

# 비대면 환경에서의 효과적인 교육 및 비즈니스를 위한 개인 맞춤형 융합 콘텐츠 생성 기반 기술

---

14주차 발표

김이홍조

2016112166 김운호

2017112085 이유경

2017112099 조민지

2017111718 홍은주

## 목차

---

1) 사용 API 개요  
- 음성 인식 테스트

2) 자막 파일 생성 및 화자 인식

3) 로컬서버 구축

The logo for Clova, featuring the word "Clova" in white lowercase letters on a green rectangular background, followed by a small white clover icon.The logo for CLOVA Speech, featuring the words "CLOVA Speech" in black uppercase and title case letters on a light blue rectangular background.

NEST (Neural End-to-end Speech Transcriber) 음성 인식 기술을 통해 빠르고 쉽게 미디어의 음성인식을 제공

길이가 긴 오디오 또는 비디오 파일에 대해 음성 인식 결과를 확인 가능

문장 자동 분리 및 타임 스탬프 지원

화자 인식 지원



# Naver CLOVA Speech

## API 호출 방식으로 진행 – 도메인 생성

Classic / CLOVA Speech / Domain

0

### Domain

도메인 기본 정보와 목록을 관리합니다

도메인 생성

상품 더 알아보기

새로 고침

수정

삭제

도메인 이름


검색어를 입력해 주세요



초기화

<input checked="" type="checkbox"/>	도메인 이름	도메인 코드	유형	생성일 (UTC+09:00)	동작
<input checked="" type="checkbox"/>	klhj-dom	dgu-klhj	Basic	2021-05-22 22:27:13	<div>빌더 실행</div>

# Naver CLOVA Speech

API 호출 방식으로 진행 – key와 url 확인

 **설정**

  주식회사 유니크유엑스 ▾

**연동 정보**    도메인 정보

Secret Key

+ 생성

복사

CLOVA Speech Invoke URL

복사

# Naver CLOVA Speech

API 호출 방식으로 진행 – 네이버 클라우드 플랫폼에서 제공하는 API 사용

```
class ClovaSpeechClient:  
    # Clova Speech invoke URL  
    invoke_url = ''  
    # Clova Speech secret key  
    secret = ''
```

앞서 확인한 url과 secret key를 해당 소스코드 내에 삽입



# Naver CLOVA Speech

## 음성 인식 테스트

Python 3.9.5

File Edit Shell Debug Options Window Help

```
text": "핸드폰 아직 안 뺐겼죠. 신고할 게 있다고 제가 손해에서 주운 건데요. 계장님이 꼭 보셔야  
할 것 같아서요. 그 안에 보시면 재미있는 사진들이 좀 있는 것 같아서요. 교도소 내 담배 반인민  
유통권으로 제 옆에 있는 아저씨를 고발하려고 합니다. 영희야 너 지금 나 협박하는 거냐 이 같은  
새끼. 그게. 보안 계장님을 이번에 이렇게 증거 사진도 있으니까 절차 밟기 쉽죠. 이 말로 십이만  
개의 새끼 진짜 씨야. 이 쓰레기 같은 새끼야. 저희가 휴대전화 반입도 신고하려고요. 그 사진들  
밖에 있는 이 아저씨의 동생들. 친구들한테 다 보내놨어요. 일주일 동안 연락이 없거나 우리가 연  
애가 안 된다 그러면 그 자식들 바로 경찰로 넘어갈 겁니다. 나야 뭐 1년 정도 추가 쓰면 그만이지
```

# 자막 파일 생성 및 화자 인식

자막 생성을 위한 Clova Speech Response API body  
-segment키에 자막 생성을 위한 정보가 있다는 것을 알 수 있다.

field	desc	type
result	결과 코드	string
message	결과 메시지	string
token	결과 토큰	string
version	엔진 버전	string
params	파라미터	object
params: service	서비스코드	string
params: domain	도메인	string
params: segment	세그먼트	string
params: morpheme	형태소	string
params: script	스크립트	string
params: completion	동기 비동기	string
params: userdata	유저데이터	object
segments	세그먼트 정보	array
segments: start	세그먼트 시작 시각 (ms)	number
segments: end	세그먼트 종료 시각(ms)	number
segments: text	세그먼트 텍스트	string
segments: textEdited	수정 내용	string
segments: diarization	인식된 화자	object
segments: diarization.label	인식화자 Number	string
segments: speaker	변경된 화자	object
segments: speaker.label	변경화자 Number	string
segments: speaker.name	변경화자명	string
segments: confidence	세그먼트 컨피던스 (0.0 ~ 1.0)	number
segments: words	세그먼트 어절	array
segments: words: [0]	세그먼트 어절 시간 시간 (ms)	number
segments: words: [1]	세그먼트 어절 종료 시간 (ms)	number
segments: words: [2]	세그먼트 어절 텍스트	string
text	전체 텍스트	string
confidence	전체 컨피던스	number



segments	세그먼트 정보	array
segments: start	세그먼트 시작 시각 (ms)	number
segments: end	세그먼트 종료 시각(ms)	number
segments: text	세그먼트 텍스트	string
segments: textEdited	수정 내용	string
segments: diarization	인식된 화자	object
segments: diarization.label	인식화자 Number	string
segments: speaker	변경된 화자	object
segments: speaker.label	변경화자 Number	string
segments: speaker.name	변경화자명	string
segments: confidence	세그먼트 컨피던스 (0.0 ~ 1.0)	number
segments: words	세그먼트 어절	array
segments: words: [0]	세그먼트 어절 시간 시간 (ms)	number
segments: words: [1]	세그먼트 어절 종료 시간 (ms)	number
segments: words: [2]	세그먼트 어절 텍스트	string



# 자막 파일 생성 및 화자 인식

## 자막 파일 생성의 과제

- Clova Speech가 반환하는 형식은 response형식이므로 사용하기 변환

```
res = ClovaSpeechClient().req_upload(file='sample.mp4', completion='sync')  
  
res_json = json.loads(res.text)
```

- srt형식을 따르기 위해서 ms로 설정된 시간을 변환

```
def convertToTime(time):  
    hours = time // 3600000  
    time = time - hours * 3600000  
    mins = time // 60000  
    time = time - mins * 60000  
    secs = time / 1000  
    str_secs = str('{0:06.3f}'.format(secs)).replace(".", " ", "  
  
    str_time = str('{0:02}'.format(hours)) + ":" + str('{0:02}'.format(mins)) + ":" + str_secs  
  
    return str_time
```

# 자막 파일 생성 및 화자 인식

## 자막 파일 생성의 과제

-화자 인식을 자막에 표현해야함

```
subtitle = open("subtitle.srt", 'w', encoding='utf-8')
num = 1
for seg in res_json["segments"]:
    start_time = convertToTime(seg['start'])
    end_time = convertToTime(seg['end'])

    subtitle.write(str(num) + "\n" + start_time + " --> " + end_time + "\n" + seg['speaker']['name'] + " - " + seg['text'] + "\n\n")
    num = num + 1
```

# 자막 파일 생성 및 화자 인식

```
1
00:00:00,000 --> 00:00:02,000
A - 먼저 인사 좀 부탁드립니다

2
00:00:02,000 --> 00:00:04,030
B - 안녕하세요 김춘식입니다 반갑습니다

3
00:00:05,880 --> 00:00:07,540
B - 감사합니다. 못 받으실 거예요.

4
00:00:08,150 --> 00:00:13,120
C - 방송 이후에 여러 경험을 해볼 기회가 되게 많이 생겨서 어제부로

5
00:00:13,510 --> 00:00:15,010
B - 얼굴 좋아졌어요.

6
00:00:15,010 --> 00:00:16,010
D - 원래

7
00:00:16,010 --> 00:00:19,010
A - 회사를 가지고 얼굴이 펴요. 어때요
```

자막 파일 생성의 결과

- srt형식의 자막 생성
- 순서, 타임 라인, 화자 인식 및 자막 텍스트까지 출력되는 것 확인



# 자막 파일 실행

자막 파일 실행



# 로컬 서버 구축

## #1. package.json 파일을 구축

```
{} package.json X < index.html
{} package.json > ...
1 {
2   "name": "klhj",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node server.js",
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "dependencies": {
11    "ejs": "^3.1.6",
12    "express": "^4.17.1",
13    "textract": "^2.5.0"
14  },
15  "devDependencies": {},
16  "author": "",
17  "license": "ISC"
18 }
19
```

npm init을 이용

빠른 실행을 위해  
start script에  
Node server.js 추가

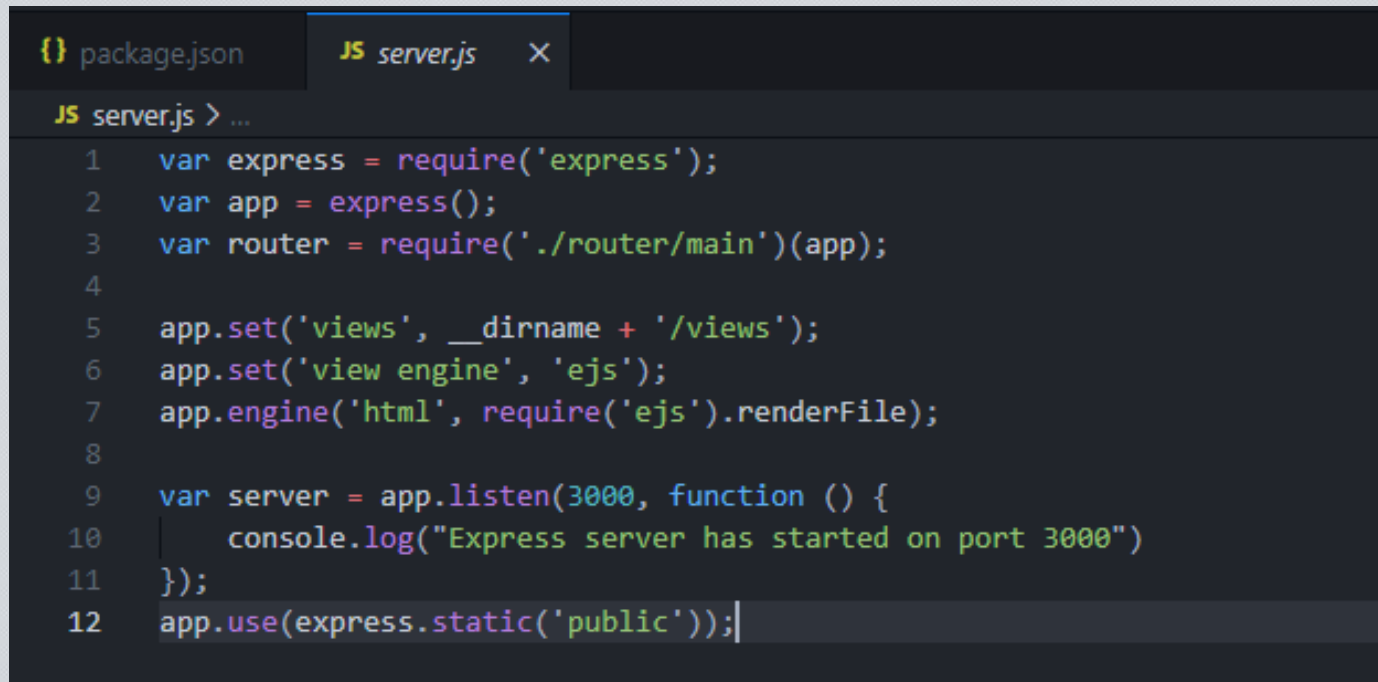
ejs, express, textract 설치

dependencies에 각 모듈 추가  
추후 사용해야하는 모듈들임



# 로컬 서버 구축

## #2. server.js 파일을 구축



The screenshot shows a code editor with two tabs: 'package.json' and 'server.js'. The 'server.js' tab is active, showing the following code:

```
1  var express = require('express');
2  var app = express();
3  var router = require('./router/main')(app);
4
5  app.set('views', __dirname + '/views');
6  app.set('view engine', 'ejs');
7  app.engine('html', require('ejs').renderFile);
8
9  var server = app.listen(3000, function () {
10    console.log("Express server has started on port 3000")
11  });
12  app.use(express.static('public'));
```

express를 이용해 라우터 구축, 로컬서버 실행 가능하도록 함



# 로컬 서버 구축

## #3. 각 사용에 알맞은 폴더 계층 구축

2021-1-CECD3-KIMLEEONGCHO-6

- > 발표자료
- > 서면보고
- > node\_modules
- public
  - index.js
  - myfile.odt
  - myfile2.odt
  - require.js
  - webodf.js
- router
  - main.js
- views
  - index.html
- .gitignore
- package-lock.json
- package.json
- README.md
- server.js

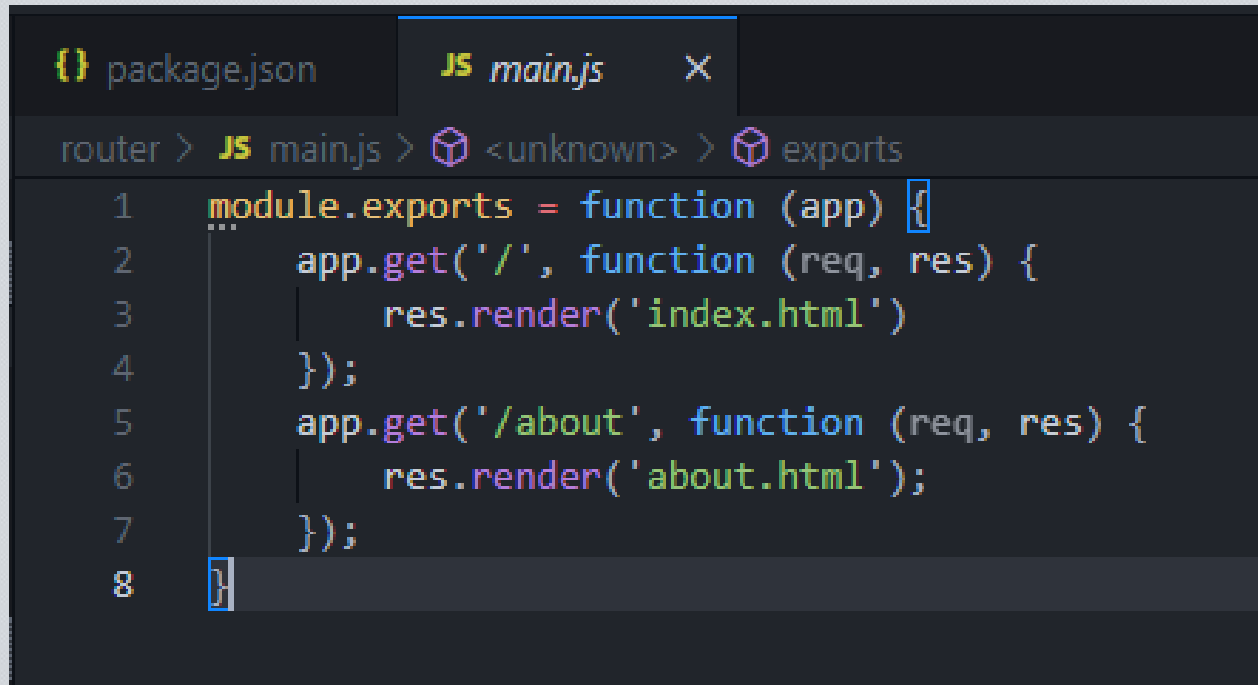
**public** → 정적 파일

**views** → 사용자에게 보이는 html파일

**router** → 라우터 모듈 파일

# 로컬 서버 구축

## #3. 각 사용에 알맞은 폴더 계층 구축



```
{} package.json  JS main.js  X  
router > JS main.js > <unknown> > exports  
1  module.exports = function (app) {  
2      app.get('/', function (req, res) {  
3          res.render('index.html')  
4      });  
5      app.get('/about', function (req, res) {  
6          res.render('about.html');  
7      });  
8  }
```

**main.js** → html 파일을 render 하여 화면에 띄우도록 하는 js 파일

# 감사합니다

김이홍조 14주차 발표

---