SSL

2022년 1학기

SEMINAR

서준혁

CONTENTS

01

주제 개요

01. 주제 소개
02. 선정 이유
03. 기대 효과

01 주제 개요

01. 주제 소개

02. 선정 이유

03. 기대 효과

금오 소식 꾸러미 웹 서비스

금오 소식 꾸러미?

본교에서는 교내 학생들을 위한 여러 혜택이나
행사에 대한 정보들을 공지해줌.

학생들은 이러한 정보를 일일히 찾아보며 챙겨야하는
꽤 번거로운 상황이 반복되며
때로는 좋은 기회를 놓칠 수 있음.

따라서 이러한 문제를 해결하기 위해
학교 및 학과의 공지들을 수집하고 정리하여
한 눈에 볼 수 있는 웹 서비스를 개발하고자 함.

01 주제 개요

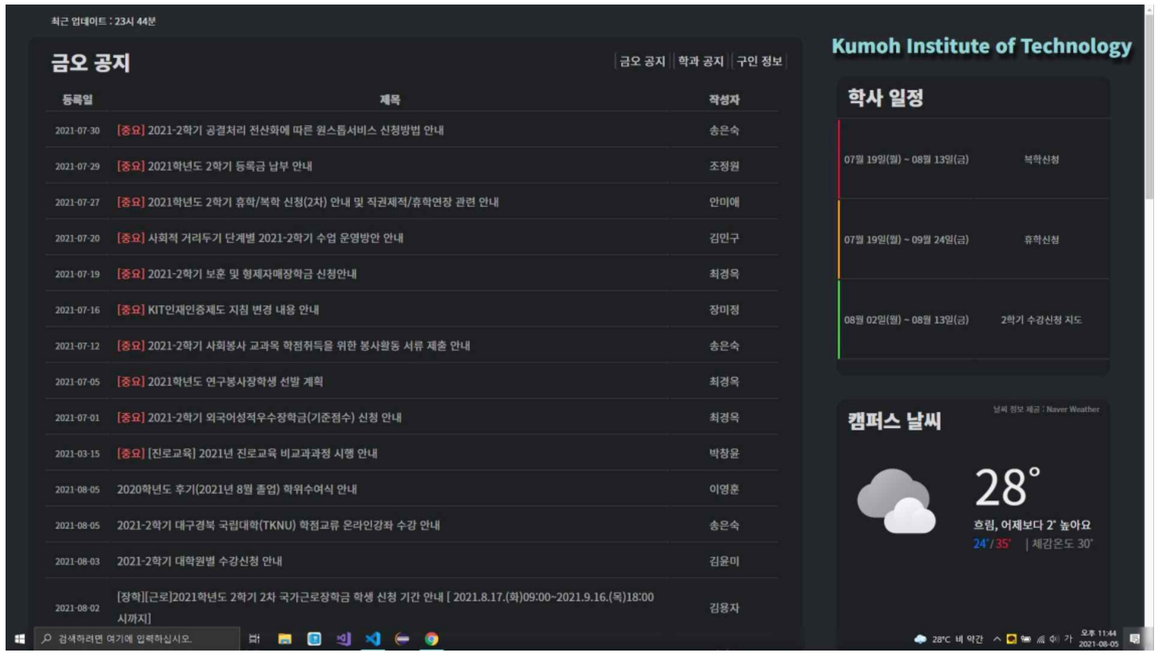
금오 소식 꾸러미 웹 서비스

해당 주제를 선정한 이유?

저학년 시절 해당 주제를 가지고
프로젝트를 진행했던 경험이 있음.

Python으로 정적 크롤러를 구현하고
파일 입출력을 이용한 데이터로
로컬 웹을 구성하였기에 한계가 존재했음.

그동안의 경험과 공부한 내용을 갖고
본격적으로 상용화 될 수 있도록
업그레이드하고자 함.



Beautiful Soup

01 주제 개요

01. 주제 소개

02. 선정 이유

03. 기대 효과

금오 소식 꾸러미 웹 서비스

기대 효과?

학생들에게 눈에 띌 정도로 큰 도움은 되지 않겠지만,
웹 프론트엔드와 백엔드를 함께 공부하고 있어
스스로에겐 뜻깊은 경험이 되리라 생각함.

모든 학과에 대한 소식통을 제공할 수 있도록
계획중이기 때문에, 그래도 이용자가 꽤 생겨
학교에서 제공하는 기회를 학생들이 많이
챙겨갈 수 있으리라 기대함.

02

프로젝트 설계

01. 구성 요소
02. 필요 기술
03. 동작 원리

02 프로젝트 설계

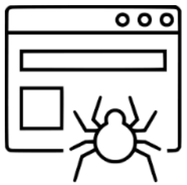
01. 구성 요소

02. 필요 기술

03. 동작 원리

금오 소식 꾸러미 웹 서비스

구성 요소는?

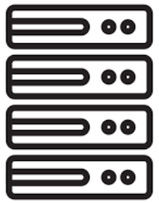


학교 공지 데이터를 얻기 위한 크롤러

학교 웹 데이터베이스에 직접 접근하여
공지 데이터를 얻는 것이 가장 이상적이지만,
불가능하기 때문에 크롤링을 하기 위한 크롤러 필요.

데이터를 저장하기 위한 데이터베이스

크롤링을 통해 얻은 데이터를 저장하고,
웹페이지에서 띄우는 등의 동작을 위한
데이터베이스 필요.



웹 호스팅을 위한 서버

나만 볼 수 있는 단순한 로컬 웹이 아닌
모두가 사용할 수 있는 웹 서비스를 구축하는 것이
목표이기 때문에 서버 필요.

서비스 이용을 위한 웹 페이지

당연하게도 사람들이 접속하여
서비스를 이용하려면 웹 페이지를 구성해야함.

02 프로젝트 설계

01. 구성 요소

02. 필요 기술

03. 동작 원리

금오 소식 꾸러미 웹 서비스

필요한 기술은?

Node.js

서버 개발의 효율적인 생산성,
손쉬운 DB 연동 등의 장점을 가지고 있는
Node.js를 메인 스택으로 사용

mongoDB, mongoose

NoSQL 기반이기에 퍼포먼스가 빠르고,
확장 및 수정이 간편한 mongoDB를 사용.
mongoDB를 Node.js에서 컨트롤할 수 있는
mongoose 모듈 또한 사용.

Axios, cheerio 모듈

Node.js에서 웹 크롤러를 제작하기 위한 모듈들로,
Axios를 통해 내가 원하는 URL로 요청을 보내 해당 URL의
HTML 문서를 파싱하고, cheerio 모듈의 selector (선택자)를 통해
내가 원하는 컴포넌트들만 얻을 수 있음.

02 프로젝트 설계

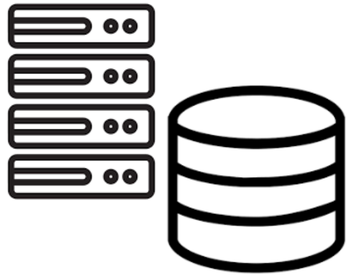
01. 구성 요소

02. 필요 기술

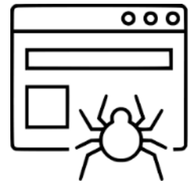
03. 동작 원리

금오 소식 꾸러미 웹 서비스

어떻게 동작하는가?



DB와 연동된 서버 동작



일정 주기마다 크롤러 동작

크롤링한 데이터 저장

해당 데이터 기반 웹 구성

접속한 사용자에게 서비스 제공

반복

03

세부 내용

01. 진행 상황
02. 소스 코드

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

현재 진행 상황은?

서버와 데이터베이스 연동 성공.

크롤러를 제작하여 학교 공지사항 게시판 글의 제목, 작성자, 날짜, [중요] 태그 상태 등의 정보를 크롤링하는 과정 성공.

크롤링한 데이터를 데이터베이스의 올바른 스키마에 알맞게 넣는 과정 성공.

```
PS C:\Users\ssam2\Desktop\kit_Notice> node app.js
Server Connected on 3000.
{
  _id: new ObjectId("6225f6219b60aa7244a850af"),
  title: '2022-1학기 공결신청 방법 및 코로나19 예방접종 백신공결 안내 (3.7-3.13)',
  author: '손우정',
  date: '2022-03-07',
  importance: true
}
{
  _id: new ObjectId("6225f6219b60aa7244a850af"),
  title: '2022학년도 1학기 등록금 추가납부 안내(추가2차)',
  author: '조정원',
  date: '2022-03-04',
  importance: true
}
{
  _id: new ObjectId("6225f6219b60aa7244a850af"),
  title: '2022-1학기 수강과목 변경,정정 신청 안내',
  author: '김민구',
  date: '2022-02-21',
  importance: true
}
```

```
PS C:\Users\ssam2\Desktop\kit_Notice> node app.js
Server Connected on 3000.
MongoDB Connected.
```

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

QUERY RESULTS **61-80 OF MANY**


```
_id: ObjectId("6225f8167a594a6a9c949cdc")
title: "[근로] 2022-1학기 학기 중 근로장학생 선발 알림"
author: "정혜련"
date: "2022-02-25"
importance: false
__v: 0
```

```
_id: ObjectId("6225f83188293d404f167cb8")
title: "2022-1학기 공결신청 방법 및 코로나19 예방접종 백신공결 안내 (3.7-3.13)"
author: "손우정"
date: "2022-03-07"
importance: true
__v: 0
```

```
_id: ObjectId("6225f83188293d404f167cba")
title: "2022-1학기 수강과목 변경,정정 신청 안내"
author: "김민구"
date: "2022-02-21"
importance: true
__v: 0
```

```
_id: ObjectId("6225f83188293d404f167cb9")
title: "2022학년도 1학기 등록금 추가납부 안내(추가2차)"
author: "조정원"
date: "2022-03-04"
importance: true
__v: 0
```

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

app.js

```
1  const express = require('express');
2  const crawler = require('./crawler');
3  const db = require('./db');
4  const app = express();
5
6  const port = 3000;
7
8  app.get('/', function(req, res){
9      res.send("Server Running");
10 });
11
12 app.listen(port, function(){
13     console.log(`Server Connected on ${port}.`);
14 });
```

app.js (Server Side)

서버 실행을 담당하는 스크립트인 app.js 이다.

소스 코드를 조금 더 간결하게하고 구현 난이도를 낮추기 위해 express 모듈을 사용해 서버를 열었다.

크롤러 및 데이터베이스 연동을 위해 각각의 스크립트가 있는 경로를 require 메소드를 이용해 불러왔다.

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

db.js

```
"use strict";

const mongoose = require('mongoose');
const databaseUrl = 'mongodb+srv://ssam2s:wnsgur1122@cluster0.vmq3j.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'

mongoose
.connect(databaseUrl, {useNewUrlParser: true, useUnifiedTopology: true})
.then(() => console.log('MongoDB Connected.'))
.catch((e) => console.log(e));

const dataSchema = mongoose.Schema({
    title: {
        type: String,
        maxlength: 100
    },
    author: {
        type: String,
        maxlength: 10
    },
    date: {
        type: String,
        maxlength: 20
    },
    importance: {
        type: Boolean
    },
})

const data = mongoose.model('data', dataSchema);
module.exports = {data};
```

db.js (Database)

데이터베이스 구현을 위한 스크립트인 db.js이다.

datebaseUrl을 통해 mongoDB에 연결하고 제목, 작성자, 날짜를 저장할 수 있도록 데이터 스키마를 작성해놓았다.

또한, app.js에서 사용할 수 있도록 말단에 module.exports를 통해 내보내기 코드도 작성했다.

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

crawler.js

```
1   const axios = require("axios");
2   const cheerio = require("cheerio");
3   const db = require('./db');
4
5   const getHTML = async() => {
6       try
7       {
8           return await axios.get("https://www.kumoh.ac.kr/ko/sub06_01_01_01.do");
9       }
10      catch (error)
11      {
12          console.error(error);
13      }
14  }
```

crawler.js (Crawler) - Axios

크롤러를 구현한 스크립트인 crawler.js 이다.

해당 스크립트는 조금 길어서 우선 Axios를 통한 URL 연결 부분을 캡쳐했다.
app.js가 실행되더라도 비동기식으로 동작하도록 async() 키워드를 사용하여 함수를 작성했으며
try - catch문을 통해 연결이 되지 않는다면 에러를 찍도록 작성했다.

03 세부 내용

01. 진행 상황

02. 소스 코드

금오 소식 꾸러미 웹 서비스

crawler.js (Crawler) - Crawling

crawler.js의 핵심 부분이다.

앞서 Axios를 통해 연결이 성공적으로 되었다면
해당 함수가 동작하도록 .then을 이용해 익명 함수로 작성하였다.

학교 웹 컴포넌트를 분석해 알맞는 데이터를 찾도록 작성했고 탭 문자나 개행 문자는 replace 함수를 통해 제거한 상태로 db에 저장되도록 구현했다.

정리한 데이터는 말단의 for문에서 각각을 db에 맞는 object로 가공해 저장될 수 있도록 했다.

```
16  getHTML()
17      .then(html => {
18
19
20          var titleList = [];
21          var authorList = [];
22          var dateList = [];
23          var importance = [];
24
25          const $ = cheerio.load(html.data);
26          const $titleList = $("span.title-wrapper");
27          const $authorList = $("td.writer");
28          const $dateList = $("td.date");
29
30          $titleList.each(function(i, elem){
31              titleList[i] = $(this).text().replace(/\t/g, '').replaceAll('\n','')
32
33              if (titleList[i].toString().startsWith('공지'))
34              {
35                  titleList[i] = $(this).text().replace(/\t/g, '').replaceAll('\n','').replace('공지', '')
36                  importance[i] = true;
37              }
38              else
39              {
40                  importance[i] = false;
41              }
42          });
43
44          $authorList.each(function(i, elem){
45              authorList[i] = $(this).text()
46          })
47
48          $dateList.each(function(i, elem){
49              dateList[i] = $(this).text()
50          })
51
52          for (var i = 0; i < $titleList.length; i++)
53          {
54              var _data = new db.data();
55
56              _data.title = titleList[i];
57              _data.author = authorList[i];
58              _data.date = dateList[i];
59              _data.importance = importance[i];
60
61              console.log(_data);
62              _data.save();
63          }
64      });
```

04

기타 사항

01. 고민 및 계획

04 기타 사항

01. 고민 및 계획

금오 소식 꾸러미 웹 서비스

앞으로...

해당 프로젝트를 시작한지 아직 3일이 채 되지 않아서 구현한 내용이 많진 않다.

그래도 핵심 문제인 크롤러는 응용만 하면 계속 사용이 가능할 것 같다.

이젠 HTML 문서와 CSS를 작성하고 DB에서 데이터를 로드해볼 계획이다.

한 가지 고민은, DB가 무한정 데이터를 저장할 수 없는 노릇이기 때문에

삽입된 데이터에 대한 기한이나 추후에 많아질 크롤링 데이터에 대한 오버헤드를

어떻게 처리할 것인지 조금 생각해봐야 할 것 같다.

감사합니다

SSL_서준혁