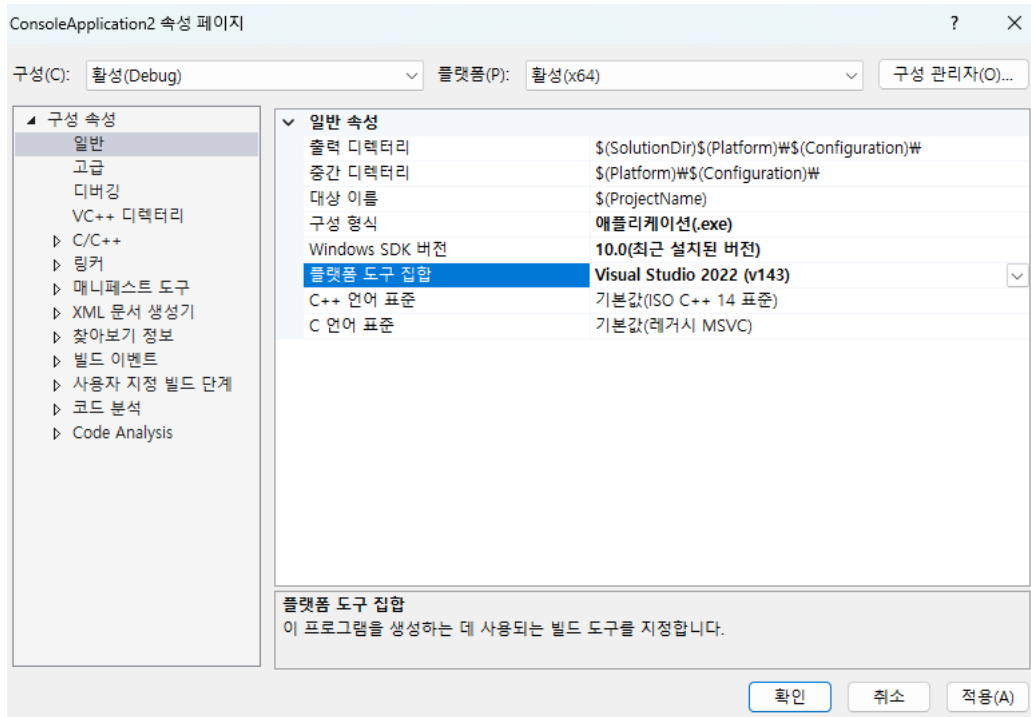
인코딩 키와 디코딩 키가 있는데, 둘 중 작동이 되는 키를 사용하도록 한다.

플랫폼 도구 집합 v142

일단, 다음을 먼저 확인한다.



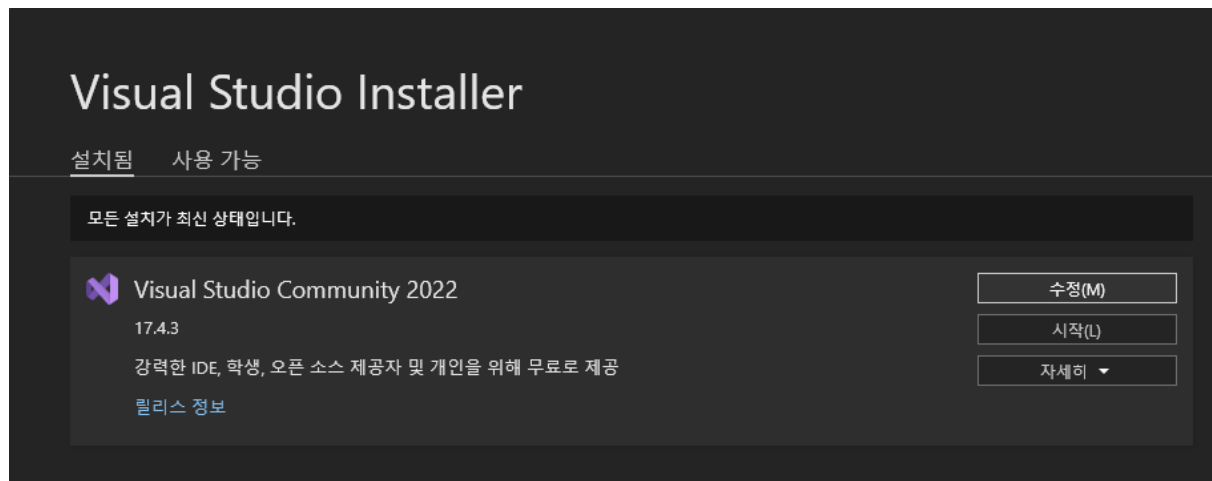
프로젝트 - 속성 클릭




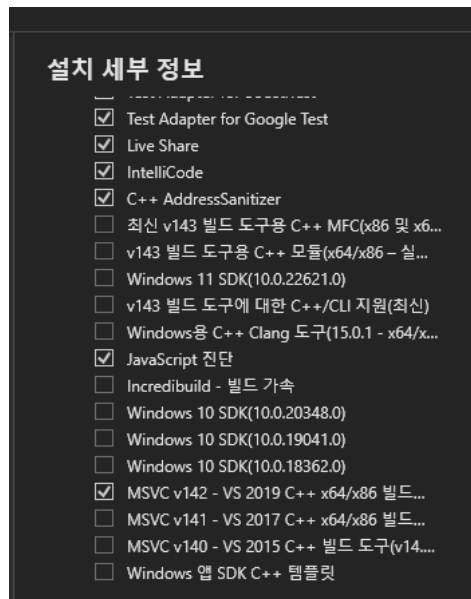
만약 플랫폼 도구 집합이 v143 이라면 (Visual Studio 2022), 추가적인 설정이 필요하다. v142 (Visual Studio 2019) 라면 다음부터 진행 될 부분은 패스해도 좋다.

일단, Visual Studio 를 잠시 종료한다.

Visual Studio Installer 실행

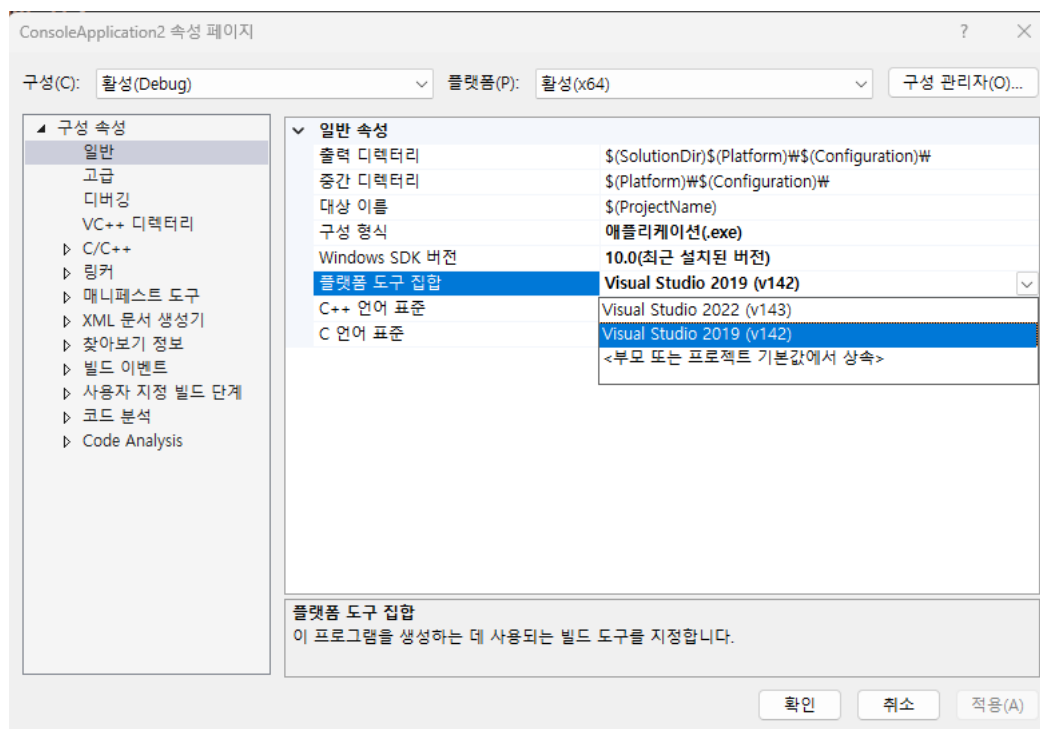


 수정 버튼 클릭



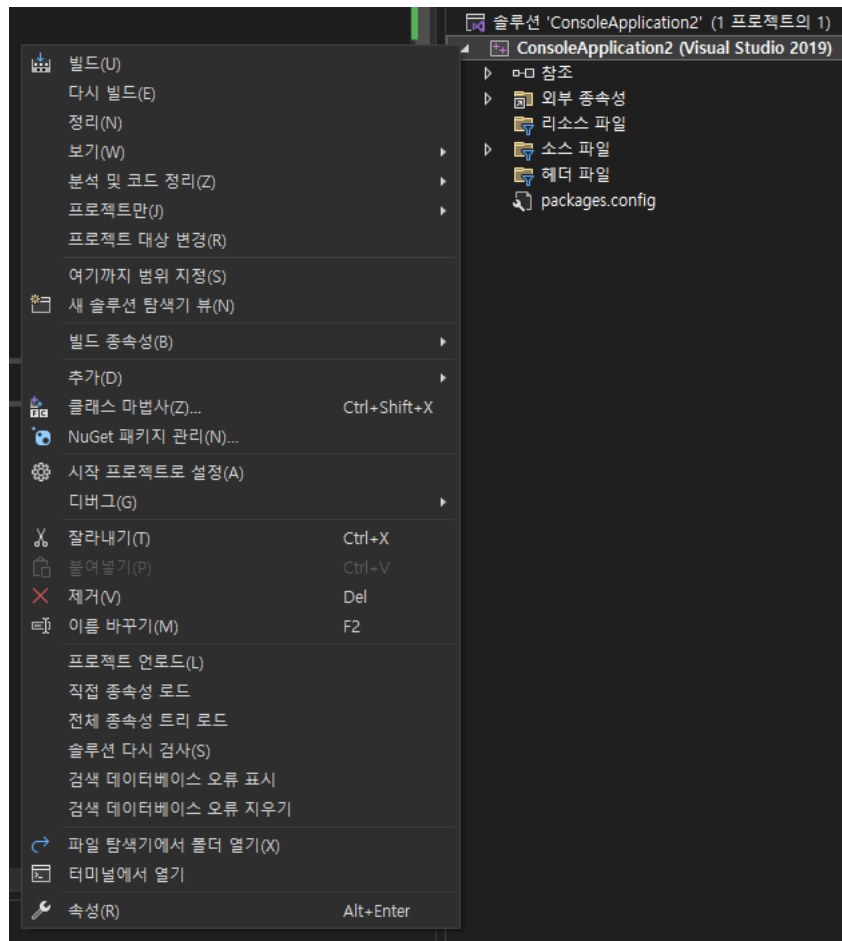
오른쪽 창 설치 세부 정보에서, **MSVC v142 - VS 2019 C++ x64/x86 빌드...** 클릭 후, 아랫쪽에 **수정** 버튼 클릭

설치 이후, Visual Studio 실행

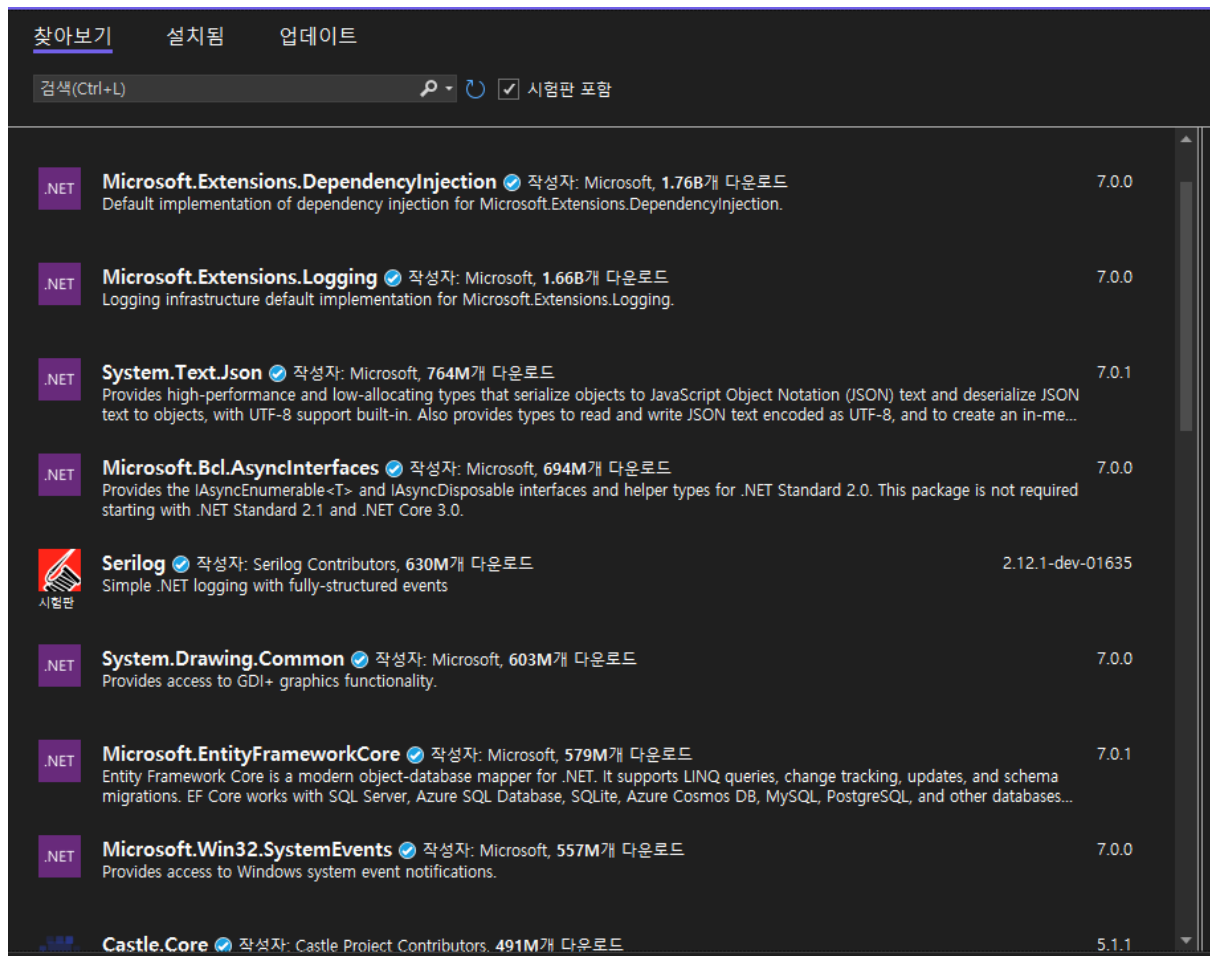


플랫폼 도구 집합 확인하면, **Visual Studio 2019 (v142)** 가 있을 것이다. 설정 후 **적용(A)** 버튼 클릭함.

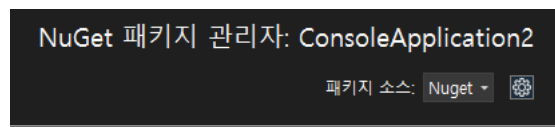
NuGet 패키지 매니저



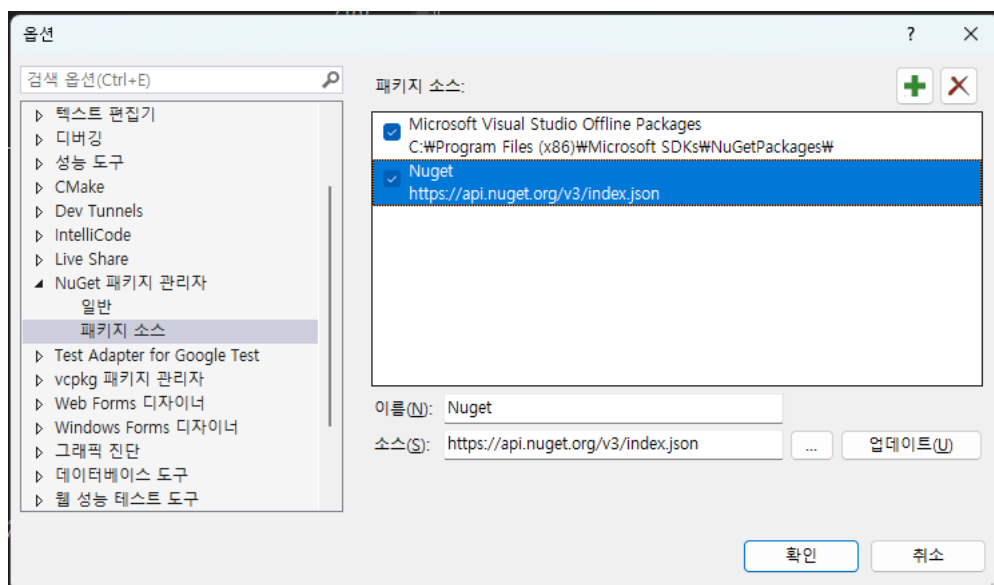
NuGet 패키지 관리 버튼 클릭



만약, **찾아보기** 탭에서 다음과 같이 나오면 정상이고, 나오지 않을 시 추가적인 설정이 필요하다.

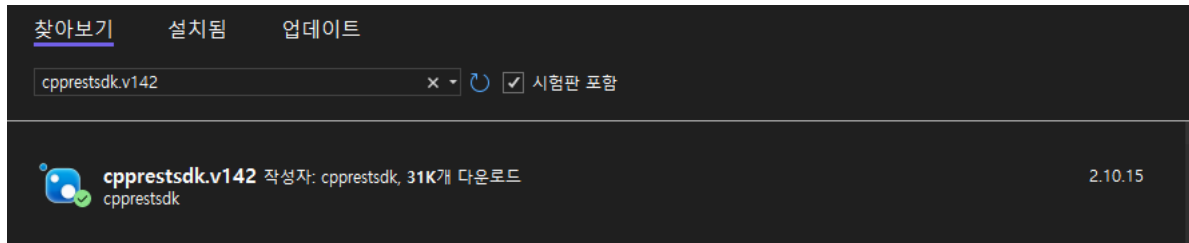


오른쪽 위에 보면, 톱니바퀴 모양 있다. 클릭하자.



다음과 같이 추가하고, 업데이트 버튼 누른 후 확인버튼 누르자.
그러면 패키지를 검색할 수 있다.

cpprestsdk.v142 를 검색한 후 설치해주자.



그리고 다음 코드를 입력해봤을 때, 빨간줄이 뜨지 않으면 `cpprest` 헤더를 가져온 것이고, 세팅 성공이다.

```
#include <iostream>
#include <vector>
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
#include <cpprest/json.h>
using namespace std;
using namespace utility;
using namespace web;
using namespace web::http;
using namespace web::http::client;
using namespace web::json;
using namespace concurrency::streams;

int main()
{
    // 유니코드 (windows only) 설정
    wcout.imbue(locale("kor"));
    HttpRequest();
    return 0;
}
```

이 중, namespace 는 사용하고자 하는 함수 명령어 중, 이름이 중복될 경우가 높을 때, 함수 이름 충돌을 방지하기 위해 설정한다.
한글 정상 출력을 위해, wcout 사용 시 기본적으로 유니코드로 출력하도록 설정해주자.

REST API 테스트

REST API 는 http 프로토콜을 사용한 요청이다. API 서버에 URL 로 된 요청(Request) 을 보내면, 응답(Response) 를 받는 형식이다.

ex)

CLIENT: API 야, 2022년 2월 13일 기준 코로나19 확진자 성별, 연령별 현황을 보여줘.

API: 자 여기있다


그러나, 사실 이것보단 복잡한데, API 가 필요로 하는 정보가 부족하기 때문이다. API 가 원하는 정보를, 명시된 변수와 함께 URL 에 보내야 한다.

이를 위해선 해당 API 의 문서 분석이 필수다.

목록 보건복지부_코로나19 확진자 성별 연령별 현황

조회

코로나19 감염현황을 성별 및 연령별로 조회하는 기능

- 활용승인 절차 개발단계 : 심의승인 / 운영단계 : 자동승인
- 신청가능  래픽 개발계정 : 10,000 / 운영계정 : 활용사례 등록시 신청하면 트래픽 증가 가능
- 요청주소 http://apis.data.go.kr/1352000/ODMS_COVID_05/callCovid05Api
- 서비스URL http://apis.data.go.kr/1352000/ODMS_COVID_05

요청주소(url): API 주소

요청변수(Request Parameter)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
서비스키	serviceKey	100	필	-	공공데이터포털에서 받은 인증키
페이지 번호	pageNo	4	필	1	페이지번호
한 페이지 결과 수	numOfRows	4	필	500	한 페이지 결과 수
결과형식(xml/json)	apiType	4	옵	xml	결과형식(xml/json)
기준일	create_dt	10	옵	2022-01-08	기준일

요청변수는 API 사용 시 같이 보낼 변수들이다.

항목구분에서 **필** 은 필수를 의미하고, **옵** 은 옵션이다.

항목명(영문) 의 변수 이름은 반드시 일치해야 한다. 대문자를 소문자로 바꿔서도 안된다.

serviceKey: 발급 받은 키

pageNo: 전체 결과를 numOfRows 에서 나눈 값이 전체 페이지 수. 보고자 하는 페이지 번호

numOfRows: 한 페이지에 들어갈 row 의 수

apiType: 보고자 하는 형식인데, 우린 JSON 사용

create_dt: 기준일

이를 기반으로, `HttpRequest()` 작성을 시작해보자.

가장 먼저, 요청 URL을 만들자.

```
void HttpRequest()
{
    wstring url = U("http://apis.data.go.kr/1352000/ODMS_COVID_05/callCovid05Api");
    wstring serviceKey = U("ws5k5995In2gtc/I2DqX3NvqnEAuZ6WhbEoRqm56kxdzbt4Lj19ezLit6Ahj3dAC3eNSjqgLG6z8McdNmcPp1XQ==");
    wstring numOfRows = U("10");
    wstring pageNo = U("1");
    wstring apiType = U("JSON");
    wstring create_dt = U("2022-12-01");
    wstring path = url + U("?")
        + U("serviceKey=") + serviceKey
        + U("&numOfRows=") + numOfRows
        + U("&pageNo=") + pageNo
        + U("&apiType=") + apiType
        + U("&create_dt=") + create_dt;
```

```

    wcout << path << endl;
}

```

여기서 `wstring` 이라는 자료형이 나왔다. 한글 사용을 위한 `string` 이라고 알아두면 된다.

`path` 변수를 만드는 과정이 중요한데, 요청 URL 주소에 변수를 이어 붙여 나간다. 맨 처음엔 `?` 로 시작하고, 추가할 변수가 있으면 `&` 로 이어나간다.

콘솔에 찍힌 결과는 아래와 같다.

```

http://apis.data.go.kr/1352000/ODMS_COVID_05/callCovid05Api?serviceKey=Ws5k5995In2gtc/I2DqX3NvqnEAuZ6WhbEoRqm56kxdzbt4Lj19ezLit6Ahj3dA

```

이것이 작동되는지 간단히 확인하려면 지금 당장 브라우저를 키고, 해당 URL 을 주소창에 입력해보면 알 수 있다.



```

{"pageNo": "1", "resultCode": "00", "totalCount": 12100, "items":
[{"criticalRate": "0", "death": "34", "deathRate": "0.11", "confCaseRate": "10.69", "createDt": "2022-12-01", "confCase": "2902798", "gubun": "0-9"},
{"criticalRate": "0.04", "death": "1259", "deathRate": "4.12", "confCaseRate": "12.86", "createDt": "2022-12-01", "confCase": "3493571", "gubun": "50-59"},
{"criticalRate": "0.01", "death": "414", "deathRate": "1.35", "confCaseRate": "15.23", "createDt": "2022-12-01", "confCase": "4136635", "gubun": "40-49"},
{"criticalRate": "0.12", "death": "3512", "deathRate": "11.49", "confCaseRate": "10.7", "createDt": "2022-12-01", "confCase": "2905387", "gubun": "60-69"},
{"criticalRate": "0", "death": "18", "deathRate": "0.06", "confCaseRate": "12.68", "createDt": "2022-12-01", "confCase": "3443743", "gubun": "10-19"},
{"criticalRate": "0", "death": "74", "deathRate": "0.24", "confCaseRate": "14.64", "createDt": "2022-12-01", "confCase": "3976879", "gubun": "20-29"},
{"criticalRate": "0.12", "death": "14930", "deathRate": "48.84", "confCaseRate": "46.54", "createDt": "2022-12-01", "confCase": "12638711", "gubun": "남성"},
{"criticalRate": "0.11", "death": "15638", "deathRate": "51.16", "confCaseRate": "53.46", "createDt": "2022-12-01", "confCase": "14517102", "gubun": "여성"},
{"criticalRate": "2.07", "death": "18141", "deathRate": "59.35", "confCaseRate": "3.23", "createDt": "2022-12-01", "confCase": "876409", "gubun": "80 이상"},
{"criticalRate": "0", "death": "141", "deathRate": "0.46", "confCaseRate": "14.56", "createDt": "2022-12-01", "confCase": "3954185", "gubun": "30-39"}], "numOfRows": "10", "resultMsg": "NORMAL SERVICE"}

```

다음과 같이 찍히면, 적어도 URL 은 잘 만들었다는 뜻이다.

우리의 1차 목표는 이 데이터를 브라우저가 아니라, C++ 을 이용해 콘솔 창에 찍어 보는 것이다.

```

void HttpRequest()
{
    wstring url = U("http://apis.data.go.kr/1352000/ODMS_COVID_05/callCovid05Api");
    wstring serviceKey = U("Ws5k5995In2gtc/I2DqX3NvqnEAuZ6WhbEoRqm56kxdzbt4Lj19ezLit6Ahj3dAC3eNsJqgL6z8McdNmcPp1XQ==");
    wstring numOfRows = U("10");
    wstring pageNo = U("1");
    wstring apiType = U("JSON");
    wstring create_dt = U("2022-12-01");
    wstring path = url + U("?")
        + U("serviceKey=") + serviceKey
        + U("&numOfRows=") + numOfRows
        + U("&pageNo=") + pageNo
        + U("&apiType=") + apiType
        + U("&create_dt=") + create_dt;
    wcout << path << endl;

    http_client client(path);
    http_request get_req(methods::GET);

    auto get_resp = client.request(get_req).get();
    cout << get_resp.status_code() << " : sync request" << endl;
    // 만약, json 이 아니라 string 으로 처리하고 싶으면(HTML 문서 크롤링 등) 다음을 대신 사용
    // get_resp.extract_string(true).get();
    auto jsonData = get_resp.extract_json(true).get(); //<- json data의 전체본
    // serialized jsonData 정상 출력을 위해 직접 정의한 함수
}

```

```
// wstring 에서 전체 출력이 안되는 에 방식을 위해 사용
printWstring(jsonData.serialize());
}
```

각각의 부분을 자세히 확인해보자.

```
http_client client(path);
http_request get_req(methods::GET);
```

먼저 두 개의 객체를 생성한다.

`http_client`, `http_request` 클래스에서 생성자를 사용해 각각 `client` 와 `get_req` 객체를 생성하고, `client` 에는 우리가 만든 URL 인 `path` 를, `get_req` 에는 `methods::GET` 을 각각 파라미터로 넣어준다.

`methods::GET` 은 정확하게는 `web::http::methods::GET` 이나, 네임스페이스에서 정의되어있으므로 간단히 `methods::GET` 으로 사용가능하다.

여기서 GET 은, HTTP 메서드 중 하나로서, 가져온다는 뜻을 가지고 있다.

```
auto get_resp = client.request(get_req).get();
```

`get_resp` 의 자료형은 `auto` 인데, 이것은 리턴값이 어떤 타입을 가지는지 모를 때 사용한다고 보면 된다. 리턴값의 자료형을 자동으로 적용한다.

결과적으로, `get_resp` 은 `web::http::http_response` 자료형이 된다.

여기엔 `client.request(get_req).get()` 의 리턴값이 담기며, 여기서 비로소 서버에 요청이 들어간다. 즉, 이 코드 이전엔 서버에 그 어떤 요청도 가지 않는다.

```
cout << get_resp.status_code() << " : sync request" << endl; // <- 결과 상태 확인
```

`web::http::http_response` 자료형은 `status_code()` 라는 메서드를 가지고 있는데, 여기서 서버와 통신 결과를 HTTP 응답 상태 코드로 확인할 가능하다. 성공했다면 `200` 이 확인된다.

```
// 만약, json 이 아니라 string 으로 처리하고 싶으면(HTML 문서 크롤링 등) 다음을 대신 사용
// get_resp.extract_string(true).get();
auto jsonData = get_resp.extract_json(true).get(); //<- json data의 전체본
// wstring 에서 전체 출력이 안되는 에러 방식을 위해 사용
printWstring(jsonData.serialize());
```

여기서 결과값을 `JSON` 포맷으로 바꾸게 되는데, `JSON` 은 추후 배울 JavaScript 객체에서 아이디어를 얻은 REST API 기본 포맷이라고만 알아두자.

이것은 그냥 출력할 순 없고, `serialize` 를 거쳐야 비로소 문자열이 된다. 이것을 우리가 직접 정의할 `printWstring` 에 넣어주어, 전체 출력이 안되는 에러를 해결해주었다.

해당 함수는 다음과 같다.

```
// wstring 에서 전체 출력이 안되는 에러 방식을 위해 사용
void printWstring(wstring wstr) {
    wstring str;
    str.assign(wstr.begin(), wstr.end());
    wcout << str << endl;
}
```

`assign` : iterator 사용해 문자열을 옮기는 함수

`begin` : 문자열의 첫글자

`end` : 문자열의 끝글자

지금까지의 코드를 실행시키면 콘솔 다음과 같은 결과가 나와야한다.

```
{
  "items": [
    {
      "confCase": "2902798",
      "confCaseRate": "10.69",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "34",
      "deathRate": "0.11",
      "gubun": "0-9"
    },
    {
      "confCase": "3493571",
      "confCaseRate": "12.86",
      "createDt": "2022-12-01",
      "criticalRate": "0.04",
      "death": "1259",
      "deathRate": "4.12",
      "gubun": "50-59"
    },
    {
      "confCase": "4136635",
      "confCaseRate": "15.23",
      "createDt": "2022-12-01",
      "criticalRate": "0.01",
      "death": "414",
      "deathRate": "1.35",
      "gubun": "40-49"
    },
    {
      "confCase": "2905387",
      "confCaseRate": "10.7",
      "createDt": "2022-12-01",
      "criticalRate": "0.12",
      "death": "3512",
      "deathRate": "11.49",
      "gubun": "60-69"
    },
    {
      "confCase": "3443743",
      "confCaseRate": "12.68",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "18",
      "deathRate": "0.06",
      "gubun": "10-19"
    },
    {
      "confCase": "3976879",
      "confCaseRate": "14.64",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "74",
      "deathRate": "0.24",
      "gubun": "20-29"
    },
    {
      "confCase": "1263871",
      "confCaseRate": "46.54",
      "createDt": "2022-12-01",
      "criticalRate": "0.12",
      "death": "14930",
      "deathRate": "48.84",
      "gubun": "남 성"
    },
    {
      "confCase": "14517102",
      "confCaseRate": "53.46",
      "createDt": "2022-12-01",
      "criticalRate": "0.11",
      "death": "15638",
      "deathRate": "51.16",
      "gubun": "여 성"
    },
    {
      "confCase": "876409",
      "confCaseRate": "3.23",
      "createDt": "2022-12-01",
      "criticalRate": "2.07",
      "death": "18141",
      "deathRate": "59.35",
      "gubun": "80 이상"
    },
    {
      "confCase": "3954185",
      "confCaseRate": "14.56",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "141",
      "deathRate": "0.46",
      "gubun": "30-39"
    }
  ],
  "numOfRows": "10",
  "pageNo": "1",
  "resultCode": "00",
  "resultMsg": "NORMAL SERVICE",
  "totalCount": "12122"
}
```

어디서 많이 보지 않았는가? 그렇다. 브라우저의 결과와 일치한다.

원하는 데이터 뽑아내기 (Parsing)

이제 이 데이터에서 원하는 데이터만 뽑아내볼것이다.

일단, 출력된 결과값부터 보고 판단해보자.

```
{
  "items": [
    {
      "confCase": "2902798",
      "confCaseRate": "10.69",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "34",
      "deathRate": "0.11",
      "gubun": "0-9"
    },
    {
      "confCase": "3493571",
      "confCaseRate": "12.86",
      "createDt": "2022-12-01",
      "criticalRate": "0.04",
      "death": "1259",
      "deathRate": "4.12",
      "gubun": "50-59"
    },
    {
      "confCase": "4136635",
      "confCaseRate": "15.23",
      "createDt": "2022-12-01",
      "criticalRate": "0.01",
      "death": "414",
      "deathRate": "1.35",
      "gubun": "40-49"
    },
    {
      "confCase": "2905387",
      "confCaseRate": "10.7",
      "createDt": "2022-12-01",
      "criticalRate": "0.12",
      "death": "3512",
      "deathRate": "11.49",
      "gubun": "60-69"
    },
    {
      "confCase": "3443743",
      "confCaseRate": "12.68",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "18",
      "deathRate": "0.06",
      "gubun": "10-19"
    },
    {
      "confCase": "3976879",
      "confCaseRate": "14.64",
      "createDt": "2022-12-01",
      "criticalRate": "0",
      "death": "74",

```

```

        "deathRate": "0.24",
        "gubun": "20-29"
    },
    {
        "confCase": "12638711",
        "confCaseRate": "46.54",
        "createDt": "2022-12-01",
        "criticalRate": "0.12",
        "death": "14930",
        "deathRate": "48.84",
        "gubun": "남성"
    },
    {
        "confCase": "14517102",
        "confCaseRate": "53.46",
        "createDt": "2022-12-01",
        "criticalRate": "0.11",
        "death": "15638",
        "deathRate": "51.16",
        "gubun": "여성"
    },
    {
        "confCase": "876409",
        "confCaseRate": "3.23",
        "createDt": "2022-12-01",
        "criticalRate": "2.07",
        "death": "18141",
        "deathRate": "59.35",
        "gubun": "80 이상"
    },
    {
        "confCase": "3954185",
        "confCaseRate": "14.56",
        "createDt": "2022-12-01",
        "criticalRate": "0",
        "death": "141",
        "deathRate": "0.46",
        "gubun": "30-39"
    }
},
"numOfRows": "10",
"pageNo": "1",
"resultCode": "00",
"resultMsg": "NORMAL SERVICE",
"totalCount": 12122
}

```

다음은 목표로 한다.

1. `items` 배열의 각각을 `item` 으로 가져와 한 줄씩 출력
2. `gubun` 필드가 `"0-9"` 인 것만 뽑아내서 출력

```

auto items = jsonData.at(U("items")).as_array();
// iterator
for (auto item : items) {
    printWstring(item.serialize());
}

wstring keyword_gubun = U("0-9");
for (auto item : items) {
    if (keyword_gubun == item.at(U("gubun")).as_string()) {
        printWstring(U("Find ") + keyword_gubun);
        printWstring(item.serialize());
    }
}

```

`web::json::value` 타입에서는 `at`, `as_array` 메서드를 제공한다. `at` 은 해당 키를 찾을 때 사용하며, `as_array` 는 해당 키의 값을 배열로 변경해준다.

`for` 의 형태가 특이한데, iterator 방식의 `for` 이다. `items` 의 각각을 `item` 으로 받아 처리한다.

결과는 다음과 같다.

1번

```
{
  "confCase": "2902798",
  "confCaseRate": "10.69",
  "createDt": "2022-12-01",
  "criticalRate": "0",
  "death": "34",
  "deathRate": "0.11",
  "gubun": "0-9"
},
{
  "confCase": "3493571",
  "confCaseRate": "12.86",
  "createDt": "2022-12-01",
  "criticalRate": "0.04",
  "death": "1259",
  "deathRate": "4.12",
  "gubun": "50-59"
},
{
  "confCase": "4136635",
  "confCaseRate": "15.23",
  "createDt": "2022-12-01",
  "criticalRate": "0.01",
  "death": "414",
  "deathRate": "1.35",
  "gubun": "40-49"
},
{
  "confCase": "2905387",
  "confCaseRate": "10.7",
  "createDt": "2022-12-01",
  "criticalRate": "0.12",
  "death": "3512",
  "deathRate": "11.49",
  "gubun": "60-69"
},
{
  "confCase": "3443743",
  "confCaseRate": "12.68",
  "createDt": "2022-12-01",
  "criticalRate": "0",
  "death": "18",
  "deathRate": "0.06",
  "gubun": "10-19"
},
{
  "confCase": "3976879",
  "confCaseRate": "14.64",
  "createDt": "2022-12-01",
  "criticalRate": "0",
  "death": "74",
  "deathRate": "0.24",
  "gubun": "20-29"
},
{
  "confCase": "12638711",
  "confCaseRate": "46.54",
  "createDt": "2022-12-01",
  "criticalRate": "0.12",
  "death": "14930",
  "deathRate": "48.84",
  "gubun": "남 성"
},
{
  "confCase": "14517102",
  "confCaseRate": "53.46",
  "createDt": "2022-12-01",
  "criticalRate": "0.11",
  "death": "15638",
  "deathRate": "51.16",
  "gubun": "여 성"
},
{
  "confCase": "876409",
  "confCaseRate": "3.23",
  "createDt": "2022-12-01",
  "criticalRate": "2.07",
  "death": "18141",
  "deathRate": "59.35",
  "gubun": "80 이상"
},
{
  "confCase": "3954185",
  "confCaseRate": "14.56",
  "createDt": "2022-12-01",
  "criticalRate": "0",
  "death": "141",
  "deathRate": "0.46",
  "gubun": "30-39"
}
```

2번

```
Find 0-9
{
  "confCase": "2902798",
  "confCaseRate": "10.69",
  "createDt": "2022-12-01",
  "criticalRate": "0",
  "death": "34",
  "deathRate": "0.11",
  "gubun": "0-9"
}
```

전체 코드는 다음과 같다.

```
#include <iostream>
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
#include <cpprest/json.h>

using namespace std;
using namespace utility;
using namespace web;
using namespace web::http;
using namespace web::http::client;
using namespace web::json;
using namespace concurrency::streams;

// wstring 에서 전체 출력이 안되는 에러 방지를 위해 사용
void printWstring(wstring wstr) {
    wstring str;
    str.assign(wstr.begin(), wstr.end());
    wcout << str << endl;
}

void HttpRequest()
{
    wstring url = U("http://apis.data.go.kr/1352000/ODMS_COVID_05/callCovid05Api");
    wstring serviceKey = U("ws5k5995In2gtc/I2DqX3NvqnEauZ6WhbEoRqm56kxdzbt4Lj19ezLit6Ahj3dAC3eNSjqgLG6z8McdNmcPp1XQ==");
    wstring numofRows = U("10");
    wstring pageNo = U("1");
    wstring apiType = U("JSON");
    wstring create_dt = U("2022-12-01");
    wstring path = url + U("?")
        + U("serviceKey=") + serviceKey
        + U("&numofRows=") + numofRows
        + U("&pageNo=") + pageNo
        + U("&apiType=") + apiType
        + U("&create_dt=") + create_dt;
    wcout << path << endl;

    http_client client(path);
    http_request get_req(methods::GET);

    auto get_resp = client.request(get_req).get();
    cout << get_resp.status_code() << " : sync request" << endl; // <- 결과 상태 확인
    // 만약, json 이 아니라 string 으로 처리하고 싶으면(HTML 문서 크롤링 등) 다음을 대신 사용
    // get_resp.extract_string(true).get();
    auto jsonData = get_resp.extract_json(true).get(); //<- json data의 전체본
    // serialized jsonData 정상 출력을 위해 직접 정의한 함수
    // wstring 에서 전체 출력이 안되는 에러 방지를 위해 사용
    printWstring(jsonData.serialize());

    // 여기서부터 원하는 데이터를 뽑아내는 훈련
    auto items = jsonData.at(U("items")).as_array();
    // iterator
```

```

    for (auto item : items) {
        printWstring(item.serialize());
    }

    wstring keyword_gubun = U("0-9");
    for (auto item : items) {
        if (keyword_gubun == item.at(U("gubun")).as_string()) {
            printWstring(U("Find ") + keyword_gubun);
            printWstring(item.serialize());
        }
    }
}

int main()
{
    // 유니코드 (windows only) 설정
    wcout.imbue(locale("kor"));
    HttpRequest();

    return 0;
}

```

만약, JSON 이 아니라 html 문서를 가져오고 싶다면?

다음 뉴스 기준 (<https://news.daum.net/>)

```

#include <iostream>
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
#include <cpprest/json.h>

using namespace std;
using namespace utility;
using namespace web;
using namespace web::http;
using namespace web::http::client;
using namespace web::json;
using namespace concurrency::streams;

void printWstring(wstring wstr) {
    wstring str;
    str.assign(wstr.begin(), wstr.end());
    wcout << str << endl;
}

void HttpRequest()
{
    wstring path = U("https://news.daum.net/");
    wcout << path << endl;
    http_client client(path);
    http_request get_req(methods::GET);

    auto get_resp = client.request(get_req).get();
    cout << get_resp.status_code() << " : sync request" << endl; // <- 결과 상태 확인
    // JSON 이 아니라 html 문서이기 때문에, 다음 부분이 JSON 할때와는 다르다.
    auto html = get_resp.extract_string(true).get();
    printWstring(html);
}

int main()
{
    // 유니코드 (windows only)
    // 응답 JSON 에 한국어 있을 시, 주석 풀어보기
    wcout.imbue(locale("kor"));
    HttpRequest();

    return 0;
}

```

```

        <li data-tiara-layer="article">
            <div class="item_photonews">
                <a href="https://gallery.v.daum.net/p/viewer/5615616/xTjEAN1ijk" class="wrap_thumb" data-tia
a-id="20230109161930462" data-tiara-type="harmony" data-tiara-ordnum="4" data-tiara-custom="contentUniqueKey=hamny-2023
109161930462">
                    
                    <span class="ico_news ico_photo">포토 갤러리</span>
                </a>
                <div class="cont_thumb">
                    <strong class="tit_g">
                        <a href="https://gallery.v.daum.net/p/viewer/5615616/xTjEAN1ijk" class="link_txt" da
a-tiara-id="20230109161930462" data-tiara-type="harmony" data-tiara-ordnum="4" data-tiara-custom="contentUniqueKey=hamn
-20230109161930462"> 무료 정장으로 '취업 날개' 달자!</a>
                    </strong>
                    <a href="https://gallery.v.daum.net/p/viewer/5615616/" class="info_cp" data-tiara-id="56
5616" data-tiara-type="gallery" data-tiara-ordnum="4" data-tiara-custom="contentUniqueKey=gallery=5615616"><span class=
screen_out">출처 </span>뉴스1 PICK</a>
                </div>
            </div>
        </li>
    </ul>
</div>

<!-- 바로잡습니다 -->
<div class="box_side" data-tiara-layer="right">
    <h3 class="tit_box">고침 기사, 정정
:Users\mincoding\source\repos\ConsoleApplication3\x64\Debug\ConsoleApplication3.exe(프로세스 3764개)이(가) 종료되었습
니다(코드: 0개).

```

HTML 문서 전체를 가져오며, 크롤링 C++ 파싱 라이브러리를 따로 찾아보던지 해야한다.