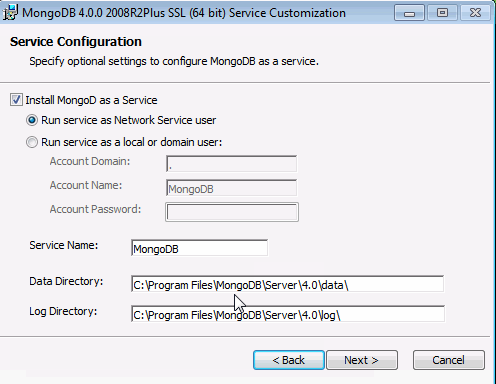
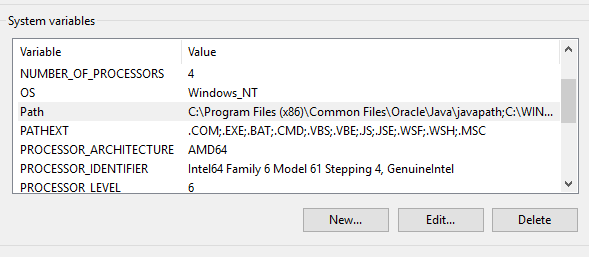
1. **MongoDB**
2. **Cài đặt**

* Java JDK: 1.8
* MongoDB: 4.2.8 - <https://www.mongodb.com/try/download/community> (file .msi)
* Chọn **Complete setup**



1. **Cấu hình**

* Thêm đường dẫn vào System variables:

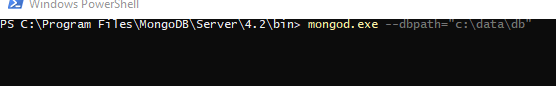




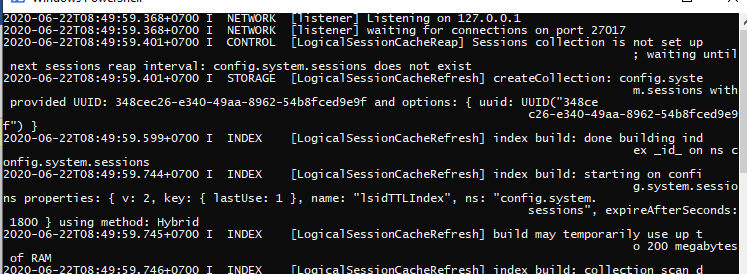
* Tạo thư mục chứa database: **C:\data\db**

1. **Chạy MongoDB**

* Vào thư mục /bin nơi cài đặt MongoDB và chạy lệnh:



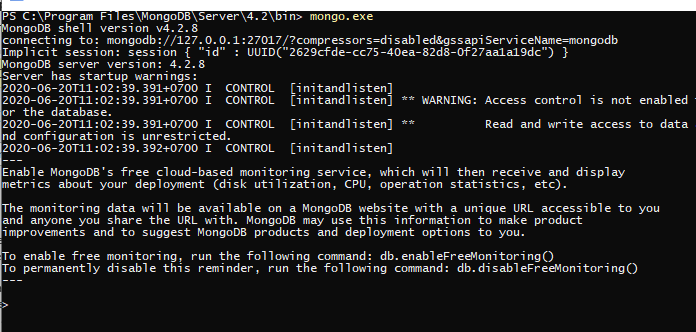
* --dbpath: là đường dẫn đến database.



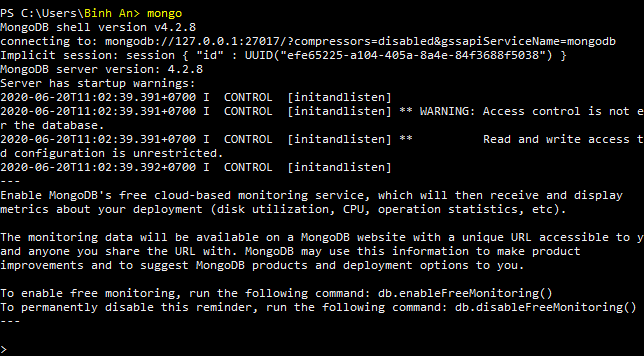
* Nếu chạy thành công sẽ có dòng **waiting for connections on port 27017.**
* Ta có thể chạy MongoDB dưới dạng một service
* Chạy Commandline quyền admin ở đường dẫn **<thư mục cài đặt>\bin.**
* Chạy lệnh **mongod –remove.** Lệnh này sẽ tắt mongodb đang chạy (nếu có).
* Sau đó chay: **mongod --dbpath="c:\data\db" --logpath="E:\MongoDB\log\log.txt" –install.** Lệnh này sẽ chạy mongodb dưới dạng service:
* **--dbpath** là đường dẫn đến database.
* **--logpath** là đường dẫn đến file log.
* Lưu ý: --logpath bắt buộc phải có nếu không sẽ không chạy được.



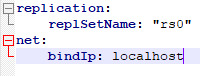
* Sau đó mở service.msc và start service MongoDB.
* Kết nối tới MongoDB bằng **mongo shell**
* Mở một PowerShell mới và chạy file **mongo.exe** tại thư mục **\bin**.



* Hoặc chạy lệnh **mongo** ở bất kỳ đâu bằng PowerShell.



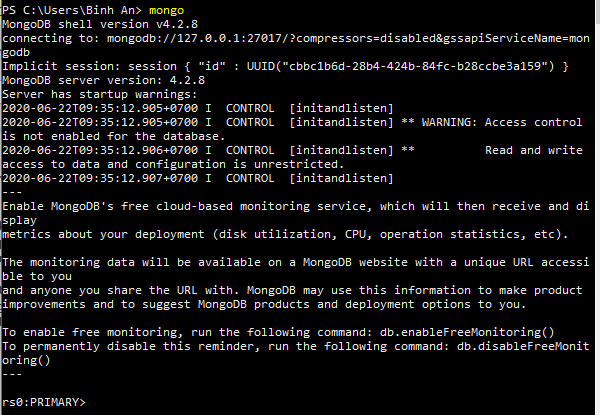
* Mặc định MongoDB sẽ chạy ở ip **127.0.0.1:27017**.
* Replica Set
* Replica Set là tập hợp các process **mongod** để duy trì cùng một database. Replica Set cung cấp khả năng dự phòng và tính sẵn sàng cao.
* Replicaton cung cấp tính dự phòng và khả dụng trên mongodb nhờ lưu trữ dữ liệu ở nhiều database server khác nhau, từ đó tăng mức độ chịu lỗi và chống việc mất data nếu có một server nào đó gặp sự cố. Nhờ vậy có thể sử dụng trong các mục đích như khôi phục, backup data.
* Cấu hình Replica Set
* Tạo thư mục config **C:\MongoDB\config** (thư mục này có thể ở bất kỳ đâu).
* Tạo file **config.yml** và thêm cấu hình:



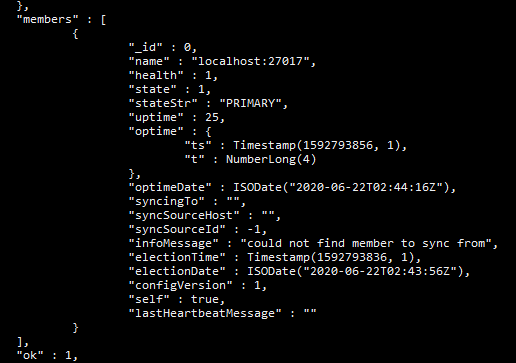
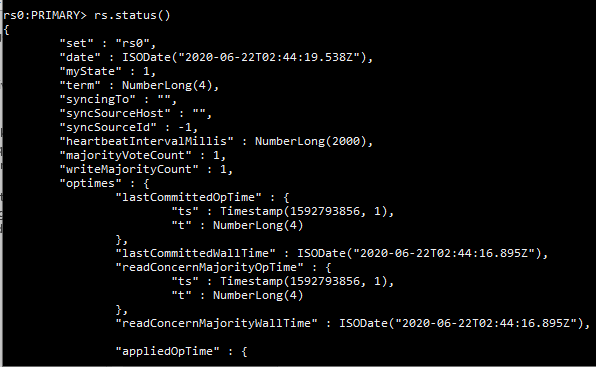
* Khởi động lại MongoDB database và bổ sung thêm vào câu lệnh:



* Chạy lại **mongo shell**.



* Gõ lệnh **rs.status()** để kiểm tra trạng thái của replica.



1. **Thao tác**
   1. **Database**
   * **Tạo database**

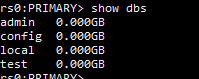
* Sử dụng lệnh **use <databasename>:**



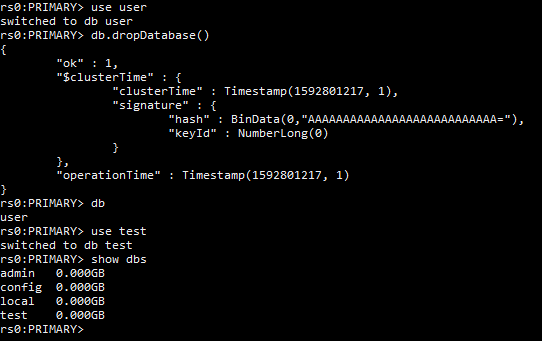
* Kiểm tra database hiện tại bằng lệnh **db:**



* Xem danh sách các database hiện có bằng lệnh **show dbs:**



* Database vừa tạo (user) sẽ không xuất hiện trong danh sách trừ khi thêm vào ít nhất một document.
* Trong trường hợp chưa tạo database thì mặc định sẽ là database **test.**
  + **Drop databse**
* Để drop database sử dụng lệnh **db.dropDatabase()**.
* Trong trường hợp không chọn database nào thì mặc định sẽ xóa database **test.**
* Để xóa database vừa tạo (**user**):



* 1. **Collection**
  + **Tạo Collection**
* Collection là tâp hợp các document, có thể hiểu Collection giống với table trong RDMS.
* Có 2 cách tạo Collection:
* Dùng câu lệnh **db.<collectionName>.insertOne({name: “John Doe”})**.
* Dùng câu lệnh **db.createCollection(<collectionName>, <options>)**.
* Cách 1, mongoDb sẽ tự động tạo collection khi insert document.
* Cách 2, mongoDB sẽ cung cấp nhiều lựa chọn hơn để quản lý collection. Xem thêm về các option tại: <https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection>
  + **Drop Collection**
* Gõ câu lệnh **db.<collectionName>.drop()**
* Kiểu dữ liệu
* MongoDB hỗ trợ nhiều kiểu dữ liệu như String, Integer, Double, ObjectID,…
* Tham khảo thêm về kiểu dữ liệu tai: <https://www.tutorialspoint.com/mongodb/mongodb_datatype.htm>
  + **Document validation**
* Mặc định các document trong cùng một collection không bất buộc phải giống nhau, các document có thể chứa các thuôc tính khác nhau, kiểu dữ liệu của các thuộc tính có thể khác nhau.
* Tuy nhiên người dùng vẫn có thể đặt các validation rule cho các document của collection(\* Chỉ áp dụng từ phiên bản **3.2** trở lên).
* Tạo collection “**student”** và validate:
* Tạo db tên **mydb.**

db.createCollection("students", {

validator: {

$jsonSchema: {

bsonType: "object",

required: [ "name", "year", "major", "address" ],

properties: {

name: {

bsonType: "string",

description: "must be a string and is required"

},

year: {

bsonType: "int",

minimum: 2017,

maximum: 3017,

description: "must be an integer in [ 2017, 3017 ] and is required"

},

major: {

enum: [ "Math", "English", "Computer Science", "History", null ],

description: "can only be one of the enum values and is required"

},

gpa: {

bsonType: [ "double" ],

description: "must be a double if the field exists"

},

address: {

bsonType: "object",

required: [ "city" ],

properties: {

street: {

bsonType: "string",

description: "must be a string if the field exists"

},

city: {

bsonType: "string",

"description": "must be a string and is required"

}

}

}

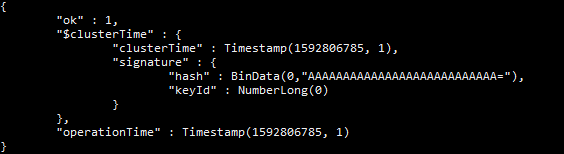
}

}

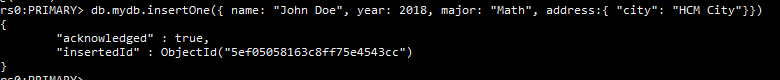
}

})

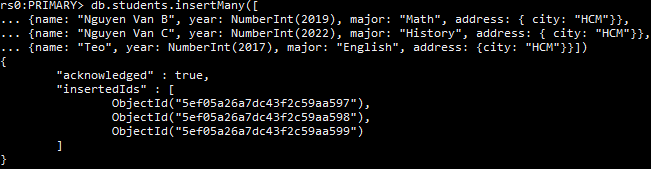




* 1. **CRUD**
  + **Insert()**



* Trong trường hợp không khai báo thuộc tính **\_id** thì MongoDB sẽ tự động thêm **\_id** vào và set giá trị mặc định cho document.
* Hàm insert() nhận vào một json document và các document có thể chứa bất kỳ data nào. Do vậy trong một collection, các document có thể chứa các thuộc tính, kiểu dũ liệu khác nhau mà không ảnh hưởng.
  + **InsertMany()**
* **db.<collectionName>.insert({…})**



* + **Update()**
* Tạo db mới tên **inventory**

**db.inventory.insertMany( [**

**{ item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },**

**{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },**

**{ item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },**

**{ item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },**

**{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },**

**{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },**

**{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },**

**{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },**

**{ item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },**

**{ item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }**

**] );**

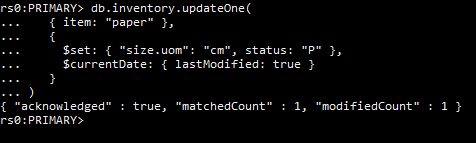
* Cấu trúc:

**db.<collectionName>.updateOne(**

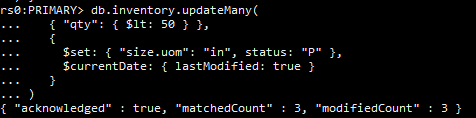
**<filter>, <update>, <options>**

**)**

* **filter:** để filter giá trị muốn cập nhật, tương tự như **find()**
* **update:** nội dung update.
* **options**: các lựa chọn khác, có thể có hoặc không. Tham khảo tại: <https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne>



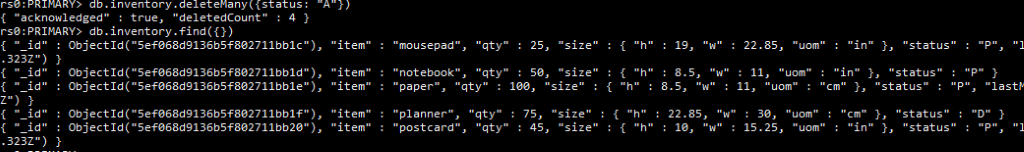
* **$set:** dùng để set giá trị mới cho thuộc tính của document.
* **$currentDate:** cập nhật giá trị **lastModified** của document, trong trường hợp không có thuộc tính **lastModified**, $currentDate sẽ tạo mới thuộc tính nay. Tìm hiểu thêm về $currentDate tại: <https://docs.mongodb.com/manual/reference/operator/update/currentDate/#up._S_currentDate>
  + **UpdateMany()**



* Update các document có **qty < 50**.
  + **Delete()**
* Xóa một document theo điều kiện



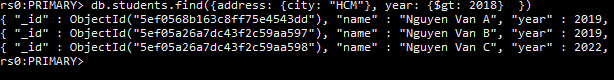
* + **DeleteMany()**
* Xóa nhiều document theo điều kiện



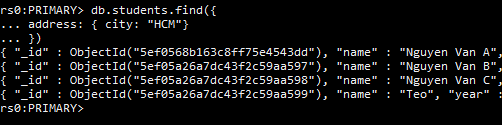
* + **Find()**
* Cú pháp: **db.<collectionName>.find({…})**
* Câu lệnh trên sẽ tìm tất cả record trong collection thỏa mãn điều kiện trong dấu {…}.



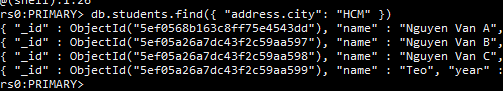
* Nếu truy vấn nhiều điều kiện thì cách nhau bởi dấu phẩy.



* Câu trên tìm các record có city = “HCM” và year > 2018.
* $gt: lớn hơn.
* $lt: nhỏ hơn.
* $or: thỏa mãn điều kiện 1 hoặc điều kiện 2.
* Tham khảo thêm tại: <https://docs.mongodb.com/manual/reference/operator/query/>
  + **Nested Query**
* Nếu document có nhiều đối tượng lồng vào nhau, có thể sử dụng 2 cách để truy vấn
* **Cách 1:**



* **Cách 2: Sử dụng dấu “.”**



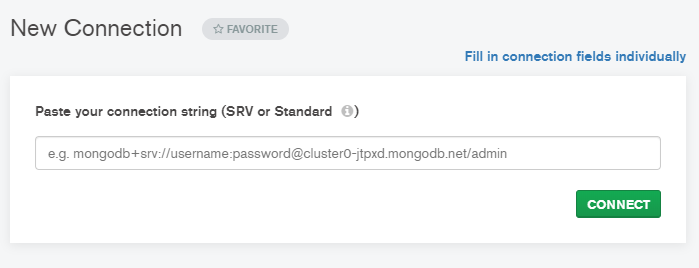
* Lúc này thuộc tính phải là một chuỗi nếu không sẽ lỗi



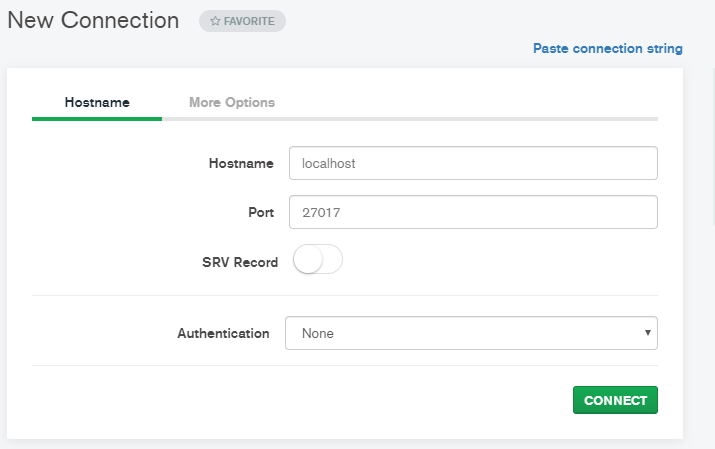
* + **Index**
* Mặc định MongoDB sẽ tạo một index là **\_id** khi tạo collection. **\_id** sẽ giúp tránh việc client insert 2 document có **\_id** giống nhau.
* Tìm hiểu thêm về **\_id** tại: <https://docs.mongodb.com/manual/core/document/#document-id-field>

1. **Spring Boot và MongoDb**
2. **Import data**

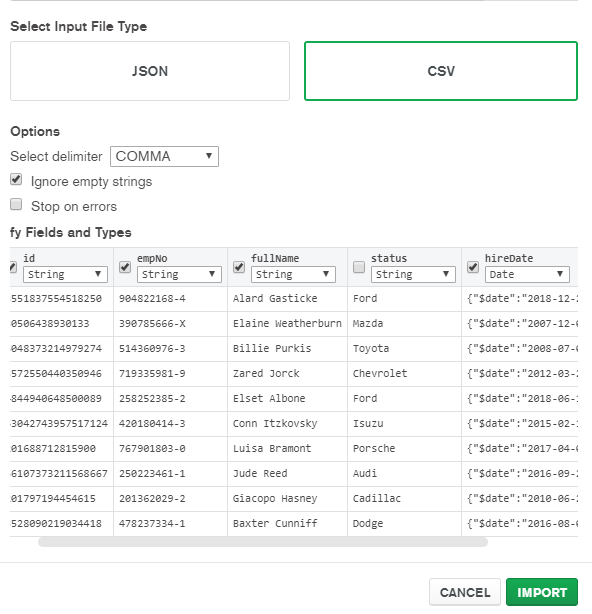
* Sử dụng Mongodb Compass để import data, mặc định khi cài đặt MongoDb sẽ có kèm Mongo Compass.
* Data file:
* <https://1drv.ms/u/s!Aj5XhBQkZ6QoiydX7FtOWHfAhIUe?e=bKBQSs>
* Tạo kết nối từ Compass đến MongoDB

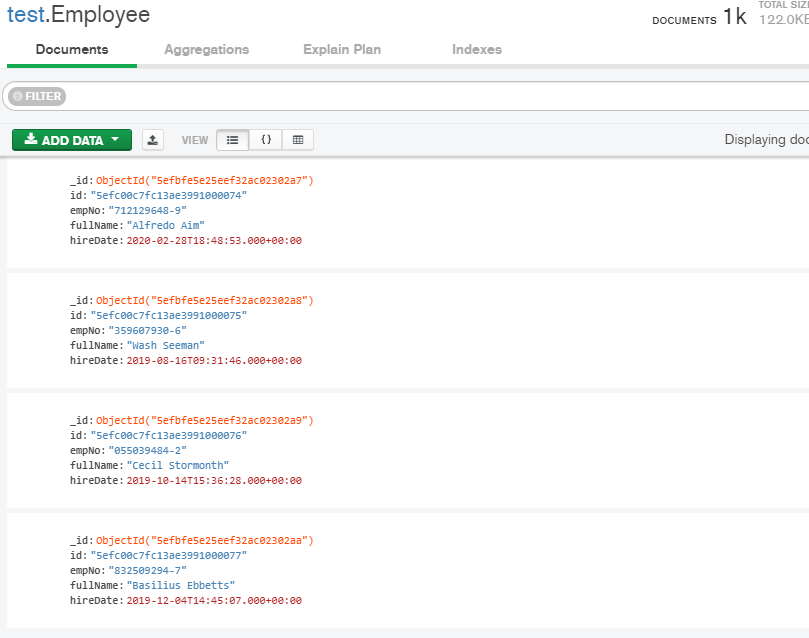


* Chọn fill connections



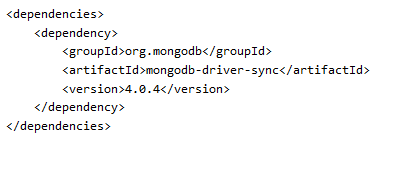
* Trong database test tạo Employee Collection
* Chọn ADD DATA -> Import File -> tìm đên file data





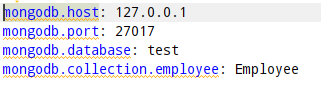
1. **Tạo ứng dụng Spring**

* Thêm dependency



1. **Kết nối ứng dụng với MongoDB**

* Thêm cấu hình vào **application.yml**



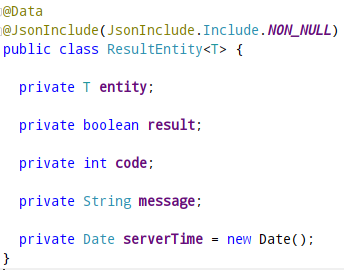
* Tạo folder **configs** và file **MongoConfigguration.java**



* **MongoConfiguration.java** có nhiệm vụ tạo kết nối tới MongoDB.

1. **Chuẩn bị project**

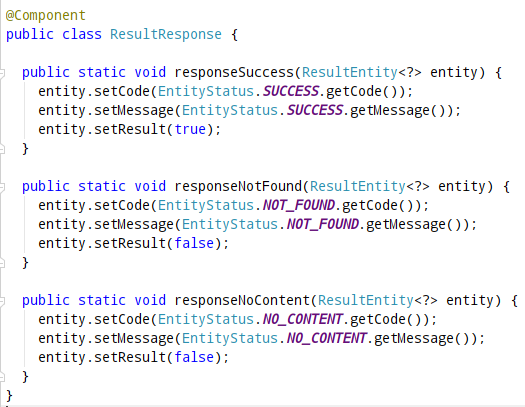
* Tạo **ResultEntity.java**, class này chỉ có nhiệm vụ làm cho response trả về trông clean hơn.



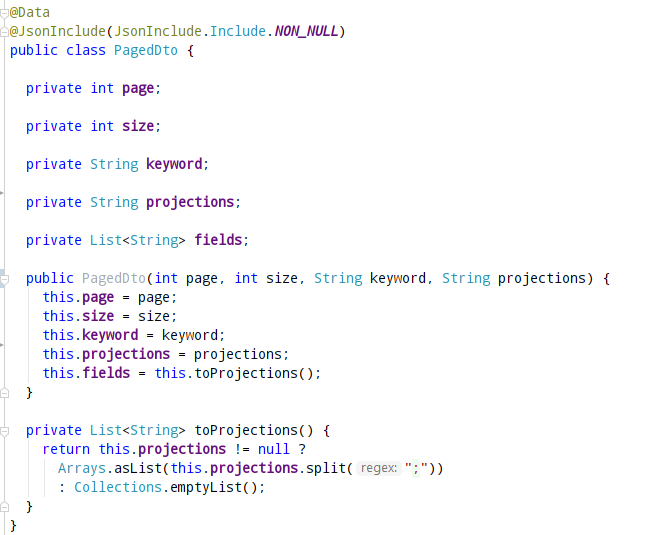
* Tạo Enum **EntityStatus.java**



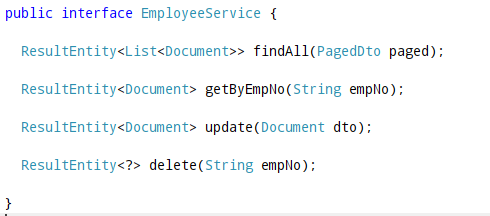
* **ResultResponse.java**



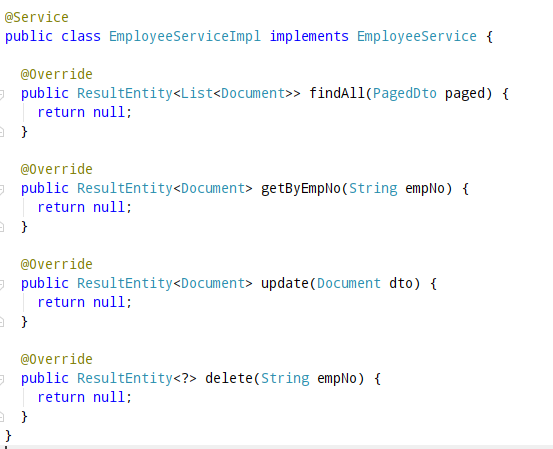
* Tạo clase **PagedDto.java** dùng để chứa các tham số khi lấy danh sách.



* Tạo interface **EmployeeService.java**



* Tạo **EmployeeServiceImpl.java** kế thừa lại interface EmployeeService.java

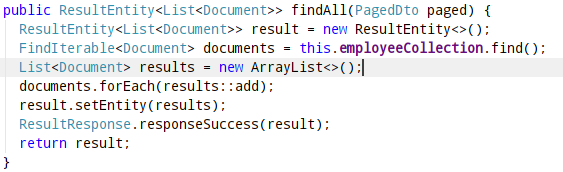


Sử dụng MongoClient để thử hiện truy vấn data từ MongoDB



* Hàm init() sẽ khởi tạo giá trị cho **employeeCollection**.

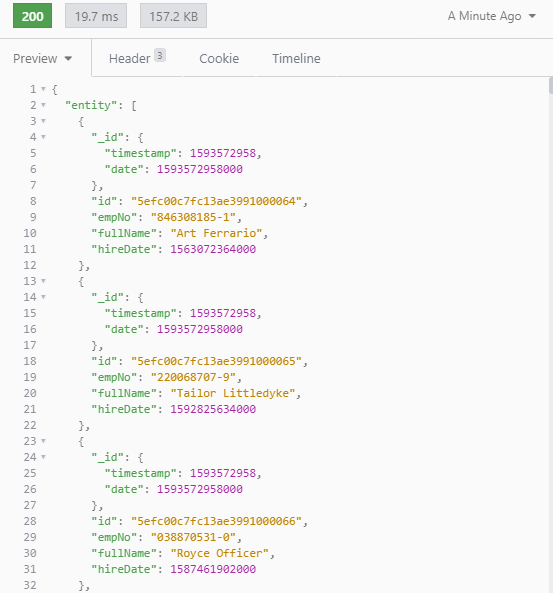
1. **Thao tác MongoDB qua ứng dụng**
   1. **Danh sách**



* Tạo **EmployeeController.java**



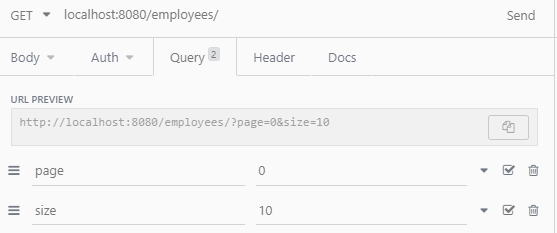
* Chạy ứng dụng, truy cập đường dẫn **localhost:8080/employees/**



* Chỉnh sửa hàm danh sách
* Tạo **CustomPaged.java** để thực hiện tìm kiếm, phân trang.

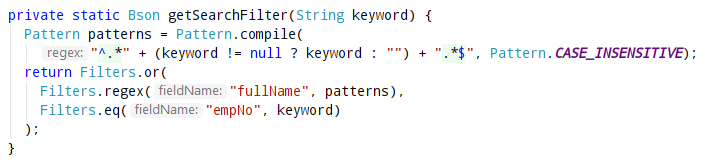


* **Phân trang**
* **skip()** sẽ bỏ qua bao nhiêu doument do người dùng nhập, **limit()** sẽ giới hạn số lượng document được lấy. Giả sử trang 1 lấy 10 giá trị thì sẽ lấy từ page = 0 và size = 10.
* Sử dụng Postman hoặc Insomnia để test.

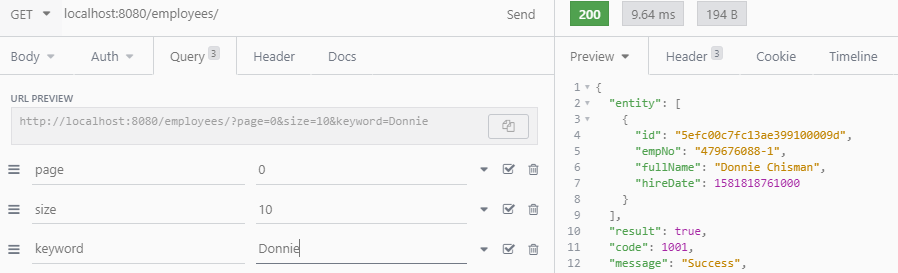




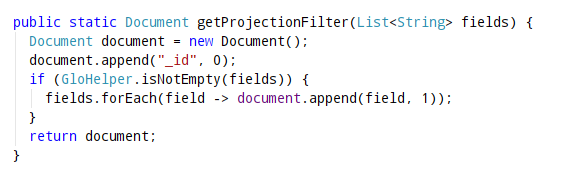
* **Tìm kiếm theo keyword**



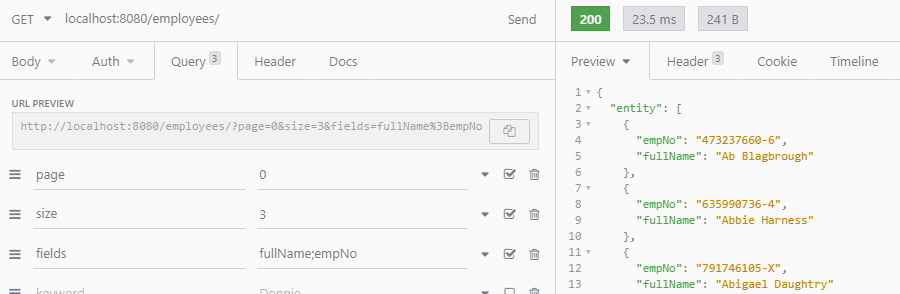
* Pattern sẽ tạo ra một pattern **^.\*keyword\*.$,** trong MongoDB đây là pattern dùng để tìm kiếm những giá trị gần giống với từ khóa. Tương tự như **LIKE** trong RDMS.
* Ta sẽ tìm kiếm theo tên hoặc mã của Employee.



* + **Projection**

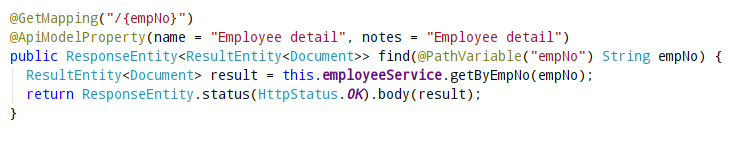


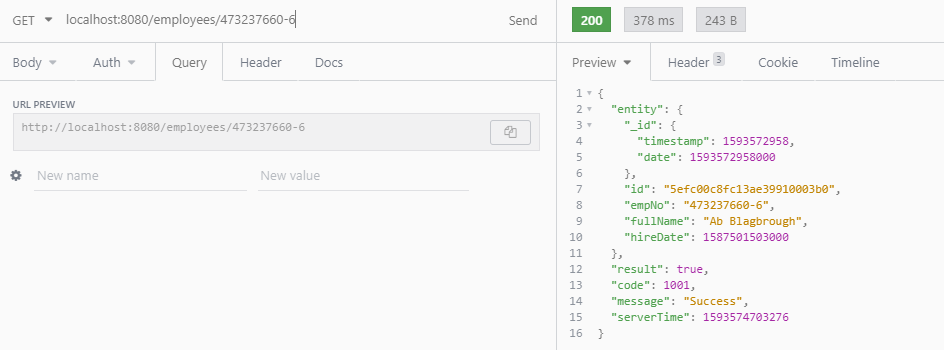
* Projection sẽ chỉ lấy những giá trị nào mà người dùng muốn.



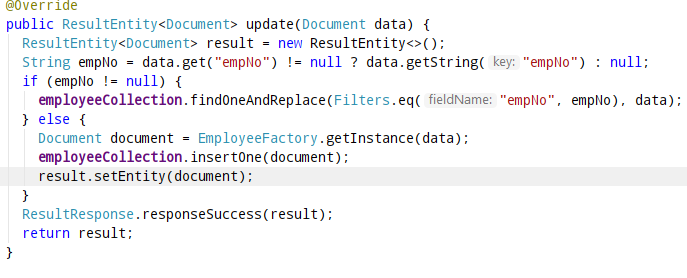
* 1. **Chi tiết**



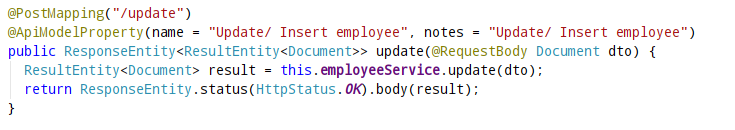




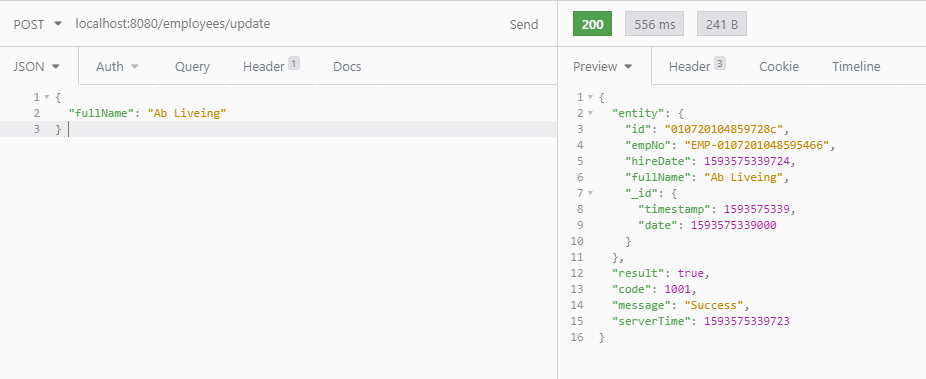
* 1. **Tạo/ cập nhật**
* Tạo **EmployeeFactory.java** để chứa hàm khởi tạo Employee**.**



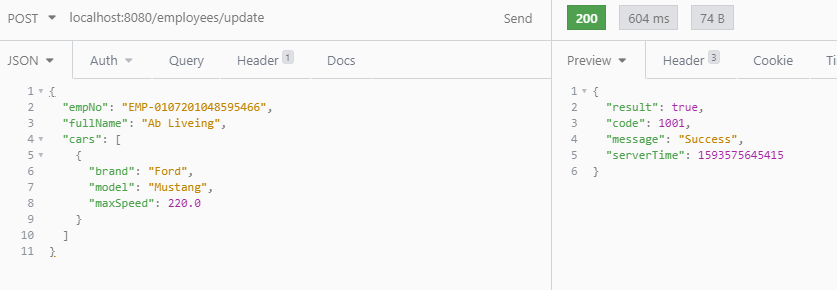
* Hàm trên cập nhật document nếu có empNo, ngược lại sẽ tạo mới.

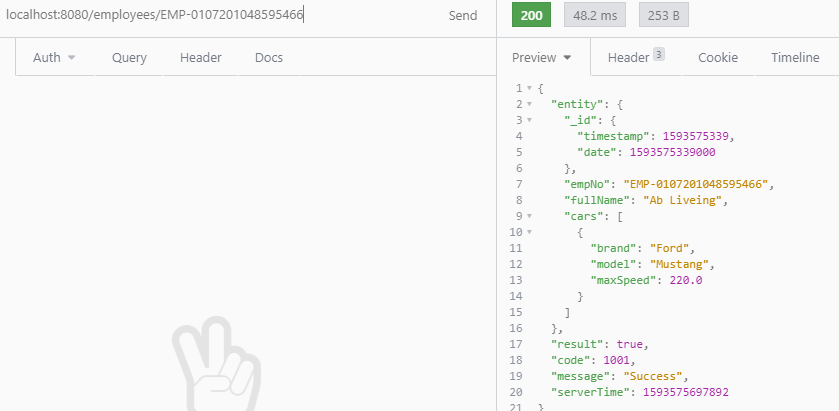


* Tạo mới

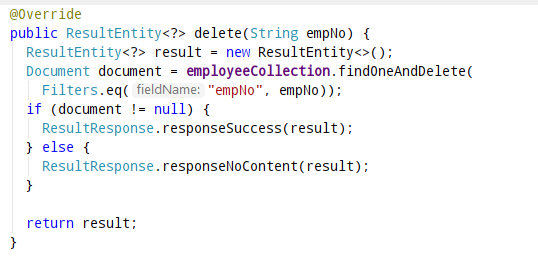


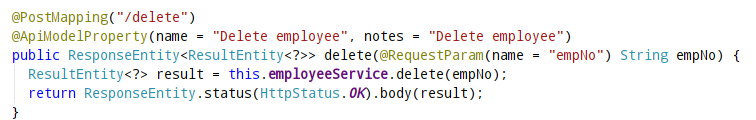
* Update

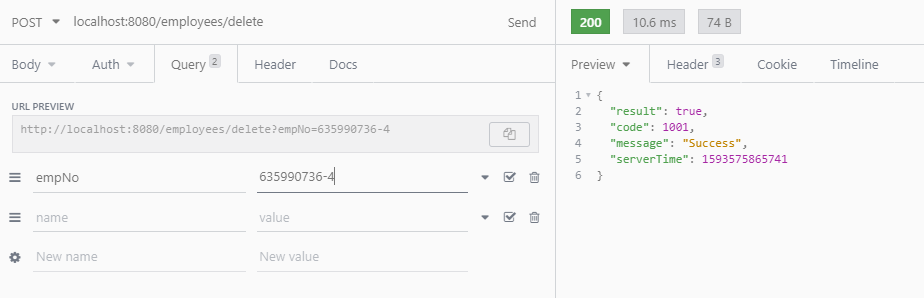


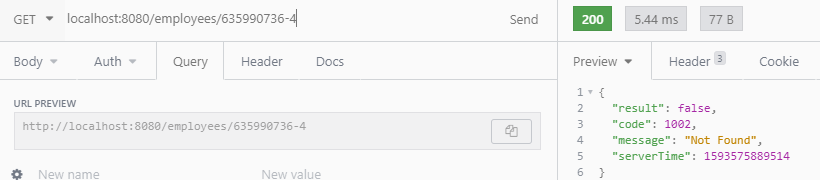


* 1. **Xóa**









* Ngoài ra còn có cách sử dụng POJO.
* Tham khảo thêm tại: <https://mongodb.github.io/mongo-java-driver/4.0/driver/getting-started/quick-start-pojo/>
* Source code: <https://BinhAn96@bitbucket.org/BinhAn96/spring-mongo-java-api.git>

Tài liệu tham khảo

MongoDB

<https://www.tutorialspoint.com/mongodb/index.htm>

<https://docs.mongodb.com/manual/crud/>

Mongo Java API

<https://mongodb.github.io/mongo-java-driver/4.0/apidocs/>