



오픈소스 SW활용 세부 설계
프로젝트 명 : webConvertor

이름	고현우 / 소완열
학과	소프트웨어 공학과
학번	32170171 / 32172110
과목명	오픈소스SW기여
분반	1분반
교수님	송인식 교수님
제출일자	2023 / 04 / 18

1. 아규먼트 정의 및 파싱

● 1 -1 아규먼트 정의

A. 필수 항목

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

B. 선택 항목 (옵션)

- (1) --delay=[milesecond] : Wait this time after the page loads
- (2) --printBackground
 - electron의 printBackground 옵션에 매핑
 - 브라우저의 "배경 그래픽" 옵션과 동일
 - switch용 옵션으로 없으면 default 값 true
- (3) --footer = "some text" / --header = "some text"
 - 브라우저의 "머리글과 바닥글" 옵션과 동일
 - 머리글과 바닥글에 들어갈 텍스트 각각 설정 가능
- (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 = [milisecond]
 - 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
- cookie json 예시

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

(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

● 1 -2 아규먼트 파싱

1. Electron의 `BrowserWindow` 객체와 `이`의 `callback` 함수를 활용.
2. `BrowserWindow`의 `.on('ready')` 시점에 `'electron-args'` 모듈을 사용하여 사전에 정의한 아규먼트를 파싱 및 저장.
3. 몇몇 아규먼트에 대해 `alias`를 정의
 - 3 - 1. `input` : `l`
 - 3 - 2. `output` : `o`
 - 3 - 3. `help` : `h`
 - 3 - 4. `delay` : `d`
 - 3 - 5. `timeout` : `t`
 - 3 - 6. `margin` : `m`
4. 필수 입력 항목인 `input`과 `output`이 `String Type`인지 (`None` 타입이 아닌지) 확인
5. 아규먼트를 잘못 입력했거나, `help`를 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]
```

2. 상세 설계

2 - 1. 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

2 - 2. 기능별 함수 설계 (sudo Code)

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

```
# inputFile이 로컬의 파일인 경우, 존재 유무를 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(cookei)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
# retrun 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 로드에 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
```


3. 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

(2) UNSUPPORT_INPUT_FILE_TYPE

- input의 file type이 지원 가능한 파일이 아닌 경우의 오류 코드
- code = 102

(3) COOKIES_FILE_NOT_FOUND

- cookie 옵션으로 받은 경로에서 cookie 파일을 찾지 못한 경우의 오류 코드
- code = 103

(4) LOG_DIRECTORY_NOT_FOUND

- 로그 파일 작성을 위한 logDir 옵션으로 받은 경로를 찾지 못한 경우
- code = 104

(5) WRITE_LOG_SYNC_ERROR

- 로그 파일 작성 중 오류가 발생한 경우의 오류 코드
- code = 105

(6) APP_TIMEOUT

- 정의된 timeout 시간 보다 지체되어 프로세스가 종료된 경우의 오류 코드
- code = 106