



[9기 스타트캠프] C++ Crawling

🕒 Created	@2023년 1월 10일 오전 10:15
🕒 Last Edited Time	@2023년 1월 10일 오후 4:27
📄 Type	Project Kickoff 🚀
📄 Status	
👤 Created By	
👤 Last Edited By	
👥 Stakeholders	
📅 날짜	



이번 단계에서는 C++을 이용한 간단한 Crawling 에 대해 배워봅니다.
차근 차근 따라해봅시다.

목차

[웹 크롤링이란](#)
[웹 크롤링 과정](#)
[웹크롤링 시작하기](#)
[웹 크롤링 샘플 코드](#)
[C++ 웹 크롤링 프로젝트](#)
 [프로젝트 소개](#)
 [프로젝트 준비](#)
 [URL 인코딩](#)
 [문자열 파싱](#)
 [프로젝트 완성 코드](#)

웹 크롤링이란

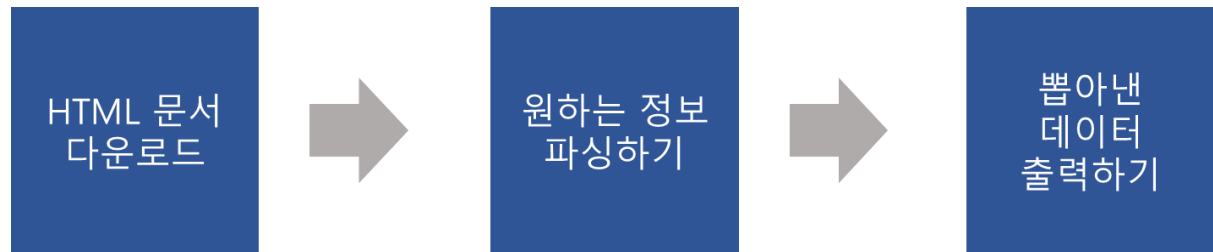
- 웹 페이지에서 자료를 긁어오는 자동 프로그램
- Web에 있는 HTML에 있는 데이터를 가져오고 파싱



웹 상에 있는 데이터를 모으기 위해 크롤링을 합니다.

웹 크롤링 과정

웹 크롤링은 총 세 가지 단계로 이루어집니다.



[참고] 로봇 배제 표준 (robot.txt)

웹 페이지 제작자가 크롤링을 허용하는 곳과 허용하지 않는 곳을 구분하여 적은 문서 권고사항이며, robot.txt 를 무시하고 데이터를 가져올 수도 있다.

ex)

- <https://www.google.com/robots.txt>
- <https://sports.news.naver.com/robots.txt>
- <http://mcdonalds.co.kr/robots.txt>


```
User-agent: googlebot      # googlebot 로봇만 적용
Disallow: /private/      # 이 디렉토리를 접근 차단한다.

User-agent: googlebot-news # googlebot-news 로봇만 적용
Disallow: /               # 모든 디렉토리를 접근 차단한다.

User-agent: *             # 모든 로봇 적용
Disallow: /something/     # 이 디렉토리를 접근 차단한다.
```

웹크롤링 시작하기

크롬을 실행해 naver.com 을 가자.

뒤로	Alt+왼쪽 화살표	자음 줄바꿈 ☐
앞으로	Alt+오른쪽 화살표	
새로고침	Ctrl+R	
다른 이름으로 저장...	Ctrl+S	
인쇄	Ctrl+P	
전송...		
Google로 이미지 검색		
 기기로 전송		
 이 페이지의 QR 코드 생성		
한국어(으로) 번역		
페이지 소스 보기	Ctrl+U	
검사		

웹 크롤링 샘플 코드

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

```

void HttpRequest()
{
    wstring path = U("https://www.naver.com/");
    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;
    auto html = get_resp.extract_string(true).get();

    wstring str;
    str.assign(html.begin(), html.end());
    wcout << str << endl;
}

int main()
{
    wcin.imbue(locale("kor"));
    wcout.imbue(locale("kor"));
    HttpRequest();
    return 0;
}

```

실행 결과

```

선택 Microsoft Visual Studio 디버그 콘솔
https://www.naver.com/
200 : sync request

<!doctype html>
<html lang="ko" data-dark="false"> <head> <meta charset="utf-8"> <title>NAVER</title> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=1190"> <meta name="apple-mobile-web-app-title" content="NAVER"> <meta name="robots" content="index,nofollow"> <meta name="description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"> <meta property="og:title" content="네이버"> <meta property="og:url" content="https://www.naver.com/"> <meta property="og:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png"> <meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"> <meta name="twitter:card" content="summary"> <meta name="twitter:title" content=""> <meta name="twitter:url" content="https://www.naver.com/"> <meta name="twitter:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png"> <meta name="twitter:description" content="네이버 메인에서 다양한 정보와 유용한 콘텐츠를 만나 보세요"> <link rel="stylesheet" href="https://pm.pstatic.net/dist/css/nmain.20221110.css"> <link rel="stylesheet" href="https://ssl.pstatic.net/sstatic/search/pc/css/sp_autocomplete.220526.css"> <link rel="shortcut icon" type="image/x-icon" href="/favicon.ico?1"/> <link rel="apple-touch-icon" sizes="114x114" href="https://s.pstatic.net/static/www/u/2014/0328/mma_204243574.png"/> <link rel="apple-touch-icon" href="https://s.pstatic.net/static/www/u/2014/0328/mma_20432863.png"/> <script>window.nmain=window.nmain||{};window.nmain.supportFlicking=!1;var nsc="navertop.v4",ua=navigator.userAgent,useleJSFlag="1";window.nmain.isIE&&(Object.create=function(n){function e(){}return e.prototype=n,new e})</script> <script>var darkmode= false;window.naver_corp_da=window.naver_corp_da||{main: {}},window.naver_corp_da.main=window.naver_corp_da.main||{};window.naver_corp_da.main.darkmode=darkmode,window.gladsk=window.gladsk||{cmd: []},window.gladsk.

```

path 출력

응답 결과 출력

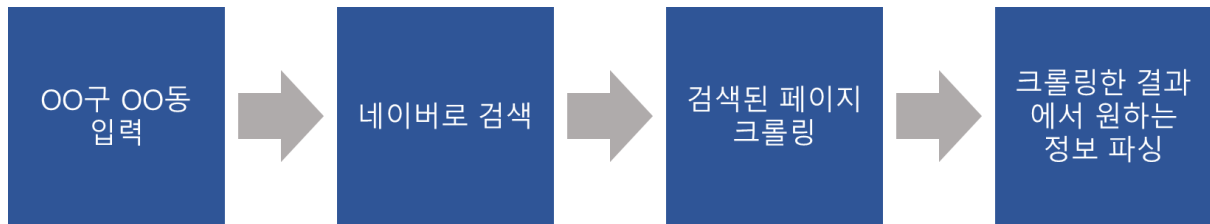
html 소스 출력, 너무 길어 전부가 나오진 않는다.

C++ 웹 크롤링 프로젝트

프로젝트 소개

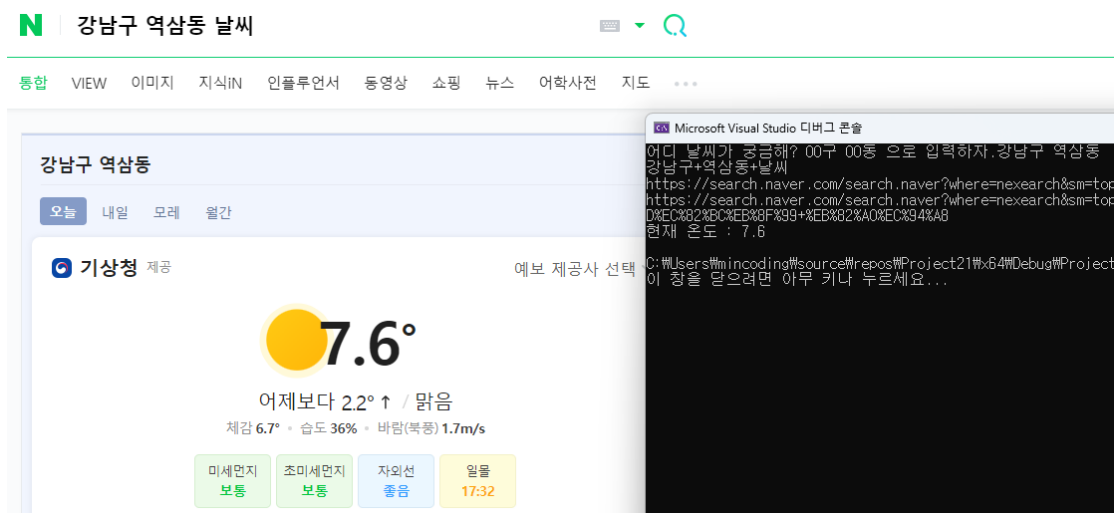
사용자가 날씨 정보를 알고자 하는 지역을 “OO구 OO동” 으로 입력하면, 네이버 검색결과를 통해 날씨 정보를 가져오는 프로젝트를 만들어보자.

전체 과정은 다음과 같다.



입력 : 강남구 역삼동

결과 : 현재온도 : 3.2



프로젝트 준비

본인이 알고 싶은 지역의 날씨를 검색하자. [naver.com](https://www.naver.com) 에 날씨를 검색하자.



온도 / 습도 / 풍속 등의 다양한 날씨 정보를 확인 할 수 있다.

주소 창을 살펴보면,

- “강남구 역삼동 날씨” 검색 했을 때,

search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=1&ie=utf8&query=강남구+역삼동+날씨

- “분당구 구미동 날씨” 검색 했을 때,

search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=0&ie=utf8&query=분당구+구미동+날씨

"https://search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=0&ie=utf8&query=" + OO
구+OO동+날씨 를 기입하면 되는 것을 알 수 있다.

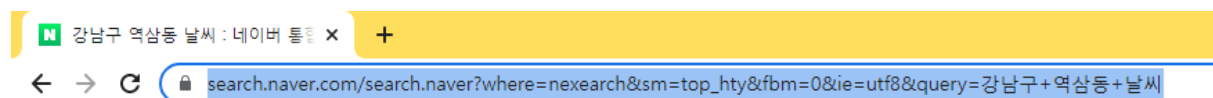
이걸 이용해서 코드를 작성한다.

URL 인코딩

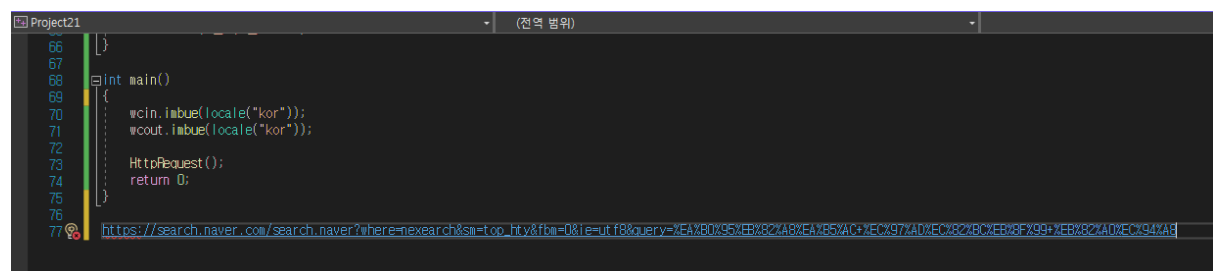
미리 언급하자면, 웹 사이트로 원하는 검색 결과를 얻기 위해서, &query= 다음에 원하는 정보를 입력할 때, 한글로 입력하는 과정에서 에러가 발생한다.

인코딩이라는 개념에 대해 알아야 하는 데, 간단한 실습을 통해 직접 확인할 수 있다.

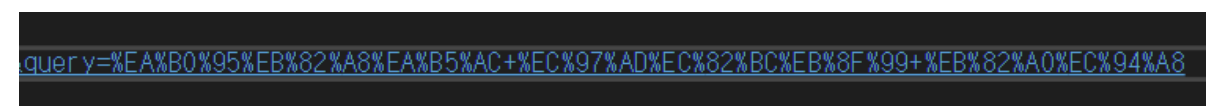
네이버 검색 결과인, 주소창의 주소 값을 복사해서, Visual Studio로 붙여넣기 하자.



와우! 분명히 한글 정보였는데, “강남구+역삼동+날씨” 데이터가 %와 함께, 이상한 숫자와 문자로 변형되어 있는 것을 알 수 있다.



확대해서 보면, “+” 가 중간 중간 있다.



URL에서 URL로 사용할 수 없는 문자 혹은 URL로 사용할 수 있지만 의미가 왜곡 될 수 있을 문자들을 “%XX”의 형태로 변환해주는 것을 URL 인코딩이라 한다.

한글로 입력된 “강남구”, “역삼동”, “날씨” 데이터가 인터넷으로 전송할 수 있도록 ASCII 문자로 변환되어 표현된 것이다.



URL 인코딩 / 디코딩 테스트 사이트

<http://www.hipenpal.com/tool/url-encode-and-decode-in-korean.php>

그래서 직접 구현한 함수를 이용해 쉽게 인코딩해서 크롤링하자!

url_encode() 함수 추가

```
string url_encode(string origin) {
    string ret = "";
    for (int index = 0; index < origin.size(); index++)
    {
        if (!(origin[index] & 0x80)) {
            ret += origin[index];
            continue;
        } // ascii code
        unsigned char el = origin[index];
        char temp[4];
        sprintf(temp, "%%%2X", el);
        ret += temp;
    }
    return ret;
}
```

HttpRequest() 함수 내용 추가

```
void HttpRequest()
{
    wstring search_gu, search_dong;
    cout << "어디 날씨가 궁금해? 00구 00동 으로 입력하자.";
    wcin >> search_gu >> search_dong;
    wstring search = search_gu + U("+") + search_dong + U("+") + U("날씨");
    wcout << search << endl;

    wstring url = U("https://search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=0&ie=utf8&query=");
    url += search;
    wcout << url << endl;

    string utf_8_url = url_encode(to_utf8string(url));

    wstring path;
    path.assign(utf_8_url.begin(), utf_8_url.end());
    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;
    auto html = get_resp.extract_string(true).get();

    wstring str;
    str.assign(html.begin(), html.end());
    wcout << str << endl; //html 결과
}
```


전체 코드

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
#include <cpprest/json.h>
#include <string>
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;
using namespace utility::conversions;

string url_encode(string origin) {
    string ret = "";
    for (int index = 0; index < origin.size(); index++)
    {
        if (!(origin[index] & 0x80)) {
            ret += origin[index];
            continue;
        } // ascii code
        unsigned char el = origin[index];
        char temp[4];
        sprintf(temp, "%%%2X", el);
        ret += temp;
    }
    return ret;
}

void HttpRequest(){
    wstring search_gu, search_dong;
    cout << "어디 날씨가 궁금해? 00구 00동 으로 입력하자.";
    wcin >> search_gu >> search_dong;
    wstring search = search_gu + U("+") + search_dong + U("+") + U("날씨");
    wcout << search << endl;

    wstring url = U("https://search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=0&ie=utf8&query=");
    url += search;
    wcout << url << endl;

    string utf_8_url = url_encode(to_utf8string(url));

    wstring path;
    path.assign(utf_8_url.begin(), utf_8_url.end());
    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;
    auto html = get_resp.extract_string(true).get();

    wstring str;
    str.assign(html.begin(), html.end());
    wcout << str << endl; //html 결과
}

int main(){
    wcin.imbue(locale("kor"));
    wcout.imbue(locale("kor"));
    HttpRequest();
}
```

```

    return 0;
}

```

실행 결과

```

선택 Microsoft Visual Studio 디버그 콘솔
어디 날씨가 궁금해? 00구 00동 으로 입력하자.강남구 역삼동
강남구+역삼동+날씨
https://search.naver.com/search.naver?where=nexearch&sm=top_hfty&fbm=0&ie=utf8&query=강남구+역삼동+날씨
https://search.naver.com/search.naver?where=nexearch&sm=top_hfty&fbm=0&ie=utf8&query=%EA%B0%95%EB%82%A8%EA%B5%AC+%EC%97%AD%EC%82%BC%EB%8F%99+%EB%82%A0%EC%94%A8
200 : sync request
<!doctype html> <html lang="ko"> <head> <meta charset="utf-8"> <meta name="referrer" content="always"> <meta name="format-detection" content="telephone=no,address=no,email=no"> <meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=2.0"> <meta property="og:title" content="강남구 역삼동 날씨 : 네이버 통합검색"/> <meta property="og:image" content="https://ssl.pstatic.net/sstatic/search/common/og_v3.png"> <meta property="og:description" content="강남구 역삼동 날씨의 네이버 통합검색 결과입니다."> <meta name="description" lang="ko" content="강남구 역삼동 날씨의 네이버 통합검색 결과입니다."> <title>강남구 역삼동 날씨 : 네이버 통합검색</title> <link rel="shortcut icon" href="https://ssl.pstatic.net/sstatic/se

```

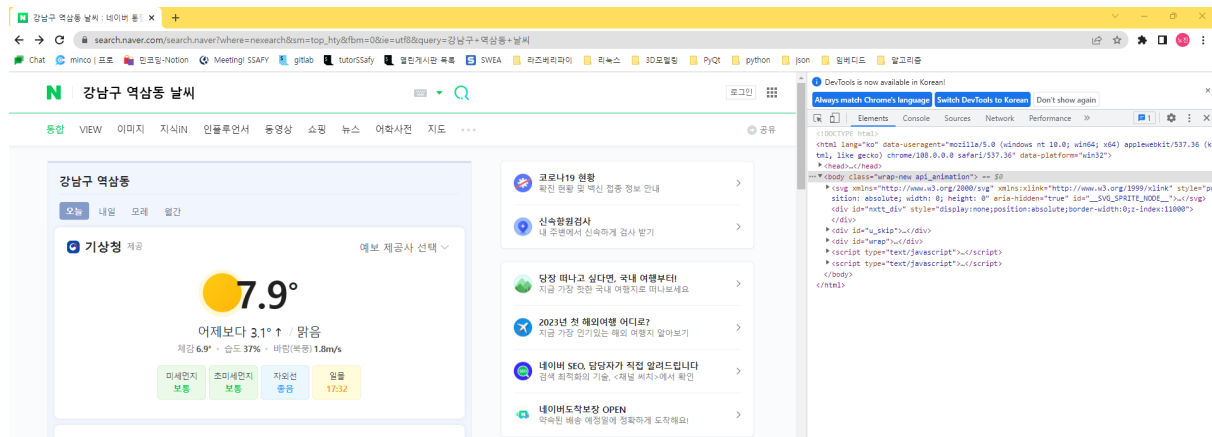
너무 많다. 이제 크롤링한 html 문서에서 원하는 정보만 파싱해서 출력하자!

문자열 파싱

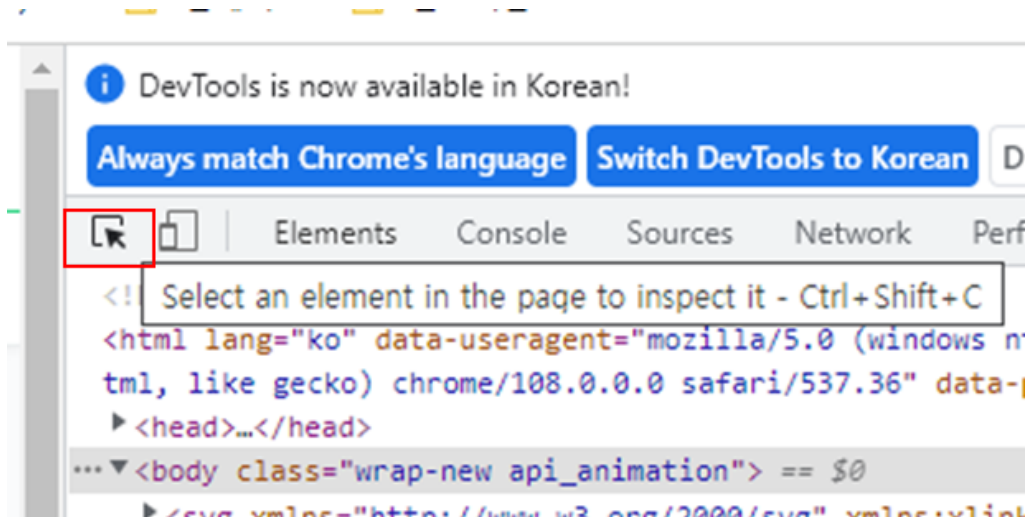
사용자 입력으로 네이버 검색 결과를 크롤링 했다.

이 많은 정보 중에서 우리가 원하는 내용만 파싱해서 콘솔에 출력할 것이다.

크롬 개발자 도구를 다시 열자.



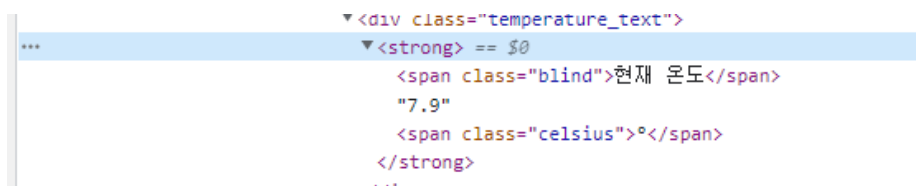
아래와 같이 개발자 도구에서 아래 버튼을 클릭한 후, 마우스를 웹 사이트에서 움직이면,



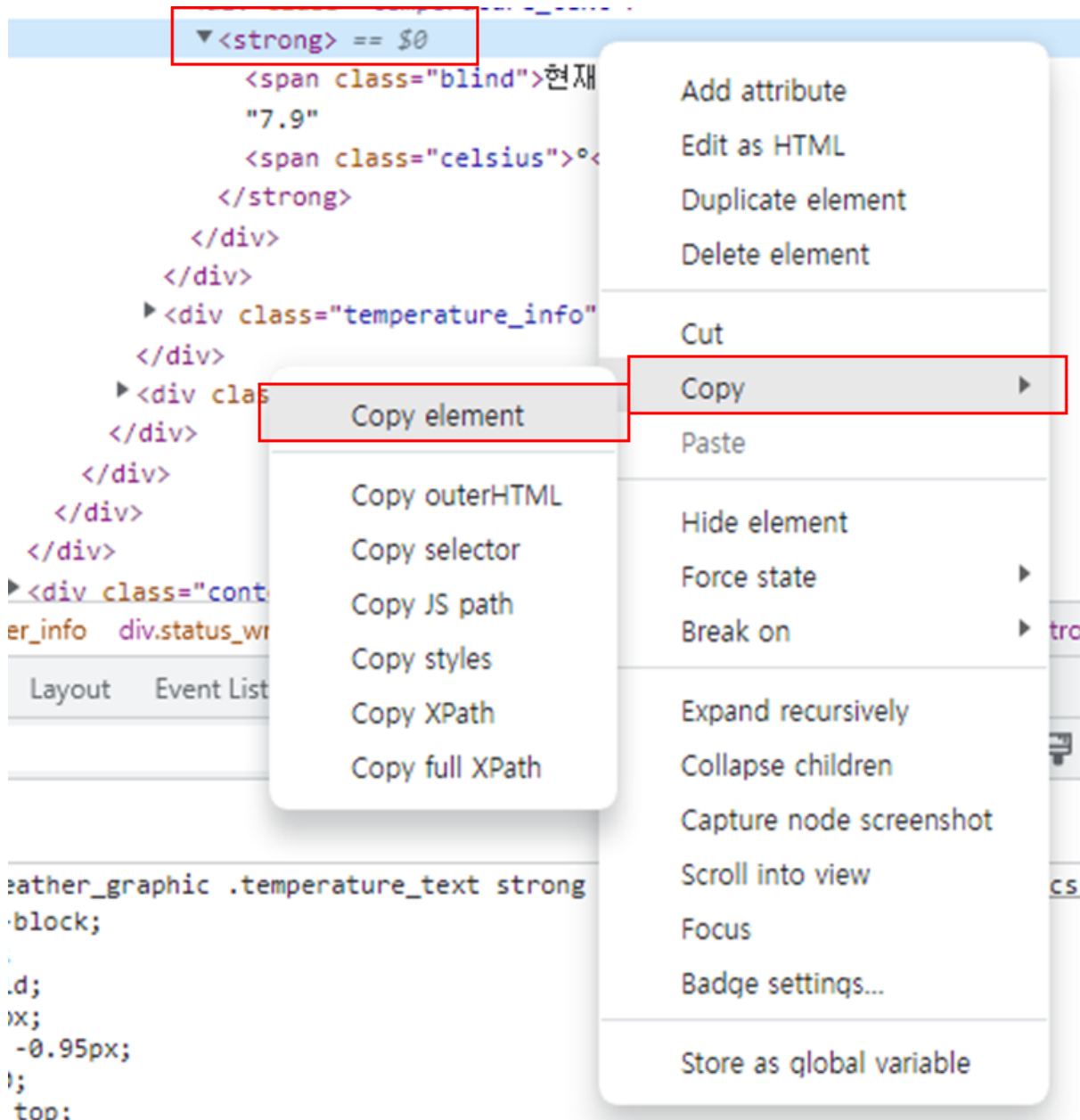
다음과 같은 화면을 볼 수 있다. html의 element를 확인할 수 있는 기능이다.



7.9도를 클릭하면 크롬 개발자 도구에 해당 element를 확인할 수 있다.



에 마우스 우클릭을 하여, Copy → Copy element를 하자.



이번에도 Visual Studio로 붙여넣기를 하면, 아래와 같이 확인할 수 있는데,

```

71
72 <strong><span class = "blind">현재 온도 < / span>7.9 < span class = "celsius" > ° < / span>< / strong>
73

```

크롤링한 결과로 얻은 html 내용은 text 인데, 그 text 안에 저 내용이 들어 있고,

우리는 문자열 "현재온도" 과 "" 사이에 있는 7.9 라는 온도 값을 파싱해내면 된다는 의미이다.

HttpRequest() 함수

```

void HttpRequest()
{
    //url 인코딩
    wstring search_gu, search_dong;
    cout << "어디 날씨가 궁금해? 00구 00동 으로 입력하자.";
    wcin >> search_gu >> search_dong;
    wstring search = search_gu + U("+") + search_dong + U("+") + U("날씨");
    wcout << search << endl;
    wstring url = U("https://search.naver.com/search.naver?where=nexearch&sm=top_hy&fbm=0&ie=utf8&query=");
    url += search;
    wcout << url << endl;

    string utf_8_url = url_encode(to_utf8string(url));

    wstring path;
    path.assign(utf_8_url.begin(), utf_8_url.end());
    wcout << path << endl;

    //크롤링
    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;
    auto html = get_resp.extract_string(true).get();
    //wcout << html << endl;

    //파싱
    wstring find_st = U("현재 온도</span>");
    wstring find_en = U("<span class=\"celsius\">");
    auto index_st = html.find(find_st);
    auto index_en = html.find(find_en);
    string::npos;
    wstring celsius;
    celsius.assign(html.begin() + index_st + find_st.size(), html.begin() + index_en);

    wcout << U("현재 온도 : ") << celsius << endl;
}

```

프로젝트 완성 코드

```

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cpprest/http_client.h>
#include <cpprest/filestream.h>
#include <cpprest/json.h>
#include <string>
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;
using namespace utility::conversions;

string url_encode(string origin) {
    string ret = "";
    for (int index = 0; index < origin.size(); index++)
    {
        if (!(origin[index] & 0x80)) {
            ret += origin[index];
            continue;

```

```

        } // ascii code
        unsigned char el = origin[index];
        char temp[4];
        sprintf(temp, "%%%2X", el);
        ret += temp;
    }
    return ret;
}

void HttpRequest()
{
    //url 인코딩
    wstring search_gu, search_dong;
    cout << "어디 날씨가 궁금해? 00구 00동 으로 입력하자.";
    wcin >> search_gu >> search_dong;
    wstring search = search_gu + U("+") + search_dong + U("+") + U("날씨");
    wcout << search << endl;
    wstring url = U("https://search.naver.com/search.naver?where=nexearch&sm=top_hyt&fbm=0&ie=utf8&query=");
    url += search;
    wcout << url << endl;

    string utf_8_url = url_encode(to_utf8string(url));

    wstring path;
    path.assign(utf_8_url.begin(), utf_8_url.end());
    wcout << path << endl;

    //크롤링
    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;
    auto html = get_resp.extract_string(true).get();
    //wcout << html << endl;

    //파싱
    wstring find_st = U("현재 온도</span>");
    wstring find_en = U("<span class=\"celsius\">");
    auto index_st = html.find(find_st);
    auto index_en = html.find(find_en);
    string::npos;
    wstring celsius;
    celsius.assign(html.begin() + index_st + find_st.size(), html.begin() + index_en);

    wcout << U("현재 온도 : ") << celsius << endl;
}

int main()
{
    //freopen("output.txt", "w", stdout);
    wcin.imbue(locale("kor"));
    wcout.imbue(locale("kor"));

    HttpRequest();
    return 0;
}

```