



오픈소스SW기여 중간보고서
프로젝트 명 : webConvertor

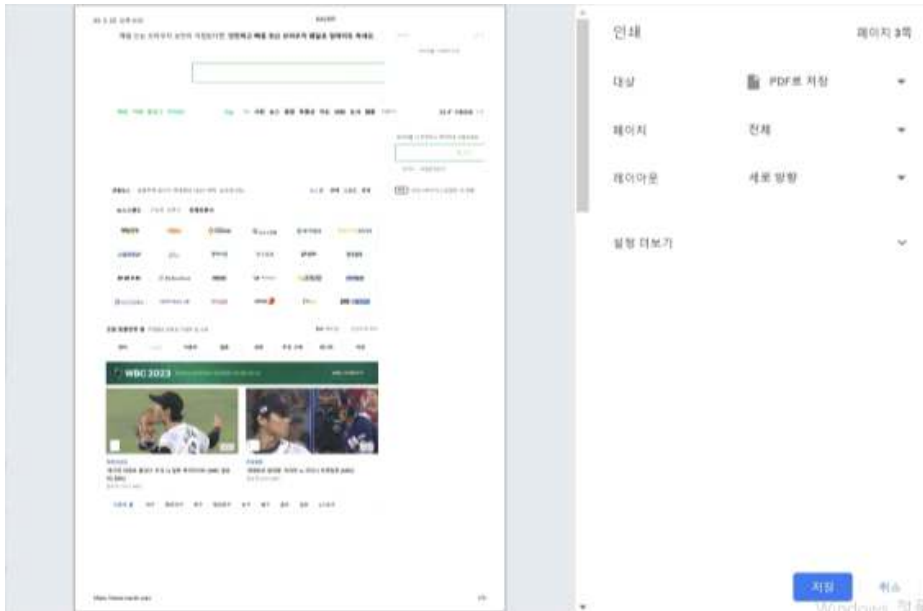
이름	고현우 / 소완열
학과	소프트웨어 공학과
학번	32170171 / 32172110
과목명	오픈소스SW기여
분반	1분반
교수님	송인식 교수님
제출일자	2023 / 05 / 09

목 차

1. 개발 목적
2. Use Case
3. 사용 기술
4. 아규먼트 정의 및 파싱
5. 상세 설계
6. 프로젝트 빌드
7. 테스트
8. 사용 예시
9. 진행 상황

1. 개발 목적

기존의 web page를 pdf로 변환하려면 아래와 같이 페이지를 뷰어로 열람 후 직접 프린트를 해야 합니다.



<그림 1. 기존의 웹페이지 -> PDF 변환 방법 >

이를 서버단에서 직접 진행하기 어려운 상황이며, url이나 파일을 pdf로 변환해주는 콘솔 어플리케이션을 제공하는 오픈소스 프로젝트 또한 확인할 수 없었습니다.

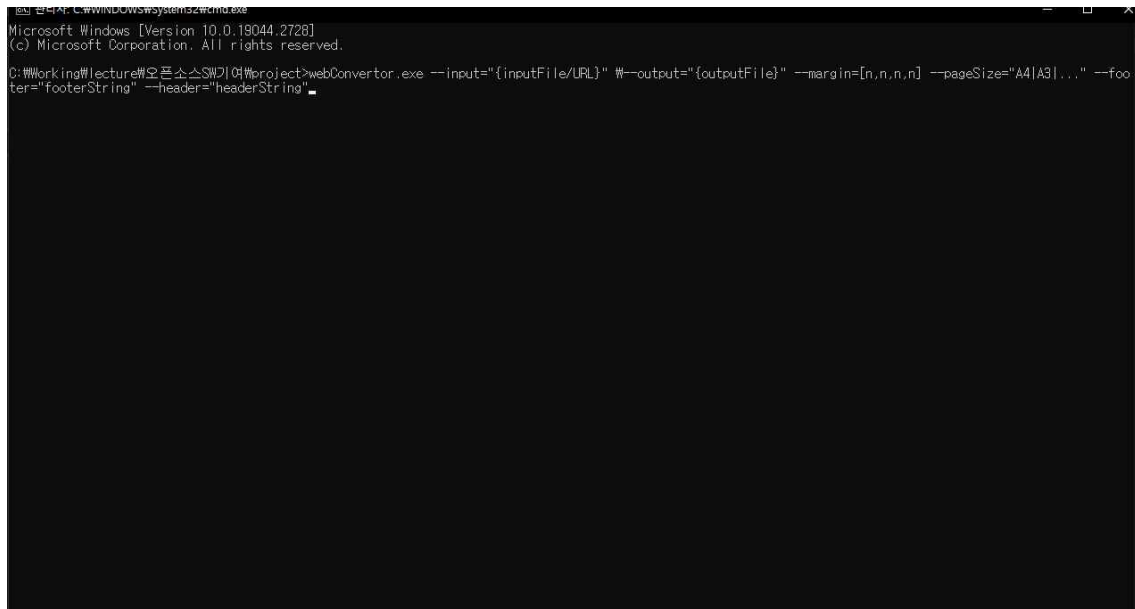


<그림 2. 국내 제공 중인 PDF 변환 솔루션>

다만, 비슷한 서비스를 제공하는 유/무료 서비스와 솔루션 제품은 국내에 몇몇 있는 것으로 확인했습니다.

위와 같이 사용자 의존적인 프로세스를 자동화할 수 있도록 지원하는 콘솔 프로그램을 개발하고자 합니다.

이러한 기능을 제공해준다면, 해당 모듈을 사용하여 다양한 오픈소스 프로젝트의 파급을 기 대해 볼 수 있을 것으로 보입니다.

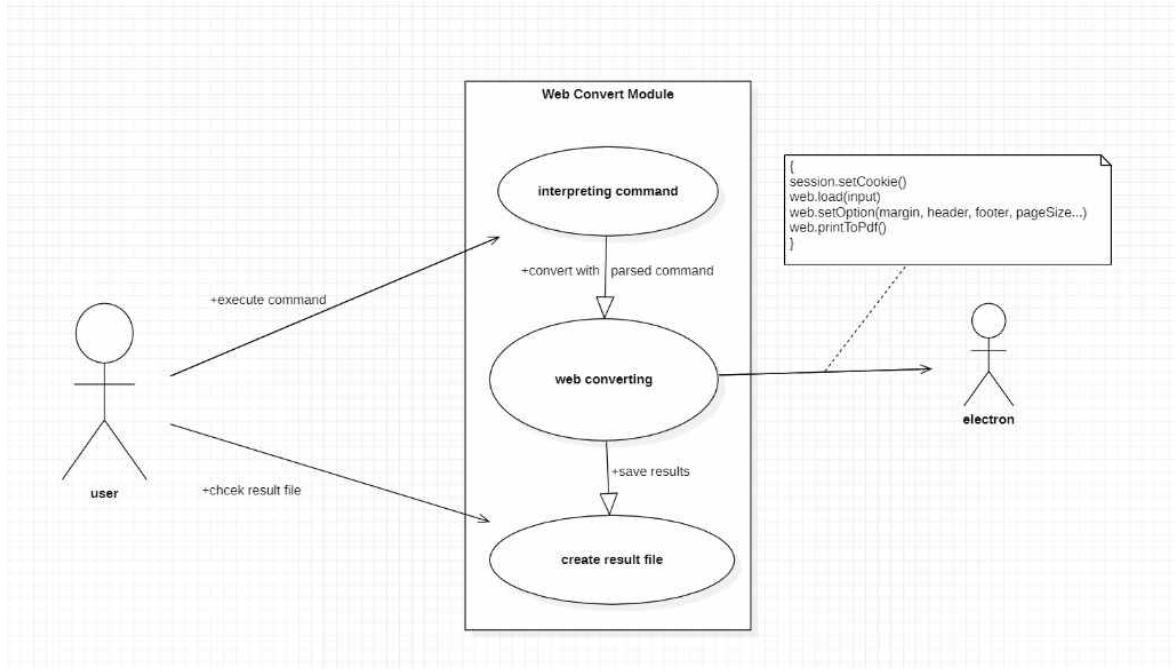


```
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Working\lecture\오픈소스\기여\project>webConverter.exe --input="{inputFile/URL}" --output="{outputFile}" --margin=[n,n,n,n] --pageSize="A4|A3|..." --footer="{footerString}" --header="{headerString}"
```

<그림 3. 프로그램 사용 예시>

2. Use Case



<그림 4. useCase>

어플리케이션 사용 시점의 use case 다이어그램입니다. 해당 그림에서 확인할 수 있듯이 사용자는 커맨드 실행 및 결과 파일의 생성 여부만을 확인하며, 이외의 다른 액션은 필요하지 않습니다.

또한 어플리케이션에서는 입력받은 커맨드 라인으로부터 필요한 설정 데이터를 파싱하여 데이터 세팅 후 PDF로 변환을 진행합니다.

3. 사용 기술

electron 모듈에서 제공하는 기술을 기반으로 웹 페이지를 로드하고, 사용자의 커스텀 세팅 후 PDF파일로 인쇄하는 기능 사용

핵심 기술 목록

1. load

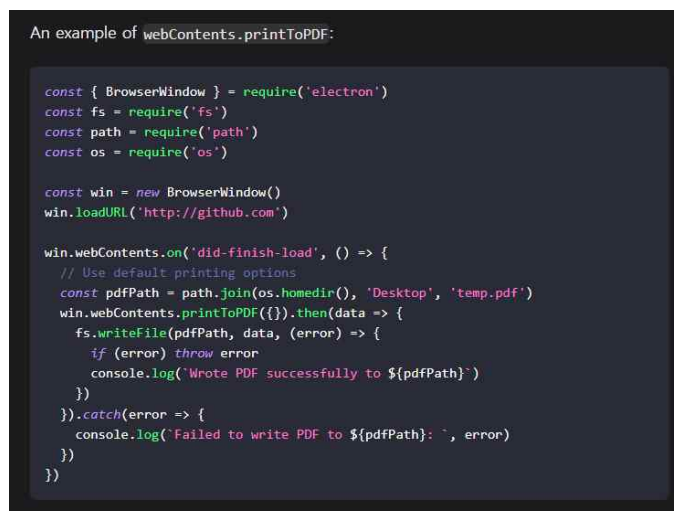
http주소나 html, mht, xml와 같이 로컬에 있는 파일을 electron상의 web contents로 로드하는 기능



<그림 5. load 함수>

2. printToPdf

webContents 상의 내용을 웹 브라우저 내의 pdf로 인쇄와 동일하게 pdf로 인쇄 진행하는 기능



<그림 6. printToPdf 함수>

3. show / openDevTools

디버깅용 툴로, webContents가 로드 된 후 스크립트로 조작할 때 이슈가 발생하거나, 의도와는 다른 결과물이 생성된 경우 이를 통하여 트래킹

Using the `ready-to-show` event

While loading the page, the `ready-to-show` event will be emitted when the renderer process has rendered the page for the first time if the window has not been shown yet. Showing the window after this event will have no visual flash:

```
const { BrowserWindow } = require('electron')
const win = new BrowserWindow({ show: false })
win.once('ready-to-show', () => {
  win.show()
})
```

<그림7. show/ opendevTools 함수>

4. executeJavaScript

web contents에서 파라미터로 전달한 js를 실행하는 함수로, 페이지의 margin이나, footer / header등의 문구를 추가하는 기능

`contents.executeJavaScript(code[, userGesture])`

- `code` string
- `userGesture` boolean (optional) - Default is `false`.

Returns `Promise<any>` - A promise that resolves with the result of the executed code or is rejected if the result of the code is a rejected promise.

Evaluates `code` in page.

In the browser window some HTML APIs like `requestFullscreen` can only be invoked by a gesture from the user. Setting `userGesture` to `true` will remove this limitation.

Code execution will be suspended until web page stop loading.

```
contents.executeJavaScript('fetch("https://jsonplaceholder.typicode.com/users/1").  
.then((result) => {  
  console.log(result) // Will be the JSON object from the fetch call  
})')
```

<그림8. executeJavaScript 함수>

4. 아규먼트 정의 및 파싱

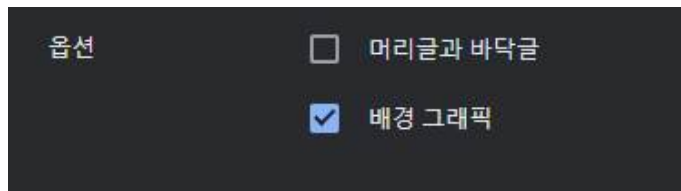
1. 아규먼트 정의

A. 필수 항목

- (1) --input="Input URL or local file path"
- (2) --output="Result PDF File path"

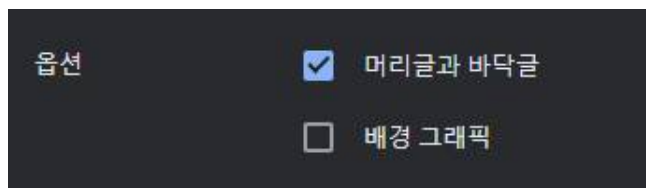
B. 선택 항목

- (1) --delay=[millisecond] : Wait this time after the page loads.
- (2) --printBackground
 - electron의 printBackground 옵션에 매핑
 - 브라우저의 "배경 그래픽" 옵션과 동일
 - switch용 옵션으로 없으면 default 값 true



<그림 9. 배경그래픽 옵션>

- (3) --footer="some text" / --header="some text"
 - 브라우저의 "머리글과 바닥글" 옵션과 동일
 - 머리글과 바닥글에 들어갈 텍스트 각각 설정 가능



<그림 10. 머리글과 바닥글 옵션>

- (4) --landscape
 - 인쇄 용지의 방향 설정. default값은 false(세로)
- (5) --margin=[no-margin|minimum|n,n,n,n]
 - 옵션이 없는 경우 default는 no-margin
 - 브라우저의 최소마진 혹은 Top,Left,Bottom,Right 순으로 입력 가능

(6) --timeout=[millisecond]

- 변환 작업이 특정 시간 이상으로 소요될 경우 강제로 종료

(7) --pageSize=[A4|A3 ...]

- 인쇄 페이지 크기 설정 default는 A4

(8) --cookies="json file path"

- url 접속 시 필요한 쿠키의 데이터 설정

```
{
  "cookies" :
  [
    {
      "url"      : "https://google.com/",
      "name"     : "hyunwoo",
      "value"    : "1234",
      "domain"   : "localhost",
      "path"     : "/",
      "expirationDate" : 1814157005,
      "secure"   : false,
      "httpOnly" : false
    }
  ]
}
```

<그림 11. cookie json 예시>

(9) --requestHeader="text file path"

- url 접속 시 필요한 request Header의 데이터 설정
- text파일 예시
key:value
key:value
...

(10) --logDir="log directory"

- webConverter 사용시에 생성될 로그의 저장 경로

2. 아규먼트 파싱

1. Electron의 BrowserWindow 객체와 이의 callback 함수를 활용.
2. Browser의 .on('ready') 시점에 'electron-args' 모듈을 사용하여 사전에 정의한 아규먼트를 파싱 및 저장.
3. 몇몇 아규먼트에 대해 alias를 정의
 - 3 - 1. input : i
 - 3 - 2. output : o
 - 3 - 3. help : h
 - 3 - 4 delay : d
 - 3 - 5. timeout : t
 - 3 - 6. margin : m
4. 필수 입력 항목인 input과 output이 String Type인지(None 타입이 아닌지) 확인인
5. 아규먼트를 잘못 입력했거나, help를 받은 경우에 가이드 출력
가이드 =>

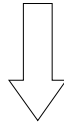
```
require:
  --input=input URL or local file path
  --output=Result PDF File path
optional:
  --delay=[millisecond]
      Wait this time after the page loads.
  --printBackground
  --footer=[some text]
  --header=[some text]
  --landscape
  --margin=[no-margin[minimum[n,n,n,n]
  --timeout=[millisecond]
  --pageSize=[A4|A3 ...]
  --cookies="json file path"
  --requestHeader="text file path"
Text Format:key:value
      key:value
  ...
  --logDir="log directory"
```

5. 상세 설계

1. 전반적인 흐름

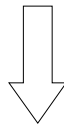
(1) 입력받은 옵션 파싱 및 데이터 저장

- electron-args 모듈을 사용하여 아규먼트 파싱



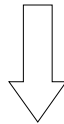
(2) checkArgsAvailable(..)을 통한 아규먼트 검증

- cookie, header 파일과 디렉토리가 있는지 확인 등의 검증을 진행



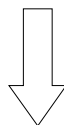
(3) print(...) 함수를 통해 electron Browser init 및 로드 옵션 설정 후 페이지 로드

- debugMode 옵션 확인 후 설정
- timeout 옵션 확인 후 timeout 설정
- webBrowser의 'did-finish-load' callback의 정의 및 이를 통한 printPage(...) 함수 호출
- input의 종류 및 기타 설정(cookie, header) 이후 loadUrl() / loadFile()을 통한 페이지 로드



(4) printPage(...) 함수를 통한 기타 옵션 설정 및 실제 변환 함수 호출 대기

- applyAttribute(..) 함수를 통해 margin, header, footer, pageSize등을 설정
- header, footer 의 유무를 파악하여 displayHeaderFooter 옵션 설정
- 옵션의 delay 값만큼 timeout 설정
- timeout 이후 printToPdf(..) 함수 호출



(4) printPage(...) 함수를 통한 기타 옵션 설정 및 실제 변환 함수 호출 대기

- electron의 window.webContents.printToPdf(..) 함수를 통해 실제 변환 수행
- 위 함수의 callback data를 'fs' 모듈을 통해 실제 파일로 변환

2. 참조 모듈 및 전역 변수

(1) 참조 모듈

- app : BrowserWindow = require('electron')
- session = require('electron')
- parseArgs = require('electron')
- fs = require('fs')

(2) 전역 변수

- logFile : 로그파일의 경로
logging(type, context) 함수를 호출할 때 마다 변수로 넘기기 보단 전역 변수로 로그파일의 경로를 핸들링하도록 함
- window : BrowserWindow 객체를 핸들링 할 변수

3. 기능별 함수 설계 (PesudoCode)

(1) const checkArgsAvailable = (input ,cookies,logDir)

```
# inputFile이 로컬의 파일인 경우, 존재 여부를 확인한 후 없다면
  INPUT_FILE_NOT_FOUND 오류 코드로 프로세스 종료
# cookie 파일을 받았는데, 해당 경로에서 쿠키파일을 찾을 수 없다면
  COOKIES_FILE_NOT_FOUND 오류 코드로 프로세스 종료
# logDir을 받았는데, 해당 디렉토리가 존재하지 않다면
  LOG_DIRECTORY_NOT_FOUND 오류 코드로 프로세스 종료
# 해당 디렉토리가 존재하는 경우, makeLogFile(logDir) 함수로 로그 파일 생성
# return void
```

(2) const checkFileFormat = (input)

```
# inputFile의 확장자가 web으로 load 가능한 파일인지 확인
# 허용 가능한 확장자 : [mhtml, mht, html, htm, xml]
# 허용가능한 경우 return true else return false
```

(3) const makeLogFile = (logDir)

```
# logDir의 위치에 YYYYMMDDHHMMSS.txt 형식으로 전역 변수인
  logFile 변수를 설정
# return void
```

(4) const logging = (type,context)

```
# 전역변수인 logFile의 경로에 'fs' 모듈을 통해 파일 작성
# 'fs' 모듈에서 sync 오류 발생 시
  WRITE_LOG_SYNC_ERROR 오류 코드로 프로세스 종료
```

(5) const insertCookies = (url, cookies)

```

# 입력받은 cookie 파일을 읽어 JSON 모듈로 파싱
# 파싱된 데이터의 key를 순회하며 cookie 객체 생성
  cookie 객체 : (url, name, value, expirationDate, path)
# 각 json key 별로 생성한 cookie 객체를 electron 모듈의
  defaultSession.cookies.set(cookie)와 같이 세팅
# cookie 설정 중 오류가 발생한 경우, 별도의 종료 프로세스 없이 계속 진행
# return void

```

(6) const makeURLOption = (reqHeader)

```

# reqHeader 경로의 header파일 (txt)를 읽어 이를 string화
# return headerString

```

(7) const applyAttribute = (margin, header, footer, pageSize)

```

# if margin !== undefined 인 경우 margin 세팅 진행
  입력받은 margin(string)을 split(',') 하여 top, bottom, right, left 순서 별로
  저장 및 각각의 값을 margin/25.4 연산을 통해 mm에서 inch로 변환
# if header !== undefined 인 경우 header 세팅을 위한 템플릿 생성
  '<span style="font-size:10px; margin-left:25px;">' + header + '</span>'
# if footer !== undefined 인 경우 footer 세팅을 위한 템플릿 생성
  '<span style="font-size:10px; margin-left:25px;">' + footer + '</span>'
# 설정한 pageSize에 margin을 문제 없이 설정할 수 있는지 검증
# 위에서 생성한 객체들을 list에 담아서 return
  return [margins, headerTemplate, footerTemplate, pageSize]

```

(8) const print = async (input, output, cookies, requestHeader, delay, timeout, margin, printBackground, landscape, header, footer, pageSize, debugMode)

```

# 페이지 로드용 사용하기 위해 전역 변수로 선언된 window init
# debugMode가 켜진 경우 devtool을 킴
# 아규먼트로 timeout을 받은 경우 프로세스의 timeout 설정
  timeout시 APP_TIMEOUT 오류 코드로 프로세스 종료
# 'did-finish-load' 콜백을 받으면 printPage(...) 함수를 호출하도록 정의
# input 파싱 및 로드 진행
  html 등의 로컬 파일을 로드 할 경우 localFilePath 변수를 사용
  http 경로를 통해 로드하는 경우 urlPath 변수를 사용
  if input이 'http://', 'https'로 시작될 경우 이를 urlPath에 삽입
else if inputFile이 .url 인 경우, 파일을 열어서 앞의 url= 부분을 제외한
  subString을 파싱
  if url= 이후에 http:// 또는 https:// 인 경우 urlPath에 삽입
  else if url= 이후에 file:///인 경우 localFilePath에 삽입
  else 디폴트로 localFilePath로 삽입

```

```

else if checkFileFormat(input)을 통해 지원 가능한 확장자 인지 확인
  지원 가능한 포맷이면 localFilePath에 삽입
# 파일 로드 진행
  if localFilePath and 'fs' 모듈을 통해 파일이 있는지 확인
    window.loadFile(localFilePath)로 로컬 파일 로드
  else if urlPath
    insertCookies(urlPath, cookies)로 session에 쿠키 세팅
    header = makeURLOption(requestHeader)를 통해 header 파싱
    window.loadUrl(urlPath, header)로 url 로드
  else
    UNSUPPORT_INPUT_FILE_TYPE 오류 코드로 프로세스 종료
# return void

```

```

(9) const printPage = (output, delay, margin, printBackground, landscape, header,
  footer, pageSize)
# print(...)에서 'did-finish-load' 콜백 시 호출되는 함수
# attr_data = applyAttribute(margin, header, footer, pageSize)
# header나 footer가 정의된 경우 displayHeaderFooter 플래그 on
# delay가 있거나, footer, header 및 margin 중 하나라도 설정된 경우
  delayTime을 설정 (기본값 100ms -> footer 등의 attr 설정 시간)
# setTimeout(printToPdf(..), delayTime) 으로 delay를 준 후 printToPdf(...)
  함수에서 실제 변환 진행
# return void

```

```

(10) const printToPdf = (filePath, margins, printBackground, landscape, pageSize,
  headerTemplate, footerTemplate, displayHeaderFooter, postCallback)
# 전달받은 파라미터를 사용하여 변환 진행
# postCallback() 함수는 printPage에서 () => app.exit(0)를 전달
# window.webContents.printToPDF({
  margins : margins,
  printBackground : printBackground,
  landscape : landscape,
  pageSize : pageSize,
  displayHeaderFooter : displayHeaderFooter,
  headerTemplate : headerTemplate,
  footerTemplate : footerTemplate}).then(data=> {
# callback 부분에서 'fs'모듈을 통해 로컬에 파일로 저장 및 넘겨받은 callback
  함수 호출
  fs.writeFile(filePath, data, () => {postCallback();})
# return void

```

4. 오류 코드 정의

(1) INPUT_FILE_NOT_FOUND

- input을 로컬에 있는 파일로 설정했을 때 파일을 찾지 못한 경우의 오류 코드
- code = 100

(2) UNSUPPORT_INPUT_FILE_TYPE

- input을 지원할 수 없는 경우에 발생하는 오류
- localFile일 수도, url 파일일 수도 있음.
- code = 101

(3) COOKIES_FILE_NOT_FOUND

- cookie 옵션으로 받은 경로에서 cookie 파일을 찾지 못한 경우의 오류 코드
- code = 102

(4) LOG_DIRECTORY_NOT_FOUND

- 로그 파일 작성을 위한 logDir 옵션으로 받은 경로를 찾지 못한 경우
- code = 103

(5) WRITE_LOG_SYNC_ERROR

- 로그 파일 작성 중 오류가 발생한 경우의 오류 코드
- code = 104

(6) APP_TIMEOUT

- 정의된 timeout 시간보다 지체되어 프로세스가 종료된 경우의 오류 코드
- code = 105

6. 프로젝트 빌드

- npm과 nvm 기반으로 빌드를 진행하며, 이를 별도의 세팅 없이 자동으로 window와 linux 용 모듈 빌드를 수행해주는 python 스크립트를 함께 제공

제공한 build.py 스크립트를 실행하면 아래와 같이 linux 및 windows용 모듈을 각각 빌드

```
C:\Git\WebConvector>build.py
'npm'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
8.11.0
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
up to date, audited 234 packages in 1s
57 packages are looking for funding
  run 'npm fund' for details
1 high severity vulnerability
To address all issues, run:
  npm audit fix
Run 'npm audit' for details.
npm notice
npm notice New major version of npm available! 8.11.0 -> 9.6.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.5
npm notice Run 'npm install -g npm@9.6.5' to update!
npm notice
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
> webconvector@1.0.0 build:windows
> run-s updateVersion pack:windows postwinbuild

> webconvector@1.0.0 updateVersion
> node updateVersion

> webconvector@1.0.0 pack:windows
> electron-packager . --out=dist --asar --overwrite --platform=win32 --arch=ia32 --prune "--ignore=(cache|dist|out|res|setup|tool)"
Packaging app for platform win32 ia32 using electron v22.0.2

> webconvector@1.0.0 postwinbuild
> node postwinbuild

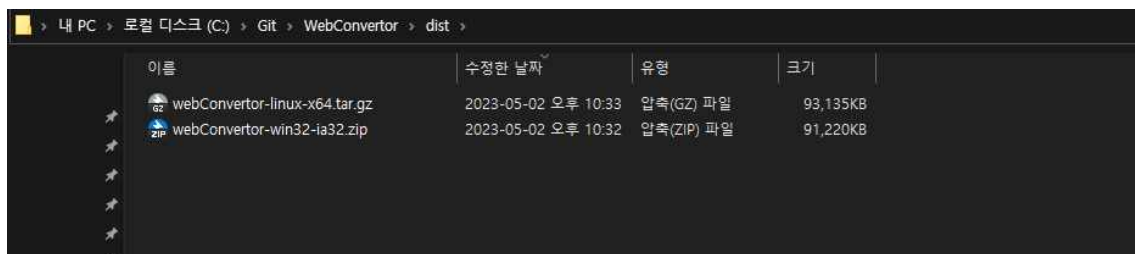
version: 1.0.0.7
success
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
> webconvector@1.0.0 build:linux
> run-s updateVersion pack:linux

> webconvector@1.0.0 updateVersion
> node updateVersion

> webconvector@1.0.0 pack:linux
> electron-packager . --out=dist --asar --overwrite --platform=linux --arch=x64 --prune "--ignore=(cache|dist|out|res|setup|tool)"
Packaging app for platform linux x64 using electron v22.0.2
```

<그림 12. build.py 실행 예시>

빌드 결과로는 각 OS별 모듈 및 패키지가 압축되어 있는 두 개의 압축파일 생성



<그림 13. dist하위에 결과 파일 예시>

이름	압축 크기	원본 크기	파일 종류	수정된 날짜
locales				
resources				
webconvector.exe	64,558,189	136,277,504	응용 프로그램	2023-05-02 오후 10:3...
vulkan-1.dll	319,579	793,088	응용 프로그램 확장	2023-05-02 오후 10:3...
vk_swiftshader_icd.json	89	106	JSON 원본 파일	2023-05-02 오후 10:3...
vk_swiftshader.dll	1,815,569	4,486,656	응용 프로그램 확장	2023-05-02 오후 10:3...
version	8	6		2023-05-02 오후 10:3...
v8_context_snapshot.bin	164,123	599,880	BIN 파일	2023-05-02 오후 10:3...
snapshot_blob.bin	56,053	300,024	BIN 파일	2023-05-02 오후 10:3...
resources.pak	5,273,420	5,374,671	PAK 파일	2023-05-02 오후 10:3...
LICENSES.chromium.html	857,395	6,810,078	Chrome HTML Docum...	2023-05-02 오후 10:3...
LICENSE	640	1,096		2023-05-02 오후 10:3...
libGLESv2.dll	2,710,358	6,727,168	응용 프로그램 확장	2023-05-02 오후 10:3...
libEGL.dll	181,250	384,000	응용 프로그램 확장	2023-05-02 오후 10:3...
icudtl.dat	4,539,057	10,462,432	DAT 파일	2023-05-02 오후 10:3...
ffmpeg.dll	1,168,861	2,525,184	응용 프로그램 확장	2023-05-02 오후 10:3...
d3dcompiler_47.dll	1,821,809	4,108,752	응용 프로그램 확장	2023-05-02 오후 10:3...
chrome_200_percent.pak	168,545	179,934	PAK 파일	2023-05-02 오후 10:3...
chrome_100_percent.pak	121,245	129,653	PAK 파일	2023-05-02 오후 10:3...

<그림 14. 압축파일 내부 파일 목록 예시 - window>

이름	압축 크기	원본 크기
locales		
resources		
webconvector		163,368,888
vk_swiftshader_icd.json		107
version		6
v8_context_snapshot.bin		483,224
snapshot_blob.bin		172,048
resources.pak		5,403,623
LICENSES.chromium.html		6,809,232
LICENSE		1,096
libvulkan.so.1		6,485,960
libvk_swiftshader.so		4,443,656
libGLESv2.so		6,552,360
libffmpeg.so		2,968,264
libEGL.so		251,936
icudtl.dat		10,462,432
chrome-sandbox		52,808
chrome_crashpad_handler		1,244,048
chrome_200_percent.pak		181,772
chrome_100_percent.pak		130,358

<그림 15. 압축파일 내부 파일 목록 예시 - linux>

해당 압축 파일 내에는 위와 같이 webConvector 모듈과 리소스 파일 및 dll 라이브러리 등이 포함됨

7. 테스트

1. 테스트 정의

(1) 단위 테스트

--debugMode 옵션을 통해 로드된 웹 페이지와 크롬의 개발자 tool을 사용.

또한, --delay 옵션을 통해 설정한 옵션(margin, header, footer 등)이 정상적으로 적용되었는지 시간을 두고 확인 가능

단위 테스트 목록

- ☐ delay 기능 : delay 시간만큼 지연 후 작업이 진행되는지 검증
- ☐ printBackground 기능 : 크롬에서 해당 옵션을 사용하여 인쇄한 결과와 같은 결과를 출력하는지 검증
- ☐ footer / header 기능 : footer 및 header가 정상적으로 삽입 되는지 검증
- ☐ landscape 기능 : 인쇄 용지 방향 설정이 정상적으로 수행되는지 검증
- ☐ margin 기능 : 여백이 정상적으로 설정되는지 검증
- ☐ timeout 기능 : timeout 시간이 경과 되었을 때 정상적으로 중지되는지 검증
- ☐ pageSize 기능 : 인쇄 페이지 크기가 정상적으로 설정되는지 검증
- ☐ cookie 기능 : 쿠키 값이 해당 세션에 정상적으로 삽입되어 있는지 검증
- ☐ header 기능 : 헤더 값을 포함하는 URL이 정상적으로 생성되는지 검증
- ☐ 로깅 기능 : 로그가 정상적으로 생성되는지 검증

(2) 통합 테스트

단위 테스트에서 검증한 각각의 기능을 통합하여 변환 작업이 정상적으로 수행되는지 테스트

통합 테스트 예시

- margin 및 footer / header를 적용하는 옵션으로 웹 페이지를 PDF로 변환
- cookie 및 header를 적용하여 로드한 페이지를 PDF로 변환
- 강제로 delay를 준 후 timeout을 delay보다 적게 주어 실패하는 변환 작업

2. 테스트 시나리오 및 절차

(1) 테스트 시나리오

해당 어플리케이션을 사용하는 시나리오는 다음과 같은 상황이 있을 수 있다.

Case A : 회사의 요청 등으로 인하여 웹에서 특정 물품의 가격 및 정보를 갖는 페이지를 취합하여 이를 일괄적으로 문서화 하는 작업

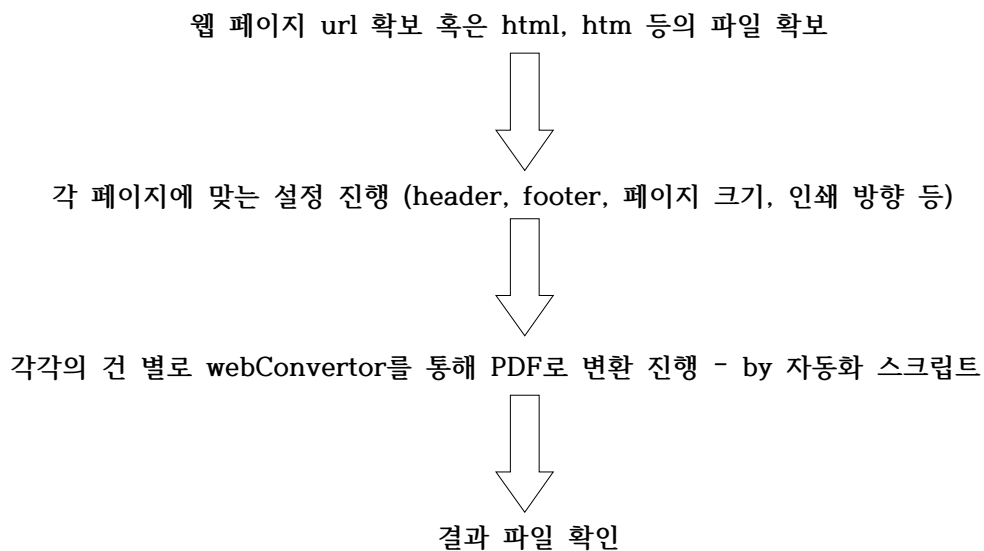
-> 해당 테스트 시나리오에서는 자동화 스크립트 혹은 UI 프로그램을 통해 input 파일들이 위치한 디렉토리를 받아 해당 파일들을 일괄적으로 변환.

Case B : 특정 파일을 백업 및 열람하는 기능을 제공하는 솔루션에서 html 및 web URL등을 PDF로 변환하여 저장하도록 하는 경우

-> 해당 테스트 시나리오에서는 python을 통한 간이 서버를 구축하여 API 호출시 파일을 multi-part 데이터로 송신 받아 이를 로컬에 저장 및 모듈을 통해 변환하여 저장

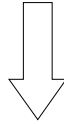
(2) 테스트 절차

Case A

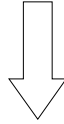


Case B

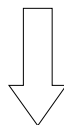
웹 페이지 url 확보 혹은 html, htm 등의 파일 확보



각 페이지에 맞는 설정 진행 (header, footer, 페이지 크기, 인쇄 방향 등)



각각의 건 별로 서버의 REST-API 호출



결과 파일 확인

3. 테스트 데이터

테스트 데이터는 별도의 제약 조건이 없습니다.

따라서 다양한 확장자의 문서와 cookie 및 header를 사용하는 테스트 케이스 등을 사용

테스트 샘플 예시

이름	수정된 날짜	유형	크기
cookieTest.html	2023-04-11 오후 9:05	Chrome HTML D...	1KB
html.html	2023-04-11 오후 9:05	Chrome HTML D...	2KB
redirect.html	2023-04-11 오후 9:07	Chrome HTML D...	1KB
redirectTest.html	2023-04-11 오후 9:05	Chrome HTML D...	1KB
res.json	2023-04-11 오후 9:05	JSON 원본 파일	1KB

- ☐ html.html : 일반적인 html 페이지 테스트 데이터
- ☐ cookieTest.html : 페이지에서 cookie 값을 가져다가 띄워 cookie 설정을 검증 데이터
- ☐ redirect.html, redirectTest.html : redirect 시점의 검증 데이터
- ☐ res.json : 쿠키 데이터

추후 htm, xml 등 다양한 포맷의 테스트 데이터 추가 예정입니다.

8. 사용 예시

```
관리자: C:\WINDOWS\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Working\lecture\오픈소스SW기여\module>webconverter.exe

C:\Working\lecture\오픈소스SW기여\module>
WebConverter : v1.0.0
require:
  --input=Input URL or local file path
  --output=Result PDF File path
optional:
  --delay=[millisecond]
    Wait this time after the page loads.
  --printBackground
  --footer=[some text]
  --header=[some text]
  --landscape
  --margin=[no-margin|minimum|n,n,n]
  --timeout=[millisecond]
  --pageSize=[A4|A3 ...]
  --cookies=[json file path]
  --requestHeader=[Text file path]
    Text Format:key:value
    key:value
  --logDir=[log directory]
  --printByScript

C:\Working\lecture\오픈소스SW기여\module>
```

<그림 16. 모듈 실행 예시>

9. 진행 상황

업무내용	담당자	3				4					5				6				
		1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	5
자료조사 Requirement Analysis	공통																		
System Design (Block diagram, System specification)	고현우																		
기능별 Coding	공통																		
Testing 및 보완	소완열																		
최종제작	공통																		
최종보고서	공통																		