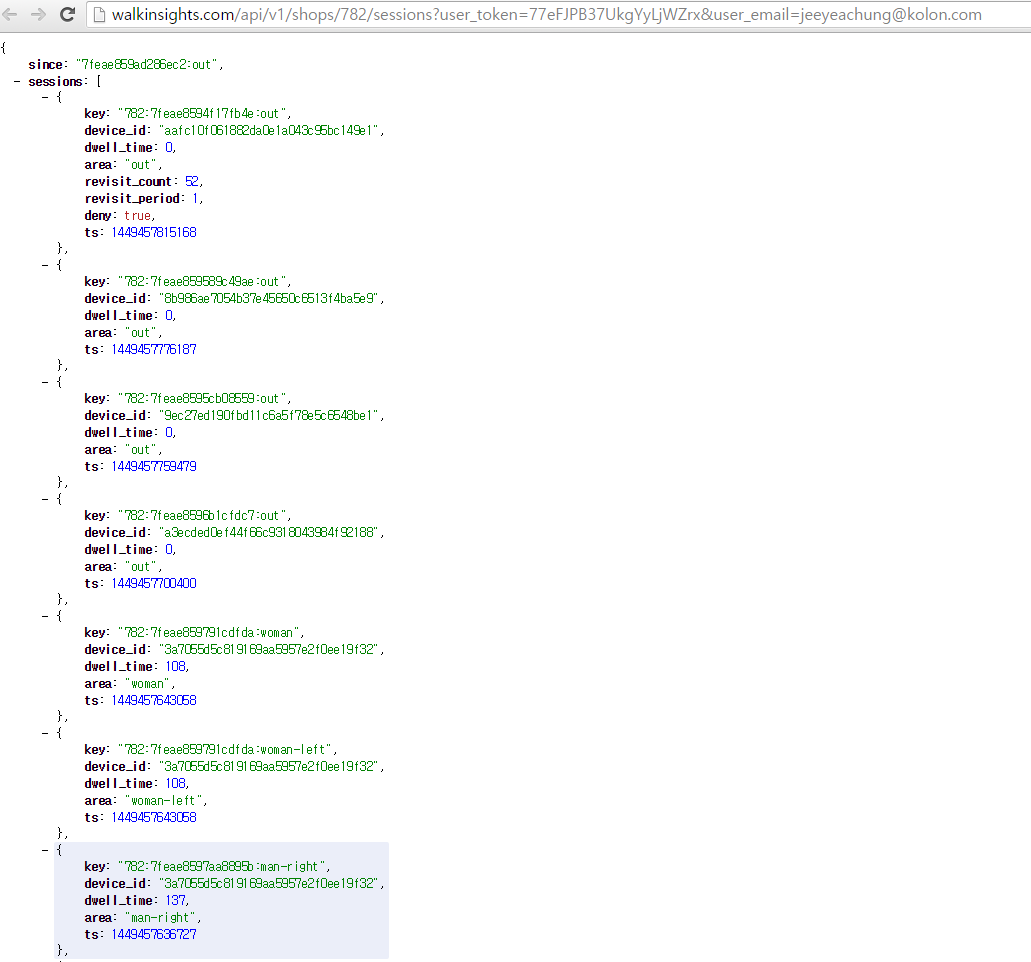
1. **매장 설명 및 고객 로그 데이터 수집 방법**



- Since 값 역산 방식  
(0x7FFFFFFFFFFFFFFF -  since) / 0x1000 = Timestamp [ msec ]

[​](https://zoyi.slack.com/archives/D073VGUMP/p1449468462000004)- Since 값 계산 방식

since = ((0x7FFFFFFFFFFFFFFF - (Time.now.to\_i \* 1000)\*0x1000))

[​](https://zoyi.slack.com/archives/D073VGUMP/p1449468497000005)- 예제

[http://walkinsights.com/api/v1/shops/786/sessions?since=](http://walkinsights.com/api/v1/shops/782/sessions?since=)7feacc565d691a63&limit= 1000&user\_token=77eFJPB37UkgYyLjWZrx&amp;user\_email=jeeyeachung@kolon.com

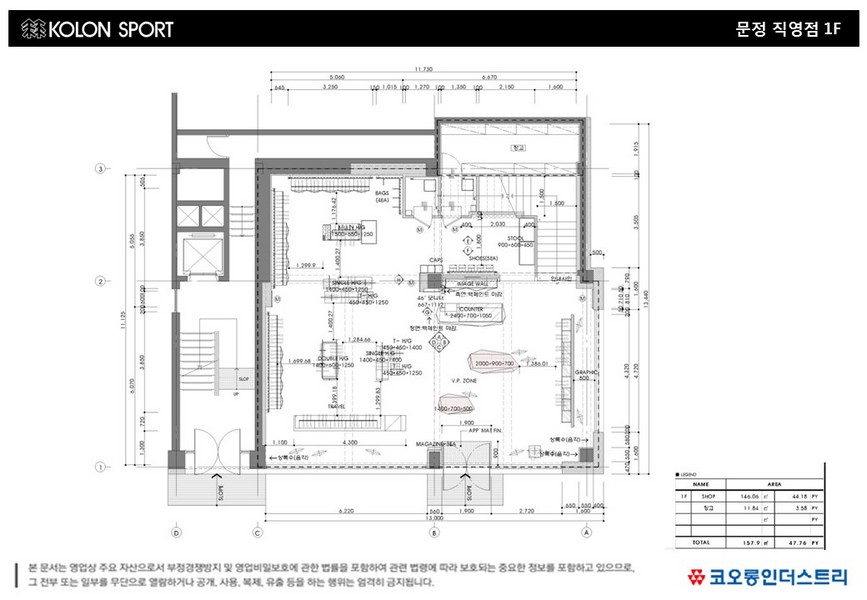
위의 주소의 Since 뒤의 값을 변환하여 고객의 로그 데이터에 접근 가능함

Since뒤의 값은 time stamp(epoch time)를 16진수로 변환한 값으로 한 번에 최대 1000개의 데이터를 볼 수 있음

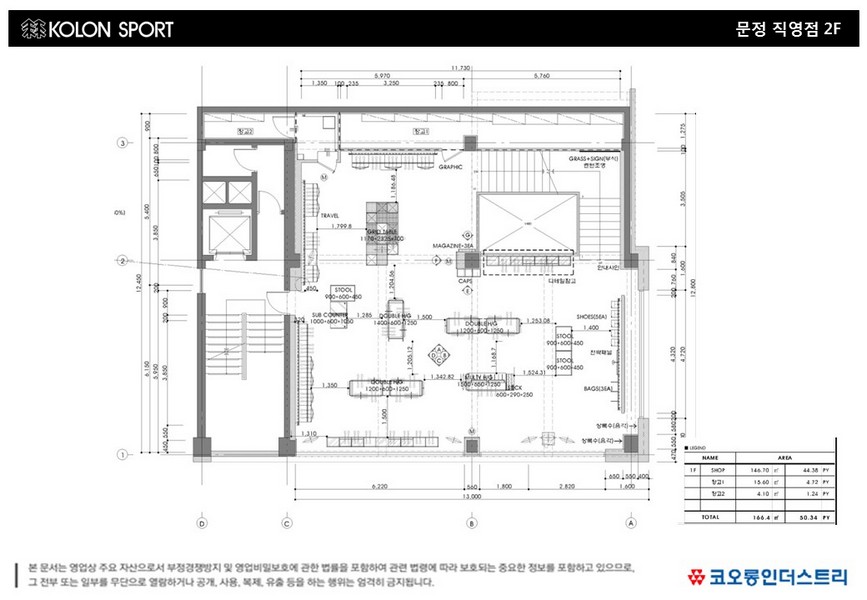
데이터의 순서가 역순이기 때문에(최신 -> 예전) 1000개의 데이터의 마지막 time stamp를 다시 since의 값으로 넣어 크롤링 해야 함

* 코오롱 스포츠 문정직영점 도면도(shop code: 786)

문정직영점 1F 전체(Area- 1F)



문정직영점 2F 존구성(Area- 2F)



Zone2  
2F-left

Zone3  
2F-right

Zone1

1F- inner

Zone2  
1F-left

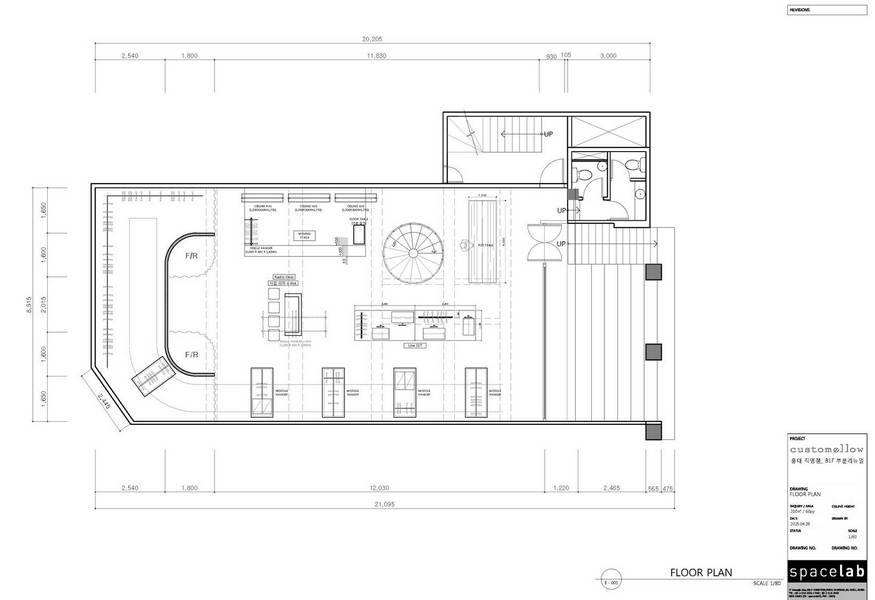
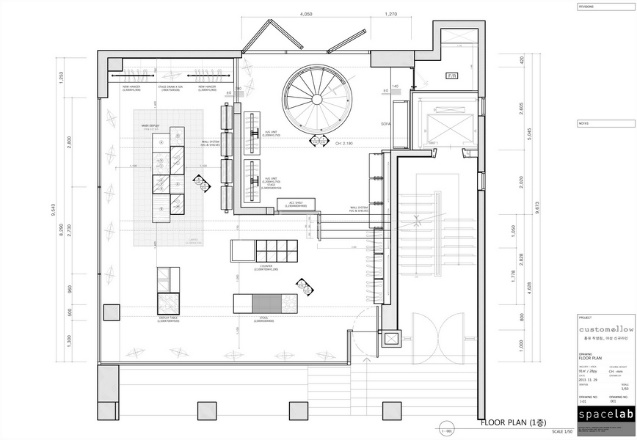
Zone3  
1F-right

Zone1  
2F-inner

* 커스텀 멜로우 홍대점 도면도(shop code: 781)

Area – 1F (1층 전체 영역 합)

Area - B1 ( 지하1층 전체 합 )



Zone1

B1-left3

Zone2  
B1-left2

Zone3  
B1-left1

Zone5  
B1-center-2

Zone4  
B1-center-1

Zone6  
B1- right

Zone1  
1F-left-1

Zone2  
1F-left-2

Zone4  
1F-Right-2

Zone3  
1F-Right-1

ENT

ENT

* 로그 데이터 속성

|  |  |
| --- | --- |
| 속성 | 의미 |
| device\_id | 핸드폰 맥 어드레스 |
| dwell\_time | 체류 시간 |
| area | 매장 내 위치 |
| revisit\_count | 재방문 횟수 |
| revisit\_period | 재방문 간격 |
| deny | 직원여부 |
| ts | 타임 스탬프 |

ts변환방법: <http://www.epochconverter.com/> 참조

1. **외부 라이브러리 사용법**

외부 라이브러리를 이클립스에서 사용하기 위하여 dependency 설정을 해주어야 함(pom.xml 파일에 dependency 추가)

* OpenCSV

|  |
| --- |
| <dependency>  <groupId>net.sf.opencsv</groupId>  <artifactId>opencsv</artifactId>  <version>2.1</version>  </dependency> |

참고 주소: <https://mvnrepository.com/artifact/net.sf.opencsv/opencsv/2.1>

* Json

|  |
| --- |
| <dependency>  <groupId>com.googlecode.json-simple</groupId>  <artifactId>json-simple</artifactId>  <version>1.1</version>  </dependency> |

참고 주소: <https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple/1.1>

1. **소스코드 설명**
   1. Craw.java

|  |  |
| --- | --- |
| 가장 최근 데이터의 json파일은 미리 하나 저장해 두어야 함(첫 input 파일로 사용) | |
| Line 55~68 | 데이터가 역순이기 때문에 타임 스탬프로부터 가장 최근 1000개를 크롤링하여 저장한 후 마지막 데이터의 타임 스탬프를 읽어 그전 데이터를 읽어 들임 |
| Line 70~74 | 또한 마지막 데이터의 타임 스탬프는 default로 epoch time으로 저장되기 때문에 해당 시간을 16진수로 변경하여야 함 |
| Line 78, 79 | “Since=” 이후의 값을 변경하며 크롤링하여야 하기 때문에 API의 중간 주소를 계속 바꾸어야 함 |
| Line 83,84 | Time limit은 크롤링하는 데이터의 시작 시점(가장 이전 날짜)으로 지정해두고, 현재 크롤링한 데이터가 time limit보다 작다면 크롤링을 멈춤 |

* 1. toCSV.java

|  |  |
| --- | --- |
| Line 33~77 | Json파일을 읽고 파싱하여 csv 파일형태로 저장 |
| Line 108 | Epoch time을 사람이 읽을 수 있는 시간대로 변경 |
| Line 99~127 | device\_id, dwell\_time, area, ts, revisit\_count, revisit\_period, deny 순으로 csv 파일에 write |

* 1. Trajectory.java

|  |  |
| --- | --- |
| Out file: User ID 별로 log 데이터 정렬(trajectory data처럼),  ID별 stay point 개수와 매장 내 총 머무른 시간,  매장 내 이동 sequence만 text파일에 저장 | |
| Line 91~126 | csv파일에서 매장이 오픈 시간일 경우에만 userID, 구역, ts, 직원 여부 등 데이터를 읽어 들임 |
| Line 129~156 | 각 고객ID별로 구역, ts, dwell\_time 등의 정보를 담을 수 있도록 LinkedHashMap을 만들어 user별로 정보를 차례차례 집어 넣음 |
| Line 163~187 | 그러나 현재 지역이 out인데 이전 구역이 out인 로그가 1개 들어있다면 이전 데이터는 불필요하기 때문에 삭제  ex) {out, out, …} |
| Line 192~311 | 고객의 데이터를 순서대로 linkedList에 insert할 때, 이전 로그와의 시간 차가 1시간 이상이면 새로운 트라젝토리로 취급함 |
| Line 322~417 | LinkedHashMap에 저장되어 있는 데이터를 다시 csv파일에 write함 |

* 1. Ordering.java, Grouping.java

|  |
| --- |
| 고객들의 trajectory파일을 읽어 들여 out, in, … 순으로 로그가 찍힌 고객들만 다시 새로운 csv파일에 저장함. 그 후 해당 파일에서 in이 찍힌 시간 순으로 정렬 한 후, 두 고객의 trajectory 길이가 5개 이상이고 in의 타임 스탬프가 60초 미만인 고객들만 두 명씩 짝을 지어 순서대로 프린트 함 |

* 1. CustomerCounting.java

|  |  |
| --- | --- |
| 지나가는 행인과 매장 내 들어온 고객을 카운팅하여 날짜 시간 별로 행인, 고객, 그 비를 프린트함. | |
| Line 81, 82 | 날짜, 시간 별로 행인과 고객을 counting하기 위하여 두 개의 LinkedHashMap을 생성함 |
| Line 92~138 | 파일을 읽어 직원 여부를 체크한 후, hashmap에 아직 고객정보가 없다면 user 데이터를 넣음 |
| Line 142~154 | 로그에 있는 타임스탬프의 시(hour)를 기준으로 현재 고객의 로그가 언제 찍혔는지 확인하고 고객의 현재 위치가 out이라면 해당 시간대의 행인 수를 하나 증가시킴 |
| Line 172~230 | 고객의 시간대를 행인 수를 증가시키는 시간대와 동일하게 하기 위하여 만약 현재 고객의 위치가 out이 아니라면 해당 고객 정보를 일단 hashmap에 저장 |
| Line 198~203 | 나중에 고객의 위치가 out인 로그가 나온다면 그 때 행인과 고객의 수를 함께 1씩 증가 |
| Line 239~250 | Array에 저장되어있는 행인 수와 고객 수를 시간 대 별로 출력 |

* 1. VisitingTimes.java

|  |  |
| --- | --- |
| 고객을 방문 횟수에 따라 1~2번 온 고객, 3~4번 온 고객, 5번 이상 방문한 고객으로 나누어 매장 내 궤적 및 stay point 개수를 구함 | |
| Line 55~137 | 각 그룹별 고객 궤적 및 체류시간을 저장할 파일 생성 |
| Line 143~164 | 유저 별 구역, 현재시간, 체류시간 등의 정보를 담을 LinkedHashMap 생성 |
| Line 208~467 | 1~2번 방문한 고객의 정보를 LinkedHashMap에 저장 |
| Line 277~467 | 고객의 데이터를 순서대로 linkedList에 삽입할 때, 이전 로그와의 시간 차가 1시간 이상이면 새로운 트라젝토리로 취급함 |
| Line 475~728 | 3~4번 방문한 고객의 정보를 LinkedHashMap에 저장 |
| Line 732~986 | 5번 이상 방문한 고객의 정보를 LinkedHashMap에 저장 |
| Line 990~1494 | 각 그룹 별로 저장되어있는 데이터를 파일에 씀 |

* 1. VisitingTimes\_Sum.java

|  |  |
| --- | --- |
| 고객의 방문 횟수(1~2번 방문, 3~4번 방문, 5번 이상 방문) 별로 고객의 수와, stay point 개수, 체류 시간을 계산 | |
| Line 17~33 | 세 개 그룹의 고객 수, stay point 개수, 체류 시간을 계산하여 저장하기 위하여 2차원 array 생성 |
| Line 36~51 | 각 array를 초기화 |
| Line 58~60 | 1~2번, 3~4번, 5번 이상 방문한 고객의 csv파일을 읽어 들임 |
| Line 68~109 | 1~2번 방문한 고객 데이터 처리 |
| Line 90~106 | 현재 요일을 계산하여 해당 시간대의 고객 수를 1만큼 증가하고, 고객의 stay point개수와 체류시간을 기존 데이터에서 더함 |
| Line 117~153 | 3~4번 방문한 고객 데이터 처리 |
| Line 160~200 | 5번 이상 방문한 고객 데이터 처리 |
| Line 204~308 | 각 방문 횟수 별 고객 수, stay point 개수, 체류 시간을 파일에 저장 |

1. **TAS**

Temporally-annotated sequence를 구하기 위하여는 먼저 frequent sequence pattern을 먼저 구해야 함(Prefixspan, SPADE 등)

Frequent sequence pattern 오픈소스:

http://www.philippe-fournier-viger.com/spmf/index.php?link=documentation.php