

■ 지도

● 필요 기술 / 라이브러리

- HTML Design
- Database
- Model
 - class Point
- Django View
- Django Template
- Kakao API (기본 지도, 여러 개 마커 표시)
- AJAX (주변 장소 데이터 가져오기)
- 위도/경도 이용 반경 계산 공식

■ 지도 - 기본

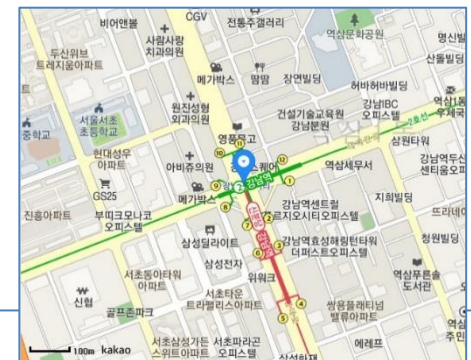
● templates/map.html (appkey : **da146a2a6ff6a54903c1d2a7caecd1c7**)

```
{% extends 'base.html' %}
{% block content %}

<div class="jumbotron">
  <div class="container text-center">
    <div id="map" style="width:500px; height:400px; margin:0 auto;"></div>
  </div>
</div>

<script src="http://dapi.kakao.com/v2/maps/sdk.js?appkey=발급받은키"></script>
<script>
var container = document.getElementById('map');
var options = {
  center : new kakao.maps.LatLng(37.4980239, 127.027572), // 강남역
  level : 4
};
var map = new kakao.maps.Map(container, options);
var markerPosition = new kakao.maps.LatLng(37.4980239, 127.027572);
var marker = new kakao.maps.Marker({
  position: markerPosition
});
marker.setMap(map);
</script>

{% endblock %}
```



■ 지도 - 기본

● board/views.py

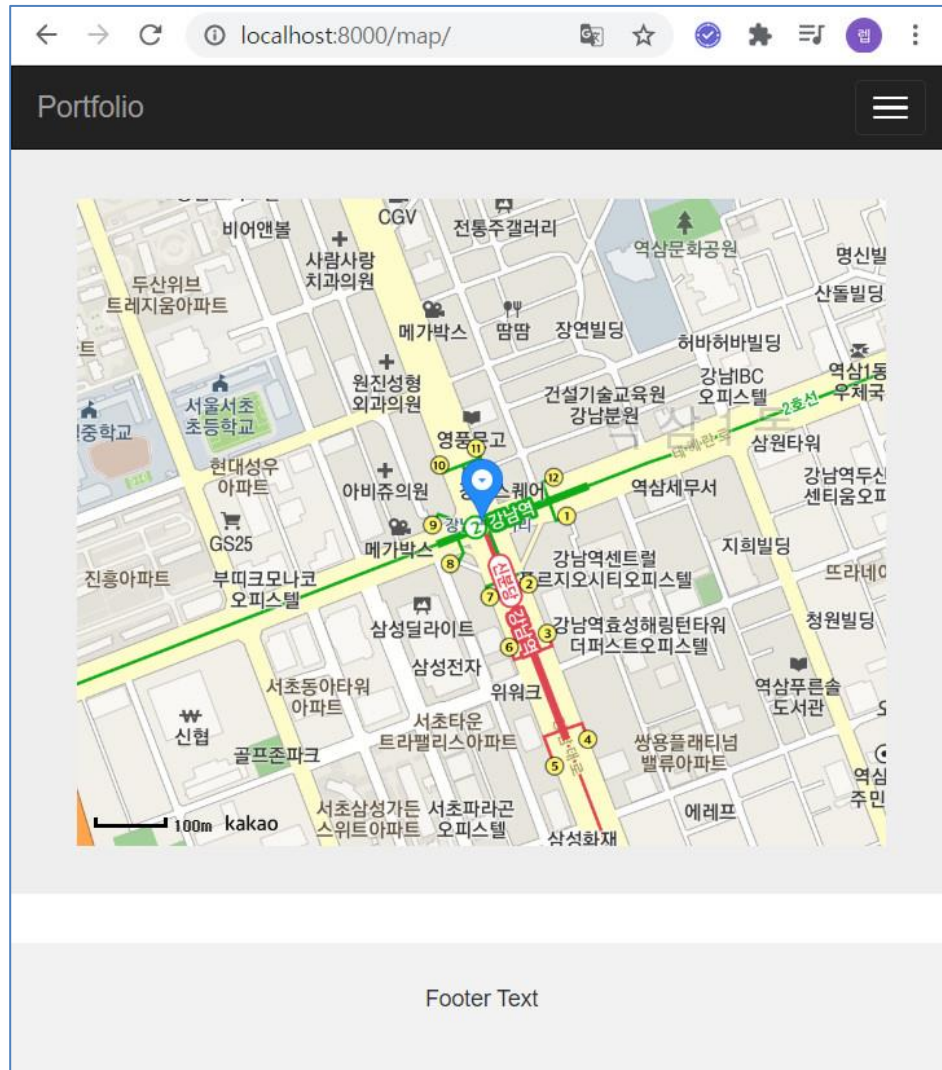
```
def map(request):  
    return render(request, 'map.html')
```

● board/urls.py

```
...  
  
urlpatterns = [  
    ...  
    path('map/', views.map),  
]
```

■ 지도 - 기본

● <http://localhost:8000/map/>



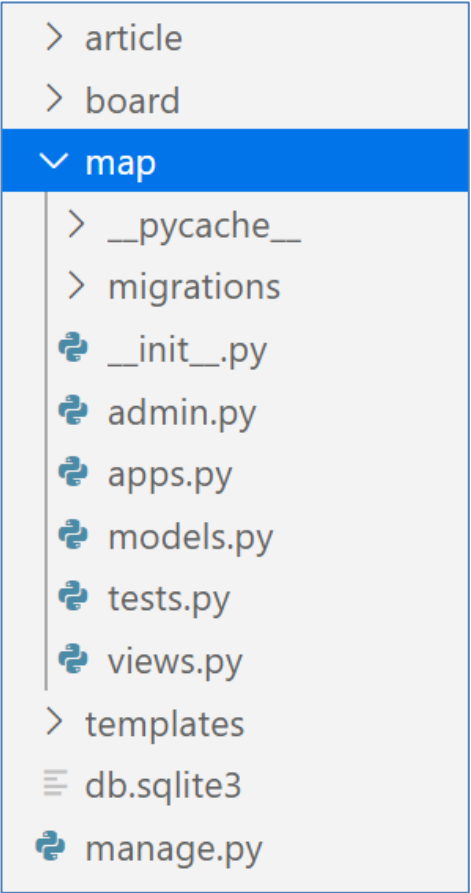
■ 지도 - 여러개 마커 표시

● App 생성

- python manage.py startapp map

● board/settings.py

```
...  
# Application definition  
INSTALLED_APPS = [  
    'article',  
    'map',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]  
...
```



- > article
- > board
- ▼ map
 - > __pycache__
 - > migrations
 - 🔗 __init__.py
 - 🔗 admin.py
 - 🔗 apps.py
 - 🔗 models.py
 - 🔗 tests.py
 - 🔗 views.py
- > templates
- ≡ db.sqlite3
- 🔗 manage.py

■ 지도 - 여러개 마커 표시





● map/models.py

```
from django.db import models

class Point(models.Model):
    title = models.CharField(max_length=100)
    lat = models.FloatField()
    lng = models.FloatField()
```

● migration

- python manage.py makemigrations
- python manage.py migrate

map_point		CREATE TABLE "map_point" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "title" varchar(100) NOT NULL, "lat" real NOT NULL, "lng" real NOT NULL)
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 title	varchar(100)	"title" varchar(100) NOT NULL
 lat	real	"lat" real NOT NULL
 lng	real	"lng" real NOT NULL

■ 지도 - 여러개 마커 표시

● point 테이블에 데이터 직접 입력

SQL 1

```
1 insert into map_point (id, title, lat, lng)
2 values (1, '코엑스', 37.5121917, 127.0566319);
3
4 insert into map_point (id, title, lat, lng)
5 values (2, '63빌딩', 37.5192502, 126.9400105);
6
7 insert into map_point (id, title, lat, lng)
8 values (3, '서울역', 37.5544681, 126.9686187);
9
10 insert into map_point (id, title, lat, lng)
11 values (4, '김포공항', 37.5578738, 126.8004231);
12
13 insert into map_point (id, title, lat, lng)
14 values (5, '정동진해변', 37.6908032, 129.032434);
```

```
insert into map_point (id, title, lat, lng)
values (1, '코엑스', 37.5121917, 127.0566319);
```

```
insert into map_point (id, title, lat, lng)
values (2, '63빌딩', 37.5192502, 126.9400105);
```

```
insert into map_point (id, title, lat, lng)
values (3, '서울역', 37.5544681, 126.9686187);
```

```
insert into map_point (id, title, lat, lng)
values (4, '김포공항', 37.5578738, 126.8004231);
```

```
insert into map_point (id, title, lat, lng)
values (5, '정동진해변', 37.6908032, 129.032434);
```

테이블(T):

map_point

	id	title	lat	lng
	필터	필터	필터	필터
1	1	코엑스	37.5121917	127.0566319
2	2	63빌딩	37.5192502	126.9400105
3	3	서울역	37.5544681	126.9686187
4	4	김포공항	37.5578738	126.8004231
5	5	정동진해변	37.6908032	129.032434

■ 지도 - 여러개 마커 표시

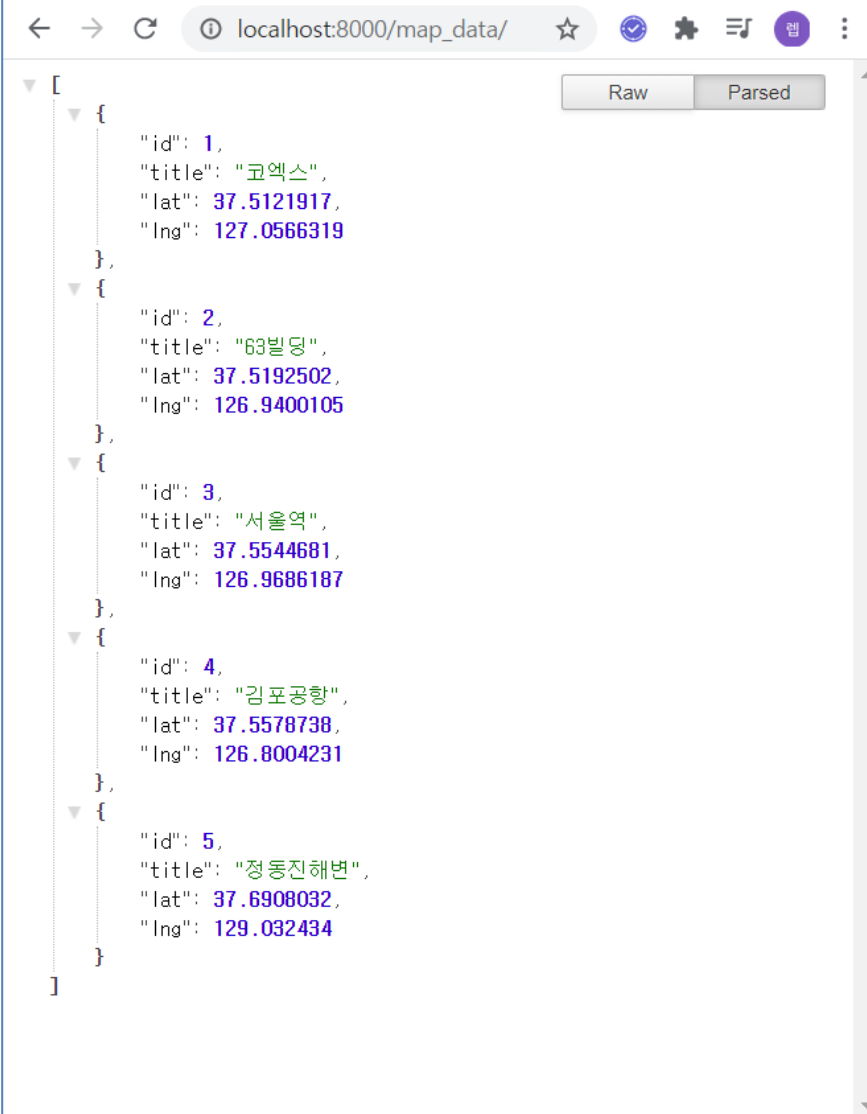
● board/views.py

```
from django.http import JsonResponse # JSON 응답
from map.models import Point
from django.forms.models import model_to_dict

def map_data(request):
    data = Point.objects.all()
    map_list = []
    for d in data:
        d = model_to_dict(d) # QuerySet -> Dict
        map_list.append(d)
    # dict가 아닌 자료는 항상 safe=False 옵션 사용
    return JsonResponse(map_list, safe=False)
```


■ 지도 - 여러개 마커 표시

● http://localhost:8000/map_data/



The screenshot shows a web browser window with the address bar displaying `localhost:8000/map_data/`. The main content area shows a JSON array of 5 objects, each representing a location marker. The JSON is displayed in a 'Parsed' view, with a 'Raw' button visible at the top right of the content area. The JSON data is as follows:

```
[
  {
    "id": 1,
    "title": "코엑스",
    "lat": 37.5121917,
    "lng": 127.0566319
  },
  {
    "id": 2,
    "title": "63빌딩",
    "lat": 37.5192502,
    "lng": 126.9400105
  },
  {
    "id": 3,
    "title": "서울역",
    "lat": 37.5544681,
    "lng": 126.9686187
  },
  {
    "id": 4,
    "title": "김포공항",
    "lat": 37.5578738,
    "lng": 126.8004231
  },
  {
    "id": 5,
    "title": "정동진해변",
    "lat": 37.6908032,
    "lng": 129.032434
  }
]
```

■ 지도 - 여러개 마커 표시

● templates/map.html (appkey : **da146a2a6ff6a54903c1d2a7caecd1c7**)

```
...
var map = new kakao.maps.Map(container, options);
var markerPosition = new kakao.maps.LatLng(37.4980239, 127.027572);
var marker = new kakao.maps.Marker({
    position: markerPosition
});
marker.setMap(map);
```

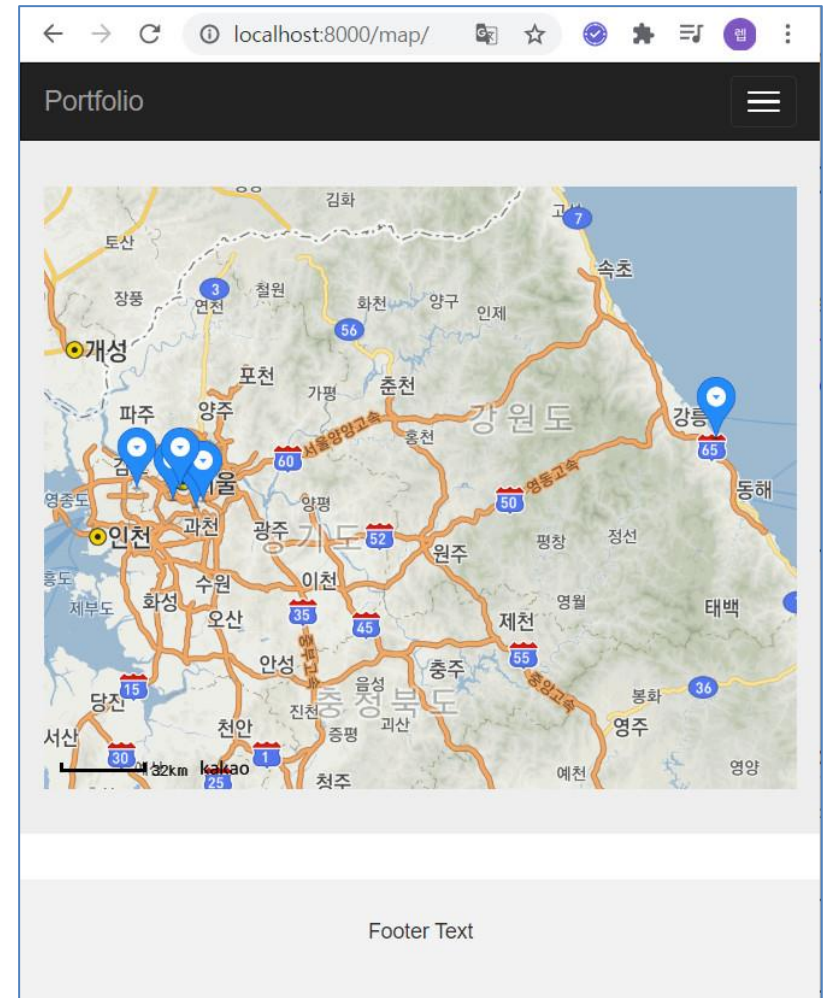
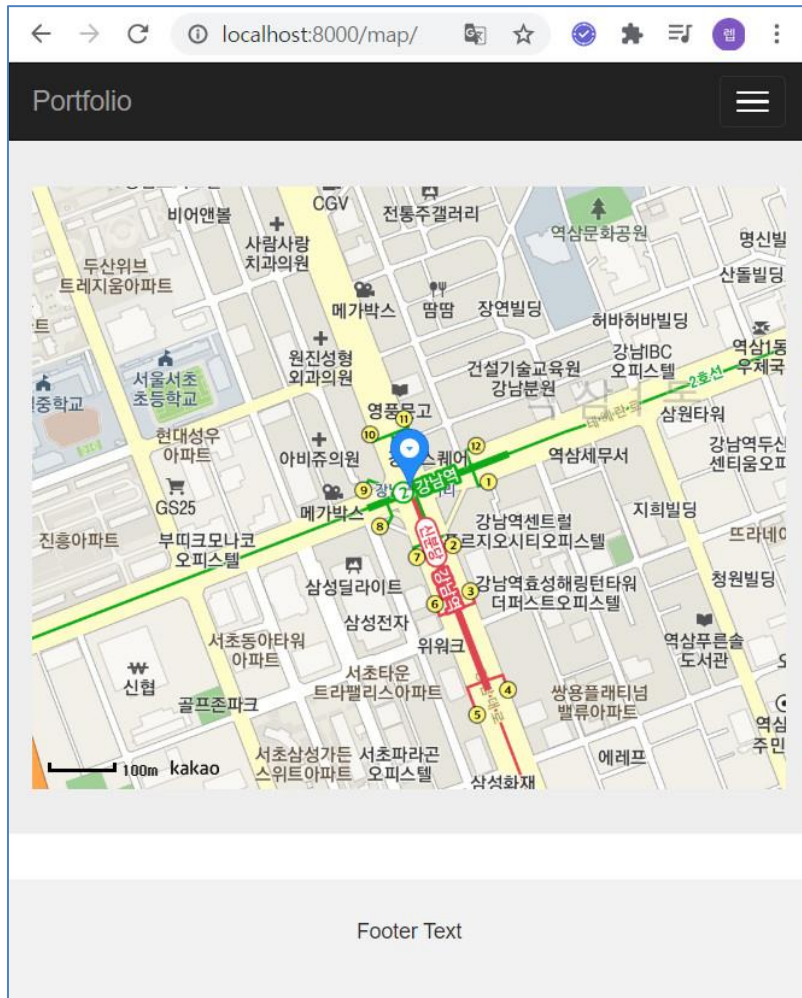
```
$.ajax({
    url: '/map_data/',
    data: {},
    success: function(res) {
        for(var i = 0; i < res.length; i++) {
            var marker = new kakao.maps.Marker({
                map : map, // 마커를 표시할 지도
                position : new kakao.maps.LatLng(res[i].lat, res[i].lng),
                title : res[i].title,
            });
        }
    }
});
```

</script>

{% endblock %}

■ 지도 - 여러개 마커 표시

● <http://localhost:8000/map/>



■ 지도 - 반경 거리 기준 표시 (파이썬 함수)

● templates/map.html (appkey : **da146a2a6ff6a54903c1d2a7caecd1c7**)

```
$.ajax({  
  url: '/map data/',  
  data: {  
    'lat' : 37.4980239, 'lng' : 127.027572  
  },  
  success: function(res) {  
    for(var i = 0; i < res.length; i++) {  
      var marker = new kakao.maps.Marker({  
        map : map, // 마커를 표시할 지도  
        position : new kakao.maps.LatLng(res[i].lat, res[i].lng),  
        title : res[i].title,  
      });  
    }  
  }  
});
```

■ 지도 - 반경 거리 기준 표시 (파이썬 함수)

● board/views.py (거리 계산 함수 추가)

```
import math
def distance(lat1, lng1, lat2, lng2) :
    theta = lng1 - lng2
    dist1 = math.sin(deg2rad(lat1)) * math.sin(deg2rad(lat2))

    dist2 = math.cos(deg2rad(lat1)) * math.cos(deg2rad(lat2))
    dist2 = dist2 * math.cos(deg2rad(theta))

    dist = dist1 + dist2

    dist = math.acos(dist)
    dist = rad2deg(dist) * 60 * 1.1515 * 1.609344

    return dist

def deg2rad(deg):
    return deg * math.pi / 180.0

def rad2deg(rad):
    return rad * 180.0 / math.pi
```

■ 지도 - 반경 거리 기준 표시 (파이썬 함수)

● board/views.py (거리 계산 함수 적용)

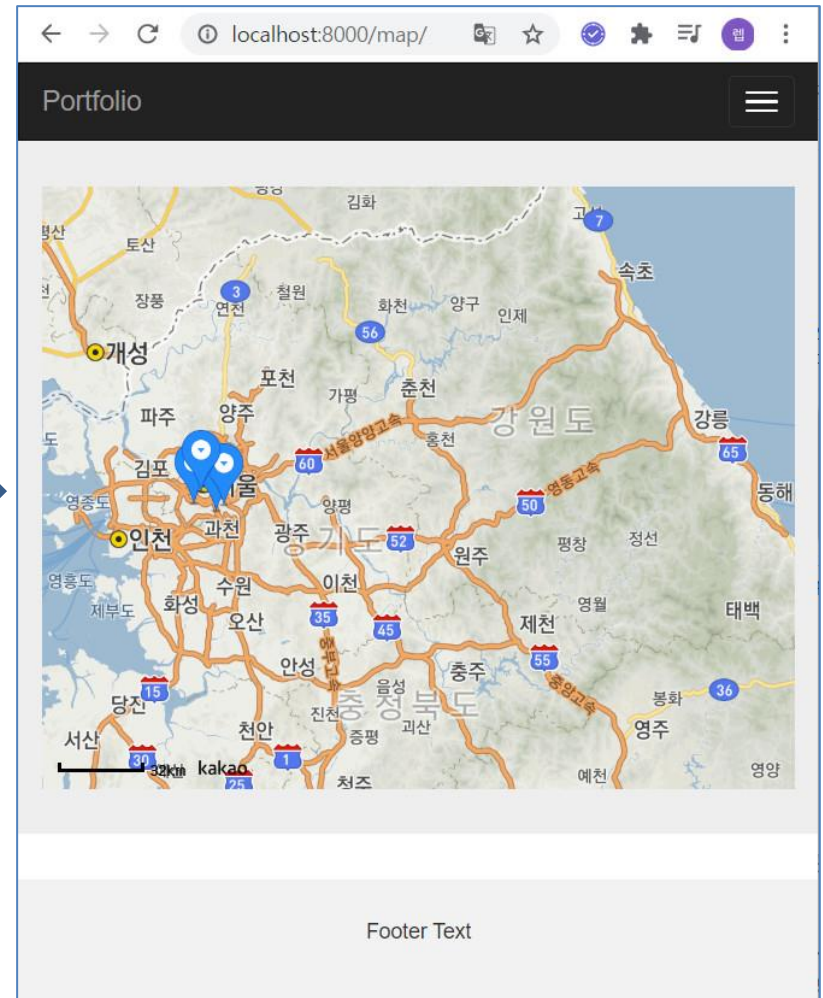
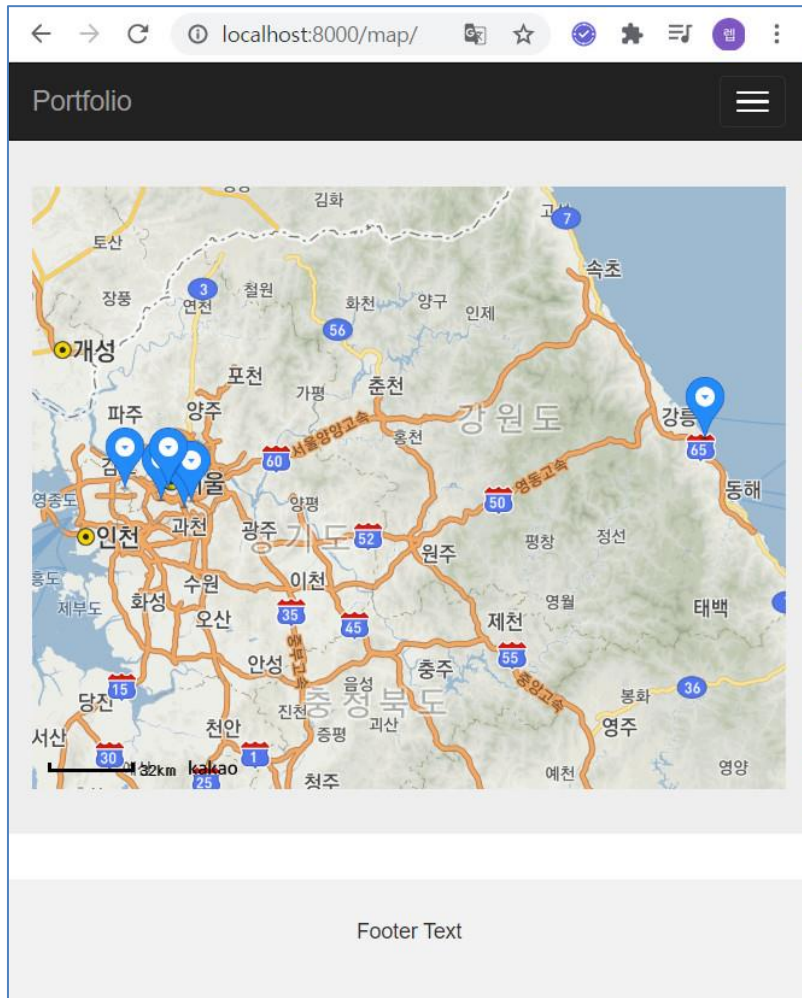
```
def map_data(request):
    data = Point.objects.all()
    lat = request.GET.get('lat')
    lng = request.GET.get('lng')
    map_list = []
    for d in data:
        d = model_to_dict(d)  # QuerySet -> Dict

        dist = distance(float(lat), float(lng), d['lat'], d['lng'])
        if(dist <= 10):  # 10km 이내의 장소만 응답결과로 저장
            map_list.append(d)

    # dict가 아닌 자료는 항상 safe=False 옵션 사용
    return JsonResponse(map_list, safe=False)
```

■ 지도 - 반경 거리 기준 표시 (파이썬 함수)

● <http://localhost:8000/map/>



■ 지도 - 반경 거리 기준 표시 (SQL)

● board/views.py (Native Query 사용) : **Oracle, MySQL** 등에서 사용

```
def map_data2(request):
    lat = request.GET.get('lat')
    lng = request.GET.get('lng')
    data = Point.objects.raw('''
        SELECT *,
            (6371 * acos(
                cos(radians(%s))
                * cos(radians(lat))
                * cos(radians(lng) - radians(%s))
                + sin(radians(%s))
                * sin(radians(lat)))) AS distance
        FROM map_point
        HAVING distance <= %s
        ORDER BY distance''' % (lat, lng, lat, 10))
    map_list = []
    for d in data:
        d = model_to_dict(d) # QuerySet -> Dict
        map_list.append(d)
    # dict가 아닌 자료는 항상 safe=False 옵션 사용
    return JsonResponse(map_list, safe=False)
```


■ 지도 - 반경 거리 기준 표시 (SQL)

● <http://localhost:8000/map/>

