제목 : 알고리즘 적용 기획서#1

**내용 : Tim sort를 이용한 게시글 정렬 서비스**

게시판 내 게시글을 정렬해 사용자가 원하는 정보에 빠르게 접근할 수 있도록 합니다.

* 게시글의 정보 중 작성자, 조회수(좋아요 수), 게시물 등록 시간을 기준으로 정렬하게 하였습니다.
  + 작성자 이름순 오름차순 정렬
  + 조회수(좋아요 수) 내림차순 정렬
  + 최신 글 순서(게시물 등록 시간) 내림차순 정렬

**적용 알고리즘 : Tim Sort**

게시글 정보에서 정렬하고자 하는 데이터의 종류가 여러 개입니다. 따라서 데이터의 종류와 관계 없이 최적으로 정렬을 해야 합니다. 그렇기 때문에 자바에서 기본 타입이 아닌 객체 타입의 배열을 정렬하게 될 때는 Tim Sort로 정렬이 됩니다. (Collections.sort()의 경우 내부에서 List를 Object 배열로 변환하여 Arrays.sort()를 사용합니다!)

위와 같은 이유로 정렬하고자 하는 데이터의 종류가 여러 가지였기 때문에 데이터 종류와 관계 없이 최적으로 정렬을 하기 위해 Tim Sort를 사용했습니다.

**알고리즘 개요**

Tim Peters에 의해 만들어진 알고리즘이며, Insertion sort와 Merge sort를 결합하여 만든 정렬입니다.

Tim sort 알고리즘은 최선의 시간 복잡도 $O(n)$, 평균은 $O(nlogn)$, 최악의 경우 $O(nlogn)$입니다. 이 알고리즘은 안정적인 두 방법을 결합했기 때문에 안정적이고, 추가 메모리를 사용하지만 기존의 Merge sort에 비해 적은 추가 메모리를 사용해 다른 $O(nlogn)$ 정렬 알고리즘의 단점을 극복한 알고리즘입니다. 따라서 Java SE7, Android, Google chrome(V8) 등 많은 프로그래밍 언어에서 표준 정렬 알고리즘으로 채택 되어 사용하고 있습니다.

Insertion sort의 시간 복잡도는 $O(n^2)$ 이지만 n이 작다면 오히려 빠르다고 알려져 있는 Quick sort가 오히려 정렬하는 비용이 클 수도 있게 됩니다.

그래서 Tim sort에서는 전체를 작은 덩어리로 잘라 부분 리스트를 얻고, 나눠진 부분 리스트에 대해 Insertion sort로 정렬한 뒤 병합하는 방식으로 정렬을 진행합니다.

Merge sort를 진행할 때도 비슷한 크기의 덩어리끼리 병합하도록 구현해 최소한의 메모리를 이용합니다. 또한 두 덩어리를 병합 할 때 둘 중 크기가 더 작은 덩어리를 복사합니다. 이후 각 덩어리의 시작 부분부터 크기 비교를 해 작은 순서대로 앞을 채우면서 병합을 진행합니다. 이 방법도 두 덩어리 중 크기가 작은 덩어리를 담을 메모리가 필요하기 때문에 병합을 진행할 때 최악의 경우 n/2의 추가 메모리를 사용하게 됩니다. 따라서 Merge sort의 가장 큰 단점이었던 n의 추가 메모리 사용을 절반으로 줄일 수 있습니다.

**5. 적용 서비스 : 게시글 정렬**

**6. 적용 서비스 개발 개요**

여행 리뷰 게시판과 공지사항 게시판에 정렬 dropdown UI를 생성했습니다.

각 dropdown 메뉴를 a태그로 생성하여 클릭 시 각 정렬을 처리할 수 있도록 Servlet으로 전달했습니다. 이때, 데이터의 길이가 길지 않고, 노출되었을 때 문제되는 데이터는 없기 때문에 GET방식을 선택했으며, 어떤 데이터를 sort할지를 QueryString으로 전달했습니다.

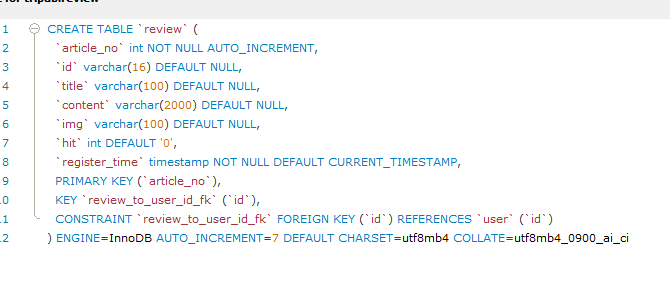
Servlet에서 request 객체를 통해 전달 받은 parameter 중 sort인 parameter의 값을 가져옵니다.

값에 따라 아래 중 하나로 정렬을 진행합니다.

1. 여행 후기 게시판 - 작성자 이름 오름차순
2. 여행 후기 게시판 - 좋아요 수 내림차순
3. 여행 후기 게시판 - 최신 작성글 내림차순

여행 후기 게시판과 공지사항 게시판의 DB 테이블 속성은 똑같이 정의 되어 있기 때문에 정렬하는 데이터를 제외한 로직은 동일하게 구현하였습니다.

* 여행 후기 게시판 :: Review 클래스
* 공지사항 게시판 :: Notice 클래스

****

* id : 작성자 id
* hit : 조회수(좋아요 수)
* register\_time : 등록 시간

**1) 작성자 이름 오름차순 정렬**

게시판에 있는 전체 게시글 데이터를 조회해 Notice(Review)타입의 리스트로 저장합니다.

Collections.sort() 를 이용해 Notice(Review) 클래스의 id로 정렬합니다.

id는 String 타입이기 때문에 String 클래스의 compareTo()를 사용하여 비교합니다.

**2) 조회수(좋아요 수) 내림차순 정렬**

게시판에 있는 전체 게시글 데이터를 조회해 Notice(Review)타입의 리스트로 저장합니다.

Collections.sort() 를 이용해 Notice(Review) 클래스의 like로 정렬합니다.

like는 int 타입이기 때문에 Integer 클래스의 compare()를 사용하여 비교합니다.

내림차순으로 정렬해야 하기 때문에 return에 - 를 붙여 역순으로 정렬되도록 하여 내림차순을 구현했습니다.

**3) 최신 작성글 내림차순 정렬**

게시판에 있는 전체 게시글 데이터를 조회해 Notice(Review)타입의 리스트로 저장합니다.

Collections.sort() 를 이용해 Notice(Review) 클래스의 registerTime로 정렬합니다.

registerTime은 String 타입이기 때문에 String 클래스의 compareTo()를 사용하여 비교합니다.