**스마트문화앱 구현**

**포트폴리오**

**평가자료**

**-주제기술-**

수업명 : **자바(JAVA),스프링(Spring)기반 스마트웹&앱 콘텐츠 개발자 과정**

소프트웨어 개발 및 응용 과정

제출일 : 2022. 10. 07

작성자 :지성진

□프로젝트 개요

|  |  |  |
| --- | --- | --- |
| **구분** | **내용** | **비고** |
| **제목** | 반려동물 병원, 동반가능 카페 등 정보 제공 사이트 |  |
| **프로젝트**  **작성기간** | ~2022.10.18 |  |
| **조원 구성** | 김명훈, 김병진, 이승현, 이영민, 이진실, 지성진 |  |
| **프로젝트 내용** | 반려동물 병원, 동반가능 카페 등 정보 제공 사이트 |  |
| **주요기능** | 반려동물 병원 정보 제공 |  |
| 반려동물 호텔, 모텔 등 정보 제공 |  |
|  |  |
| **학습내용** | 스프링 기반 웹 개발 |  |
| 자바스크립트 |  |
| CSS |  |
| HTML |  |

□최종 보고서

|  |  |  |
| --- | --- | --- |
| **구분** | **내용** | **비고** |
| **제목** | 반려동물 병원, 동반가능 카페 등 정보 제공 사이트 |  |
| **프로젝트 내용** | 반려동물 병원, 동반가능 카페 등 정보 제공 사이트 |  |
| **완료내용** | Security 로그인 기능 | 프로젝트  개요와 동일 |
| 웹 페이지 데이터베이스 연동 | 프로젝트  개요와 동일 |
| 웹 디자인 | 프로젝트  개요와 동일 |
| **프로젝트 활동 후기** | | |
| **좋았던 점** | 팀원과의 협업을 통하여  페이지 제작 할 수 있었음 |  |
| **아쉬웠던 점 및**  **어려웠던 점** | 업무 분담이 쉽지 않음 |  |
| 버전관리 어려움 |  |
| 합치는 과정이 힘듬 |  |
| **학습 효과** | 어려운 일을 같이 해내서 성취감을 느낌 |  |

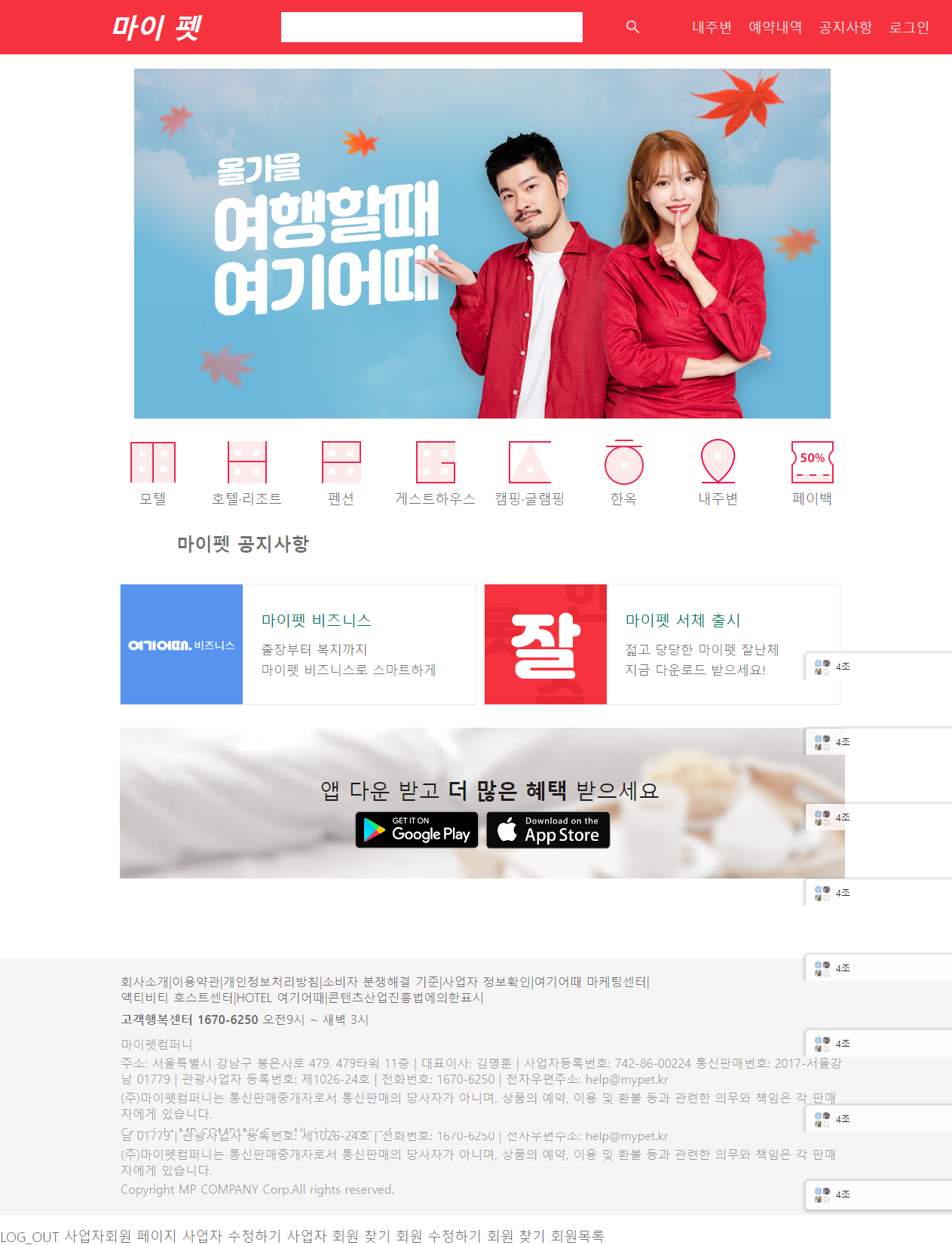
□프로젝트 세부 사항 분석 및 구조 설계

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **주요 기능/구문** | **내용** | **입출력** |
| **1** | HTML | 웹 페이지 틀 제작 | X |
| **2** | CSS | 웹 디자인 | X |
| **3** | Spring | 웹 페이지 연동 | X |
| **4** | JavaScript | 기능 추가 | X |

□작성 과정, 실행결과

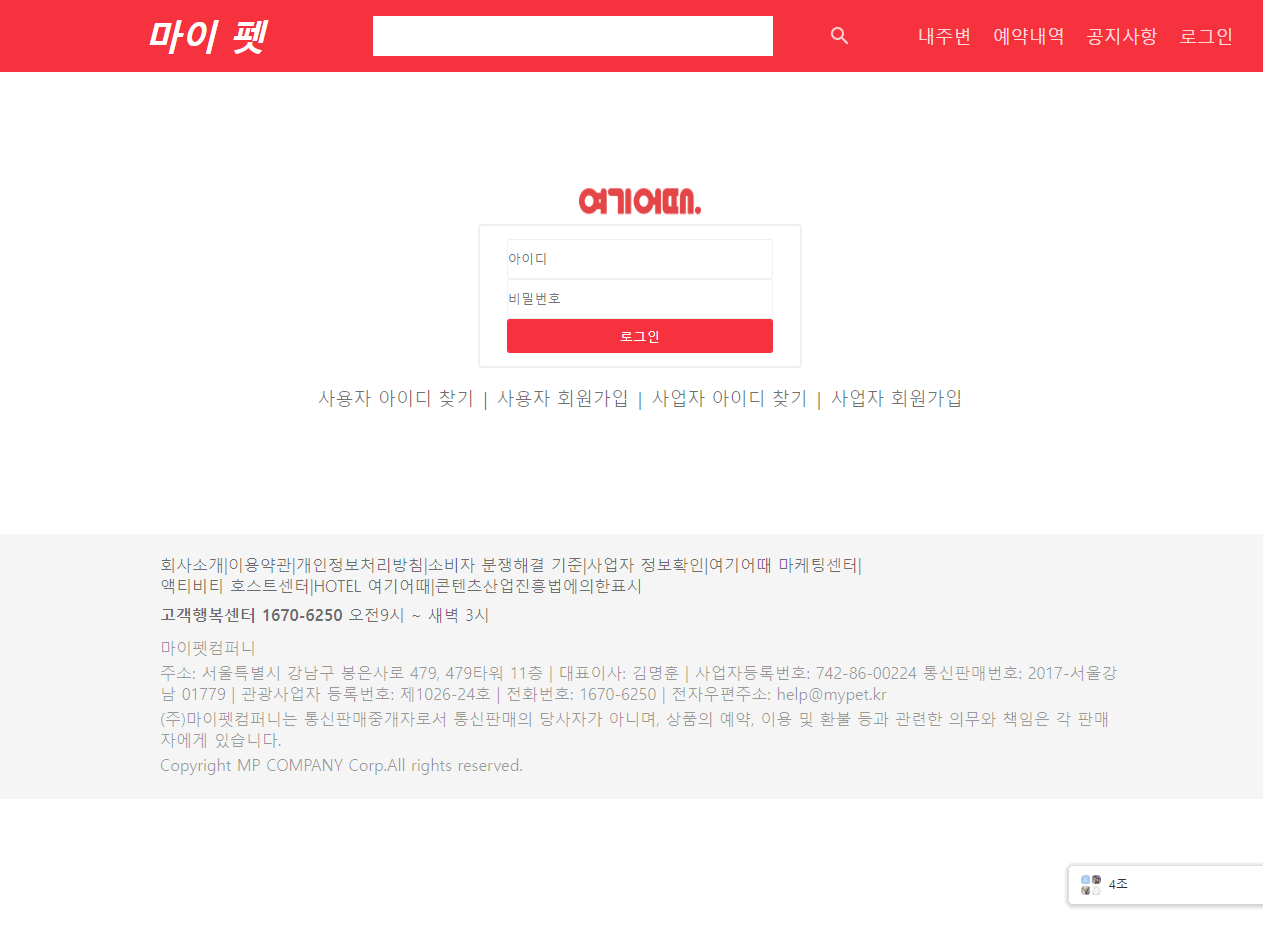
|  |  |  |
| --- | --- | --- |
| **구현 내용** | **결과** | **비고** |
| 홈페이지 구동 및 회원가입, 로그인 | O |  |

**메인 페이지**



처음 실행하면 나오는 메인페이지

@GetMapping("/index")  
public String index() {  
 return "index";  
}

**로그인페이지**DB에 있는 로그인을 불러서 대조한 후 올라갑니다.

@GetMapping("/Login")  
public void loginView() {  
}  
  
//로그인  
@PostMapping("/Login")  
public String login(Member member, Model model) {  
 Member findMember = memberService.getMember(member);  
 // 아이디, 비번 알치해야지 로그인 가능  
 // 회원 로그인을 할 경우 아이디와 비밀번호가 일치하지 않으면 로그인 페이지로 되돌아 온다.  
 // 로그인 성공시 회원 페이지로 이동  
 if (findMember != null  
 && findMember.getPassword().equals(member.getPassword())  
 && findMember.getId().equals(member.getId())) {  
 model.addAttribute("member", findMember);  
 System.*out*.println("로그인 됐습니다!");  
 return "redirect:/Member/loginPage";  
 } else {  
 System.*out*.println("아이디, 비밀번호 다시 입력해주세요!");  
 return "redirect:/Member/Login";  
 }  
}

**사용자 회원가입 페이지**



사용자 회원가입을 DB에 넣는 작업을 합니다.

@GetMapping("/mJoin/Join")  
public String insertMember(Member member, Model model) {  
 System.*out*.println("get mapping account !!");  
 System.*out*.println("get방식으로 인한 Join페이지 = 우리가 처음 join페이지를 들어갈 떄는 null값이 뜰 수 밖에없다.");  
 System.*out*.println("왜냐!!?!? 값이 없으니까!");  
 System.*out*.println(member.getName());  
 System.*out*.println(member.getPetW());  
 Member member\_1 = new Member(  
 member.getMember\_Number\_Seq(), // 회원인원 몇명인지!  
 member.getId(), //아이디, 회원등록 번호 : PK  
 member.getPassword(), //비밀번호  
 member.getName(), //이름  
 member.getYear(), //(회원) 생년월일  
 member.getPhoneNumber(), // 핸드폰 번호  
 member.getAddress(), //주소  
 member.getPetT(), // 애견, 애묘 : 종류  
 member.getPetD(), // 애견, 애묘 : 생년월일  
 member.getPetW(), // 애견, 애묘 : 몸무게  
 member.getRole(), // 권한 : 관리자, 사업자, 회원  
 member.getJoinM());// Y는 현재 가입상태 => 모든 회원은 처음 가입할 떄 가입상태 Y로 시작을 한다. /  
 // 탈퇴할 경우는 update로 N으로 수정된다.  
 model.addAttribute("member", member\_1);  
 return "Member/mJoin/Join";  
}

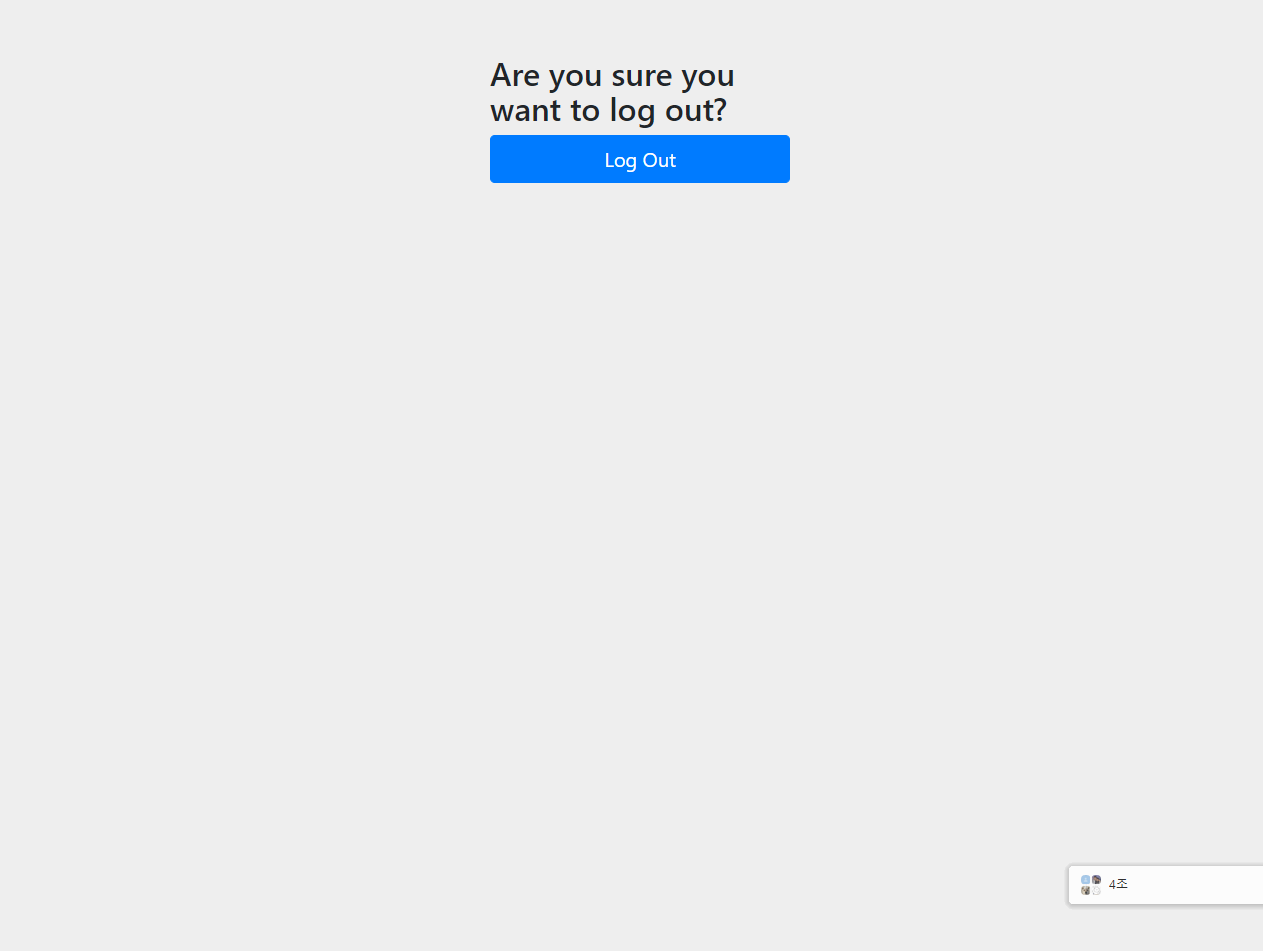
**사용자 수정하기**



로그인하고 사용자 수정하는 페이지

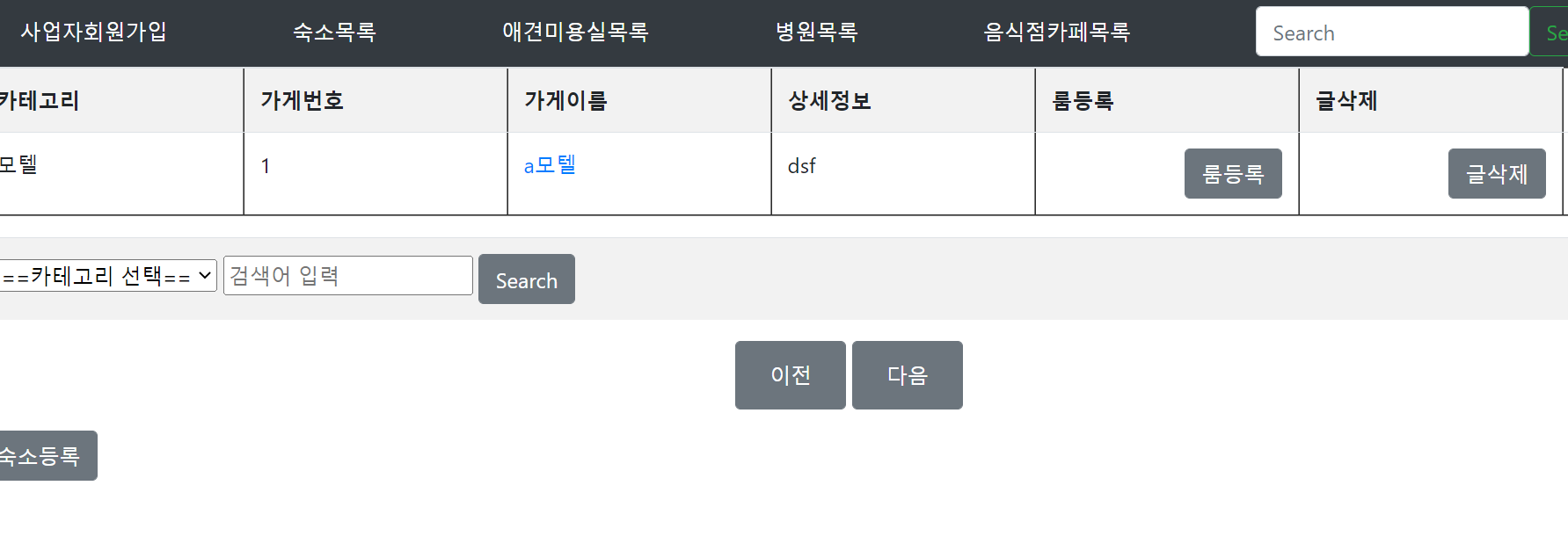
@GetMapping("/mUpdate/Update") //마이 페이지 수정폼  
public String myPage(Member member, Model model) {  
 System.*out*.println("get mapping account !!");  
 System.*out*.println("get방식으로 인한 Join페이지 = 우리가 처음 join페이지를 들어갈 떄는 null값이 뜰 수 밖에없다.");  
 System.*out*.println("왜냐!!?!? 값이 없으니까!");  
 System.*out*.println(member.getName());  
 System.*out*.println(member.getPetW());  
 Member member\_1 = new Member( // null값이라서 값을 받아올 수가 없다....  
 member.getMember\_Number\_Seq(), // 회원인원 몇명인지!  
 member.getId(), //아이디, 회원등록 번호 : PK  
 member.getPassword(), //비밀번호  
 member.getName(), //이름  
 member.getYear(), //(회원) 생년월일  
 member.getPhoneNumber(), // 핸드폰 번호  
 member.getAddress(), //주소  
 member.getPetT(), // 애견, 애묘 : 종류  
 member.getPetD(), // 애견, 애묘 : 생년월일  
 member.getPetW(), // 애견, 애묘 : 몸무게  
 member.getRole(), // 권한 : 관리자, 사업자, 회원  
 member.getJoinM());// Y는 현재 가입상태 => 모든 회원은 처음 가입할 떄 가입상태 Y로 시작을 한다. /  
 // 탈퇴할 경우는 update로 N으로 수정된다.  
 model.addAttribute("member", member\_1);  
 ;  
 return "/Member/mUpdate/Update";  
}  
  
  
@PostMapping("/mUpdate/Update") // 실제로 수정 되는 메소드  
public String updateM(@Valid Member member, Errors errors, Model model) {  
 System.*out*.println("---회원 정보 수정이 이루어 진다.---");  
 System.*out*.println("---check---");  
 System.*out*.println("---PostMapping 실제로 여기서 값이 들어감---");  
 System.*out*.println("아이디 : " + member.getId());  
 System.*out*.println("비밀번호 : " + member.getPassword());  
 System.*out*.println("이름 : " + member.getName());  
 System.*out*.println("폰번 : " + member.getPhoneNumber());  
 System.*out*.println("주소 : " + member.getAddress());  
 System.*out*.println("펫 종류 :" + member.getPetT());  
 System.*out*.println("펫 생년 : " + member.getPetD());  
 System.*out*.println("펫 몸무게 :" + member.getPetW());  
 System.*out*.println("권한 : " + member.getRole());  
 System.*out*.println("가입상태 : " + member.getJoinM());  
 //@Valid : 클라이언트 입력 데이터가 dto클래스로 캡슐화되어 넘어올 때, 유효성을 체크하라는 어노테이션  
 //Member에서 작성한 어노테이션을 기준으로 유효성 체크  
 //여기서 Errors객체는 Member의 필드 유효성 검사 오류에 대한 정보를 저장하고 노출한다.   
 //errors.hasErrors() : 유효성 검사에 실패한 필드가 있는지 확인  
 if (errors.hasErrors()) {  
 //회원가입 실패 시, 입력 데이터를 유지  
 model.addAttribute("member", member);  
 System.*out*.println(member.getId());  
 //회원가입 실패 시, 회원가입 페이지에서 입력했던 정보들을 그대로 유지하기 위해 입력받았던 데이터를 그대로 할당합니다.  
 //insertMember(Member member) 함수에 파라미터를 정의해준 이유입니다.  
 //Validation 관점에서는 필요없는 부분이지만, UX 측면에서 구현해주는 것이 좋다.  
 //물론, thymeleaf에서도 코드가 들어가야 한다.  
  
 //유효성 통과 못한 필드와 메세지를 핸들링  
 Map<String, String> member\_Availability = memberService.member\_Availability(errors);  
 for (String key : member\_Availability.keySet()) {  
 model.addAttribute(key, member\_Availability.get(key));  
 }  
 return "redirect:/Member/mUpdate/Update"; //로그안 페이지로 왜 안돌아가는지 모르겠다.  
 }  
 memberService.updateMember(member);  
  
 return "redirect:/Member/loginPage";

**로그아웃 페이지**



로그아웃 기능을 구현했습니다.

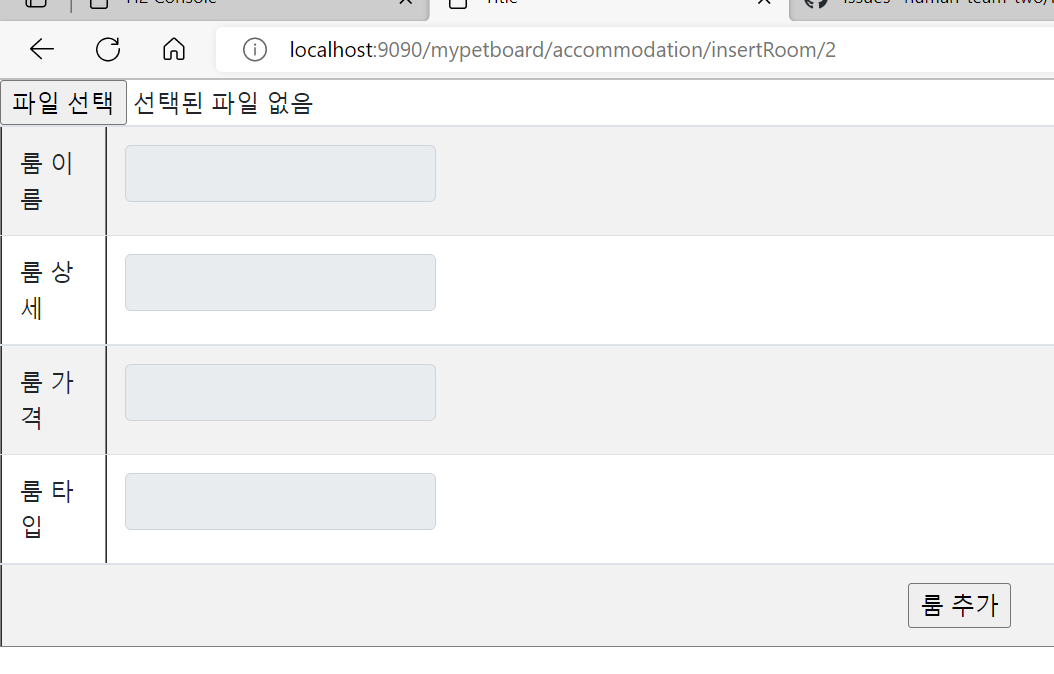
**숙소리스트**



현재 가능한 숙소 리스트가 보여집니다.

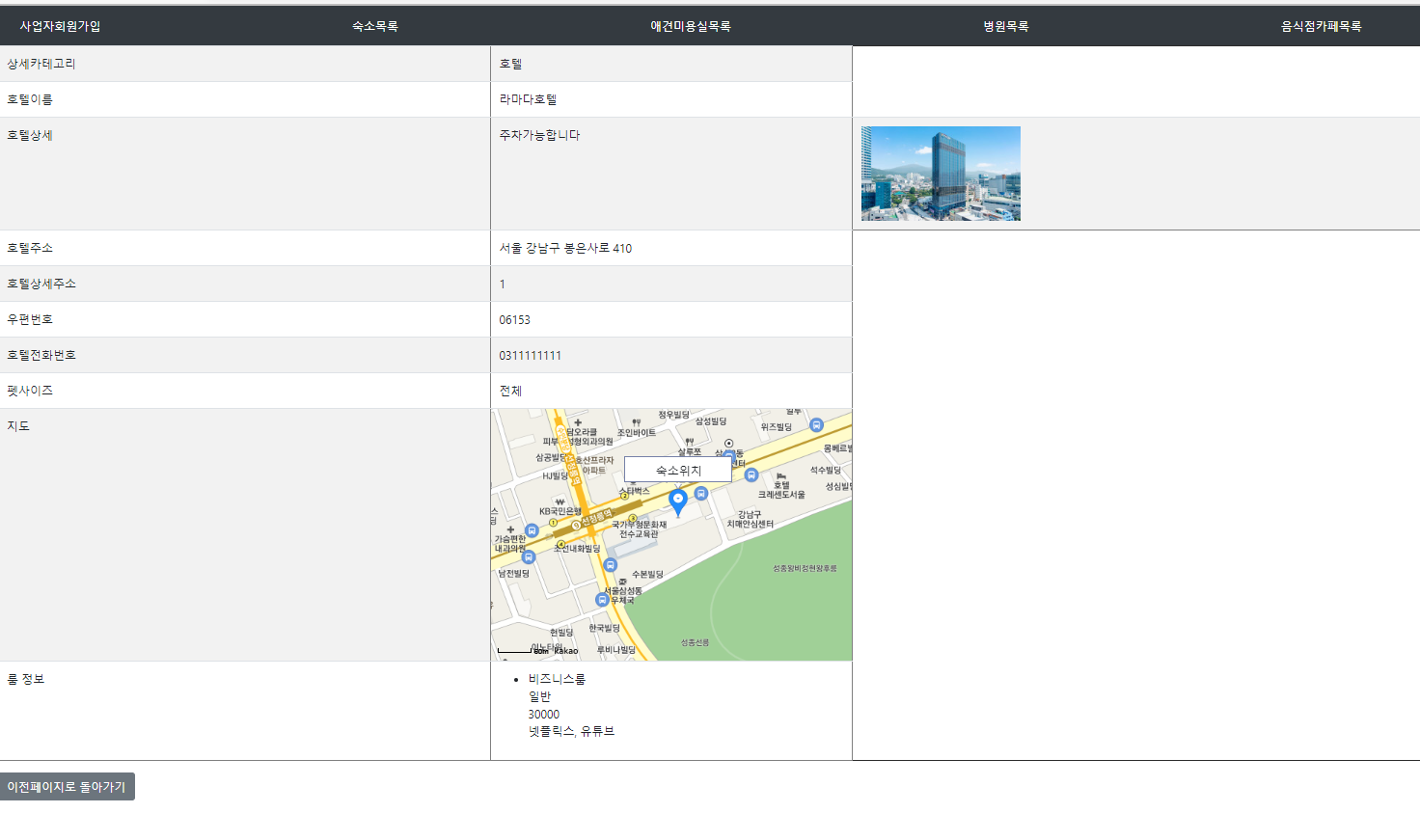
//@PageableDefault(size = 10 \* 한페이지에 10개출력 , sort = "HotelSeq \* HotelSeq기준으로 오름차순 청렬) pageable 페이징처리  
@GetMapping ("/getAccommodationList")  
public String getAccommodationList(  
 @RequestParam(value= "searchKeyword" , required = false , defaultValue = "")String searchKeyword,  
 @RequestParam(value = "detailCategory" , required = false, defaultValue = "")String detailCategory,  
 Model model, @PageableDefault(size = 10 , sort = "seq", direction = Sort.Direction.ASC ) Pageable pageable) {  
 /\*한 화면에 size 글 10개 , sort = "seq" 정렬(시퀀스 번호로 정렬 하겠다) , direction = Sort.Direction.ASC(오름차순으로 정렬) \*/  
 // 객체 초기화  
 Page<Accommodation> accommodation = null;  
  
 // 카테고리분류(search기능) -> 카테고리와 숙소 이름으로 검색  
 if(detailCategory.equals("모텔")){  
 System.out.println("모텔");  
 accommodation = accommodationService.getAccommodationContainsName(searchKeyword, pageable);  
 }  
 else if(detailCategory.equals("호텔")){  
 System.out.println("호텔");  
 accommodation = accommodationService.getAccommodationContainsName(searchKeyword, pageable);  
 }  
 else if(detailCategory.equals("펜션")){  
 System.out.println("펜션");  
 accommodation = accommodationService.getAccommodationContainsName(searchKeyword, pageable);  
 }  
 else {  
 accommodation = accommodationService.getAccommodationContainsName(searchKeyword, pageable);  
 }  
 System.out.println("[detailCategory= " + detailCategory + "]," + " [searchKeyword= " + searchKeyword + "]");  
  
 model.addAttribute("detailCategory", detailCategory );  
 model.addAttribute("searchKeyword", searchKeyword );  
 model.addAttribute("accommodationList", accommodation );  
  
  
 //페이징처리 이전 ,다음  
 model.addAttribute("previous" , pageable.previousOrFirst().getPageNumber());  
 model.addAttribute("next" , pageable.next().getPageNumber());  
  
 return "/mypetboard/accommodation/getAccommodationList";  
}

**숙소 룸 등록 페이지**



숙소 리스트에 출력이 되는 업체의 룸을 등록하는 페이지

@RequestMapping("/insertRoom/{seq}")  
public String insertRoomView(@PathVariable("seq")Long seq, Model model , Room room) {  
  
 Accommodation accommodation = accommodationService.getAccommodationRequest(seq);  
 model.addAttribute("accommodation" , accommodation);  
 System.*out*.println("룸등록page");  
 return "/mypetboard/accommodation/insertRoom";  
}

**숙소 상세페이지** 

숙소리스트에서 클릭하면 업체의 상세페이지가 표시됩니다.

@PostMapping("/insertRoom/{seq}")  
public String insertRoom(@PathVariable("seq")Long seq , Model model ,  
 @RequestParam String roomName,  
 @RequestParam String roomInfo,  
 @RequestParam Integer roomPrice,  
 @RequestParam String roomType  
 ) {  
  
 Accommodation accommodation = accommodationService.getAccommodationRequest(seq);  
 Room room = new Room();  
 room.setRoomName(roomName);  
 room.setRoomType(roomType);  
 room.setRoomPrice(roomPrice);  
 room.setRoomInfo(roomInfo);  
 room.setAccommodation(accommodation);  
  
  
 roomService.insertRoom(accommodation, room);  
  
 model.addAttribute("accommodation" , accommodation);  
  
  
 return String.*format*("redirect:/mypetboard/accommodation/insertRoom/%s", seq);

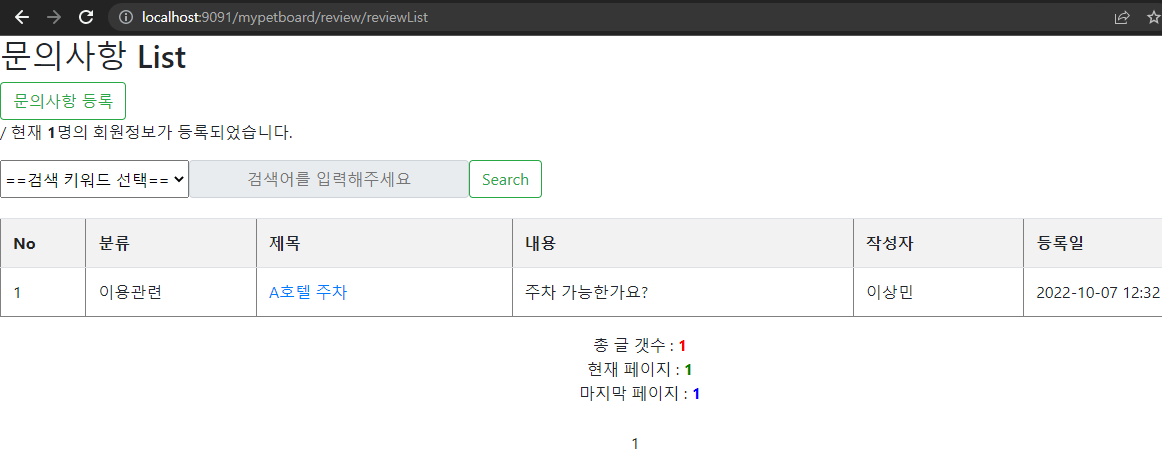
**메인 페이지**

문의사항을 클릭하면 문의사항 List Page로 이동합니다.



**문의사항 리스트 페이지**

문의사항 등록 버튼을 클릭하면 등록 page로 이동합니다.



문의사항 List Controller Source Code입니다.

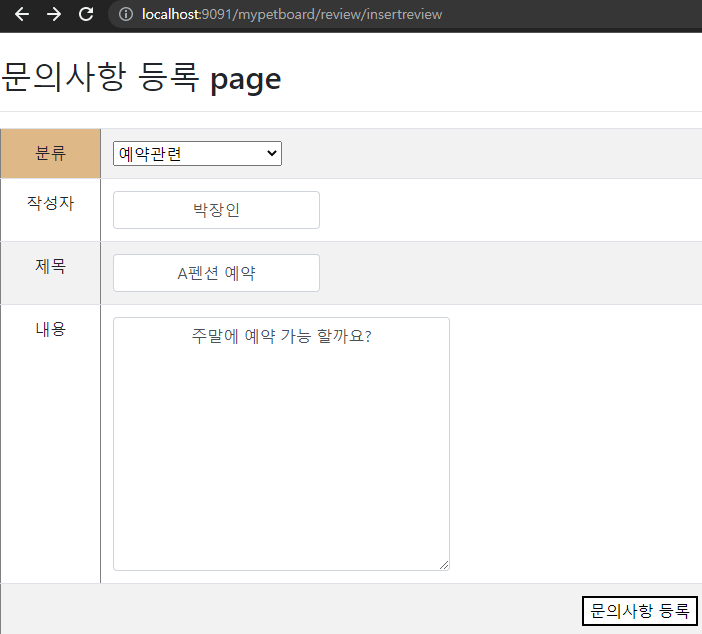
*@GetMapping*("/mypetboard/review/reviewList")  
*public* String getreviewList(  
 *@RequestParam*(value = "scategory",required = *false*, defaultValue = "")String scategory,  
 *@RequestParam*(value = "keyword", required = *false*, defaultValue = "")String keyword,  
 Pageable pageable, Model model) {  
 System.out.println("-----GET getreviewList");  
 System.out.println("scategory= " + scategory);  
 System.out.println("keyword= " + keyword);  
  
 Page<Review> reviews = *null*;  
 *if*(scategory.equals("title")){  
 reviews = reviewService.findByTitle(pageable, keyword);  
 }*else if* (scategory.equals("content")) {  
 reviews = reviewService.findByContent(pageable, keyword);  
 } *else if* (scategory.equals("writer")) {  
 reviews = reviewService.findByWriter(pageable, keyword);  
 }*else*{  
 reviews = reviewService.findAll(pageable);  
 }  
 model.addAttribute("reviewList", reviews);  
 model.addAttribute("scategory", scategory);  
 model.addAttribute("keyword", keyword);  
 *return* "/mypetboard/review/reviewList";  
}

문의사항 List Service Source Code입니다.

*public* List<Review> getReviewList(Review review){  
 System.out.println("-----service getReviewList-----");  
 *return* (List<Review>) reviewRepo.findAll();  
}

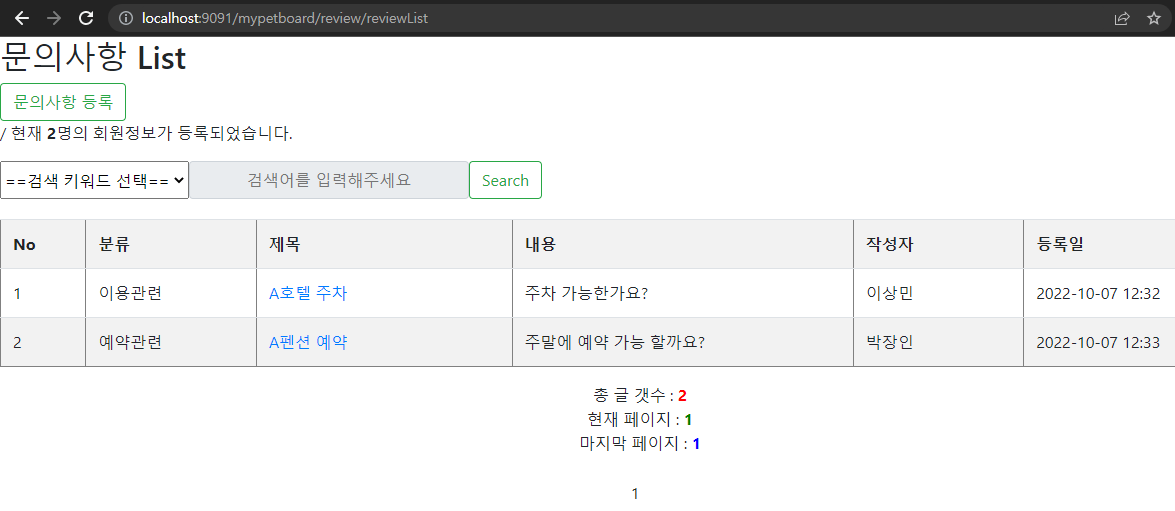
**문의사항 등록 페이지**

작성자, 제목, 내용을 등록후 문의사항 등록 버튼을 클릭하면



입력한 내용이 List에 표시 되는 것을 볼 수 있습니다.

제목의 A호텔 주차 내용을 클릭하게 되면 세부내용 page로 이동하고 등록한 글을 수정할 수도 있습니다.



문의사항 등록page Controller Source Code입니다.

*@GetMapping*("/mypetboard/review/insertreview")  
*public void* insertReview() {  
 System.out.println("-----GET insertreview");  
}

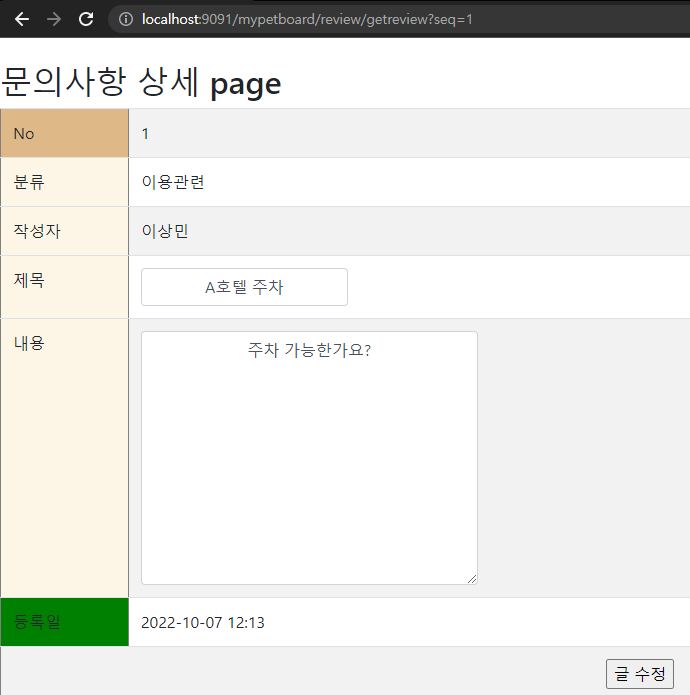
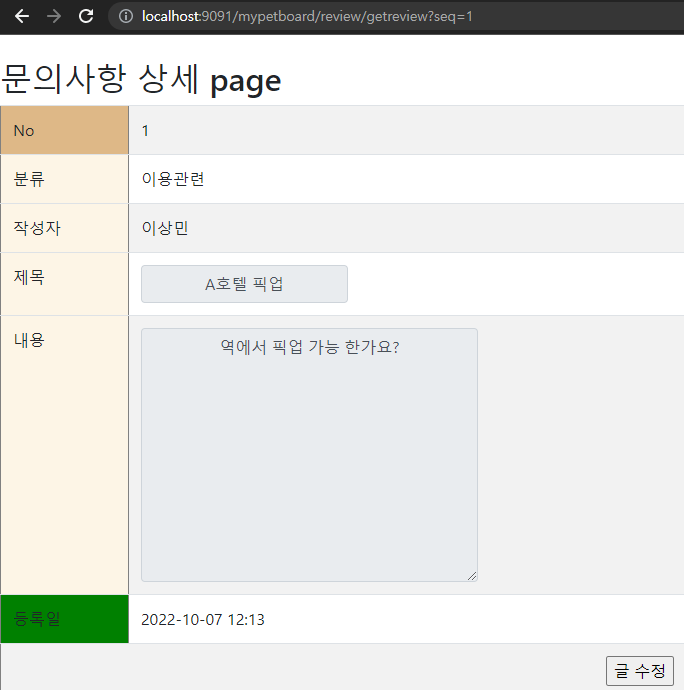
*@PostMapping*("/mypetboard/review/insertreview")  
*public* String insertReview(Review review) {  
 System.out.println("-----POST insertreview");  
 review.setCreateDate(*new* Date());  
 reviewService.insertReview(review);  
 System.out.println("-----insertreview= " + review);  
 *return* "redirect:/mypetboard/review/reviewList";  
}

문의사항 등록page Service Source Code입니다.

*public void* insertReview(Review review){  
 System.out.println("-----service insertReview-----");  
 reviewRepo.save(review);  
}

**문의사항 상세 페이지**

세부내용 page로 이동 후 오른쪽 사진처럼 제목, 내용을 수정 후 글 수정을 클릭하면

List Page에서 변경되는 것을 볼 수 있습니다.



문의사항 상세page Controller Source Code입니다.

*@GetMapping*("/mypetboard/review/getreview")  
*public* String getreview(Model model, Review review) {  
 System.out.println("-----GET getreview");  
 model.addAttribute("reviewList", reviewService.getReview(review));  
 *return* "/mypetboard/review/getreview";  
}

문의사항 상세page Service Source Code입니다.

*public* Review getReview(Review review){  
 System.out.println("-----service getReview-----");  
 *return* reviewRepo.findById(review.getSeq()).get();  
}

문의사항 수정 Controller Source Code입니다.

*@PostMapping*("/mypetboard/review/updateReview")  
*public* String updatereview(Review review) {  
 System.out.println("-----POST updatereview");  
 reviewService.updateReview(review);  
 *return* "redirect:/mypetboard/review/reviewList";  
}

문의사항 수정 Service Source Code입니다.

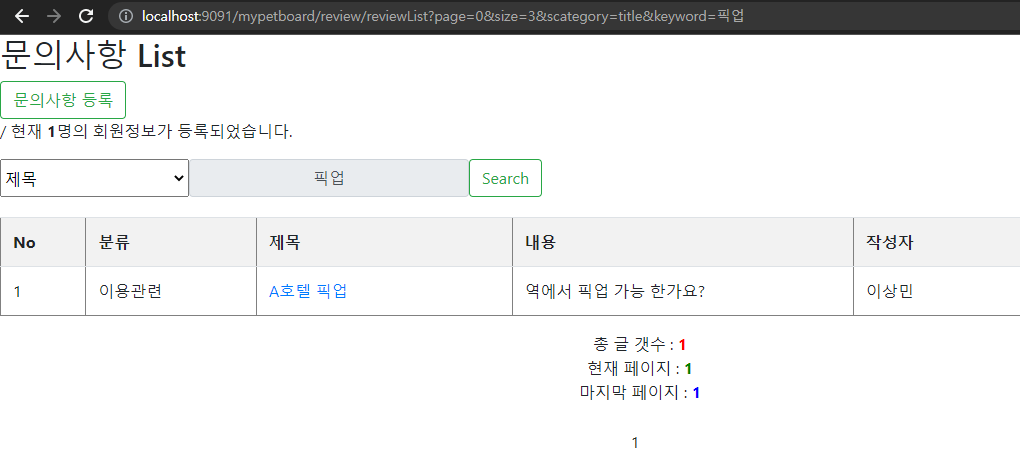
*public void* updateReview(Review review){  
 System.out.println("-----service updateReview-----");  
 Review findReview = reviewRepo.findById(review.getSeq()).get();  
 findReview.setTitle(review.getTitle());  
 findReview.setContent(review.getContent());   
 reviewRepo.save(findReview);  
}

**키워드 선택 후 검색기능**

원하는 키워드 선택 후 검색어를 입력 후 Search버튼을 누르면 선택한 키워드에 입력한 내용을 검색 할 수 있습니다.



키워드 제목 선택 후 픽업으로 검색한 Page입니다.



문의사항 검색 Controller Source Code입니다.

*@GetMapping*("/mypetboard/review/reviewList")  
*public* String getreviewList(  
 *@RequestParam*(value = "scategory",required = *false*, defaultValue = "")String scategory,  
 *@RequestParam*(value = "keyword", required = *false*, defaultValue = "")String keyword,  
 Pageable pageable, Model model) {  
 System.out.println("-----GET getreviewList");  
 System.out.println("scategory= " + scategory);  
 System.out.println("keyword= " + keyword);  
  
 Page<Review> reviews = *null*;  
 *if*(scategory.equals("title")){  
 reviews = reviewService.findByTitle(pageable, keyword);  
 }*else if* (scategory.equals("content")) {  
 reviews = reviewService.findByContent(pageable, keyword);  
 } *else if* (scategory.equals("writer")) {  
 reviews = reviewService.findByWriter(pageable, keyword);  
 }*else*{  
 reviews = reviewService.findAll(pageable);  
 }  
 model.addAttribute("reviewList", reviews);  
 model.addAttribute("scategory", scategory);  
 model.addAttribute("keyword", keyword);  
 *return* "/mypetboard/review/reviewList";  
}

문의사항 검색 Service Source Code입니다.

*public* Page<Review> findByTitle(Pageable pageable, String keyword) {  
 System.out.println("-----service findByTitle-----");  
 *return* reviewRepo.findByTitleContaining(pageable, keyword);  
}

*public* Page<Review> findByContent(Pageable pageable, String keyword) {  
 System.out.println("-----service findByContent-----");  
 *return* reviewRepo.findByContentContaining(pageable, keyword);  
}

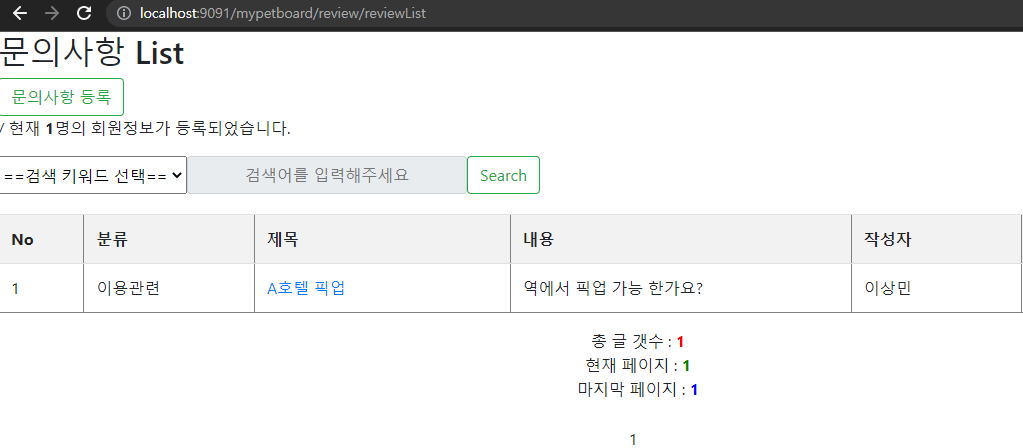
*public* Page<Review> findByWriter(Pageable pageable, String keyword) {  
 System.out.println("-----service findByWriter-----");  
 *return* reviewRepo.findByWriterContaining(pageable, keyword);  
}

*public* Page<Review> findAll(Pageable pageable){  
 System.out.println("-----service findAll-----");  
 *return* reviewRepo.findAll(pageable);  
}

**글 삭제 기능**

해당 글에 글 삭제 버튼을 클릭하면 해당 글이 삭제 되는 것을 볼 수 있습니다.





문의사항 삭제 Controller Source Code입니다.

*@GetMapping*("/mypetboard/review/removeReview")  
*public* String removereview(Review review) {  
 System.out.println("-----GET removereview");  
 reviewService.removeReview(review);  
 *return* "redirect:/mypetboard/review/reviewList";  
}

문의사항 삭제 Service Source Code입니다.

*public void* removeReview(Review review){  
 System.out.println("-----service deleteReview------");  
 reviewRepo.deleteById(review.getSeq());  
}