G조 발표자료

개발 환경

- 매일 오전 녹스 미팅으로 인사 및 진행 사항 공유
- 브랜칭 전략: Github Flow 방식
  - 신규 브랜치는 master 브랜치에서 생성
  - 이슈 단위로 브랜치 네이밍하여 생성, PR을 통해 master 브랜치로 merge
- CI: Circle CI 적용, 빌드 및 TC 검증
- 이슈 관리: Github 이슈 탭 사용
- 개발 현황 관리: Github Project Kanban 사용
- 리뷰 정책: 1명 이상 Approved -> Merge

개발 진행 과정

1. 구조 설계 및 인터페이스 구현 코드 PR
2. 역할 분배: 전처리 모듈 2명, 파싱 & 커맨드 1명, 비교 연산 모듈 1명
3. 각자 개발 및 PR, 메신저로 리뷰 요청, 오후 4시에는 정기적 리뷰시간

주요 PR Point - 1

항목	상세
배경	Refactor Query Preprocessor
이유/목적/의도	Query Preprocessor 모듈의 구조를 리팩토링 했습니다. QueryPreprocessor 가 너무 많은 책임을 가지고 있어서 책임을 분산 시켰습니다.
활동 내용	https://github.ecodesamsung.com/Best-Reviewer-3-11/TeamProject_GoodToMe/pull/59 기존 QueryProcessor 가 칼럼 종류에 따라 Option 처리 함수를 호출하고 있었던 부분을 변경했습니다. 쿼리로 들어온 칼럼 종류 따라 FieldOptionHandler 인터페이스를 구현한 클래스를 새로 만들고 칼럼에 따라 핸들러가 생성 및 처리하게 되는 구조로 변경했습니다.
기대 효과	FieldOptionHandler 인터페이스를 추가하고, 칼럼 별로 구현 클래스를 따로 가지게 되어, 칼럼에 따른 옵션 처리에 변경 사항이 생기더라도 QueryPrereprocessor 는 변경하지 않고 해당하는 Handler 만 변경할 수 있습니다. 유지보수성 향상
카테고리	Refactoring

Conversation 4 | Commits 2 | Checks 0 | Files changed 8 | +107 −57

Changes from all commits | File filter... | × Clear filters | Jump to... | 0 / 8 files viewed | Review changes

18	src/main/java/com/sec/bestreviewer/command/queryprocessor/BirthDayOptionHandler.java	Viewed
10	src/main/java/com/sec/bestreviewer/command/queryprocessor/CareerLevelOptionHandler.java	Viewed
10	src/main/java/com/sec/bestreviewer/command/queryprocessor/CertificationOptionHandler.java	Viewed
10	src/main/java/com/sec/bestreviewer/command/queryprocessor/EmployeeNumberOptionHandler.java	Viewed
7	src/main/java/com/sec/bestreviewer/command/queryprocessor/FieldOptionHandler.java	Viewed
18	src/main/java/com/sec/bestreviewer/command/queryprocessor/NameOptionHandler.java	Viewed
17	src/main/java/com/sec/bestreviewer/command/queryprocessor/PhoneNumberOptionHandler.java	Viewed
74	src/main/java/com/sec/bestreviewer/command/queryprocessor/QueryPreprocessor.java	Viewed

```
8   8   public class QueryPreprocessor {
9   9       public String process(QueryCondition queryCondition, Employee employee) {
10      -       switch(queryCondition.getColumn()) {
        10  +       String option = queryCondition.getOption2();
        11  +       String column = queryCondition.getColumn();
        12  +
        13  +       FieldOptionHandler optionHandler = getFieldOptionHandler(column);
        14  +       return optionHandler.process(option, employee);
        15  +   }
        16  +
        17  +   private FieldOptionHandler getFieldOptionHandler(String column) {
        18  +       switch(column) {
11      19              case FIELD_NAME:
12      -                   return processName(queryCondition, employee);
        20  +               return new NameOptionHandler();
13      21              case FIELD_PHONE_NUMBER:
14      -                   return processPhoneNumber(queryCondition, employee);
        22  +               return new PhoneNumberOptionHandler();
15      23              case FIELD_BIRTH_DAY:
16      -                   return processBirthDay(queryCondition, employee);
        24  +               return new BirthDayOptionHandler();
17      25              case FIELD_CAREER_LEVEL:
18      -                   return processCareerLevel(queryCondition, employee);
        26  +               return new CareerLevelOptionHandler();
19      27              case FIELD_EMPLOYEE_NUMBER:
20      -                   return processEmployeeNumber(queryCondition, employee);
21      -               case FIELD_CERTIFICATION:
22      -                   return processCertification(queryCondition, employee);
        28  +               return new EmployeeNumberOptionHandler();
        29  +           case FIELD_CERTIFICATION:
        30  +               return new CertificationOptionHandler();
```

주요 PR Point - 2

항목	상세
배경	각 column 값 비교를 위한 비교로직 리팩토링
이유/목적/의도	SCH, 등의 명령어에서 입력받은 각 column 값을 비교를 해야 하는데, 각 옵션과 column 값에 따라 비교 연산하는 부분이 달라져야 함. 부등호 옵션에 따라 확인하기 쉽도록 함수명으로 분리하고, 각 column별로 비교하는 부분을 처리할 수 있도록 클래스를 구분하여 역할을 위임하도록 변경
활동 내용	https://github.ecodesamsung.com/Best-Reviewer-3-11/TeamProject_GoodToMe/pull/38 column값과 비교 옵션에 따라 if/else 로 처리하는 부분을 별도 클래스와 함수로 분리하여 가독성과 유지보수성을 더 좋도록 리팩토링
기대 효과	신규 column 값이 추가되더라도 기존 코드를 거의 수정하지 않고, 추가 Comparator 클래스에서 구현할 수 있도록 하여 유지보수성이 용이
카테고리	TDD, CleanCode, Refactoring

Conversation 9 | Commits 5 | Checks 0 | Files changed 5 | +146 −5

Changes from all commits | File filter... | Clear filters | Jump to... | 0 / 5 files viewed | Review changes

23	src/main/java/com/sec/bestreviewer/command/queryprocessor/ComparisonQueryProcessor.java	Viewed
34	src/main/java/com/sec/bestreviewer/command/queryprocessor/EmployeeNumComparator.java	Viewed
9	src/main/java/com/sec/bestreviewer/command/queryprocessor/QueryComparator.java	Viewed
19	src/main/java/com/sec/bestreviewer/command/queryprocessor/StringQueryComparator.java	Viewed
66	src/test/java/com/sec/bestreviewer/ComparisonQueryProcessorTest.java	Viewed

```
+         String column = queryCondition.getColumn();
+
+         if (column.equals("employeeNum")){
+             int queryNumber = Integer.parseInt(queryValue);
+             int convertedQueryNumber = queryNumber >= YEAR_1990_EMPLOYEE_NUMBER ?
+                     YEAR_PREFIX_19 + queryNumber : YEAR_PREFIX_20 + queryNumber;
+             int processedNumber = Integer.parseInt(processedValue);
+             int convertedTargetNumber = processedNumber >= YEAR_1990_EMPLOYEE_NUMBER ?
+                     YEAR_PREFIX_19 + processedNumber : YEAR_PREFIX_20 + processedNumber;
+
+             switch (compareOption) {
+                 case COMPARE_OPTION_GREATER_THAN:
+                     return convertedQueryNumber > c
+                 case COMPARE_OPTION_GREATER_THAN_EQ
+                     return convertedQueryNumber >=
+                 case COMPARE_OPTION_SMALLER_THAN:
+                     return convertedQueryNumber < c
+                 case COMPARE_OPTION_SMALLER_THAN_EQ
+                     return convertedQueryNumber <=
+                 default:
+                     return convertedQueryNumber ==
+             }
+         } else {
+             switch (compareOption) {
+                 case COMPARE_OPTION_GREATER_THAN:
+                     return queryValue.compareTo(pro
+                 case COMPARE_OPTION_GREATER_THAN_EQ
+                     return queryValue.compareTo(pro
+                 case COMPARE_OPTION_SMALLER_THAN:
+                     return queryValue.compareTo(pro
+                 case COMPARE_OPTION_SMALLER_THAN_EQ
+                     return queryValue.compareTo(pro
+                 default:
+                     return queryValue.equals(proces
+             }
+
```

TDD 로 우선 구현

같은 PR 내에서
리팩토링 후 커밋

```
12 12     public boolean process(QueryCondition queryCondition, String processedValue) {
13 13         String compareOption = queryCondition.getOption3();
14 14         String queryValue = queryCondition.getValue();
   15 +       String column = queryCondition.getColumn();
   16 +
   17 +       QueryComparator comparator = getComparator(column);
   18 +
15 19         switch (compareOption) {
16 20             case COMPARE_OPTION_GREATER_THAN:
17    -              return queryValue.compareTo(processedValue) > 0;
   21 +           return comparator.greaterThan(queryValue, processedValue);
18 22             case COMPARE_OPTION_GREATER_THAN_EQUALS:
19    -              return queryValue.compareTo(processedValue) >= 0;
   23 +           return comparator.greaterThanEquals(queryValue, processedValue);
20 24             case COMPARE_OPTION_SMALLER_THAN:
21    -              return queryValue.compareTo(processedValue) < 0;
   25 +           return comparator.smallerThan(queryValue, processedValue);
22 26             case COMPARE_OPTION_SMALLER_THAN_EQUALS:
23    -              return queryValue.compareTo(processedValue) <= 0;
   27 +           return comparator.smallerThanEquals(queryValue, processedValue);
24 28             default:
25    -              return queryValue.equals(processedValue);
   29 +           return comparator.equals(queryValue, processedValue);
   30 +       }
   31 +   }
   32 +
   33 +   private QueryComparator getComparator(String column) {
   34 +       if (column.equals("employeeNum")){
   35 +           return new EmployeeNumComparator();
   36 +       } else {
   37 +           return new StringQueryComparator();
26 38         }
27 39     }
   40 +
28 41 }
```

주요 PR Point - 3

항목	상세
배경	입력되는 명령어로부터 QueryProcessor로 전달되는 QueryCondition 객체 생성
이유/목적/의도	- 주어진 명령어를 파싱하여 TokenGroup으로 전달 된 옵션과 파라미터를 가공하여 QueryCondition 객체를 생성하고 command에 전달함 - 가능하면 기존 API 수정 없이 수정사항을 적용하기 위해 ParameterParser 클래스 생성
활동 내용	https://github.ecodesamsung.com/Best-Reviewer-3-11/TeamProject_GoodToMe/pull/30 옵션과 파라미터를 QueryCondition의 옵션과 파라미터로 변경하여 QueryCondition객체 생성하도록 어댑테이션 함
기대 효과	- API 변경이 없기 때문에 TC 수정 없이 변경된 구조에서 명령어 수행이 가능하도록 수정됨
카테고리	Refactoring

Conversation 1 | Commits 5 | Checks 0 | Files changed 2 | +58 −32

Changes from all commits | File filter... | Clear filters | Jump to... | 0 / 2 files viewed | Review changes

34 src/main/java/com/sec/bestreviewer/CommandFactory.java Viewed

56 src/main/java/com/sec/bestreviewer/command/querycondition/QueryCondition.java Viewed

```
34
35  +   static class ParameterParser {
36  +       ArrayList<QueryCondition> queryConditions = new ArrayList<QueryCondition>();
37  +       String joinOperator = "";
38  +       void parse(String cmd, List<String> options, List<String> params) {
39  +           if (CMD_ADD.equals(cmd) || CMD_CNT.equals(cmd))
40  +               return;
41  +           String[] parameters = params.toArray(new String[0]);
42  +           queryConditions.add(makeQuery(options, parameters));
43  +       }
44  +
45  +       private QueryCondition makeQuery(List<String> options, String[] parameters) {
46  +           QueryCondition queryCondition = new QueryCondition();
47  +           queryCondition.setOptions(options.toArray(new String[2]));
48  +           queryCondition.setParams(parameters);
49  +           return queryCondition;
50  +       }
51  +   }
52      static Command buildCommand(String cmd, List<String> options, List<String> params)
```

회고(진행 시 느낀점 포함)

- 좋았던 점
  - 팀원들의 적극적인 PR 리뷰 참여
  - git을 사용해 소스 관리 및 리뷰를 하며 프로젝트를 진행한 경험
  - TDD 방법론을 실제 프로젝트에서 진행한 경험
- 아쉬웠던 점
  - 프로젝트 시작 단계에서 구조 설계에 대해 더 충분한 의사소통이 되지 못한 점
  - 비대면 환경으로 인해 집합교육 대비 프로젝트에 온전히 집중하기 어려웠던 점
- Keep
  - 적극적인 리뷰 참여
  - TDD 적용
- Problem
  - 프로젝트 시작 단계에서 구조설계에 대한 의사소통 부족
- Try
  - 프로젝트 시작 단계에서 프로젝트 이해 및 설계와 관련하여 팀원 간 충분히 논의하기