Spring Boot + MongoDB

단계	주제	주요 내용
[1]	환경 구성	- MongoDB 설치 및 실행- Spring Boot + MongoDB 의존성 설정- application.yml 또는 .properties 로 접속 설정
[2]	Java 기반 설정 & MongoTemplate	- @Configuration 을 통한 MongoClient 및 MongoTemplate 직접 설정- MongoTemplate 의 기본 CRUD 사용
[3]	도메인 & 매핑 어노테이션	- @Document, @Id, @Field, @DBRef, @Transient 등 어노테이션 학습- POJO ↔ Document 매핑 구현
[4]	MongoRepository 기반 CRUD	- MongoRepository<T, ID> 상속- 기본 메서드 (save, findAll, delete, findById) 사용- 메서드 명 규칙 기반 쿼리 작성
[5]	Query, Criteria, Update DSL	- Query, Criteria, Update 클래스를 통한 조건 기반 쿼리 작성 - MongoTemplate 으로 실행 (find, updateFirst 등)
[6;	예외 처리 & 트랜잭션	- Mongo 예외 , Spring DataAccess 예외 자동 변환- @Transactional + 멀티 도큐먼트 트랜잭션 사용
[7]	도큐먼트 이벤트 리스너	- AbstractMongoEventListener<T> 상속 - onBeforeConvert, onAfterSave 등 라이프사이클 이벤트 핸들링
[8]	QueryDSL 연동	- 타입 안전 쿼리 작성: QEmp.emp.sal.gt(3000) - QueryDSL + Spring Data Mongo 설정 필요
[9]	GeoSpatial 기능	- 위치 필드(Point) 포함 도큐먼트 - 2dsphere 인덱스 생성, near, geoWithin 쿼리 실습

[1 단계] 환경구성

프로젝트 구조

```
SpringBootLab10/
├─── src/
|    └─── main/
|         ├─── java/
|         |    └─── com/ lab10/
|         |         ├─── SpringBootLab10Application.java        ← main 클래스
|         |         ├─── config/
|         |         |    └─── MongoConfig.java           ← [2 단계] MongoClient + MongoTemplate 설정
|         |         ├─── domain/
|         |         |    ├─── Emp.java        ← [3 단계] @Document, @Id 등 도메인 클래스
|         |         |    └─── Dept.java
|         |         ├─── service/
|         |         |    └─── EmpService.java      ← [4~5 단계] MongoTemplate 기반 CRUD
|         |
|         |         ├─── controller/
|         |         |    └─── EmpController.java       ← [5 단계] REST API
|         |         └─── repository/                        ← MongoRepository 용 패키지
|         |              └─── EmpRepository.java
|         └─── resources/
|              ├─── application.yml
|              └─── static/
└─── pom.xml                                          ← Spring Boot + MongoDB 의존성
```

dept.json, emp.json 파일 업로드

```
//기존 컬렉션 삭제
mongosh myemp --eval "db.emp.drop()"
mongosh myemp --eval "db.dept.drop()"

// 새로 생성
mongoimport --db myemp --collection dept --file dept.json --jsonArray
mongoimport --db myemp --collection emp --file emp.json --jsonArray
```

pom.xml

```
<!-- Spring Data MongoDB -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-mongodb</artifactId>
    </dependency>
```

application.yml

```
spring:
  data:
    mongodb:
      uri: mongodb://localhost:27017/myemp
  application:
    name: SpringBootLab10
server:
  port: 8080
```

[2 단계] Java 기반 설정과 MongoTemplate 사용

- @Configuration 으로 MongoClient 및 MongoTemplate 수동 설정
- MongoTemplate 을 사용한 CRUD 구조 구성

src/main/java/com/lab10/config/MongoConfig.java

```
@Configuration
public class MongoConfig {

    @Bean
    public MongoClient mongoClient() {
        return MongoClients.create("mongodb://localhost:27017");
    }

    @Bean
    public MongoTemplate mongoTemplate() {
        return new MongoTemplate(mongoClient(), "myemp");
    }
}
```

[3 단계] 도메인 & 매핑 어노테이션

MongoDB 에서 관계를 맺는 2 가지 방식

방식	설명	현재 구조
참조 저장 (@DBRef)	외부 문서를 참조하는 방식 , Spring Data 에서 조인처럼 보이지만 실사용 시 제한 많음	사용 안 함
수동 조회 (애플리케이션에서)	필요 시 service 에서 두 컬렉션을 나눠 조회하고 합침 (JOIN 수동 처리)	현재 방식

Emp. src/main/java/com/lab10/domain/Emp.java , Dept.java

```
@Document("emp")
@NoArgsConstructor
@AllArgsConstructor
public class @Data  Emp {
    @Id
    private Integer empno;
    private String ename;
    private String job;
    private Integer sal;
    private Integer dept;
}
```

```
@Document("dept")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Dept {
    @Id
    private Integer deptno;
    private String dname;
    private String loc;

    //  지도 API 마커 표시를 위한 위도/경도 정보
    private Double latitude;     // 위도
    private Double longitude;    // 경도
}
```

지도 API 와 연동 시 활용 포인트

필드	역할
latitude, longitude	네이버 지도에서 마커 표시용 위치 좌표로 사용
dname	마커 타이틀로 사용
REST API → /api/depts	위치 정보 조회용 JSON API 응답에 포함

[4 단계] com.lab10.repository.DeptRepository, EmpRepository

```
public interface DeptRepository extends MongoRepository<Dept, Integer> {
        Dept findByDname(String dname);

        //부서명에 특정 키워드가 포함된 부서 목록 조회
    List<Dept> findByDnameContaining(String keyword);
}
```

```
public interface EmpRepository extends MongoRepository<Emp, Integer> {
        List<Emp> findByEnameContaining(String keyword);
    List<Emp> findBySalGreaterThanEqual(Integer sal);
    List<Emp> findByDept(Integer deptno);
}
```

[5 단계] com.lab10.service.EmpService, DeptService

com.lab10.service.EmpService

메서드명(매개변수)	리턴형	설명
save(Emp emp)	void	사원 정보를 MongoDB 에 저장 (등록 또는 수정)
findAll()	List<Emp>	모든 사원 목록을 조회
findByEmpno(int empno)	Emp	사번으로 단일 사원 조회
delete(int empno)	void	사번으로 사원 삭제
searchByName(String keyword)	List<Emp>	이름에 특정 키워드가 포함된 사원 목록 조회
findBySalaryAbove(int sal)	List<Emp>	급여가 특정 값 이상인 사원 목록 조회
findByDeptno(int deptno)	List<Emp>	부서번호로 해당 부서의 사원 목록 조회

com.lab10.service. DeptService

메서드명(매개변수)	리턴형	설명
save(Dept dept)	void	부서 정보를 MongoDB 에 저장 (등록 또는 수정)
findAll()	List<Dept>	모든 부서 목록을 조회
findByDeptno(int deptno)	Dept	부서번호로 단일 부서 조회
findByDnameContaining(String keyword)	List<Dept>	부서명에 특정 키워드가 포함된 부서 목록 조회
findByDname(String dname)	Dept	부서명을 기준으로 단일 부서 조회

[6 단계] controller (com.lab10.controller)

EmpController API 목록

HTTP 메서드	URL 경로	설명
POST	/emp	사원 등록 또는 수정
GET	/emp	전체 사원 목록 조회
GET	/emp/{empno}	사번으로 단일 사원 조회
DELETE	/emp/{empno}	사번으로 사원 삭제
GET	/emp/search?keyword=...	이름 포함 키워드로 사원 검색
GET	/emp/sal?min=...	급여 이상 사원 검색
GET	/emp/dept/{deptno}	특정 부서 소속 사원 목록 조회

DeptController API 목록

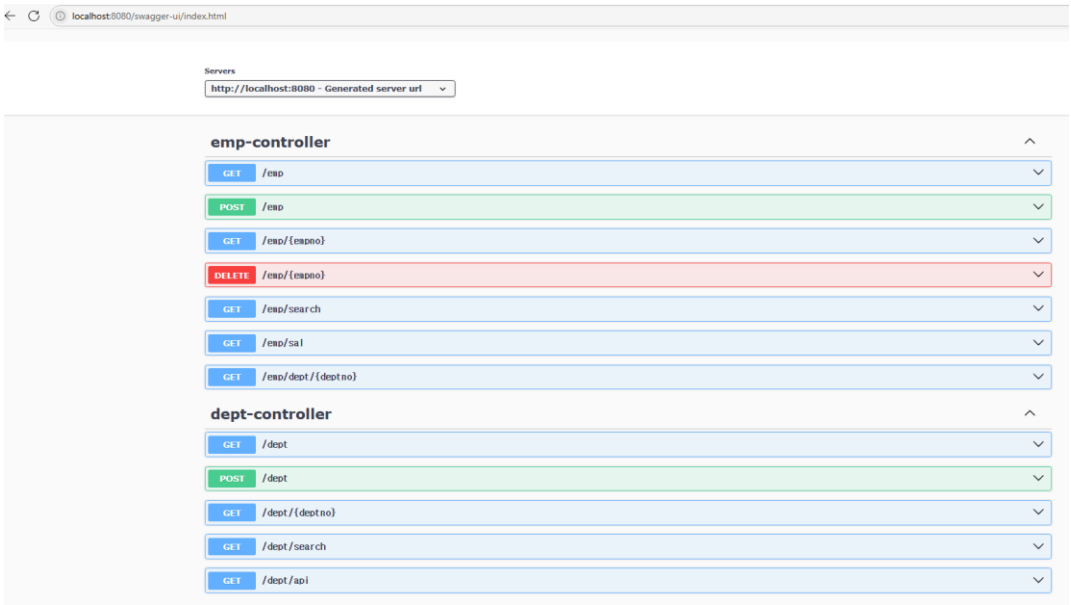
HTTP 메서드	URL 경로	설명
POST	/dept	부서 등록 또는 수정
GET	/dept	전체 부서 목록 조회
GET	/dept/{deptno}	부서번호로 단일 부서 조회
GET	/dept/api	네이버 지도용: 모든 부서의 위치 정보(JSON) 반환
GET	/dept/search?keyword=...	부서명 키워드로 검색

[7 단계] Swagger 의존성 추가

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.8.8</version>
</dependency>
```

Swagger UI 접속

http://localhost:8080/swagger-ui/index.html



[8 단계] 지도 마커 표시를 위한 /dept/api 호출 및 HTML 연동

- /map 페이지에서 Google Map 띄우기
- /dept/api 에서 받은 데이터를 기반으로 지도에 마커 표시
- 마커 클릭 시 dname(부서명) 표시

1. Prerequisites: Google Maps API 등록 및 활성화 절차

1-1. 구글 클라우드 콘솔 접속

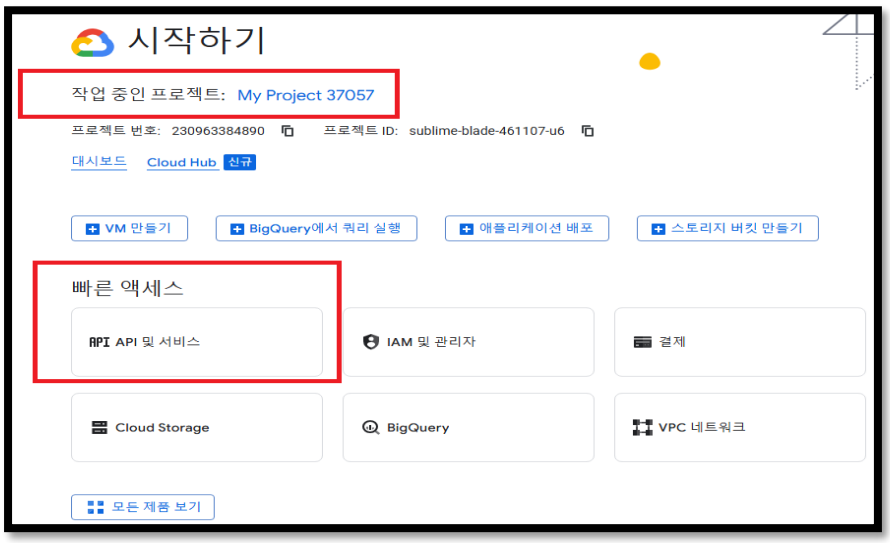
- URL: https://console.cloud.google.com/
- Google 계정으로 로그인

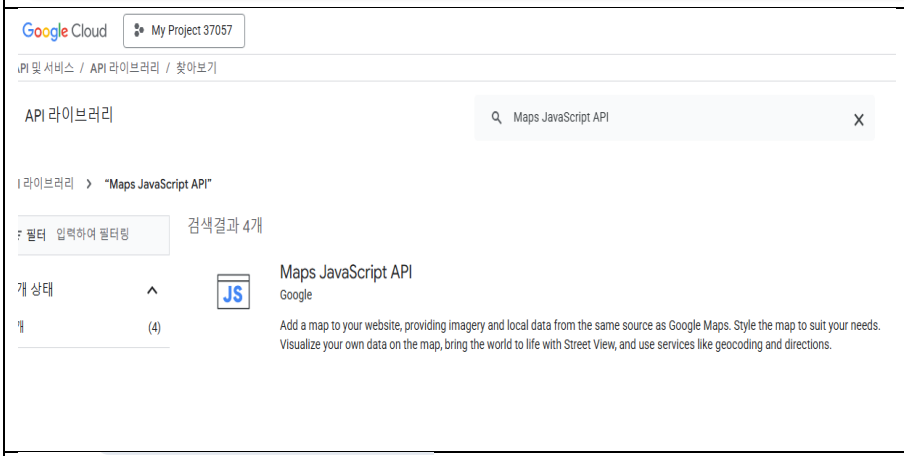
1-2. 새 프로젝트 생성

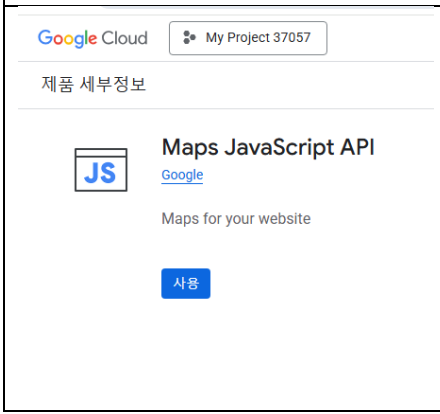
1. 상단의 **프로젝트 선택** 클릭
2. **[+ 새 프로젝트]** 버튼 클릭
3. 프로젝트 이름 입력 (예: SpringBootMapProject)
4. **생성** 클릭

1-3. Maps JavaScript API 활성화

1. 왼쪽 메뉴 → **API 및 서비스 > 라이브러리**
2. 검색창에 "Maps JavaScript API" 입력
3. **Maps JavaScript API** 클릭
4. 우측 상단의 **[사용]** 버튼 클릭


시작하기
작업 중인 프로젝트: My Project 37057
프로젝트 번호: 230963384890
프로젝트 ID: sublime-blade-461107-u6
대시보드
Cloud Hub 신규
VM 만들기
BigQuery에서 쿼리 실행
애플리케이션 배포
스토리지 버킷 만들기
빠른 액세스
API API 및 서비스
IAM 및 관리자
결제
Cloud Storage
BigQuery
VPC 네트워크
모든 제품 보기


Google Cloud
My Project 37057
API 및 서비스 / API 라이브러리 / 찾아보기
API 라이브러리
Maps JavaScript API
라이브러리 › "Maps JavaScript API"
필터 입력하여 필터링
검색결과 4개
개 상태
(4)
JS
Maps JavaScript API
Google
Add a map to your website, providing imagery and local data from the same source as Google Maps. Style the map to suit your needs. Visualize your own data on the map, bring the world to life with Street View, and use services like geocoding and directions.


Google Cloud
My Project 37057
제품 세부정보
JS
Maps JavaScript API
Google
Maps for your website
사용

1-4 Google Maps API 키 발급 및 보안 설정

1. 사용자 인증 정보 메뉴로 이동

1. 왼쪽 메뉴에서 **API 및 서비스 > 사용자 인증 정보** 클릭
2. 상단의 **[+ 사용자 인증 정보 만들기]** 버튼 클릭
3. **API 키** 선택

✅ 잠시 후 팝업으로 API 키가 생성

AIzaSy**********hB2G9z8

2. 보안 설정 (HTTP referrer 제한 – 웹에서 사용할 경우 권장)

발급된 API 키 오른쪽 연필 아이콘(수정)을 눌러 **사용 제한** 설정.

애플리케이션 제한

- HTTP referrers (웹사이트) 선택

허용 referrer 항목 추가 (예: 로컬 개발용)

복사편집
http://localhost:8080/*

향후 실제 배포 시에는 배포 도메인도 함께 추가하
https://yourdomain.com/*

3. 저장

- 제한 설정이 완료되었으면 **저장** 클릭

결과: 완성된 API 키를 프로젝트에 적용

application.yml

```
google:
  map:
    api-key: AIzaSy***********hB2G9z8
```

2. application.yml 의 Google Maps API 키를 전달

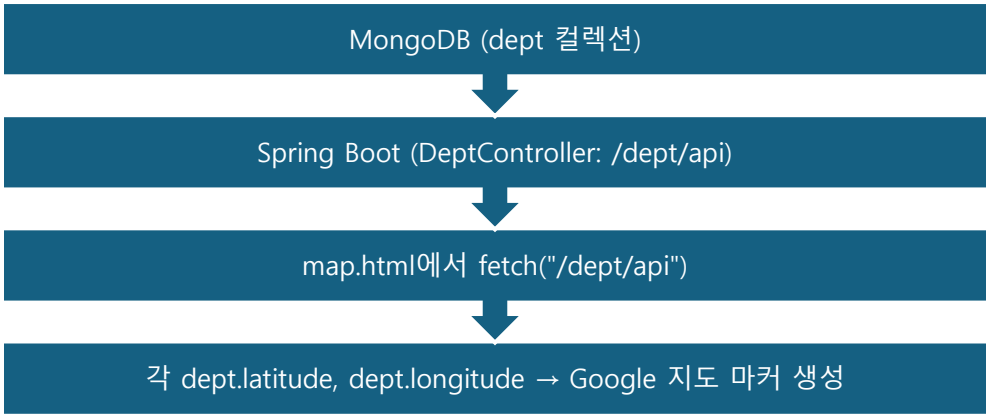
Controller

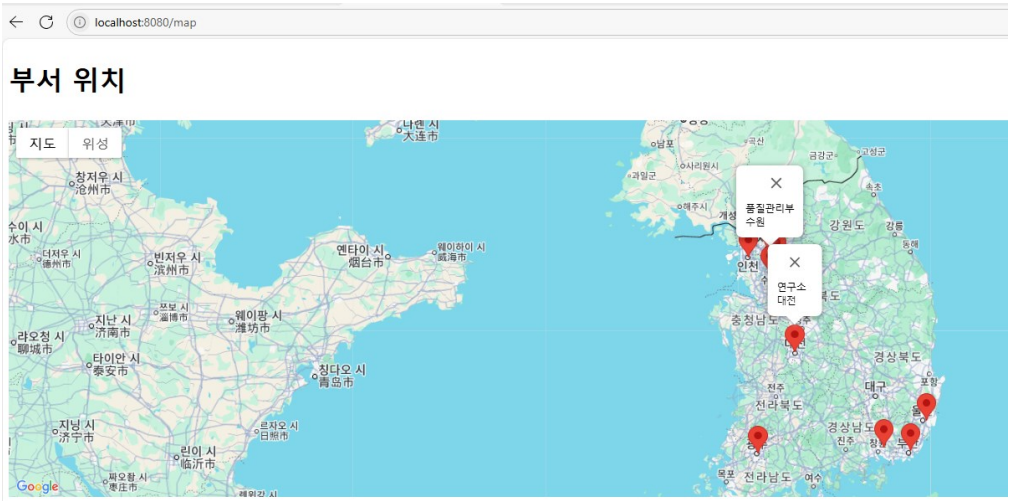
```
@Controller
public class MapController {

    @Value("${google.map.api-key}")
    private String apiKey;

    @GetMapping("/map")
    public String showMapPage(Model model) {
        model.addAttribute("apiKey", apiKey);
        return "map";
    }
}
```

Google Maps JavaScript API 샘플 갤러리

https://developers.google.com/maps/documentation/javascript/examples/marker-simple

문서	링크
전체 JavaScript API 문서	https://developers.google.com/maps/documentation/javascript/overview
마커(marker) 예제 목록	https://developers.google.com/maps/documentation/javascript/examples/#markers
InfoWindow 예제	https://developers.google.com/maps/documentation/javascript/examples/infowindow-simple





[추가 코드 01] EmpController 의 부서번호로 사원 검색하는 명령을 사용해서 커 클릭 시, 해당 부서에 소속된 사원 목록을 출력하자.

```
// 7. 부서번호로 사원 검색
    @GetMapping("/dept/{deptno}")
    public List<Emp> findByDept(@PathVariable Integer deptno) {
        return empService.findByDeptno(deptno);
    }
```

기능	설명
마커 클릭	부서명 + 해당 부서 소속 사원 목록 출력
데이터	MongoDB 의 emp 컬렉션에서 dept 필드 기준으로 조회
연동 방식	GET /emp/dept/{deptno} API 호출 → 마커 클릭 시 fetch 로 출력

Map.html 수정

```
            marker.addListener('click', () => {
                        fetch(`/emp/dept/${dept._id}`)
                            .then(response => response.json())
                            .then(emps => {
                                let content =
`<strong>${dept.dname}</strong><br>${dept.loc}<br>`;
                                content += `<ul style='margin:5px 0;padding-left:15px;'>`;
                                emps.forEach(emp => {
                                    content += `<li>${emp.ename} (${emp.job})</li>`;
                                });
                                content += `</ul>`;
                                infoWindow.setContent(content);
                                infoWindow.open(map, marker);
                            });
                    });
});
```

localhost:8080/map

위치

[실습 추가 코드 02]

MongoTemplate + Query/Criteria 사용

MongoTemplate 주요메소드

메서드	설명
find(query, Class<T>)	조건에 맞는 전체 목록 조회
findOne(query, Class<T>)	조건에 맞는 첫 문서 하나만 조회
updateFirst(query, update, Class<T>)	첫 번째 문서만 수정
updateMulti(query, update, Class<T>)	조건에 맞는 모든 문서 수정
remove(query, Class<T>)	조건에 맞는 문서 삭제

주요 클래스

클래스	설명
Query	조회 조건을 설정하는 DSL
Criteria	필드 조건을 표현하는 빌더
Update	수정할 필드와 값을 설정
MongoTemplate	위 DSL 을 실행해주는 핵심 클래스

Ex01: 특정 부서 사원의 급여 일괄 인상 (부서번호 10 → +500)

```
public void increaseSalary(int deptno, int amount) {
    Query query = new Query(Criteria.where("dept").is(deptno));
    Update update = new Update().inc("sal", amount); // +amount
    mongoTemplate.updateMulti(query, update, Emp.class);
}
```

Ex02: 조건에 맞는 사원 조회 (ex. 직책이 'CLERK')

```
public List<Emp> findClerks() {
    Query query = new Query(Criteria.where("job").is("CLERK"));
    return mongoTemplate.find(query, Emp.class);
}
```

Ex03: 부서번호 10 번 사원 중 급여 3000 초과하는 사람 조회

```
public List<Emp> findHighSalaryInDept10() {
    Query query = new Query();

query.addCriteria(Criteria.where("dept").is(10).and("sal").gt(3000));
    return mongoTemplate.find(query, Emp.class);
}
```

Ex04: ename 이 '홍길동'인 사원의 직책을 'MANAGER'로 변경

```
public void updateJobToManager(String ename) {
    Query query = new Query(Criteria.where("ename").is(ename));
    Update update = new Update().set("job", "MANAGER");
    mongoTemplate.updateFirst(query, update, Emp.class);
}
```

Ex05: loc 이 '서울'인 부서의 위도 변경

```
public void updateLatitudeForSeoul(double lat) {
    Query query = new Query(Criteria.where("loc").is("서울"));
    Update update = new Update().set("latitude", lat);
    mongoTemplate.updateMulti(query, update, Dept.class); // 여러 부서
가능성 있을 때
}
```