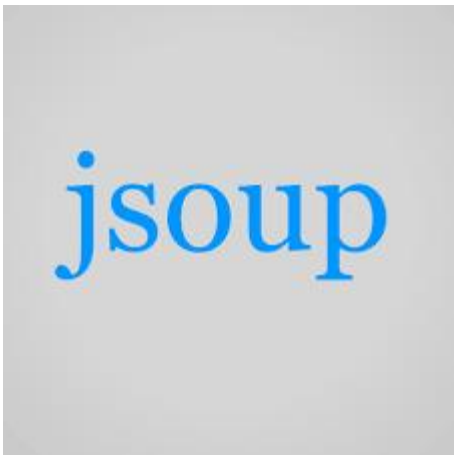
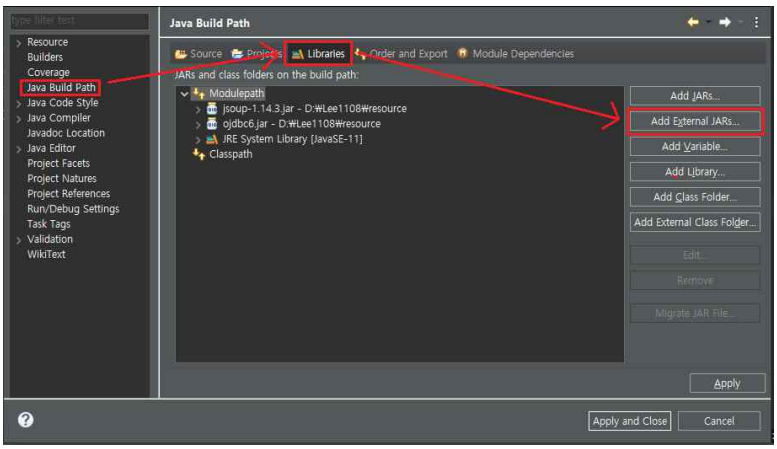
웹 크롤링

팀4
팀장 이재환
팀원 강지원
이민경
이재호
허정연

크롤링이란, 수집한 데이터들을 분류하는 것을 의미합니다. 즉 웹 크롤링 이란, 웹상의 데이터를 수집해서 분류하는 것입니다.



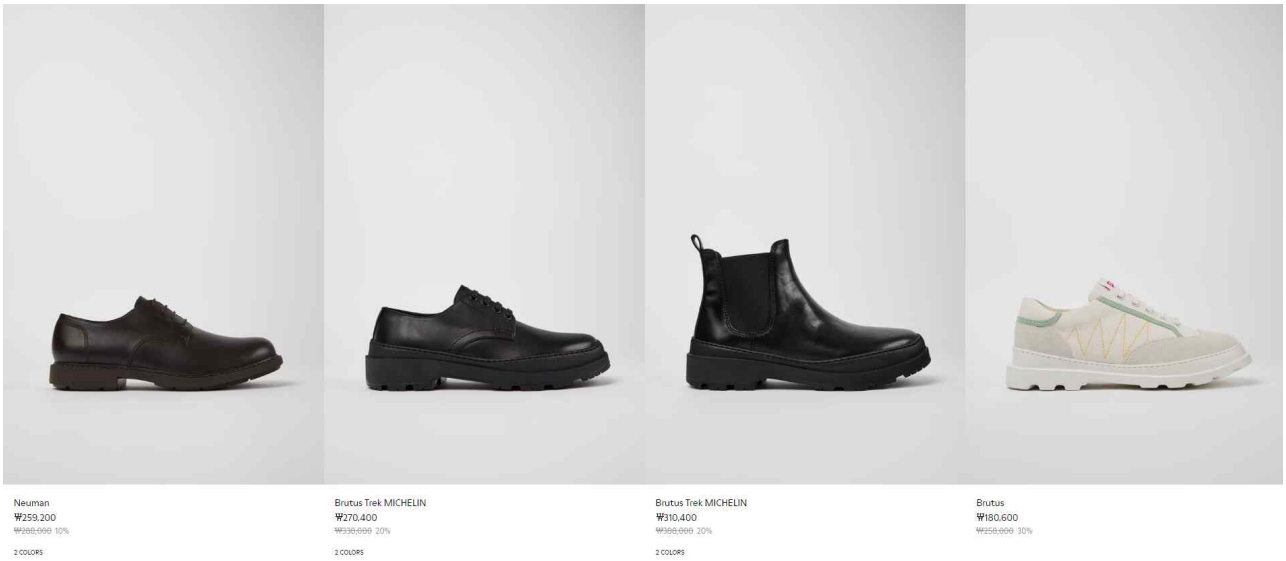


웹 크롤링을 진행하기 위해서는 jsoup 라이브러리를 사용합니다. 사전에 프로젝트 안에 라이브러리를 넣어주어야 하는데, 프로젝트를 우클릭한 고 Properties를 클릭한뒤 Java Build Path > Libraries > Add External JARs를 클릭한 후, 사용할 jsoup 파일을 넣어주면 됩니다.

이제 웹 크롤링를 실제로 해보겠습니다

저희가 이번에 크롤링한 사이트는 신발 판매 사이트인 camper이라는 곳입니다.

https://www.camper.com/ko_KR/men/shoes



실제 사이트는 이런식으로 되어있습니다.

```
ArrayList<ProductVO> pdatas = new ArrayList<ProductVO>();
```

우선, 같은 이름의 신발이 있는 경우가 있어 중복을 검사하기 위해 배열리스트를 추가하였습니다.

```
final String url = "https://www.camper.com/ko_KR/men/shoes";

Document doc = null;
try {
    doc = Jsoup.connect(url).get();
} catch (IOException e) {
    e.printStackTrace();
}
```

크롤링을 할 사이트의 주소(url)을 가져옵니다.

이제 크롤링할 대상을 분석하여야 합니다.

해당 사이트에서 저희가 가져와야 할 데이터는 총 3가지입니다.

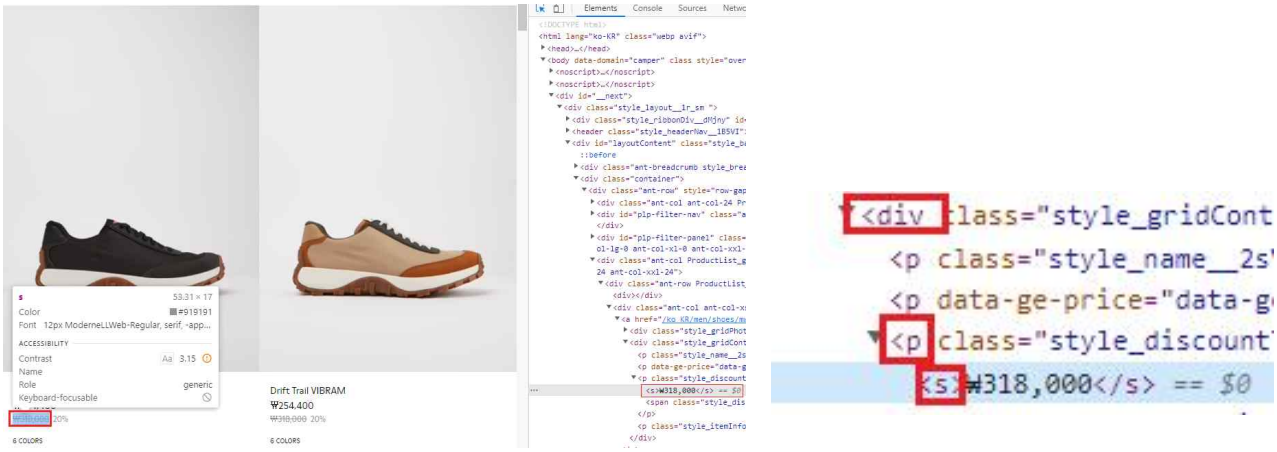
· 제품 이름

· 제품 원가

· 제품 판매가

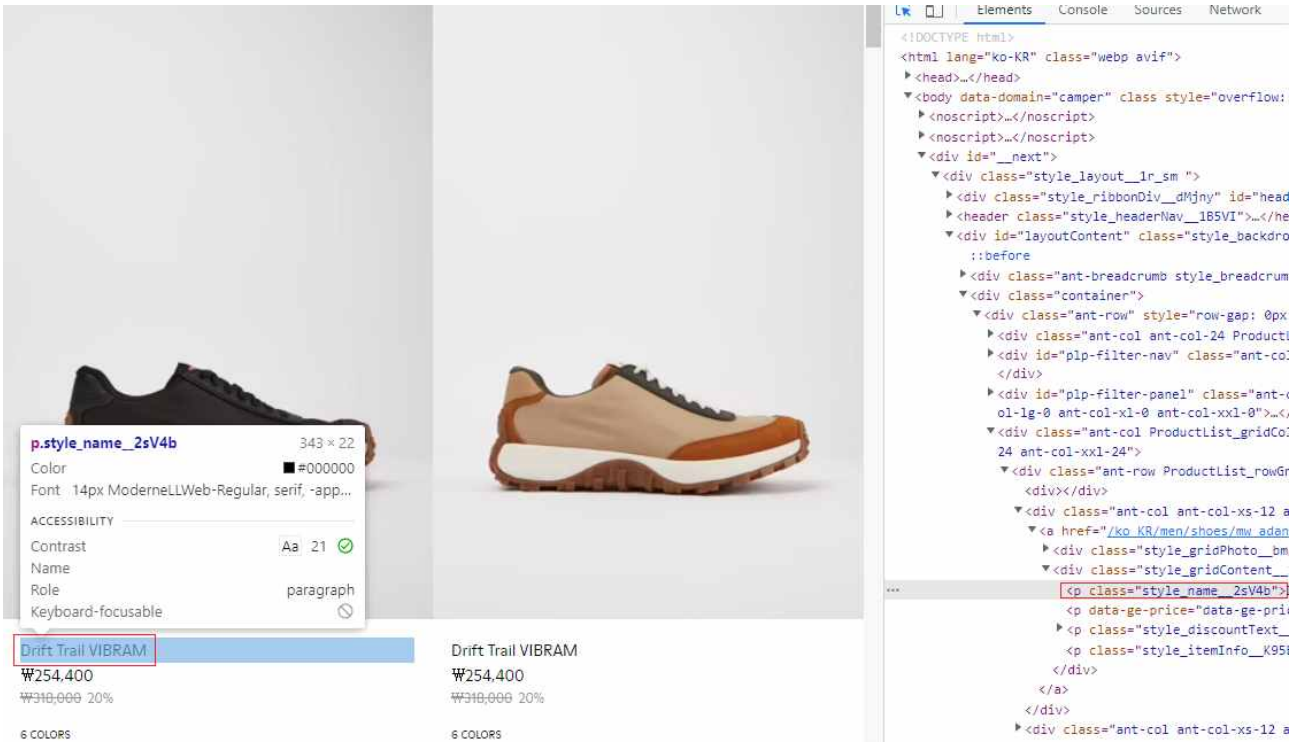
Drift Trail VIBRAM
₩254,400
~~₩318,000~~ 20%

제품 화면에 필요한 정보가 다 나와있습니다. 이제 해당 정보를 가져오기 위해 코드상의 경로를 파악하여야 합니다.



우선 제품원가입니다. 크롤링을 위해서 경로를 지정해주면 되는데, html태그를 이용합니다. 해당 정보는 div > p > s 안에 있습니다.

다음은 판매가격(할인가격)입니다. 기존처럼 태그를 통하여 지정해도 되지만, 해당 정보는 p 태그의 class = style_name_1FG3L 클래스 속성에 속해있습니다. 때문에 해당 클래스 이름을 이용하여 바로 정보를 가져올 수 있을 것 같습니다.



마지막으로 이름입니다. 제품 이름 또한 p 태그의 class = style_name_2sV4b 클래스 속성에 속해있습니다. 제품판매가와 마찬가지로 클래스 속성을 이용하여 바로 정보를 가져올 수 있을것입니다.

```
Elements eles = doc.getElementsByClass("style_name__2sV4b");
Iterator<Element> itrN = eles.iterator();

Elements eles2 = doc.getElementsByClass("style_price__1FG3L");
Iterator<Element> itrRP = eles2.iterator();

Elements eles3 = doc.select("div > p > s");
Iterator<Element> itrSP = eles3.iterator();
```

위의 정보를 바탕으로, 이름과 할인가격은 클래스 이름을 지정하고, 원가는 html 태그를 통하여 검색하였습니다.

```
while (itrN.hasNext()) {

    if(pdatas.size()==10) {
        break;
    }
```

다음은 정보를 가져오는 구문입니다. 우선 최대 10개의 정보를 가져오기로 했습니다.

Drift Trail VIBRAM
₩254,400
~~₩310,000~~ 20%

Drift Trail VIBRAM
₩254,400
~~₩310,000~~ 20%

다음은 정보를 가져와야 하는데, 위의 사진과 같이 색깔만 다르고 동일한 이름과 가격을 가진 제품이 따로 있는 것을 확인하였습니다. 이 경우에 크롤링 시 동일 제품이 저장되는데 현재 동일 제품 정보는 불필요합니다.

```
boolean isOverlap   = false;
ProductVO pvo = new ProductVO();
String name = itrN.next().text();
pvo.setName(name);

for(int i=0; i<pdatas.size(); i++) {
    if(pdatas.get(i).getName().contains(pvo.getName())) {
        isOverlap = true;
        break;
    }
}
if(isOverlap) {
    continue;
}
```

때문에 같은 제품은 한번만 표시하기 위하여 중복검사를 위한 flag 변수를 만들었습니다. 이후 pvo객체를 생성하고 이름 정보를 가져옵니다. 이후 가져온 이름이 저장한 정보와 동일한 경우, flag변수인 isOverlap을 true로 바꾸어, 저장하지 않고 다음 제품을 크롤링하도록 continue를 사용했고, 만약 flag가 false 라면 continue를 만나지 않으며, 다음 구문을 실행하게 됩니다.

```
String realPrice = itrRP.next().text();
String salePrice = itrSP.next().text();

int realIntPrice = toInt(removeNotNumeric(realPrice));
int saleIntPrice = toInt(removeNotNumeric(salePrice));
```

가격 데이터를 가져오기 때문에 int타입이 필요합니다. 하지만 처음에 크롤링 데이터를 가져올 때 String realPrice = itrRp.next().text()를 통해 가져왔기 때문에 기본 타입이 String 이며, 위에 제품판매 사진을 보면 원화 표시와 , 표시가 있어 int타입으로 변환을 해줄 수 없습니다. 때문에, 이를 제거하고 int타입으로 형변환을 위하여 두가지 메서드를 생성하여 사용하였습니다.

```
private static String removeNotNumeric(String str) {
    return str.replaceAll("\\W", "");
```

우선 replaceAll은 앞의 문자를 뒤의 문자로 대체합니다. 또한, 정규식 표현 역슬래시W 는 문자 및 숫자가 아닌 것을 매치시킵니다. 때문에 숫자가 아닌 , 이나 원화 등을 지워주었습니다.

```
private static int toInt(String str) {
    return Integer.parseInt(str);
}
```

이후, String타입을 받아 parseInt를 통해 Integer형식으로 변환하고 int타입을 반환합니다.

```
pdatas.add(pvo);
pdao.insert(pvo);
```

마지막으로, 중복검사를 위해 객체에 저장합니다.

```
----------------------------------
    < Drift Trail VIBRAM >

Serial No.          1001
[원  가]            318000₩
[할인가]            254400₩
[재  고]            4
----------------------------------
```

정상적으로 크롤링한 요소들을 가져왔습니다. 이를 정렬후 배치하여 사용자화면으로 구현하였습니다.