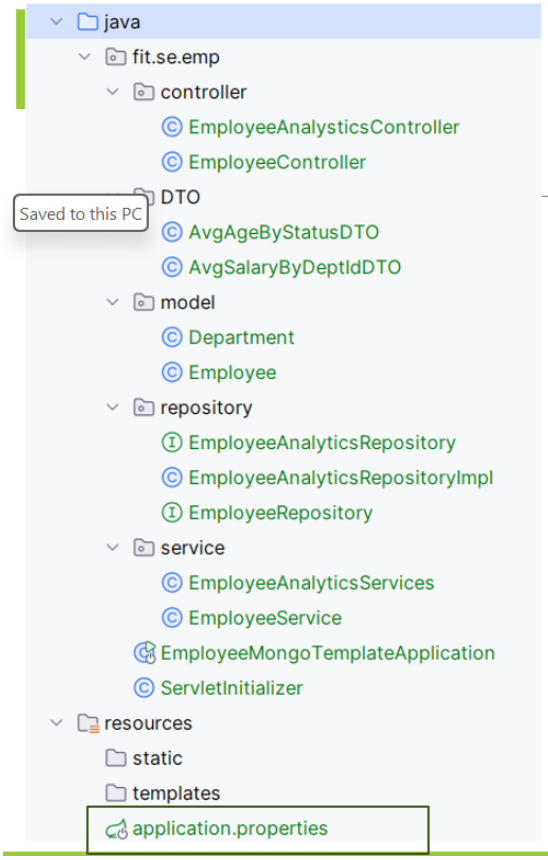

Example structure

New Project Spring Boot – Add dependencies
Step 0: Config Connection and Server
`application.properties`
Step 1: Entities Document
Step 2: Repository
- EmployeeRepository: MongoRepository
CRUD Method (Basic Query)
- EmployeeAnalyticsRepository: MongoTemplate (Complex Query - Aggregate)
Step 3: Service
Step 4: Controller
Step 5: (Optional) DTO define class for result of MongoTemplate, that sechema is different from entites documents.
Depend on the

Collection employees.json, department.json

Tạo Project Spring Boot sử dụng Spring Data Mongodb thực hiện các chức năng sau:

- CRUD Employees
- Hiển thị thông tin của các employees có salary cao nhất
- Hiển thị thông tin của các employees có Age cao nhất
- Xuất ra thông tin số employee và mức salary trung bình từng department
- Xuất ra thông tin số employee và mức salary trung bình từng loại status
- Xuất ra danh sách employee theo mức lương tăng dần từng theo department.

Hướng dẫn:

Bước 1: Entities Document

Employee.java

```
…
@Document(collection = "employees")
public class Employee {

    @Id
    @JsonIgnore   // ẩn không trả về cho client do Mongo quản lý, nhưng không
expose ra ngoài)
    private String id;// Mongodb generate _id

    private String empId;
    private String empName;
    private String email;
    private int age;
    private Double salary;
    private int status;
    private String deptId;

    // Constructors
    public Employee() {
    }

    public Employee(String empId, String empName, String email, int
age,Double salary, int status, String deptId) {
        this.empId = empId;
        this.empName = empName;
        this.email = email;
        this.age = age;
        this.salary = salary;
        this.status = status;
        this.deptId = deptId;
    }

    // Getters & Setters


}
```

Step 2: Repository

- (interface) EmployeeRepository: MongoRepository
  CRUD Method (Basic Query)

EmployeeRepository.java

```
@Repository
public interface EmployeeRepository extends MongoRepository<Employee, String>
{


        List<Employee> findAll();

        Employee findByEmpId(String empId);

        List<Employee> findByEmpNameContainingIgnoreCase(String name);

        List<Employee> findByStatus(int status);
```

```
        // Lấy employee theo deptId
        List<Employee> findByDeptId(String deptId);
}
```

- (interface)EmployeeAnalyticsRepository: MongoTemplate (Complex Query - Aggregate)

EmployeeAnalyticsRepository.java

```
@Repository
public interface EmployeeAnalyticsRepository {

    List<AvgAgeByStatusDTO> getCountandAvgAgeByStatus();
     List<AvgSalaryByDeptIdDTO> getCountandAvgSalaryByDept();
    public List<Document> findAllMaxEmployees();
  //…

}
```

- **EmployeeAnalyticsRepositoryImpl: MongoTemplate (Complex Query - Aggregate)**

EmployeeAnalyticsRepositoryImpl.java

```
@Repository
public class EmployeeAnalyticsRepositoryImpl implements
EmployeeAnalyticsRepository {
    private MongoTemplate mongoTemplate;

    public EmployeeAnalyticsRepositoryImpl(MongoTemplate mongoTemplate) {
        this.mongoTemplate = mongoTemplate;
    }


//  đếm số nhân viên và tính lương trung bình theo từng status
//    db.employees.aggregate([
//    {
//        $group: {
//            _id: "$status",
//            employeeCount: { $sum: 1 },
//            averageAge: { $avg: "$age" }
//        }
//    }
//])
    @Override
    public List<AvgAgeByStatusDTO> getCountandAvgAgeByStatus() {
        Aggregation agg = newAggregation(
                group("status")
                        .count().as("count")        // ánh xạ đúng tên field
DTO
                        .avg("age").as("avgAge"),
                project("count", "avgAge")
                        .and("_id").as("status")    // map _id -> status
        );
```

```
public class AvgSalaryByDeptIdDTO { 9 usages  new *
    private String deptId; 3 usages
    private int count; 3 usages
    private double avgSalary; 3 usages
```

```
        AggregationResults<AvgAgeByStatusDTO> results =
mongoTemplate.aggregate(agg, "employees", AvgAgeByStatusDTO.class);
        return results.getMappedResults();
    }

    //    Đếm số nhân viên theo từng mã phòng ban và tính độ tuổi trung bình
trong mỗi phòng ban
//db.employees.aggregate([
//    {
//        $group: {
//            _id: "$deptId",
//            count: { $sum: 1 },
//            avgSalary: { $avg: "$salary" }
//        }
//    }
//])
    @Override
    public List<AvgSalaryByDeptIdDTO> getCountandAvgSalaryByDept() {
        Aggregation agg = newAggregation(
                group("deptId")
                        .count().as("count")
                        .avg("salary").as("avgSalary"),
                project("count", "avgSalary")
                        .and("_id").as("deptId")
        );

        AggregationResults results = mongoTemplate.aggregate(agg,
"employees", AvgSalaryByDeptIdDTO.class);

        return results.getMappedResults();

    }

    // Xuất ra danh sách employee lớn tuổi nhất

//    db.employees.aggregate([
//    {
//        $group: {
//            _id: null,
//                    maxSalary: { $max: "$salary" }
//        }
//    },
//    {
//        $lookup: {
//            from: "employees",
//                    localField: "maxSalary",
//                    foreignField: "salary",
//                    as: "employees"
//        }
//    },
//    {
//        $unwind: "$employees"
//    },
//    {
//        $replaceRoot: { newRoot: "$employees" }
//    }
//])
```

```
    @Override
    public List<Document> findAllMaxEmployees() {
        Aggregation agg = newAggregation(
                group()
                        .max("salary").as("maxSalary"),
                lookup("employees", "maxSalary", "salary", "employees"),
                unwind("employees"),
                replaceRoot("employees")
        );
        AggregationResults<Document> maxSalaryResults =
mongoTemplate.aggregate(agg, "employees", Document.class);
        return maxSalaryResults.getMappedResults();
    }
}
```

Step 3: Service

Step 4:Controller

Step 5: DTO