

박종진

- 공공DB를 이용한 영문  
민물 & 바다 낚시  
+ 자유 게시물 업로드  
+ 챗봇  
+ 채팅

# 프로젝트 일정- (전체 3주)

1번째 주:

- 공공DB 데이터 분석+ 조회 기능 구현.
- 유저 게시물 CRUD 기능 구현 + 댓글 Insert 구현
- summernote 기능 추가, 파일업로드 기능 추가

# 프로젝트 일정- (전체 3주)

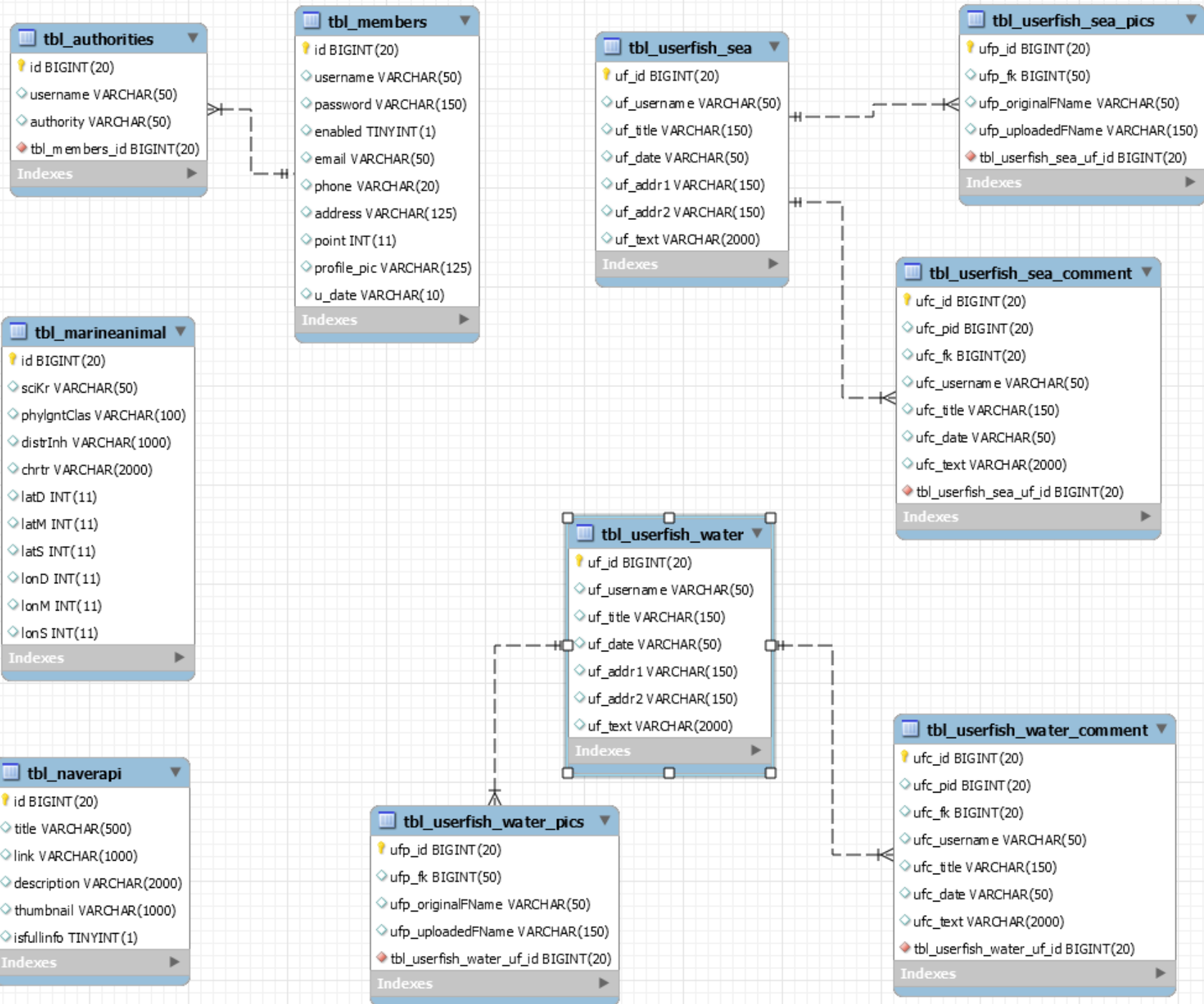
2번째 주:

- 회원가입 기능 구현
- Jasypt, Bcrypt, Spring Validate 기능 추가
- 전체 채팅 기능 구현
- 페이지네이션, 검색 기능 추가

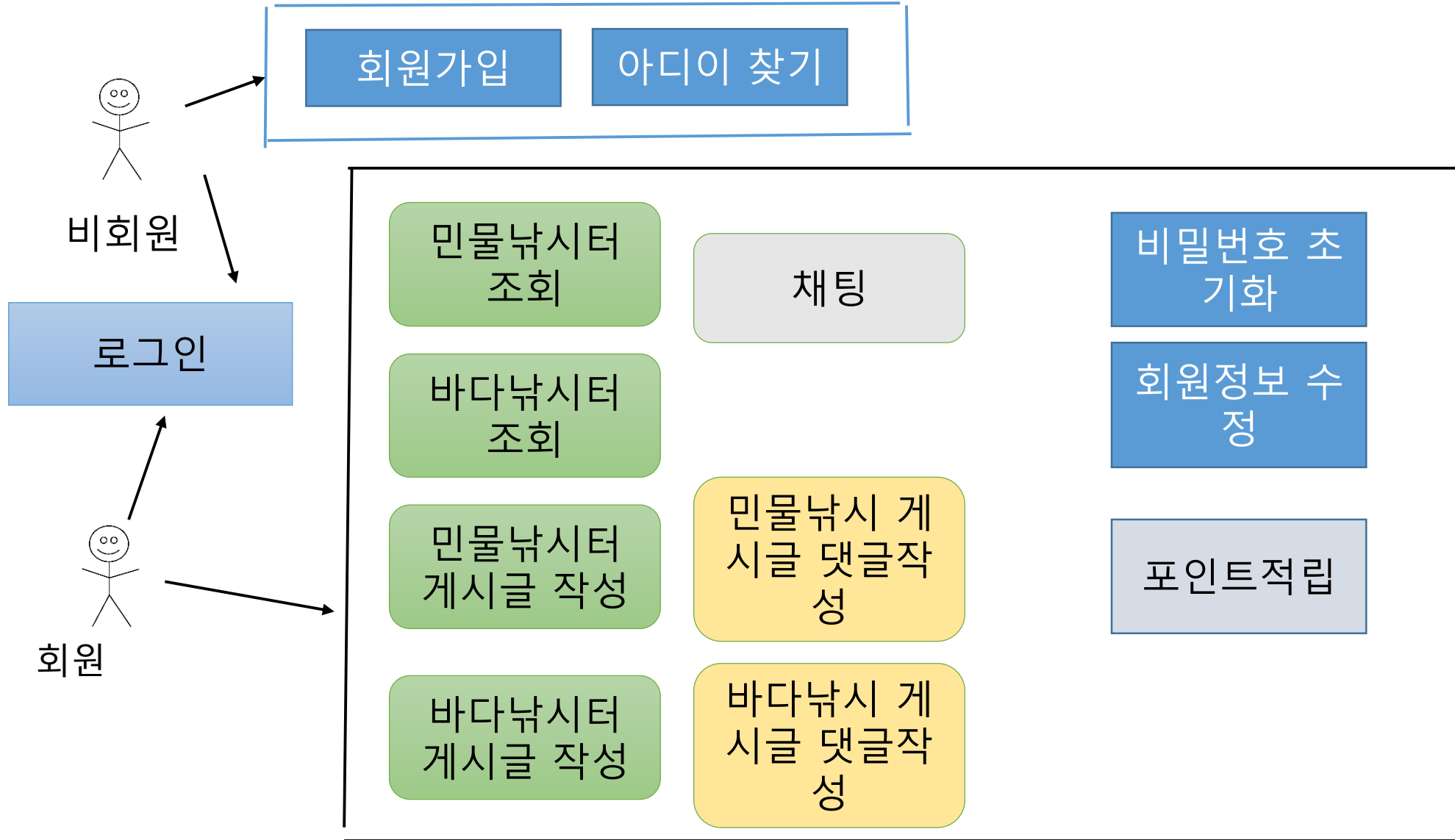
# 프로젝트 일정- (전체 3주)

3번째 주:

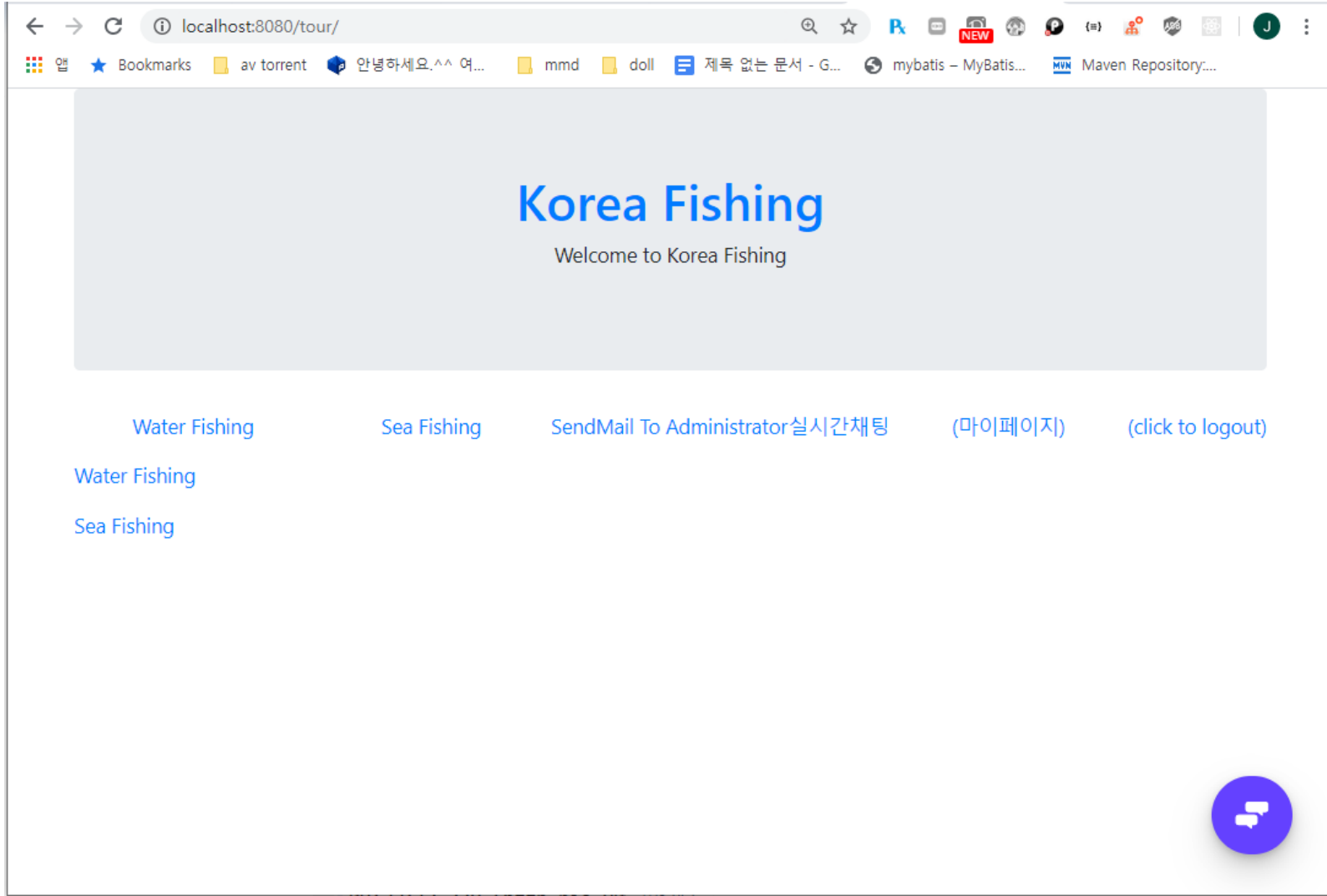
- 마이페이지(회원정보 수정, 비밀번호 초기화, 아이디 찾기) 기능 구현
- 포인트 적립기능 구현
- Tour API, 해양생물 API, NaverAPI 데이터 연동 후 해양생물 DB 구축
- 이메일 인증 추가.
- 운영자에게 이메일 보내기 기능 구현.
- Spring Security 기능 추가.



# UML



# 메인화면




# 공공DB 영문 낚시터

localhost:8080/tour/fish/water?searchOption=

Water Fishing Sea Fishing SendMail To Administrator실시간채팅 (마이페이지) (click to logout)

검색 제목으로 검색


## Andongho Lake (안동호)



Andongho Lake (안동호)  
Hoban-ro, Andong-si, Gyeongsangbuk-do

Detail

## Yedang Reservoir (예당저수지(예당관광지))



Yedang Reservoir (예당저수지(예당관광지))  
33, Sajik-ro, Yesan-gun, Chungcheongnam-do

Detail

[처음]<1>[끝]



# 공공DB 영문 API- Controller

```
@RequestMapping(value = "/water",method=RequestMethod.GET)
public String waterFishing(@RequestParam(value = "pageno",defaultValue = "1") String strPageno,
    APISearchVO apiSearchVO,Model model) {
    int pageno=Integer.valueOf(strPageno);
    //String fishList=tourService.getFishingAreaBased("areaBased","water",strPageno);
    //return fishList;
    List<FishAreaBasedVO> fishList=tourService.getFishingAreaBased("areaBased","water",strPageno);
    if(apiSearchVO.getSearchOption().equalsIgnoreCase("titleSearch")) {
        fishList=searchApiservice.findAPIbyTitle(fishList, apiSearchVO.getInputStr());
    }
    PageDTO pageDTO=pagiService.makePageNation(fishList.size(), pageno, 10);
    model.addAttribute("MODE", "water");
    model.addAttribute("fishList", fishList);|
    model.addAttribute("PAGE", pageDTO);
    return "fishing/fishingList";
}
```

apiSearchVO : 검색유지 페이지네이션을 위한 VO. SessionAttribute을 사용.  
strPageno, PageDTO : 페이지네이션 구현을 위한 변수+객체  
findAPIbyTitle : api로 받아온 데이터에서 검색 기능

# 공공DB 영문 API- 제목검색 service

```
public class SearchApiDataByTitleImp implements SearchApiDataByTitle{  
    public List<FishAreaBasedVO> findAPIbyTitle(List<FishAreaBasedVO> dataList,String strInput) {  
        strInput=strInput.trim();  
        List<FishAreaBasedVO> newList=new ArrayList<FishAreaBasedVO>();  
        for(FishAreaBasedVO vo:dataList) {  
            if(vo.getTitle().toLowerCase().contains(strInput.toLowerCase())) {  
                newList.add(vo);  
            }  
        }  
        return newList;  
    }  
}
```

dataList : 공공DB로부터 받은 data

java의 contains() 를 사용하여 특정 문자열이 포함되어있는 리스트 추출

# 공공DB 영문 API- service

```
public List<FishAreaBasedVO> getFishingAreaBased(String urlLoc, String type, String strPageno) {
```

```
    String strQuery = getAreaBasedQuery(type, strPageno);
```

```
    HttpHeaders header = new HttpHeaders();
```

공공DB 에서 제공하는 pagination을 사용하기

```
    header.setAccept(Collections.singletonList(org.springframework.http.MediaType.APPLICATION_XML));
```

```
    HttpEntity<String> entity = new HttpEntity<String>(header);
```

```
    RestTemplate restTemp = new RestTemplate();
```

```
    URI restURI = null;
```

위해 URI에 pageno를 포함

```
    ResponseEntity<String> strResult = null;
```

```
    ResponseEntity<FishAB_RestResponse> result = null;
```

```
    try {
```

```
        restURI = new URI(strQuery);
```

```
        //strResult = restTemp.exchange(restURI, HttpMethod.GET, entity, String.class);
```

```
        result = restTemp.exchange(restURI, HttpMethod.GET, entity, FishAB_RestResponse.class);
```

```
        //String strRV0 = strResult.getBody();
```

```
        FishAB_RestResponse rV0 = result.getBody();
```

```
        List<FishAreaBasedVO> fishingList = rV0.getBody().getItems().getItem();
```

```
        //return strRV0;
```

```
        return fishingList;
```

```
    } catch (URISyntaxException e) {
```

```
        // TODO Auto-generated catch block
```

```
        e.printStackTrace();
```

```
    }
```

```
    return null;
```

```
}
```

RestTemplate을 이용하여 공공DB API 연결후  
데이터 받아옴

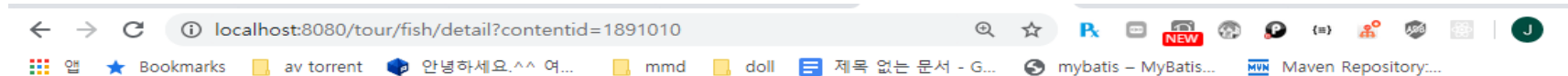
# 공공DB 영문 API- 제목검색 jsp

```
<c:if test= "${MODE=='water' }">
<section class= "apiSearch">
  <form method= "get" action= "${rootPath }/fish/water">
    <label for= "searchOption">검색</label> <select name= "searchOption">
      <option value= "titleSearch">제목으로 검색</option>
    </select> <input name= "inputStr">
    <button id= "userSearch">검색</button>
  </form>
  <br />
</section>
```

HTML select tag+input tag의 값을 form 태그를 이용하여 컨트롤러로 전달.

controller에선 위의 정보를 바탕으로 검색을 실행.

# 공공DB 낚시터 상세 페이지



area code: 35

city code: 11

Addr 1: Hoban-ro, Andong-si, Gyeongsangbuk-do

Addr 2:

zipcode: 36710

128.7719600867

36.5843448602

mlevel: 6



description : Known as a popular fishing spot and recreation site, Andongho Lake is an man-made lake created by obstructing the upper stream of the Nakdonggang River, as part of the government's Four Major Rivers Restoration Project. Andong Dam is the second largest dam in Korea, following Soyanggang Dam. The x\_height of the dam is 83 meters and the length is 612 meters.

The landscape surrounding Andongho Lake is nothing short of magnificent. There is a dock for the boats on the lake, which is also popular as a fishing destination. Bass caught here are especially strong and tasty. An international bass fishing tournament is held here every year. There are also tourist attractions nearby, such as the Dosanseowon Confucian Academy and Andong Hahoe Folk Village.


Andongho Lake is known for its incomparable scenery, and is considered one of the most famous tourist sites in the Hoban Region. The nearest tourist sites include Dosanseowon Confucian Academy, Seokbinggo (Stone Ice Storage). There are plenty of other sites to visit as well, such as Andong Folk Village and its museum, where the historical Korean drama "Taejo Wang Geon" was filmed.



# 공공DB 낚시터 상세 페이지 + 해양생물 데이터

이곳의 생태계는?



기수우렁이







참서대

테두리고둥

갯닢싸리

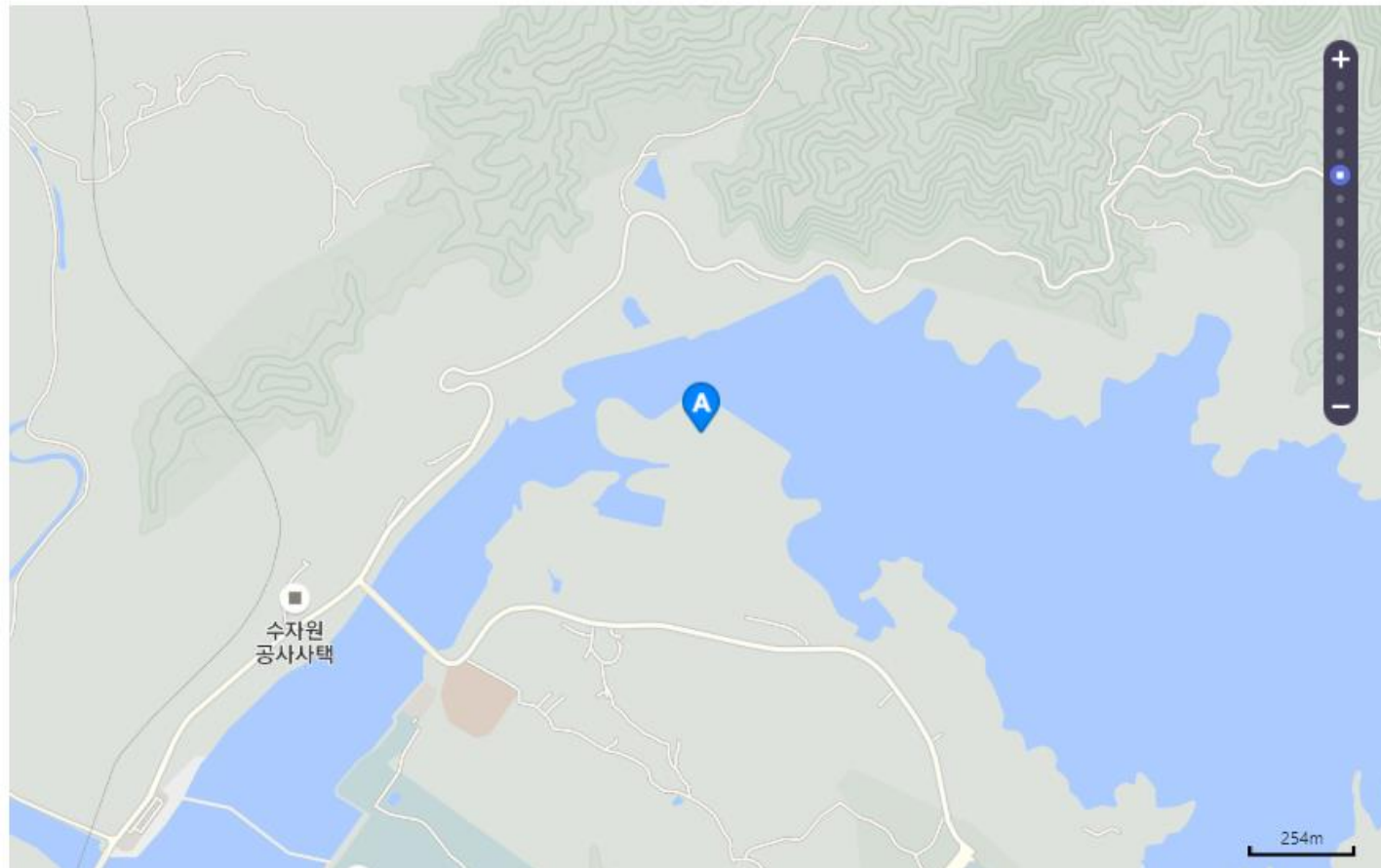
< >

1 / 5

# 공공DB 낚시터 상제 페이지 - Tmap

map: (it only shows in korean... sorry for inconvenience... for detailed explanation, please use google map

or navigation application on your gadget.)





# 유저 낚시터 게시물

mode:water

## User Fishing Spot

검색

제목으로 검색 ▾

검색

t2

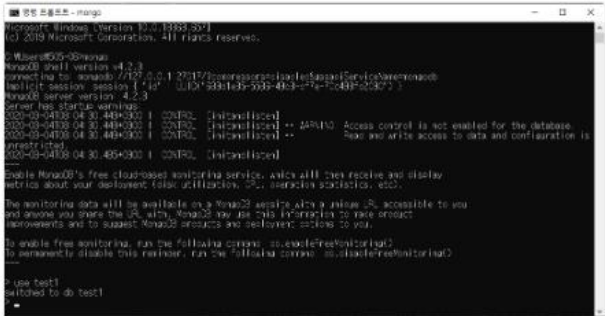


t2

Detail

insert new water fishing spot

1u



1u

Detail



# 유저 낚시터 게시물- dao Pagination+ join

```
@Select("select * from tbl_userfish_sea_pics where ufp_fk=#{uf_id}")
public List<FishUserSeaPicsVO> findPicsByFK(long uf_id);

@Select("select * from tbl_userfish_sea where uf_id=#{uf_id}")
public FishUserSeaVO findById(Long uf_id);

@Select("select * from tbl_userfish_sea where uf_title like CONCAT ('%', #{uf_title} ,'%') limit #{pageno},#{itemLimit} ")
@Results( //constraint 조건이 걸린
    value= {
        @Result(property="uf_id", column = "uf_id"),
        @Result(property = "picsList", column = "uf_id", javaType = List.class,
            many=@Many(select = "findPicsByFK"))
    }
)
public List<FishUserSeaVO> findByTitle(@Param("uf_title") String uf_title,@Param("pageno") int pageno,@Param("itemLimit") :
```

@Results : 부모 테이블과 자식 테이블을 Join 하기위한 Annotation. tbl\_userfish\_sea 와 tbl\_userfish\_sea\_pics 테이블이 자동으로 join.

limit #{pageno},#{itemLimit} : DB에서 제공하는 페이지네이션 기능

# 유저 낚시터 게시물-controller

```
@RequestMapping(value = "/findAndShow", method = RequestMethod.GET)
public String findAndShow(UserSearchVO userSearchVO,
    @RequestParam(value = "pageno", defaultValue = "1") String strPageno, Model model) {
    int pageno = Integer.valueOf(strPageno);
    int maxListSize=-1;
    List<FishUserSeaVO> userList = new ArrayList<FishUserSeaVO>();
    if (userSearchVO.getSearchOption().equalsIgnoreCase("titleSearch")) {
        maxListSize=uSeaService.countFindByTitle(userSearchVO.getInputStr());
        userList = uSeaService.findByTitle(userSearchVO.getInputStr(), (pageno-1)*itemLimit, itemLimit);
    } else {
        maxListSize=uSeaService.countFindAll();
        userList = uSeaService.findAll((pageno-1)*itemLimit,itemLimit);
    }
    if (!userList.isEmpty()) {
        try {
            for (FishUserSeaVO vo : userList) {
                if (vo.getPicsList().size() > 0 && vo.getPicsList() != null) {
                    vo.setMainPic(vo.getPicsList().get(0).getUfp_uploadedFName());
                }
            }
        } catch (Exception e) {
            // TODO: handle exception
        }
    }
    }// 메인 사진 고르기
    PageDTO pageDTO=pagiService.makePageNation(maxListSize, pageno, itemLimit);
```

userSearchVO :  
검색유지 페이지네이  
이션 VO.  
sessionAttribute에  
등록됨

strPageno : 페이지네이  
션을 구현할때 필요한  
변수

PageDTO : 페이지 하단  
( 1 2 3 4 5 ) 번호를 표  
시할때 필요한 정보를  
담은 VO

# 유저 낚시터 상세 페이지

uf\_username: admin

date: 2020-05-22

uf\_addr1: t2

uf\_addr2: t2

content:

Water Fishing

Sea Fishing

멀티파일 업로드 가능

섬머노트+파일업로드 가능

1. click a menu above

t2

insert new water fishing spot

**2. click this button  
at the bottom**

t2

t2



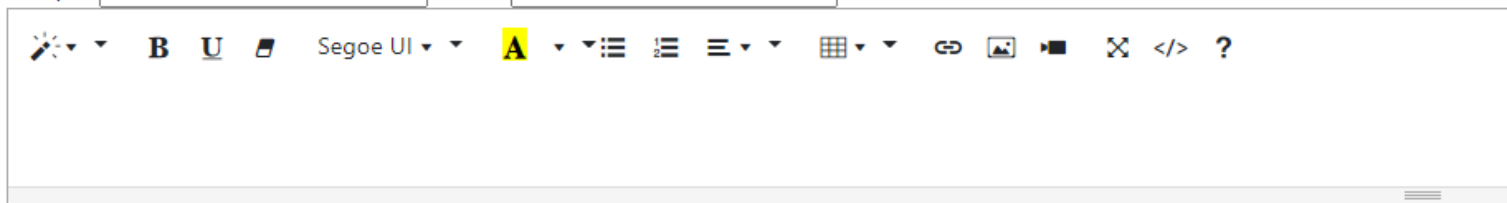
# 유저 낚시 터 상세 페이지 + 무한댓글 + 페이지 네이션

id-37, pid-0 usr-a : text-



ufc\_pid:37

ufc\_fk:4



reply

id-38, pid-0 usr-a : text-



id-40, pid-38 usr-a : text- re:



id-39, pid-0 usr-a : text-



# 유저 낚시터 상세페이지안의 댓글 controller

```
@RequestMapping(value = "/comments", method = RequestMethod.GET)
public String comments(@RequestParam("ufc_fk") String strFk, @RequestParam(value = "pageno", defaultValue = "1") String strPageno, Model model) {
    int pageno=Integer.valueOf(strPageno);
    long ufc_fk = -1;
    int maxListSize=-1;
    try {
        ufc_fk = Long.valueOf(strFk);
    } catch (Exception e) {
        // TODO: handle exception
        log.debug("cmt 문자열 변환중 오류: " + ufc_fk);
    }
    maxListSize=uSCommentService.countFindByFk(ufc_fk);
    List<FishUserSeaCommentVO> commentList = uSCommentService.findByFk(ufc_fk, (pageno-1), itemLimit);
    model.addAttribute("commentList", commentList);

    PageDTO pageDTO=pagiService.makePageNation(maxListSize, pageno, itemLimit);
    model.addAttribute("PAGE", pageDTO);
    return "fishing/userComment";
}
```

pageno 와 itemLimit을 이용하여 java code  
pagination 구현

pageDTO : 페이지 하단 ( 1 2 3 4 5 ) 표시할때 필요한 정보를 담은 VO

# 유저 낚시터 게시물- java pagination

```
public List<FishUserSeaCommentVO> findByFk(long ufc_fk, int pageno, int itemLimit) {  
    // 댓글중 타 댓글에 의존이 없는 (pid=0)인 댓글들 뽑음  
    List<FishUserSeaCommentVO> commentList=scDao.findByFk(ufc_fk);  
  
    //댓글 계층구조가 다 완성된 list를 담은 객체  
    List<FishUserSeaCommentVO> sortedCommentList=new ArrayList<FishUserSeaCommentVO>();  
  
    //재귀 함수를 통해서 pid 가 없는 답글 기준으로...  
    for(FishUserSeaCommentVO vo:commentList) {  
        sortedCommentList.addAll(sortReplyHierarchy(vo, 0));  
    }  
    List<FishUserSeaCommentVO> pgStedCmtList=new ArrayList<FishUserSeaCommentVO>();  
    for(int i=(pageno*itemLimit);i<(pageno+1)*itemLimit;i++) {  
        try {  
            pgStedCmtList.add(sortedCommentList.get(i));  
        } catch (Exception e) {  
            // TODO: handle exception  
        }  
    }  
    return pgStedCmtList;  
}
```

DB 에서 가져온 List 에서, pageno 만큼 건너뛰고, itemLimit만큼 데이터를 따로 뽑아서 pgStedCmtList에 담음.  
그럼 Pagination을 Java 코드에서 구현 가능.

# 유저 낚시터 게시물- 무한댓글

```
private List<FishUserSeaCommentVO> sortReplyHeriarchy(FishUserSeaCommentVO cmtVO, int depth){  
    // pid 0 인 댓글 하나 기준으로 child 댓글 heriarchy 를 만들 list  
    List<FishUserSeaCommentVO> retList=new ArrayList<FishUserSeaCommentVO>();  
    if(depth>0) {  
        String c_header="&nbsp;";  
        for(int i=0;i<depth;i++) {  
            c_header+=" re: ";  
        }  
        String moddedText=c_header+cmtVO.getUfc_text();  
        cmtVO.setUfc_text(moddedText);  
    }  
    // 매개 변수로 받은 댓글vo(부모)vo를 list에 먼저 추가 시켜줌  
    retList.add(cmtVO);  
  
    //부모 id와 같은 pid를 가진 child 댓글들 뽑음.  
    List<FishUserSeaCommentVO> tempList=scDao.findByPId(cmtVO.getUfc_id());  
    // child 댓글이 더이상 없으면 재귀함수 exit  
    if(tempList.size()<1) return retList;  
  
    for(FishUserSeaCommentVO vo:tempList) {  
        // 각 child 댓글들을 기준으로 또 child-child 댓글이 있나 재귀함수로 확인.  
        retList.addAll(sortReplyHeriarchy(vo, depth+1););  
    }  
    return retList;  
}  
}
```

계층 구조가 완성된 댓글 list를  
return

부모 댓글이 없는 댓글만 DB에서 뽑은후,  
sortRepltheriarchy() 에 전달.,

부매댓글id 를 pid로 가진  
자식댓글을 찾음

자식댓글의 자식댓글을 찾기위해  
자식댓글을 부모로 기준잡고 재귀함수 호출

# 유저 낚시터 상세 페이지- ajax 댓글 페이지네이션

```
$(document).on("click", ".page-item", function(){
    $.ajax({
        url: "${rootPath}/fishUserWater/comments",
        data: {ufc_fk:ufc_fk,pageno:$(this).data("page")},
        type: "GET",
        success: function(result){
            $(".comments").html("")
            $(".comments").html(result)
        }
    })//페이징 민물 낚시글 답글들 가져오기
})
```

댓글 리스트를 불러오는 컨트롤러에 현 게시글의 id, pageno 를 ajax 코드로 전달해줌.



# 유저 낚시터 상세 페이지- 댓글 form 동적생성

```
<c:forEach items="${commentList}" var="vo">
<section class="preply" id="${vo.ufc_id}" data-ufc_pid="${vo.ufc_pid}" data-ufc_fk="${vo.ufc_fk}">
id-${vo.ufc_id }, pid-${vo.ufc_pid } usr-${vo.ufc_username } : text-${vo.ufc_text }

</section>
<article id="${vo.ufc_id }a" class="replyreplyAtc">
```

jsp 페이지에서 특정영역 tag에 id를 동적으로 부여해줌

```
$(document).on("click", ".preply", function(){
    $(".replyreplyAtc").html("")
    let id=$(this).attr("id")
    let ufc_pid=$(this).data("ufc_pid")
    let ufc_fk=$(this).data("ufc_fk")
    $.ajax({
        url:"${rootPath}/fishUserWater/replyForm",
        data:{ufc_pid:ufc_pid, ufc_fk:ufc_fk},
        type:"GET",
        success:function(result){
            $("#"+id+"a").html("")
            $("#"+id+"a").html(result)
        }
    })
})
```

특정 tag 클릭시 id를 뽑아옴

동적으로 생성된 id를 가진 tag에 html 코드(form) 를 뿌림

# 유저 닉시터 작성 form

## form tag 안에서의 멀티파일 업로드 섬머노트를 이용한 멀티파일 업로드

# 유저 낚시터 작성 controller

```
@Secured(value = {"ROLE_ADMIN", "ROLE_USER"})
@RequestMapping(value = "/seaInsert", method = RequestMethod.POST)
public String insert(FishUserSeaVO userVO, MultipartHttpServletRequest uploaded_files,
    Model model) {
    MemberVO memberVO = (MemberVO) SecurityContextHolder
        .getContext()
        .getAuthentication().getPrincipal();
    int ret = uSeaService.insert(userVO, memberVO.getUsername());
    long fk = userVO.getUf_id();

    // 파일 업로드와 파일 이름을 DB에 저장할 같이함.
    fUploadService.filesUp(uploaded_files, "sea", fk);
    return "redirect:/fish/sea";
}

<insert id="insert" parameterType="fishUserSeaVO" useGeneratedKeys="true" keyProperty="uf_id">
```

1. 게시글 insert 수행
2. insert 된 게시글의 id를 userGeneratedKeys 기능을 사용하여 리턴받음
3. 리턴된 게시글 id 를 파일업로드 서비스에 넘겨줌
4. 파일을 서버에 업로드하고 DB에 정보 저장

# 유저 뉁시터 파일업로드 service

```
public String filesUp(MultipartHttpServletRequest uploaded_files, String whichTable, long fk) {  
    // uploaded_files.GetFiles("uploaded_files") 이부분은 jsp form input 에서 지정한 name과  
    // 동일해야함  
    for (MultipartFile file : uploaded_files.GetFiles("uploaded_files")) {  
        if (file.isEmpty())  
            return null;  
        fileUp(file, whichTable, fk);  
    }  
    return null;  
}
```

멀티파일 업로드를 위해 for 루프를 사용.  
싱글파일 업로드를 forloop 만큼 반복

```
public String fileUp(MultipartFile uploadedFile, String whichTable, long fk) {  
    if (uploadedFile.isEmpty())  
        return null;  
  
    log.debug("## uploadedFile: "+uploadedFile.getOriginalFilename().toString());  
    // upload할 filePath가 있는지 확인을 하고  
    // 없으면 폴더를 생성  
    File dir = new File(filePath);  
    if (dir.exists()) {  
        dir.mkdirs();  
    }  
  
    // 파일이름을 추출(그림.jpg)
```

싱글파일 upload method.  
파일의 원본 이름을 뺌.

# 유저 뉁시터 파일업로드 service

```
// 파일이름을 추출(그림.jpg)
```

```
String originalFileName = uploadedFile.getOriginalFilename();
```

```
// UUID가 부착된 새로운 이름을 생성(UUID그림.jpg)
```

```
String strUUID = UUID.randomUUID().toString();
```

```
String UploadedFName = strUUID + originalFileName;
```

동명 파일 충돌 방지를 위해 UUID+원본 파일이름으로 조합

```
// filePath와 변경된 파일이름을 결합하여 empty 파일 객체를 생성
```

```
File serverFile = new File(filePath, UploadedFName);
```

특정 경로에 file object 생성

```
// upFile을 serverFile 이름으로 복사 수행
```

```
try {
```

```
uploadedFile.transferTo(serverFile);
```

원본파일의 내용물을 file object 에 복사

```
// water 이면 wtarePics 테이블에 저장
```

```
if (whichTable.equalsIgnoreCase("water")) {
```

```
    FishUserWaterPicsVO uwPicsVO = new FishUserWaterPicsVO();
```

```
    //long fk = uwService.getMaxID();
```

```
uwPicsVO.setUfp_fk(fk);
```

```
uwPicsVO.setUfp_originalFName(originalFileName);
```

```
uwPicsVO.setUfp_uploadedFName(UploadedFName);
```

```
int ret = uwPicsService.insert(uwPicsVO);
```

```
log.debug("!!! pic upload ret : " + ret);
```

```
}
```

```
else if (whichTable.equalsIgnoreCase("sea")) {
```

메게변수로부터 전달받은 테이블에  
UUID+업로드된 파일의 이름을 저장

# 섬머노트를 사용하기 위한 jsp 페 이지의 Script

```
$(document).ready(function() {
```

```
    $('summernote').summernote({  
        disableDragAndDrop : false,  
        callbacks:{  
            onImageUpload:function(files,editor,isEditted){  
                for(let i=files.length-1;i>=0;i--){  
                    //파일을 한개씩 업로드할 함수  
                    upFile(files[i],this) // this==editor(summernote 자체)  
                }  
            }  
        }  
    });  
});
```

섬머노트로 파일을 업로드  
할시 발생하는 callback 함  
수

```
function upFile(file,editor) {  
    var formData=new FormData()  
    //upFile 변수에 file정보를 담아서 보내기 위한 준비  
    /*  
    editor.insertImage:= summernote 의 내장 함수를 callback 형태로  
    호출해서 현재 summernote box에 표시하고자  
    이미지의 src부분을 url값으로 대치  
    img src="data:base64..." -> img src="UUIDfile.jpg" 형태로 변경  
    */  
    formData.append('upFile',file) // 'upFile'은 컨트롤러에서 받을 파라미터 이름  
    $.ajax({
```

```
        url:"${rootPath}/image_up",  
        type:"POST",  
        data:formData,  
        contentType:false,  
        processData:false,  
        enctype:"multipart/form-data",  
        beforeSend:function(ax){  
            ax.setRequestHeader("${_csrf.headerName}","${_csrf.token}")  
        },  
        success:function(result){  
            result="${rootPath}/files/"+result  
            $(editor).summernote('editor.insertImage',result)
```

ajax를 통하여 파일업로드

spring security에서 formdata 와 csrf를 같이

보낼수 없기에, beforSEND함수로 csrf를 헤더  
에 첨가.

# 회원가입

회원가입

test

....

....

회원가입

비밀번호를 잃어버렸다...

## Email 인증

회원가입을 완료하려면 email인증을 수행해야 합니다.

najongjin5@naver.com

인증 email 다시 보내기

E-mail을 열어서 인증코드를 확인한 후 아래 입력란에 입력 후 인증 버튼을 클릭하세요

인증코드 입력

인증하기

# 회원가입 controller

```
@SessionAttributes("memberVO")
@RequestMapping(value = "/member")
public class MemberController {
    private final MemberService memService;
```

```
@ModelAttribute("memberVO")
public MemberVO makeMemVO() {
    return new MemberVO();
}
```

```
@RequestMapping(value = "/register",method=RequestMethod.GET)
public String register(@ModelAttribute("memberVO") MemberVO memberVO,Model model) {
    memberVO=makeMemVO();
    model.addAttribute("memberVO", memberVO);
    return "member/register";
}
```

```
/**
 *@since 2020-04-21
 *최초회원 가입 화면에서 username과 password를 입력한 후 회원가입 버튼을 클릭하면
 *memberVO에 데이터를 받아서 sessionAttributes에 설정된 저장소에 저장해두고 이메일 인증 화면 보여주기
 */
```

```
@RequestMapping(value = "/register_next",method=RequestMethod.POST)
public String join_next(@ModelAttribute("memberVO") MemberVO memberVO,BindingResult result) {
    if(result.hasErrors()) {
        return "member/register";
    }
}
```

1 회원가입  
form

2 email 입력 페이지



# 회원가입 controller

```
@RequestMapping(value = "/register_last",method=RequestMethod.POST)
public String join_last(@ModelAttribute("memberVO") MemberVO memberVO,Model model) {
    //jsp로 내려보낸 인증코드(token)은 암호화 시킨거고, 메일로 보낸건 암호화 안시킨 인증코드(token)임.
    //인증코드(token)만들면서 이메일도 같이 보냄.
    String email_token=memService.insert_getToken(memberVO);//요건 암호화 된 인증코드(token)
    model.addAttribute("My_Email_Secret", email_token);
    model.addAttribute("JOIN", "EMAIL_OK");
    model.addAttribute("username", PbeEncryptor.getEncrypt(memberVO.getUsername()));
    return "member/register_email";
}
```

## 3 이메일 코드 발송 + 회

```
/**
 *@since 2020-04-21
 *이메일 인증폼에서 인증키와 인증값을 받아서 인증처리
 */
@ResponseBody
@RequestMapping(value = "/email_token_check",method=RequestMethod.POST)
public String email_token_check(@RequestParam("secret_id") String username,
    @RequestParam("secret_key") String secret_key,@RequestParam("secret_value") String secret_value) {
    boolean bKey=memService.email_token_ok(username,secret_key,secret_value);
    //jsp로 내려보낸 인증코드(token)은 암호화 시킨거고, 메일로 보낸건 암호화 안시킨 인증코드(token)임.
    if(bKey) return "OK";
    else return "FAIL";
}
```

## 원정보 DB에 저장

## 4 인증코드 검사 + 회원

## status 활성화

SessionAttribute에 회원정보VO를  
등록함으로,  
회원가입절차를 여러 페이지로 나누어 구현

# 회원가입 이메일인증 service

```
@Transactional
public String insert_getToken(MemberVO memberVO) {
    // DB에 저장
    memberVO.setEnabled(false);
    String encPassword = passwordEncoder.encode(memberVO.getPassword());
    memberVO.setPassword(encPassword);
    int ret = memDao.insert(memberVO);

    List<AuthorityVO> authList = new ArrayList<AuthorityVO>();
    authList.add(AuthorityVO.builder().username(memberVO.getUsername()).authority("USER").build());
    authList.add(AuthorityVO.builder().username(memberVO.getUsername()).authority("ROLE_USER").build());
    ret = authDao.insert(authList);

    String email_token = UUID.randomUUID().toString().split("-")[0].toUpperCase();
    String enc_email_token = PbeEncryptor.getEncrypt(email_token);
    // email보내기
    mailService.email_auth(memberVO, email_token);
    return enc_email_token;
}
```

1. 입력된 회원정보를 DB에 저장. 활성화권한은 없는 상태
2. 자식 테이블(권한 테이블)에 해당 유저의 권한 저장
3. 해당 유저가 입력한 이메일로 인증코드 발송

# 회원가입 이메일인증 jsp Script

```
$(document).on("click", "#btn_email_ok", function(){
    let secret_key=$("#span#secret").text()
    let secret_value=$("#input#email_ok").val()
    if(secret_value==""){
        alert("인증코드를 입력한후 인증버튼을 클릭하세요")
        $("#input#email_ok").focus()
        return false
    }
    $.ajax({
        url:"${rootPath}/member/email_token_check",
        method:"POST",
        data:{
            "${_csrf.parameterName}": "${_csrf.token}",
            secret_id:"${username}",
            secret_key:secret_key,
            secret_value:secret_value
        },
        success:function(result){
            alert(result)
            document.location.replace("${rootPath}/member/login")
        },
    })
})
```

(jsp 페이지에 해당 유저에게 발송한 인증코드를 같이 랜더링 한 상태)  
인증코드 확인전용 컨트롤러에 유저이름, 인증코드, 입력한 인증코드 를 보내서 인증코드 검사.  
검증이 되면 해당유저 활성화한 활성화시킴

# 채팅

나: gh(전체)

보내는사람

a

받는사람

전체

메시지

메시지 입력

전송

# 채팅 component

```
@Component
public class ChatSocketComponent extends TextWebSocketHandler{
    List<WebSocketSession> sessionList;
    Map<String,MessageVO> messageMap;
```

TextWebSocketHandler 라는 클래스를 상속 받음

```
@Override
public void afterConnectionEstablished(WebSocketSession session) throws Exception {
    // TODO Auto-generated method stub
    super.afterConnectionEstablished(session);
    /*
     * 최초 어떤 사용자가 접속하면 사용자에게 대한 메시지 정보를 담을 변수 초기화
     *
     * 세션 id를 key값으로 하는 비어있는 사용자 정보 map에 저장
     */
    sessionList.add(session);
    MessageVO mVO=new MessageVO();
    messageMap.put(session.getId(), mVO);
}
```

socket을 통해서 접속이 이루어 지면

각 유저의 socketSession을 저장

```
@Override
protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
    // TODO Auto-generated method stub

    //현재 접속자를 제외한 나머지 접속자에게 메시지 보내기
    private void sendNotMeMessage(WebSocketSession session,String textMessage) throws IOException {
        TextMessage sendMessage=new TextMessage(textMessage);
        for(WebSocketSession ws:sessionList) {
            //자신이 보낸 메시지는 자신에게 보낼 필요는 없다.
            if(!ws.getId().equals(session.getId())) {
                ws.sendMessage(sendMessage);
            }
        }
    }
}
```

채팅메시지가 전달되면  
socketSession에 등록된  
각 유저에게 채팅 메시지 전달

# 채팅 jsp Script

채팅을 구현하기 위해 sockjs 이용

```
<script src="https://cdn.jsdelivr.net/npm/sockjs-client@1/dist/sockjs.min.js"></script>
```

```
<script type="text/javascript">  
var socket = new SockJS('http://localhost:8080/tour/chat');  
//소켓 서버에 저소 시자
```

socket-context.xml 에 등록한 채팅 컴로넌트 경로로 연결

```
$(document).on("click", "#btn_send", function(){
```

```
    socket.send(JSON.stringify(message))  
})
```

메시지 전송 버튼 클릭시 입력된 message와 유저 정보를 string json으로 만들어서 socket.send() 를 통해 채팅 컴포넌트로 전송

# Jasypt 암호화를 위한 java main class

```
Scanner scanner = new Scanner(System.in);  
StandardPBEStrngEncryptor pbEnc = new StandardPBEStrngEncryptor();
```

jasypt 암호화를 사용하기 위한  
인스턴스

```
// 환경변수 가져오기  
Map<String, String> envList = System.getenv();  
String saltPass = envList.get("testsalt");  
System.out.println("SaltPass : " + saltPass);  
  
if (saltPass == null || saltPass.isEmpty()) {  
    System.out.println("salt Password Not Found");  
    return;  
}
```

```
// 암호화 설정  
pbEnc.setAlgorithm("PBEWithMD5AndDES");  
pbEnc.setPassword(saltPass);
```

암호 방식, salt password 설정

```
Map<String, String[]> secFileList = new TreeMap<String, String[]>();  
secFileList.put("mysql.properties", new String[] { "mysql.user", "mysql.password" });  
  
String proFileDir = "src/main/webapp/WEB-INF/spring/appServlet/props";
```

```
String plainString=scanner.nextLine();  
if(plainString.isEmpty()) {  
    System.out.println("Exit!!!");  
    out.close();  
    return;  
}
```

jasypt 로 암호화된 문자열을 저장하기 위한 파일

```
String encString = pbEnc.encrypt(plainString);  
  
//변수=ENC(값)  
encString=String.format("s=ENC(s)",key,encString);  
System.out.println(encString);  
out.println(encString);  
out.flush();
```

java의 파일에 write 기능을 통해 암호화된 문자열 저장

# Spring Security- implements

```
@Slf4j
public class AuthProviderImpl implements AuthenticationProvider {
    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        /*
         * authentication으로부터 로그인 폼에서 보낸 username과 password를 추출
         */
        String username = (String) authentication.getPrincipal();
        String password = (String) authentication.getCredentials();

        // service->dao를 통해서 DB로부터 사용자 정보 가져오기
        MemberVO memberVO = (MemberVO) userService.loadUserByUsername(username);
        if (!passwordEncoder.matches(password.trim(), memberVO.getPassword().trim())) {
            throw new BadCredentialsException("비밀번호 오류");
        }
        if (!memberVO.isEnabled()) {
            throw new BadCredentialsException(username + ": 접근권한 없음");
        }
    }
}
```

1. Spring Security를 사용하기위해 AuthenticationPriver를 상속받은 서비스를 만듬.
2. Authenticauiou 메서드를 오버라이딩
3. 입력폼으로부터 전송되온 username과 password를 authentication 객체로부터 뽑음
4. loadUserByusername() 통해서 DB에서 조회



# Spring Security- implements

```
@Service("userDetailsService")
```

```
@Slf4j
```

```
public class UserDetailsServiceImpl implements UserDetailsService{
```

UserDetailsService를 상속받은  
클래스를 만듦

```
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
```

```
//spring security가 사용할 detailVO 선언
```

```
MemberVO memberVO=memberDao.findByUName(username);
```

Dao를 통해서 DB로부터 전달되어온 user가

```
if(memberVO==null) {
```

```
    throw new UsernameNotFoundException("User Name이 없습니다");
```

```
}
```

```
memberVO.setAccountNonExpired(true);
```

있나 검사

```
memberVO.setAccountNonLocked(true);
```

```
memberVO.setCredentialsNonExpired(true);
```

```
//userDetails 안의 authority property는 이상한 객체로 되었어서
```

```
//getAuthorities()라는 보일러 코드로 한번 전처리 해주고 값 셋팅
```

```
memberVO.setAuthorities(this.getAuthorities(username));
```

```
return memberVO;
```

```
}
```

```
private Collection<GrantedAuthority> getAuthorities(String username){
```

authDao 를 통해서 해당 유저  
의 권한들 가져옴

```
List<AuthorityVO> authList=authDao.findByUserName(username);
```

```
List<GrantedAuthority> authorities=new ArrayList<GrantedAuthority>();
```

```
for(AuthorityVO vo:authList) {// db에 저장된 권한 목록들을
```

```
    if(!vo.getAuthority().trim().isEmpty()) {
```

```
        //spring security용 auth list로 복사
```

```
SimpleGrantedAuthority sAuth=new SimpleGrantedAuthority(vo.getAuthority());
```

```
authorities.add(sAuth);
```

```
    }
```

```
}
```

```
return authorities; spring security용 권한 class type에 맞게 변환
```

# Spring Security VO

UserDetails를 상속받은 VO 만듦

```
public class MemberVO implements UserDetails{  
    /**  
     *  
     */  
    private static final long serialVersionUID = 1L;  
  
    private long id;  
  
    @NotBlank(message = "* 아이디를 입력해 주세요")  
    private String username;  
  
    @NotBlank(message = "* 비밀번호를 입력해 주세요")  
    private String password;  
  
    private boolean enabled;  
  
    private boolean accountNonExpired;  
    private boolean accountNonLocked;  
    private boolean credentialsNonExpired;  
  
    private Collection<? extends GrantedAuthority> authorities;  
}
```

필수로 있어야 하는 프로퍼티들을  
@NotBlank annotation을 정의

@NotBlank annotation은  
spring validation을 위한  
annotation

# Spring Security- xml

```
<context:component-scan  
    base-package="com.biz.tour.service.member.auth" />  
<bean id="authProvider"  
    class="com.biz.tour.service.member.auth.AuthProviderImpl" />  
  
<security:authentication-manager>  
    <security:authentication-provider  
        ref="authProvider" />  
</security:authentication-manager>
```

1. spring security와 관련된 서비스들은 servlet-context를 읽기전에 먼저 요구됨.
2. root-context와 같은 레벨에 spring security xml을 작성해줌.
3. security를 스캔할 패키지를 따로 만들어주고 security와 관련된 서비스들을 하위 패키지에 작성
4. security-xml에 security용 service가 있는 패키지를 스캔하도록 지정해줌
5. authProvider 를 bean으로 등록해줌
6. authentication-manager의 reference 부분에 bean으로 등록한 서비스를 지정해줌

# 챗봇(인공지능)

TourAPIFish  
공유하기 ~

새로고침

✕

고객센터에 문의하기

오후 2:04

hi

오후 2:04

TourAPIFish

Hi, Welcome to Korea Fishing site. This site introduces fishing locations in South Korea.

오후 2:05

what is this site about

오후 2:05

TourAPIFish

This site about introducing decent fishing spots for users.

if you press one of menu in th picture, it will direct you to fishing locations

If you want a sample video, type

1. 'i want to know what ind of fishes live in certain region'

2. '(the name of location)' Example: 'mokpo'

3. copy and paste the URL to web browser

if you want to do chat or upload your own lucky fishing spot, please login to this site, then

Water Fishing

Sea Fishing

1. click a menu above

insert new water fishing spot

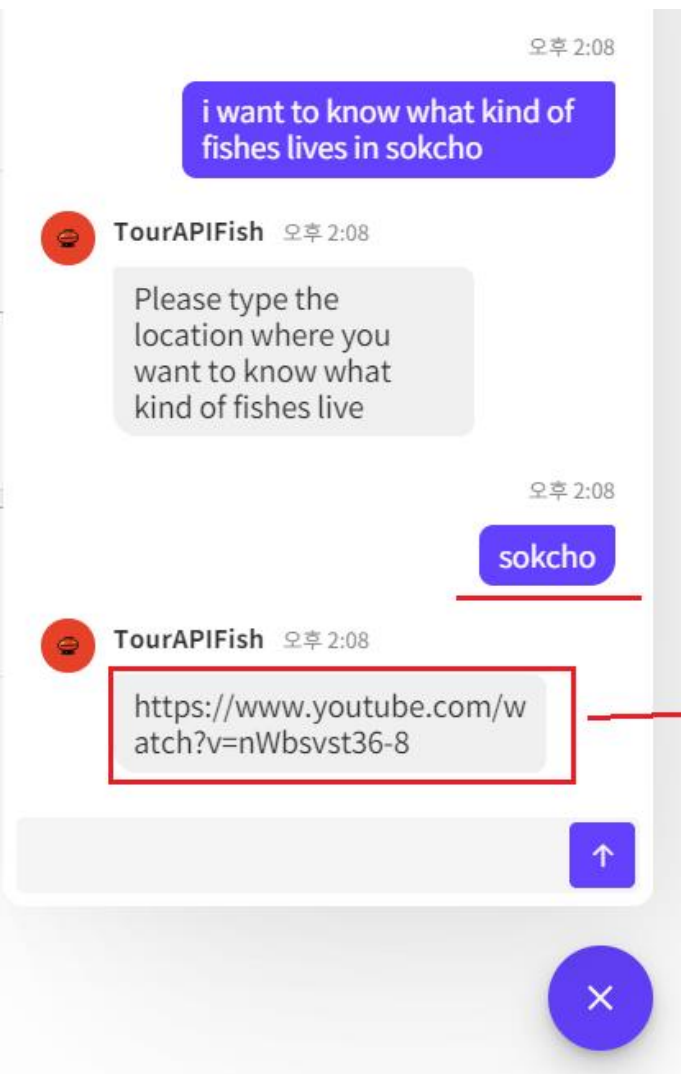
2. click this button at the bottom

↑





# 챗봇(인공지능)



챗봇 제작 사이트:  
<https://builder.latte.ai/>

# 챗봇 제작 + youtube API 화면

사용자 입력 봇 응답

API

필터 +

GET

https://www.googleapis.com/youtube/v3/search?part=snippet&key=AlzaSyBUSZiS0Drtaz\_oNSiFy7iraJyQMqQ&q=what%20kind%20of%20fishes%20live%20in%20{{2: regionForFish }}

테스트

Request

Headers +

Key	Value
<input type="checkbox"/>	<input type="text"/>

Response

값을 저장할 속성 값 ?

[[3: regionForFi...]] = items.0.id.vide -

+

테스트 결과

대화 해보기

챗봇 제작 사이트 내의 API 호출기능을 이용

기본적인 npl 기능과 api 호출 연동이 가능

전문 제작 사이트를 사용하기때문에  
시나리오 관리가 편하고 유용

# NLP 기술이 탑재된 사이트를 이용

1. “저는 문장입니다. 여러가지 단어와 문단들이 합해져 있죠  
ㅎㅎ. 정규문법일수도 있고 오타가 만을후도 있어요. ㅋ.ㅋ 은  
어일수도 있고요” 문장 읽어들임
2. 각 문장을 여러가지 수학기법과 컴파일러 언어기법과 인공지능 기법들을 혼합하여 분석. 단어분석-> 단어 전처리-> 문장분석-> 문장 전처리 등등의 복잡한 과정이 필요
3. 분석한 문장을 수학수식으로 표현.  
수학수식은 결론적으로 값을 가짐  
예: “문장1은 결론적으로 값을 가지게 될거예요” == 11

# NLP 기술이 탑재된 사이트를 이용

4. 예시문장 == 11 에 답을 맵핑해줌

예: 예시문장==11 -> "저는 전 페이지의 예시문장의 답변"

5. 사용자 문장을 입력으로 받아들임.

"저는 사용자가 입력한 문장입니다... 뭐뭐뭐가 적혀 있겠죠"  
를 수식으로 표현. == 13이라고 가정

6. 사용자 입력문장== 13은 11에 수렴한다고 가정

7. 챗봇은 "저는 전 페이지의 예시문장의 답변" 을 출력