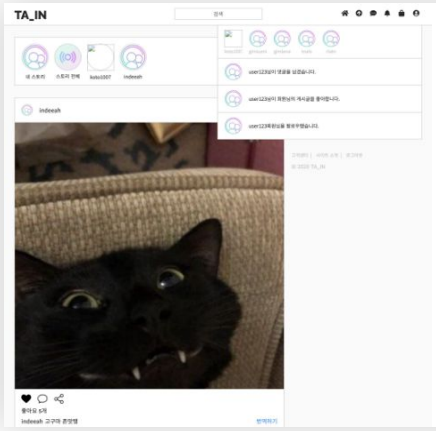
# PORTFOLIO - TA_IN




## TA_IN

타인과 일상을 소통하다.

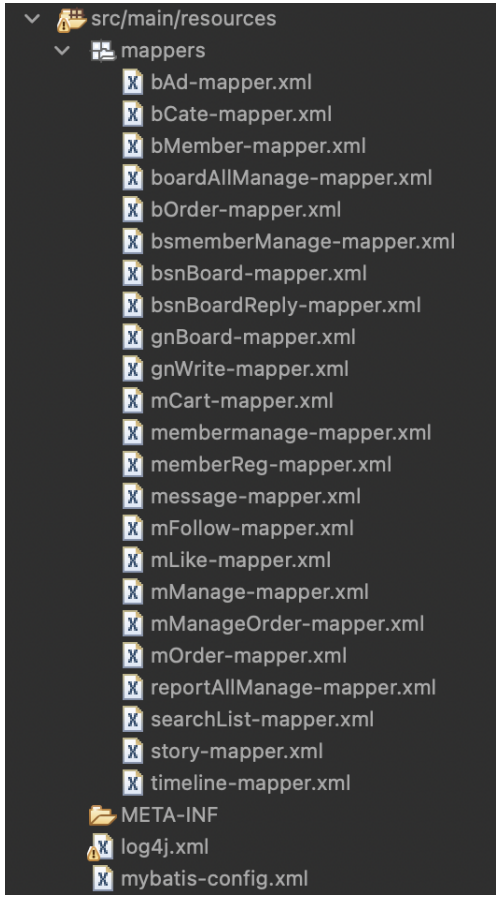
코로나19 팬데믹은 우리의 삶을 전적으로 변화시켰습니다. 이에 우리는 개인과 개인 간의 소통뿐만 아니라 개인과 기업 간의 연결과 소통을 보다 쉽게 열 수 있도록 “타인” 프로젝트를 기획하게 되었습니다.

Git Link is here

go!

문서 드라이브 링크 바로가기

# PORTFOLIO - TA_IN




*XML 설정*

```
<!-- DBCP 연결 -->
<beans:bean id="dataSource"
    class="org.apache.commons.dbcp.BasicDataSource">
    <beans:property name="driverClassName"
        value="oracle.jdbc.driver.OracleDriver" />
    <beans:property name="url"
        value="jdbc:oracle:thin:@localhost:1521:xe" />
    <beans:property name="username" value="TA_IN" />
    <beans:property name="password" value="1234" />
</beans:bean>

<beans:bean id="sqlSessionFactory"
    class="org.mybatis.spring.SqlSessionFactoryBean">
    <beans:property name="dataSource" ref="dataSource" />
    <beans:property name="configLocation"
        value="classpath:mybatis-config.xml" />
</beans:bean>

<beans:bean id="sqlSession"
    class="org.mybatis.spring.SqlSessionTemplate">
    <beans:constructor-arg ref="sqlSessionFactory" />
</beans:bean>
```

*DB 연동*
*sqlSession*

# 검색 기능


일반&비즈니스 해시태그
계정 아이디를 DB에서 꺼내온다.

```
<select id="autocomplete" parameterType="string"
                resultType="arrayList" resultMap="resultTimeLine">
        <!-- 해시태그는 '#'를 붙여서 출력, 계정 아이디는 '@'를 붙여서 출력 -->
                select '#'||h_tag word from hashtag
        <!-- 대문자, 소문자 구분 없이 모두 검색한다. -->
                where h_tag like concat('%',concat(UPPER(#{word}),'%')) or
                h_tag like concat('%',concat(LOWER(#{word}),'%'))
                union
                select '#'||h_tag word from businesshashtag
                where h_tag like concat('%',concat(UPPER(#{word}),'%')) or
                h_tag like concat('%',concat(LOWER(#{word}),'%'))
                union
                select '@'||m_id word from member
        where m_id like concat('%',concat(UPPER(#{word}),'%')) or
                m_id like concat('%',concat(LOWER(#{word}),'%'))
        </select>
```

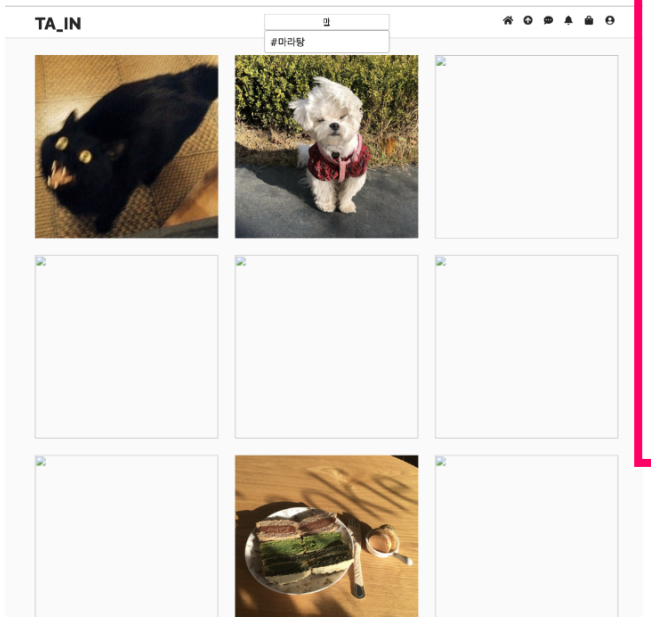
## 자동 검색어 완성 기능

```
    // 자동검색어완성
$(function() {
    $("#search").autocomplete({
        source: function(request, response) {
            var request = $("#search").val();
            console.log(request);
            $.ajax({
                url: "${pageContext.request.contextPath}/autocomplete.do",
                type: "post",
                dataType: "json",
                data: {
                    word: request
                },
                success: function(data) {
                    var value = new Array();
                    for (var i = 0; i < data.length; i++) {
                        // 배열형태로 넣어준다.
                        value.push(data[i].word)
                    }
                    response(
                        $.map(value, function(item) {
                            return {
                                value: item,
                                test: item + "test"
                            }
                        })
                    )
                },
                error: function(request, status, error) {
                    alert("code:" +
                        request.status +
                        "\n" +
                        "message:" +
                        request.responseText +
                        "\n" + "error:" +
                        error);
                }
            });
        }
    });
```

# 검색 기능

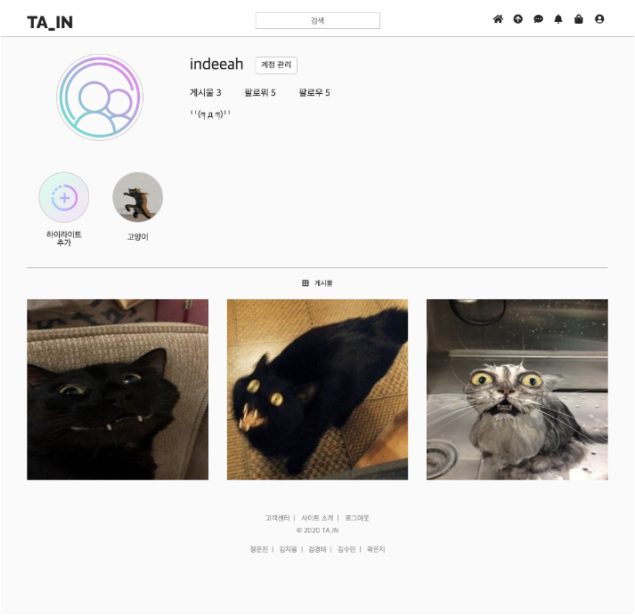
## 검색어에 따른 페이지 이동 기능

검색 완료 페이지로 이동




```
// 해시태그, 계정 검색 분류
$("#search").on("keypress", function(event) {
    // 검색 버튼이 아닌 Enter 눌렀을 때 이벤트 발생
    if (window.event.keyCode == 13) {
        var at = "@";
        var hashtag = $("#search").val();
        // 사용자가 입력한 검색어가 '@'로 시작할 때
        if (hashtag.startsWith(at)) {
            // '@'를 없애준다.
            hashtag = hashtag.replace(/[@]/g, '');
            // 개인 계정 프로필로 이동
            var url = "${pageContext.request.contextPath}/gnMain?m_id=" + hashtag;
            $(location).attr('href', url);

            // '@'로 시작하는 것이 아닌 검색어
        } else {
            // '#'이 있다면 없애준다.
            hashtag = hashtag.replace(/[#]/g, '');
            // 검색 완료 페이지로 이동
            var url = "${pageContext.request.contextPath}/explore?hashtag=" + hashtag;
            $(location).attr('href', url);

        }
    }
});
```

계정 프로필 페이지로 이동



# 게시물 스크롤 페이징

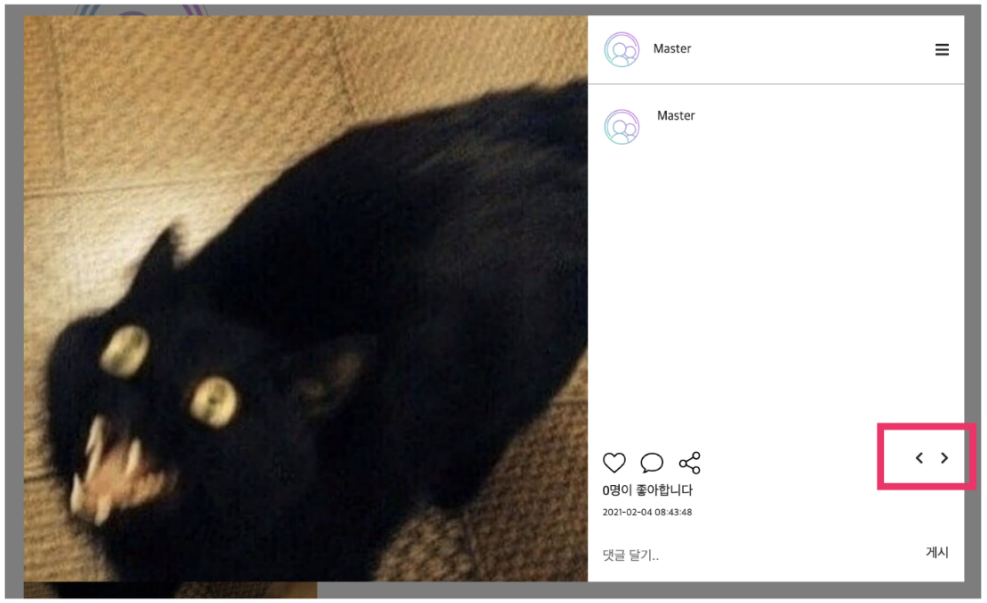
```javascript
// 스크롤 페이징
// page선언
var page = 1;

$(function() {
    // 리스트 추가 조회
    getList(page);
    // 첫 화면 로드시 page 증가.
    page++;
});

//스크롤이 최하단으로 내려가면 리스트 조회 후 page 증가.
$(window).scroll(function() {
    if ($(window).scrollTop() >= $(document).height() - $(window).height()) {
        getList(page); // 리스트 추가 조회
        page++;
    }
});

// 리스트 추가 조회
function getList(page) {
    $.ajax({
        url: 'timeLineScroll.do',
        method: 'post',
        data: {
            "page": page
        },
        dataType: 'json',
        success: function(resp) {
            var data = resp.list.length;
            var htmls = "";
            if (page == 1) { //페이지 1일 경우 id가 page인 html을 비운다.
                $("#page").html("");
            }
            console.log(resp.currentPage + ":" + resp.maxPage);
            if (resp.currentPage <= resp.maxPage) {
                if (data > 0) {
                    for (var i = 0; i < data; i++) {
                        htmls += // 나타낼 코드
                    }
                }
```

```java
// 게시물 스크롤 페이징
@ResponseBody
@RequestMapping(value="/timeLineScroll.do", method = RequestMethod.POST)
public HashMap<String, Object> timeLineScroll(Model model, HttpServletRequest request,
                @RequestParam(name="page", defaultValue = "1") int page) {
        HashMap<String, Object> result = new HashMap <String,Object>();
        HttpSession session = request.getSession();
        String my_name = (String) session.getAttribute("my_name");
        int currentPage=page;
        int listCount=tService.timeLineListCount(my_name);
        int maxPage=(int)((double)listCount/LIMIT+0.9);
        List<TimeLine> logList = tService.showTimeLineListPage(my_name, currentPage, LIMIT);
        result.put("count", tService.timeLineListCount(my_name));       // 게시물카운트
        result.put("currentPage", currentPage);                         // 현재 페이지
        result.put("maxPage", maxPage);                                         // 최대 페이지
        result.put("list", logList);    // 게시물 텍스트정보
        System.out.println("list:"+ logList);
        return result;
}
```

## 무한 스크롤 기능

*타임라인, 검색 완료 페이지, 일반 계정 프로필 페이지에 사용*

# 이미지 슬라이드 기능




## 이미지 슬라이드 기능

*타임라인, 일반 계정 프로필 페이지, 게시물 페이지, 스토리에 사용*
*이미지의 넓이와 개수로 계산하여 버튼을 누를 때마다*
*x좌표를 바꿔주는 기능을 추가하였습니다.*

```css
.timeline_photo {
        width: 660px;
        height: 660px;
        position: relative;
        overflow: hidden;
}

.timephoto {
        width: 660px;
        height: 660px;
        overflow: hidden;
        display: flex;
        position: absolute;
        justify-content: center;
        top: 0px;
        left: 0px;
}

.show_t_img_con {
        width: 660px;
        height: 660px;
        overflow: hidden;
}

.show_t_img {
        width: 660px;
        z-index: 2;
}
```

```javascript
var slideWrapper = document.querySelector('.timeline_photo' + t_id);
var slides = document.querySelectorAll('.show_t_img_con' + t_id);
var totalSlides = slides.length;
var sliderWidth = slideWrapper.clientWidth;
var slideIndex = 0;
var slider = document.querySelector('.photo' + t_id);
slider.style.width = sliderWidth * totalSlides + 'px';

function showSlides(n) {
    slideIndex = n;
    if (slideIndex == -1) {
        slideIndex = totalSlides - 1;
    } else if (slideIndex === totalSlides) {
        slideIndex = 0;
    }
    slider.style.left = -(sliderWidth * slideIndex) + 'px';
}

function plusSlides(n) {
    showSlides(slideIndex += n);
}

function currentSlide(n) {
    showSlides(slideIndex = n);
}
var nextBtn = document.querySelector('.slide_btn_next' + t_id);
var prevBtn = document.querySelector('.slide_btn_prev' + t_id);
nextBtn.addEventListener('click', function() {
    plusSlides(1);
});
prevBtn.addEventListener('click', function() {
    plusSlides(-1);
});
```

# 게시물 업로드


```
<!-- 텍스트 업로드 -->
<insert id="insertboard" parameterType="GnWrite"
        flushCache="true" statementType="PREPARED" timeout="20">
        insert into board
        (B_ID, M_ID, B_CONTENT) values
        ('BO'||TO_CHAR(SYSDATE,'RRMMDD')|| LPAD(#{seq},3,'0'),
        #{my_name},
        #{b_content})
</insert>

<!-- 다중 파일 업로드 -->
<insert id="insertboardimg" parameterType="GnWrite"
    flushCache="true" statementType="PREPARED" timeout="20">
    <!-- 10개까지 이미지 업로드 가능 -->
    insert into
    boardadd values(
    'BO'||TO_CHAR(SYSDATE,'RRMMDD')|| LPAD(#{seq},3,'0'),
    #{b_img1}, #{b_img2}, #{b_img3}, #{b_img4},
    #{b_img5}, #{b_img6}, #{b_img7}, #{b_img8}, #{b_img9}, #{b_img10})
</insert>
```

## 다중 파일 업로드 기능

*똑같은 b_id로 inert 해주기 위해 load 컨트롤러에서 nextval을 통해 번호를 받아온 뒤 해당 번호를 insert 하였습니다.*

```
@RequestMapping(value = "/insertboard.do", method = RequestMethod.POST)
public ModelAndView insertboard(GnWrite gw, @RequestParam(name = "upfile") MultipartFile report,
        MultipartHttpServletRequest request, ModelAndView mv, @RequestParam(name = "h_tag") String h_tag,
        @RequestParam(name = "seq") int seq) {

// 배열 생성
    String[] bImgList = { gw.getB_img1(), gw.getB_img2(), gw.getB_img3(), gw.getB_img4(), gw.getB_img5(),
            gw.getB_img6(), gw.getB_img7(), gw.getB_img8(), gw.getB_img9(), gw.getB_img10() };
    try {
        if (request != null && !request.equals("")) {
            uploadFiles(request);
        }
        System.out.println("게시물 등록 파일저장성공");
        List<MultipartFile> fileList = request.getFiles("upfile");
        for (int i = 0; i < fileList.size(); i++) {
            bImgList[i] = fileList.get(i).getOriginalFilename();
        }

        // set 되지 않은 컬럼에는 null 표시
        gw.setB_img1(bImgList[0]);
        gw.setB_img2(bImgList[1]);
        gw.setB_img3(bImgList[2]);
        gw.setB_img4(bImgList[3]);
        gw.setB_img5(bImgList[4]);
        gw.setB_img6(bImgList[5]);
        gw.setB_img7(bImgList[6]);
        gw.setB_img8(bImgList[7]);
        gw.setB_img9(bImgList[8]);
        gw.setB_img10(bImgList[9]);
        gwService.insertboard(gw);
        gwService.insertboardimg(gw);
```

*배열을 생성해주고 set 해주는 방식을 사용했습니다.*
*board 테이블의 b_id가 boardadd의*
*primary key 이므로 insertboard를 먼저 호출하고,*
*다음으로 insertboardimg를 호출하였습니다.*

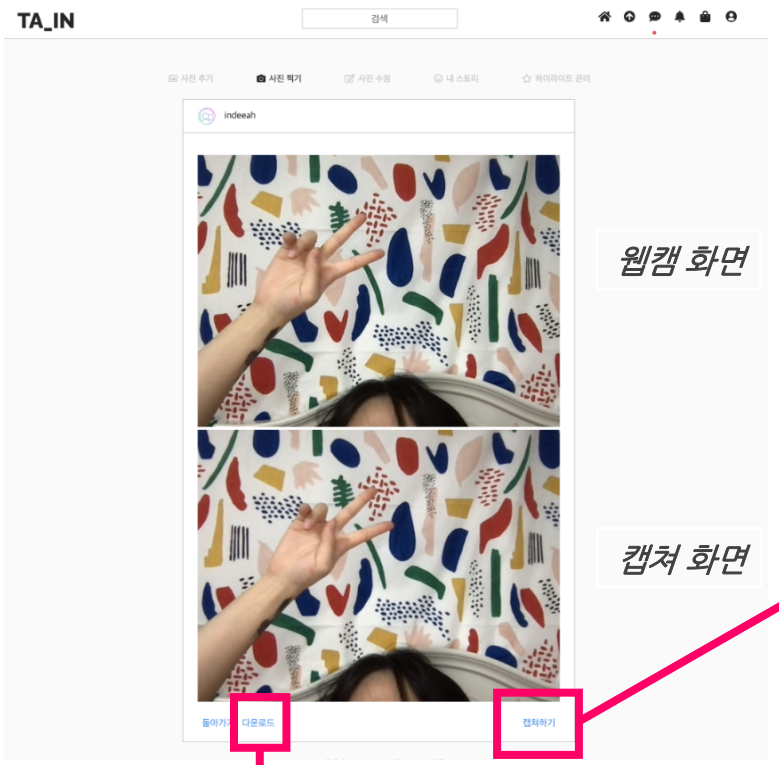
# 게시물 업로드

## 해시태그 업로드 기능


```
// hashtag insert
int h_taglen = h_tag.length();
if (h_taglen != 0) {

    // 해시태그는 '#해시태그1 #해시태그2 #해시태그3' 으로 입력되기 때문에
    // split을 위해서 맨 앞의 '#'을 잘라주기 위해 substring(1)을 사용했다.
    // 여기서 substring을 사용했을 때 null 값이 들어오면 에러가 발생하기 때문에
    // h_tag의 length를 먼저 확인해주고 0이 아닐 때의 조건을 주었다.
    h_tag = h_tag.substring(1);
    // " #"을 split 해주어 array에 넣어준다.
    String[] array = h_tag.split(" #");
    for (int i = 0; i < array.length; i++) {
        gw.setH_tag(array[i]);
        gwService.insertboardhashtag(gw);
    }
}
```

# 게시물 업로드

## 웹캠 연동 및 캡쳐 기능

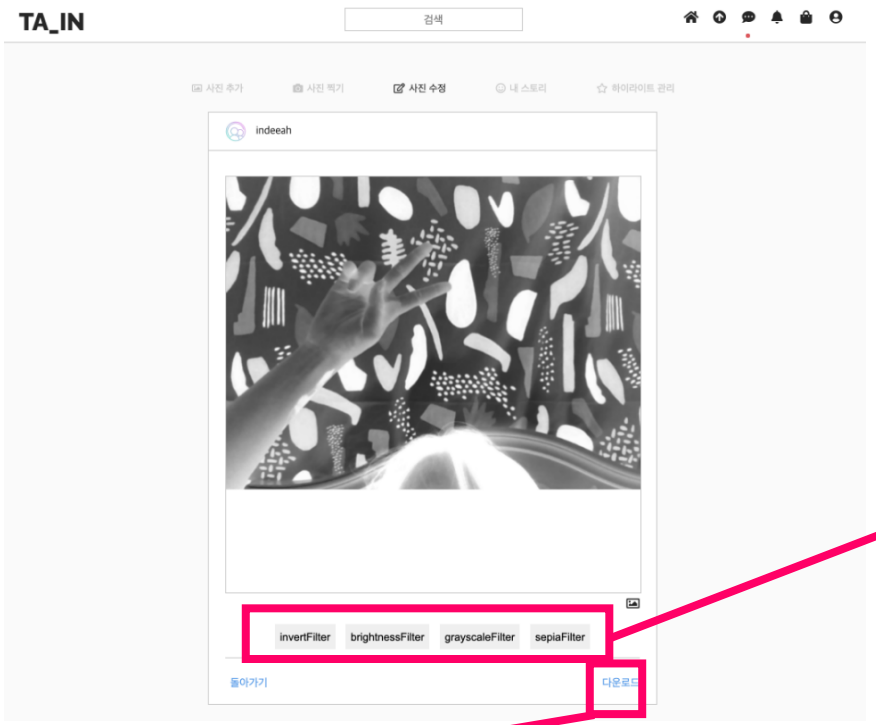



웹캠 화면

캡쳐 화면

```
// 웹캠 연결
if (navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
    navigator.mediaDevices.getUserMedia({
        video: true
    }).then(function(stream) {
        // webcam 연결 승인하면 id가 'video'인 곳에 화면이 보여진다.
        var video = document.getElementById('video');
        video.srcObject = stream;
        video.play();
    });
}

// 웹캠 화면 캡쳐
var canvas = document.getElementById('canvas');
var context = canvas.getContext('2d');
var video = document.getElementById('video');
document.getElementById("webcamBtn").addEventListener("click", function() {
    $("#canvas").css("display", "block");
    // 'webcamBtn'을 누르면 가로 500px 세로 375px 사이즈로 캡쳐된다.
    context.drawImage(video, 0, 0, 500, 375);
});

// download="capture.png"로 정해놓은 a태그를 누르면 다운로드 된다.
document.querySelector('a').addEventListener('click', event =>
    event.target.href = canvas.toDataURL()
);
```

# 게시물 업로드




버튼 누르면 해당 필터 호출

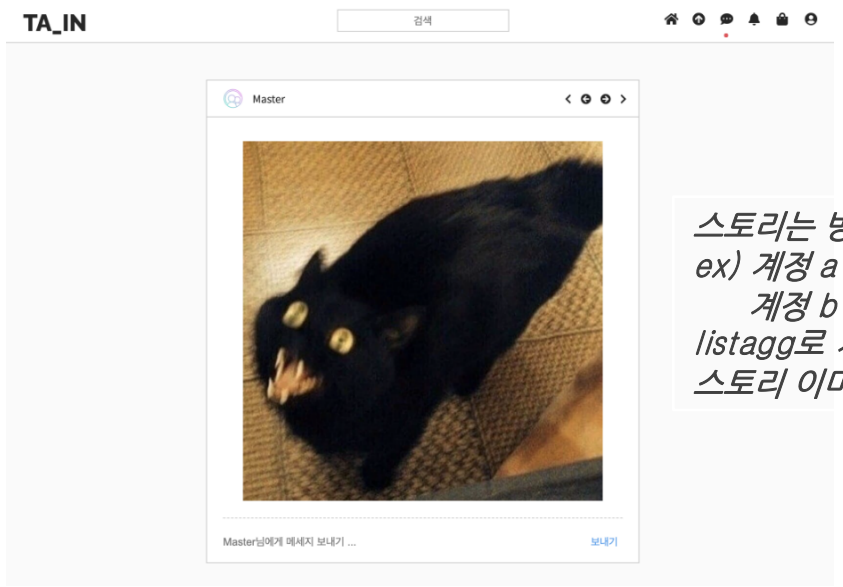
이미지 필터 적용 후 쉽게 다운로드가 가능하게 하기 위해 img 태그가 아닌 canvas 태그를 사용하였으며, 다운로드 클릭 시 from-canvas.png의 이름으로 저장되게 하였습니다.

```
// 수정본 다운로드
$("#canvas_down").on('click', function() {
    var dataURL = canvas1.toDataURL('image/png');
    dataURL = dataURL.replace(/^data:image\/[^;]*/, 'data:application/octet-stream');
    dataURL = dataURL.replace(/^data:application\/octet-stream/, 'data:application/octet-stream;headers=Content-Dispos
    // a태그를 만들고
    var aTag = document.createElement('a');
    // 클릭하면 'from_canvas.png'의 이름으로 저장됨
    aTag.download = 'from_canvas.png';
    aTag.href = dataURL;
    aTag.click();
});
```

## 이미지 필터 적용

```
// 필터
function invertFilter(pixels) {
    var d = pixels.data;
    for (var i = 0; i < pixels.data.length; i += 4) {
        d[i] = 255 - d[i]; // R
        d[i + 1] = 255 - d[i + 1]; // G
        d[i + 2] = 255 - d[i + 2]; // B
        d[i + 3] = 255; // Alpha
    }
    return pixels;
}

function brightnessFilter(pixels, value) {
    var d = pixels.data;
    for (var i = 0; i < d.length; i += 4) {
        d[i] += value / 3;
        d[i + 1] += value / 3;
        d[i + 2] += value / 3;
    }
    return pixels;
}

function grayscaleFilter(pixels) {
    var d = pixels.data;
    for (var i = 0; i < d.length; i += 4) {
        var r = d[i];
        var g = d[i + 1];
        var b = d[i + 2];

        var v = 0.2126 * r + 0.7152 * g + 0.0722 * b;
        d[i] = d[i + 1] = d[i + 2] = v
    }
    return pixels;
}

function sepiaFilter(pixels) {
    var d = pixels.data;
    for (var i = 0; i < d.length; i += 4) {
        var r = d[i];
        var g = d[i + 1];
        var b = d[i + 2];

        d[i] = r * 0.3588 + g * 0.7044 + b * 0.1368;
        d[i + 1] = r * 0.2990 + g * 0.5870 + b * 0.1140;
        d[i + 2] = r * 0.2392 + g * 0.4696 + b * 0.0912;
    }
    return pixels;
}
```

# 전체 스토리 조회




*스토리는 병렬식 조회이므로*
*ex) 계정 a : 스토리1, 스토리2, 스토리3*
*계정 b : 스토리1, 스토리2*
*listagg로 계정 아이디를 조회한 뒤*
*스토리 이미지는 ajax로 조회하였습니다.*

```
function split(a, a_t, b, b_t) {
    // a : 일반스토리의 m_id
    // a_t : 일반스토리의 type
    // b : 광고스토리의 m_id
    // b_t : 광고스토리의 type
    var arr = a.split("|");
    var arrb = b.split("|");
    for (i = 0; i < arr.length; i += 3) {
        for (j = i; j < i + 3; j++) {
            if (arr[j] != undefined) {
                $("#story").append(
                    '<div class="story_photo_con" onclick="showeach(\'' + arr[j] + '\',\'' + a_t + '\');">' +
                    '<div class="scon_con scon_con' + arr[j] + '"><div class="scon scon' + arr[j] + a_t + '"></div></d
            }
        }
        if (arrb[i] != "") {
            $("#story").append(
                '<div class="story_photo_con" onclick="showeach(\'' + arrb[i] + '\',\'' + b_t + '\');">' +
                '<div class="scon_con scon_con' + arrb[i] + '"><div class="scon scon' + arrb[i] + b_t + '"></div></div
        }
    }
```

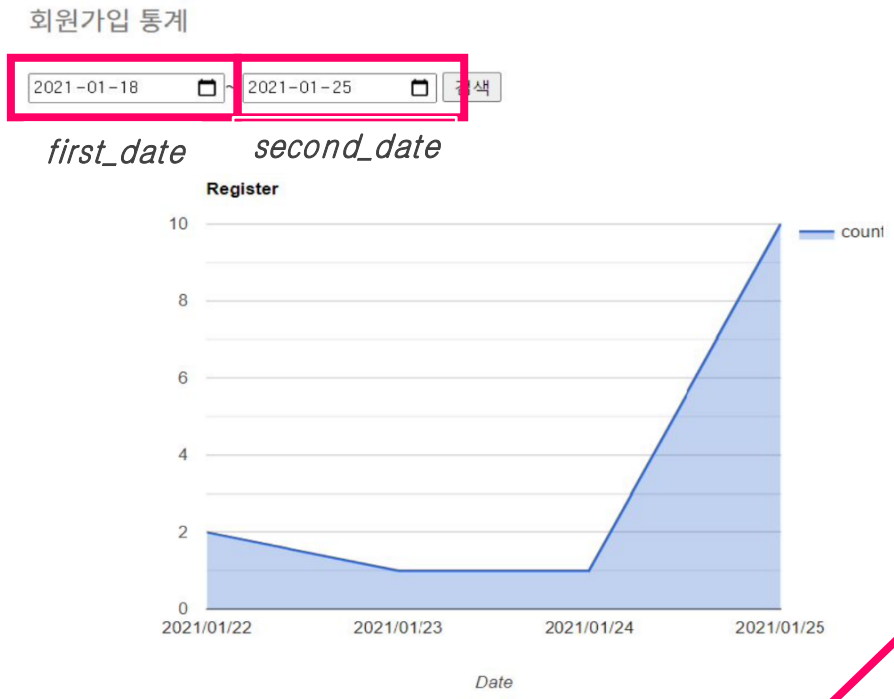
*일반 스토리 3개 --> 광고 스토리 1개 --> 일반 스토리 3개 순*
*3 --> 1 --> 3 ---> 1 순서 만들기*

```
<select id="showAllStory" parameterType="string"
                resultType="Story">
<!-- 일반 스토리 조회 -->
    select listagg(m_id,'|') within group (order by s_date desc) as m_id, s_type
    from (select max(s_date) as s_date, m_id, s_type
    from story join follow
    on m_id = m_id2
    where m_id1=#{m_id}
    and s_type='G'
    and s_date > SYSDATE - 1
    group by m_id, s_type
    union
    select max(s_date) as s_date, m_id, s_type
    from story
    where m_id=#{m_id}
    and s_type='G'
    and s_date > SYSDATE - 1
    group by m_id, s_type
    )
    group by s_type
</select>
<select id="showAllAStory" resultType="Story">
<!-- 광고 스토리 조회 -->
    select listagg(m_id,'|')
    within group (order by s_date desc) as m_id, s_type
    from (select
    max(s_date) as s_date, m_id, s_type
    from story join follow on m_id =
    m_id2
    and s_type='A'
    group by m_id, s_type)
    group by s_type
</select>
```

*1. 내가 팔로우하는 계정들 중 스토리를 올린 계정 조회*
*2. 스토리를 여러개 업로드 했을 시 가장 최근 날짜*

*3. 내 스토리 조회*

# 회원가입 통계 기능




배열을 생성해준 뒤 날짜와 회원가입 수를 만들어준 컬럼의 형태로 넣어주었습니다.

```
<select id="forRegChart" parameterType="TimeLine"
    resultType="arrayList" resultMap="resultTimeLine">
    select count(m_id) count, to_char(m_joindate, 'yyyy/mm/dd') m_joindate
    from member where m_joindate between #{first_date} and #{second_date}
    group by to_char(m_joindate, 'yyyy/mm/dd') order by m_joindate asc
</select>
```

```
$.ajax({
    url : "${pageContext.request.contextPath}/forchartTest.do",
    method : "POST",
    // 시작 날과 끝 날을 지정해 보내준다.
    data : {
        first_date : rfdate,
        second_date : rsdate
    },
    success : function(chart, chartl) {
        var value = new Array();
        for (var i = 0; i < chart.length; i++) {
            // 회원가입 날짜와 회원가입 수를 배열로 넣어준다.
            value.push([ chart[i].m_joindate, chart[i].count ]);
        }
        google.charts.load('current', {
            'packages' : [ 'corechart' ]
        });
        google.charts.setOnLoadCallback(drawChart);

        // 차트를 그려준다.
        function drawChart() {
            var data = new google.visualization.DataTable();
            // 날짜와 회원가입 수 컬럼 추가
            data.addColumn('string', 'date1');
            data.addColumn('number', 'count');
            // 만들어준 배열을 넣어준다.
            data.addRows(value);

            // 차트 style에 관한 설정
            var options = {
                title : 'Register',
                hAxis : {
                    title : 'Date',
                    titleTextStyle : {
                        color : '#333'
                    }
                },
                vAxis : {
                    minValue : 0
                }
            };

            // 'chart_div'에 차트를 그려준다.
            var chart = new google.visualization.AreaChart(
                    document.getElementById('chart_div'));
            chart.draw(data, options);
```