

목록

Spring 게시판 만들기#1.회원가입.....	1
Spring 게시판 만들기#2.로그인.....	15
Spring 게시판 만들기#3.게시판.....	25
Spring 게시판 만들기#4.페이징.....	37
Spring 게시판 만들기#5.마이페이지.....	49

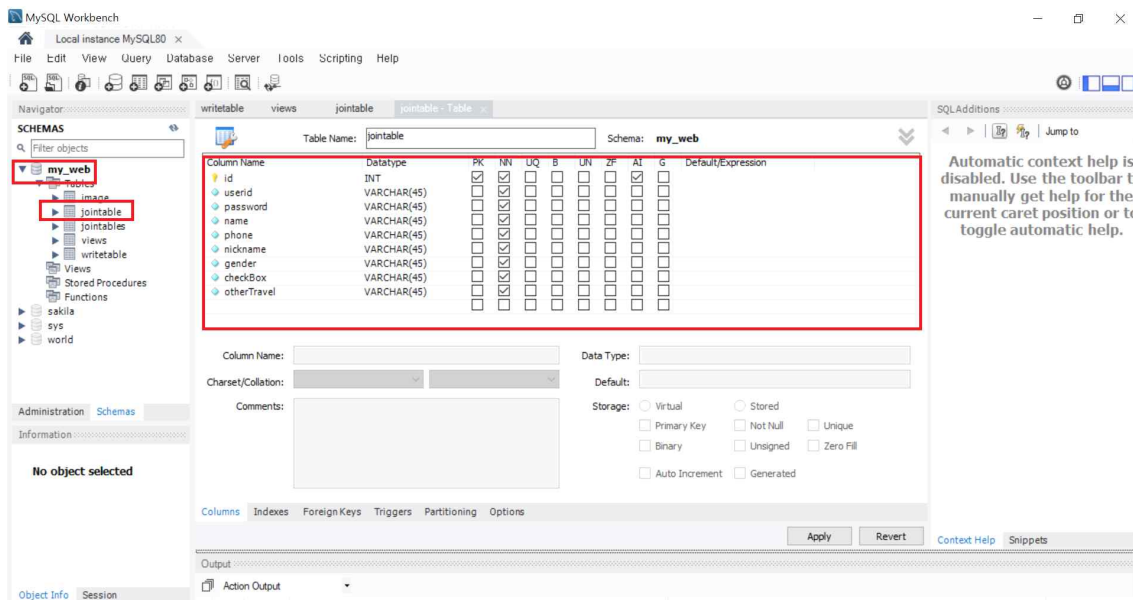
1.회원가입 part

1.회원가입을 할 때 사용자가 작성해야하는 내용.

※mysql 스키마-myweb,테이블이름-jointable, pk-id

- 아이디/String userId/mysql-> userId(varchar(45))
- 패스워드/String password/mysql-> password(varchar(45))
- 패스워드 확인/String passwordCheck/mysql-> 생략
- 이름/String name/mysql-> name(varchar(45))
- 핸드폰/String phone/mysql-> phone(varchar(45))
- 닉네임/String nickName/mysql-> nickName(varchar(45))
- 성별/String gender/mysql-> gender(varchar(45))
- 최근여행지 체크박스/String checkBox/mysql-> checkBox(varchar(45))
- 기타 여행지(체크박스에 해당되지 않을 때)
/String otherTravel/mysql-> otherTravel(varchar(45))

2.mysql 테이블 설정



3.사용자에게 가장 먼저 보여지는 welcome.jsp에서 회원가입 창으로 넘어가는 방법

여행정보사이트

접속하기 ▾

로그인

회원가입

1)welcome.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <meta name="viewport" content="width=device" ,initial-scale="1">
8 <link rel="stylesheet" href="/resource/css/bootstrap.min.css">
9 <title>여행정보사이트</title>
10 </head>
11 <body>
12
13 <nav class="navbar navbar-default">
14 <div class="navbar-header">
15 <button type="button" class="navbar-toggle collapsed"
16   data-toggle="collapse" data-target="#bs-example-navbar-collapse-1"
17   aria-expanded="false">
18 <span class="icon-bar"></span> <span class="icon-bar"></span> <span
19   class="icon-bar"></span>
20 </button>
21 <a class="navbar-brand" href="/">여행정보사이트</a>
22 </div>
23 <div class="collapse navbar-collapse"
24   id="bs-example-navbar-collapse-1">
25 <ul class="nav navbar-nav">
26
27 </ul>
28
29 <ul class="nav navbar-nav navbar-right">
30 <li class="dropdown"><a href="#" class="dropddown-toggle"
31   data-toggle="dropdown" role="button" aria-haspopup="true"
32   aria-expanded="false">접속하기<span class="caret"></span></a>
33 <ul class="dropdown-menu">
34 <li><a href="/mapping/Login">로그인</a></li>
35 <li><a href="/paths/join">회원가입</a></li>
36 </ul>
37 </li>
38 </ul>
39
40 </div>
41 </nav>
42
43 <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
44 <script src="/resource/js/bootstrap.min.js"></script>
```

welcome.jsp 살펴보자!

1)부트스트랩 설정

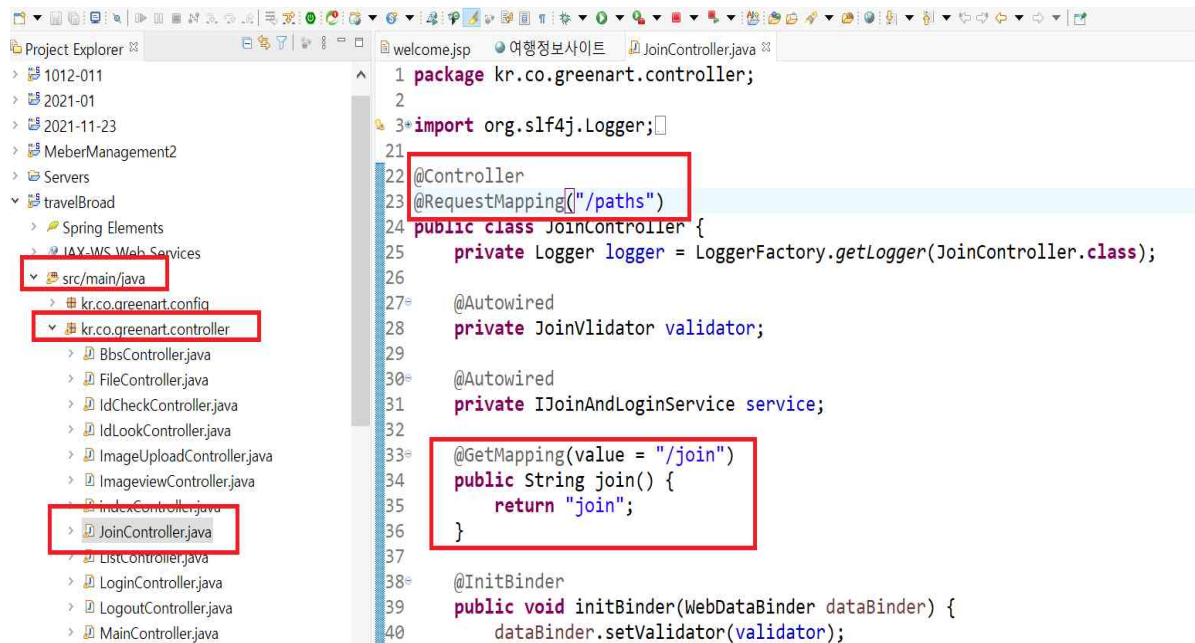
```
<link rel="stylesheet" href="/resource/css/bootstrap.min.css">
<script src="/resource/js/bootstrap.min.js"></script>
```

2)회원가입 창으로 이동할 수 있게 하는 html<a>태크.

```
<li><a href="/paths/join">회원가입</a></li>
```

- 1.html<a>태크를 사용하여 사용자가 회원가입을 클릭을 하게 되면 URL(인터넷 주소창)에 http://localhost:8080/paths/join 주소가 생기게 됨.
- 2.회원가입 창을 보여주는 컨트롤러가 get방식으로 해당 주소를 받아 회원가입 jsp를 뷰에게 전달을 합니다.

2)joinController



joinController를 살펴보자!

1)@Controller 어노테이션 추가.

어노테이션이 적용된 클래스는 "Controller"임을 나타내고, bean으로 등록되며 해당 클래스가 Controller로 사용됨을 Spring Framework에 알립니다.

2)@RequestMapping 어노테이션 추가.

해당 어노테이션이 선언된 클래스의 모든 메소드가 하나의 요청에 대한 처리를 할경우 사용한다. 예를들어, "/Paths" 요청에 대해 공통적으로 처리해야될 클래스라는 것을 의미합니다. 또한, 요청 url에 대해 해당 메소드에서 처리해야 되는 경우에도 사용됩니다.

3)@GetMapping 어노테이션(value = "/join")작성

```
@GetMapping(value = "/join")
public String join() {
    return "join";
}
```

HTML <a>태그를 통해 → http://localhost:8080/paths/join(get방식) →

Controller에서 @RequestMapping("/paths")→@GetMapping(vale="/join")

→ String join 메서드에서 return 값을 사용자에게 보여줄 jsp 이름을 적어줍니다.

저의 경우에는 join.jsp가 있기 때문에 return "join"을 적어주었습니다.

WebConfig.java 설정에서 자동적으로 뷰이름.jsp를 붙여주는 설정을 하였기 때문에 return "join"으로 작성을 합니다. 설정 매뉴얼을 다시 살펴보면 좋을 것 같습니다.

3)join.jsp

1. 사용자에게 보여지는 join.jsp

2. view에 있는 join.jsp

<html 타입>

- 아이디/input type =“text”
- 패스워드/input type =“password”
- 패스워드 확인/input type =“password”
- 이름/input type =“text”
- 핸드폰/input type =“text”
- 닉네임/input type =“text”
- 성별/ “button”
- 최근여행지 체크박스/ “checkbox”
- 기타 여행지(체크박스에 해당되지 않을 때)
/input type =“text”

실제 예시)

```
<div class="container">
  <div class="col-Lg-4"></div>
  <div class="col-Lg-4">
    <div class="jumbtron" style="padding-top: 20px;">
      <form:form modelAttribute="joinInfo" method="post" action="/paths/join" onsubmit ="return che
      <h3 style="text-align: center;">회원가입화면</h3>

      <div class="form-group id_div">
        <div class="col-auto id_input">
          <form:input type="text" class="form-control" placeholder="아이디"
            id="user" name="userId" path="userId" maxlength="20" />
          <form:errors path="userId" />
        </div>

        <div class="col-auto id_button">
          <button class="btn btn-secondary" id="duplicate_check"
            type="button" >중복체크</button>
        </div>
      </div>
    </div>
  </div>
</div>
```



```

<div class="form-group">
  <form:input type="password" class="form-control"
    id="pass" placeholder="비밀번호" name="userPassword" path="password"
    maxlength="20" />
  <form:errors path="password" />
</div>
<div class="form-group">
  <form:input type="password" class="form-control"
    placeholder="비밀번호 확인" name="userPasswordCheck"
    id="passcheck" path="passwordCheck" maxlength="20" />
  <form:errors path="passwordCheck" />
</div>
<div class="form-group">
  <form:input type="text" class="form-control" placeholder="이름"
    id="names" name="userName" path="name" maxlength="20" />
  <form:errors path="name" />
</div>
<div class="form-group">
  <form:input type="text" id="phone" class="form-control"
    placeholder="핸드폰 번호" name="phones" path="phone" maxlength="20" />
  <form:errors path="phone" />
</div>
<div class="form-group">
  <form:input type="text" class="form-control" placeholder="닉네임"
    id="nickId" name="UsernickName" path="nickName" maxlength="20" />
  <form:errors path="nickName" />
</div>
<div>
  <div class="col-auto">
    <button class="btn btn-secondary" id="nickName_check"
      type="button">닉네임 중복체크</button>
  </div>
</div>

<div class="form-group" style="text-align: center">
  <div class="btn-group" data-toggle="buttons">
    <label class="btn btn-primary active"><form:radio button
      id="men" path="gender" label="남자" value="남자" />
    </label> <label class="btn btn-primary"><form:radio button
      id="woman" path="gender" label="여자" value="여자" />
    </label>
    <form:errors path="gender" />
  </div>
</div>

<div>최근여행지는?</div>
<div class="form-group">
  <form:checkbox class="form-group-input" path="checkBox"
    id="inlineCheckbox1" value="서울" />
  <label class="form-group-input" for="inlineCheckbox1">서울</label>

  <form:checkbox class="form-group-input" path="checkBox"
    id="inlineCheckbox2" value="부산" />
  <label class="form-group-input" for="inlineCheckbox1">부산</label>

```

```

<div class="form-group">
  <form:input type="text" class="form-control" placeholder="기타여행지"
    name="travel" path="otherTravel" maxlength="20" />
  <form:errors path="otherTravel" />
</div>

<input type="submit" id="btn" class="btn btn-primary form-control"
  value="회원가입" disabled >
</form:form>

```

3. HTML <FORM>을 이용

● <FORM>이란 웹 페이지에서의 입력 양식을 의미합니다.

텍스트 필드에 글자를 입력하거나, 체크박스나 라디오 버튼을 클릭하고 제출 버튼을 누르면 백엔드 서버에 양식이 전달되어 정보를 처리하게 됩니다.

```

<div class="jumbotron" style="padding-top: 20px;">
  <form:form modelAttribute="joinInfo" method="post" action="/paths/join" onsubmit="return checkit()">
    <h3 style="text-align: center;">회원가입화면</h3>

    <div class="form-group id_div">
      <div class="col-auto id_input">
        <input type="text" class="form-control" placeholder="아이디"
          id="user" name="userId" path="userId" maxlength="20" />
        <form:errors path="userId" />
      </div>

      <div class="col-auto id_button">
        <button class="btn btn-secondary" id="duplicate_check"
          type="button" >중복체크</button>

        <input type="submit" id="btn" class="btn btn-primary form-control"
          value="회원가입" disabled >
      </form:form>
    </div>

```

저의 경우에는 <form:form></form:form>으로 되어 있는데 이부분은 modelAttribute를 사용하기 위해 작성된 내용입니다. 원래 형식은 <form></form>입니다.

사용자가 입력한 양식 내용을 폼에 담아 method = "post" action = "/paths/join"으로 컨트롤러를 찾아 정보를 전달을 합니다. method는 폼 전송 방식으로 get과post로 나뉩니다. get으로 보내게되면 url 주소창에 사용자의 모든 정보가 노출되어 보내지기 때문에 사용자가 입력한 양식 데이터는 post 형식으로 보내줍니다.

그렇다고 해서 사용자의 정보가 안전한 것은 아닙니다.

하지만 일단은 주소창에 사용자가 입력한 정보를 보여주지 않기 위해서 post방식으로 보냅니다.

4. modelAttribute을 이용하기.

modelAttribute을 사용하여 사용자의 이름,아이디,폰 등 정보를 한꺼번에 담아 랑앤락 통에 담아 controller에게 전달을 합니다. 이렇게 보내면 어떤 이점이 있을까요?

● joincontroller가 받을 때 @RequestBody 와 @RequestParam의 형태로 값을 하나하나씩 다 받아야 한다. 예를 들면

@PostMapping("/receive")

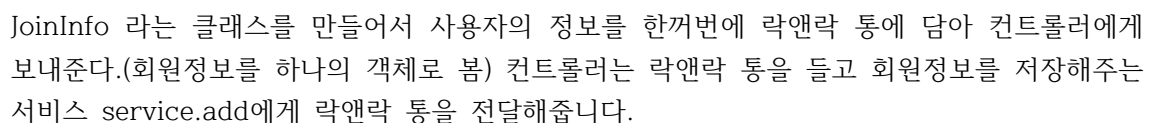
public String age(@RequestParam

String name,String password,String phone,String nickname) {

return "login";

●JoinInfo클래스 만들기 get,setter 둘다 다 만들어주세요!

●joinController



해당 커리문을 통해서 사용자가 입력한 정보를 데이터베이스에 저장을 하게 됩니다.

마지막으로 return값에 “login”을 적어서 회원정보를 다오에 저장 후 로그인 페이지 (login.jsp)를 사용자에게 를 보여주게 합니다.

```

14
15 @Repository
16 public class JoinAndLoginDao implements IJoinAndLoginDao {
17
18
19     @Autowired
20     private JdbcTemplate jdbcTemplate;
21
22
23
24     @Override
25     public int add(JoinInfo u) {
26         String query = "INSERT INTO jointable(userid,password,name,phone,nickname,gender,checkboxbox,othertravel)"
27             + "VALUES(?,?,?, ?,?, ?, ?)";
28         return jdbcTemplate.update(query, u.getUserId(), u.getPassword(), u.getName(), u.getPhone()
29             , u.getNickName(), u.getGender(), u.getCheckBox(), u.getOtherTravel());
30     }
31

```


5. modelAttribute 사용 주의점.

1)먼저 폼을 하나 더 만들어주고 input앞에 form폼을 하나 더 만들어 줍니다.

```
<div class="col-lg-4">
  <div class="jumbotron" style="padding-top: 20px;">
    <form:form modelAttribute="joinInfo" method="post" action="/paths/join" onsubmit="return checkit()">
      <h3 style="text-align: center;">회원가입화면</h3>

      <div class="form-group id_div">
        <div class="col-auto id_input">
          <form:input type="text" class="form-control" placeholder="아이디"
            id="user" name="userID" path="userId" maxlength="20" />
          <form:errors path="userId" />
        </div>
      </div>
    </form>
  </div>
</div>
```

2)HTML에 있는 path="userId"와 JoinInfo String userId 이름이 같아야함.

```
<div class="form-group id_div">
  <div class="col-auto id_input">
    <form:input type="text" class="form-control" placeholder="아이디"
      id="user" name="userID" path="userId" maxlength="20" />
    <form:errors path="userId" />
  </div>
  <div class="col-auto id_button">
    <button class="btn btn-secondary" id="duplicate_check"
      type="button" >중복체크</button>
  </div>
</div>
```

```
public class JoinInfo {
  private String userId;
  private String password;
  private String passwordCheck;
  private String name;
  private String phone;
  private String nickName;
  private String gender;
  private String checkBox;
  private String otherTravel;
  public JoinInfo() {
    super();
  }
}
```

이름이 동일 해야 함

※만약 이와 같이 path="userid"와 JoinInfo String userId 동일하지 않으면 아래와 같은 오류메시지가 뜨게 된다. 그럼 꼭 html과 JoinInfo의 변수 이름들을 확인하기 바랍니다.

```
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
979:
980:
981:
982:
983:
984:
985:
986:
987:
988:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
1000:
```

●joinController

```
@ModelAttribute
public JoinInfo joinInfo() {
  JoinInfo user = new JoinInfo();
  return user;
}
```

@ModelAttribute 어노테이션을 등록을 하고 JoinInfo를 등록을 해준다.

```
@PostMapping("/join")
public String add(@ModelAttribute @Valid JoinInfo user, BindingResult result) {
```

add메소드 입력타입에 @ModelAttribute JoinInfo user를 쓰게되면 JoinInfo에 회원정보 객체를 사용할 수 있게 됩니다.

6. 사용자가 입력창을 다 입력하지 않고 회원가입 버튼을 눌렀을 때.

사용자가 회원가입 창에서 회원정보를 다 입력하지 않고 회원가입 버튼을 누르는 경우가 있을 수 있습니다. 이런 경우에는 alert 창을 보여주는 방법이 있습니다.

localhost:8080 내용:
이름을 입력해주세요.

확인

...

이름

010-4848-4848

도로시

닉네임 중복체크

남자 여자

최근여행지는?
☐ 서울 ☐ 부산 ☐ 대구 ☒ 경주 ☐ 울산 ☐ 전라남도 ☐
☐ 하동 ☐ 제주도 ☐ 창원 ☐ 지리산

기타여행지

회원가입

1. input태크 안에 있는 패스워드,패스워드 확인, 닉네임 id="값"을 먼저 확인을 합니다.
2. <script></script>를 활용하여 회원가입 버튼을 클릭했을 때 이벤트가 발생할 수 있도록 자바스크립트를 이용합니다.
3. function checkit() 은 버튼은 onsubmit 이벤트를 처리를 활용을 하였습니다.
4. \$("pass").val()와 비밀번호 input태크 안에 있는 id="pass"값이 동일해야함. 사용자가 비밀번호를 입력을 하고 회원가입 버튼을 누르게 되면 사용자가 입력한 비밀번호 값이 \$("#pass")에 담기게 됩니다. 그 값을 password에 담아 if문으로 사용하여 값이 있는지 없는지를 확인하여 alert창을 띄어주면 됩니다. 항상 input 태그에 있는 id="값"을 가져올 때는 #을 꼭 활용해야 하는 점을 알아두면 좋을 것 같습니다.

```
<script>
function checkit(){
    let password = $("#pass").val();
    let passwords = $("#passwordcheck").val();
    let name = $("#names").val();

    if( password == null || password == "" ){
        alert("비밀번호를 입력해주세요.");
        console.log("확인 해주세요 " + password);
    }

    return false;
    }else if(passwords == null || passwords == ""){
        alert("확인비밀번호를 입력해주세요.");
    }

    return false;
    }else if(name == null || name == ""){
        alert("이름을 입력해주세요.");
    }
}
```

<div class="form-group">
<form:input type="password" class="form-control"
id= "pass" placeholder="비밀번호" name="userPassword" path="password"
maxlength="20" />
<form:errors path="password" />
</div>
<div class="form-group">
<form:input type="password" class="form-control"
placeholder="비밀번호 확인" name="userPasswordCheck"
id = "passwordcheck" path="passwordCheck" maxlength="20" />
<form:errors path="passwordCheck" />
</div>
<div class="form-group">
<form:input type="text" class="form-control" placeholder="이름"
id = "names" name="userName" path="name" maxlength="20" />
<form:errors path="name" />
</div></div>

7. 아이디 중복체크 버튼 만들기 (jquery 사용)

개인적으로 jquery를 사용하는 부분이 많이 어려웠다. 구글링을 해서 이곳저곳을 찾아보았지만, error 메시지도 읽는 부분도 어렵고 어디서 error가 발생했는지 찾기가 어려웠기 때문이다.

jquery를 사용하면 장점이 있다고 합니다.

1. 해당 버튼에 대해서만 이벤트 처리를 할 수 있습니다.

저의 경우에는 join.jsp에 1)회원가입 버튼, 2)아이디 중복버튼, 3)닉네임 중복버튼

총 3가지의 버튼이 있는데 이 버튼들이 각각 따로 이벤트 처리를 해야 하는데 이때 도움을 준 것이 jquery입니다.

2. 호환성이 뛰어나 거의 모든 웹브라우저에 사용할 수 있다고 합니다.



1) 중복체크 버튼을 만든다. (jquery를 사용 방법)

```
<div class="col-auto id_button">
  <button class="btn btn-secondary" id="duplicate_check"
    type="button">중복체크</button>
</div>
```

2) 사용자가 아이디 값을 입력을 하고 중복체크 버튼을 눌리게 되면 입력된 값을 가져온다.

중복체크 버튼의 id값은 "duplicate check"이다. id값을 통해 서버는 아이디 중복체크를 알아듣고 input 태그에 있는 아이디 값을 가져옵니다. var id = \$("#user")는 아이디 input 태그에서의 id 값입니다.

```
var clicks = 1;
$("#duplicate_check").click(function(){
  var id = $("#user").val();
  name, "id"
  if(id == ""){
    alert("아이디를 입력해주세요.");
  }
});
```

3)jquery를 작성

1) var id = \$("user").val();

사용자가 입력한 아이디 값을 var id에 담아 ajax에 넘겨줍니다.

2)IdCheckController를 만들어 줍니다.

3)ajax의 url 주소는 Controller의 RequestMapping.PostMapping 주소와 동일하게 만들어 줍니다.

4)type: "POST"

5)dataType: 'text'

6)contentType: 'text/plain; charset=utf-8;'

```
console.log(id);
$.ajax({
    url : '/paths/ID_Check',
    type : 'POST',
    dataType : 'text',
    contentType : 'text/plain; charset=utf-8;',

    data : id,
    success : function(data) {
        console.log(data);
        if (data == 0) {
            idS = data;
            console.log("아이디가 없음");
            alert("사용하실 수 있는 아이디입니다.");
            if(idS == 0 && nickS == 0){
                $("#btn").attr('disabled', false);
            }
        }
    }
});
```

```
IdCheckController.java
1 package kr.co.greenart.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Controller
6 @RequestMapping("/paths")
7 public class IdCheckController {
8
9     @Autowired
10     private IJoinAndLoginService service;
11
12     @PostMapping("/ID_Check")
13     public @ResponseBody String ID_Check(@RequestBody String ID) {
14         //public 앞에 @ResponseBody 적은 이유는 @ResponseBody를 사용하지 않을 때는 생략 가능
15         //return "-1"을 경로로 인식하고 -1 view를 반환
16         String ID = paramData.trim();
17         System.out.println(ID);
18         int dto = service.Idcheck(ID);
19         System.out.println(dto);
20     }
21 }
```

4) IdController가 ajax로부터 받은 아이디값을 다오를 통해 처리.

결과 사용자가 입력한 아이디 값(asdf)은 ajax를 통해 Controller에게 넘어갑니다.

@RequestBody을 통해 아이디 값(asdf)을 받아 String ID 변수 통에 담아 둡니다.(실제는 참조) ID값을 service.Idcheck()에 넘겨줍니다. 다오에서 같은 아이디가 있는지 확인을 한 뒤 중복된 아이디가 있으면 1, 없으면 2라는 결과 값을 받습니다.

```
//아이디 중복값 찾기
@Override
public int IdCheck( String user) {

    return jdbcTemplate.queryForObject("select count(*) from jointable where userid = ?",
        , int.class, user);

}
```

5) 다오에서 처리한 값을 다시 join.jsp로 넘겨주기.

if문을 통해 return을 해주는데 이전과 다르게 뷰페이지를 return 해주지 않는 모습을 볼 수 있습니다.

IdCheckController에서 return값을 “.jsp”로 주는 것이 아닌 값으로 전달을 해야지 join.jsp에서 그 값을 받아 if문을 통해 처리를 할 수 있습니다.

이전까지는 Controller에서 리턴 값을 주게 되면 뷰 페이지로 다 이동하는 것으로 처리가 되었습니다.

```
@Autowired
private IJoinAndLoginService service;

@PostMapping("/ID_Check")
public @ResponseBody String ID_Check(@RequestBody String paramData ) {
    //public 앞에 @ResponseBody 적은 이유는 @ResponseBody 를 적지 않으면
    //return "-1"을 경로로 인식을 하고 -1 view를 찾기 때문에 꼭 @ResponseBody 적어줘서 값을 넘겨줘야 함.
    String ID = paramData.trim();
    System.out.println(ID);
    int dto = service.Idcheck(ID);
    System.out.println(dto);

    if(dto == 1) {
        return "-1";
    }else {
        return "0";
    }
}
```

하지만 값을 전달해주고 싶은 경우

```
console.log(id);
$.ajax({
    url : '/paths/ID_Check',
    type : 'POST',
    dataType : 'text',
    contentType : 'text/plain; charset=utf-8;',

    data : id,
    success : function(data) {
        console.log(data);
        if (data == 0) {
            id = data;
            console.log("아이디가 없음");
            alert("사용하실 수 있는 아이디입니다.");
            if(id == 0 && nickS == 0){
                $("#btn").attr('disabled', false);
            }
        }
    }
});
```

```
@PostMapping("/ID_Check")
public @ResponseBody String ID_Check(@RequestBody String paramData ) {
    //public 앞에 @ResponseBody 적은 이유는 @ResponseBody 를 적지 않으면
    //return "-1"을 경로로 인식을 하고 -1 view를 찾기 때문에 꼭 @ResponseBody 적어줘서 값을 넘겨줘야 함.
    String ID = paramData.trim();
    System.out.println(ID);
    int dto = service.Idcheck(ID);
    System.out.println(dto);

    if(dto == 1) {
        return "-1";
    }else {
        return "0";
    }
}
```

public 뒤에 @ResponseBody를 붙여 줍니다. 그러면 return값이 뷰페이지가 아닌 값으로 join.jsp에 전달이 됩니다.

```
@PostMapping("/ID_Check")
public @ResponseBody String ID_Check(@RequestBody String paramData ) {
    //public 앞에 @ResponseBody 적은 이유는 @ResponseBody 를 적지 않으면
    //return "-1"을 경로로 인식을 하고 -1 view를 찾기 때문에 꼭 @ResponseBody 적어줘서 값을 넘겨줘야 함.
    String ID = paramData.trim();
    System.out.println(ID);
    int dto = service.Idcheck(ID);
    System.out.println(dto);
}
```


8. 아이디 중복체크와 닉네임 중복체크를 클릭해야만 회원가입 버튼이 활성화

회원가입 버튼이 항상 활성화가 되어 있는 경우 사용자가 아이디, 닉네임 중복체크를 하지 않고 회원가입 버튼을 누리는 경우를 생각해 보았습니다. 저라도 충분히 까먹고 중복체크를 하지 회원가입 버튼 누른 경험이 있기에 이런 경우를 어떻게 해결해야 할지 고민을 하고 구글링을 해 보았습니다.

회원가입화면

아이디 중복체크

비밀번호

비밀번호 확인

이름

핸드폰 번호

닉네임

닉네임 중복체크

남자 여자

최근여행지는?

☐ 서울 ☐ 부산 ☐ 대구 ☐ 경주 ☐ 울산 ☐ 전라남도 ☐ 하동 ☐ 제주도 ☐ 창원 ☐ 지리산

기타여행지

회원가입

회원가입화면

moindoen 중복체크

비밀번호

비밀번호 확인

이름

핸드폰 번호

jabpar

닉네임 중복체크

남자 여자

최근여행지는?

☐ 서울 ☐ 부산 ☐ 대구 ☐ 경주 ☐ 울산 ☐ 전라남도 ☐ 하동 ☐ 제주도 ☐ 창원 ☐ 지리산

기타여행지

회원가입

1)onsubmit을 활용하기

1. action뒤에 onsubmit을 만들어준다. onsubmit을 만들어주면 form데이터가 전송되기 전 일정한 조건이 충족되지 않으면 전달하는 것을 막아준다.
2. submit버튼에서 disabled 작성하여 버튼을 비활성화 해준다.

```
<form:form modelAttribute="joinInfo" method="post" action="/paths/join" onsubmit="return checkit()">
  <h3 style="text-align: center;">회원가입화면</h3>

  <div class="form-group id_div">
    <div class="col-auto id_input">
      <form:input type="text" class="form-control" placeholder="아이디"
        id="user" name="userId" path="userId" maxlength="20" />
      <form:errors path="userId" />
    </div>

    <input type="submit" id="btn" class="btn btn-primary form-control"
      value="회원가입" disabled />
  </form:form>
..
```

3.제이커리 속성추가

속성과 값을 설정하는 경우는 \$("#btn").attr(attribute,value)

데이터베이스에서 아이디,닉네임을 검색을 하여 중복된 값이 없을 때 해당 if문을 통해서 버튼을 활성화 해줍니다.

그결과 사용자는 아이디 중복체크와 닉네임중복체크를 해야지만 회원가입 버튼이 활성화가 되고 가입을 할 수 있는 조건을 만들게 되었습니다.

```
<input type="submit" id="btn" class="btn btn-primary form-control"
      value="회원가입" disabled />
/form:form>
'>
```

```
console.log("아이디가 없음");
alert("사용하실 수 있는 아이디입니다.");
if(ids == 0 && nicks == 0){
    $("#btn").attr('disabled', false);
}
```

이로써 회원가입은 끝입니다. 감사합니다.

#로그인하기

2.로그인 part

- 로그인창에서 로그인하기
- 세션이용을 하여 사용자 개별 데이터 담기
- 아이디 찾기 및 비밀번호 찾기 서비스 제공
- 로그인을 해야지만 게시글을 볼 수 있게 하는 서비스 제공(인터셉터 활용)

1. 로그인창에서 로그인하기

1)로그인을 할 때 사용자가 작성해야하는 내용.

<html 타입>

○아이디/input type ="text"

○패스워드/input type ="password"

```
<form:form modelAttribute="loginInfo" method="post" action="/mapping/Login">
  <h3 style="text-align: center;">로그인화면</h3>
  <div class="form-group">
    <form:input type="text" class="form-control" placeholder="아이디"
      id = "userId" name="userID" path="userId" maxlength="20"/>
  </div>
  <div class="form-group">
    <form:input type="password" class="form-control"
      id = "userPw" placeholder="비밀번호" name="userpassword" path="password" maxlength="20"/>
  </div>
  <input type="submit" id = "btnLogin" class="btn btn-primary form-control"
    value="로그인">
  <li><a href="/mapping/idlook">아이디 찾기</a></li>
  <li><a href="/mapping/password">비밀번호 찾기</a></li>
</form>
```

2)LoginInfo.java 만들기 (사용자의 아이디, 비밀번호를 담을 통)

```
1 package kr.co.greenart.model;
2
3 public class LoginInfo {
4
5     private int id;
6     private String userId;
7     private String password;
8     private String name;
9     private String nickname;
10
11
12     public LoginInfo() {
13
14     }
15
16
17     public LoginInfo(int id, String userId, String password, String name, String nickname) {
18         super();
19         this.id = id;
20         this.userId = userId;
21         this.password = password;
22         this.name = name;
23         this.nickname = nickname;
24     }
25
26
27     public LoginInfo(String userId, String name, String nickname) {
28         super();
29         this.userId = userId;
30         this.name = name;
```

3)modelAttribute을 이용하기



회원가입 경우 JoinInfo.java를 만들어서 회원정보를 담아 한꺼번에 컨트롤러에게 전달을 하였습니다. 로그인도 마찬가지로 사용자가 입력한 로그인 정보인 id,password를 LoginInfo에 담아 컨트롤러에게 한꺼번에 전달을 하였습니다.

●LoginContrller

LoginInfo user 값을 service.loginCheck()에 넘겨줍니다. LoginInfo에는 사용자가 입력한 아이디:asdf 비밀번호:123 이 담겨 있습니다. 해당 값을 service.loginCheck()넘겨줍니다. JoinAndLoginDao에서 아이디와 비밀번호를 검색한 뒤 해당하는 값이 있으면 true로 반환해 줍니다. 그럼 반환된 true값으로 if문을 작성하여 Main.jsp 페이지로 이동합니다.



2. 세션이용을 하여 사용자 개별 데이터 담기

session을 활용하는 이유

- 1) 사용자마다의 개별 데이터를 저장하는 공간 활용
- 2) 로그인 상태를 유지하는 기능을 수행
- 3) 일정시간이 되면 세션이 만료되어 데이터 자동적으로 삭제

●JoinAndLoginService.java

LoginController에서 HttpSession을 이용하여 session -> service.loginCheck에 넘겨줍니다. **JoinAndLoginService**에서 id:user,password:123 값이 담긴 user를 viewMember에 넘겨줍니다. **JoinAndLoginDao**에서 아이디와 비밀번호를 다시 한 번 확인을 하고 아이디,이름,닉네임을 가지고 LoginInfo에 담아 vo2 담아놓습니다.

vo2는 아이디,이름,닉네임을 가지고 있습니다.(실제는 참조 변수로써 아이디,닉네임,이름의 인스턴스 주소 값을 가르키고 있습니다.) 그럼 이제 session을 이용하여 아이디,닉네임,이름을 활용합니다.

```
//로그인 처리
@PostMapping("/login")
public ModelAndView add(@ModelAttribute LoginInfo user, HttpSession session) {
    boolean results = service.loginCheck(user, session);
    System.out.println("정말 두르가 나왔까? " + results);
    ModelAndView mav = new ModelAndView();
    logger.info(user.toString());

//로그인 체크
@Override
public boolean loginCheck(LoginInfo user, HttpSession session) {
    boolean result = dao.loginCheck(user);
    System.out.println("정말 두르가 나왔까?!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! " + result);
    if(result) {
        System.out.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
        LoginInfo vo2 = viewMember(user);
        System.out.println("이거 맞아라!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! " + vo2.toString());

//로그인 정보
@Override
public LoginInfo viewMember(LoginInfo vo) {
    System.out.println("백여과!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! " + vo.toString());
    return dao.viewMember(vo);
}

//회원 정보
@Override
public LoginInfo viewMember(LoginInfo user) {
    return jdbcOssUtil.queryForObject("select * from jointable where userid = ? and password = ?",
        new RowMapper<LoginInfo>() {
            @Override
            public LoginInfo mapRow(ResultSet rs, int rowNum) throws SQLException {
                return new LoginInfo(rs.getString("userId"), rs.getString("name"), rs.getString("nickname"));
            }
        }, user.getUserId(), user.getPassword());
}
}
```

●JoinAndLoginService.java

setAttribute(String name, Object value) 메서드를 활용하여 이름과 객체를 저장을 합니다.

```
LoginInfo vo2 = viewMember(user);
System.out.println("이거 맞아라!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! " + vo2.toString());
session.setAttribute("userId", vo2.getUserId());
session.setAttribute("userName", vo2.getName());
session.setAttribute("nickName", vo2.getNickname());
pnickNickName = vo2.getNickname();
pnickname = vo2.getName();
System.out.println("pnickname 값을 알려주세요 " + pnickname);
```


●main.jsp(메인화면)

jsp에서 if문을 편하게 쓰기 위해서 jstl을 사용하였다. 사용하기 위해서는 아래와 같이 prefix를 쓰고 싶은 알파벳을 주고 uri주소를 아래와 같이 사용하면 됩니다.

그 결과 <c:if>문을 쓸 수 있다. if말고도 다양한 when, forEach 등을 사용할 수 있으니 인터넷에 jstl core 사용방법을 검색해보면 좋을 것 같습니다.

1. \${msg == 'success'} 조건문 같은 경우에는 ●LoginContrller에서 확인을 해보면 됩니다.

ModelAndView를 사용하여 mav.setViewName('main') 화면으로 이동을 하게 됩니다.

mav.addObject('msg','success') msg를 key값으로 하여 success를 value값으로 지정하였습니다.

2.if문을 통과를 하고 \${sessionScope.userName}을 적어줍니다. userName의 경우에는

●JoinAndLoginService.java 에서 session.setAttribute의 key 값이고

사용자에게는 userName의 value값이 보여 지게 됩니다.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.io.PrintWriter"%>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
5 <!DOCTYPE html>
6 <html>
7 <head>
8 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
9 <meta name="viewport" content="width=device, initial-scale=1">
10 <link rel="stylesheet" href="/resource/css/bootstrap.min.css">
11 <title>여행정보사이트</title>

<ul class="nav navbar-nav navbar-right">
  <li style="padding-top:15px" >
    <c:if test = "${msg == 'success'}">
      <h8 ${sessionScope.userName }(${sessionScope.userId})님 환영합니다.</h8>
    </c:if>
  </li>
  <li class="dropdown"><a href="#" class="dropddown-toggle"
    data-toggle="dropdown" role="button" aria-haspopup="true"
    aria-expanded="false">${userId}<span class="caret"></span></a>
    <ul class="dropdown-menu">
      <li><a href="/out/Logout">로그아웃</a></li>
    </ul></li>
  </ul>
```

그 결과 아래와 같이 닉네임과 아이디가 보이게 됩니다.

3. 로그아웃 기능 만들기.

1. LogoutController를 만들어 줍니다.

로그아웃 컨트롤러의 주소는 /out/logout으로 만들어줍니다.

2. 세션을 종료하는 메서드를 만들어줍니다.

invalidate()를 활용하여 session을 종료 해줍니다.

3. redirect를 활용하여 종료가 되었다는 것을 알려줍니다.

redirectAttributes 클래스를 사용하여 key=msg, value='logout'을 값을 login.jsp에 전달을 합니다.

저의 경우에는 로그아웃 기능을 활용하기 위해

request와response 값이 새롭게 만들어지는 redirect를 사용하였습니다.

4. return 값을 redirect 형식으로 만들어 줍니다.

●LoginContrller에서 섹션을 종료한 뒤 return값에 다시 돌아갈 페이지 주소를 적어줍니다.

이때 주소 값 앞에 redirect 형식으로 적어서 보내줍니다.

짧은 지식으로 redirect를 활용했던 이유를 설명을 드리면 기존의 로그인 페이지가 아닌 새로운 페이지를 요청하기 위해서 redirect를 활용하였습니다.

구글에 redirect와 forward의 차이점을 한번 검색을 해보시면 좋을 것 같습니다.

5. EL태그를 이용하여 결과 값을 사용자에게 보여줍니다.

```
<?import javax.servlet.http.HttpSession;

@Controller *LogoutController
@RequestMapping("/out")
public class LogoutController {

    @Autowired
    private IJoinAndLoginService service;

    @GetMapping("/logout")
    public String logout(HttpSession session, RedirectAttributes redirectAttributes) {

        service.logout(session);
        redirectAttributes.addFlashAttribute("msg", "logout");
        return "redirect:/mapping/login";
    }
}

//로그아웃 서비스
@Override *JoinAndLoginService
public void logout(HttpSession session) {
    session.invalidate();
}

</out>

</c:if> *login.jsp
<c:if test = "${msg == 'logout'}">
    <div style = "color: red">
        로그아웃이 되었습니다.
    </div>
</c:if>
```

※사용자에게 보여지는 로그아웃된 페이지 결과창.

로그인화면

- 아이디 찾기
- 비밀번호 찾기

로그아웃이 되었습니다.

4. 아이디 찾기 및 비밀번호 찾기 서비스 제공

사용자가 아이디와 비밀번호가 생각나지 않는 경우가 있습니다. 그럴 경우를 대비하여

1)아이디찾기, 2)비밀번호찾기 서비스를 만들었습니다.

아이디찾기와 비밀번호 찾기 로직은 비슷하니 아이디 찾기를 대표해서 작성하겠습니다.

로그인화면

- 아이디 찾기
- 비밀번호 찾기

1.아이디 찾기를 누르게 되면 아이디를 찾을 수 있는 뷰 페이지(ID.jsp)를 만듭니다.

아이디찾기

2.사용자가 입력한 이름값, 핸드폰 번호 값을 가지고 데이터베이스에서 일치하는 값이 있는지 찾는다.

첫 번째 조건)

핸드폰 값은 11자리 숫자이기 때문에

1)11자리 이상 숫자까지 입력 불가능하게 만들기

html에서 maxlength를 이용하여 11자리 이상 입력을 하지 못하도록 만들었습니다.

```
<div class="form-group">
  <input type="text" class="form-control" id="phone" placeholder="핸드폰번호(-)하이픈을빼고 입력하세요" name="userpassword" maxlength="11"/>
</div>
```

2) 11자리 이하 입력하게 입력 불가능하게 만들기
제이커리를 사용하여 사용자가 11자리 미만으로

```
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        var username = $("#userName").val();
        var userPhone = $("#phone").val();
        // 11자리 이하 입력하게 입력 불가능하게 만들기
    }else{
        if(userPhone.length < 10){
            alert("핸드폰번호를 11자리까지 입력해주세요.");
        }
    }
}
```

사용자가 입력한 핸드폰 번호는 01048485656

데이터베이스 저장된 핸드폰 번호는 010-4848-5656

이므로 사용자가 입력한 핸드폰 번호에 하이픈(-) 추가해서 DB에 넘겨줘야 한다.

아래와 같이 rephone 메서드를 만들어 사용자가 입력한 핸드폰 번호를 입력받습니다.

01048485050이라는 핸드폰번호를 substring으로 자르고 나서 "-" 붙여 합쳐서 출력하는
형식으로 하였습니다.

그 결과 DB에는 010-4848-5050으로 조회가 됩니다.

```
private String rephone(String phoned) {
    String frist = phoned.substring(0,3);
    System.out.println("첫번째 값은 무엇인가요?" + frist );
    String seconed = phoned.substring(3,7);
    System.out.println("두번째 값은 무엇인가요?" + seconed );
    String third = phoned.substring(7,11);
    System.out.println("세번째 값은 무엇인가요?" + third );

    String phoneNumber = frist+"-"+seconed+"-"+third;

    return phoneNumber;
}
```

3. 첫 번째 조건이 만족하게 되면 phoneNumber과 유저 name을 DB에 넘겨줍니다.

```
String phoneNumbers = rephone(phones);
//이름과 핸드폰 번호를 붙여서 찾는 서비스
//이름과 핸드폰이 일치하지 않으면 결과값이 "fail"이 나오게 되어 있습니다.
//이름과 핸드폰이 일치할 하면 유저의 아이디값을 "dto"에 담습니다.
String dto = service.idFind(names, phoneNumbers);
System.out.println("dot값은 무엇인가요?" + dto);
//다오에서 fail이 나오게 되면 return "fail"을해주고
if(dto == "fail") {
    return "fail";
}
}else {
    //다오에서 유저와 아이디 값이 나오게 되면 그대로 아이디값을 전달함.
    return dto;
}
```

4. 첫 번째 조건이 만족하게 되면 phoneNumber과 유저 name을 DB에 넘겨줍니다.

●JoinAndLoginDao

```
// 아이디 찾기
@Override
public String idFind(String name, String phone) {
    try {
        return jdbcTemplate.queryForObject("select userid from jointable where name = ? and phone = ?",
            String.class, name, phone);
    } catch (DataAccessException e) {
    }
    return "fail";
}
```

유저 name과 핸드폰 번호를 입력받아 DB에서 결과 값을 찾게 됩니다.

1) 일치하는 결과 값이 있는 경우 -> 유저 아이디 출력

2) 일치하는 결과 값이 없는 경우 -> "fail"이 출력

```
//다오에서 fail이 나오게 되면 return "fail"을해주고
if(dto == "fail") {
    return "fail";
} else {
    //다오에서 유저의 아이디 값이 나오게 되면 그대로 아이디값을 전달함.
    return "dto";
}
```

5. ●IdlookController로 부터 받은 결과 값으로 사용자에게 조건에 맞는 alert창 보여주기.

출력된 값은 success:functon(data1)에 있는 data1에 남기게 됩니다.

data1에 있는 값을 가지고 if문으로 만들어 조건에 맞는 alert창을 사용자에게 보여줍니다.

```
success: function(data1) {
    //IdlookController로 부터 받은 결과 값은 functon(data1)에 남기게 됩니다.
    //그 결과 값을 아래의 if문으로 통해 조건식을 만들어 사용자에게 해당하는 alert 창을 보여주게 됩니다.
    if (data1 == "fail") {
        alert("아이디가 존재하지 않습니다.");
    } else if(data1 == "0"){

    } else{
        alert("회원님의 아이디는" + data1)
    }
}
```

5. 주의할 점.

저의 경우에는 사용자 아이디 찾기 서비스에서 1)이름을 입력하지 않는 경우 2)핸드폰 번호를 입력하지 않는 경우에는 submit 버튼을 눌러도 폼이 전송되지 않게 onsubmit을 사용하였습니다.

```
<form id= "idLook" method="post" action="/Id/IdLook" onsubmit = "return checkit()">
    <h3 style="text-align: center;">아이디찾기</h3>
```

onsubmit에 return을 꼭 적어주고 checkit()을 적어주게 되면 이벤트를 발생시킬 수 있습니다. 아래의 주석처럼 return false를 만나게 되면 컨트롤러에게 폼을 전송하지 않고 이름을 입력해주세요 라는 알림창만 띄어주는 형태로 됩니다.

```
<script>
function checkit(){
    if(username == "" || username == null){
        alert("이름을 입력해주세요!");
        return false;
        //onsubmit의 경우에는 return false를 만나게 되면
        //사용자가 입력한 폼을 컨트롤러에게 전송하지 않고
        //그대로 멈춰 있게 한다.
    }
}
```


5. 로그인을 해야지만 게시글을 볼 수 있게 하는 서비스 제공(인터셉터 활용)

인터셉터(Interceptor)란?

Interceptor란 컨트롤러에 들어오는 요청 HttpRequest와 컨트롤러가 응답하는 HttpResponse를 가로채는 역할을 한다고 합니다.

저의 경우에는 로그인을 하지 않고 게시글을 볼 수 있었는데, 그러한 부분을 막고 로그인을 해야지만 게시글을 볼 수 있게 만들고 싶어 인터셉터를 이용하게 되었습니다.

인터셉터는 필터와 다르게 DispatcherServlet이 실행이 된 후 실행이 되는 점을 꼭 알아두시면 좋을 것 같습니다.

※사용방법

1)HandlerInterceptor 클래스를 상속받습니다.

2)preHandle 메서드를 꺼내옵니다.

3)getSession(false)로 하여 session 값이 있는 경우 true로 할 수 있도록 만듭니다.

4)session.getAttribute("userId")에서 userId는 로그인 세션을 만드는 당시 key 값입니다.

5)사용자가 로그인 상태라면 세션이 있으니 true 가 반환되고/ 그게 아니라면 Redirect를 이용하여 login주소로 이동을 시킵니다.

6)WebConfig에서 FindAllInterceptor를 사용할 수 있도록 추가를 한 뒤 addInterceptors 메서드를 추가해줍니다.

7)addPathPatterns을 통해 인터셉터를 적용할 페이지 주소를 적어줍니다.

```
@Component
public class FindAllInterceptor implements HandlerInterceptor {

    //각각시간에 별다른 코딩은
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
        throws Exception {
        HttpSession session = request.getSession(false);

        if(session.getAttribute("userId") != null) {
            return true;
        } else {
            response.sendRedirect("/mapping/login");
            return false;
        }
    }
}

public class WebConfig implements WebMvcConfigurer {

    @Autowired
    private FindAllInterceptor interceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(interceptor).addPathPatterns("/BBS/bbsView", "/my/mypage");
    }
}
```

실행을 해보시면 로그인을 하지 않은 상태에서는 게시판이나 마이페이지를 볼 수 없는 결과를 볼 수 있습니다. 이로써 로그인 part는 끝이 났습니다.

처음 만들어보는 게시판으로 부족한 점이 많으니 넓은 마음으로 이해를 해주시면서 참고해주시면 감사하겠습니다.

#게시판 만들기 part1

주제

- 게시판 양식 만들기
- 글을 작성 후 게시판에 추가해보기
- 게시판에서 추가한 글 보기
- 게시판 글을 수정하기
- 게시판 글을 삭제하기

1.게시판 양식 만들기.

1)게시판 틀 양식

- 게시판번호- int id
 - 제목- String title
 - 작성자- String nicName
 - 작성일- String writeDate
 - 조회수- int count
- ※mysql 스키마-myweb,테이블이름-writetable, pk-id

Cloumn Name	id	nick Name	title	location	date	write Date	text	start
한글 표현	게시판 번호	작성자	제목	여행지역	여행 날짜	작성일	여행 내용	평점
Data type	Int	VAR CHAR (45)	VAR CHAR (45)	VAR CHAR (45)	DATE	DATE	TEXT	DOBLE

※ListInfo 객체(사용자가 쓴 글 내용을 담는 객체)

사용목적- 사용자가 글쓰기 버튼을 누른 후 글 양식에 맞게 작성한 내용을 객체로 만들어서 DB에 저장 및 다시 가져올 때 한번 에 가져오기 위해 만들게 되었습니다.

```
private int id;
private String nicName;
private String title;
private String location;
private String date;
private String writeDate;
private String text;
private int star;
private int count;
```

사용자에게 보여줄 Tab 내용은 번호, 제목, 작성자, 작성일, 조회 수 로 정하였습니다.

번호	제목	작성자	작성일	조회수
127	dddddsdfsafasfdffff	아이리스	2021-11-10	10
126	asdfsdfasdfsdfasdfsdf	아이리스	2021-11-09	2

●bbs.jsp

먼저 사용자에게 보여줄 번호, 제목, 작성자, 작성일, 조회 수
들을 bbs.jsp에 만들어 줍니다.

```
<tr>
  <th style="background-color: #eeeeee; text-align: center;">번호</th>
  <th style="background-color: #eeeeee; text-align: center;">제목</th>
  <th style="background-color: #eeeeee; text-align: center;">작성자</th>
  <th style="background-color: #eeeeee; text-align: center;">작성일</th>
  <th style="background-color: #eeeeee; text-align: center;">조회수</th>
</tr>
```

3)글쓰기 버튼 만들기.

글쓰기 버튼을 누르게 되면 글쓰기 양식 페이지로 가는 버튼을 만들어 줍니다.

저는 a태그를 활용하여 글쓰기 양식이 있는 /write/writeView 페이지 주소로 이동하게 하였습니다.

```
<a href="/write/writeView" class="btn btn-primary pull-right">글쓰기</a>
```

4)여행 후기 쓰는 양식.

게시판 글쓰기 양식	
제목	
대행 지역	
연도-월-일	

<html 타임>

- 제목/input type = "text"
- 여행지역/input type = "text"
- 여행날짜/input type = "date"
- 내용/textarea CKEDITOR 사용
- 평점/input type = "number"

- write.jsp

```
<tr>
  <td><form:input type="text" class="form-control"
    placeholder="제목" name="title" path="title" maxlength="50"/><form:errors path="title"/></td>
</tr>
<tr>
  <td><form:input type="text" class="form-control"
    placeholder="여행 지역" name="location" path="location" maxlength="50"/><form:errors path="location"/></td>
</tr>
<tr>
  <td><form:input type="date" class="form-control"
    placeholder="여행 날짜" name="date" path="date" maxlength="50"/></td>
</tr>
<tr>
  <td><form:textarea id = "editor4" name = "editor4" path="text"/></form:errors path="text"/></td>
</tr>
<tr>
  <td><form:input type="number" step="0.1" class="form-control"
    placeholder="평점" name="bbsTitle" path="star" maxlength="50"/></td>
</tr>
```

2.사용자가 게시물을 작성하게 되면 내용을 DB에 저장하기.

사용자가 입력한 내용

- 제목/수원여행을 다녀오다
- 여행지역/수원
- 여행날짜/2021-12-03
- 내용/즐거웠던 수원여행이었다.
- 평점/1점.

수원여행을 다녀오다

수원

2021-12-03

✂

📄

📁

📁

📁

↶

↷

🔍

🔍

🚩

🖼️

📅

📅

📅

🔍

🔍

🔍

소스

B

I

S

I_x

☰

☰

☰

☰

☰


☰

스타일

☑

☑

?



즐거웠던 수원여행이었다.

body p

1

글쓰기

DB에 저장되는 내용

- 닉네임/아이리스(추가되는 부분)
- 제목/수원여행을 다녀오다
- 여행지역/수원
- 여행날짜/2021-12-03
- 작성날짜/2021-12-04(추가되는 부분)
- 내용/즐거웠던 수원여행이었다.
- 평점/1점.

1)사용자가 작성한 내용은 제목/여행지역/여행날짜/내용/평점이었지만 DB에 저장되는 경우에는 닉네임과 작성날짜가 추가가 되는 것을 볼 수가 있습니다.

●write table(DB)

id	nickName	title	location	date	writeDate	text	star
108	아이리스	asdfsadfsafsdaf	asdfsadfsafsdaf	2021-11-04	2021-11-11	asdfsadfsafsdaf	0

2)먼저 사용자가 입력한 내용을 DB에 저장하는 방법부터 작성하겠습니다.

1.write.jsp에서 form에 modelAttribute를 사용하여 사용자가 입력한 내용을 객체로 만들어줍니다.

2.입력한 내용을 객체로 만들기 위해서 WriteInfo 클래스를 만들어 줍니다.

※주의 input 타입에 적은 path="title"과 WriteInfo클래스의 String title 변수이름이 동일해야 합니다.

3.WriteController에서 ModelAttribute를 등록을 하여 사용합니다.

4.DB에 사용자가 입력하지 않은 닉네임을 추가시키기 위해 JoinAndLoginService 클래스에서 닉네임을 가져옵니다.

```
*write.jsp
<form:form
    <table class="table table-striped"
        style="text-align: center; border
    <tr>
        <td><form:input type="text" class="form-control"
            placeholder="제목" name="title" path="title" max
        </td>
    </tr>
    <tr>
        <td><form:input type="text" class="form-control"
            placeholder="여행 지역" name="location" path="Loc

*WriteInfo.java
public class WriteInfo {
    private String title;
    private String location;
    private String date;
    private String text;
    private double star;

*WriteController.java
@PostMapping("/writeView")
public String add(@ModelAttribute(name = "writeInfo") @Valid WriteInfo writeInfo, BindingResult result, Model model) {
    //로그인 부분에서 이용했던 세션 부분에서 닉네임을 static으로 만들어서 해당 컨트롤러에서 이용하였습니다.
    //pinckNickName을 활용하기 위해서 위에 보시면 Autowired로 JoinAndLoginService를 등록을 하였습니다.
    //닉네임을 가져와서 String nickName에 담아줍니다.
    String nickName = services.pnickNickName;
    System.out.println("이게 맞을 까!!!!!!!!!!!!!!!!!!!!!!" + nickName);
    logger.info(writeInfo.toString());
    //사용자가 제목, 여행날짜, 여행지역, 내용을 입력했는지 확인하는 부분입니다.
    //이중에 하나라도 입력하지 않으면 입력하라는 내용이 나오며 write.jsp로 리턴을 하여 계속 해당페이지에 머물게 합니다.
    if(result.hasErrors()) {
        return "write";
    }

    //ModelAttribute를 이용하여 사용자가 입력한 값을 writeInfo에 담아 객체를 만든다음 게시글을 등록해주는 쿼리에
    //writeInfo와 닉네임을 함께 넘겨줍니다
    int results = service.add(writeInfo, nickName);
```

작성날짜도 함께 저장을 해야 하므로 Date클래스를 사용하여 작성날짜도 함께 저장을 해줍니다.

6.결과

2.DB에 저장된 내용을 다시 사용자에게 보여주기

하지만 게시판 화면의 탭은 번호/제목/작성자/작성일/조회 수 되어있습니다.

DB에서 해당 탭 내용만 뽑아서 보여줘야 합니다.

●WriteController.java

```
//페이징을 하는 부분입니다.  
//0~10번 게시글을 보여주기 사용함  
int a = 1;  
//service.look 서비스를 통해 DB에서 게시글 번호,타이틀,작성자,작성일,조회수를 보여줍니다.  
List<ListInfo> list = service.look(a);  
//list에 정보를 담아 model을 통해 뷰로 전달을 합니다.  
model.addAttribute("list", list);
```

●WriteDao.java

bbs.jsp에 게시글 번호, 닉네임, 타이틀, 작성날짜, 조회 수 를 보여주는 쿼리문입니다.

해당 쿼리문을 살펴보면

- (1)select id,nickName,title,date,count from writetable
- (2)order by id desc limit 10 offset ?
- (3)as a left join view as b on a id = b.id

총 3가지 쿼리문 으로 나눌 수 있습니다.

(1) DB에 저장되어 있는 id, nickname, title, date, count를 bbs.jsp에 보여주는 역할을 수행합니다.

(2) 페이징을 할 때 사용하는 쿼리 입니다.

(3) left join을 사용하여 조회 수 테이블인 views 테이블을 활용하여 bbs.jsp에 조회 수를 보여주는 역할을 수행합니다.

결론- 하나의 쿼리문에 데이터 보여주는 역할, 페이징역할, views 테이블의 조회 수를 보여주는 3가지 역할을 수행한다고 보시면 됩니다.

```
//bbs.jsp에 게시글 번호, 닉네임, 타이틀, 작성날짜, 조회 수 를 보여주는 쿼리문입니다.
//해당쿼리문을 살펴보면 1) select id,nickName,title,date,count from writetable 까지와
//2)order by id desc limit 10 offset ?
//3)as a left join view as b on a id = b.id
//로하여 총 3가지 쿼리문으로 나눌 수 있습니다.
//1) DB에 저장되어 있는 id, nickname, title, date, count를 bbs.jsp에 보여주는 역할을 수행합니다.
//2) 페이징을 할 때 사용하는 쿼리 문입니다.
//3) left join을 사용하여 조회 수 테이블인 views 테이블을 활용하여 bbs.jsp에 조회 수를 보여주는 역할을 수행합니다.
//결론- 하나의 쿼리문에 데이터 보여주는 역할, 페이징역할, views 테이블의 조회수를 보여주는 3가지 역할을 수행한다고 보시면 됩니다.
@Override
public List<ListInfo> look(int page) {
    int a = (page-1)*10;
    return jdbcTemplate.query("SELECT a.id,a.nickName,a.title, a.date, b.count FROM writetable as a left join views as b on a.id = b.id "
        + "ORDER BY a.ID DESC limit 10 offset ? ",new ListInfoRowMapper(), a);
}

private class ListInfoRowMapper implements RowMapper<ListInfo>{
    @Override
    public ListInfo mapRow(ResultSet rs, int rowNum) throws SQLException {
        int id = rs.getInt("id");
        String name = rs.getString("nickName");
        String title = rs.getString("title");
        String date = rs.getString("date");
        int count = rs.getInt("count");
        return new ListInfo(id,name,title,date,count);
    }
}
```

●WriteController.java

model.addAttribute를 사용하여 DB 정보가 담긴 list를 bbs.jsp에 전달을 합니다.

bbs.jsp에서 list를 key 값으로 받아 forEach문으로 info에 value 값을 하나씩 꺼내어 EL형식으로 사용자에게 탭 내용을 보여줍니다.

```
*WriteController.java
int a = 1;
//service.look 서비스를 통해 DB에서 게시글 번호, 타이틀, 작성자, 작성일, 조회수를 보여줍니다.
List<ListInfo> list = service.look(a);
//list에 정보를 담아 model을 통해 보러 전달을 합니다.
model.addAttribute("list", list);

*bbs.jsp
<c:if test ="${not empty list}">
    <c:forEach var = "Info" items ="${list}">
        <tr>
            <td>${Info.id}</td>
            <td><a href="/view/viewMapping?bbsID=${Info.id}&count=${Info.count}">${Info.title}</a></td>
            <td>${Info.nicName}</td>
            <td>${Info.date}</td>
            <td>${Info.count}</td>
        </tr>
    </c:forEach>
</c:if>
```

●bbs.jsp(결과)

번호	제목	작성자	작성일	조회수
127	dddddsdfsasfddddd	아이리스	2021-11-10	11
126	asdfsasdfsasdfsasdfsasdf	아이리스	2021-11-09	3
125	asdfsasdf	아이리스	2021-11-09	3
124	asdf	아이리스	2021-11-08	1
118	asdfsasdf	아이리스	2021-11-08	0
117	asdfsasdfsasdf	아이리스	2021-11-04	1
116	asdfsasdfsasdf	아이리스	2021-11-04	0
115	asdfsasdfsasdf	아이리스	2021-11-04	0
114	asdfsasdfsasdf	아이리스	2021-11-04	0
113	asdfsasdfsasdf	아이리스	2021-11-04	0

3.게시판에서 추가한 글 보기

전체 게시판에서 자신이 작성한 글뿐만 아니라 다른 사람들이 작성한 글 또한 볼 수 있어야 합니다.

저의 경우에는 게시글 제목을 누르게 되면 작성한 글을 볼 수 있도록 만들었습니다.

●bbs.jsp

1)타이틀에A 태그를 추가를 합니다.

```
<tr>
  <td>${Info.id}</td>
  <td><a href="/view/viewMapping?bbsID=${Info.id}&count=${Info.count}">${Info.title}</a></td>
  <td>${Info.nicName}</td>
  <td>${Info.date}</td>
  <td>${Info.count}</td>
</tr>
```

2)a태그의 주소는 /view/viewMapping?bbsID=\${info.id}입니다.

bbsID의 value 값인 127을 ●ViewWriteController에서 @RequestParam int bbsID로 127를 받았습니다. int ID에 127을 담아 둡니다.

① localhost:8080/view/viewMapping?bbsID=127&count=12

```
@Controller
@RequestMapping("/view")
public class ViewWriteController {
    private Logger logger = LoggerFactory.getLogger(ViewWriteController.class);

    @Autowired
    private IWriteService service;

    @GetMapping(value = "/viewMapping")
    public String join(@RequestParam int bbsID, int count, Model model) {
        //게시판 아이디
        int ID = bbsID;
```

●ViewWriteController.java

- 1)@RequestParam int bbsID에서 받은 value 값인 127을 int ID에 저장을 합니다.
- 2)게시글 양식에 작성한 내용을 보여주는 service.lookId에 ID 값 127을 던져줍니다.
- 3)DB에서 ID value 값인 127로 Writetable pk인 id를 검색하여 127번 게시물의 내용들을 보여줍니다.
- 4)viewWrite.jsp로 이동을 합니다.

```
//@RequestParam int bbsID에서 받은 value 값인 127을 int ID에 저장을 합니다.\n//게시글 양식에 작성한 내용을 보여주는 service.lookId에 ID 값 127을 던져줍니다.\n//DB에서 ID value 값인 127로 Writetable pk인 id를 검색하여 127번 게시물의 내용들을 보여줍니다.\nListInfo list = service.lookId(ID);\n\n//127번 게시물의 내용들을 model을 통해 뷰로 전달을 합니다.\nmodel.addAttribute("list", list);\n\nreturn "viewWrite";
```

●viewWrite

list.nicName,list,title 등 ListInfo를 이용하여 DB값을 보여줍니다.

```
<tr>\n    <td>닉네임 :    ${list.nicName}</td>\n\n</tr>\n<tr>\n    <td>제목 :      ${list.title}</td>\n</tr>\n<tr>\n    <td>여행지역 :   ${list.location}</td>\n</tr>\n\n<tr>\n    <td>여행날짜 :   ${list.date}</td>\n\n</tr>\n<tr>\n    <td>글쓴날짜 :   ${list.writeDate}</td>\n\n</tr>\n<tr>\n    <td>내용 :       ${list.text}</td>\n\n</tr>\n<tr>\n    <td>평점 :       ${list.star}</td>\n\n</tr>
```


●viewWrite.jsp(결과)

메인

게시판

회원관리

닉군(asdf)님 환영

게시판 글쓰기 양식
닉네임 : 아이리스
제목 : 수원여행을 다녀오다!!
여행지역 : 수원
여행날짜 : 2021-11-10
글쓴날짜 : 2021-12-04
내용 : 여행 정말 재미있었다

평점 : 3

4.게시판 글을 수정하기

자신의 게시판을 수정해야하는 경우도 있기 때문에 수정 버튼을 만들어 주었습니다.

게시판 글쓰기 양식

닉네임 : 아이리스


제목 : 수원여행을 다녀오다!!

여행지역 : 수원

여행날짜 : 2021-11-10

글쓴날짜 : 2021-12-04

내용 :
여행 정말 재미있었다



평점 : 3

수정 삭제

●viewWrite.jsp

(해당 게시판 번호는 127번입니다.)

1)먼저 수정 버튼을 만들어 줍니다.

2)마우스 클릭을 하면 이벤트가 발생하게 되는 onclick을 이용하였습니다.

3)onclick에 url주소와 \${list id}(게시판 번호 127)를 작성합니다.

4)수정버튼을 클릭하게 되면 이벤트가 발생하여 onclick에 있는 주소의 컨트롤러에게 이동을 하게 됩니다.

```

:if test = "${sessionScope.nickName == list.nicName}">
  <button class="btn btn-primary pull-left" type="button" onclick="Location.href='/update/updating?bbsID=${list.id}'">수정</button>
  <button class="btn btn-primary pull-left" type="button" onclick="Location.href='/update/delete?bbsID=${list.id}'">삭제</button>
</if>

```

●UpdateController.java

1)viewWrite.jsp에서 넘어온 게시판 번호 127번을 받아야 합니다.

① localhost:8080/update/updating?bbsID=127

@RequestParam int bbID를 통해 번호 127번을 받습니다.

※URL에서 bbsID는 key값이고 127은 value 값이라고 생각을 하시면 좋을 것 같습니다.

2)127번 게시글 내용을 불러오는 service.lookId 메소드에 bbsID를 넘겨줍니다.

3)ListInfo 클래스를 활용하여 DB에서 불러온 값을 저장을 합니다.

4)model을 사용하여 list를 writeAtion.jsp에 넘겨줍니다.

```
//viewWrite.jsp에서 수정 버튼을 누르게 되면 get방식으로 bbsID의 value 값인 list.id가 넘어오게 됩니다.
//update 메소드에서 @RequestParam 으로 list.id 값을 받습니다.
@GetMapping(value = "/updating")
public String update(@RequestParam int bbsID, Model model) {
    //list.id 값을 사용자가 작성한 여행 후기 내용을 불러오는 service.lookId에 넘겨줍니다.
    //list.id는 게시물의 번호입니다.
    //service.lookId에 담은 값을 ListInfo 객체에 담아 줍니다.
    ListInfo list = service.lookId(bbsID);
    System.out.println("이 리스트 값은 20202020202020 + list);
    //list 값을 model을 통해 writeAtion.jsp에 넘겨줍니다.
    model.addAttribute("list", list);
    //writeAion.jsp페이지로 이동합니다.
    return "writeAion";
}
```

●writeAion.jsp

1)DB에서 불러온 데이터를 writeAion.jsp에 보여줄 때 수정이 가능한 상태로 보여줘야 합니다.

2)modelAttribute를 활용하여 ●UpdateController.java 로 받은 list를 각 input태그에 정보를 뿌려줍니다.

3)이때 수정이 가능한 input태그를 사용하여 데이터를 받습니다.

4)사용자가 자신의 게시글을 수정을 한 뒤 수정 완료 버튼을 누릅니다.

5)127번 게시글을 가져왔으니 127번 게시글에 다시 저장을 하기 위해 type:hidden으로 하여 다시 ListInfo 클래스에서 id값에 담아 ●UpdateController.java 보내줍니다.

```
47 </nav>
48 <div class="container">
49 <div class="row">
50 <form:form modelAttribute="list" method="post" action="/update/updating">
51 <table class="table table-striped"
52 style="text-align: center; border: 1px solid #dddddd">
53 <thead>
54 <tr>
55 <th colspan="2"
56 style="background-color: #eeeeee; text-align: center;">게시판
57 글쓰기 양식</th>
58 </tr>
59 </thead>
60 <tbody>
61 <tr>
62 <td><form:input type="text" class="form-control"
63 placeholder="제목" name="title" path="title" maxlength="100"/></td>
64 </tr>
65 <tr>
66 <td><form:input type="text" class="form-control"
67 placeholder="여행 지역" name="Location" path="Location" maxlength="100"/></td>
68 </tr>
69 <tr>
70 <td><form:input type="hidden" class="form-control"
71 path="id" name="id" value="{list.id}" /></td>
72 </tr>
73 <tr>
74 <td><form:input type="date" class="form-control"
75 placeholder="여행 날짜" name="date" path="date" maxlength="50"/></td>
76 </tr>
77 <tr>
78 <td><form:input type="date" class="form-control"
79 placeholder="여행 날짜" name="date" path="date" maxlength="50"/></td>
80 </tr>
81 <tr>
82 <td><form:textarea id = "editor4" name = "editor4" path="text"></form:errors path="text">
83 </td></tr>
84 </tbody>
85 </table>
86 </form>
87 </div>
88 </div>
89 </div>
```

●UpdateController.java

사용자가 수정한 내용을 담은 listInfo를 service.Update로 넘겨주어 DB에 다시 저장을 합니다.

```
//사용자가 자신의 게시글을 수정을 한 뒤 수정완료 버튼을 누르게 되면 post방식으로 넘어오게 됩니다.
@PostMapping("/updating")
public String add(@ModelAttribute ListInfo listInfo, Model model) {
    //수정된 내용을 담은 listInfo를 service.Update 메소드에 넘겨줍니다.
    //수정된 내용을 DB에 저장을 합니다.
    int results = service.Update(listInfo);
    System.out.println("업데이트 구문 이다!!!!!!!!!!!!!!!!!!!!!!" + results);
}
```

●WriteDao.java

```
@Override
public int Update(ListInfo l) {
    Date date = new Date();
    String query = "update writetable set title = ?, location = ?, date =?, writedate = ?,text = ?,star =? where id = ?";
    return jdbcTemplate.update(query, l.getTitle(),l.getLocation(),l.getDate(),date,l.getText(),l.getStar(),l.getId());
}
```

●UpdateController.java

마지막으로 return“bbs” 적어 사용자에게 전체게시판 (bbs.jsp) 다시 보여줍니다.

```
//사용자가 자신의 게시글을 수정을 한 뒤 수정완료 버튼을 누르게 되면 post방식으로 넘어오게 됩니다.
@PostMapping("/updating")
public String add(@ModelAttribute ListInfo listInfo, Model model) {
    //수정된 내용을 담은 listInfo를 service.Update 메소드에 넘겨줍니다.
    //수정된 내용을 DB에 저장을 합니다.
    int results = service.Update(listInfo);
    System.out.println("업데이트 구문 이다!!!!!!!!!!!!!!!!!!!!!!" + results);

    //페이징을 하는 부분입니다.
    //0~10번 게시글을 보여주기 사용함
    int a = 1;

    //페이지당 게시글 10개를 bbs.jsp에 model를 통해 넘겨줍니다.
    List<ListInfo> list = service.look(a);
    model.addAttribute("list", list);
    //총 게시글 숫자를 알려주는 역할을 수행합니다.
    int resultd = service.total();
    //총 게시글 숫자에서 나누기 10을 해줍니다.
    //나누기 10을 해주는 이유는 페이지당 게시글 10개를 보여주기 때문입니다.
    int totalpage = (int)Math.ceil(resultd/10.0);

    System.out.println("이 숫자를 알려주세요!!!!!!!!!!!!!!!!!!!!!!" +totalpage);
    //list에 정보를 담아 model을 통해 뷰로 전달을 합니다.
    model.addAttribute("totalpage",totalpage );

    return "bbs";
}
```

5.게시판 글을 삭제하기

●viewWrite.jsp

(해당 게시판 번호는 127번입니다.)

1)먼저 수정 버튼을 만들어 줍니다.

2)마우스 클릭을 하면 이벤트가 발생하게 되는 onclick을 이용하였습니다.

3)onclick에 url주소와 \${list id}(게시판 번호 127)를 작성합니다.

4)삭제버튼을 클릭하게 되면 이벤트가 발생하여 onclick에 있는 주소의 컨트롤러에게 이동을 하게 됩니다.

```
:if test = "${sessionScope.nickName == list.nicName}">
<button class="btn btn-primary pull-left" type="button" onclick="Location.href='/update/updating?bbsID=${list.id}'">수정</button>
<button class="btn btn-primary pull-left" type="button" onclick="Location.href='/update/delete?bbsID=${list.id}'">삭제</button>
</if>
```

●UpdateController.java

1)get방식으로 @RequestParam 으로 bbsID value 값을 받습니다.

2) 게시글을 삭제 서비스를 제공하는 service.delete 메서드에 bbsID value 값을 전달을 합니다

3)DB에서 해당 글번호 row를 삭제하고 다시 전체 글 리스트를 가져옵니다.

4)model을 이용하여 bbs.jsp에 다시 전달을 합니다.

5)bbs.jsp에서 삭제된 내용을 빼고 사용자에게 게시글을 보여줍니다.

```
//get방식으로 @RequestParam 으로 bbsID value 값을 받습니다.
//bbsID value 값을 게시글을 삭제 서비스를 제공하는 service.delete 메서드에 전달을 합니다.
//DB에서 해당 글번호 row를 삭제하고 다시 전체 글 리스트를 가져옵니다.
//model을 이용하여 bbs.jsp에 다시 전달을 합니다.
//bbs.jsp에서 삭제된 내용을 빼고 사용자에게 게시글을 보여줍니다.
@GetMapping("/delete")
public String delete(@RequestParam int bbsID, Model model) {

    System.out.println("bbsID는 무엇인가" + bbsID);
    //bbsID value 값을 게시글을 삭제 서비스를 제공하는 service.delete 메서드에 전달을 합니다.
    int result = service.delete(bbsID);
    System.out.println("삭제 구문이다 이다!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!" + result);
    int a = 1;
    //DB에서 해당 글번호 row를 삭제하고 다시 전체 글 리스트를 가져옵니다.
    List<ListInfo> list = service.look(a);
    model.addAttribute("list", list);

    return "bbs";
}
```

이로써 게시판의 기본적인 글쓰기, 추가하기,수정하기,삭제하기 기능에 대한 설명은 끝이 났습니다.
부족한 글을 읽어주셔서 다시 한 번 감사를 드립니다.

#게시판 만들기 part2

주제

○CKEDITOR 사용하기.

○페이징 만들기.

1.CKEDITOR 사용하기

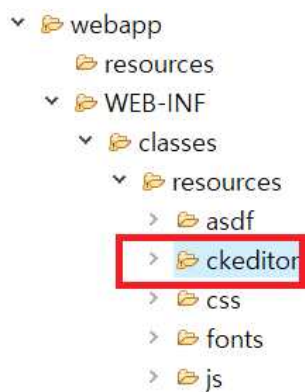
사용이유- 여행 후기를 작성할 때 후기 사진과 글을 함께 작성을 위해 에디터를 사용을 하였습니다.
대부분 여행 후기를 작성할 때 자신이 다녀온 사진과 함께 글을 함께 작성하는 경우가 많습니다.
에디터를 사용하면 쉽게 글과 사진을 함께 사용할 수 있다고 하여 CK에디터를 사용하였습니다.
다만 아쉬운 점은 사진의 용량이 작아야 올릴 수 있다는 단점이 있었습니다.

예시)



1)CK에디터 설정하기

1.CK에디터를 다운을 받은 뒤 파일을 resources 파일에 복사를 합니다.



2.사용할 jsp에 등록을 합니다.

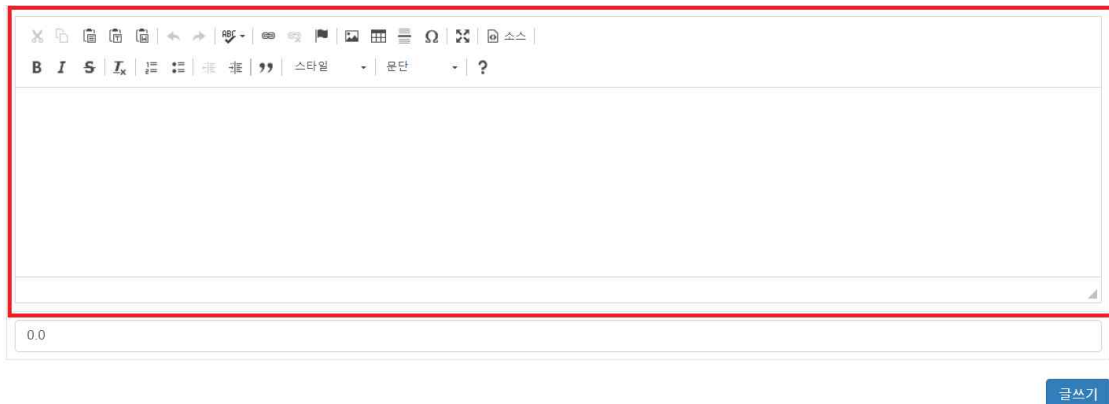
```
<link rel="stylesheet" href="/resource/css/bootstrap.min.css">
<script type="text/javascript" src="${path}/resource/ckeditor/ckeditor.js"></script>
<title>여행정보사이트</title>
```

textarea에 editor를 사용해줍니다.

```
style="height: 300px; /></textaread></td></tr>
</tr> -->
<tr>
<td><form:textarea id = "editor4" name= "editor4" path="text"/></textaread><form:errors path="text"/></td>
</tr>

</script>
CKEDITOR.replace('editor4',{filebrowserUploadUrl:'/mine/imageUpload.do'});</script>
</html>
```

아래와 같이 ck에디터 area가 생성 됩니다.



글과 사진을 추가를 추가한 뒤 에디터의 소스 부분을 클릭을 하게 되면 아래와 같이 글 부분과 사진의 소스 부분이 나뉘는 것을 볼 수 있습니다. 소스는 에디터가 자동적으로 생성을 해줍니다. 그 다음 글과 소스를 그대로 DB에 저장을 합니다.



DB에 저장된 내용입니다.

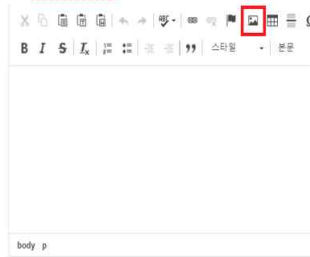
```
<p>여행 정말 재미있었다</p> <p></p>
```

DB에 저장된 내용을 그대로 보여주게 되면 글과 사진이 사용자에게 함께 보여 지게 됩니다.

3.파일 업로드

파일 업로드 경우에는 아래 순서와 같이 업로드를 해주시면 됩니다.

1. 그림버튼



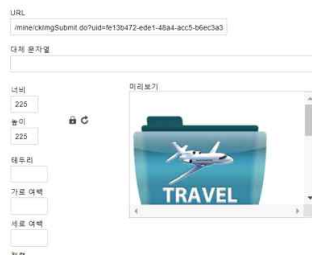
2. 파일선택



3. 서버 전송



4. 결과



●ImageUploadController

```
@Controller
public class ImageUploadController {

    //이미지 업로드를 하는 메소드입니다.
    @CrossOrigin("*")
    @RequestMapping(value="/mine/imageUpload.do",method = RequestMethod.POST)
    public void imageUpload(HttpServletRequest request, HttpServletResponse response, MultipartHttpServletRequest meultFile,@RequestParam MultipartFile upload) {
        //랜덤 문자 생성을 하여 그림 파일에 대한 고유한 이름 주는 역할을 함.
        UUID uid = UUID.randomUUID();

        OutputStream out = null;
        PrintWriter printWriter = null;

        //인코딩
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        try {

            String fileName = upload.getOriginalFilename();
            byte[] bytes =upload.getBytes();

            //이미지 저장소
            String path = "d:\\resources\\ckImage\\";
            String ckUploadpath = path + uid + "_" +fileName;
            File folder = new File(path);

            System.out.println(folder.getCanonicalPath());

            //폴더가 없으면 폴더를 생성해줍니다.
            if(!folder.exists()) {
                try {
                    folder.mkdirs();
                }catch(Exception e) {
                    e.printStackTrace();
                }
            }
            //이미지를 파일에 저장을 해줍니다.
            out = new FileOutputStream(new File(ckUploadpath));
            out.write(bytes);
            out.flush();

            String callback = request.getParameter("CKEditorFuncNum");
            printWriter = response.getWriter();
            String fileUrl = "/mine/ckImgSubmit.do?uid=" + uid + "&fileName="+fileName;

            printWriter.println("{\"filename\":\""+fileName+"\", \"uploaded\":1, \"url\":\""+fileUrl+"\"}");
            printWriter.flush();
        }catch(IOException e) {
            e.printStackTrace();
        }finally {
            try {
                if(out !=null) {out.close();}
                if(printWriter !=null) {printWriter.close();}
            }catch(IOException e) {e.printStackTrace();}
        }
        return;
    }
}
```

1)그림파일에 고유한 이름을 주기 위해 random클래스를 이용하여 랜덤문자를 생성합니다.

```
UUID uid = UUID.randomUUID();
```

2)확장자를 뺀 파일을 이름을 가져옵니다.

```
String fileName = upload.getOriginalFilename();
```

3)그림파일의 경우 바이트 단위로 되어 있기 때문에 byte배열을 만들어서 담아 줍니다.

```
byte[]bytes =upload.getBytes();
```

4)이미지 저장소를 만들어줍니다.

```
String path = "d:\\resources\\ckImage\\";
```

```
String ckUploadpath = path + uid + "-" +fileName;
```

```
File folder = new File(path);
```

5)out스트림으로 저장소에 그림파일을 전송합니다.

```
out = new FileOutputStream(new File(ckUploadpath));
```

```
out.write(bytes);
```

```
out.flush();
```

6)ck에디터를 쓰기 위해서는 꼭 callback을 해줘야 한다고 합니다. 구글을 참고 하였습니다.

```
String callback = request.getParameter("CKEditorFuncNum");
```

```
printWriter = response.getWriter();
```

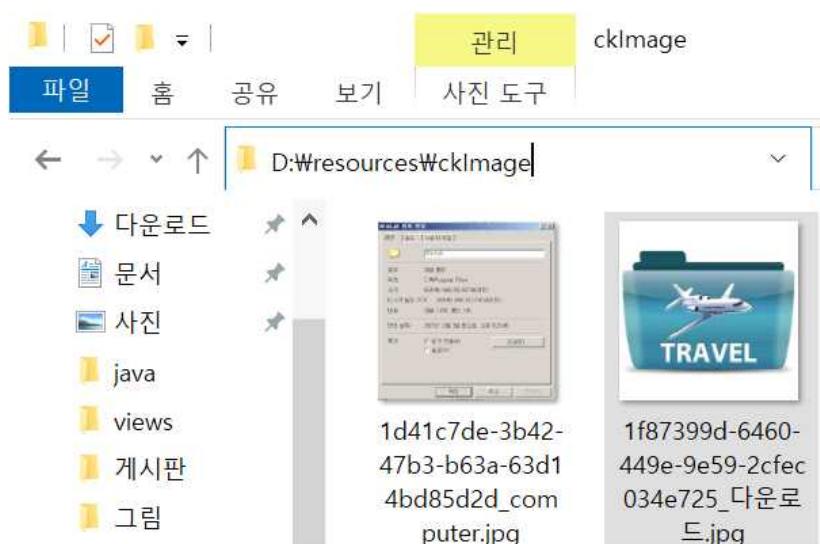
```
String fileUrl = "/mine/ckImgSubmit.do?uid=" + uid + "&fileName="+fileName;
```

```
printWriter.println("{\"filename\":\""+fileName+"\",\"uploaded\":1,\"url\":\""+fileUrl+"\"}");
```

```
printWriter.flush();
```

그럼 아래와 같이 이미지 파일이 "d:\\resources\\ckImage\\" 저장소에 저장된 것을 확인 할 수 있습니다.

●이미지파일이 저장된 폴더



3. 업로드된 이미지 파일을 하여 textarea에 붙이기

●ImageUploadController

```
//업로드된 이미지를 textarea에 그림을 붙여주는 역할을 합니다.  
@RequestMapping(value="/mine/ckImgSubmit.do")  
public void ckSubmit(@RequestParam(value="uid") String uid,@RequestParam(value="fileName")String fileName  
    ,HttpServletRequest request, HttpServletResponse response) throws IOException {  
  
    String path = "d:\\resources\\ckImage\\";  
    String sDirPath = path +uid+"_"+fileName;  
    File imgFile = new File(sDirPath);  
    System.out.println("이미지 파일을 읽어와주세요!!!!!!!!!!" +fileName);  
  
    if(imgFile.isFile()) {  
        byte[]buf= new byte[1024];  
        int readByte =0;  
        int length = 0;  
        byte[]imgBuf= null;  
  
        FileInputStream fileInputStream = null;  
        ByteArrayOutputStream outputSteam = null;  
        ServletOutputStream out = null;  
  
        try {  
            fileInputStream = new FileInputStream(imgFile);  
            outputSteam = new ByteArrayOutputStream();  
            out = response.getOutputStream();  

```

1) 아래의 URL로부터 value uid와 value fileName을 RequestParam으로 받아옵니다.

```
public void ckSubmit(@RequestParam(value="uid")  
String uid,@RequestParam(value="fileName")String fileName  
    ,HttpServletRequest request, HttpServletResponse response) throws IOException {
```

URL-/mine/ckImgSubmit.do?uid=198d4243-2358-45a9-94f1-c4bc24e6e738&fileName=다운로드.jpg

이미지 속성 ✕

이미지 정보

링크

업로드

URL
/mine/ckImgSubmit.do?uid=fe13b472-edc1-48a4-acc5-b6ec3a3


대체 문자열

너비
225

높이
225

테두리

가로 여백

미리보기


2)경로+랜덤값+파일네임으로 만들어진 이미지 파일을 찾아 sDirPath에 담아줍니다.

```
String sDirPath = path +uid+"_"+fileName;
```

3)파일을 생성해줍니다.

```
File imgFile = new File(sDirPath);
```

4)이미지 파일을 찾아줍니다.

```
if(imgFile.isFile()) {
```

```
byte[]buf= new byte[10000000]; //jpg 파일 크기를 잡음.
```

5) 찾은 이미지파일을 while문으로 처음부터 끝까지 읽어 output 스트림으로 HTML로 보내줍니다.

```
try {
```

```
fileInputStream = new FileInputStream(imgFile);
```

```
outputSteam = new ByteArrayOutputStream();
```

```
out = response.getOutputStream();
```

```
while((readByte = fileInputStream.read(buf))!=-1) {
```

```
outputSteam.write(buf,0,readByte);
```

```
imgBuf = outputSteam.toByteArray();
```

```
length = imgBuf.length;
```

```
out.write(imgBuf,0,length);
```

```
out.flush();
```


6) 결과,


게시판

제목

여행 지역

연도-월-일





body p

0.0

2.페이징 만들기.

113	83
[1] [2] [3] 다음	이전 [4] [5] [6] 다음

●bbs.jsp

totalpage가 3미만일 때

forEach문으로 1에서 totalpage까지 페이징 숫자를 만들어 줍니다. 만약 totalpage가 2인 경우에는 [1],[2]로 나타나게 하였습니다.

```
</table>
<c:if test="${a == null || a == 0}">
  <c:if test="${totalpage <= 3}">
    <c:forEach var = "pageNum1" begin = "1" end = "${totalpage}" >
      <a href = "/BBS/coutView?page=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
  </c:if>
  <c:if test="${totalpage > 3}">
    <c:forEach var = "pageNum1" begin = "1" end = "3" >
      <a href = "/BBS/view?viewpage=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
    <a href = "/BBS/coutView?page=3">다음</a>
  </c:if>
</c:if>
```

●BbsController.java

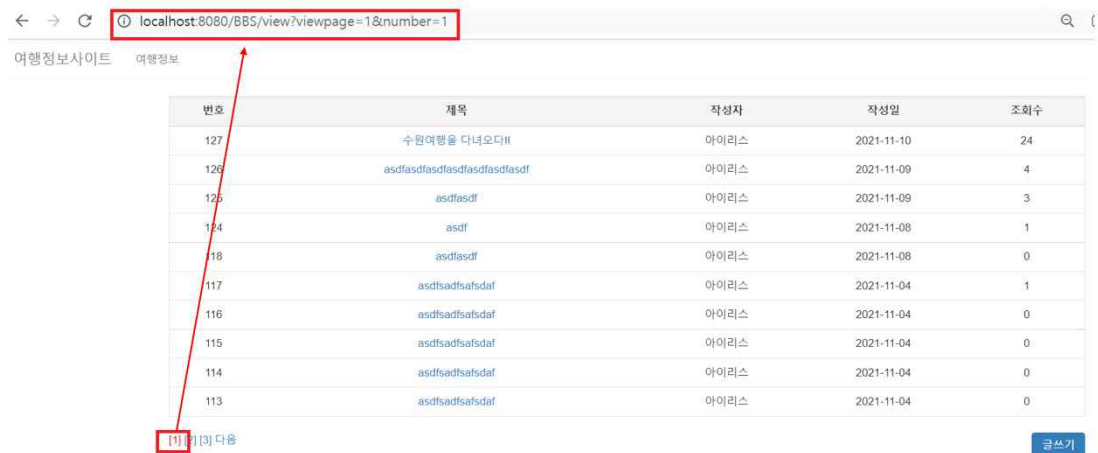
total페이지는 컨트롤러에서 service.total 메소드를 통해서 view페이지에 알려줍니다.

```
int result = service.total();
int totalpage = (int)Math.ceil(result/10.0);

System.out.println("이 숫자를 알려주세요!!!!!!!!!!!!!!!!!!!!!!" +totalpage);

model.addAttribute("totalpage",totalpage );
```

1)페이지 [1]버튼을 눌렀을 때 나타나는 URL 주소
<http://localhost:8080/BBS/view?viewpage=1&number=0>



●bbs.jsp

totalpage가 3초과일 때

forEach문으로 1에서 totalpage까지 페이지징 숫자를 만들어 줍니다. 만약 totalpage가 2인 경우에는 [1],[2],[3]로 나타나게 하였습니다.

[\${pageNum1}]

a태그를 사용하여 사용자가 페이지 번호를 누르게 되면 \${pageNum1}에 담기고, number=\${0}에 는 숫자 0이 컨트롤러에게 넘어갑니다.

예) 사용자가 페이지 2번을 누르게 되면 URL 주소에 viewpage =2,number = 0 적히는 것을 볼 수 있습니다.

<http://localhost:8080/BBS/view?viewpage=2&number=0>

```

</table>
<c:if test="${a == null || a == 0}">
  <c:if test="${totalpage <= 3}">
    <c:forEach var = "pageNum1" begin = "1" end = "${totalpage}" >
      <a href = "/BBS/coutView?page=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
  </c:if>
  <c:if test="${totalpage > 3}">
    <c:forEach var = "pageNum1" begin = "1" end = "3" >
      <a href = "/BBS/view?viewpage=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
    <a href = "/BBS/coutView?page=3">다음</a>
  </c:if>

```


●BbsController.java

조건1) 사용자가 게시판 2페이지를 보고 싶은 경우 = [2]를 클릭을 한 경우

조건2) [2] 2페이지를 클릭하게 되면 DB에서 게시판 1번~10번을 띄어 넘고 11~20번을 보여줌.

해당 두 조건을 만족을 시켜야 합니다. 사용자가 2페이지를 보고 싶은 경우라서

조건(1) URL 주소에 viewpage =2를 ●BbsController.java 에서 @RequestParam in viewpage로 숫자 2값을 받습니다. 그 후 숫자 2의 값을 int a에 담아놓습니다.

조건(2) 숫자 2를 가지고 있는 변수 a를 service.look(a) 담아줍니다.

```
@GetMapping(value = "/view")
public String view(@RequestParam int viewpage, @RequestParam int number, Model model) {
    int a = viewpage;
    int d = number;
    System.out.println("d의 값이 궁금해요!!!!!!!!!!!!!!!!!!!!!! " + d);
    System.out.println("a의 값이 무엇인지 알고 싶어요!!!!!!!!!!!!!!!!!!!!!! " + a);
    List<ListInfo> list = service.look(a);
    model.addAttribute("list", list);
}
```

●WriteDao.java

다오에서 숫자2의 값을 입력받아 int a = (page-1)*10;/int a = (2-1)*10;

a=10이 되고, offset=10이 됩니다.

offset을 이용하여 내림차순으로 10개의 게시물을 띄어 넘고 10개의(limit=10) 게시물만 가져옵니다.

끝으로 사용자에게 10개의 게시물을 보여줍니다.

```
@Override
public List<ListInfo> look(int page) {
    int a = (page-1)*10;
    return jdbcTemplate.query("SELECT a.id,a.nickName,a.title, a.date, b.count FROM writetable as a left join views as b on a.id = b.id "
        + "ORDER BY a.ID DESC limit 10 offset ? ",new ListInfoRowMapper(), a);
}
```

●BbsController.java

http://localhost:8080/BBS/view?viewpage=2&number=0

number=0의 사용여부에 대해서 말씀을 드리겠습니다. @RequestParam int number로 받아 int d 변수에 담아둡니다. modelAttribute로 key=a value=0 으로 bbs.jsp에 넘겨줍니다.

```
@GetMapping(value = "/view")
public String view(@RequestParam int viewpage, @RequestParam int number, Model model) {
    int a = viewpage;
    int d = number;
    System.out.println("d의 값이 궁금해요!!!!!!!!!!!!!!!!!!!!!! " + d);
    System.out.println("a의 값이 무엇인지 알고 싶어요!!!!!!!!!!!!!!!!!!!!!! " + a);
    List<ListInfo> list = service.look(a);
    model.addAttribute("list", list);

    int result = service.total();
    int totalpage = (int)Math.ceil(result/10.0);

    System.out.println("이 숫자를 알려주세요!!!!!!!!!!!!!!!!!!!!!! " +totalpage);

    model.addAttribute("totalpage",totalpage );
    model.addAttribute("a",d);

    return "bbs";
}
```

●bbs.jsp

컨트롤러에서 modelAttribute로 key=a value=0 받습니다. 사용자는 현재 1~3페이지 페이징을 보구 있기 때문에 bbs.jsp에서 ■ 페이징을 보여줘야 합니다. 그렇기 때문에 number=\${0}을 사용하게 되었습니다.

```

<:if test="${a == null || a == 0}">
  <:if test="${totalpage <= 3}">
    <:forEach var = "pageNum1" begin = "1" end = "${totalpage}" >
      <a href = "/BBS/coutView?page=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
  </c:if>
  <:if test="${totalpage > 3}">
    <:forEach var = "pageNum1" begin = "1" end = "3" >
      <a href = "/BBS/view?viewpage=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
    <a href = "/BBS/coutView?page=3">다음</a>
  </c:if>
</c:if>
<:if test="${a != null && a != 0}">
  <:if test="${totalpage <= a+2}">
    <:if test="${a != 1}">
      <a href = "/BBS/maius?maius=${a-3}" >이전</a>
    </c:if>
    <:forEach var = "pageNum1" begin = "${a}" end = "${totalpage}" >
      <a class="pages" href = "/BBS/view?viewpage=${pageNum1}&number=${a}" >[${pageNum1}]</a>
    </c:forEach>
  </c:if>
  <:if test="${totalpage > a+2}">
    <:if test="${a != 1}">
      <a href = "/BBS/maius?maius=${a-3}" >이전</a>
    </c:if>
    <:forEach var = "pageNum1" begin = "${a}" end = "${a+2}" >
      <a class="pages" href = "/BBS/view?viewpage=${pageNum1}&number=${a}" >[${pageNum1}]</a>
    </c:forEach>
    <a href = "/BBS/coutView?page=${a+2}">다음</a>
  </c:if>
</c:if>

```

사용자가 보고 있는 페이지

a의 값이 0,null,1이면 if문때문에 아래 내용이 실행되지 않습니다.

2. 다음을 누르게 되면 [1] [2] [3] → [4] [5] 페이지가 보이게 하기

●bbs.jsp

조건1)total페이지= 4~5

조건2)사용자가 4페이지를 보고 싶어 함.

조건3)다음페이지를 누르게 되면 [4],[5] 보여줘야 함.

[1] [2] [3] 다음

●bbs.jsp

1.사용자가 다음을 누를 수 있는 a태크를 만들어 줍니다.

2.다음을 누르게 되면 url에 key=page, value =3 값을 컨트롤러에게 넘겨줍니다.

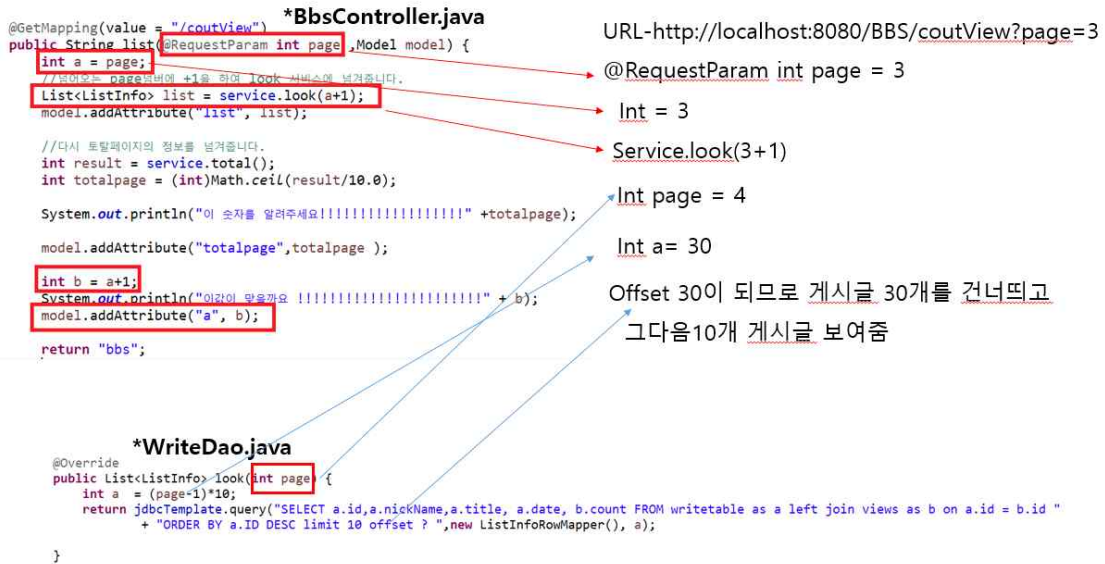
```

<:if test="${a == null || a == 0}">
  <:if test="${totalpage <= 3}">
    <:forEach var = "pageNum1" begin = "1" end = "${totalpage}" >
      <a href = "/BBS/coutView?page=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
  </c:if>
  <:if test="${totalpage > 3}">
    <:forEach var = "pageNum1" begin = "1" end = "3" >
      <a href = "/BBS/view?viewpage=${pageNum1}&number=${0}" >[${pageNum1}]</a>
    </c:forEach>
    <a href = "/BBS/coutView?page=3">다음</a>
  </c:if>
</c:if>

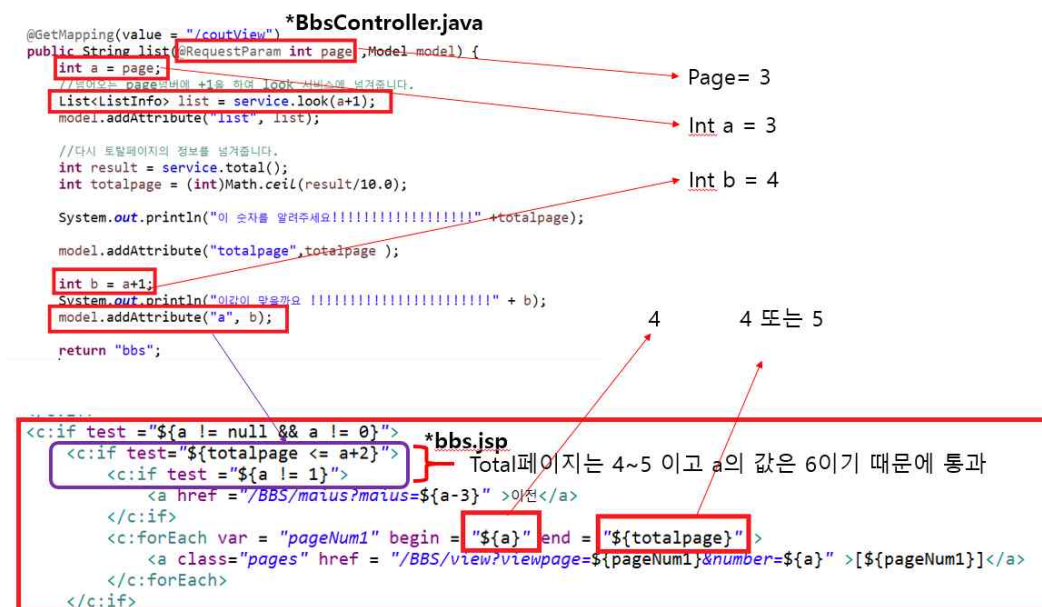
```

●BbsController.java

- 1) @RequestParam으로 key=page, value= 3의 값을 받습니다.
- 2) value 3의 값을 int a 변수에 담아 둡니다.
- 3) 4페이지의 게시글은 앞에서 보았던 [1],[2],[3] 게시글 30개 다음의 게시글을 보여줘야 하기 때문에 service.look()에 4(3+1)의 값을 전달합니다.
- 4) WriteDao에서 4의 값을 입력받아 (4-1)*10을 계산하여 int =a 변수에 담아둡니다.
- 5) int a= 30이 되고 쿼리문을 통해 offset = 30되어 게시글 30개를 건너뛰고 그 다음 게시글 10개를 보여줍니다.



- 5) int b = 4
- 6) bbs.jsp에서 key=a,value=4 값을 받아 if문을 통과를 하고 begin=%{a}에는 4 end = \${totalpage}에는 4~5 값을 담아줍니다.
- 7) 결과 사용자에게는 [4] 또는 [4],[5]페이지가 보이게 됩니다.



3.이전으로 돌아가기 만들기 [4] [5] [6] → [1] [2] [3]페이지가 보이게 하기

이전 [4] [5] [6] 다음

- 1) a href = "/BBS/maius?maius=\${a-3}"에서 \${4-3} 됩니다.
- 2) controller에서 int a = 1이 됩니다.
- 3) service.look(a+2)에서 +2를 해주는 이유는 이전을 누르게 되면 3페이지의 게시글을 보여줘야 합니다. 3페이지 게시글을 보여줄 수 있도록 1+2 =3의 값을 Dao에 넘겨줘서 쿼리를 통해 게시글 20개를 건너뛰고 다음 게시글 10개를 보여주도록 만듭니다.

이전 [4] [5] [6] 다음 → [4] [5] [6] 에서 이전을 눌렀을 때

```
</c:if>
<c:if test="${a != null && a != 0}">
  <c:if test="${totalpage <= a+2}">
    <c:if test="${a != 1}">
      <a href = "/BBS/maius?maius=${a-3}" >이전</a>
    </c:if>
```

a의 값은 4

```
@GetMapping(value = "/maius")
public String mius(@RequestParam int maius, Model model) {
```

```
    int a = maius; → a의 값은 1
```

```
    List<ListInfo> list = service.look(a+2);
    model.addAttribute("list", list);
```

→ 다오에서 offset 20으로 게시글 20개를 건너뛰고 다음 게시글 10개 보여줌

```
    int result = service.total();
    int totalpage = (int) Math.ceil(result/10.0);

    System.out.println("이 숫자를 알려주세요!!!!!!!!!!!!!!!!!!!!!!" + totalpage);

    model.addAttribute("totalpage", totalpage );
```

```
    System.out.println("이값이 맞을까요 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!" + a);
    model.addAttribute("a", a);
```

```
    return "bbs";
```

이로써 페이징 만들기는 끝이 납니다. 감사합니다.

#마이페이지

작성자: 이상민

주제

○사용자 프로필

이름(수정불가)

닉네임(수정불가)

아이디(수정불가)

연락처(수정가능)

성별(수정불가)

○프로필 사진 올리기

○본인이 작성한 게시글 보기

1.사용자 프로필

1)마이페이지에 들어가게 되면 회원가입 당시 본인이 작성했던

이름,닉네임,아이디,연락처,성별을 확인을 할 수 있습니다.

2)이름,닉네임,아이디,성별은 수정이 불가능하게 설정되었습니다.

3)연락처만 수정이 가능하도록 하였습니다.

No.	제목
127	수원여행길 다녀오다
126	asdfasdfasdfasdfasdf
125	asdfasdf
124	asdf
118	asdfasdf

●MyController.java

1)세션에서 네임을 들고와서 service.lookIds 서비스에 넘겨줍니다.

2)DB에서 이름을 검색해서 해당 row 값을 가져옵니다.

```
@GetMapping(value = "/mypage")
public String list(Model model) {

    System.out.println("이름은 : " + services.pnickname);
    //이름으로 Joindtable에 있는 이름, 닉네임, 아이디, 핸드폰번호, 성별 row를 가져옵니다.
    //가져온 row를 JoinInfo에 담아보겠습니다.
    JoinInfo list = service.lookIds(services.pnickname);
    System.out.println("리턴된 값들 : " + list);
    //model을 이용하며 my.jsp에 넘겨줍니다.
    model.addAttribute("list", list);
}
```


●WriteDao.java

1)MyController에서 이름 값을 받아 쿼리문을 실행하여 아이디,네임,폰,닉네임,성별을 JoinInfo 클래스에 담아줍니다.

```
@Override
public JoinInfo lookIds(String name) {

    return jdbcTemplate.queryForObject("select userid, name,phone,nickname,gender from jointable where name = ?",
        new RowMapper<JoinInfo>() {
            @Override
            public JoinInfo mapRow(ResultSet rs, int rowNum) throws SQLException {

                return new JoinInfo(rs.getString("userid"),rs.getString("name"),rs.getString("phone"),rs.getString("nickname"),rs.getString("gender"));
            }, name);
        }
    );
}
```

●MyController.java

1)model를 사용하여 my.jsp에 key=list, value=list 값을 넘겨줍니다.

```
@GetMapping(value = "/mypage")
public String list(Model model) {

    System.out.println("이같은 코딩인가요" + services.pnickname);
    //이름으로 Jointable에 있는 이름, 닉네임, 아이디, 핸드폰번호, 성별 row를 가져옵니다.
    //가져온 row를 JoinInfo에 담아줍니다.
    JoinInfo list = service.lookIds(services.pnickname);
    System.out.println("리스트의 값을 알려줘" + list);
    //model를 이용하여 my.jsp에 넘겨줍니다.
    model.addAttribute("list", list);
    name = list.getNickName();
    System.out.println("////////////////////////////////////" + name);
}
```

●my.jsp

1)JoinInfo에 있는 값들을 list.name,list.nickName으로 값을 꺼내줍니다.

2)핸드폰은 수정 가능하도록 input 태그를 사용해줍니다.

```
<div class="li">
    <div class="tt">이름</div>
    <div class="tx">${list.name}</div>
</div>
<div class="li">
    <div class="tt">닉네임</div>
    <div id="nick" class="tx">${list.nickName}</div>
</div>
<div class="li">
    <div class="tt">아이디</div>
    <div class="tx">${list.userId}</div>
</div>
<div class="li">
    <div class="tt pn_tt">연락처</div>
    <input type="text" class="tx pn_input" size="10"
        maxlength="13" id="phoned" name="phoned" placeholder="- 핸드폰을 불러주세요" value="${list.phone}"/>
    <input type="button" id="btn2" class="btn btn-primary pull-right tx pn_btn" value="수정">
</div>
<div class="li">
    <div class="tt">성별</div>
    <div class="tx">${list.gender}</div>
</div>
```


2.핸드폰 번호 수정하기.

- 조건1) 핸드폰 숫자와 하이픈(-)을 합치면 길이가 13입니다. 사용자가 연락처를 입력하지 않거나 13자리 미만으로 입력하게 되는 경우 경고 메시지 보여주기
- 조건2) 핸드폰 숫자 길이를 13자리 초과로 입력하지 않도록 만들기
- 조건3) 핸드폰 번호를 컨트롤러를 거쳐 DB에서 처리한 후 사용자에게 보여주기.

이름	닉온
닉네임	마미리스
아이디	asdf
연락처	010-4340-3830 수정
성별	남자

조건1)에 대한 방법: 제이쿼리 사용.

- 1.버튼 id="btn2"를 \$("#btn2").click에 담아 버튼 클릭 시 이벤트가 발생하도록 만듭니다.
- 2.input 태그에 있는 id="phoned"를 \$("#phoned").val();담아 사용자가 입력한 핸드폰 값을 var phones 변수에 담습니다.
- 3.if문을 활용하여 경고 메시지를 사용자에게 알려줍니다.

```
<input type="text" class="tx pn_input" size="10"
      maxlength="13" id="phoned" name="phonesed" placeholder="- 하이픈을 붙여주세요" value="${list.phone}"/>
<input type="button" id="btn2" class="btn btn-primary pull-right tx pn_btn" value="수정">

<script>
$("#btn2").click(function(){
    var phones = $("#phoned").val();
    if(phones == "" || phones == null || phones.length <=12){
        alert("- 하이픈 또는 핸드폰번호를 다시 확인해주세요. ");
    }else{
        console.log("andjtdlsrkyd" + phones);
        $.ajax({
            url : '/myphones/phonenummer',
            type : 'POST',
            data : phones,

            success : function(data) {
                console.log(data);
                if (data == 1) {
                    alert("핸드폰 번호가 수정되었습니다.");
                }else{
                    alert("수정되지 않았습니다.");
                }
            }
        });
    }
});
```

조건2)에 대한 방법: input태그에서 maxlength 사용하기

위처럼 조건식을 만들어서 사용자에게 경고창을 통해 알려줄 수 있지만, 13자리 초과로 입력되지 않도록 만들어주는 것이 조금 더 직관적이므로 input태그에서 maxlength를 사용하였습니다.

```
<input type="text" class="tx pn_input" size="10"
      maxlength="13" id="phoned" name="phonesed" placeholder="- 하이픈을 붙여주세요" value="${list.phone}"/>
<input type="button" id="btn2" class="btn btn-primary pull-right tx pn_btn" value="수정">
</div>
```

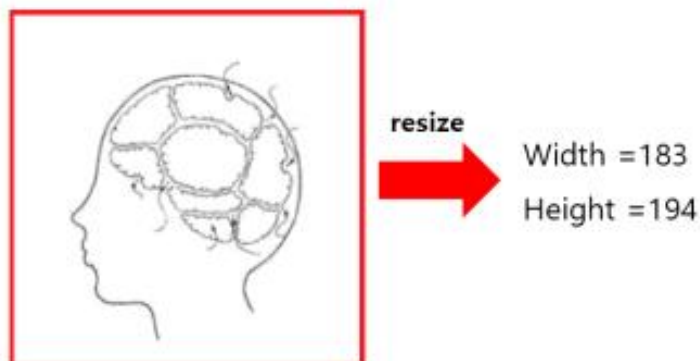
7.my.jsp에서 값을 받아 사용자에게 결과창을 알려줍니다.



조건1) 파일을 올릴 수 있는 파일 태그를 만들어줍니다.



조건2) 다운 받은 사진마다 크기가 다르기 때문에 resizing을 해줍니다.



조건3) 변경을 누르게 되면 프로필 사진이 변경이 됨.



조건1)에 대한 방법: file태그 사용하기.

- 1.input type= "file"로 해줍니다.
- 2.위 그림처럼 파일 선택 버튼이 생성됩니다.
- 3.파일선택 후 해당 이미지 파일을 컨트롤러에게 줄 수 있는 버튼 타입을 하나 더 만듭니다.

```
<form id="FILE FORM" method="post" action="/file/file_Check" enctype="multipart/form-data">
  <input type="file" id="FILE_TAG" name="userfile" class="btn btn-primary pull-left img_input">
  <input type="button" id="btn" class="img_btn" value="변경">
</form>
```

조건2)에 대한 방법: 리사이징 해주기.

1. 사용자가 이미지 파일을 선택 후 변경 버튼을 누르게 됩니다.
2. 이미지파일을 리사이징 해주는 컨트롤러에게 넘어가게 됩니다.
3. 이미지파일을 컨트롤러에게 넘겨줄 때 ajax와 formdata를 사용하였습니다.
4. formdata 사용은 form안에 form을 더 만들어 묶어줍니다. form id="FILE.FORM"으로 해줍니다.

```
<form>
  <div class="my_imgdiv" >
    
    <div class="my_imgfile">
      <form id="FILE FORM" method="post" action="/file/file_Check" enctype="multipart/form-data">
        <input type="file" id="FILE_TAG" name="userfile" class="btn btn-primary pull-left img_input">
        <input type="button" id="btn" class="img_btn" value="변경">
      </form>
    </div>
  </div>
```

5.ajax를 사용해줍니다.

- 1.변경 버튼에 대한 id="btn" 값을 가져옵니다.
- 2.var form 변수에 파일을 담아줍니다.
- 3.formDate를 선언을 해주고 form을 담아줍니다.
- 4.url,post,formData를 FileController에게 넘겨줍니다.

```
<script>
$( "#btn" ).click(function(){
  var form = $("#FILE FORM")[0];
  var formData = new FormData(form);
  formData.append("userfile", $("#FILE_TAG")[0].files[0]);
  console.log("andjtdlsrkdy" + formData);
  $.ajax({
    url: '/file/file_Check',
    type: 'POST',
    data: formData,
    processData: false,
    contentType: false,
    success: function(data) {

@PostMapping("/file_Check")
public @ResponseBody String file_Check (@RequestBody MultipartFile userfile) throws IOException {
  String fileName =userfile.getOriginalFilename();
  byte[] filebyte =userfile.getBytes();

  String nickName = my.name;

  System.out.println("===== " +fileName);
  System.out.println("닉네임을 알려줘" + nickName);
}
```

```
//마지막에 이미지 파일 크기
int newWidth = 183;
int newHeight = 194;
String imgFormat = "jpg";

try {
    image = ImageIO.read(new ByteArrayInputStream(filebyte));
    imageWidth = image.getWidth(null);
    imageHeight = image.getHeight(null);

    System.out.println("이미지 크기값 알려주세요!" + imageWidth);
    System.out.println("이미지 크기값 알려주세요!" + imageHeight);

    Image resizeImage = image.getScaledInstance(newWidth, newHeight, Image.SCALE_SMOOTH);
    BufferedImage newImage = new BufferedImage(newWidth, newHeight, BufferedImage.TYPE_INT_RGB);
    Graphics g = newImage.getGraphics();
    g.drawImage(resizeImage, 0,0,null);
    g.dispose();
    System.out.println("newImage 는 무엇인가?" + newImage);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ImageIO.write(newImage, imgFormat, baos);
    byte[] bytes = baos.toByteArray();
```

●FileController.java

- 11.사용자가 처음 마이페이지에 접속을 하게 되면 초기의 이미지가 없기 때문에 이미지 파일을 추가해주는 서비스를 제공합니다.
- 12.리사이징한 bytes, fileName, nickName을 서비스에 넘겨줍니다.
- 13.사용자가 프로필 사진을 변경을 하고 싶은 경우 이미지 파일과 nickName만 서비스로 넘겨 update 구문으로 image 테이블의 이미지를 변경해줍니다.

```
// result 값이 0이면 파일이 없어서 add추가할 함
// result 값이 1이면 파일이 있어서 업데이트로 함.
int result = service.filecheck(nickName);
if(result == 0) {
    int results = service.fileAdd(bytes, fileName, nickName);
    System.out.println("파일의 결과값을 알려주세요" +results);

    if(results ==1) {
        //성공
        return "1";
    }else {
        return "2";
    }
}
}else {
    int resultUpdate = service.fileout(bytes, nickName);
    System.out.println("파일의 업데이트 값을 알려주세요" +resultUpdate);

    if(resultUpdate ==1) {
        //성공
        return "3";
    }else {
        return "4";
    }
}
}
```

●WriteDao.java

```
//image 테이블에 파일이 없으면 추가하는 다오.
@Override
public int fileAdd(byte[] bytes,String fileName, String nickName) {
    String query = "INSERT INTO image(nickname,filename,file)VALUES(?,?,?)";
    try {
        return jdbcTemplate.update(query,nickName,fileName,bytes);
    } catch (DataAccessException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return 0;
}
//image 테이블에 파일이 있으면 이미지를 변경해주는 다오.
@Override
public int fileUpdate(MultipartFile file, String nickName) {
    String query = "update image set filename = ?, file = ? where nickname = ?";
    try {
        return jdbcTemplate.update(query,file.getOriginalFilename(),file.getBytes(),nickName);
    } catch (DataAccessException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return 0;
}
```


14. 이미지 경우에는 `img src=""`를 통해서 보여줍니다.
15. `FileController` 클래스에서 이미지를 추가를 하거나 변경을 해주는 작업까지는 하였지만 `my.jsp`로 이미지를 넘겨주는 부분이 없습니다. 텍스트 경우 `model`을 사용하여 텍스트 값을 넘겨주었습니다.
16. 이미지 경우에는 방법을 잘 알지 못하여 `ImageViewControllder` 클래스를 만들어 `ResponseEntity` 클래스를 사용을 하였습니다.
17. `ResponseEntity`를 사용하여 `my.jsp`에 header에는 "content-type", "image/jpeg"를 주겠다. body에는 이미지 파일을 전달을 하는 방식으로 작성을 하였습니다.
18. 결과 `/mine/ckImgSubmit.do?주소`를 통해서 사용자가 지정한 이미지 파일을 보여줍니다.

```
*my.jsp
<div class="my_imgdiv">
  
  <div class="my_imgorite">

*ImageviewController.java
@Controller
public class ImageviewController {

    @Autowired
    private IWriteService service;

    @GetMapping(value="/mine/ckImgSubmits.do")
    public ResponseEntity<Resource> ckSubmit(@RequestParam String nickName) throws IOException {
        System.out.println("닉네임이 받아와 지나요 알려주세요!!!!!!!!!!!!!!!!!!!!!!" + nickName);

        Resource resource = new ByteArrayResource(service.fileRead(nickName));
        return ResponseEntity.ok().header("content-type", "image/jpeg").body(resource);
    }
}
```

