**최종 인수인계 보고서**

**2024.07 ~ 08**

**서원형 실습생**

목 차

[1. 개요 1](#_Toc175744746)

[1.1. 배경 1](#_Toc175744747)

[1.2. 용어 정리 3](#_Toc175744748)

[1.3. 문제점 요약 6](#_Toc175744749)

[2. 구조 9](#_Toc175744750)

[2.1. 전체 구조 9](#_Toc175744751)

[2.2. c# 내 구조 11](#_Toc175744752)

[2.3. 크롬 익스텐션(js) 내 구조 12](#_Toc175744753)

[3. 각 문제점 및 해결방법 13](#_Toc175744754)

[3.1. frame 새로고침, reload 출력에러 13](#_Toc175744755)

[3.2. frame 페이지 에러 16](#_Toc175744756)

[3.3. 1pixel 통신 개선 18](#_Toc175744757)

[3.4. Edge 좌표 보정 20](#_Toc175744758)

[3.5. CSP 에러 해결 23](#_Toc175744759)

[3.6. NM 전체 스크린샷 25](#_Toc175744760)

[3.7. NM 전체 스크린샷 28](#_Toc175744761)

[4. 추가정보 30](#_Toc175744762)

[4.1. 익스텐션웹 스토어 업데이트 방법 30](#_Toc175744763)

[5. 남은 문제점 33](#_Toc175744764)

[5.1. 익스텐션 UIPath 충돌 문제 33](#_Toc175744765)

[5.2. <frame> 보정 35](#_Toc175744766)

[5.3. 하이라이트 실행 불가 35](#_Toc175744767)

[5.4. 구조 개선점 35](#_Toc175744768)

[5.5. 무분별 로그 37](#_Toc175744769)

[5.6. 이미지 셀렉터 37](#_Toc175744770)

# 1. 개요

## 1.1. 배경

2024년 7월, 전임자인 이승우 실습생으로부터 받은 최종인수인계를 받아 셀렉터의 문제를 착수하게 되었다.

문제를 수정하던 중 Selector와 관련된 추가 문제점들을 발견했고 크롬 익스텐션(js)와 beBOT 소스코드(C#)을 수정해 해결하였다

각 위원님이 지시한 과제에 대해서는 과제보고서로 작성한바 있다.

|  |
| --- |
| **보고서명** |
| 과제 보고서\_서원형\_전임자토론\_01.docx |
| 과제 보고서\_서원형\_새로고침,reload\_frame\_출력에러\_02.docx |
| 과제 보고서\_서원형\_frame\_페이지\_04 .docx |
| 과제 보고서\_서원형\_전자약\_자료조사\_05.docx |
| 과제 보고서\_서원형\_selector\_1pixel\_개선\_06.docx |
| 과제 보고서\_서원형\_Edge\_selector\_좌표보정\_07.docx |
| 과제 보고서\_서원형\_전자약\_LLM 모델\_구현\_08.docx |
| 과제 보고서\_서원형\_Virtual\_click\_CSP\_09.docx |
| 과제 보고서\_서원형\_NM\_통신에러\_10.docx |
| 과제 보고서\_서원형\_NM\_전체\_스크린샷\_11.docx |

Selector를 개발하는데 있어 문제점에 의해 어려움을 겪었다.

**(1) 코드가 의미없이 작성되어 있는 부분이 많고 모듈화가 되어 있지 않다.**

각 코드가 의미 없이 깨진 독에 반창고 붙히듯 해결되어 있고, 모듈화가 제대로 되어 있지 않아 코드에 대한 이해, 로그 검출에 어려움이 있다.

돌아가기 위한 코드임은 분명하나, 리소스 최적화, beBOT이 비정상 종료되는 문제가 매우 빈번하게 발생한다.

**(2) 제대로된 구조적 이해 없이 개발한 경우가 생긴다.**

beBOT은 openRPA[[1]](#footnote-1)를 기반으로 개발한 소스코드인데, openRPA 소스 자체가 의미를 알 수 없는 코드가 많다.



*(openRPA – content.js에 있는 true가 false인지 체크하는 구문)*

이 외에도 인터넷 상에 무분별하게 돌아다니는 소스코드를 그대로 복사해 사용하는 등 의미 없는 코드 조각이 굉장히 많기 때문에 이해하기 어렵다.

최종 인수인계를 통해 이루고자 하는 바는 다음과 같다.

1. 실습생의 Selector에 대한 이해

2. 정확한 인수인계

3. 개발 중 발생 가능한 문제상황에 대한 대처 방법

Selector는 크게 보면 실행을 담당한 동작부와 ‘녹색 오버레이’, ‘적색 highlight’를 표시하는 출력부로 나뉜다.

나는 동작부를 주로 개발하였기 때문에 출력부와 관련한 구조에 대해선 이 문서를 통해 크게 도움 받을 수 없을 것이다. (출력부와 관련한 문제점은 전임자가 맡아 수정했으나 종종 에러가 생김. 동작부가 조금 더 에러가 생겼을 때, 해결이 어렵다.)

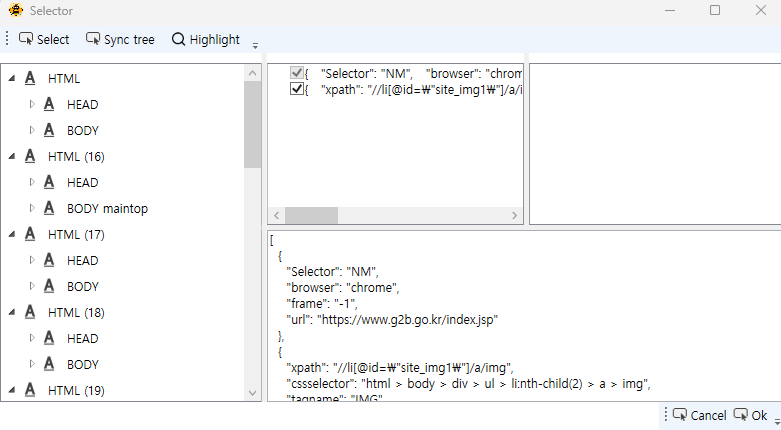
또한 기본적인 javascript , C#의 기본적인 지식을 알고 있다는 것을 전제로 작성한다. 웹 구성 방식, js로 요소 접근 방법, DOM, CORS, getBoundingClientRect(), preventDefault(), 이벤트 및 이벤트 리스너는 기본적으로 알고 있다 가정하고 작성한다.

개요 및 용어정리의 부분은 전임자가 문서정리를 해둔 것을 기준으로 차용하고, 해결한 오류 및 필요한 개발 사항을 최신화하여 전달하고자 한다.

익스텐션은 2024년 8월 기준 최신인 0.0.7 버전으로 설명한다.

## 1.2. 용어 정리

**[ Selector, 셀렉터 ]**



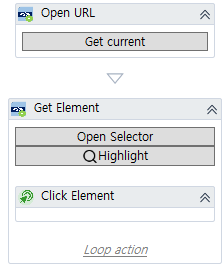
특정 요소를 선택하고 조작하기 위한 beBOT 내 도구이다. 자바 셀렉터, 윈도우 셀렉터, NM 셀렉터 등의 종류가 있으나 제대로 구현된 것은 윈도우 셀렉터와 NM 셀렉터이다. **이 보고서에서는 브라우저를 조작하는 NM 셀렉터를 뜻한다.**

비봇의 기능단위, 액티비티 중 대표적으로 많이 사용하는 Get Element도 셀렉터를 사용한다.



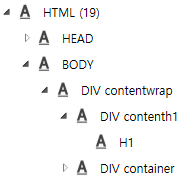
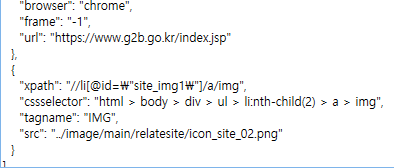
Select 버튼 : 요소 선택 Highlight 버튼 : 선택한 요소 강조 표시

셀렉터를 통해 웹 브라우저 내 마우스 클릭한 요소의 정보를 가져올 수 있다. 이 정보를 토대로 Click element 등의 액티비티를 연결시켜 웹 내 조작을 만든다.



*자주 쓰는 형태로 인터넷 열기 – 요소 가져오기 – 클릭 실행*

**[ 요소 ]**

[[2]](#footnote-2)[[3]](#footnote-3)

Selector를 통해 선택할 수 있는 단위이며, 웹브라우저 내 DOM을 통해 인식할 수 있는 Element와 동일하다.

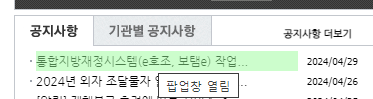
**[ select ]**

Selector의 select 버튼을 통해 선택하는 행위이다. ‘버튼A를 클릭한다’를 실행하기 위해 ‘버튼 A’ 요소를 Selector를 통해 지정하는 과정이 **select** 이다.

이 보고서에서는 select 버튼을 누른 뒤 beBOT 창이 최소화되며 녹색 오버레이가 나오는 상황과 웹 내 요소를 클릭하는 행위 뜻한다.



**[ 녹색 오버레이 ]**



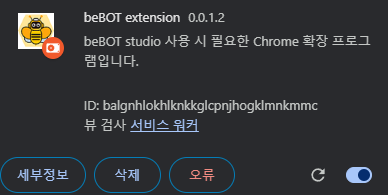
Selector의 select를 통해 웹 브라우저 내에서 마우스 위에 요소를 선택하는데 해당 요소의 위치 및 크기를 녹색 오버레이로 출력시켜준다.

**[ Highlight, 적색 Highlight ]**



Selector UI 의 “Highlight” 버튼을 통해 실행할 수 있으며 이전에 select한 요소의 위치와 크기를 적색 오버레이로 출력시켜준다.

**[ 익스텐션 ]**



beBOT의 웹 브라우저 내 데이터를 얻어오기 위한 크롬 익스텐션이다.

content.js, background.js, manifest.json[[4]](#footnote-4)(설정 파일)로 구성되어있다.

## 1.3. 문제점 요약

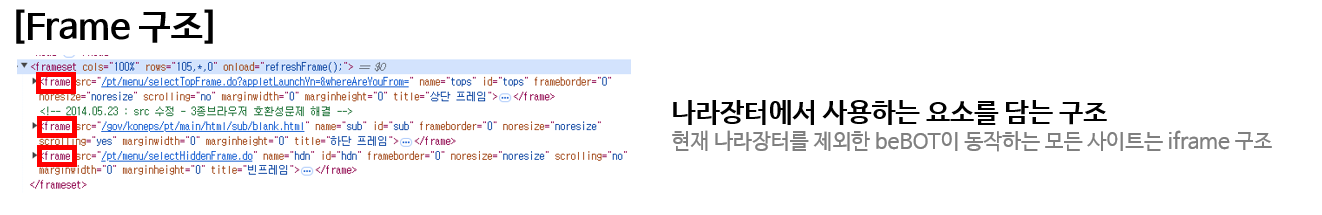
셀렉터는 오류가 굉장히 많이 나고, 전임자마다 개발 방식도 달라서 코드에 익숙해지기가 어려운 점이 있다. 또한, 통신구조가 굉장히 복잡해 파악에 많은 시간이 소요된다.

큰 부류로 3가지의 문제점을 해결했다.

**(1.3.1) 요소가 frame 내부에 있을 경우**

**(1.3.2) 엣지 브라우저를 사용할 경우**

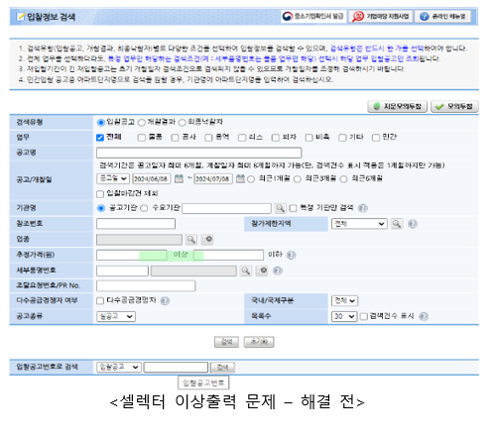
**(1.3.3) beBOT과 통신이 필요한 액티비티 개발이 필요한 경우**



*(우선 나라장터의 Frame 구조이다.)*

입찰 공고 번호에 오버레이가 출력되어야 정상이지만 화면 상단에 출력된다.

해당 문제의 원인은 나라장터가 frame 내에 속해있기 때문이다.



frame 내의 요소의 경우 전임자가 iframe에 대해서는 많은 에러를 해결하고 고쳤으나 오래된 구조인 frame에 대해서는 코드 상 설계가 올바르지 않아 나라장터 사이트에 오버레이 이상 출력 문제가 있었음.

이 보고서 이전 시점에 남은 오버레이 이상 출력은 다음과 같다.

1. beBOT과 셀렉터가 계속 이상한 X, Y좌표를 갖고 정보를 송수신 하는 것.

2. iframe구조와 frame 구조를 분리하여 크롬 익스텐션을 설계하지 않았다.

**(1.3.2) 엣지 브라우저를 사용한 경우**



위 Edge 브라우저에 대해 셀렉터가 좌표를 아예 잘못 짚어 beBOT의 정상적인 실행이 불가능한 상태로, 브라우저가 Edge임을 확인하고, Edge에 알맞은 보정 코드를 넣어줘야함



위 코드는 현재 사용자의 브라우저가 Edge임을 인식하여 보정을 추가하기 위해 처리한 로직이다.

beBOT과 익스텐션이 연결하여 메시지를 주고 받을 때, 익스텐션에서 해당 브라우저의 정보를 포함하게 코드를 구현하였다. Chrome과 Edge, FireFox를 모두 한번에 가져올 수 있는 코드가 존재하지 않았기 때문에 Edge 브라우저를 가져오는 것을 document를 통해 확인하고, 코드를 가져왔다.

**(1.3.3) NM 통신이 필요한 액티비티 개발이 필요한 경우**

beBOT과 익스텐션이 연결하여 메시지를 주고 받으며, 액티비티를 개발해야 하는 필요가 있다. 이때 내가 개발한 것은 전체 스크린샷을 캡쳐하는 액티비티이며 3장에서 자세히 설명하도록 하겠다.

# 2. 구조

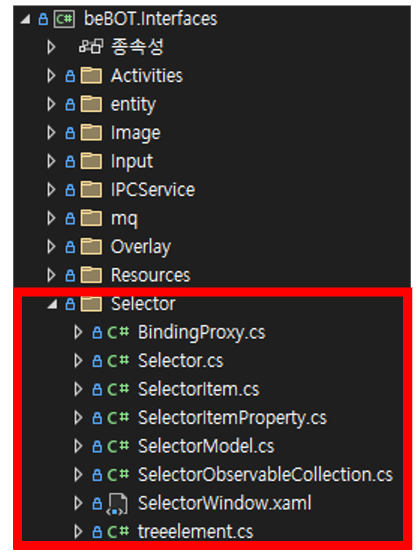
## 2.1. 전체 구조

※ 2.1장의 ‘구조’는 셀렉터에서 요소의 좌표를 얻는 과정을 다룬다.

셀렉터는 익스텐션의 3개 파일과 C#의 2개 프로젝트로 주로 구성된다. (NM 셀렉터 기준)

|  |  |  |
| --- | --- | --- |
| 구분 | 파일/프로젝트명 | 역할 |
| 익스텐션 (js) | content.js | 좌표 계산, 실행 등 실질적 구현부 |
| background.js | 포트 연결, 메시지 최종 전송 |
| manifest.json | 웹 익스텐션 설정 파일 |
| beBOT (C#) | beBOT.Interfaces  (內 Selector 폴더) | Selecor 인터페이스 구현 파일 (디자인, item 인식 등)”  \*NM 셀렉터, 윈도우 셀렉터에서 공통적으로 사용 |
| beBOT.NM | NM 셀렉터 액티비티, 연결 구현 |

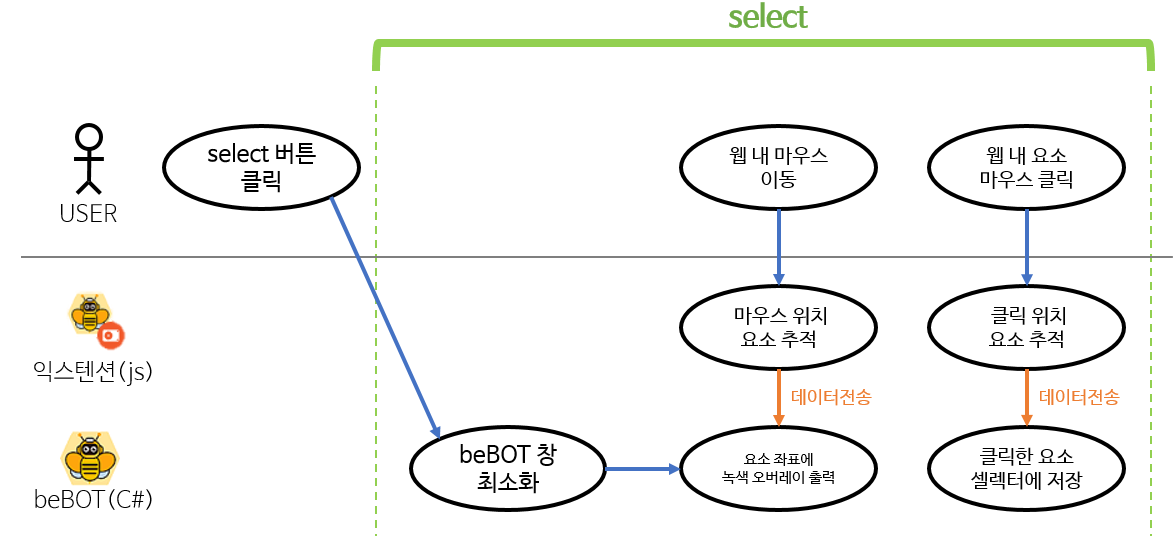
주로 개념 설명 부분은 전임자의 인수인계 문서에 출력부가 잘 정리되어 있어서 차용하였고, 나는 주로 동작부를 개발하여 문제해결에 더 집중하려고 한다. 만약 이에 대한 설명이 더 필요하다면 이승우 셀렉터 종합개선 보고서를 참고하면 된다.



개발 중 조심해야할 사항으로 beBOT.Interfaces\Selector 폴더 내 파일들을 수정 시 NM 셀렉터 및 윈도우 셀렉터 양쪽에서 모두 사용하기 때문에 문제가 발생하지 않았는지 확인해야한다.

셀렉터에 대한 전체 동작 구조를 확인하려면 중단점을 찍고 디버그로 프로시저 단위로 실행(VS기준 F10)하며 확인하는 것이 이해하기 좋다. select 한 가지 동작에도 파일 5~7개를 돌아다니며 동작하기 때문에 디버그를 통해 확인하는 것이 이해하기 쉽다. js도 동일하게 분석한다.

select 과정은 다음과 같이 이루어진다.



1.1. (사용자) 셀렉터의 select 버튼 클릭

1.2. (C#) beBOT Studio 및 셀렉터 창 최소화

2.1. (사용자) 웹 브라우저 내 마우스 이동

2.2. (js) 마우스 오버 [[5]](#footnote-5)된 요소 추적, 좌표/크기 수집해 C#으로 데이터 전송

2.3. (C#) 전송받은 좌표/크기에 녹색 오버레이 출력

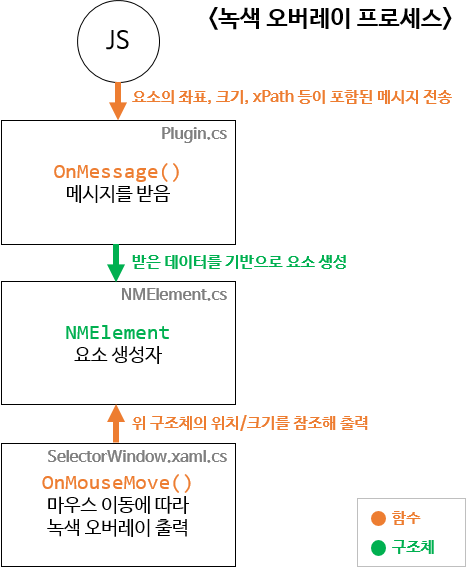
3.1. (사용자) 웹 내 요소 마우스 클릭

3.2. (js) 클릭된 위치의 요소 추적

3.3. (C#) 녹색 오버레이 출력종료 및 클릭한 요소를 셀렉터 내 구조체로 저장

## 2.2. c# 내 구조

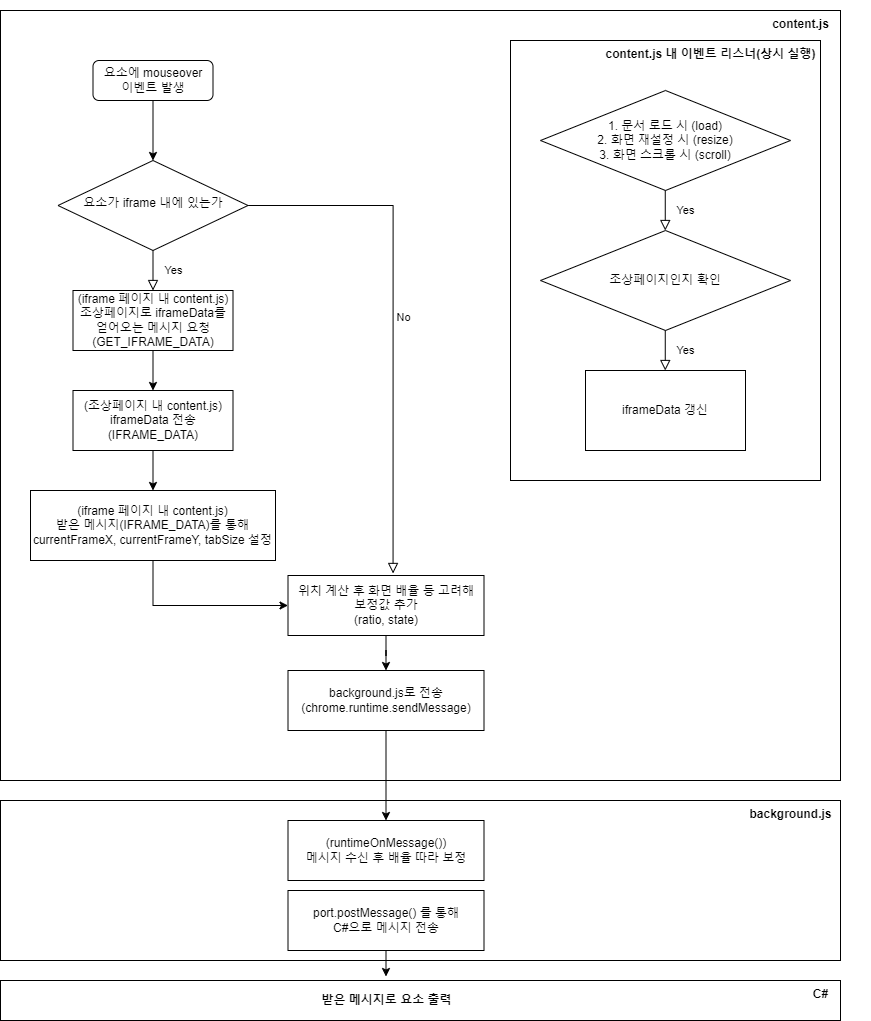
C#에선 데이터를 받고 해당 데이터대로 오버레이를 출력하게 된다.



녹색 오버레이를 출력하는 프로세스의 핵심 구조체 및 함수는 위와 같다.

메시지를 받고 파싱 후 원하는 좌표, 크기를 뽑아내는 과정 등도 있으나 수정된 내용이 없기 때문에 이 보고서에는 기술하지 않고 수정한 핵심 기능을 중심으로 기술한다.

## 2.3. 크롬 익스텐션(js) 내 구조



화면에 오버레이를 출력하기 위해 좌표와 크기를 얻는 과정의 전체 흐름은 위와 같다.

녹색 오버레이와 highlight는 C# 내에선 다르게 처리하지만, 좌표와 크기를 얻는 js 에서는 highlight에 요소 추적이 들어가는 점을 제외하고는 같다.

# 3. 각 문제점 및 해결방법

**# 7~8월 해결 문제점**

이 보고서에서는 다음과 같이 설명한다.

|  |  |
| --- | --- |
| **소제목** | **해결한 문제점** |
| **3.1. frame 새로고침, reload 출력에러** | - 녹색 오버레이 이상 출력  - frame 비동기 처리 |
| **3.2. frame 페이지** | - URL 미이동시 대응  - frame 로드 시점 변경 |
| **3.3. 1pixel 통신 개선** | - 1 pixel 통신 개선 |
| **3.4. Edge 좌표 보정** | - Edge pixel 보정 |
| **3.5. CSP 에러 해결** | - 마우스 이벤트/js 동작 문제 |
| **3.6. NM 전체 스크린샷** | - 전체 화면 스크린샷 개발 |
| **3.7. NM 통신 불가** | - 통신 불가 원인 분석 |

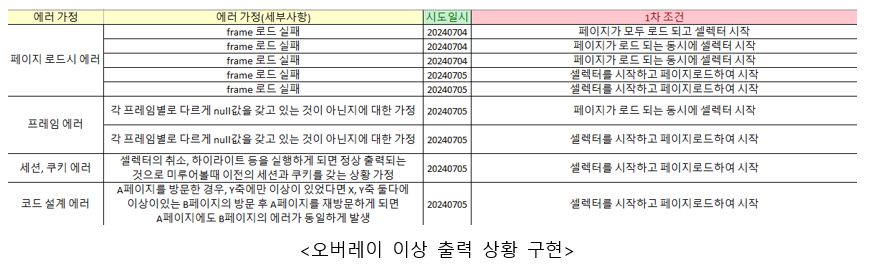
## 3.1. frame 새로고침, reload 출력에러

frame 페이지를 새로고침하거나, 로드되는 특정 시점에 셀렉터가 잘 짚이지 않는 부분을 발견했다.



우선 문제 상황을 재현하고, 다양한 시도를 통해 문제를 규명하였다.

생각한 방안으로는 1. 페이지 로드시 에러, 2. 프레임 에러, 3. 세션, 쿠키 에러, 4. 코드 설계에러로 구분하여 다양한 상황에서 문제를 구현해보고 로그를 통해 코드를 역추적하였다.



위는 일부 예시이고, 이런식으로 했다~ 정도만 알고 넘어가면 추후에 셀렉터에서 알 수 없는 에러가 발생했을 때, 에러 지점을 찾기 용이할 것이라고 생각한다.

하지만,위 표로만은 이것으로는 에러를 규명할 수 없었고, frame 요소의 특성에 대해 조금 더 깊게 찾아보았다. 이때, 최신 구조인 iframe보다 느리며 페이지 로드에 시간이 오래 걸릴 수 있다는 점을 착안하여 페이지가 로드되는 네트워크 부분을 확인하였다. 또한, 간헐적으로 원인 불명한 에러가 발생했기 때문에 네트워크 로드 속도에 따라 에러가 발생할 수도 있겠다고 생각하였다.



위 두 사진에서 볼 수 있듯, 간헐적으로 네트워크가 느려져 frame이 늦게 도착하여 에러가 발생하는 것으로 정확한 원인을 규명할 수 있었고 이에 대한 대응책으로 frame을 불러오는 코드는 비동기 처리를 통해 모든 frame이 로드된 이후 셀렉터에 정보를 보내는 함수를 실행하는 방향으로 코드를 설계하였다.



비동기처리를 통하여 frame이 로드되지 않을 경우에는 beBOT으로 데이터를 보내지 않고 정상적으로 frame이 로드되고 그 때부터 셀렉터의 오버레이가 정상작동되게 구현하였다.

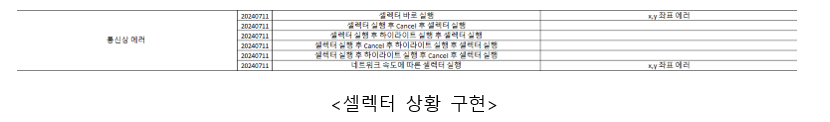


위 그림과 같이 새로고침을 하거나 reload를 하여도 정상적으로 출력된다. 네트워크 속도에 따라 차이가 있지만, 로드되지 네트워크로 인해 정상적으로 받아올 수 없었던 frame 정보가 있다면, 다시 frame 정보를 reload하는 비동기 로직을 구현하여 모두 정상적으로 구현된다.

## 3.2. frame 페이지 에러

frame 페이지 에러는 우선 문제 상황을 재현하고, 셀렉터와 beBOT의 통신 구조를 보며 해결하였다.

셀렉터를 바로 켰을 경우에는 통신이 되지 않지만, cancel 버튼을 누르거나 하이라이트 버튼을 누르고 다시 셀렉터를 켰을 경우에는 정상적으로 통신이 되는 점을 발견하고 하나씩 중단점을 찍어보며 로그를 실행하였다.



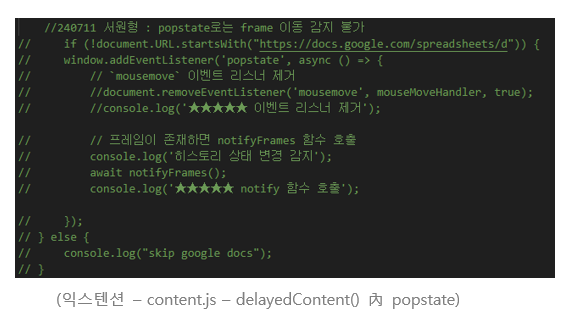
이때, 코드 설계 상의 에러가 있음을 확신하여 loop를 따라가며 확인해보았다. 단, iframe에서는 잘 작동했고 frame에서 작동하지 않기 때문에, frame의 구조의 차이점에 주목했다. frame의 구조는 초기 load를 제외하면 frame 상에서 바뀌는 것은 load로 이벤트를 발생시키지 않았다. 따라서 notifyFrame 함수가 재실행되지 않았고, 초기화 되지 않아 정상적인 X, Y좌표를 beBOT에게 전달할 수 없었던 것이다.

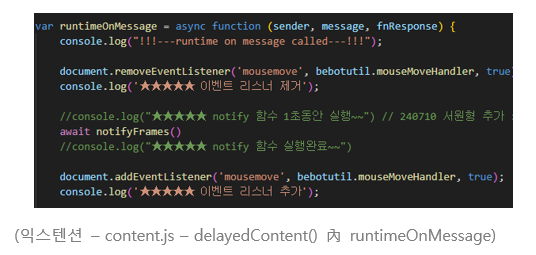
따라서 해결방안을 2가지로 잡았다.

1. beBOT이 셀렉터를 연결할 때 익스텐션의 X,Y좌표의 부모가 변하지 않았다면 리로드를 요청하는 방법.

2. 페이지가 이동할 때(마우스 클릭, popstate, reload등)을 감지하여 notifyFrame 함수를 재 실행시켜 항상 최신 상태를 유지하는 방법이었다.

현재 로그의 메시지가 무분별하게 나와 파악이 어렵고 마우스 move 이벤트로 비효율적인 통신을 하고 있기 때문에 우선적으로 항상 최신 상태를 유지하려고 하였다. 그렇지만, 마우스 클릭이 아닌 popstate 같은 리로드 함수로 frame의 변화를 감지 할 수 있는 가능성을 찾아보고 직접 코딩해 보았다.



****이때 popstate는 frame의 로드 상태를 받아올 수 있었으나, 네트워크 상태에 따라 frame이 느리게 로드되면 결국 에러가 발생하고 간헐적으로 정상적으로 작동하지 않는 경우가 있어 폐기하였다.

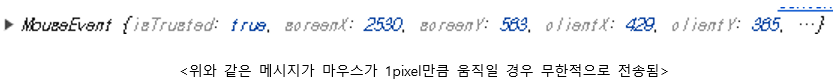
따라서 1안이었던, beBOT이 통신을 요청할 때 함수를 재실행하는 방식을 선택했다. 이때, beBOT과의 통신을 통해 select 버튼을 클릭함을 감지한다. 사용자의 시나리오로는 셀렉터 버튼을 사용해야 녹색 오버레이가 보인다. 따라서 이때 함수를 재실행하고 만약 네트워크가 정상적으로 로드되지 않는다면 해당 오버레이에 대하여 리로드 로직을 하여 결국 정상적으로 사용자가 이용할 수 있게 코딩하였다.

****

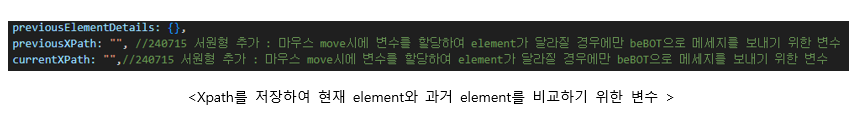
frame이 존재하는 페이지에서 frame이 네트워크 에러로 인하여 정상적으로 로드되지 않더라도 리로드 로직을 통해 셀렉터가 정상적으로 작동한다.

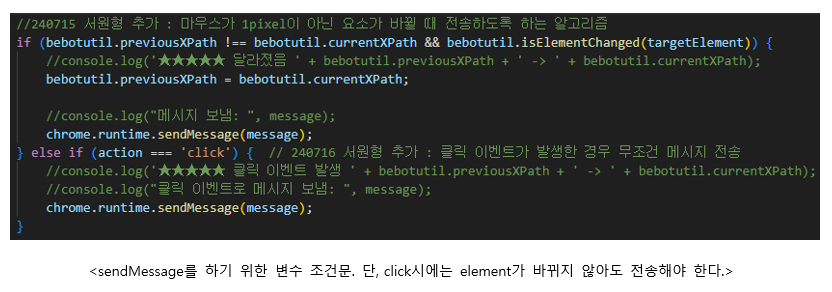
## 3.3. 1pixel 통신 개선

beBOT 익스텐션의 경우 마우스 move를 감지할 경우 해당 좌표를 바로 beBOT에게 전송하여 같은 element의 경우에도 좌표를 지속적으로 전송하여 불필요한 리소스가 소모되는 현상이 발생하였음

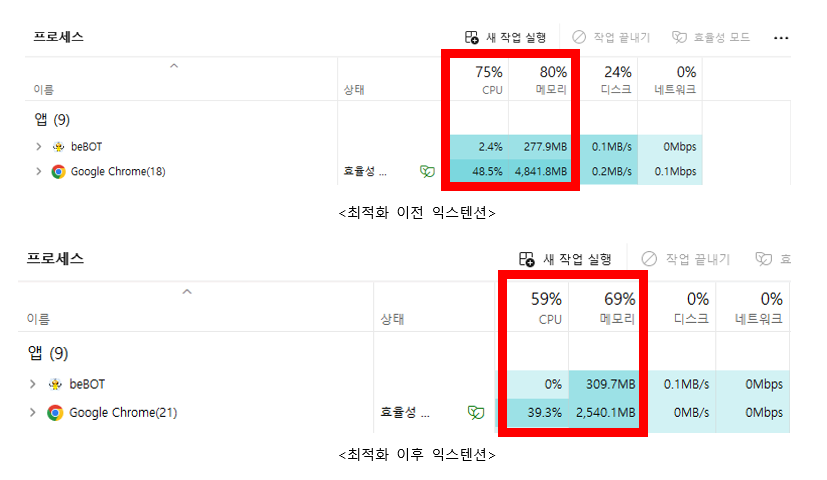


message를 보내는 주기를 mouse move가 아닌 element가 변할 경우로 바꾸었다. 이때, 이전 mouse가 가르키고 있던 element를 과거 element라고 하고 현재 mousr가 가르키고 있는 element를 현재 element라고 할 때 두 element가 같으면 메시지를 보내지 않고 다르면 보내는 로직으로 변경하였다. 이때, 함수를 계속 비교해야한다는 단점은 있으나 beBOT과의 지속적인 통신에 비한 리소스 소모량이 적다고 판단하여 진행하였다.





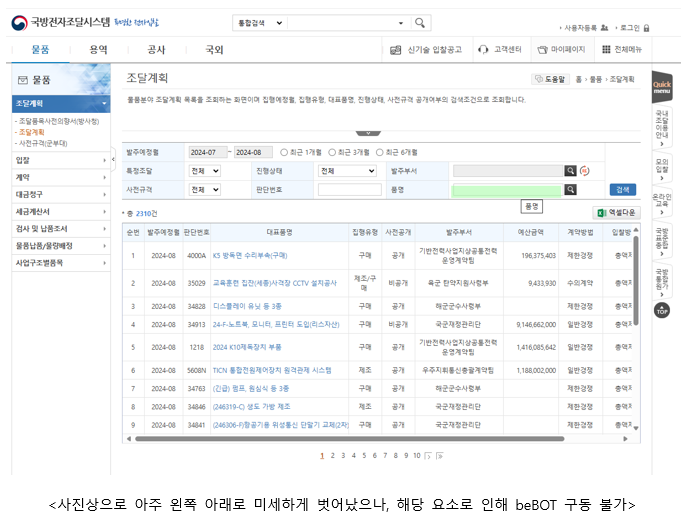


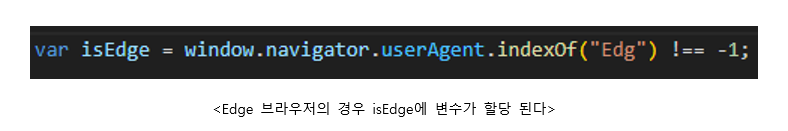
나라장터 입찰정보 검색의 페이지에서 beBOT을 이용해 마우스 이동을 반복한 결과 감지된 Event와 연계되는 beBOT으로의 메시지 전송이 1831건에서 463개로 25%로 줄었다. 최대한 동일한 환경에서 구현하려고 했으나 완벽하게 동일하진 않을 것이다. 정확한 수치의 차이가 있을 수는 있지만 보내고 있는 message의 양이 줄어 자원을 덜 소모하는 것 만큼은 명확하다.

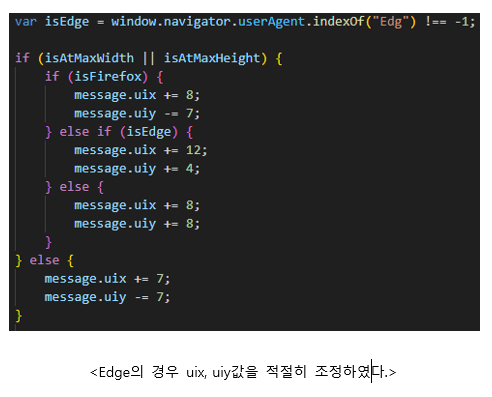
메모리 활용 부분에서 80% 이상을 절감하였고, CPU 소모율도 20% 개선하였다. 최적화 이전 익스텐션으로 마우스를 빠르게 움직이면 CPU 100% 사용률을 보였고, 일부 컴퓨터에서 다운되는 현상이 발생하여 고치고자 하였다. 환경은 최대한 동일하게 구성하였고, 환경의 차이가 있음을 감안하더라도 통신이 적어지며 효율적으로 코드가 돌아갈 수 있게 되었다.

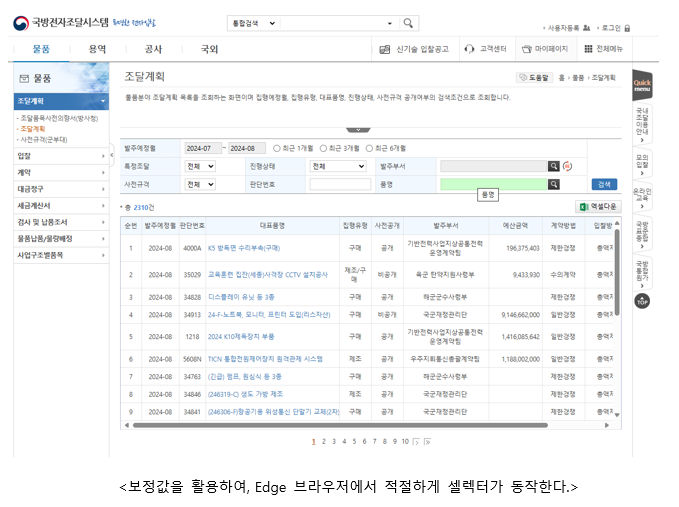
## 3.4. Edge 좌표 보정

Edge 브라우저에 대해 셀렉터가 좌표를 아예 잘못 짚어 beBOT의 정상적인 실행이 불가능한 상태로, 브라우저가 Edge임을 확인하고, Edge에 알맞은 보정 코드를 넣어줘야함.



beBOT과 익스텐션이 연결하여 메시지를 주고 받을 때, 익스텐션에서 해당 브라우저의 정보를 포함하게 코드를 구현하였다. Chrome과 Edge, FireFox를 모두 한번에 가져올 수 있는 코드가 존재하지 않았기 때문에 Edge 브라우저를 가져오는 것을 document를 통해 확인하고, 코드를 가져왔다. 





서연진 실습생이 제기한 문제가 되는 국방전자조달시스템에서 보정이 이루어졌다. 따라서 셀렉터가 좌표를 적절히 반환하여 beBOT 동작이 가능해진 모습이다.

[남은 문제점]

Edge에 대하여 보정좌표도 존재하지 않았고, QA가 되어 있지 않다.

위의 국방전자조달시스템에서 에러가 나서 고쳤지만, 확인하지 못한 다른 사이트에서 추가로 보정이 이루어지거나, 로직을 변경해야 할 경우도 있을 것이라고 생각한다. 많이 써보고, 빠르게 고치는 것이 최선의 방법이라고 생각한다.

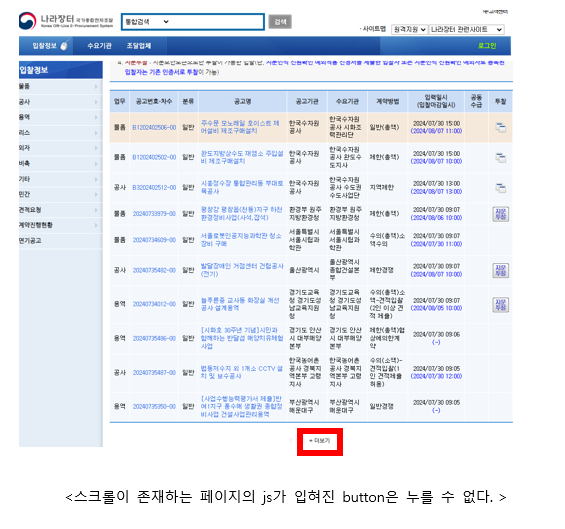
## 3.5. CSP 에러 해결

전임자 과제로 Vitrual Click이 되지 않아 나라장터 사이트에서 클릭을 할 경우에는 Virtual Click을 쓰지 않는 방법으로 우회하였으나, 스크롤등이 포함된 사이트에서는 X, Y좌표를 정상적으로 찍어 올 수 없어 OpenURL로 계속 이동을 한 상태임(스크롤이 뜨지 않고 한번에 뜨기 때문).

Virtual Click은 요소의 X,Y좌표를 바탕으로 해당 좌표에 클릭을 발생시켜 요소의 정보를 가져오지 않는 것이다.

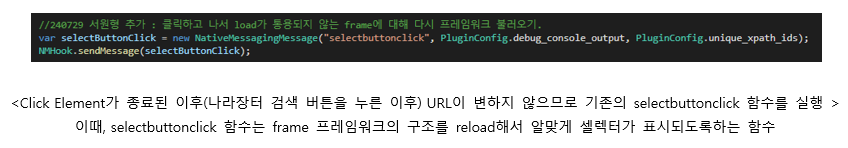


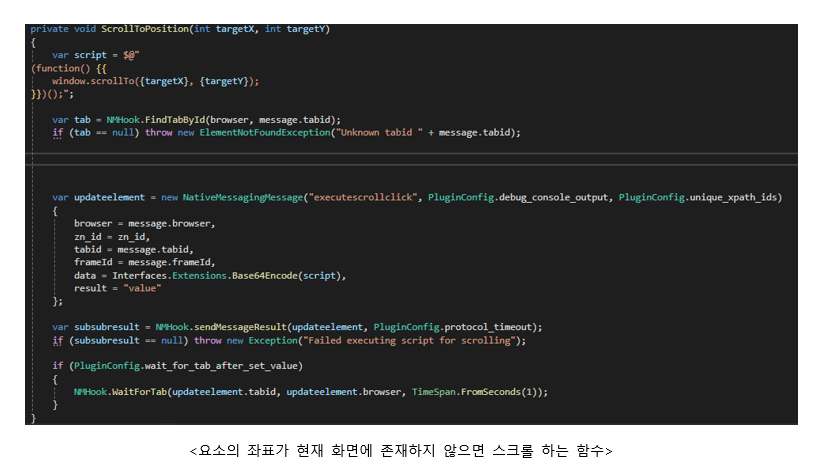
예시 상황 1



예시 상황 2

X, Y좌표를 통해 기존에 셀렉터에 저장된 좌표가 화면 안에 없으면 해당 좌표까지 스크롤을 하고 클릭을 하는 방안으로 생각하였다. 이때, 간과한 점이 있었다. 나라장터는 Frame 요소로 URL이 이동하지 않아 요소를 재불러오는 로직이 없어 X, Y좌표를 찍을 때 이상한 좌표를 찍을 수 밖에 없었다. 따라서 재요청 로직을 보내고, 페이지가 다시 한번 로드된 이후 스크롤을 내려 X, Y좌표를 찍는 방법으로 구현하였다.



재요청 로직을 보내고, 페이지가 다시 한번 로드된 이후 스크롤을 내려 X, Y좌표를 찍는 방법으로 구현하였다. 



## 3.6. NM 전체 스크린샷

제안요청 봇을 사용 중 화면 캡쳐가 필요한 부분이 존재한다. 이때, 기존 스크린샷은 해당 모니터의 좌표를 기준으로 상단 요소를 캡쳐하기 때문에, 인터넷 브라우저에 스크롤이 있다면 해당 요소를 캡쳐할 수 없다.



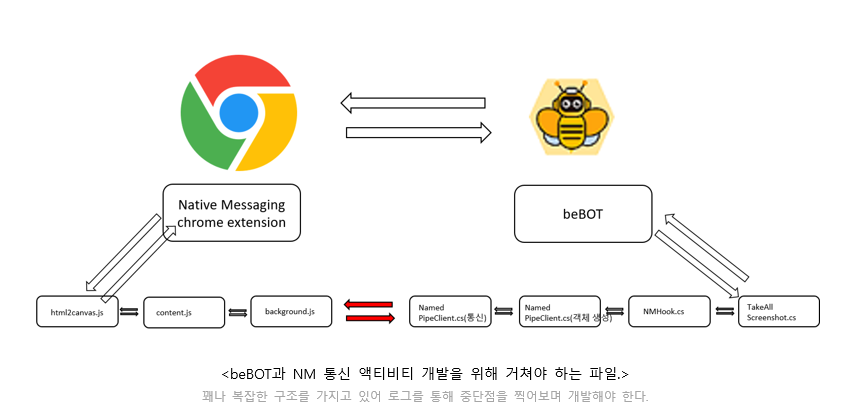
해결 방법에는 2가지 해결 방법이 있었다.

1. html2canvas 패키지를 사용하여 beBOT에 신규 액티비티 생성 – 개발 난이도 : 상

2. beBOT에 스크롤 액티비티를 만들어 화면마다 스크롤 해가며 캡쳐하고 콜라주하는 방법- 개발 난이도 : 중

2번의 방법은 사용자 입장에서 너무 비효율적이고, 코딩할 액티비티를 추가하는 단점이 있으며, 모든 사이트에 통용되기 어렵다는 결론에 1번 방법으로 시도하였다. 다만, NM과 통신하는 액티비티는 연관되어 있는 파일이 굉장이 복잡하여 먼저 Flow를 파악하고 코딩하는 것이 요구되었다.

HTML2CANVAS를 사용하고 있으므로, 해당 패키지의 업데이트(취약점)가 발견되면 폴더를 갈아줘야 한다. html2canvas github에서 js파일을 다운받아 chrome extension\lib 경로에 넣어주면 된다.

****

새 액티비티를 생성하여 NM 파트의 TakeAllScreenshot.cs가 추가 되었으며 NMHook.cs를 통해 chrome extension과 image 정보를 주고 받는다. 이때, 이미지는 base64 기반으로 인코딩, 디코딩 되어 주고 받는 로직을 구현하였다.

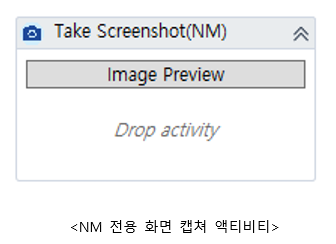
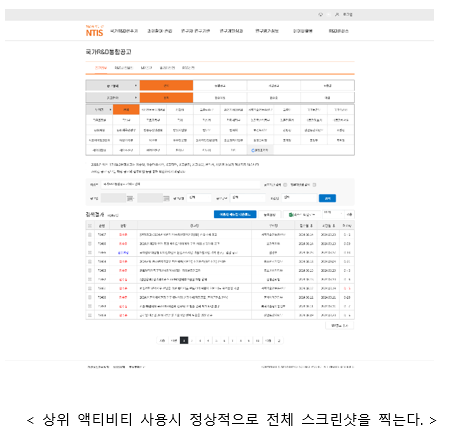
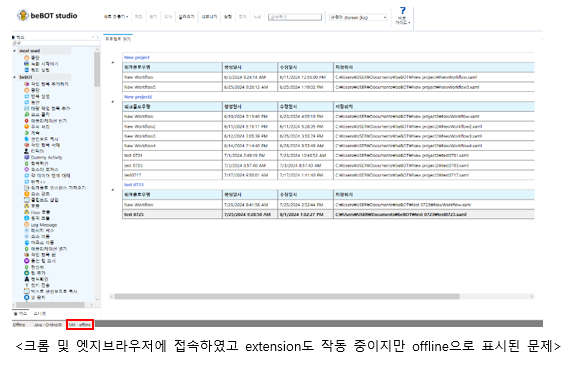


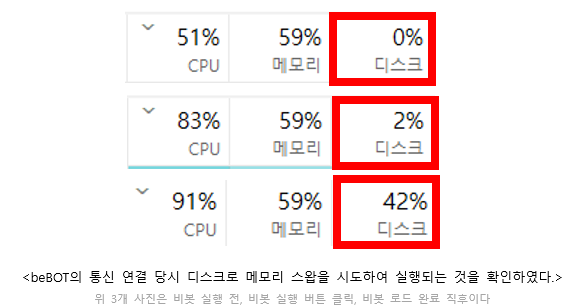
Image Preview를 통해 get url 이후 해당 화면 전체 캡쳐 예시를 확인할 수 있고, 해당 액티비티를 통해 정상적으로 image가 캡쳐 된다.



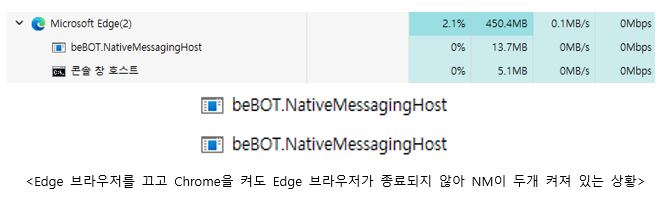
## 3.7. NM 연결

2가지의 에러가 NM 연결 관련하여 존재하였음. 간헐적으로 NM 연결이 되지 않는 문제와 간헐적으로 NM연결이 같은 컴퓨터에서 너무 잘되는 현상이었다. 간헐적이라는 에러는 개발자 입장에서 규명하기 굉장히 까다로운 문제였다.

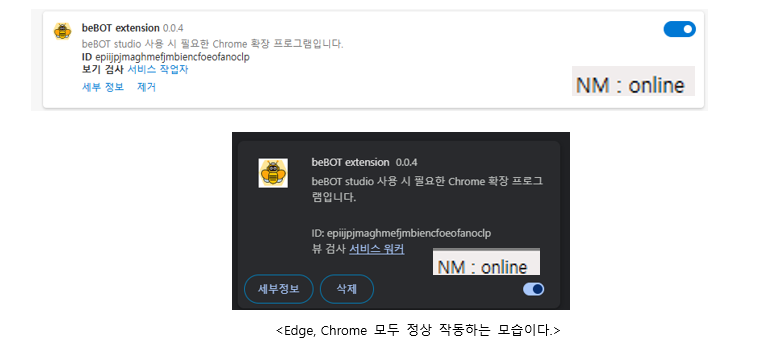


간헐적으로 에러가 발생하였으며, 이때 디스크의 용량을 주목했다. 자주(일 3회 이상) 에러가 발생하는 서연진 실습생과 에러가 나지 않는 배유선 실습생의 환경 차이는 용량차이가 있었고, 본 실습생과의 차이도 용량의 차이 였다. 다른 브라우저 버전, beBOT 버전등 모두 동일. 이로 인하여 추론한 바로는, Nmhost.exe파일이 CPU와 메모리가 부족할 경우 디스크로 메모리 스왑을 통해 가상 메모리를 쓰고 있지 않을까 하는 가정으로 접근하였다.

하드웨어의 한계로 Edge를 종료해도 NM host는 꺼지지 않는 경우가 종종 발생하고(보통의 컴퓨터면 Edge 관련 프로세스가 같이 종료 된다.) 이로 인해 background로 포트가 꺼진 Edge와 연결이 되어 있는 것이다. Chrome의 경우에는 바로 프로세스가 종료가 된다.



따라서 컴퓨터의 자원이 여유있을 경우에 beBOT을 실행시켜야 하며(그렇지 않으면 Nmhost가 실행되지 못해 연결을 할 수 없다.), 만약 그렇지 못한 경우에는 작업관리자에서 beBOT.Nmhost.exe를 찾고 강제 종료시키면 beBOT 內에서는 offline 상태가 되는데 이때 재연결을 요청하면 바로 연결이 가능하다.



결론적으로 문제는 디스크 용량 부족으로 인한 용량 스왑 불가, Nmhost.exe 정상 종료 이상으로 인한 무한 재연결 요청이었다.

디스크 용량 확보를 통해 스왑 불가 문제를 해결했다. 다만, 크롬 상에서는 정상 종료되는 Nmhost가 Edge에서만 종료가 되지 않는 경우가 있다. 이는 beBOT 실행과 무관하게 Edge에서 실행이 지속되어 beBOT 자체의 문제는 아니라고 판단하나(브라우저 內에서 익스텐션 로드가 불안하다. Edge 브라우저에서 Chrome의 익스텐션을 사용) 예방책을 세운 것이지 새로운 특정 상황에서 연결이 불안정할 수 있는 여지가 있다.

# 4. 추가정보

## 4.1. 익스텐션웹 스토어 업데이트 방법

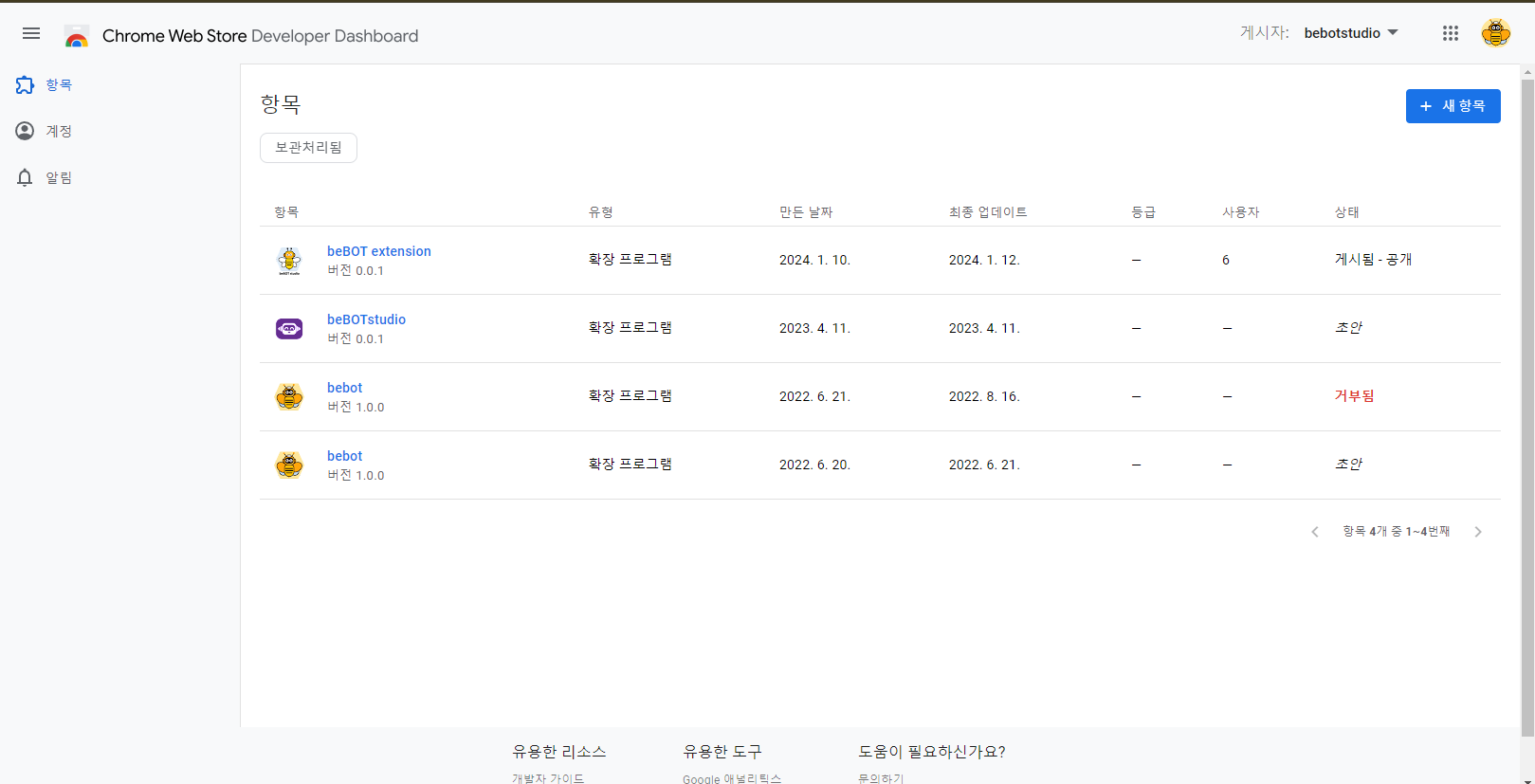
※ 박진혁 전임자의 Extension 업데이트 등록 방법.pptx[[6]](#footnote-6)를 참고하였다.

**1. 구글 로그인**

ID : bebotstudio@gmail.com PW : bluegood2022

**2. 크롬 웹스토어 개발자 대시보드 접속 및 beBOT extension 패키지 클릭**

<https://chrome.google.com/webstore/devconsole>

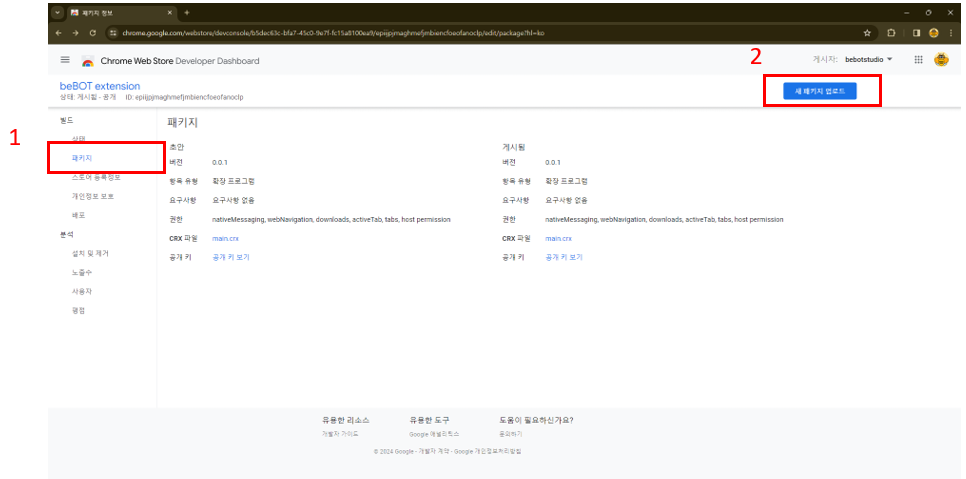


**3. Manifest.json 파일 수정 (버전 정보)**



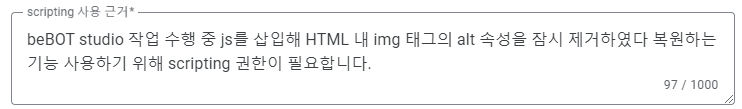
version 을 수정해야한다.

**4. 패키지 등록(업데이트)**

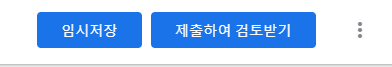


위 버튼의 순서에 따라 업로드를 진행하면 된다.

이 때 추가된 permission이 있다면 패키지 업로드 시 자동으로 추가 permission 작성 페이지로 넘어가진다. (개인정보 보호 – 권한 요청 이유)



2024년 4월 개발에서 scripting permission을 추가하여, 권한 추가에 대한 근거를 웹스토어에 제출했다.



제출하여 검토받기를 통해 크롬 웹스토어의 검토를 통해 업데이트를 진행하면 된다.

보통 검토에는 1~3일이 걸리고, 이는 크롬사의 상황에 따라 달라집니다.

정상적으로 배포가 된 상황이면 gitea의 master 브랜치에 push 하시면 됩니다.

# 5. 남은 문제점

※ 5장은 우려되는 문제점, 예상 해결법 등 추측성 정보가 존재한다. 보통 우선순위가 밀려 해결하지 못한 과제가 많고, 해당 과제들을 수정하던 도중 다른 중요한 문제가 발생하여 중단된 것이 3건이다. 이처럼 셀렉터는 긴박하게 해결해야하는 문제가 많이 발생하기 때문에, 셀렉터 구조에 대한 이해가 매우 중요하다.

## 5.1. 익스텐션 UIPath 충돌 문제

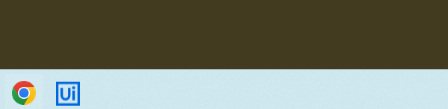
추가로 개발해할 사항 1건과 수정해야할 사항 1건이 존재한다.

현재 비봇 익스텐션은 UIPath 익스텐션과 충돌이 있어 사용하려면 UIPath 익스텐션을 비활성화하고 사용해야한다.

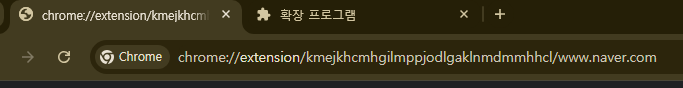
비봇 익스텐션은 브라우저가 켜진 후 메시지 전송 방식 (NativeMessaging) 으로 브라우저를 인식한다. 이 때 C# - js 간의 연결을 설정하고 하이라이트, 좌표 계산 등의 역할을 담당하는 것이 background.js 이다.

UIPath 익스텐션은 브라우저를 킬 때 백그라운드에 브라우저를 하나 더 연다.

(chrome://extension/(UIPath익스텐션 key)/(url) 형태)



*웹 브라우저를 열 시 익스텐션 상에서 여는 브라우저가 작업표시줄에 잠시 노출되는 현상*

**

*Open URL 액티비티 오류 현상 중 하나로 UIPath에서 연 웹으로 접속하는 상황*

*(kmej… 는 UIPath 익스텐션의 key 값이다.)*

비봇 익스텐션은 Open URL 이후 Get Element에서 위 UIPath가 연 브라우저를 최근 브라우저로 인식하고 오류가 발생한다.



현재 비봇은 크롬 메니페스트 버전3을 사용하고 있는데 정책상 다른 익스텐션의 접근을 차단한다.



*익스텐션 – manifest.json*

UIPath에서 여는 chrome://extension/~~ 으로 비봇 익스텐션에서 접근하려 하고 위와 같이 차단되기 때문에 background.js가 정지, 이후 정상적인 요소 인식 및 동작이 불가능했던 것이다.

|  |  |
| --- | --- |
| **기능** | **동작 여부** |
| 녹색 오버레이 | 가능 |
| **하이라이트** | **불가능** |
| **클릭** | **불가능** |

녹색 오버레이는 위와 같이 오류가 발생해도 동작했는데 이는 background.js를 통하지 않고 content.js 내에서 처리하기 때문이다.

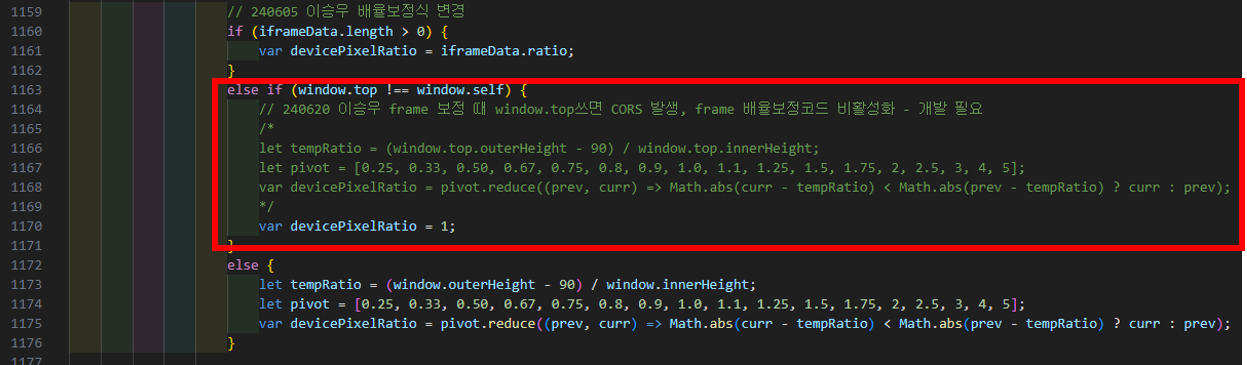
UIPath 익스텐션을 비활성화하니 정상적으로 background.js가 동작함을 확인하였다

본인이 해결 과정에 집중하고 있었으나, 강부장님의 우선순위 판단 및 셀렉터에 중대한 오류 상황이 생겨 개발이 중단되었다.

예상되는 해결방법으로는 마지막 url이 chrome extension이라면 이전의 url을 기반으로 작동하게 하면 될 것이다. 이를 위해 크롬 익스텐션의 “기록” 권한도 승인을 받은 상태이다. 크롬 익스텐션을 업데이트 할 경우 권한이 추가되면, 심사가 5~7일로 늘어나는 경우가 있으니 빠르게 신청해야 한다.

## 5.2. <frame> 보정

나라장터 사이트의 공고목록의 경우 현대 표준에서는 사용하지 않는 태그인 <frame>, <frameset>으로 화면을 출력한다.



*익스텐션 content.js – applyPhysicalCords*

배율 계산은 주석처리 후 100% 배율로 적용 중이다.

계산을 하지 않고 100%로 가정하고 있기 때문에 화면 배율에 따른 좌표 보정이 되지 않는다. 현재 윈도우 배율에서도 에러가 난 상황이라 윈도우 배율 수정을 더 급하게 수정했다. 위 문제의 해결 방법으로는 frame 구조에서 브라우저의 배율을 가져올 수 있는 방법을 찾아보면 되는데 uix, uiy의 좌표 계산을 통해 근접한 배율 값을 찾을 수 있지 않을까 예측해본다.

## 5.3. 하이라이트 실행 불가

특정 PC(속도가 느린(예측))에서 셀렉터를 사용한 이후 하이라이트가 Edge 브라우저에서 동작하지 않는 문제를 발견했다. 프로세스 점유를 못하거나 통신에 리소스가 필요한데, 적절하게 존재하지 않아서 발생하는 문제라고 생각한다. 예상되는 해결 방법으로는 beBOT의 최적화만이 가능할 것 같은데 난이도가 꽤나 높은 문제라고 생각한다.

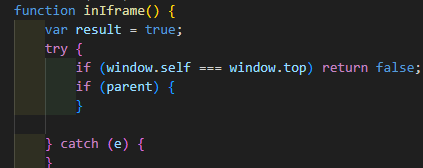
## 5.4. 구조 개선점

추가적인 코드 구조 개선점은 다음과 같다.

**- (js) 모든 구조가 하나의 함수안에 들어가 있어 모듈화가 되어 있지 않고 코드 파악에 어려움을 겪는다.**

**- (C#, js) 쓰레기코드 수정**

beBOT C# 코드와 js 코드 모두 사용하지 않거나 의도를 이해할 수 없는 구조의 코드가 많다.



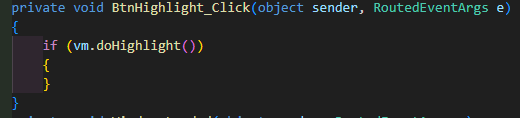
*익스텐션 – content.js – function inIframe() 12, 13 line*

parent임을 체크한 후 아무런 동작을 하지 않는 조건문 (조건도 window.self === window.top으로 체크하면 frame 도 해당 조건에 의해 최종적으로 false가 반환된다.)



*익스텐션 – content.js – function inIframe() 21 line*

true와 false가 같은지 비교하는 코드 (해당 구문내 내용을 확인하면 추측하기로는 개발 중 true == true, true == false로 분기를 바꿔가며 테스트하기 위함으로 보이나 계속해서 코드에 남아있다.)



*기존 beBOT.Interfaces\Selector\SelectorWindow.xaml.cs – BtnHighlight\_Click() 376 line*

함수호출을 if 문 내에서 하고 아무것도 안하는 코드 (해당 코드는 주석처리했다.)



*익스텐션 – content.js 내 stackoverflow 검색 결과*

인터넷 상에서 떠도는 오픈소스를 복사해 그대로 사용한 코드

이 경우 의미 없는 값을 반환하거나 동작하는 경우가 많다.

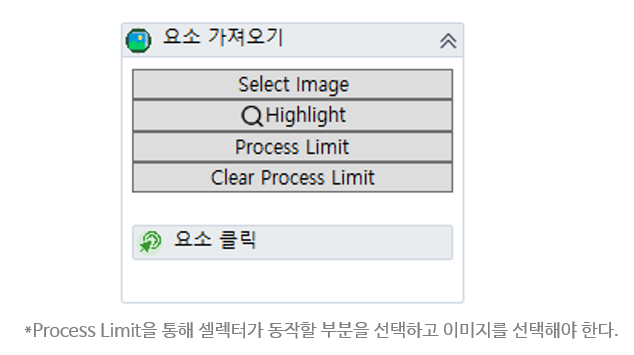
이 외 GPT를 통해 생성한 코드이나 충분한 검증을 하지 않고 단순 복사-붙여넣기로 넣은 코드 등 “쓰레기 코드” 가 전범위적으로 존재하기 때문에 대대적인 개선이 필요하다.

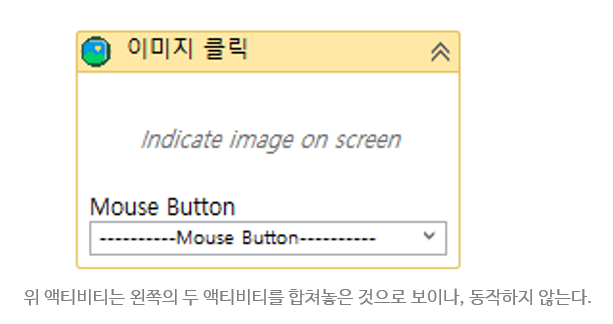
## 5.5. 무분별 로그

|  |  |
| --- | --- |
| **background.js** | **content.js** |
|  |  |
| 73개 | 201개 |

js 와 C# 모두 무분별한 로그가 많다. 변수에 대한 확인, 메소드 시작 확인 등 개발 및 실사용에 도움이 되지 않는 개인 확인용 로그가 많이 존재하기 때문에 판단 후 제거해야한다 판단한다.

## 5.6. 이미지 셀렉터

윈도우 셀렉터로 짚지 못하는 부분은 이미지 셀렉터로 수행하게 해야 하지만 특정 상황에서만 동작한다.



이미지 셀렉터와 윈도우 셀렉터를 언제 써야 하고, 어떤 역할을 하는지 beBOT을 코딩하는 실습생들이 정확하게 알아햐 함

NM 셀렉터 : 인터넷 요소를 가져온다.

윈도우 셀렉터 : 윈도우 운영체제의 요소를 가져온다.(폴더, 하단바등)

이미지 셀렉터 : 보안, 동작상 짚지 못하는 부분을 사진을 찍어 아예 가져오게 된다.

이미지 셀렉터가 작동하는 액티비티가 존재한다. 요소 가져오기 -> 클릭, 하지만 이미지 클릭 같은 액티비티는 작동하지 않는다.

판교에서 비봇화 도중 필요한 동작으로 우선 이미지 요소 가져오기 요소 클릭으로 동작을 구현할 수는 있지만, 오류는 수정이 필요한 부분이다.

1. https://github.com/open-rpa/openrpa [↑](#footnote-ref-1)
2. HTML 트리 구조 (각각의 노드가 “요소”이다.) [↑](#footnote-ref-2)
3. Select를 통해 받아온 요소의 정보 [↑](#footnote-ref-3)
4. 크롬 익스텐션의 설정파일로, 버전은 v3를 사용 중이다. (버전마다 양식이 다르다.) [↑](#footnote-ref-4)
5. js 의 mouseover 이벤트로 판단 [↑](#footnote-ref-5)
6. JHP\Extension 스토어 등록\Extension 업데이트 등록 방법.pptx [↑](#footnote-ref-6)