```
Custom methods user-defined MongoRepository
                                                                                                                                                                     note:: finder methods in spring boot data mongoDB are similar to the finder
                   a) using finder/findBy methods (same as spring data jpa)
                                                                                                                                                                                   methods of spring boot data jpa
                   b) using @Query methods (best)
           Example on finder method
                                                                                                                                                                            OComparent
public dass Mangalleportinny Enderhöndiguery Menthasis Runner Implements Licon
Glassoulerd
printes Broducktepolatory prodilepo;
Glassoulerd
printes Broducktepolatory prodilepo;
Glassoulerd
public void nryfytring... angly brown Exception {
prodilepo, find Syrical Enwer(2000, 0, 2000, 0), ford auth [System.]
          -----
                  code in Repository
                       public List<Product> findByPriceBetween(double start, double end);
                                                                                                                                                                              )//run(-)
]//class
 MongoDB @Query method with Projections (Querles In MongoDB)
->By default @Query placed MongoDB app selects all the fleids of the Document class

⇒ Use "fields" attribute of @Query annotation having field name with 0 or 1 value
syntax:: @Query [fields" (property:0/1, property:0/1)") to select/deselect sp

⇒ 1 indicates involve the field/variable/property in the select query.

For all properties default value is 0, but for @id property the default value is 1.
                                                                                                                                                       ->Projections /scalar operations in Querying means selecting either specific single color multiple column values of db table/collection
                                                                                                                                                         ->Entity operation in querying means selecting all col
values of db table /collection
     =>"value" attribute of @Query is useful to specify the where condition clauses.

eg1: @Query( value="(cadd:?0)", cno is id property (@id)

fields="(cname:1,billAmr::1)") cno is id property (@id)
         is equal to "SELECT_CND,CNAME,BILLAMT FROM CUSTOMER WHERE CADD=?" (SQL)
                                                                                                                                                      use
Only for @Id property we can 0 or 1 in Projection opertaions for remaning only "1" is allowed (i.e involve the property) .. if u do not want to involve the property do not take "0" .. just
                                 (By Default @Id property will be seleted becoz its default value is 1)
    eg2: @Query( value="{cadd:?0}",
fields="{cno:0,cname:1,billAmt:1}")
                                                                                                                                                      ignore to place that propery.
                                                                                                                                                                  @\ Query(fields = "\{id:0,cno:1,cname:1,cadd:1,billAmt:1\}"\ ,value = "\{cadd:?0\}"\}
         Is equal to "SELECT CNAME, BILLAMT FROM CUSTOMER WHERE CADD-?" (SQL)
                                                                                                                                                                                                                valid
                                                                                                                                                           @Query(fields = "{id:0,cno:0,cname:1,cadd:1,billAmt:1}",value = "{cadd:?0}")
invalid
      eg3:: To get all fileds/property values
                @Query(value="{cadd:?0}",
fleids="{ )")
                                                                   (special case)
                                                                                                                                         ⇒in MongoDB Queries params position starts with 0 where as in IPQL/HQL the params position starts with 1 ⇒in fisleds attribute of gQuery 0 or 1 is allowed for id prosprty and remaining popertoes only 1 is allowed.
                @Query(value="{cadd:70}")
                is equal to "SELECT * FROM CUSTOMER WHERE CADD=?" (SQL)
                                                                                                                                       => In value attribute @Query we can 7 cn > with any number like 70,71,72,73 and etc..
       eg4:: To use multiple fields in where clause
                @Query(value="{cadd:70,cname:71}")
                   is equal to "SELECT" FROM CUSTOMER WHERE CADD=? AND CNAME=?" (SQL)
                                                                                                                                         notes: In MongoDB, we do not have query language like HQL/IPQL but the bring that effect we take the support of fields and value params of the @Query annotation
                  oublic class Employee (
                     //Repository Interface
                    package com.nt.repository;
                    Import Java.utiLUst;
                    Import org.springframework.data.mongodb.repository.Mongoitepository;
Import org.springframework.data.mongodb.repository.Query;
                    Import com.nt.document.Employee:
                    public interface (EmployeeRepo extends MongoRepositorysEmployee, Integer> [
                            &*Guery(value - "feacht?O,ename:?1)"] //where gadd-? and ename.? public ListsEmployees getEmpAliDataByAddrsAndName(String addrs,String name);
                                     // 6/Query(value = "(salary:(Spige20,Slige21))")
@Query(value = "(salary:(Spige20,Slige21))") // where salary>=? and salary>=?
public Tist-Employaec gerEmpAllDataBySalaryRanage(double startSalary,double entSalary);
                                                                                                                                                                                 The queries in MongoDB does not allow named params i.e we must take ordinal Positional Params
                                                                                                                                                                                       -> tradtional positional params :: ?
                                     (PQuery(value = "[$or:[{eadd:?0},{eadd:?1}]]"] //where eadd=? or eadd=? public 1ist=Employeeo gerEmpAlDataAddresses(String addrs1,5tring addrs1,5tring addrs1,5tring addrs1);
                                                                                                                                                                                      -> ordinal positional params :: ?<n>
-> named params :: :<name>
                                     //@Query(value="(ename;|Sregex': 70, 'Soptions': '?})") | / Y for case insensitivity is applied 
@Query(value="[ename;|Sregex': 70]]"] | //where ename like [K_36] is applied] 
public List<Imployee> getImpAlDataByEnameInitialChars(String initialChars);
                                     https://cheatography.com/davechild/cheat-sheets/regular-express
( For more info on regular expressions)
                                                                                                                                                                package com.nt.service
                   service Interface
                   package com.nt.service;
                                                                                                                                                               Import com.nt.document.Employee;
import com.nt.repository.IEmployeeRepo;
                   Import com.nt.document.Employee;
                                                                                                                                                                  9Scrvice("ampScrvice")
aublic class EmployeeMgmtServiceImpl implements (EmployeeMgmtService (
                   public interface (LimployeeMgmtService (
                   public List-Object[]> shnorfingDataRyWaidns(String addrs);
public List-Implayees shnowtmodDataByWaidns(String addrs);
public List-Replayees showtmodDataByWaidna(String addrs);
public List-Replayees shnowtmodDataByWaidna(MarkerString addrs, String name);
public List-Replayees shnowtmodDataByWaidna(String addrs), String addrs);
public List-Replayees shnowtmodDataByWaidna(String addrs), String addrs);
public List-Replayees shnowtmodDataByWaidna(String addrs), String addrs);
                                                                                                                                                                        private iEmployeeReno empReno:
                                                                                                                                                                        public ListsObject||> showEmpDataByAddrs(String addrs) (
                                                                                                                                                                                return empitepo.getLmpDataByAddrs(addrs);
                                                                                                                                                                        public List<Employee> showEmpAliDataByAddrs(String addrs) {
                                                                                                                                                                                return empllepo.getLmpAllDataByAddrs(addrs);
                                                                                                                                                                        public List<Employee> showEmpAliDataByAddrsAndName(String addrs, String name) {
                                                                                                                                                                                return empllepo.getEmpAllDataByAddrsAndName(addrs, name):
                                                                                                                                                                        @Overnide
public List<Employee> showEmpAliDataBySalaryRange(double start, double end) {
```

 $return\ empitepo.getLmpAliDataBySalaryRanage(start, end);$ 

**Custom methods user-defined MongoRepository** a) using finder/findBy methods (same as spring data jpa) b) using @Query methods (best) **Example on finder method** ======= code in Repository public List<Product> findByPrice Between (double start, double end); MongoDB @Query method with **Projections (Queries in MongoDB)** =>By default @Query placed MongoDB app selects all the fields of the Document class => Use "fields" attribute of @Query annotation having field name with 0 or 1 value syntax:: @Query( fields="{property:0/1, property:0/1}") to select /deselect specific field => 1 indicates involve the field/variable/property in the select query. => 0 indicates do not involve the field/variable/property in the select query. For all properties default value is 0, but for @ld property the default value is 1. =>"value" attribute of @Query is useful to specify the where condition clauses.. ordinal positional param who index starts with 0 eg1: @Query(value="{cadd:?0}"," cno is id property (@ld) fields="{cname:1,billAmt:1}") is equal to "SELECT CNO, CNAME, BILLAMT FROM CUSTOMER WHERE CADD=?" (SQL) (By Default @ld property will be seleted becoz its default value is 1) eg2: @Query(value="{cadd:?0}", fields="{cno:0,cname:1,billAmt:1}") is equal to "SELECT CNAME, BILLAMT FROM CUSTOMER WHERE CADD=?" (SQL) eg3:: To get all fileds/property values @Query(value="{cadd:?0}", fields="{}") @Query(value="{cadd:?0}") (special case) is equal to "SELECT \* FROM CUSTOMER WHERE CADD=?" (SQL) eg4:: To use multiple fields in where clause.. @Query(value="{cadd:?0,cname:?1}") note:: finder methods in spring boot data mongoDB are similar to the finder methods of spring boot data jpa Runner class code

@Component

 $public\ class\ MongoRepositoryFinderAndQueryMethodsRunner\ implements\ CommandLineRunner\ \{\ @Autowired\ Autowired\ Aut$ 

```
}//class
private IProductRepository prodRepo;
@Override
public void run(String... args) throws Exception {
prodRepo.findByPrice Between(1000.0, 20000.0).forEach(System.out::println);
}//run(-)
=>Projections /scalar operations in Querying means selecting either specific single col or multiple column
values. of db table /collection
=>Entity operation in querying means selecting all col values of db table /collection
use
Only for @ld property we can 0 or 1 in Projection opertaions for remaning only "1" is allowed (i.e involve the
property) .. if u do not want to involve the property do not take "O" .. just ignore to place that propery.
@Query(fields = "{id:0,cno:1,cname:1,cadd:1,billAmt:1}",value = "{cadd:?0}")
@Query(fields = "{id:0,cno:0,cname:1,cadd:1,billAmt:1}",value = "{cadd:?0}")
invalid
=>In MongoDB Queries params position starts with 0
where as in JPQL/HQL the params position starts with 1
=> In fileds attribute of @Query 0 or 1 is allowed for id proeprty and remaining popertoes only 1 is allowed
use
=> In value attribute @Query we can ?<n> with any number like ?0,?1,?2,?3 and etc..
is equal to "SELECT * FROM CUSTOMER WHERE CADD=? AND CNAME=?" (SQL)
//Document class
@Document(collection = "Employee_Info")
public class Employee {
private Integer eno;
private String ename;
private String eadd;
private Double salary;
private Boolean isVaccinated;
}
//Repository Interface
note:: In MongoDB, we do not have query language like HQL/JPQL, but the bring that effect we take the support of
fields and value params of the @Query annotation
```

package com.nt.repository;

```
import org.springframework.data.mongodb.repository.MongoRepository; import
org.springframework.data.mongodb.repository.Query;
import com.nt.document.Employee;
public interface IEmployeeRepo extends MongoRepository<Employee, Integer> {
//====== Projection Query ======== //@Query(fields="{eno:0,eadd:1,salary:1}",value = "{eadd:?0}")
//where eadd=? @Query(fields="{ename:1,eadd:1,salary:1}",value = "{eadd:?0}") // where eadd=? public
List<Object[]> getEmpDataByAddrs (String addrs);
//=====Entity Queries =====
//@Query(fields="{}",value = "{eadd:?0}")
@Query(value = "{eadd:?0}") //where eadd=?
public List<Employee> getEmpAllDataByAddrs(String addrs);
@Query(value = "{eadd:?0,ename:?1}") //where eadd=? and ename=?
public List<Employee> getEmpAllDataByAddrsAndName(String addrs, String name);
//@Query(value = "{salary:{$gte:?0,$lte:?1}}")
@Query(value = "{salary:{$gte:?0},salary:{$lte:?1}}") //where salary>=? and salary<=? public List<Employee>
getEmpAllDataBySalary Ranage(double startSalary,double endSalary);
@Query(value = "{$or:[{eadd:?0},{eadd:?1}]}") //where eadd=? or eadd=? public List<Employee>
getEmpAllDataAddresses(String addrs1,String addrs2); //@Query(value="{ename:{"$regex': ?0, '$options': 'i'}}") // 'i'
for case-insensityity is applied @Query(value="{ename:{"$regex': ?0}}") //where ename like (%_% is applied) public
List<Employee> getEmpAllDataByEnameInitialChars(String initialChars);
https://cheatography.com/davechild/cheat-sheets/regular-expressions/ (For more info on regular
expressions)
The queries in MongoDB does not allow named params
i.e we must take ordinal Positional Params
-> tradtional positional params :: ?
-> ordinal positional params :: ?<n> -> named params
:<name>
service Interface
//IEmployeeMgmtService.java package com.nt.service;
import java.util.List;
import com.nt.document. Employee;
public interface IEmployeeMgmtService {
public List<Object[]> showEmpDataByAddrs(String addrs); public List<Employee> showEmpAllDataByAddrs(String
addrs);
public List<Employee> showEmpAllDataByAddrsAndName(String addrs, String name); public List<Employee>
showEmpAllDataBySalary Range(double start, double end); public List<Employee>
```

import java.util.List;

```
showEmpAllDataByAddresses(String addrs1,String addrs2); public List<Employee> show
EmpAllDataByEnameInitialChars(String initialChars);
//EmployeeMgmtServiceImpl.java package com.nt.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.nt.document.Employee;
import com.nt.repository.IEmployeeRepo;
@Service("empService")
public class EmployeeMgmtServiceImpl implements IEmployeeMgmtService {
@Autowired
private IEmployeeRepo empRepo;
@Override
public List<Object[]> show EmpDataByAddrs(String addrs) {
return empRepo.getEmpDataByAddrs(addrs);
@Override
//Rinnner class
//QueryMethodsTestRunner.java
package com.nt.runners;
import java.util.Optional;
import java.util.Random;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import com.nt.document.Employee;
import com.nt.service.lEmployeeMgmtService;
@Component
public class QueryMethodsTestRunner implements CommandLineRunner {
@Autowired
private IEmployeeMgmtService service;
@Override
public void run(String... args) throws Exception {
/*service.showEmpDataByAddrs("hyd").forEach(row->{
for(Object val:row) {
}
```

```
System.out.print(val+" ");
}
System.out.println();
});*/
System.out.println("======
System.out.println("====
//service.showEmpAllDataByAddrs("hyd").forEach(System.out::println);
//service.showEmpAllDataByAddrsAndName("hyd", "rajesh").forEach(System.out::println);
====");
:=====");
======");
//service.showEmpAllDataBySalaryRange(80000,200000).forEach(System.out::println);
System.out.println("====
//service.showEmpAllDataByAddresses("hyd", "delhi").forEach(System.out::println);
System.out.println("=======
service.showEmpAllDataByEnameInitialChars("r").forEach(System.out::println);
public List<Employee> show EmpAllDataByAddrs(String addrs) {
return empRepo.getEmpAllDataByAddrs(addrs);
@Override
public List<Employee> showEmpAllDataByAddrsAndName(String addrs, String name) {
return empRepo.getEmpAllDataByAddrsAndName(addrs, name);
@Override
public List<Employee> showEmpAllDataBySalaryRange(double start, double end) {
return empRepo.getEmpAllDataBySalary Ranage(start, end);
@Override
public List<Employee> showEmpAllDataByAddresses(String addrs1, String addrs2) {
return empRepo.getEmpAllDataAddresses(addrs1,addrs2);
}
@Override
public List<Employee> show EmpAllDataByEnameInitialChars(String initialChars) {
return empRepo.getEmpAllDataByEnameInitialChars(initialChars);
}//class
```

For more MongoDB queries refer this :: https://www.mongodb.com/docs/manual/tutorial/query-documents/