DEVIEW 2021

클라우드 Open AI API를 활용한 슬기로운 Toy Playground 개발하기

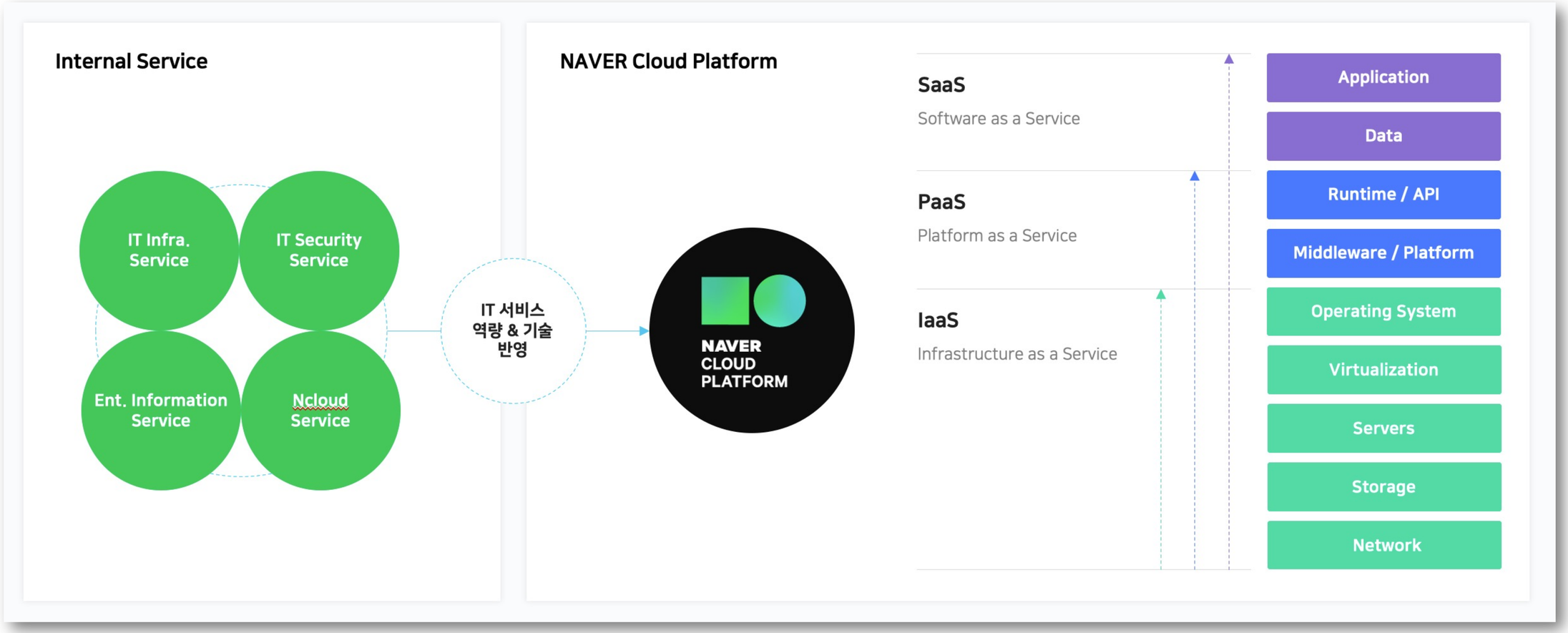
장창환 NAVER Cloud

CONTENTS

1. NAVER Cloud Platform Service Overview

Cloud Platform & Service

2017년 본격적으로 시작하여 매년 빠르게 성장하고 있는 퍼블릭 클라우드 서비스, 1999년부터 네이버와 계열사들의 서비스를 안정적으로 구축 및 운영해온 Internal Service 두 개의 사업 영역으로 나누어져 있습니다.



Service Map

안정적인 필수 인프라, 기업 특성에 맞춘 하이브리드 클라우드와 네이버의 데이터를 학습한 AI 및 기업정보시스템과 협업 툴 서비스를 제공하고 있습니다.

Compute

탄탄한 인프라와 오랜 운영 경험에 기반한 최상의 컴퓨팅 자원 제공

Network

언제 어디서나 빠르고 안정적인 네트워크 환경으로 막힘 없는 서비스 구현

Security

세계적 수준의 보안 기술로 외부 위협으로부터 서비스를 안전하게 보호

Management

인프라와 서비스를 실시간 모니터링 및 관리하여 안정적인 서비스 제공

Business Application

성장하는 기업을 위한 클라우드 기반 기업 정보 시스템 및 협업툴 제공

Media

쉽고 빠르게 고품질의 미디어 콘텐츠 변환 및 서비스 플랫폼 구축 가능

Game

게임 서비스를 위한 SDK 제공 및 다양한 필수 부가 서비스 연동 지원

Hybrid & Private Cloud

기업의 필요에 맞춰 다양한 형태의 하이브리드 클라우드 환경 제공

Database

서비스 특성에 맞는 다양한 데이터 플랫폼 제공, 관계형, NoSQL, 캐시 등

Global

전 세계 주요 거점에 구축한 안정적 인프라를 통해 글로벌 서비스 지원

Analytics

서비스 및 시스템 데이터의 효과적인 수집 및 통합 분석 서비스 제공

Storage

안전하고 유연한 스토리지 상품 제공, 아카이빙, 객체 스토리지, 백업 등

AI Service

네이버의 풍부한 데이터를 기반으로 학습된 경쟁력 있는 AI 서비스 활용 가능

Application Service

지도부터 캡챠까지 네이버의 기술과 서비스를 손쉽게 사용할 수 있도록 제공

IoT

수백만 대의 디바이스와 클라우드를 연결하여 쉽고 안전한 인사이트 확보

Dev Tools

DevOps를 적용하여 신속하고 안전하게 S/W 개발 환경 구축 및 배포

Blockchain

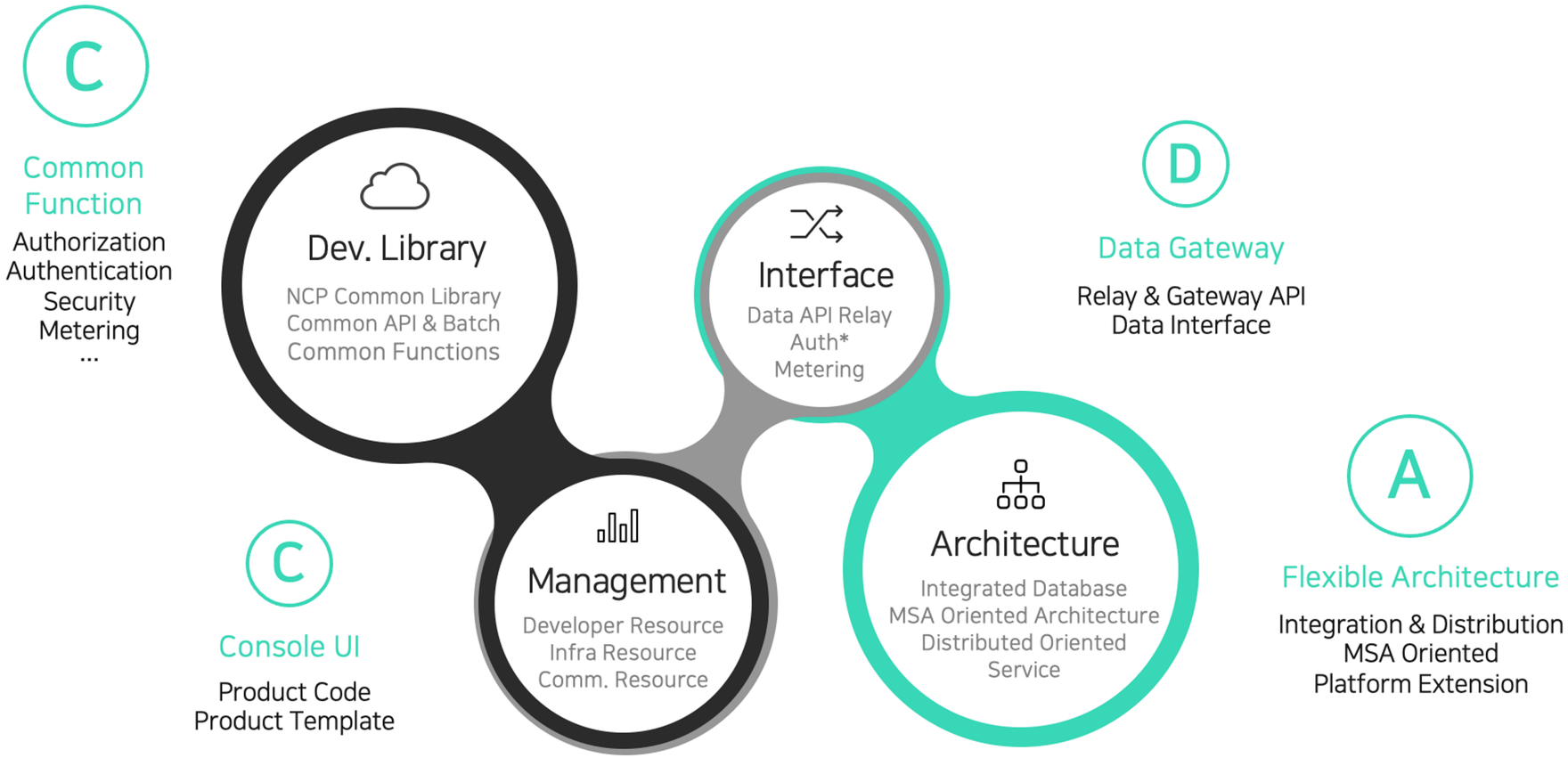
프라이빗 블록체인에서 블록체인 기반 서비스까지 클라우드에 구현

2. NAVER Cloud Platform SaaS Platform Overview

Key Feature of SaaS Development Platform

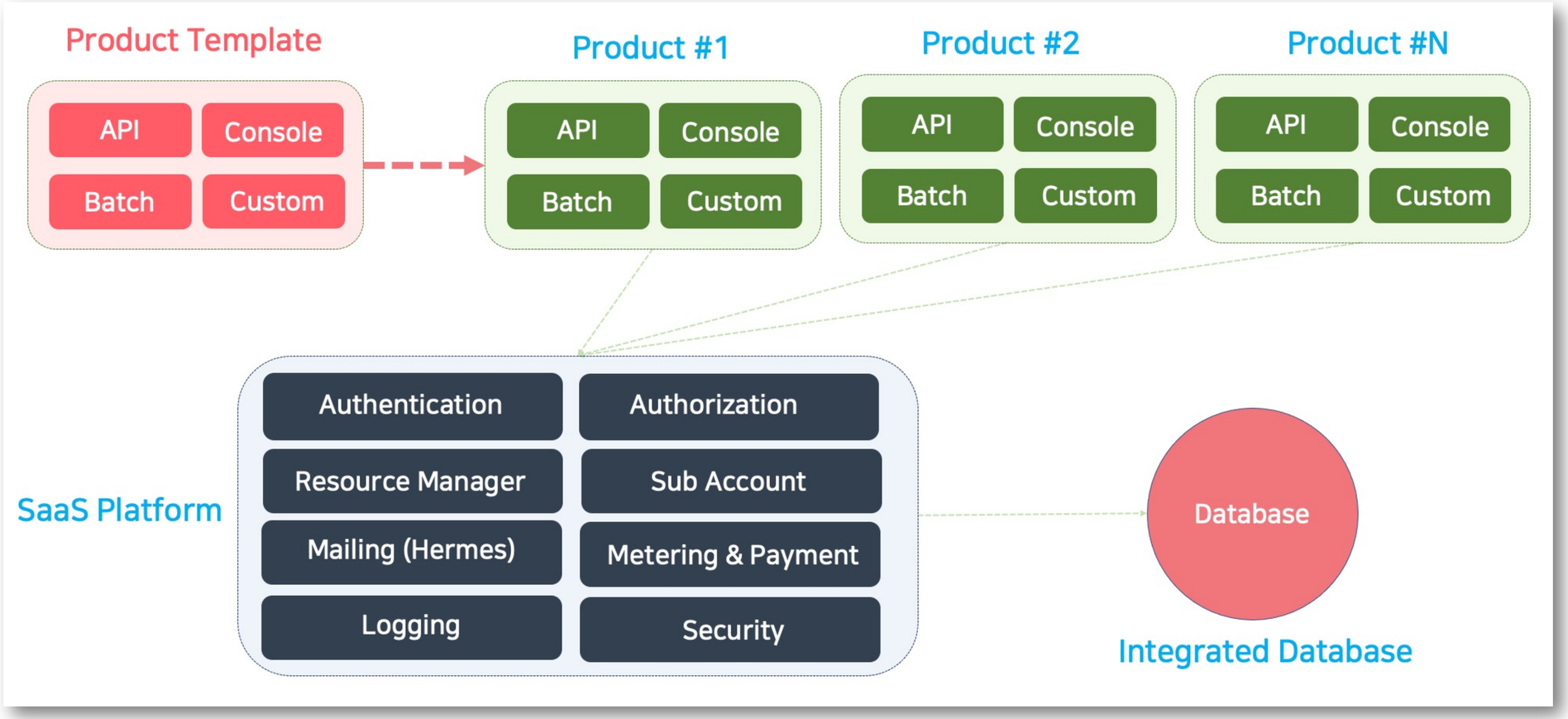
DEVIEW 2021

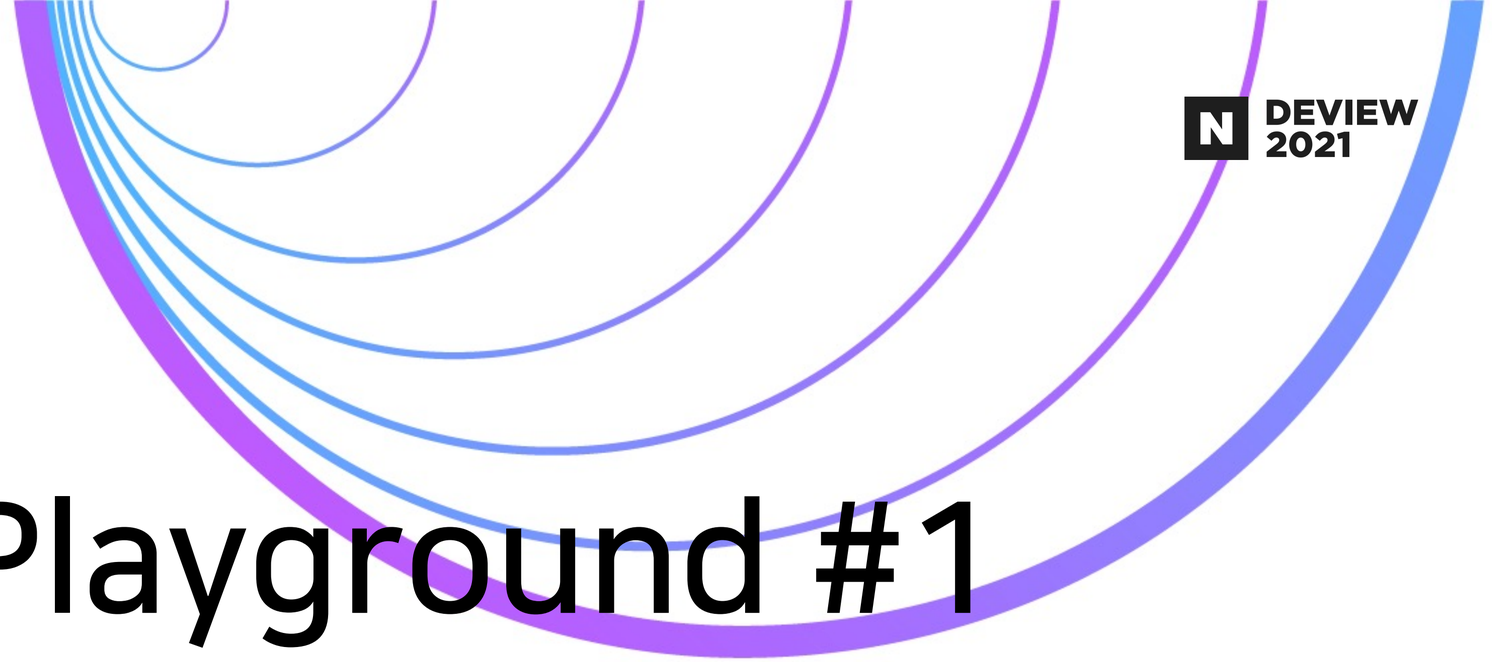
- Effective Resource Management of SaaS Products – Developer, Infra, Communication
- Advocation of MSA(Micro Service Architecture) & Minimized Dependency Points
- Support & Release Latest Common Cloud Library & SDKs, API



SaaS Platform Service Architecture

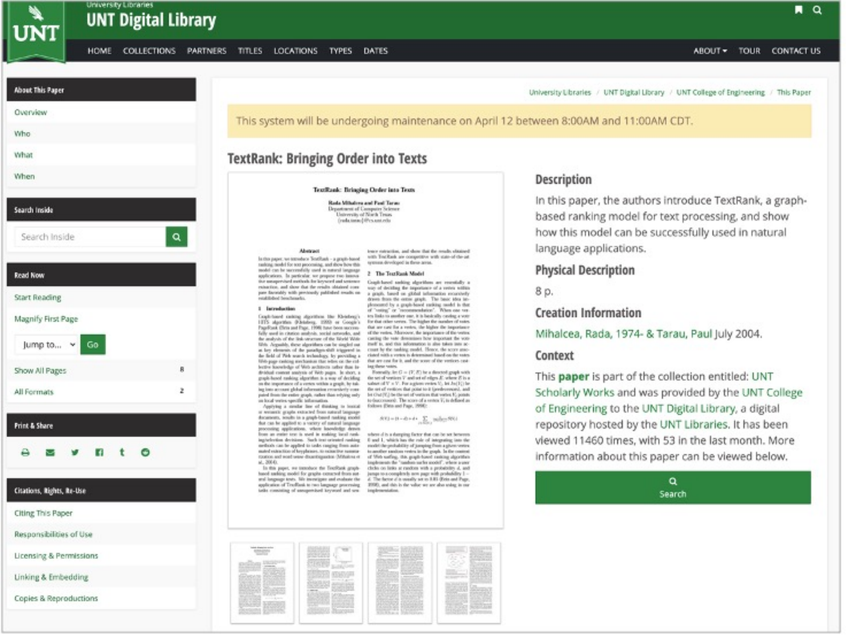
Common Function과 Custom Function 분리 아키텍처 적용, 상품 별 Customizing 기능은 별도 상품 서버 사용 가능
Monolithic Architecture를 지양하고, 최대한 MSA(Micro Service Architecture) 마이그레이션 가능하도록 설계

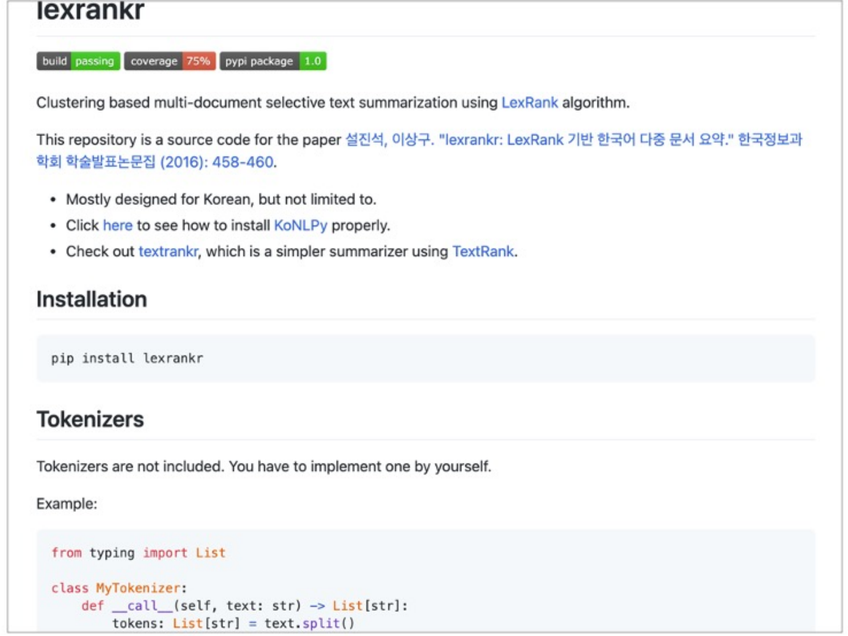


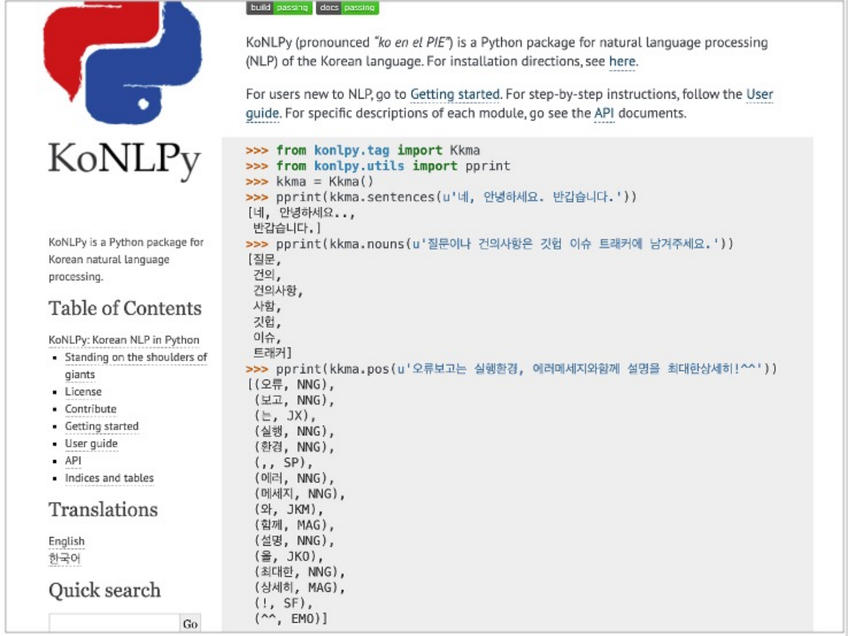


3. Toy Playground #1

TextRank Python 라이브러리를 활용한 My Summary Bot 서비스 구현하기

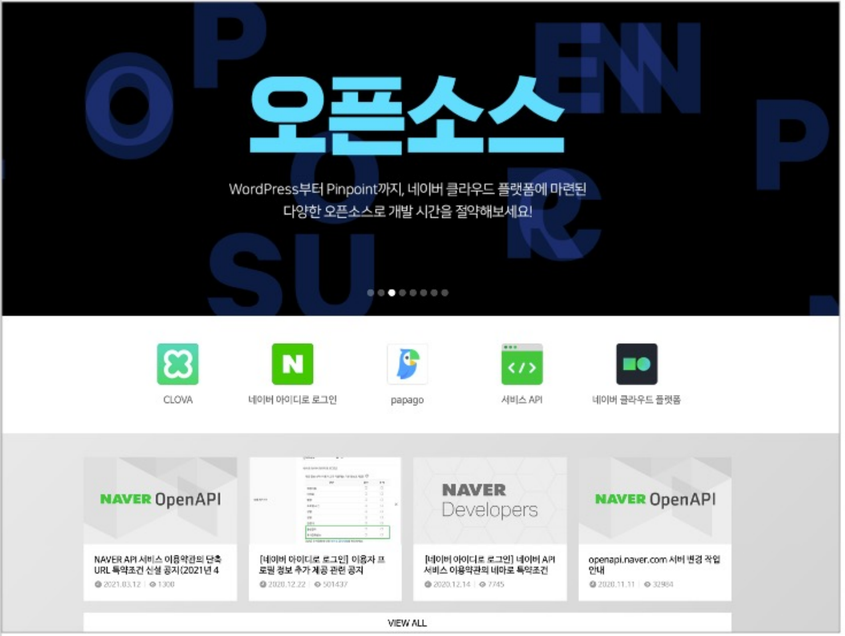


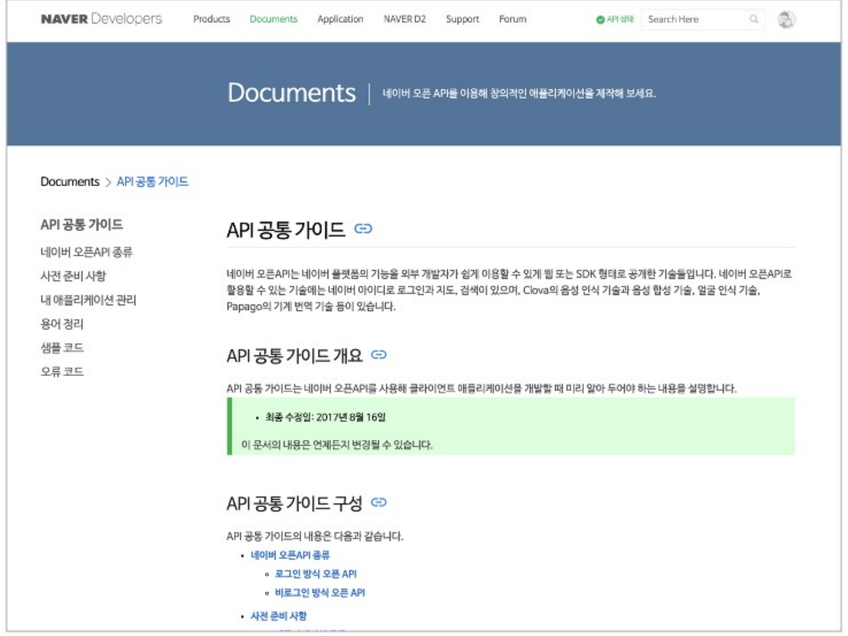


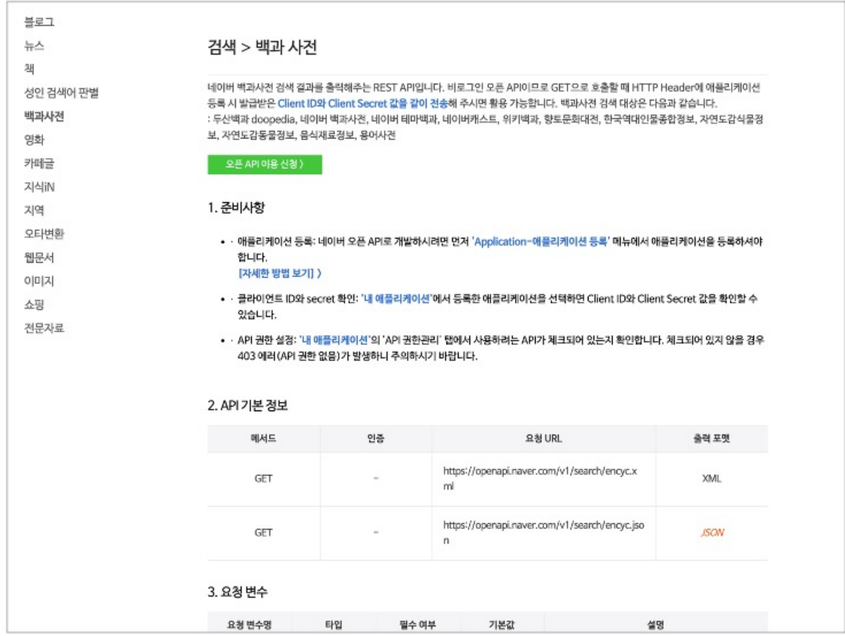


Textrankr, lexrankr with Python

- Graph-Based Ranking Model인 TextRank 활용, 자연어 응용 분야 활용
- Tokenizer 비 포함, KoNLPY를 활용하여 구현 가능
- Python Library & Node.js/Java 조합하여 독립 모듈 활용 가능

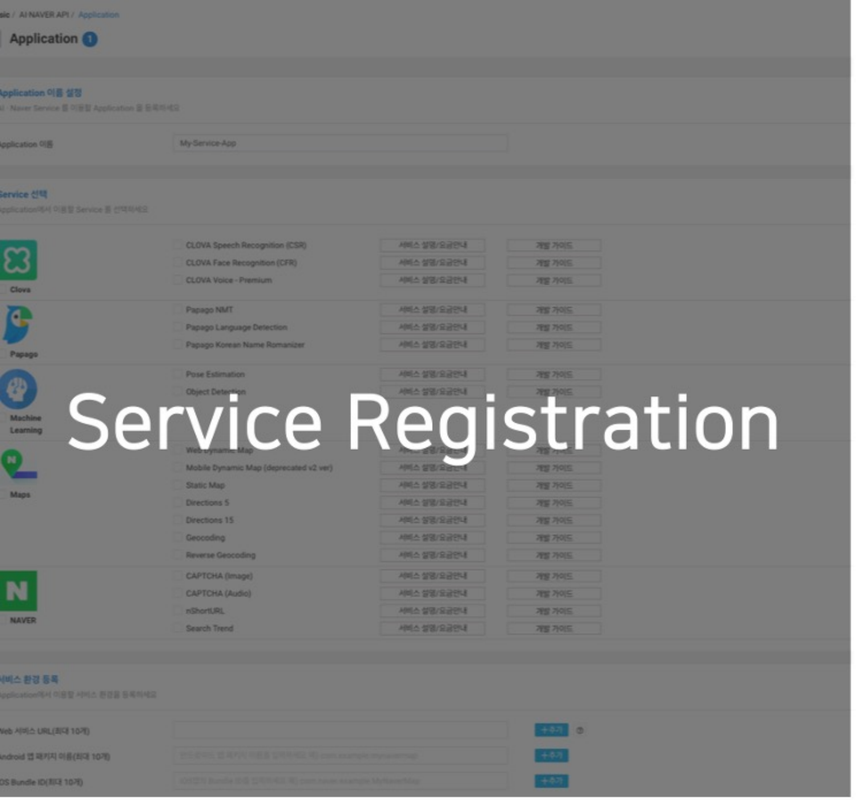




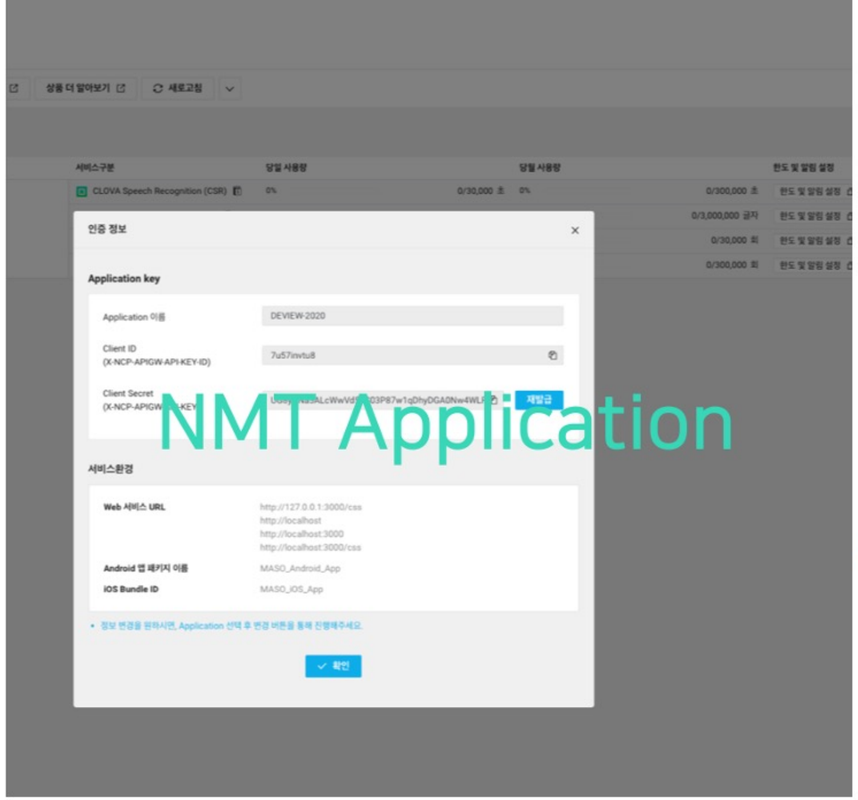


NAVER Developers API

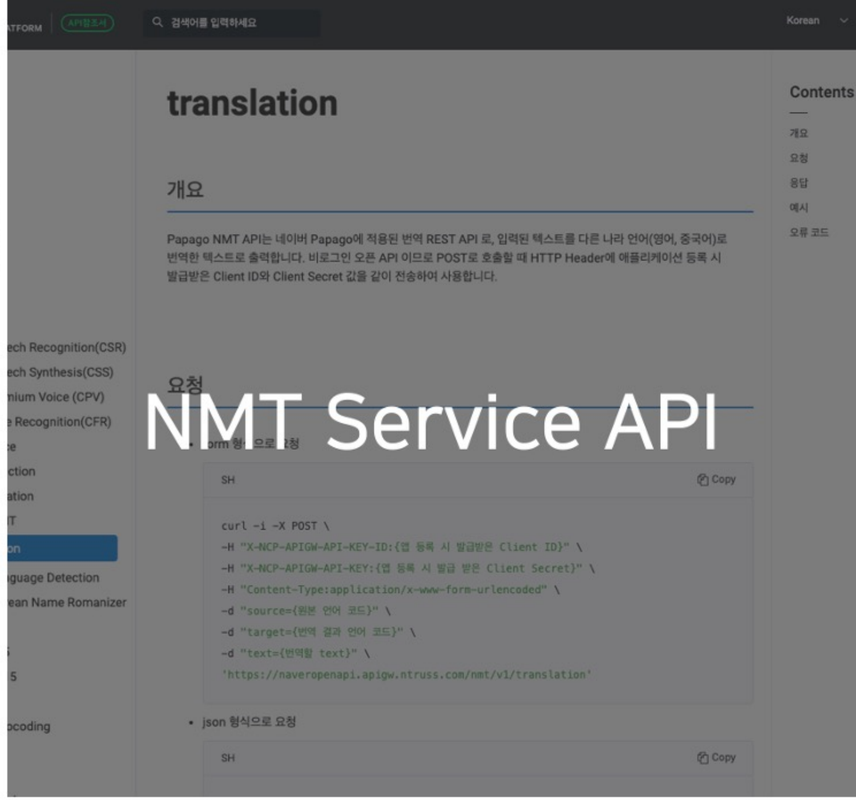
- NAVER 서비스에 축척된 데이터 및 기반 기술을 API 활용으로 서비스 적용
- B2B 서비스 구축 시에는 NAVER Cloud Platform 기반 API 서비스 활용 가능
- NAVER 사용자 서비스, AI 및 Big Data 기술, 오픈 메인 등 다양한 API 제공



Service Registration



NMT Application



NMT Service API

NAVER Cloud Platform PapagoNMT

- 가장 뛰어난 한국어 번역 품질을 제공하는 인공 신경망기반 기계 번역 API
- 풍부한 한국어 언어 처리 경험과 학습을 통해 높아지는 성능 기반 서비스 가능
- 안전한 개인정보 관리와 자동 언어감지 기능, 높임말 번역 기능 제공

PapagoNMT API Specification

DEVIEW 2021

Client ID / Secret Key / Context-Type / Text(Search)

PapagoNMT API Request

요청 헤더

헤더명	설명
X-NCP-APIGW-API-KEY-ID	앱 등록 시 발급받은 Client ID `X-NCP-APIGW-API-KEY-ID:{Client ID}`
X-NCP-APIGW-API-KEY	앱 등록 시 발급 받은 Client Secret `X-NCP-APIGW-API-KEY:{Client Secret}`

Content-Type	전송할 콘텐츠 형식
	`Content-Type:application/x-www-form-urlencoded`
	`Content-Type:application/json`

요청 바디

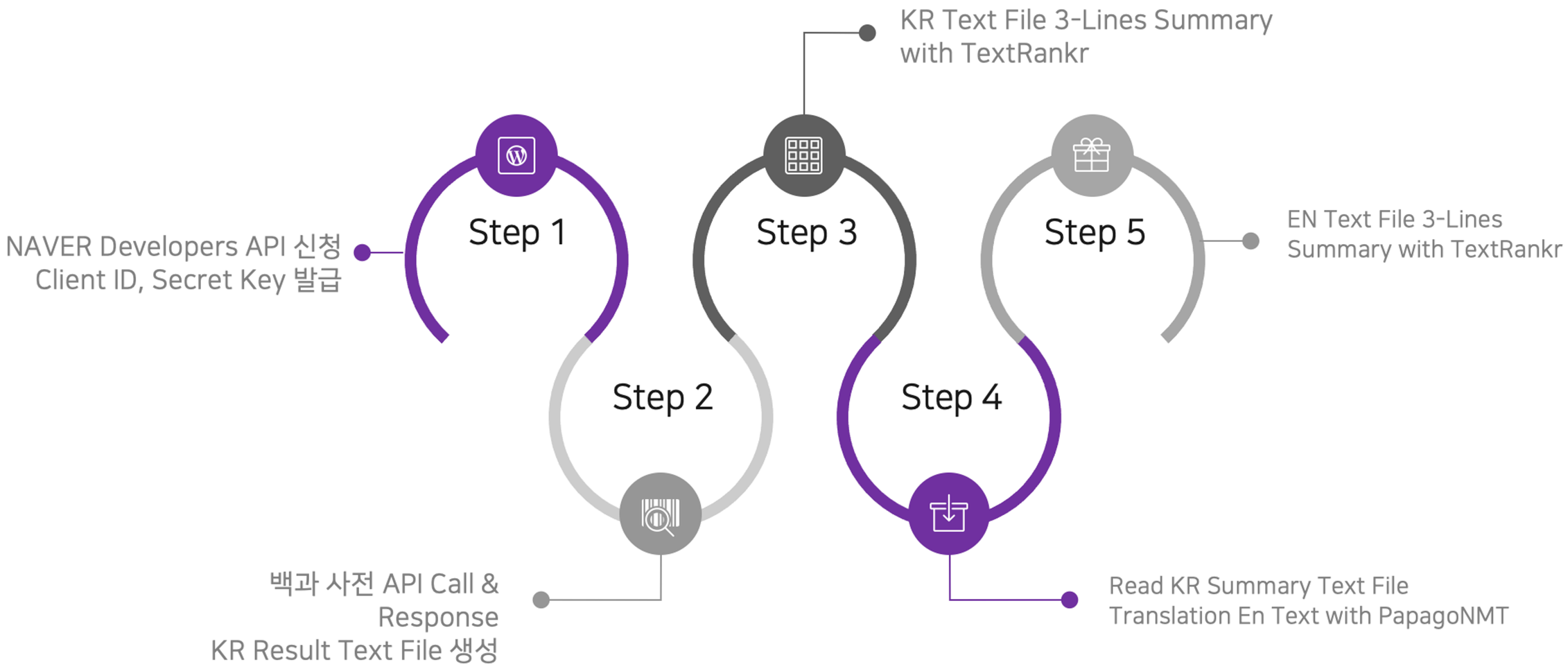
파라미터	타입	필수 여부	설명
source	String	Y	원본 언어(source language)의 언어 코드
target	String	Y	목적 언어(target language)의 언어 코드
text	String	Y	번역할 텍스트. 1회 호출 시 최대 5,000자까지 번역할 수 있습니다.
honorific	Boolean	N	높임말 여부. 영>한 번역에서만 적용됩니다. 기본값은 False

PapagoNMT API Response

응답 바디

필드명	타입	설명
srcLangType	string	원본 언어 코드
tarLangType	string	번역 결과 언어 코드
translatedText	string	번역된 문장

My Summary Bot Service



Implementation of My Summary Bot

Implementation (1/5)

- NVM 설치
- NPM Module 설치

```
{
  "name": "myapp",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "boolbase": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/boolbase/-/boolbase-1.0.0.tgz",
      "integrity": "sha1-aN/1++YMUes3cl6p4+0xDcwed24="
    },
    "cheerio": {…
    },
    "cheerio-select-tmp": {…
    },
    "css-select": {…
    },
    "css-what": {…
    },
    "dom-serializer": {…
    },
    "domelementtype": {…
    },
    "domhandler": {…
    },
    "domutils": {…
    },
    "entities": {…
    },
    "htmlparser2": {…
    },
    "nth-check": {…
    },
    "parse5": {…
    },
    "parse5-htmlparser2-tree-adapter": {…
    },
    "python-shell": {
      "version": "0.5.0",
      "resolved": "https://registry.npmjs.org/python-shell/-/python-shell-0.5.0.tgz",
      "integrity": "sha512-+jgmFZvwk1yMBBDisDlkXXMYv1eEJKbGCtwHLppGIyEV83cKeX9hjOjfR2yONWK3yQFhum0M2r7UE0U//hi
    }
  }
}
```

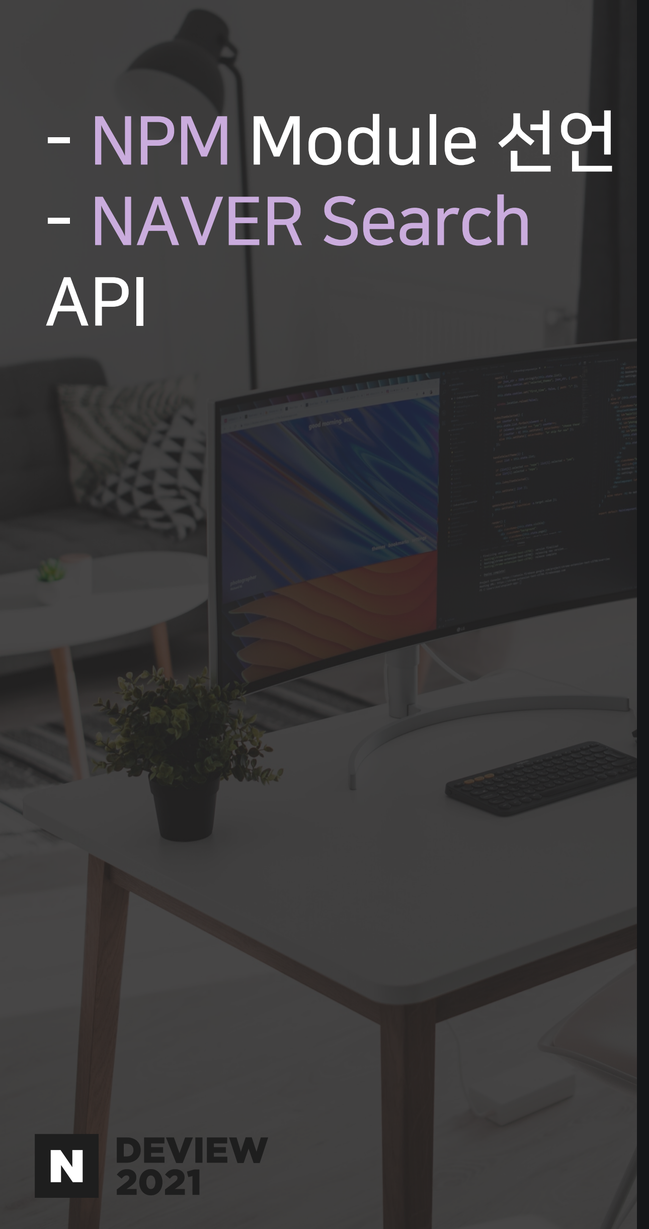
① package.json

② Required NPM Package

③ Execution of Python Package

Implementation (2/5)

- NPM Module 선언
- NAVER Search API



```
1  // import 'express' module
2  var express = require('express');
3
4  // import 'axios' module
5  const axios = require('axios');
6
7  // Encoding QueryString
8  const qs = require('querystring');
9
10 // File Stream
11 var fs = require('fs');
12
13 // Nodejs Dom Service
14 const cheerio = require('cheerio');
15
16 // Run Python in node js
17 var PythonShell = require('python-shell');
18
19 var app = express();
20 var client_id = '{YOUR_NMT_CLIENT_ID}';
21 var client_secret = '{YOUR_NMT_SECRET_KEY}';
22
23 app.get('/summaryContens/:query/:en_file', function (req, res) {
24
25   console.log('::: Search Keyword : ' + req.params.query);
26   console.log('::: En Filename : ' + req.params.en_file);
27
28   // for NAVER Search API (백과사전))
29   var config = {
30     headers: {
31       'X-Naver-Client-Id' : client_id,
32       'X-Naver-Client-Secret' : client_secret
33     }
34   };
35
```

① Required NPM Package

② Execution Python Module In Node.js

③ NAVER Search API Options

Implementation (3/5)

- NAVER Search API 호출 & 결과
- KR Text 파일 저장

```
40  var textStr = '';
41  var resultStr;
42
43  // for Browser Print
44  var htmlStr = '<p><b><font color="orange">[ Search Keyword ]</font></b></p>'+
45                '<b>' + req.params.query + '</b><br><br>' +
46                '</b><br><br><p><b><font color="blue">[ Result of Search ]</font></b></p>';
47
48  // NAVER Search API (백과사전))
49  axios.get(
50    `https://openapi.naver.com/v1/search/encyc.json?query=${qs.escape(req.params.query)}&display=1&start=1`,
51    config
52  )
53    .then( response=>{
54
55      res.writeHead(200, { 'Content-Type': 'text/html;charset=UTF-8' });
56
57      htmlStr += response.data.items[0].title + '<br>' +
58                 response.data.items[0].link + '<br>' +
59                 response.data.items[0].description + '<br>' +
60                 '<img src="' + response.data.items[0].thumbnail + '"><br>';
61
62      var http = require('http');
63      var dest = req.params.en_file;
64      var url = response.data.items[0].link;
65
66      var fileTextStr;
67
68      // NAVER Search API Call
69      axios(url)
70      .then(res => res.data)
71      .then(html => {
72
73          // Convert HTML to Plain Text
74          const $ = cheerio.load(html);
75
76          // Text File Create & Save
77          fileTextStr = $('#size_ct').text();
78          fs.writeFileSync('./text_contents/ko/'+dest+'.txt', fileTextStr.trim());
79
80          // Text Summary with Python
81          // Exectue to TextRank
82          var options = {
83            mode: 'text',
84            pythonPath: '',
85            pythonOptions: ['-u'],
86            scriptPath: '',
87            args: [fileTextStr.trim()]
88          };
```

1 NAVER Search API URL

2 Response Search Result

3 Write KR Text File with Search Contents

4 Python Shell Execution Options

Implementation (4/5)

- 3-Lines 요약
- PapagoNMT API

```
90          var arrayStr='';
91
92          // Execution for Python in node js
93          PythonShell.run('summary_contents.py', options, function (err, results) {
94            if (err) throw err;
95
96            for (var i = 0; i < results.length; i++) {
97              arrayStr += '*.'+results[i] + "\n";
98            }
99
100           // Save Summary Korean File
101           fs.writeFileSync('./text_contents/ko/'+dest+'_summary.txt', arrayStr.trim());
102         });
103
104       }).then(returnStr => {
105
106         // Translated News to Translate with Papago NMT
107         var api_url = 'https://naveropenapi.apigw.ntruss.com/nmt/v1/translation';
108         var request = require('request');
109         var tranTextStr = fs.readFileSync('./text_contents/ko/'+dest+'.txt', 'utf8');
110
111         // PapagoNMT HTTP Options
112         var options = {
113             url: api_url,
114             form: {'source':'ko', 'target':'en', 'text':tranTextStr},
115             headers: {
116               'X-NCP-APIGW-API-KEY-ID':client_id,
117               'X-NCP-APIGW-API-KEY': client_secret,
118               'Content-Type': 'application/json'
119             }
120         };
121
122         request.post(options, function (error, response, body) {
123           var tranlatedJSON = JSON.parse(body);
124           var tranlatedJSONStr = JSON.stringify(tranlatedJSON);
125
126           // Save Summary En File
127           fs.writeFileSync('./text_contents/en/'+dest+'.txt', JSON.stringify(tranlatedJSON.message.result.trans
128
129           var options = {
130             mode: 'text',
131             pythonPath: '',
132             pythonOptions: ['-u'],
133             scriptPath: '',
134             args: [tranlatedJSONStr.trim()]
135           };
```

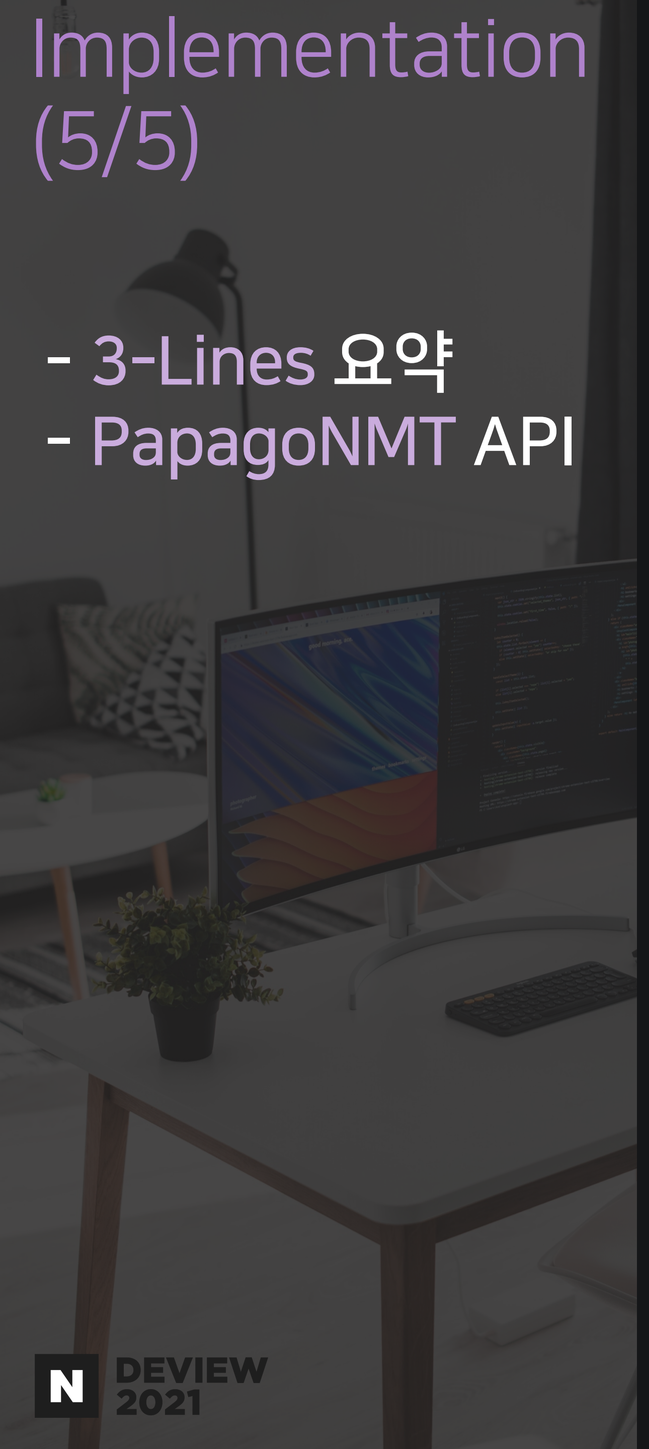
① Summary 3-Lines KR Text

② PapagoNMT Translation API

③ Write EN Text File with Translation Contents

Implementation (5/5)

- 3-Lines 요약
- PapagoNMT API



```
137            var arrayStr_en='';
138
139            // for Python in node js
140            PythonShell.run('summary_contents.py', options, function (err, results) {
141
142              if (err) throw err;
143
144              for (var i = 0; i < results.length; i++) {
145                arrayStr_en += '*.'+results[i] + "\n";
146              }
147
148              // Save Summary English File
149              fs.writeFileSync('./text_contents/en/'+dest+'_summary.txt', arrayStr_en.trim());
150            });
151
152          });
153
154          var summaryKoTextStr = fs.readFileSync('./text_contents/ko/'+dest+'_summary.txt', 'utf8');
155          htmlStr += '</b><br><br><p><b><font color="green">[ Summary Ko Text ]</font></b></p>'+
156                  '<b>' + summaryKoTextStr + '</b><br><br>';
157
158          var summaryEnTextStr = fs.readFileSync('./text_contents/en/'+dest+'_summary.txt', 'utf8');
159          htmlStr += '</b><br><br><p><b><font color="#1E90FF">[ Summary En Text ]</font></b></p>'+
160                  '<b>' + summaryEnTextStr + '</b><br><br>';
161
162          res.write(htmlStr);
163          res.end();
164
165        });
166      })
167      .catch( error =>{
168        console.log( error );
169      })
170  });
171
172  app.listen(3000, function () {
173    console.log('::: MySummaryBot Service App listening on port 3000!');
174  });
```

① Summary 3-Lines EN Text

② Write EN Text File with EN_Contents

③ Web Print Text KR + EN Contents

④ Start Node Server

Execution Result of My Summary Bot

Execution Result (1/6)

1 Node Server Start

```
~/CloudToyService/MySummaryBot git master ❯                                    10.19.0
~/CloudToyService/MySummaryBot git master ❯ node app_server.js                 10.19.0
::: MySummaryBot Service App listening on port 3000!
```

2 Call Log from Web Browser

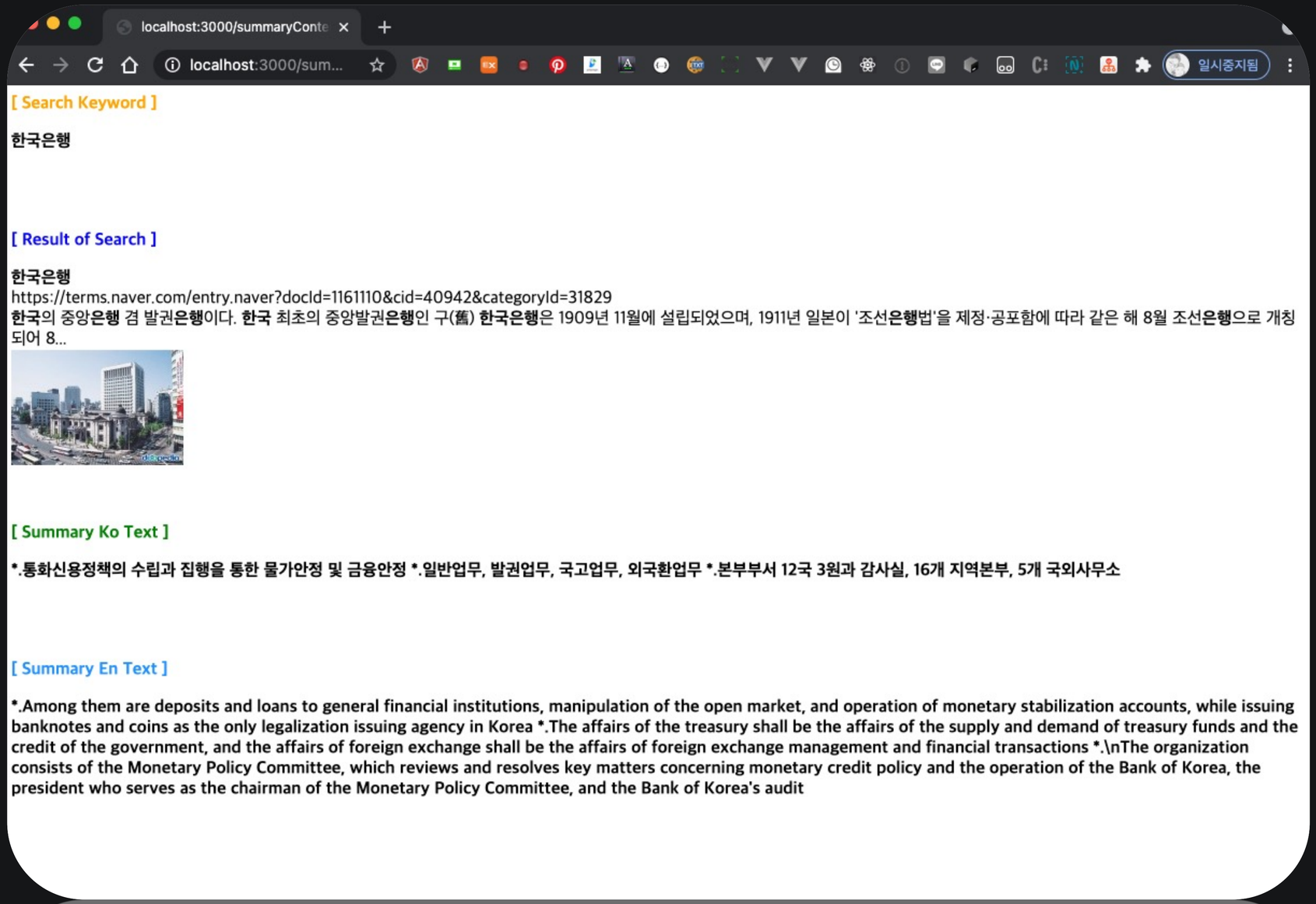
```
~/CloudToyService/MySummaryBot git master ?4 ❯ node app_server.js   ✘ INT ⌛ 3m 23s 10.19.0
::: MySummaryBot Service App listening on port 3000!




::: Search Keyword : 한국은행
::: En Filename : krbank
```

3 Project Scaffolding

```
~/CloudToyService/MySummaryBot git master ?4 ❯ ll
total 3840
-rw-r--r--   1 naver  staff   154B  3 24 16:08 MyTokenizer.py
-rw-rw-r--@  1 naver  staff   329B  3 24 13:26 README.md
-rw-rw-r--@  1 naver  staff   5.4K  4 12 15:18 app_server.js
-rw-r--r--   1 naver  staff   1.8M  3 24 14:15 get-pip.py
drwxr-xr-x  17 naver  staff   544B  3 24 15:03 node_modules
-rw-r--r--   1 naver  staff   5.0K  3 24 15:03 package-lock.json
-rw-rw-r--   1 naver  staff   657B  3 24 15:03 package.json
-rwxr-xr-x@  1 naver  staff   910B  3 24 17:43 summary_contents.py
-rw-rw-r--@  1 naver  staff   595B  3 24 13:26 test_summary.js
drwxrwxr-x@  5 naver  staff   160B  3 24 16:43 text_contents
drwxrwxr-x@ 13 naver  staff   416B  3 24 14:42 textrankr-master
~/CloudToyService/MySummaryBot git master ?4 ❯
~/CloudToyService/MySummaryBot git master ?4 ❯ _
```

Execution Result (2/6)

http://localhost:3000/summaryContens/한국은행/krbank



Execution Result (3/6)

KR Files in Directory

```
 ~/CloudToyService/MySummaryBot/text_contents/ko git master ?4 ❯
 ~/CloudToyService/MySummaryBot/text_contents/ko git master ?4 ❯ pwd
/Users/naver/CloudToyService/MySummaryBot/text_contents/ko
 ~/CloudToyService/MySummaryBot/text_contents/ko git master ?4 ❯
 ~/CloudToyService/MySummaryBot/text_contents/ko git master ?4 ❯
 ~/CloudToyService/MySummaryBot/text_contents/ko git master ?4 ❯ ll
total 432
-rw-r--r--  1 naver  staff   2.2K  3 25 18:57 air.txt
-rw-r--r--  1 naver  staff   171B  3 25 18:57 air_summary.txt
-rw-r--r--  1 naver  staff   2.3K  3 25 18:58 badair.txt
-rw-r--r--@ 1 naver  staff   349B  3 25 18:58 badair_summary.txt
-rw-r--r--@ 1 naver  staff   1.3K  3 24 17:10 bank.txt
-rw-r--r--@ 1 naver  staff   319B  3 24 17:10 bank_summary.txt
-rw-r--r--@ 1 naver  staff   3.9K  3 24 18:24 koreanbank.txt
-rw-r--r--@ 1 naver  staff   221B  3 24 18:24 koreanbank_summary.txt
-rw-r--r--@ 1 naver  staff   2.5K  3 24 17:43 kosdak.txt
-rw-r--r--@ 1 naver  staff   608B  3 24 17:43 kosdak_summary.txt
-rw-r--r--@ 1 naver  staff   948B  3 24 17:15 kospi.txt
-rw-r--r--@ 1 naver  staff   561B  3 24 17:15 kospi_summary.txt
-rw-r--r--@ 1 naver  staff   3.9K  4 12 15:20 krbank.txt
-rw-r--r--  1 naver  staff   221B  4 12 15:20 krbank_summary.txt
-rw-r--r--  1 naver  staff   919B  3 24 18:19 midbank.txt
-rw-r--r--@ 1 naver  staff   153B  3 24 18:19 midbank_summary.txt
-rw-r--r--  1 naver  staff    23K  3 25 18:56 minyo.txt
-rw-rw-r--@ 1 naver  staff    14K  3 24 13:57 olympic.txt
-rw-r--r--  1 naver  staff   8.1K  3 24 18:13 securemoney.txt
-rw-r--r--@ 1 naver  staff   415B  3 24 18:13 securemoney_summary.txt
-rw-rw-r--@ 1 naver  staff    18K  3 24 17:52 security.txt
-rw-rw-r--@ 1 naver  staff   548B  3 24 13:26 security_summary.txt
-rw-r--r--  1 naver  staff   3.9K  3 25 18:56 song.txt
-rw-r--r--  1 naver  staff   205B  3 25 18:56 song_summary.txt
-rw-r--r--  1 naver  staff   9.5K  3 24 14:30 space.txt
-rw-r--r--  1 naver  staff    17K  3 24 17:10 sun.txt
-rw-r--r--@ 1 naver  staff   265B  3 24 17:10 sun_summary.txt
-rw-r--r--@ 1 naver  staff    14K  3 25 18:55 tree.txt
-rw-r--r--@ 1 naver  staff   1.1K  3 24 16:55 tree_summary.txt
-rw-r--r--  1 naver  staff   2.5K  3 25 18:56 trot.txt
-rw-r--r--  1 naver  staff   309B  3 25 18:56 trot_summary.txt
```

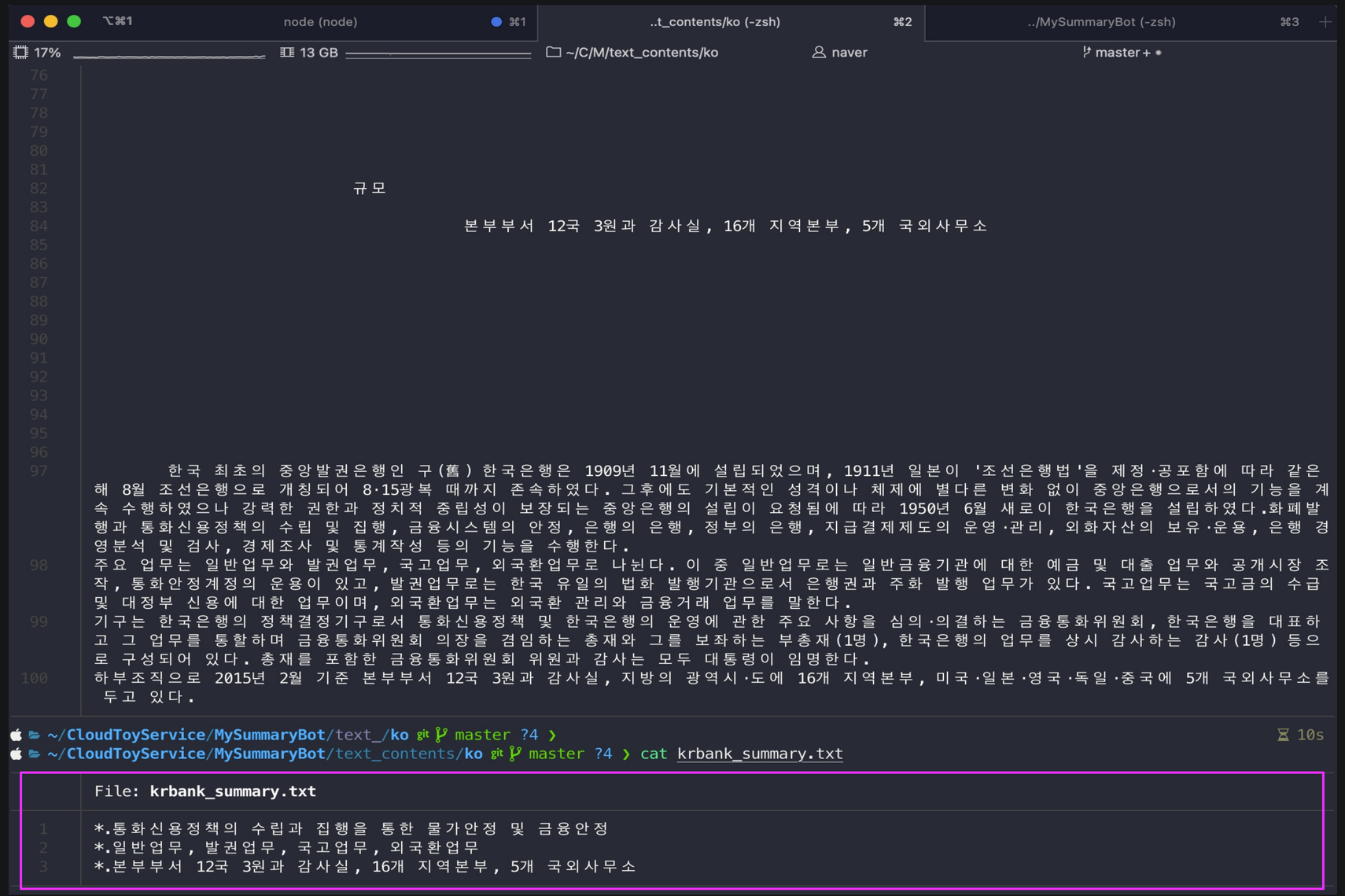
Execution Result (4/6)

EN Files in Directory

```
 ~/CloudToyService/MySummaryBot/text_contents/en  master ?4 ❯ pwd
/Users/naver/CloudToyService/MySummaryBot/text_contents/en
 ~/CloudToyService/MySummaryBot/text_contents/en  master ?4 ❯
 ~/CloudToyService/MySummaryBot/text_contents/en  master ?4 ❯
 ~/CloudToyService/MySummaryBot/text_contents/en  master ?4 ❯ ll
total 224
-rw-r--r--  1 naver  staff   1.8K  3 25 18:57 air.txt
-rw-r--r--  1 naver  staff   337B  3 25 18:57 air_summary.txt
-rw-r--r--  1 naver  staff   2.1K  3 25 18:58 badair.txt
-rw-r--r--@ 1 naver  staff   548B  3 25 18:58 badair_summary.txt
-rw-rw-r--@ 1 naver  staff   2.8K  3 24 13:26 koreabank.txt
-rw-rw-r--@ 1 naver  staff   1.9K  3 24 13:26 koreabank_summary.txt
-rw-r--r--  1 naver  staff   4.1K  3 24 18:24 koreanbank.txt
-rw-r--r--  1 naver  staff   761B  3 24 18:24 koreanbank_summary.txt
-rw-r--r--@ 1 naver  staff   2.5K  3 24 17:43 kosdak.txt
-rw-r--r--@ 1 naver  staff   665B  3 24 17:43 kosdak_summary.txt
-rw-r--r--@ 1 naver  staff   4.1K  4 12 15:20 krbank.txt
-rw-r--r--@ 1 naver  staff   761B  4 12 15:20 krbank_summary.txt
-rw-r--r--  1 naver  staff   1.0K  3 24 18:19 midbank.txt
-rw-r--r--  1 naver  staff   630B  3 24 18:19 midbank_summary.txt
-rw-r--r--  1 naver  staff   8.6K  3 24 18:13 securemoney.txt
-rw-r--r--@ 1 naver  staff   803B  3 24 18:13 securemoney_summary.txt
-rw-rw-r--@ 1 naver  staff   1.7K  3 24 13:26 security.txt
-rw-rw-r--@ 1 naver  staff   504B  3 24 13:26 security_summary.txt
-rw-r--r--  1 naver  staff   3.5K  3 25 18:56 song.txt
-rw-r--r--  1 naver  staff   288B  3 25 18:56 song_summary.txt
-rw-r--r--@ 1 naver  staff   868B  3 24 16:55 tree.txt
-rw-r--r--@ 1 naver  staff   929B  3 24 16:55 tree_summary.txt
-rw-r--r--  1 naver  staff   2.1K  3 25 18:56 trot.txt
-rw-r--r--  1 naver  staff   431B  3 25 18:56 trot_summary.txt
```

Execution Result (5/6)

KR Summary File Contents



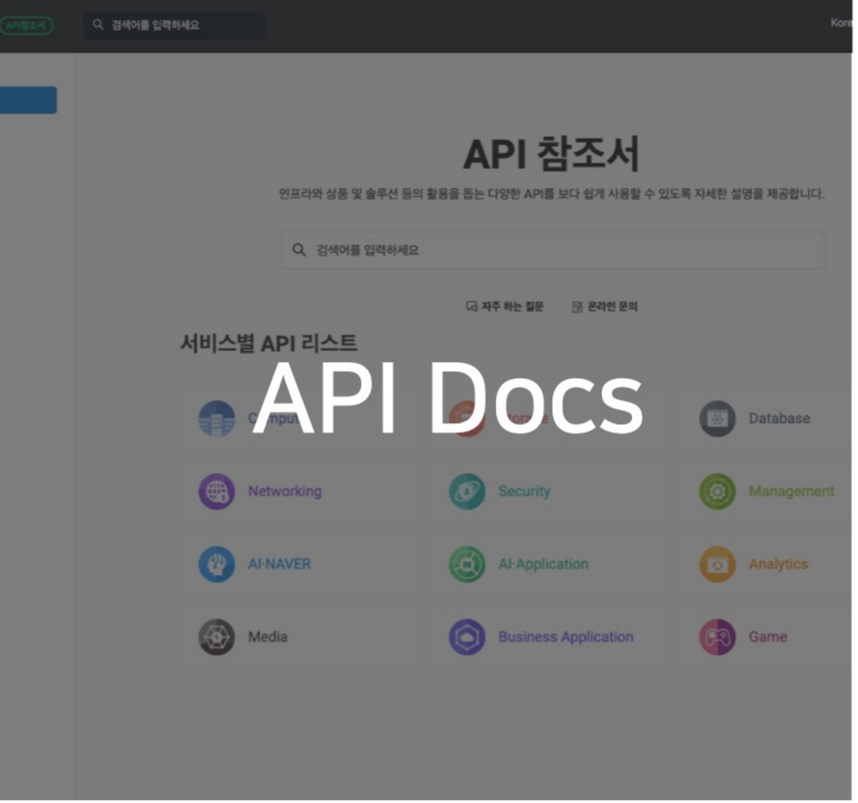
Execution Result (6/6)

EN Summary File Contents

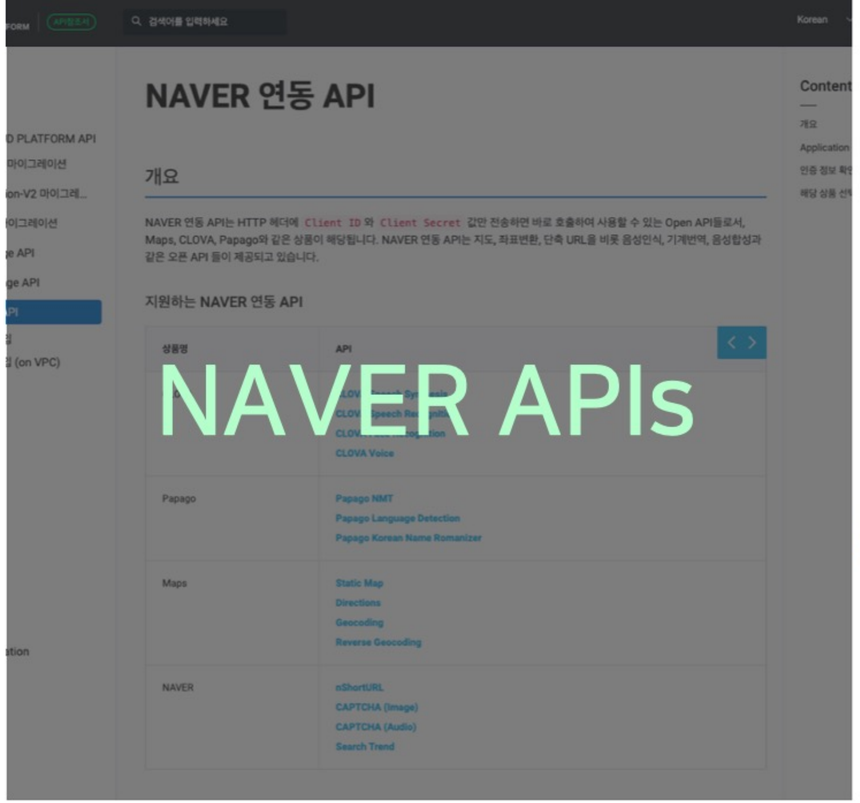
```
⌥⌘1 node (node) ⌘1 ..t_contents/en (-zsh) ⌘2 ../MySummaryBot (-zsh) ⌘3
12% 14 GB ~/C/M/text_contents/en naver master+ •

 \n \n\n\n \n Large View of Image\n \n \n\n\n Bank of Korea Main Building in
Seoul\n\n \n \n \n\n\t\n \n \n \n \n
 \n \n \n\n\n \n Sortation\n
 \n a financial institution\n\n \n
\n\n\n \n Date of establishment\n \n
 1950\n\n \n \n\n\n \n
 Purpose of establishment\n \n Price stability a
nd financial stability through the establishment and implementation of monetary credit policies;\n\n \n
 \n\n\n \n Major activities/tasks\n
 \n General business, issuing business, treasury business, foreign exchange business\n\n
 \n \n\n\n \n Location\n
 \n 39 Namdaemun-ro, Jung-gu, Seoul (110 Namdaemun-ro 3-ga)\n\n
 \n \n\n\n \n Scale\n
 \n Headquarters Department 3 won, audit office, 16 regional headquarters, 5 forei
gn offices;\n\n \n \n \n \n \n \n \n\n\n\n
 \n \n The former Bank of Korea, Korea's first central issuing bank, was established in November 1909, and was renamed the
 Chosun Bank in August of the same year and remained until the liberation of August 15. It continued to function as a central bank wit
hout any changes in its basic character or system, but was newly established in June 1950 when it was requested to establish a central
 bank with strong authority and political neutrality.The institution implements functions such as issuing currency and implementing mo
netary and credit policies, stabilizing financial systems, operating and managing banks, government banks, payment systems, holding an
d operating foreign currency assets, bank management analysis and inspection, economic surveys and statistics. \nThe main tasks are di
vided into general affairs, issuance, treasury affairs, and foreign exchange affairs. Among them are deposits and loans to general fin
ancial institutions, manipulation of the open market, and operation of monetary stabilization accounts, while issuing banknotes and co
ins as the only legalization issuing agency in Korea. The affairs of the treasury shall be the affairs of the supply and demand of tre
asury funds and the credit of the government, and the affairs of foreign exchange shall be the affairs of foreign exchange management
and financial transactions. \nThe organization consists of the Monetary Policy Committee, which reviews and resolves key matters conce
rning monetary credit policy and the operation of the Bank of Korea, the president who serves as the chairman of the Monetary Policy C
ommittee, and the Bank of Korea's audit. All members and auditors of the Monetary Policy Committee, including the president, are appoi
nted by the president. \nAs of February 2015, it has three sub-organizations in 12 countries and audit offices, 16 regional headquarte
rs in local metropolitan cities and provinces, and five overseas offices in the United States, Japan, the United Kingdom, Germany, and
 China."

~/CloudToyService/MySummaryBot/text_contents/en git master ?4 ❯ cat krbank_summary.txt

File: krbank_summary.txt

1 *.Among them are deposits and loans to general financial institutions, manipulation of the open market, and operation of monetary stab
ilization accounts, while issuing banknotes and coins as the only legalization issuing agency in Korea
2 *.The affairs of the treasury shall be the affairs of the supply and demand of treasury funds and the credit of the government, and th
e affairs of foreign exchange shall be the affairs of foreign exchange management and financial transactions
3 *.\nThe organization consists of the Monetary Policy Committee, which reviews and resolves key matters concerning monetary credit poli
cy and the operation of the Bank of Korea, the president who serves as the chairman of the Monetary Policy Committee, and the Bank of
Korea's audit
```

4. Toy Playground #2

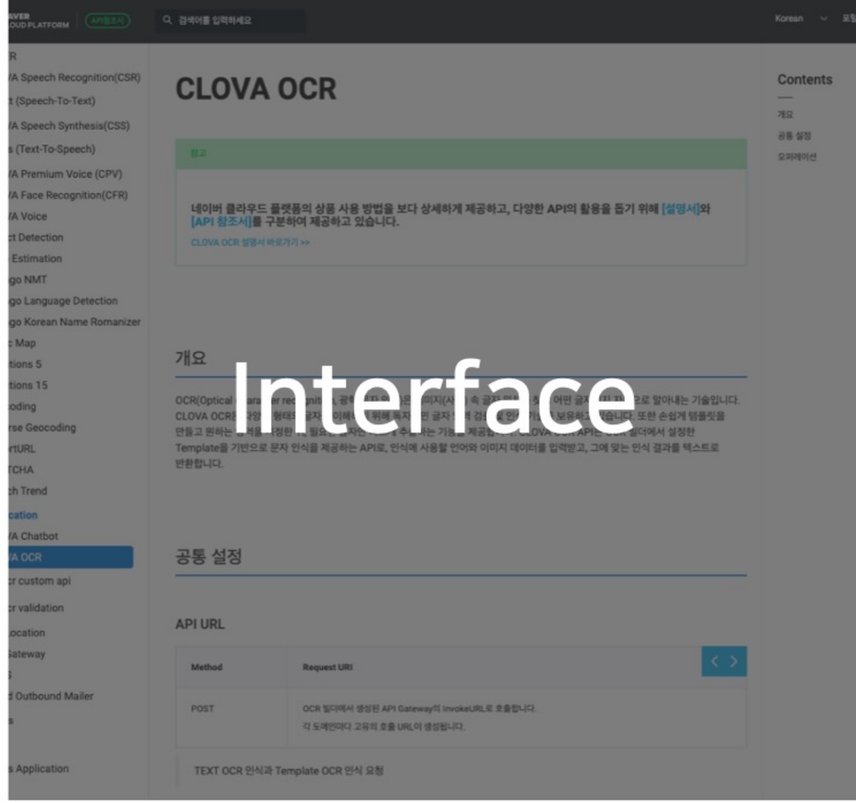
CFR(Clova Face Recognition) API를 활용한 My Gallery 서비스 구현하기



API Docs



NAVER APIs

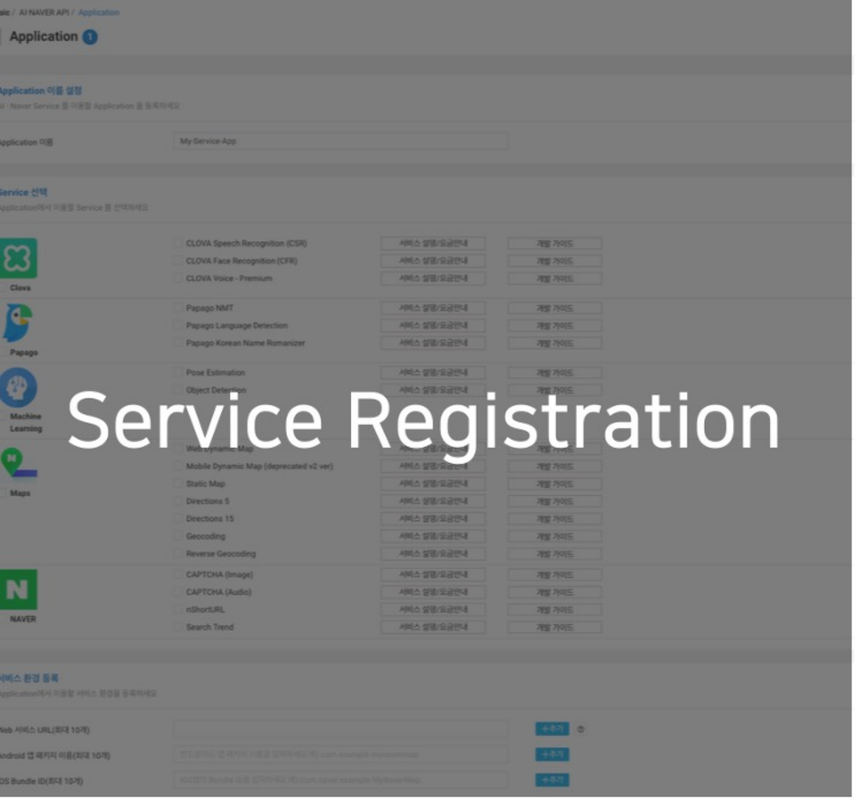


Interface

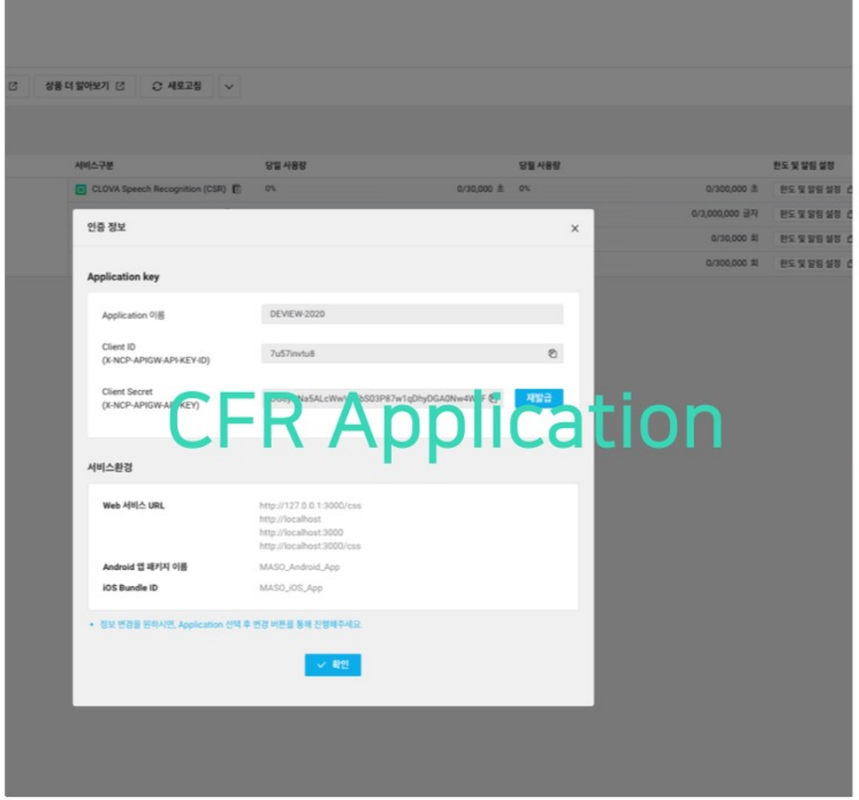
NAVER Cloud Platform API References

NAVER Cloud Platform NAVER AI & API Service 문서
AI & API Request/Response 적용 및 참고 문서

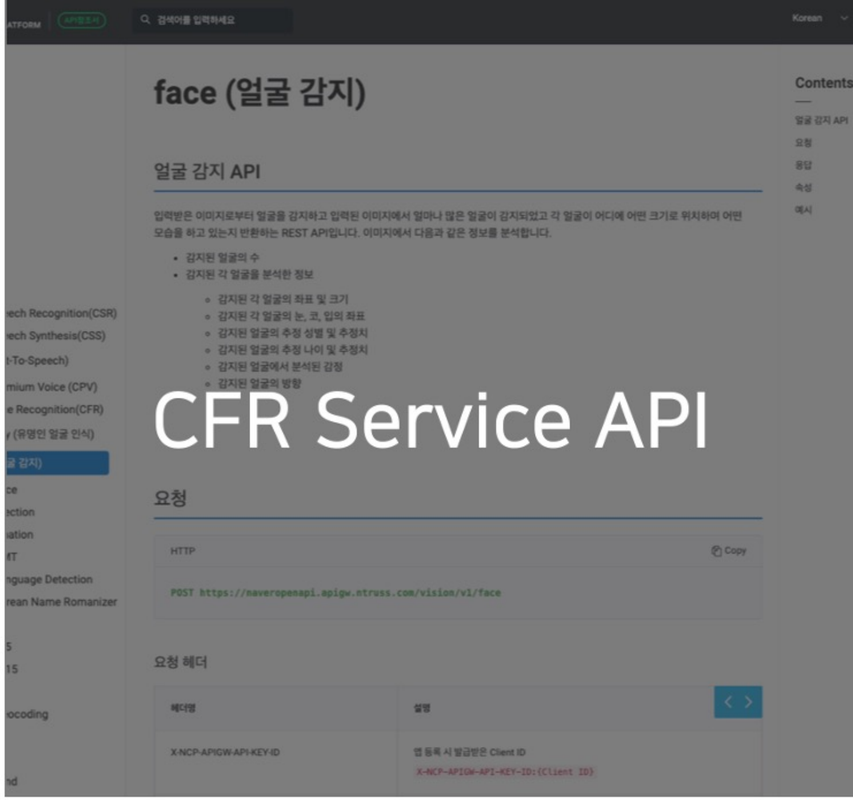
DEVIEW 2021



Service Registration



CFR Application



CFR Service API

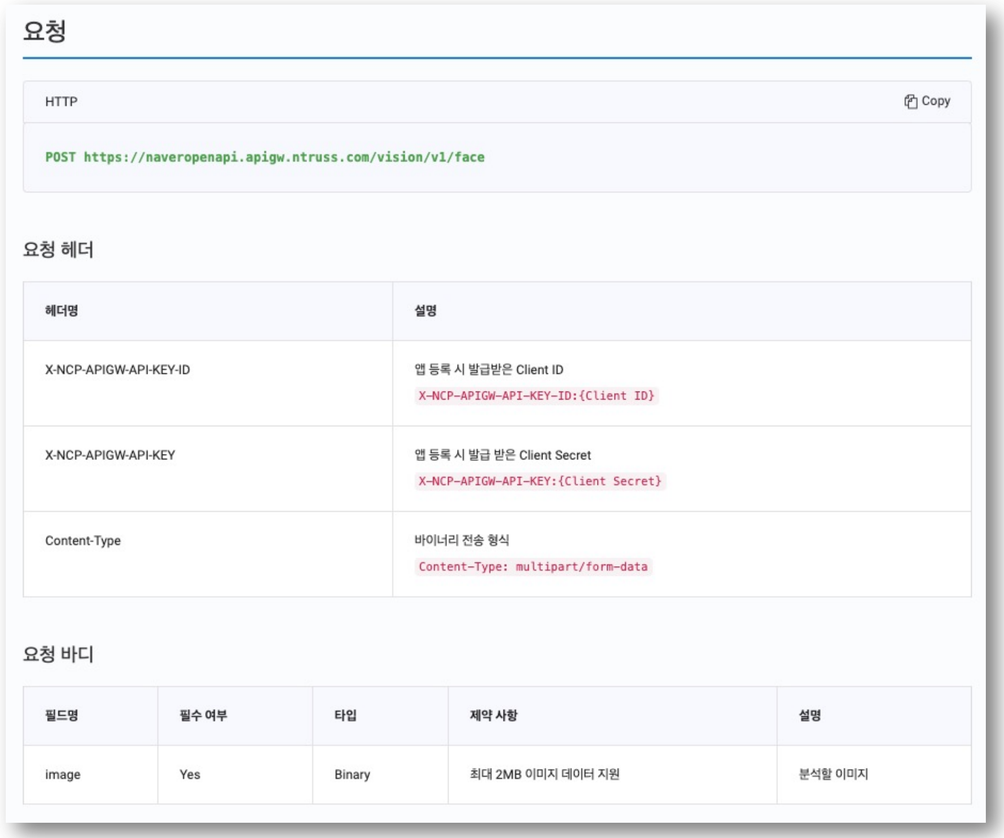
NAVER Cloud Platform CFR

- 얼굴과 관련된 다양한 정보를 제공하는 얼굴 인식 API
- 유명인 얼굴 인식 및 정확한 얼굴 감지 기능
- 머신 러닝(Machine Learning)을 사용하여 지속적인 학습이 가능한 서비스

CFR API Specification

Client ID / Secret Key / Context-Type / Image(Face)

CFR API Request



요청

HTTP Copy

```
POST https://naveropenapi.apigw.ntruss.com/vision/v1/face
```

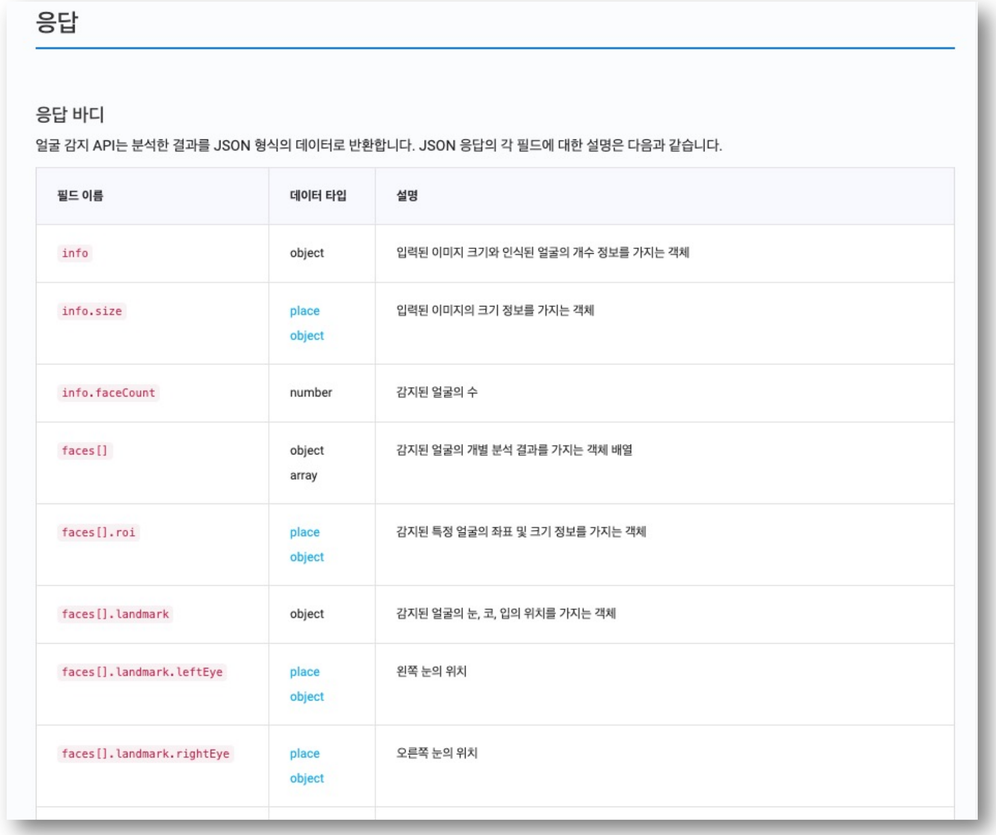
요청 헤더

헤더명	설명
X-NCP-APIGW-API-KEY-ID	앱 등록 시 발급받은 Client ID `X-NCP-APIGW-API-KEY-ID:{Client ID}`
X-NCP-APIGW-API-KEY	앱 등록 시 발급 받은 Client Secret `X-NCP-APIGW-API-KEY:{Client Secret}`
Content-Type	바이너리 전송 형식 `Content-Type: multipart/form-data`

요청 바디

필드명	필수 여부	타입	제약 사항	설명
image	Yes	Binary	최대 2MB 이미지 데이터 지원	분석할 이미지

CFR API Response

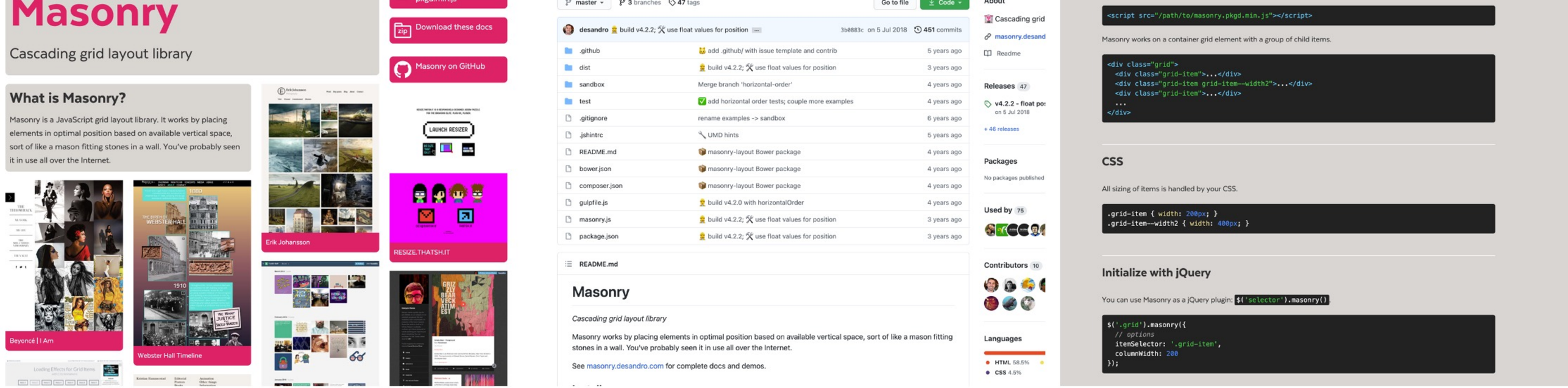


응답

응답 바디

얼굴 감지 API는 분석한 결과를 JSON 형식의 데이터로 반환합니다. JSON 응답의 각 필드에 대한 설명은 다음과 같습니다.

필드 이름	데이터 타입	설명
`info`	object	입력된 이미지 크기와 인식된 얼굴의 개수 정보를 가지는 객체
`info.size`	place object	입력된 이미지의 크기 정보를 가지는 객체
`info.faceCount`	number	감지된 얼굴의 수
`faces[]`	object array	감지된 얼굴의 개별 분석 결과를 가지는 객체 배열
`faces[].roi`	place object	감지된 특정 얼굴의 좌표 및 크기 정보를 가지는 객체
`faces[].landmark`	object	감지된 얼굴의 눈, 코, 입의 위치를 가지는 객체
`faces[].landmark.leftEye`	place object	왼쪽 눈의 위치
`faces[].landmark.rightEye`	place object	오른쪽 눈의 위치



출처 : https://masonry.desandro.com

Masonry – Javascript Image Gallery

- Masonry is a JavaScript grid layout library
- Placing elements optimal position based on available vertical space
- Cascading grid layout library, sort of like a mason fitting stones in a wall

Implementation of My Gallery Service

Implementation imgProcessor.js (1/4)

- NPM Module 설치
- CFR API

```
1  var express = require('express');
2  var app = express();
3
4  var fs = require('fs');
5  var fs = require('fs-extra')
6
7  var client_id = '{YOUR_NMT_CLIENT_ID}';
8  var client_secret = '{YOUR_NMT_SECRET_KEY}';
9
10 // Images Directory
11 var TOTAL_IMAGE_DIR = "/Users/naver/CloudToyService/MyGalleryService/public/images/";
12 var FACE_IMAGE_DIR = "/Users/naver/CloudToyService/MyGalleryService/public/face_images/";
13 var EXFACE_IMAGE_DIR = "/Users/naver/CloudToyService/MyGalleryService/public/exface_images/";
14
15 var filesInDir = [];
16
17 app.get('/faceCognito/:imageFile', function (req, res) {
18
19     var request = require('request');
20
21     // NAVER Cloud Platform Clova CFR API
22     //var api_url = 'https://naveropenapi.apigw.ntruss.com/vision/v1/celebrity'; // 유명인 인식
23     var api_url = 'https://naveropenapi.apigw.ntruss.com/vision/v1/face'; // 얼굴 감지
24
25     // Clova CFR API Information
26     var _formData = {
27         image:'image',
28         // Source File Name
29         image: fs.createReadStream(TOTAL_IMAGE_DIR + req.params.imageFile)
30     };
31
32     // Clova CFR API Options
33     var options = {
34         url: api_url,
35         formData: _formData,
36         headers: {
37             'X-NCP-APIGW-API-KEY-ID':client_id,
38             'X-NCP-APIGW-API-KEY': client_secret,
39             'Content-Type': 'multipart/form-data'
40         }
41     };
42
```

① Client_ID, Secret_Key

② Image Directories

③ CFR API URL

④ CFR API Options

Implementation imgProcessor.js (2/4)

- Extract Face Image

```
43      request.post(options, function (error, response, body) {
44
45          var tranlatedJSON = JSON.parse(body);
46
47          console.log('');
48          console.log('::: Processing FileName : ' + req.params.imageFile);
49          console.log('::: Face Cognition Count: ' + tranlatedJSON.info.faceCount);
50
51          // 얼굴 인식 여부에 따라 사진을 분류
52          if(tranlatedJSON.info.faceCount > 0) {
53              console.log('::: 얼굴 포함 사진입니다.');
54              console.log('::: 앨범 디렉터리로 이동합니다.');
55
56              fs.move(TOTAL_IMAGE_DIR + req.params.imageFile, FACE_IMAGE_DIR + req.params.imageFile, function (err) {
57                  if (err) return console.error(err)
58                  console.log("success!")
59              })
60          } else {
61              console.log('::: 얼굴 미 포함 사진입니다.');
62              console.log('::: 앨범 제외 디렉터리로 이동합니다.');
63
64              fs.move(TOTAL_IMAGE_DIR + req.params.imageFile, EXFACE_IMAGE_DIR + req.params.imageFile, function (err) {
65                  if (err) return console.error(err)
66                  console.log("success!")
67              })
68          }
69
70          console.log('');
71
72      });
73
74      res.end();
75
76  });
77
```

① Categories Image

② Extract Face Images

③ Extract Non-Face Images

Implementation imgProcessor.js (3/4)

- Extract Face Image

```
78  app.get('/MyGallery', function (req, res) {
79
80      console.log('::: MyGallery is called');
81
82      var files = fs.readdirSync(FACE_IMAGE_DIR);
83      var ext_files = fs.readdirSync(EXFACE_IMAGE_DIR);
84
85      var filesInDir = files;
86      var extFilesInDir = ext_files;
87
88      console.log('::: Face FilesInDir.length : ' +filesInDir.length);
89      console.log('::: Ext Face FilesInDir.length : ' +extFilesInDir.length);
90
91      var fileName, extFileName;
92      var tempDiv = '';
93      var extTempDiv = '';
94
95      for ( var i = 0; i < filesInDir.length-1; i++) {
96          fileName = filesInDir[i];
97
98          if(fileName == '.DS_Store') continue;
99
100         tempDiv += ' <div class="item"><img src="./face_images/'+fileName+'" width="110" height="110"/></div>'
101     }
102
103     for ( var i = 0; i < extFilesInDir.length-1; i++) {
104         extFileName = extFilesInDir[i];
105
106         if(extFileName == '.DS_Store') continue;
107
108         extTempDiv += ' <div class="item"><img src="./exface_images/'+extFileName+'" width="110" height="110"/></div>'
109     }
110
```

① Read Image Directory

② Create HTML Tag Face Image

③ Create HTML Tag Non-Face Image

Implementation imgProcessor.js (4/4)

- Extract Face Image

```
                    '</style>' +
                    '<script src="https://unpkg.com/masonry-layout@4/dist/masonry.pkgd.min.js"></script>' +
                '</head>' +
                '<body>' +
                '<p><b><font color="1E3269" size="5">[ My Face Gallery ]</font></b></p>'+
                '<p><b><font color="red">[ Contained Face Images ]</font></b></p>'+

                '<div class="parent" position:relative;>' +
                '<div id="container" position:absolute;>' +
                tempDiv +
                '</div>' +

                    '<script>' +
                    'var container = document.querySelector( \'#container\' );' +
                    'var msnry = new Masonry( container, {' +
                        '// options' +
                        'columnWidth: 110,' +
                        'itemSelector: \'.item\',' +
                    '} );' +
                    '</script>' +

                    '</b><br><br></b><br><br></b><br><br></b><br><br><p><b><font color="blue">[ Not Contained Face Images ]

                    '<div id="extcontainer" position:absolute;>' +
                    extTempDiv +
                    '</div>' +

                        '<script>' +
                        'var extcontainer = document.querySelector( \'#extcontainer\' );' +
                        'var msnry = new Masonry( extcontainer, {' +
                            '// options' +
                            'columnWidth: 110,' +
                            'itemSelector: \'.item\',' +
                        '} );' +
                        '</script>' +
                        '</div>' +
                '</body>' +
                '</html>';

    res.write(htmlStr);
    res.end();
});

app.listen(3000, function () {
  console.log('::: My Image Gallery App listening on port 3000!');
  app.use(express.static('public'));
});
```

1 Print Face Images

2 Print Non-Face Images

3 Image Gallery with Masonry

4 Start Node Server

mainApp.js (1/1)

- Extract Face Image

```
1  // Filesystem Library
2  var fs = require('fs');
3
4  // Execute OS Command in Node JS
5  var exec = require('child_process').exec,
6      child;
7
8  // Total Image Directory
9  var TOTAL_IMAGE_DIR = "/Users/naver/CloudToyService/MyGalleryService/public/images/";
10
11 // Read Directory with existed Album Images
12 var files = fs.readdirSync(TOTAL_IMAGE_DIR);
13 var filesInDir = files;
14
15 var fileName;
16
17 for ( var i = 0; i < filesInDir.length-1; i++) {
18
19     if(filesInDir[i] == '.DS_Store') continue;
20
21     fileName = filesInDir[i];
22     console.log('====================================');
23     console.log('::: Source FileName : ' + fileName);
24     console.log('====================================');
25
26     // Execute Clova CFR URL
27     child = exec("curl http://localhost:3000/faceCognito/"+fileName, function (error, stdout, stderr) {
28         if (error !== null) {
29             console.log('exec error: ' + error);
30         }
31     });
32
33     child = exec("sleep 2", function (error, stdout, stderr) {
34         if (error !== null) {
35             console.log('exec error: ' + error);
36         }
37     });
38 }
```

① OS Command 실행

② imgProcessor.js 호출

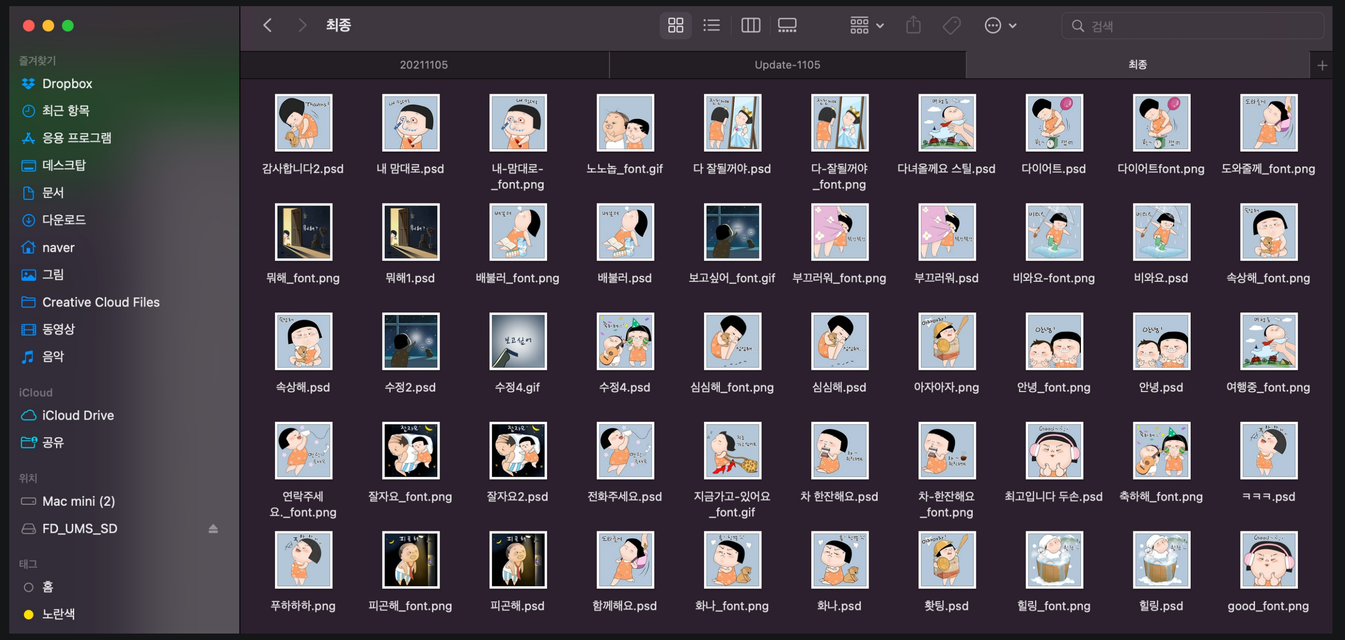
③ Sleep OS 명령어 실행

Execution Result of My Gallery Service

Execution Result (1/5)

Source Images

```
~/CloudToyService/MyGalleryService/p/images git master !19 ?6 ❯ ll
total 4256
-rw-rw-r--@ 1 naver  staff   109K  4  1 14:07 1.jpg
-rw-rw-r--@ 1 naver  staff    76K  4  1 14:07 10.jpg
-rw-rw-r--@ 1 naver  staff    39K  4  1 14:07 122.jpg
-rw-rw-r--@ 1 naver  staff    37K  4  1 14:07 2.jpg
-rw-rw-r--@ 1 naver  staff   161K  4  1 14:07 3.jpg
-rw-rw-r--@ 1 naver  staff    96K  4  1 14:07 4.jpg
-rw-rw-r--@ 1 naver  staff    38K  4  1 14:07 5.jpg
-rw-rw-r--@ 1 naver  staff   150K  4  1 14:07 7.jpg
-rw-rw-r--@ 1 naver  staff    64K  4  1 14:07 76.jpg
-rw-rw-r--@ 1 naver  staff    40K  4  1 14:07 78.jpg
-rw-rw-r--@ 1 naver  staff   113K  4  1 14:07 8.jpg
-rw-rw-r--@ 1 naver  staff    26K  4  1 14:07 9.jpg
-rw-rw-r--@ 1 naver  staff    21K  4  1 14:07 9cy77122_1.jpg
-rw-rw-r--@ 1 naver  staff   365K  4  1 14:07 BED21.jpg
-rw-rw-r--@ 1 naver  staff    52K  4  1 14:07 sea_1.jpg
-rw-rw-r--@ 1 naver  staff   289K  4  1 14:07 sea_2.jpg
-rw-rw-r--@ 1 naver  staff    88K  4  1 14:07 sea_3.jpg
-rw-rw-r--@ 1 naver  staff    40K  4  1 14:07 sea_4.jpg
-rw-rw-r--@ 1 naver  staff   278K  4  1 14:07 sea_5.jpg
```



Execution Result (2/5)

Execution imgProcessor.js

```
~/CloudToyService/MyGalleryService git master !20 ?6 ❯ node imgProcessor.js                    10.19.0
::: My Image Gallery App listening on port 3000!




::: Processing FileName : 2.jpg
::: Face Cognition Count: 1
::: 얼굴 포함 사진입니다.
::: 앨범 디렉터리로 이동합니다.


::: Processing FileName : 122.jpg
::: Face Cognition Count: 1
::: 얼굴 포함 사진입니다.
::: 앨범 디렉터리로 이동합니다.


::: Processing FileName : sea_3.jpg
::: Face Cognition Count: 0
::: 얼굴 미 포함 사진입니다.
::: 앨범 제외 디렉터리로 이동합니다.


::: Processing FileName : 9cy77122_1.jpg
::: Face Cognition Count: 1
::: 얼굴 포함 사진입니다.
::: 앨범 디렉터리로 이동합니다.


::: Processing FileName : sea_1.jpg
::: Face Cognition Count: 0
::: 얼굴 미 포함 사진입니다.
::: 앨범 제외 디렉터리로 이동합니다.
```

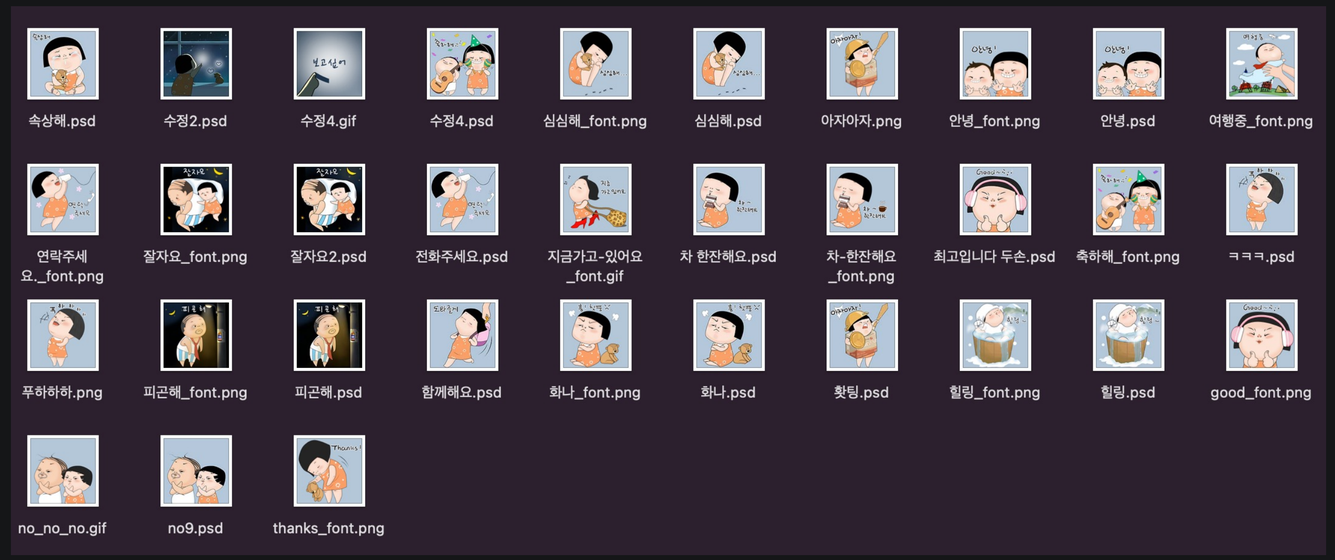
Execution Result (3/5)

Execution mainApp.js

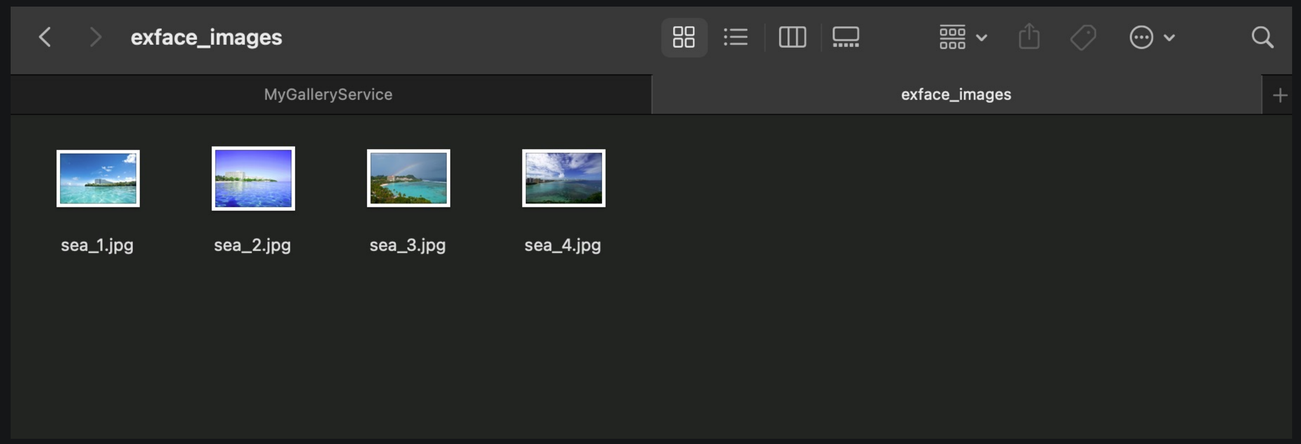
```
~/CloudToyService/MyGalleryService git master !19 ?6 ❯ node mainApp.js
========================================
::: Source FileName : .DS_Store
========================================
========================================
::: Source FileName : 1.jpg
========================================
========================================
::: Source FileName : 10.jpg
========================================
========================================
::: Source FileName : 122.jpg
========================================
========================================
::: Source FileName : 2.jpg
========================================
========================================
::: Source FileName : 3.jpg
========================================
========================================
::: Source FileName : 4.jpg
========================================
```

Execution Result (4/5)

1 Face-Included Images 디렉터리

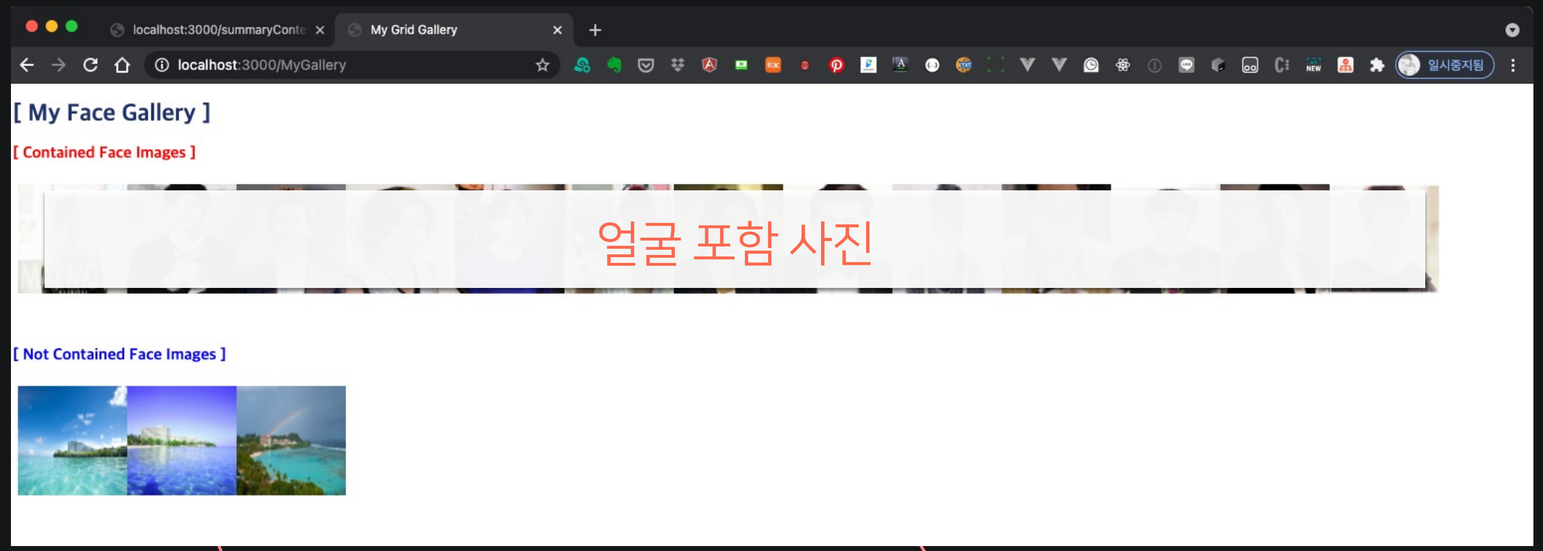


2 Not Face-Included Images 디렉터리



Execution Result (4/5)

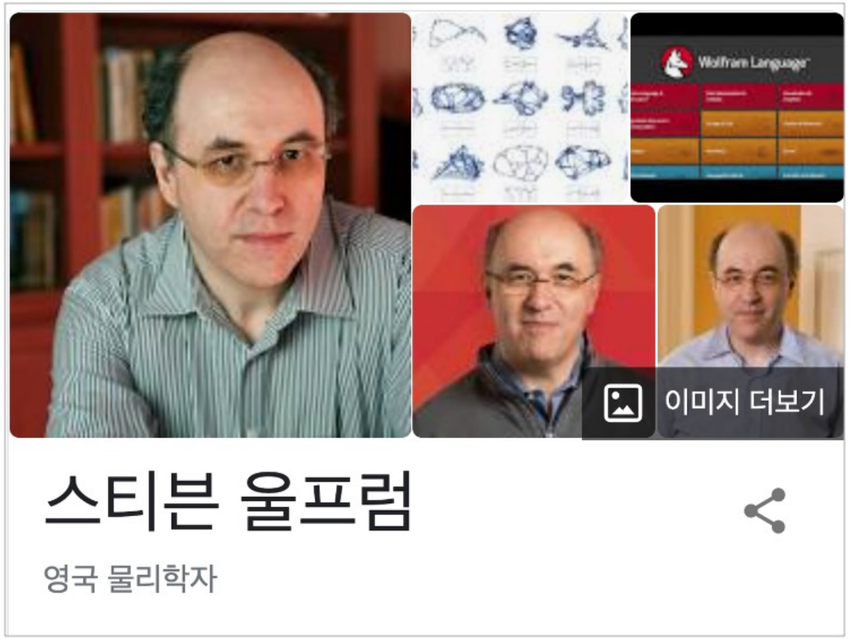
My Face Gallery 실행



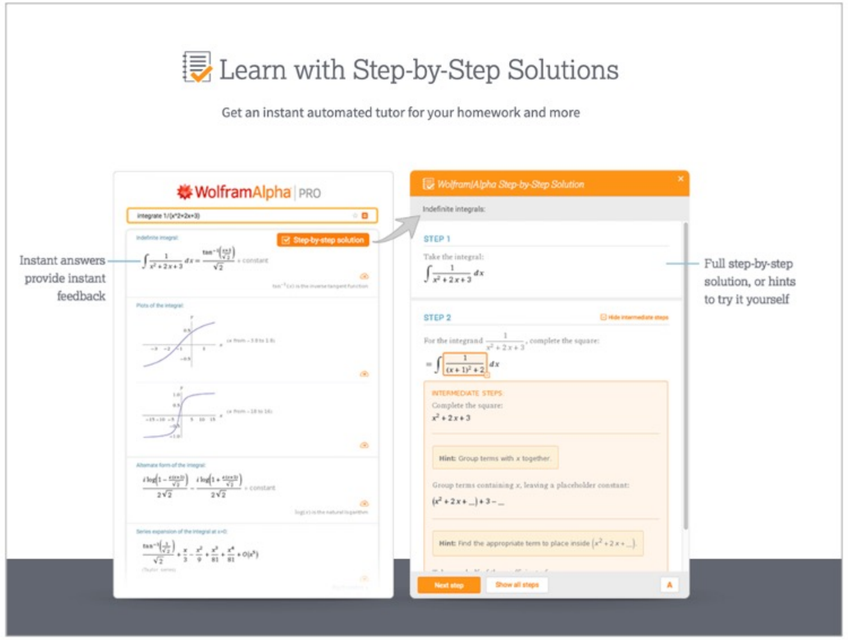
얼굴 미포함 사진

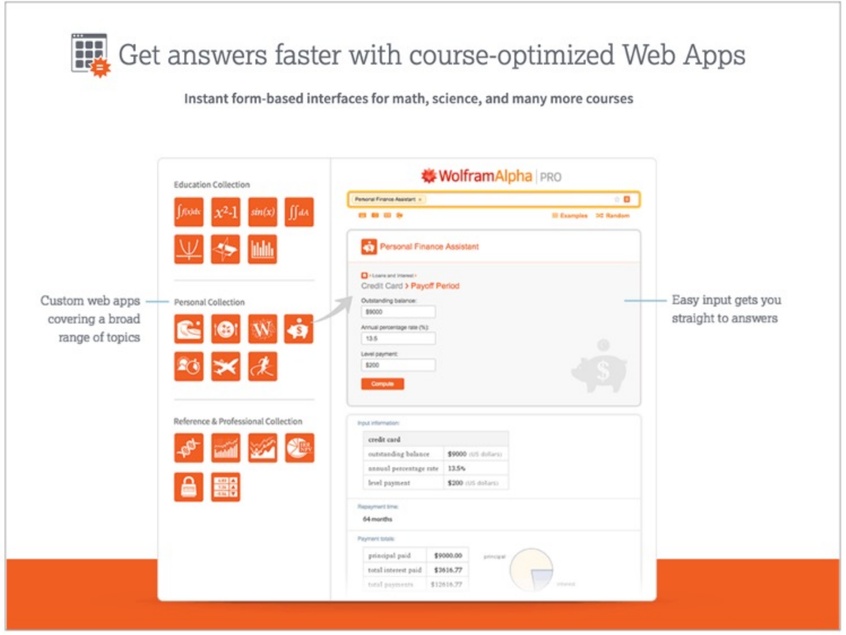
얼굴 포함 사진

5. Toy Playground #3
Universal Knowledge Playground 서비스 구현하기



출처 : https://bit.ly/3BRxmHm



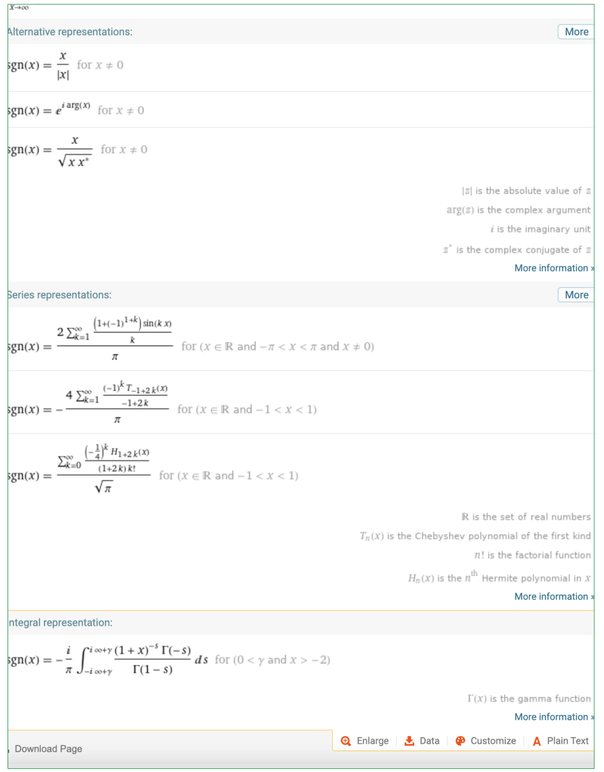


출처 : https://www.wolframalpha.com

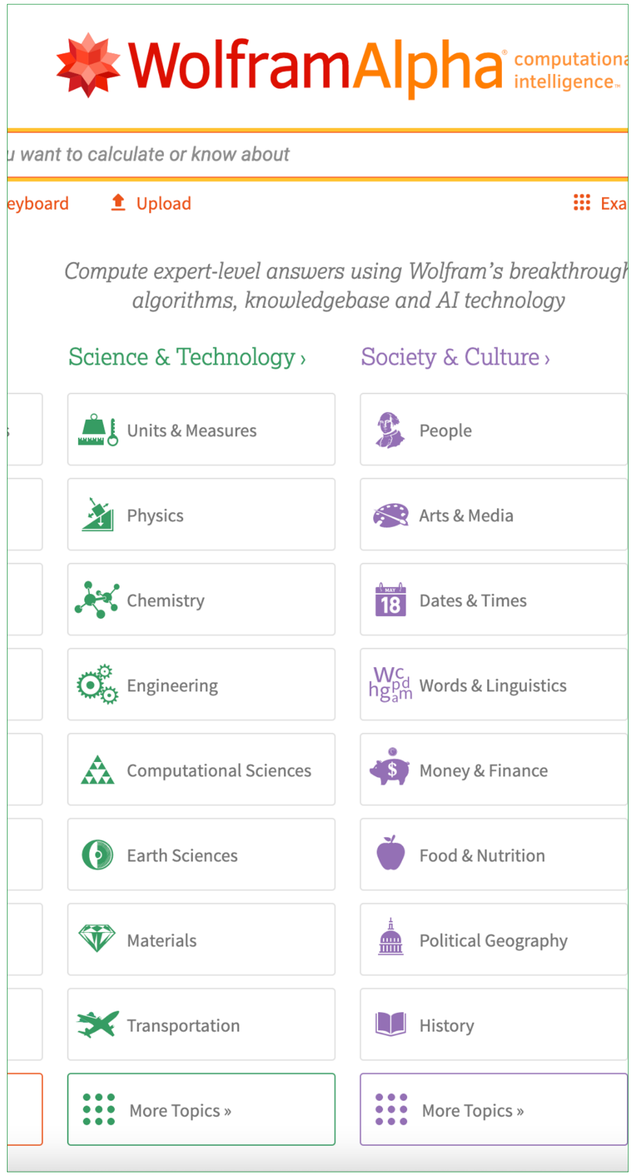
WolframAlpha with Computational Inelligence

- 과학/공학용 프로그램 Mathematica 개발자 Stephen Wolfram이 개발한 검색엔진
- 슈퍼컴퓨터를 통한 인공지능을 통해 웹 상의 지식을 재구성하여 사용자에게 제공
- 수학적 연산을 직접 수행하고, Simulating 그래픽 결과를 제공
- Mathematica 700만 Lines Code 프로그래밍 & 약 1만 개의 CPU 사용 및 구동

Wolfram Search Engine & API



출처 : https://www.wolframalpha.com



Wolfram|Alpha is a unique engine for computing answers and providing knowledge

LINGUISTIC ANALYSIS
New kinds of algorithms for 1,000+ domains

+

CURATED DATA
10+ trillion pieces of data from primary sources with continuous updating

+

DYNAMIC COMPUTATION
50,000+ types of algorithms and equations

→

COMPUTED PRESENTATION
5,000+ types of visual and tabular output

WolframAlpha API Getting Stared

DEVIEW 2021

1.Authorization

Wolfram Site Sign up & Login

2. APP ID

Obtaining an App ID with Application Information

3. Sample Query

http://api.wolframalpha.com/v2/query?appid=DEMO

4. Formatting Input

All URLs used to make queries must be URL encoded (ex.LaTeX or MathML)

5. Adding Parameters

You can add URL-encoded parameters to customize output

6. Parameter Reference

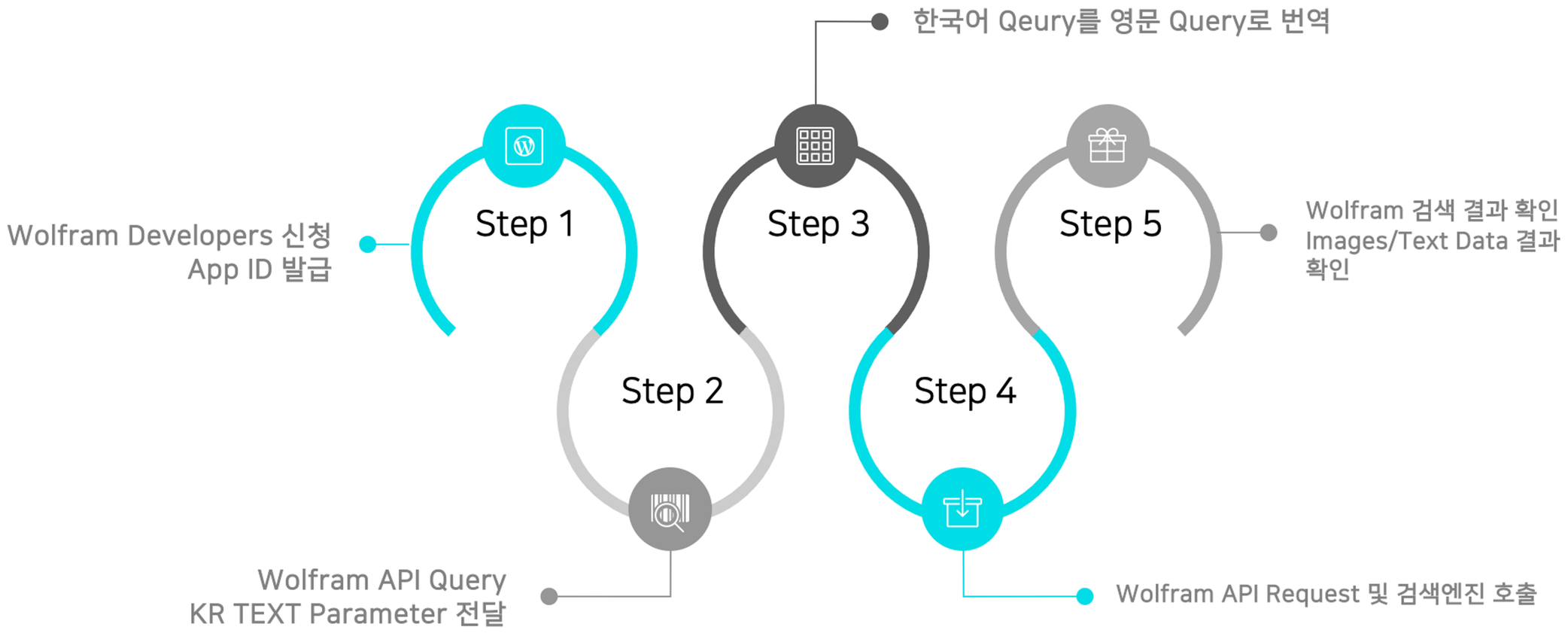
Basic Parameters, Pod Selection, Location, Size, Timeouts/Async, Miscellaneous

7. List of XML Result Elements

hierarchy of XML results from the Full Results API

Implemetation Step
Universal Knowledge Playground

DEVIEW 2021



Implementation of Universal Knowledge Playground

Implementation (1/6)

- NVM 설치
- NPM Module 설치

```
{
  "name": "myapp",
  "version": "1.0.0",
  "description": "my test app",
  "main": "index.js",
  ▷ 디버그
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.16.2",
    "crypto-js": "^3.1.9-1",
    "dotenv": "^4.0.0",
    "express": "^4.16.2",
    "libxmljs": "^0.19.7",
    "wolfram": "^0.3.4",
    "wolfram-alpha": "^0.6.0",
    "wolfram-alpha-api": "https://products.wolframalpha.com/api/libraries/javascript/wolfram-alpha-api-1.0.0-rc.1.tgz"
  }
}
```

① package.json

② Required NPM Package

```
"libxmljs": "^0.19.7",
"wolfram": "^0.3.4",
"wolfram-alpha": "^0.6.0",
"wolfram-alpha-api": "https://products.wolframalpha.com/api/libraries/javascript/wolfram-alpha-api-1.0.0-rc.1.tgz"
```

③ Wolfram & libxmljs NPM Package

Implementation (2/6)

- NPM Module 선언
- Test Wolfram API

```
1  // import 'express' module
2  var express = require('express');
3
4  var app = express();
5
6  /* API Authorization start [ --- */
7  //var client_id = '{YOUR_CLIENT_ID}';
8  var client_id = '7f      w';
9
10 //var client_secret = '{YOUR_SECRET_KEY}';
11 var client_secret = 'yyGS                    tIgzam9am11UUK';
12 /* API Authorization end --- ] */
13
14 // Test Korean Query
15 var query = "현재 금 시세는 얼마인가요?";
16
17 /* WolframAplha Simple Test start [ --- */
18 app.get('/simpleWolf', function (req, res) {
19
20   var resultStr;
21   // WolframAlpha Call API
22   var wolfram = require('wolfram').createClient("RUK6JE-56KU9Q7EX4");
23   //var wolfram = require('wolfram-alpha').createClient("APIKEY-HERE", opts);
24
25   // Test Translated English Query
26   wolfram.query("What is the current price of gold", function(err, result) {
27       if(err) throw err
28       console.error("result", result);
29       resultStr = result;
30
31       //res.writeHead(200, {'Content-Type': 'text/json;charset=utf-8'});
32       res.write(JSON.stringify(resultStr));
33       res.end();
34   });
35 });
36 /* WolframAplha Simple Test end --- ] */
```

1 Required NPM Package

2 Simple Wolfram API Test Routing

3 Request of Wolfram API Query & Parameters

Implementation (3/6)

- Papago NMTAPI 호출 & 결과

```
38  /* Papago NMT Translate Simple Test start [ --- */
39  app.get('/papagoNMTTrans', function (req, res) {
40    var api_url = 'https://naveropenapi.apigw.ntruss.com/nmt/v1/translation';
41    var request = require('request');
42    // Setting Options
43    var options = {
44        url: api_url,
45        form: {
46          'source':'ko',
47          'target':'en',
48          'text':query
49        },
50        headers: {
51          'X-NCP-APIGW-API-KEY-ID':client_id,
52          'X-NCP-APIGW-API-KEY': client_secret,
53          'Content-Type': 'application/json'
54        }
55    };
56
57    request.post(options, function (error, response, body) {
58      if (!error && response.statusCode == 200) {
59        res.writeHead(200, {'Content-Type': 'text/json;charset=utf-8'});
60        res.end(body);
61      } else {
62        res.status(response.statusCode).end();
63        console.log('error = ' + response.statusCode);
64      }
65    });
66  });
67  /* Papago NMT Translate Simple Test end --- ] */
```

① Simple PapagoNMT API Test Routing

② PapagoNMT API Headers

③ Request/Response of PapagoNMT Translation

Implementation (4/6)

- Execute PapagoNMT
- Execute Wolfram

```
/* WolframAplha Service start [ --- */
app.get('/getMyKnowledgeInfo/:query', function (req, res) {

  res.writeHead(200, { 'Content-Type': 'text/html' });

  // papagoNMT Setting variables
  var papagoNMTStr;
  var papagoNMTJSON;

  // papagoNMT node.js file load
  var papagoQuery = require('./papagoNMTTrans.js');
  // papagoNMT Translation + wolframAlpha API Call
  var mypapagoQuery = papagoQuery.papagoNMTTranslation(req.params.query,function (response) {

    papagoNMTStr = JSON.stringify(response);
    papagoNMTJSON = JSON.parse(papagoNMTStr).message.result.translatedText;

    console.log('::: Original Text : ' + req.params.query);
    console.log('::: Traslated Text : ' + papagoNMTJSON);

    // wolframAlpha Setting variables
    var wolframStr;
    var wolframJSON;
    var htmlStr = '';

    // wolframAlpha node.js file load
    var wfQuery = require('./wolfQuery.js');
    var mywfQuery = wfQuery.queryWolframAlpha(papagoNMTJSON,function (response) {

      console.log('::: wolframAlpha Length : ' + response.length);
      console.log('wolframAlpha Str : ' + JSON.stringify(response[0].title));

      for ( var i=0 ; i < response.length; i++) {
        htmlStr += response[i].title + '<br>' + '<img src=\'' + response[i].subpods[0].image + '\'><br><br><br>';
      }

      res.write(htmlStr);
      res.end();

    });
  });
});
```

① Import Module papagoNMTTrans.js

② Import ModulewolfQuery.js

Implementation (5/6)

- PapagoNMTTran.js

```
// 네이버 Papago NMT API 예제
var express = require('express');
var app = express();

// PapagoNMT API Cliend_Id & Secret Key
/* API Authorization start [ --- */
//var client_id = '{YOUR_CLIENT_ID}';
var client_id = '7fgu5rxj2w';

//var client_secret = '{YOUR_SECRET_KEY}';
var client_secret = 'yyGSIHhrG02ve9mcidRkuf6mlatIgzam9am11UUK';
/* API Authorization end --- ] */

var query;
var resultStr;

module.exports =  {

    papagoNMTTranslation: function(queryStr,callback) {
        console.log('::: papagoNMTTranslation() is called.');

        query = queryStr;

        var api_url = 'https://naveropenapi.apigw.ntruss.com/nmt/v1/translation';
        var request = require('request');

        // PapagoNMT HTTP Options
        var options = {
            url: api_url,
            form: {'source':'ko', 'target':'en', 'text':textStr},
            headers: {
              'X-NCP-APIGW-API-KEY-ID':client_id,
              'X-NCP-APIGW-API-KEY': client_secret,
              'Content-Type': 'application/json'
            }
        };

        request.post(options, function (error, response, body) {
            resultStr = JSON.parse(body);
            // retrun callback function
            return callback(resultStr) ;
        });
    }
}
```

① Authorization NCP PapagoNMT API

② PapagoNMT API Function

③ PapagoNMT API Request Header

④ Request PapagoNMT API

Implementation (6/6)

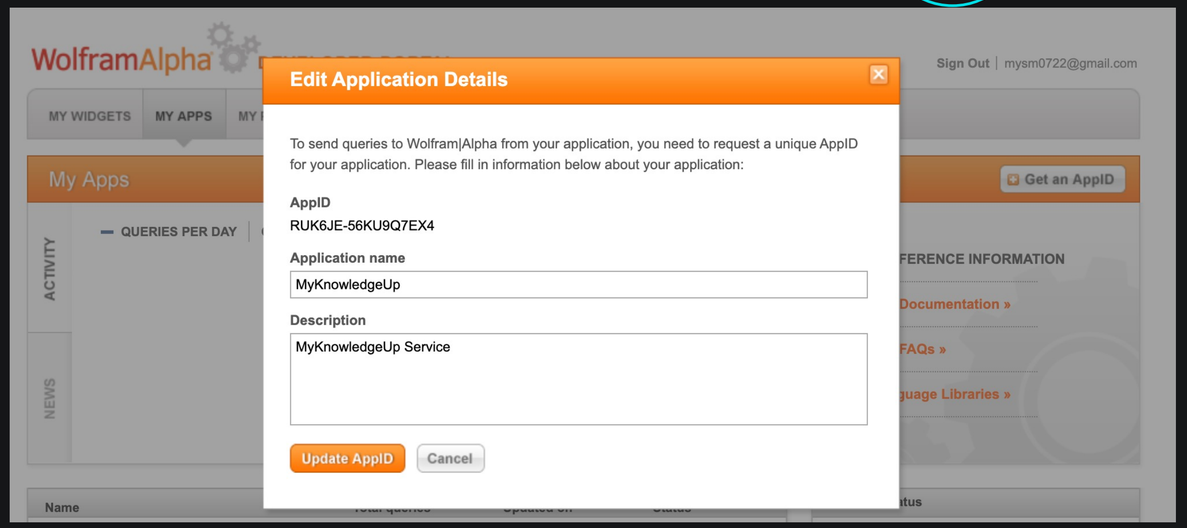
- wolfQuery.js

```
var express = require('express');
var app = express();

// WolframAlpha API Call
var wolfram = require('wolfram').createClient("RUK6JE-56KU9Q7EX4");

module.exports = {
    queryWolframAlpha: function(queryStr,callback) {
        console.log('::: queryWolframAlpha() is called.');

        var resultStr;
        return wolfram.query(queryStr, function(err, result) {

            if(err) throw err
            console.error("result", result);
            resultStr = result;

            // retrun callback function
            return callback(resultStr) ;
        });
    }
}
```

1 Authorization Wolfram API

2 Request & Response Wolfram API Query

3 Key/Token Wolfram API

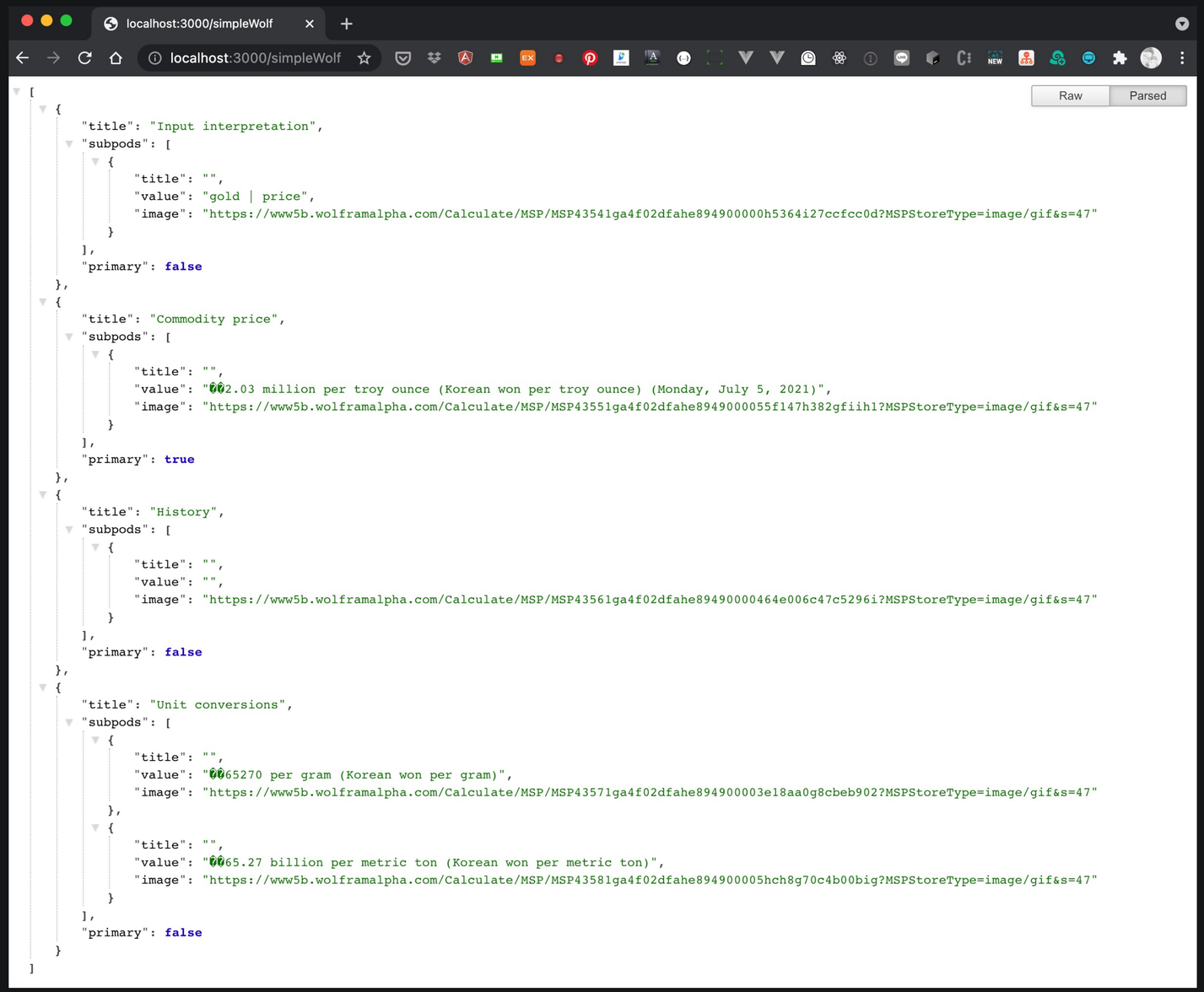


Execution Result of Universal Knowledge Playground

Execution Result (1/2)

- /simpleWolf 실행 결과

http://localhost:3000/simpleWolf



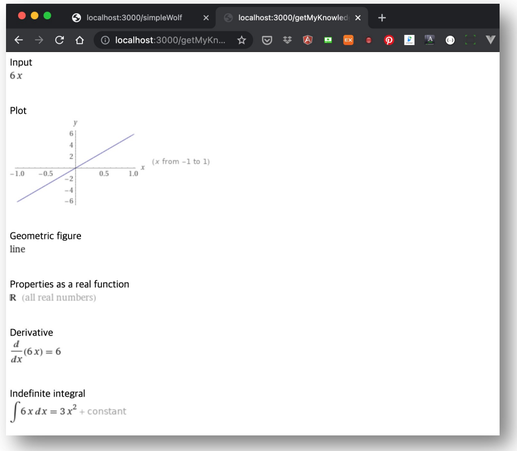

```
[
  {
    "title": "Input interpretation",
    "subpods": [
      {
        "title": "",
        "value": "gold | price",
        "image": "https://www5b.wolframalpha.com/Calculate/MSP/MSP43541ga4f02dfahe894900000h5364i27ccfcc0d?MSPStoreType=image/gif&s=47"
      }
    ],
    "primary": false
  },
  {
    "title": "Commodity price",
    "subpods": [
      {
        "title": "",
        "value": "₩2.03 million per troy ounce (Korean won per troy ounce) (Monday, July 5, 2021)",
        "image": "https://www5b.wolframalpha.com/Calculate/MSP/MSP43551ga4f02dfahe8949000055f147h382gfiih1?MSPStoreType=image/gif&s=47"
      }
    ],
    "primary": true
  },
  {
    "title": "History",
    "subpods": [
      {
        "title": "",
        "value": "",
        "image": "https://www5b.wolframalpha.com/Calculate/MSP/MSP43561ga4f02dfahe89490000464e006c47c5296i?MSPStoreType=image/gif&s=47"
      }
    ],
    "primary": false
  },
  {
    "title": "Unit conversions",
    "subpods": [
      {
        "title": "",
        "value": "₩65270 per gram (Korean won per gram)",
        "image": "https://www5b.wolframalpha.com/Calculate/MSP/MSP43571ga4f02dfahe894900003e18aa0g8cbeb902?MSPStoreType=image/gif&s=47"
      },
      {
        "title": "",
        "value": "₩65.27 billion per metric ton (Korean won per metric ton)",
        "image": "https://www5b.wolframalpha.com/Calculate/MSP/MSP43581ga4f02dfahe894900005hch8g70c4b00big?MSPStoreType=image/gif&s=47"
      }
    ],
    "primary": false
  }
]
```

Universal Knowledge Playground Examples

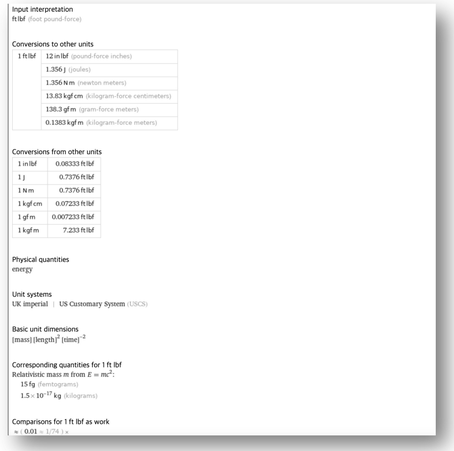
DEVIEW 2021

현재 금 시세는 얼마인가요?

six(x)를 계산해 주세요.

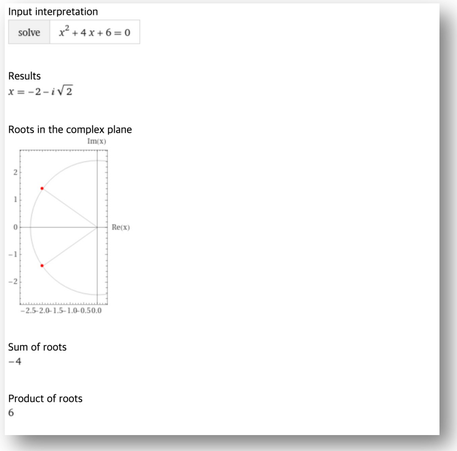


Units

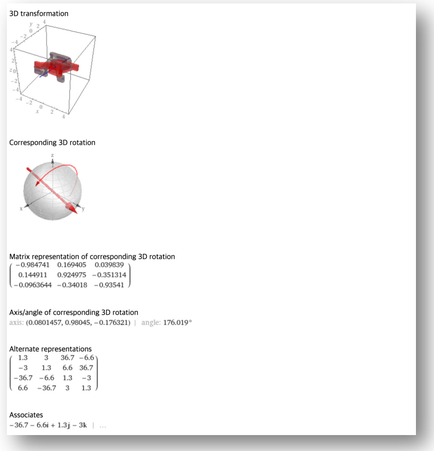


● ● ● ● ●

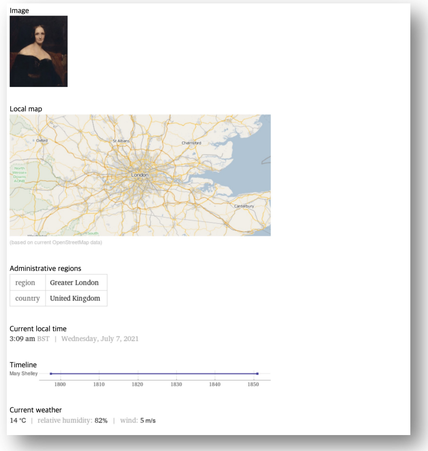
Mathmatics



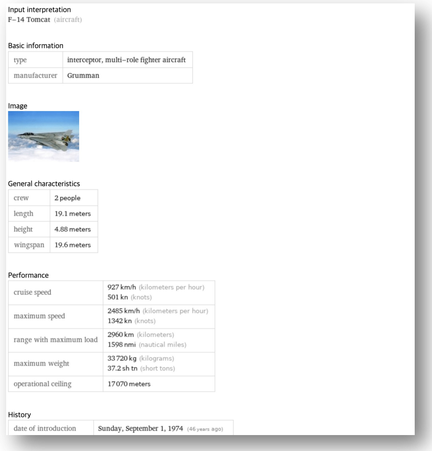
ML



People



Machines



● ● ● ● ●

6. Toy Playground #6

CFR(Clova Face Recognition) API를 활용한 Face Image Analyzer Service 구현하기

Implemetation Step
Face Image Analyzer Service

DEVIEW 2021

Step1

CFR API Image Call

CFR API Request Headers
분석 필요한 Image 전송

Step2

Face Recognition

각 Image별 얼굴 인식 수행
얼굴 포함/비포함 이미지 분석

Implemetation Step
Face Image Analyzer Service

DEVIEW 2021

Step3

Face Position

Face Position Index
Face Information

Step4

Analysis Face Images

Face Analytic Information
CFR API with JSON

Step5

Face Analysis Service

Print JSON to Browser
사진 내 얼굴 이미지 정보 분석

Implementation of
Face Image Analyzer Service

Implementation (1/4)

- NPM Module 설치
- CFR API Token

```
1  // import 'express' module
2  var express = require('express');
3  var app = express();
4
5  // import 'axios' module
6  const axios = require('axios');
7
8  // API Authorization (Client ID & Secret)
9  // var client_id = '{YOUR_CLIENT_ID}';
10 var client_id = '      tu8';
11
12 // var client_secret = '{YOUR_SECRET_KEY}';
13 var client_secret = 'UG8                  P87w1qDhyDGA0Nw4WLF';
14
15 var fs = require('fs');
16
17 // Image for Destination Image
18 var arvgStr = process.argv[2];
19
20 // for Browser Print
21 var htmlStr = '<p><b><font color="orange">[ 변환 대상 이미지 ]</font></b></p>'+
22               '/Users/naver/CloudToyService/MyFaceAnalyzer/public/images/' + arvgStr + '<br><br>' +
23               '<p><b><font color="orange">[ 원본 이미지 ]</font></b></p>'+
24               '<img src="images/' + arvgStr + '"><br><br>' +
25               '<p><b><font color="blue">[ 사진 기본 정보 ]</font></b></p>';
26
```

① Import Required Module

② CFR API Authorization

③ HTML Query String

Implementation (2/4)

- URL Routing
- CFR API

```
27  app.get('/faceAnalyzer', function (req, res) {
28
29    console.log('::: Image Parameter : ' + arvgStr);
30
31    var request = require('request');
32
33    // Clova Face Recognition API URL(얼굴 감지)
34    // var api_url = 'https://naveropenapi.apigw.ntruss.com/vision/v1/celebrity'; // 유명인 인식
35    var api_url = 'https://naveropenapi.apigw.ntruss.com/vision/v1/face'; // 얼굴 감지
36
37    // Image formData for NCP CFR API
38    var _formData = {
39      image:'image',
40      image: fs.createReadStream('/Users/naver/CloudToyService/MyFaceAnalyzer/public/images/'+arvgStr)
41      // FILE 이름
42    };
43
44    // Authorization Information
45    var config = {
46      headers: {
47        'X-NCP-APIGW-API-KEY-ID':client_id,
48        'X-NCP-APIGW-API-KEY': client_secret,
49        'Content-Type': 'multipart/form-data'
50      }
51    }
```

① /faceAnalyzer Routing

② CFR API Request URL

③ CFR API Request Headers

Implementation (3/4)

- CFR API Request
- Face Image Analysis

```
53  // Photo Infomation Variables
54  var faceCounts = 0;
55  var photoSize = 0;
56
57  request({
58    method:'post',
59    url:api_url,
60    formData: _formData,
61    headers: {
62      'X-NCP-APIGW-API-KEY-ID':client_id,
63      'X-NCP-APIGW-API-KEY': client_secret,
64      'Content-Type': 'multipart/form-data'
65    },
66    json: true,
67  }, function (error, response, body) {
68
69    res.writeHead(200, { 'Content-Type': 'text/html;charset=UTF-8' });
70
71    // Photo Infomation Objects
72    faceCounts = body.info.faceCount;
73    photoSize = body.info.size;
74
75    // Basic Information for Inclued Face Image
76    htmlStr += '<b>사진 크기 : '+ body.info.size.width + 'x' + body.info.size.height + '</b><br/>';
77    htmlStr += '<b>얼굴 포함 인원 : '+ faceCounts + '</b><br/><br/>';
78
79    // Detail Information for Image
80    htmlStr += '<p><b><font color="blue">[ 사진 분석 상세  정보 ]</font></b></p>';
```

① CFR API Request Object

② Caculate Face Count

③ HTML Information String

Implementation (4/4)

- CFR API Response
- Face Information

```
75      // Basic Information for Inclued Face Image
76      htmlStr += '<b>사진 크기 : '+ body.info.size.width + 'x' + body.info.size.height + '</b><br/>';
77      htmlStr += '<b>얼굴 포함 인원 : '+ faceCounts + '</b><br/><br/>';
78
79      // Detail Information for Image
80      htmlStr += '<p><b><font color="blue">[ 사진 분석 상세  정보 ]</font></b></p>';
81
82      for (var i=0; i < faceCounts; i++) {
83        //htmlStr += '-----------------------------------<br>';
84        htmlStr += '<b>[ Person #' + i + ' 정보 '+' ]</b><br/><br/>';
85        htmlStr += '성별 : ' + body.faces[i].gender.value +'<br/>';
86        htmlStr += '성별 신뢰도 : ' + body.faces[i].gender.confidence +'<br/><br/>';
87        htmlStr += '나이 : ' + body.faces[i].age.value +'<br/>';
88        htmlStr += '나이 신뢰도 : ' + body.faces[i].age.confidence +'<br/><br/>';
89        htmlStr += '감정 : ' + body.faces[i].emotion.value +'<br/>';
90        htmlStr += '감정 신뢰도 : ' + body.faces[i].emotion.confidence +'<br/><br/>';
91        htmlStr += '얼굴 방향 : ' + body.faces[i].pose.value +'<br/>';
92        htmlStr += '얼굴 방향 신뢰도 : ' + body.faces[i].pose.confidence +'<br/><br/><br/>';
93      }
94
95      // String to HTML
96      res.write(htmlStr);
97      res.end();
98
99    });
100
101 });
102
103 app.use(express.static('public'));
104
105 app.listen(3000, function () {
106   console.log('::: My FaceAnalyzer app listening on port 3000!');
107   app.use(express.static('public'));
108 });
```

1 Face Information Analysis

2 Start Node Server

Execution Result of
Face Image Analyzer Service

Execution Result (1/3)

- Face Analyzer 실행 방법

App 실행 방법 및 설명

Application 실행 방법

Node Express 서버를 실행하여, Web 서버를 구동합니다.

```
$ node app.js {Image Path & File Name}
```

Application Overview

아래와 같이 파라미터로 전달된 사진을 분석하여 사용자에게 정보를 제공합니다.

- 사진 이름 : 전달된 사진 Path + FileName
- 사진 기본 정보 : 사진 크기(가로 X 세로), 사진에 포함된 얼굴 수(faceCount)
- 얼굴 상세 정보 (위의 각 사람 별 정보)
  - 성별 (남, 여, 아이 등)
  - 나이 (나이 범위 제공, ex. 34~38)
  - 감정 (화남, 즐거움, 무표정 등)
- 얼굴 상세 정보는 값과 신뢰도("1"에 가까울 수록 일치)를 함께 제공함

Application Execution

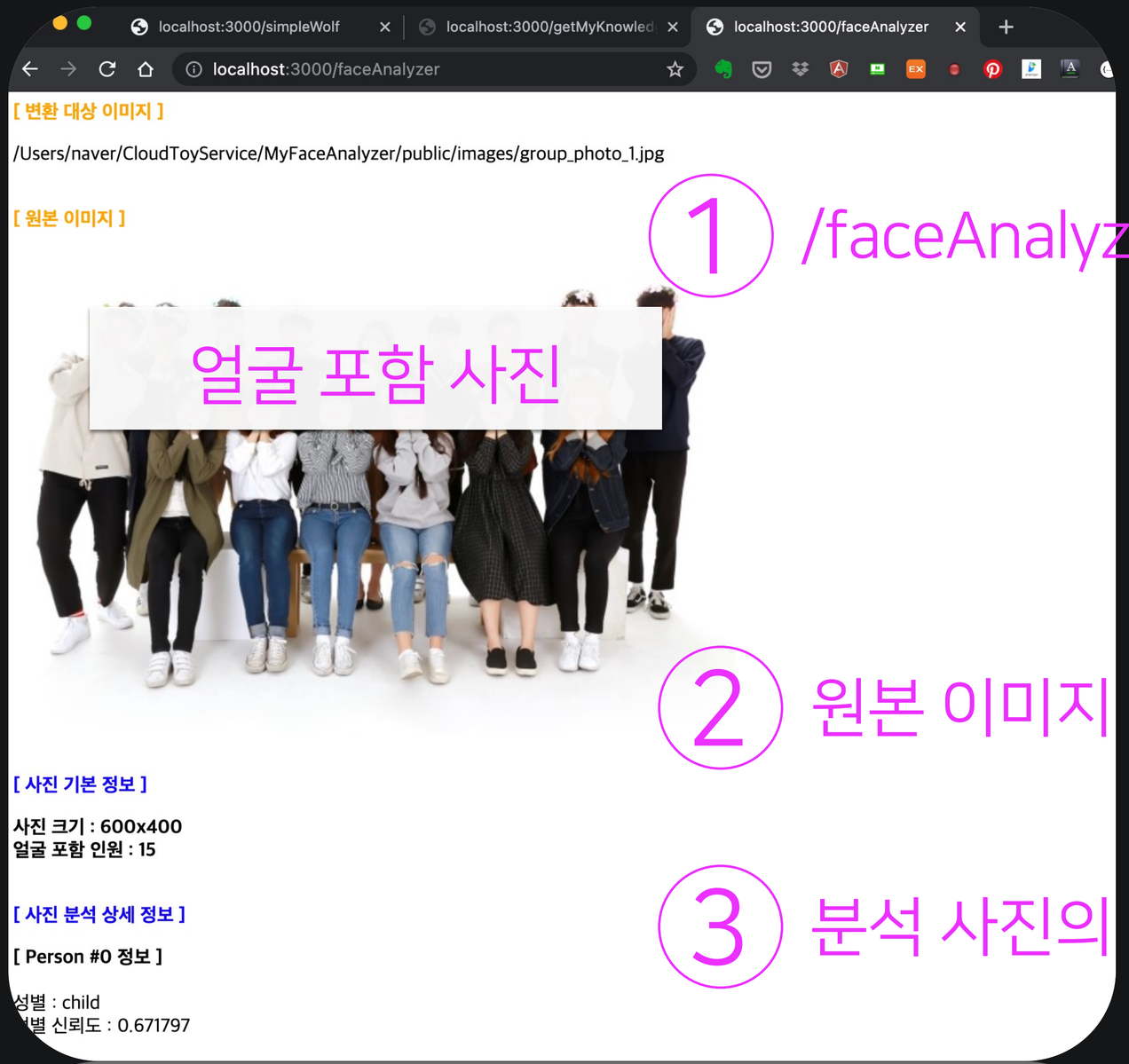
아래와 같이 브라우저의 주소창에 입력하면, 해당 Router가 실행됩니다.

```
$ http://{server_name}:{server_port}/faceAnalyzer
```

Execution Result (2/3)

- Face Analyzer 실행 결과

http://localhost:3000/faceAnalyzer



① /faceAnalyzer Route 실행 결과

② 원본 이미지

③ 분석 사진의 기본 정보

Execution Result (3/3)

- Face Analyzer 실행 결과

상세 분석 정보 : 성별, 감정, 얼굴 방향 분석 정보 제공

[사진 기본 정보]

사진 크기 : 600x400
얼굴 포함 인원 : 15

[사진 분석 상세 정보]

[Person #0 정보]

성별 : child
성별 신뢰도 : 0.671797

나이 : 38~42
나이 신뢰도 : 0.0208397

감정 : smile
감정 신뢰도 : 0.893313

얼굴 방향 : frontal_face
얼굴 방향 신뢰도 : 0.880512

[Person #1 정보]

성별 : female
성별 신뢰도 : 0.909999

나이 : 40~44
나이 신뢰도 : 0.0299906

감정 : smile
감정 신뢰도 : 0.716931

얼굴 방향 : false_face
얼굴 방향 신뢰도 : 0.755758

[Person #2 정보]

성별 : female
성별 신뢰도 : 0.999897

나이 : 22~26
나이 신뢰도 : 0.880144

감정 : smile
감정 신뢰도 : 0.992987

얼굴 방향 : frontal_face
얼굴 방향 신뢰도 : 0.652764

[Person #3 정보]

성별 : female
성별 신뢰도 : 0.630627

나이 : 23~27
나이 신뢰도 : 0.945816

감정 : neutral
감정 신뢰도 : 0.939889

얼굴 방향 : false_face
얼굴 방향 신뢰도 : 0.784295

[Person #4 정보]

성별 : female
성별 신뢰도 : 0.999969

●●●●●

[Person #12 정보]

성별 : male
성별 신뢰도 : 0.555792

나이 : 37~41
나이 신뢰도 : 0.0261736

감정 : neutral
감정 신뢰도 : 0.639275

얼굴 방향 : false_face
얼굴 방향 신뢰도 : 0.791699

[Person #13 정보]

성별 : male
성별 신뢰도 : 0.949783

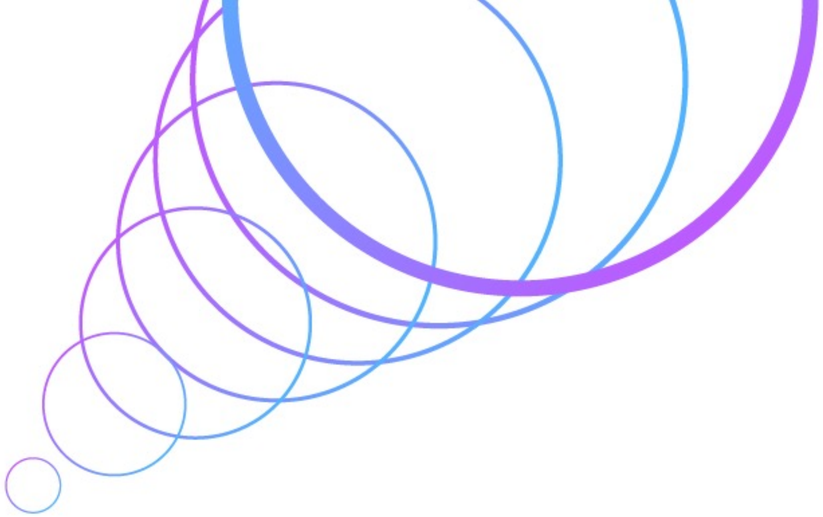
나이 : 24~28
나이 신뢰도 : 0.732963

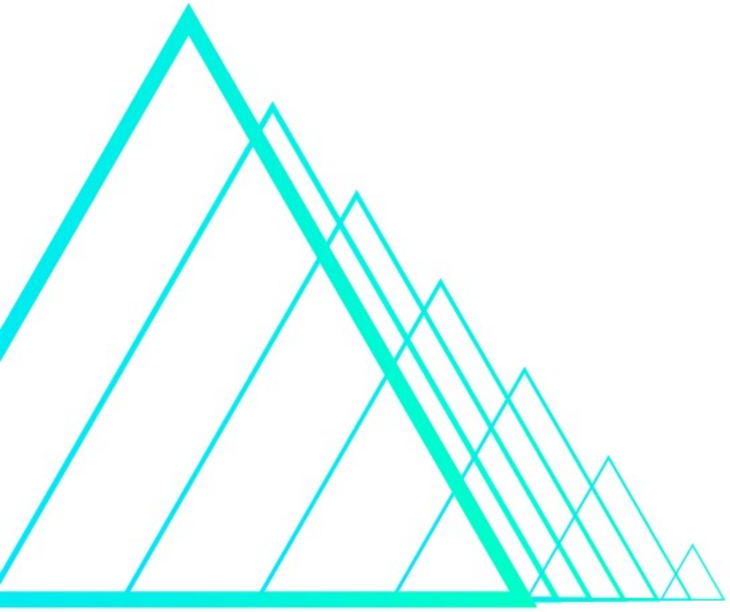
감정 : smile
감정 신뢰도 : 0.997658

얼굴 방향 : frontal_face
얼굴 방향 신뢰도 : 0.930083

[Person #14 정보]

성별 : child
성별 신뢰도 : 0.521873





Q & A

Thank You