9주차 진행 보고서 세부 사항

2021.04.29~ 2021.05.12 B71116장혜리

코드는 아래 gitnub에서 확인 가능합니다.

[**[github]MBTI-Lovers 안드로이드**](https://github.com/hyerijang/MBTI-Lovers)

[**[github]MBTI-Lovers-webServer**](https://github.com/hyerijang/MBTI-Lovers-webServer)

# 매칭 구현

## 맵 로딩 시 현재 위치를 맵 중심으로 지정

Kakao 지도 Web API에서 제공하는 샘플 “geolocation으로 마커 표시하기”를 참고하여 제작하였습니다. Chrome 브라우저는 https 환경에서만 geolocation을 지원하기 때문에 도메인과 SSL을 발급하여 https 웹사이트로 변경하였습니다. 자세한 사항은 뒷부분에 기술하겠습니다.

function setPosition*() {* // HTML5의 geolocation으로 사용할 수 있는지 확인합니다  
 if *(****navigator***.geolocation*) {* // GeoLocation을 이용해서 접속 위치를 얻어옵니다  
 ***navigator***.geolocation.getCurrentPosition*(*function *(*position*) {* var lat = position.coords.latitude, // 위도  
 lon = position.coords.longitude; // 경도  
  
 var locPosition = new kakao.maps.LatLng*(*lat, lon*)*; //접속 위치  
  
 ***map***.setCenter*(*locPosition*)*;  
  
 // 지도 중심좌표를 얻어옵니다  
 var latlng = ***map***.getCenter*()*;  
  
 ***userPosition***.x = latlng.getLat*()*;  
 ***userPosition***.y = latlng.getLng*()*;  
  
 ***console***.log*(*"현재 내 위치 :" + ***userPosition***.x + "," + ***userPosition***.y*)*;

// 위치 좌표 DB에 저장

// "/members/position" 에 post  
 ***document***.getElementById*(*"positionX"*)*.setAttribute*(*"value", lat*)*;  
 ***document***.getElementById*(*"positionY"*)*.setAttribute*(*"value", lon*)*;  
 ***document***.userPosition.submit*()*;  
 *})*;  
 *}  
  
}*

## 유저 위치 정보 DB에 저장

javascript 단에서 setPosition()함수가 호출되면 "/members/position"에 현재 위치를 post합니다. MapController는 이를 받아 유저 위치 정보를DB에 저장합니다. 프론트 단에서 보내주는 위치 좌표는 double 형태이기 때문에 DB에 저장하기 전 이를 Point형태로 변환해주어야 합니다.

@Override  
public void setPoint*(*Optional*<*Member*>* optMember, double latitude, double longitude*) {* if *(*optMember.isPresent*()) {* Member member = optMember.get*()*;

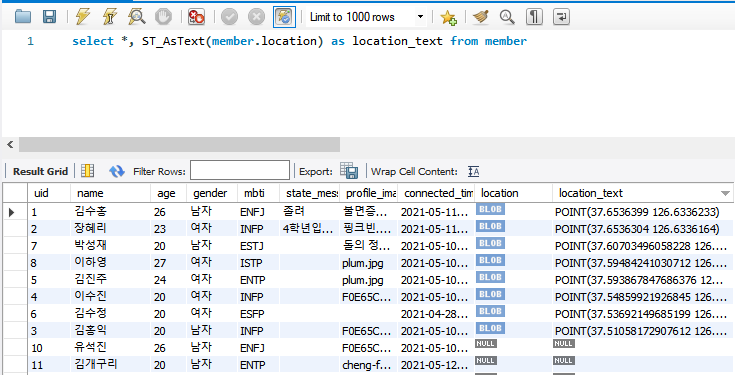
//위치좌표 Point로 변환

String pointWKT = String.*format(*"POINT(%s %s)", latitude, longitude*)*;  
 try *{* Point point = *(*Point*)* new WKTReader*()*.read*(*pointWKT*)*;  
 member.setLocation*(*point*)*;  
 edit*(*member*)*;  
 *log*.info*(*latitude + " " + longitude*)*;  
 *}* catch *(*ParseException e*) {* e.printStackTrace*()*;  
 *}  
 }  
}*

[**POINT**](https://dev.mysql.com/doc/refman/5.7/en/gis-class-point.html)**는 지도상의 경도, 위도 값을 표현하는 객체로 MySQL의** Spatial Data Type중 하나입니다. java에서 POINT타입으로 MySQL DB에 값을 전달하기 위해서는 org.locationtech.jts.geom.Point클래스를 사용하여야합니다.

import org.locationtech.jts.geom.Point;

public class Member *{* // private static final long serialVersionUID = 1L;  
 @Id  
 @NonNull  
 private String uid; //firebase uid  
 private String name;  
 private String gender;  
 private int age;  
 private String mbti;  
 private String stateMessage;  
 private String profileImage;  
 private LocalDateTime connectedTimeAt;  
 @JsonSerialize*(*using = GeometrySerializer.class*)* @JsonDeserialize*(*contentUsing = GeometryDeserializer.class*)* @Column*(*name = "location"*)* private Point location; //추가  
  
*}*



## 주변 유저 정보를 유저 마커로 생성

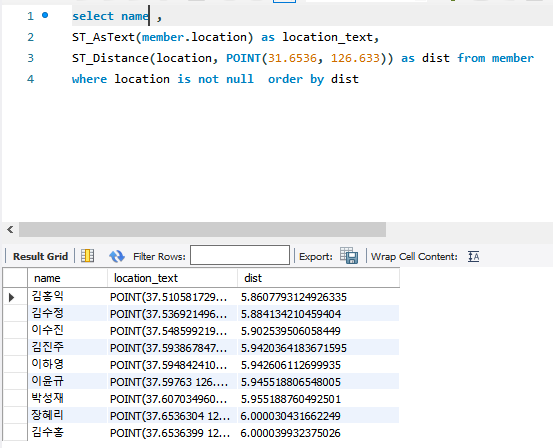
### 현재 위치에서 가까운 위치에 있는 유저 n명의 리스트 반환

JPA의 @Query 어노테이션을 사용하여 쿼리를 직접 작성하였습니다. 해당 쿼리는 member테이블의 location을 (x,y)와 가까운 순으로 정렬한 후 그 중 n명의 정보를 리스트로 반환합니다.

두 지점사이의 거리를 계산할 때는 MySQL의 [ST\_Distance](https://dev.mysql.com/doc/refman/8.0/en/spatial-relation-functions-object-shapes.html#function_st-distance)함수를 사용하였습니다. 이를 통해 두 지점의 거리를 미터단위로 계산할 수 있습니다.

public interface MemberRepository *{* Member save*(*Member member*)*;  
 Optional*<*Member*>* findByName*(*String name*)*;  
 List*<*Member*>* findAll*()*;  
 Optional*<*Member*>* findByUid*(*String uid*)*;

//추가  
 @Query*(*value = "select \* from member where location is not null order by ST\_Distance(location, POINT(?1, ?2)) limit ?3", nativeQuery=true*)* List*<*Member*>* findNearPoint*(*double x, double y, int n*)*;  
*}*



### 유저 리스트를 json형태로 변환

호출된 유저정보는 json형태로 변환되어 리턴됩니다.

@RequestMapping*(*value = *{*"/matching/near/{x}/{y}"*}*, method = RequestMethod.*GET)* @ResponseBody*()* public List*<*Member*>* getNearUser*(*@PathVariable String x, @PathVariable String y*) {  
  
 log*.info*(*"근처유저 가져오기 x:" + x + " y:" + y*)*;  
 Double latitude = Double.*parseDouble(*x*)*;  
 Double longitude = Double.*parseDouble(*y*)*;  
  
 List*<*Member*>* members = memberService.findNearUser*(*latitude, longitude, NUM\_NEAR\_USER*)*;  
  
 return members;  
  
 *}  
}*

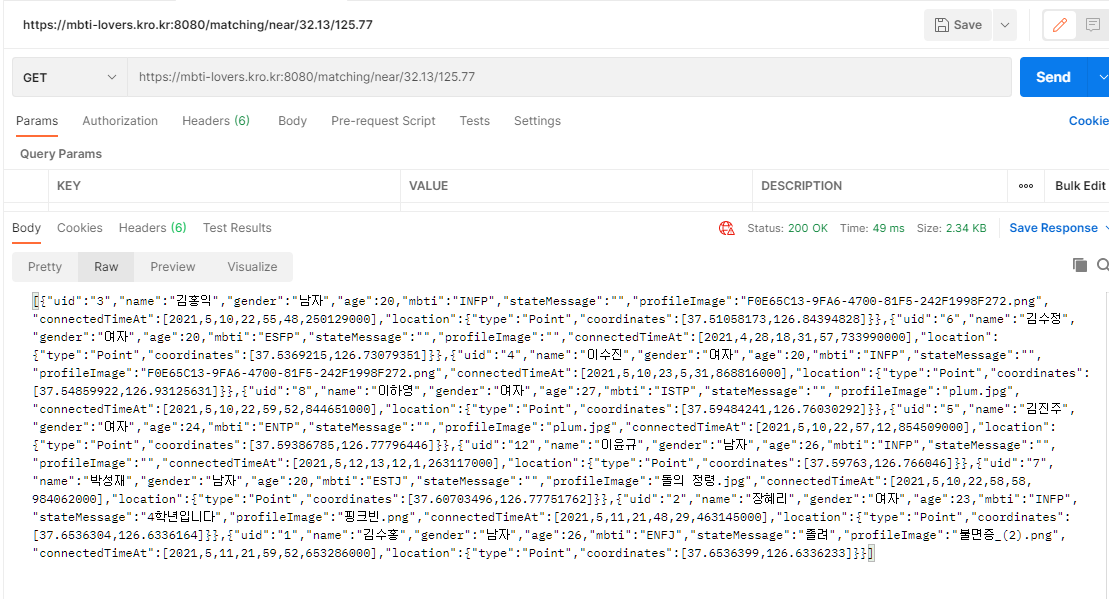


그림 json 호출 결과

REST API을 통해 json을 받아오는 데 성공하면 이를 활용하여 유저 마커 및 마커 트레이서를 만듭니다. 마커는 유저의 위치에 표시되는 원형의 아이콘이고 마커 트레이서는 마커가 지도영역 밖을 벗어났을 때 이를 추적하여 마커의 방향을 보여주는 커스텀 오버레이입니다.

|  |  |
| --- | --- |
| 그림 2마커 예시 | 그림 3마커 트레이서 예시 |

//마커 생성  
function makerNearUserMarker*(*json*){* ***console***.log*(*"가까운 유저 10명의 마커를 만듭니다."*)*;  
 json.forEach*(*function *(*member*) {* var uid = member.uid;  
 var position = new kakao.maps.LatLng*(*member.location.coordinates*[*0*]*, member.location.coordinates*[*1*])*;  
 var name = member.name;  
 const mbti = member.mbti;  
  
 var imageSrc = ***S3url*** + uid + '/' + encodeURI*(*member.profileImage*)*;  
 if *(*member.profileImage == ""*) {* imageSrc = ***defaultProfileImgPath***;  
 *}* // console.log(imageSrc);  
  
 var markerContent = createMarkerContent*(*imageSrc*)*;  
 var marker = createMarker*(*position, markerContent*)*;  
 createTracer*(*marker, imageSrc*)*;  
  
 //말풍선 오버레이  
 var content = "[" + mbti + "]" + name;  
 var speechBubbleOverlay = createSpeechBubbleOverlay*(*marker, uid, createSpeechBubbleContent*(*content*))*;  
  
 speechBubbleOverlay.setMap*(****map****)*;  
  
 *})*;  
*}*

마커 트레이서 관련 javascript는

결과물은 다음과 같습니다.



## 참조

[geolocation으로 마커 표시하기 - Kakao 지도 Web API](https://apis.map.kakao.com/web/sample/geolocationMarker/)

[Spring JPA를 활용한 MySQL 위치 데이터 다루기](https://momentjin.tistory.com/136)