신발 시세 모니터링 서비스

전자공학과 20175214 나병희

목차

[신발 시세 모니터링 서비스 1](#_Toc105779017)

[전자공학과 20175214 나병희 1](#_Toc105779018)

[**I.** **서론** 2](#_Toc105779019)

[１. 프로젝트의 필요성 2](#_Toc105779020)

[２. 프로젝트의 목적 2](#_Toc105779021)

[**II.** **본론** 3](#_Toc105779022)

[１. 시스템 구조 3](#_Toc105779023)

[２. 소스코드 설명 4](#_Toc105779024)

[３. 실행화면 8](#_Toc105779025)

[**III.** **결론** 12](#_Toc105779026)



# **서론**

# 프로젝트의 필요성

요즘에는 평범함 보다는 자신의 개성을 옷차림으로 표현하는 사람들이 많다. 이에 따라 평범한 신발보다 한정판 신발들의 관심이 많아지면서 리셀 시장이 급격하게 커지면서 리셀 플랫폼 또한 생겨났다. 이러한 리셀 플랫폼에서 한정판 신발들의 가격을 확인할 수 있지만, 광고 등 쓸모없는 정보들도 많이 노출 되어있다. 이런 문제를 해결하기위해 웹 크롤링을 통해 원하는 신발의 가격 데이터만 가져와서 볼 수 있게 구현하였다.

# 프로젝트의 목적

간략하게 알고 싶은 신발의 가격만 볼 수 있게, 웹 크롤링을 이용해 리셀 플랫폼 사이트에서 가격 부분만 가져와서 DB로 저장을 하고 Web을 통해 확인할 수 있게 하는 것이 목적이고, 6개의 신발의 가격을 일정 시간마다 갱신해서 Web을 통해 볼 수 있게 구현하였다.

# **본론**

# 시스템 구조

그림 1은 신발시세현황의 시스템 구조를 보여준다. 본 시스템은 신발가격데이터, MQTT Broker, Node.js, Resource DB, Web UI로 구성된다. 신발가격 데이터는 리셀 플랫폼 사이트에서 웹 크롤링을 통해 20초 주기로 수집한 뒤, MQTT Broker 서버에 Publish 하는 역할을 수행한다. MQTT Broker는 웹 크롤링을 통해 Publish 된 데이터를 MQTT Subscriber 에게 전달한다. Node.js는 신발가격 데이터를 받고 DataBase에 저장하는 MQTT subscriber의 역할과 Web UI를 제공하는 HTTP server의 역할을 수행한다. DataBase는 수신되는 신발가격 데이터를 저장하고, Node.js의 요청을 받으면 신발가격 데이터를 제공한다.

시스템의 동작 과정은 다음과 같다. 먼저 Node.js의 MQTT Subscriber가 MQTT Broker에게 신발가격 토픽의 구독을 요청하고, 구독요청을 받은 MQTT Broker는 MQTT Subscriber를 토픽의 구독자 리스트에 추가한 후, 웹 크롤링을 통해 신발가격데이터를 Publish하면, MQTT Broker는 해당 데이터를 MQTT Subscriber에게 전송한다. Node.js는 수신되는 신발가격 데이터를 DataBase의 보내서 저장하고 Web UI를 통해서 데이터를 보여준다.

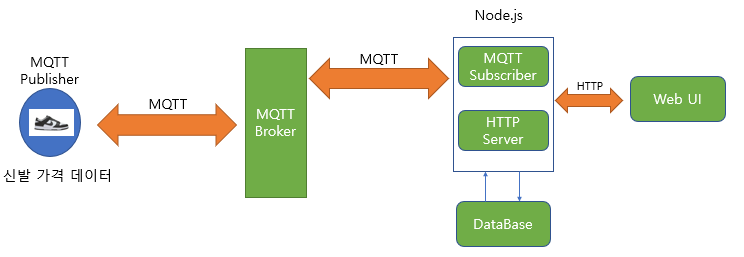


그림1. 신발 시세 모니터링 시스템

# 소스코드 설명

(1)Node.js 코드

텍스트이(가) 표시된 사진

자동 생성된 설명

초기설정 부분으로 3000번 포트설정, HTTP서버 생성, Mongo DB와 연결.

텍스트이(가) 표시된 사진

자동 생성된 설명

MQTT서버와 Mongo DB와 연결, client.on함수로 신발데이터 토픽을 구독하면 34~35함수 실행.

텍스트이(가) 표시된 사진

자동 생성된 설명

62~65: client.on 함수를 통해 메시지가 수신되면, 수신한 메시지 topic을 출력하고, 수신한 메시지의 데이터를 json객체로 변환해서 출력한다.

69~73: 토픽이 신발1 이면 cursor를 Resources.Shoe로 두고, insertOne메소드를 통해 json형식인 obj를 넣고, 에러가 발생하면 72줄, 정상 실행되면 73줄 실행.

텍스트이(가) 표시된 사진

자동 생성된 설명

115~117: io.on 메소드를 통해 웹사이트에 접속하면 socket.on 메소드를 통해 이벤트(socket\_evt\_update)를 받으면 119~126 함수 실행.

119~126: cursor변수를 DB의 Resources.Shoe로 두고 옵션을 설정해서 가장 최근 데이터를 저장해서 if문을 통해 에러가 아니면 socket.emit함수를 통해 가장 최근 데이터를 html 페이지로 보내준다.

(2)MQTT publisher 코드

텍스트이(가) 표시된 사진

자동 생성된 설명

MQTT Client 객체를 생성하고, MQTT broker서버에 접속한 후 신발 가격 데이터를 얻는 함수들을 30~35 6번 실행하고 얻어온 데이터를 publish\_data 메소드를 통해 신 발 가격 데이터를 발행을 20초에 한 번씩 실행.

텍스트이(가) 표시된 사진

자동 생성된 설명

브로커 서버에 접속하는 메소드로 주소와 client ID를 정해주고 접속의 옵션을 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Topic 객체를 설정해주고, QoS level은 0으로 설정 후 try-catch문을 통해 토픽 데이터 값을 출력하고, 예외가 발생했을 때 출력 값을 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

웹 크롤링할 주소를 String url에 저장하고 try-catch문과 Jsoup을 이용해 데이터를 가져온다. 가져온 데이터에서 필요한 부분만 element객체로 저장하고 element.text메소드를 사용해 String객체로 저장 후 json에서 호환이 안되는 쉼표를 없애 주고 return한다.

# 실행화면

최종 실행 화면을 제시하고, 최종 구현 결과에 대해 서술한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림2. Publisher 화면

웹 크롤링을 통해 데이터를 얻은 후 출력한 화면이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림3. Node.js화면

Publisher가 생성한 데이터를 구독을 통해 받음

테이블이(가) 표시된 사진

자동 생성된 설명

그림4. Mongo DB 화면으로 6개의 신발 가격 데이터가 저장 되어있음

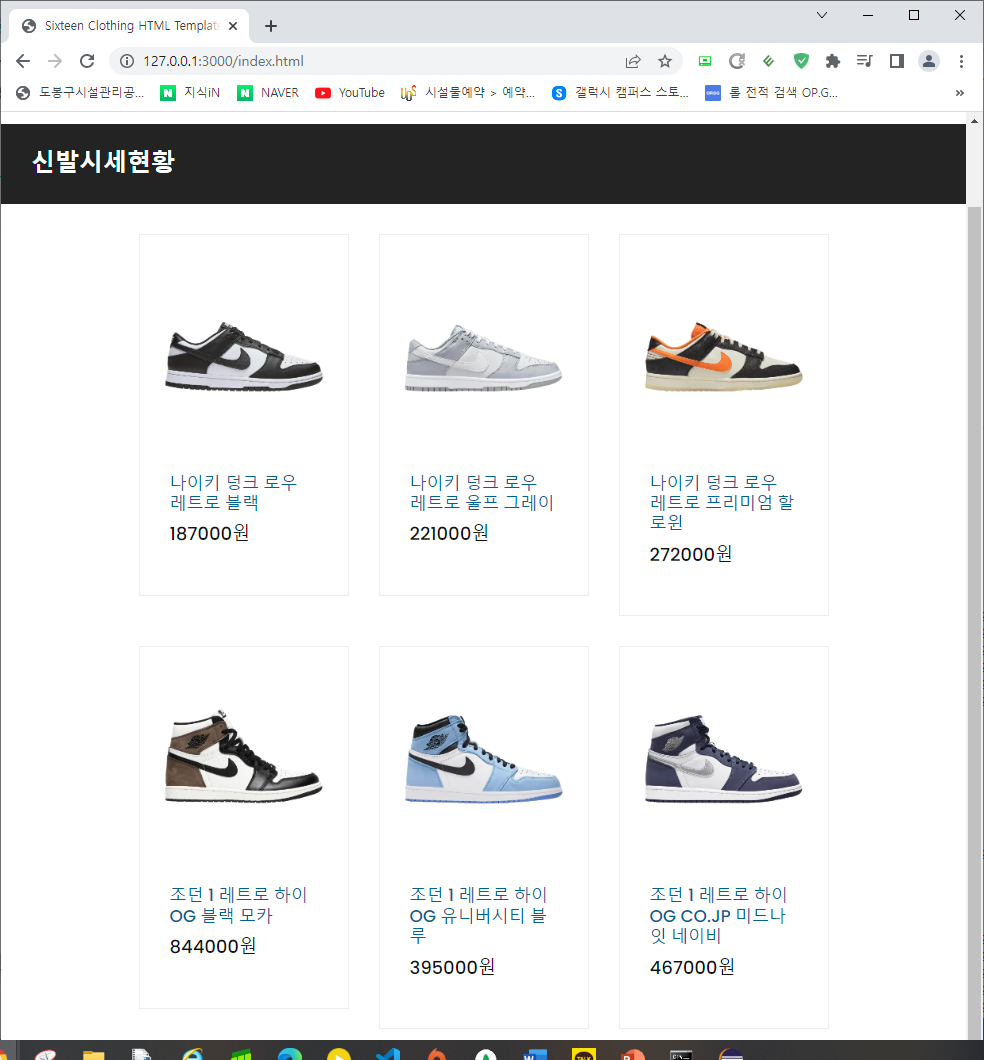


그림5. Web UI 화면

# **결론**

간략하게 알고 싶은 신발의 가격만 볼 수 있게, 웹 크롤링을 이용해 리셀 플랫폼 사이트에서 가격 부분만 가져와서 DB로 저장을 하고 Web을 통해 확인할 수 있게 하는 것이 목적이었다.

프로젝트의 진행과정은 웹 크롤링에 필요한 리셀 플랫폼 사이트를 정하고, 데이터 publish를 위한 자바코드를 수정한 후 Node.js에서 publish한 데이터를 구독 후 사용할 수 있도록 코드를 수정하고 마지막으로 Web UI를 보기 쉽게 수정하였다.

프로젝트의 결과는 20초에 한 번씩 웹 크롤링을 통해 데이터를 publish하고 MQTT Broker를 통해 Node.js에게 보내서 DB에 저장하고 html파일로 전송해주도록 구현하였다.

이론시간에 들은 MQTT 개념을 직접 프로젝트를 진행하니까 이해하기가 쉬워졌고, 신발 가격 뿐만 아니라 가격변동이 있는 다른 종류의 토픽들도 사용하면 좋을 것 같다는 생각이 든다. 앞으로는 20초마다 토픽을 받아오는 것뿐만 아니라 원할 때 Web에서 새로 고침을 하면 데이터를 새로 받아오도록 구현하면 더욱 좋은 결과물이 될 것 같다.