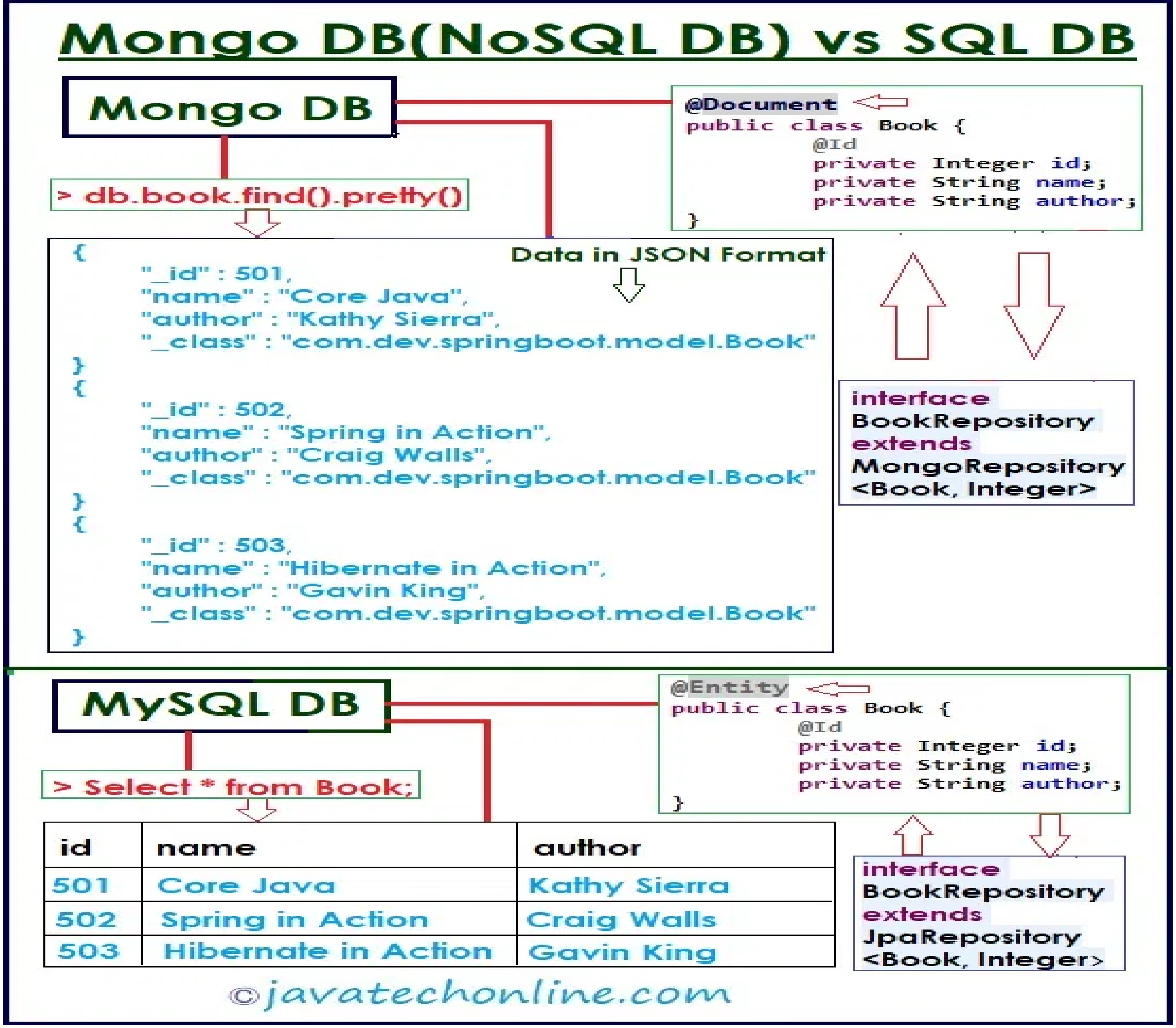
How To Work In Spring Boot With MongoDB?

As a Java developer, we can’t develop an insightful application without the use of a database software. Traditionally, we use a relational database to work with an application. Of course, A relational database is a structured database and contains multiple tables to maintain meaningful relations between them. Additionally, it uses SQL like queries to operate with data stored in the tables. In contrast, suppose we have to work with a large amount of unstructured data which is not beneficial to store in the form of tables, keeping other factors in mind as well, then how will we store our data? The simple answer is ‘we should use MongoDB’ in that case. Further, you might even have some other questions in your mind. Of course, our topic on ‘How to work in Spring Boot with MongoDB?’ Will answer all your questions.

In order to start working on Mongo DB confidently, you just need to go through this article step by step. Let’s start learning a new hot topic ‘How to work in Spring Boot with MongoDB?’ accordingly.



```
> db.book.find().pretty()
```

```
{
        "_id" : 501,
        "name" : "Core Java",
        "author" : "Kathy Sierra",
        "_class" : "com.dev.springboot.model.Book"
}
{
        "_id" : 502,
        "name" : "Spring in Action",
        "author" : "Craig Walls",
        "_class" : "com.dev.springboot.model.Book"
}
{
        "_id" : 503,
        "name" : "Hibernate in Action",
        "author" : "Gavin King",
        "_class" : "com.dev.springboot.model.Book"
}
```

```
@Document
public class Book {
        @Id
        private Integer id;
        private String name;
        private String author;
}
```

```
interface
BookRepository
extends
MongoRepository
<Book, Integer>
```

```
> Select * from Book;
```

id	name	author
501	Core Java	Kathy Sierra
502	Spring in Action	Craig Walls
503	Hibernate in Action	Gavin King

```
@Entity
public class Book {
        @Id
        private Integer id;
        private String name;
        private String author;
}
```

```
interface
BookRepository
extends
JpaRepository
<Book, Integer>
```

Table of Contents (Click on links below to navigate) [hide]

What will you learn from this Topic?

1) What is MongoDB ? Why do we need it ?
2) What is a NoSQL DB with examples of some popular NoSQL Databases?
3) Additionally, How many types of NoSQL databases are there with examples ?
4) Equally importantly, What are the benefits and drawbacks of using a NoSQL database?
5) In addition, How to install MongoDB on your system?

6) How to do basic operations in MongoDB collections like insert, retrieve, update, delete ?
7) Finally, How to use MongoDB in a Spring Boot Application with step by step process ?
8) Also, How to use annotation @Document and other annotations ?
9) Last but not the least you will learn 'How to work in Spring Boot with MongoDB?'

What is Mongo DB ?

Needless to say, 'What is Mongo DB' is equally important as 'How to work with MongoDB : A NoSQL Database?' is. Undoubtedly, Mongo DB is a NoSQL database. Moreover, it stores data in collection format. Each Collection stores data in document format. Additionally, Each document is similar to a JSON Object. To make it simple to understand, consider Collection as a table in SQL DB and document as a row. We also call MongoDB as a document-oriented database. It means you can store your records without worrying about the type of fields and number of fields. However, MongoDB doesn't support advanced analytics and joins like SQL DB. In summary, It just stores unstructured data in JSON format.

Every JSON document is identified by an ID(_id) which is in hexadecimal by default. This ID (primary key) is auto-generated by MongoDB only. Furthermore, We can even modify this ID. We retrieved below student data from MongoDB which is in JSON format as shown below. As shown in the first student data, auto-generated id is represented by ["_id" : ObjectId("5f8ff72f5b175156a2ba3c3d")] whereas in remaining three students we have modified value of _id attribute. Additionally, one _class attribute will be autogenerated which represents the package name of the entity. **One object in Java is equivalent to one JSON document in MongoDB.** For example, below JSON format code represents data in Mongo DB.

Student data in Mongo DB

```
> db.student.find().pretty()

{
        "_id" : ObjectId("5f8ff72f5b175156a2ba3c3d"),
        "stdId" : 5001,
        "stdName" : "George",
        "stdFee" : 2740,
        "_class" : "com.dev.springboot.model.Student"
}
{
        "_id" : "A10102B54DEF",
        "stdId" : 5002,
        "stdName" : "Victoria",
        "stdFee" : 2795,
        "_class" : "com.dev.springboot.model.Student"
}
{
        "_id" : "A10102B54DE0",
        "stdId" : 5003,
        "stdName" : "Mary",
        "stdFee" : 2999,
        "_class" : "com.dev.springboot.model.Student"
}
{
        "_id" : "dc35e897456",
        "stdId" : 5004,
        "stdName" : "David",
        "stdFee" : 2910,
        "_class" : "com.dev.springboot.model.Student"
}
```

What is NoSQL DB ? What are some popular examples of NoSQL databases?

NoSQL databases are those who store data in a format other than relational tables. These databases can store relational data but they just store it differently than relational databases do. Some examples of popular NoSQL databases are **MongoDB**, CouchDB, CouchBase, Cassandra, HBase, Redis, Riak, Neo4J. **MongoDB**, CouchDB, CouchBase are **document-oriented NoSQL databases**, Redis and Riak are key-value stores, Cassandra and HBase are column family stores and Neo4J is a graph database.

SQL data models allow relations by joining tables. On the other hand, NoSQL data models allow related data to be nested within a single data structure. Some people in the industry say the term "NoSQL" stands for "non SQL" while others say it stands for "not only SQL."

How many types of NoSQL DB are there?

Key-Values Stores, Wide-column Stores, Document Databases, Graph Databases are the types of NoSQL Databases.

Key-Values Stores

The Key-Values Stores are simplest and easiest to implement. They internally use a hash table where there is a unique key and a pointer to a particular item of data. Key-value Stores are great for use cases where you need to store large amounts of data, but you don't need to perform complex queries to retrieve it. **Examples:** Tokyo Cabinet/Tyrant, Redis, DynanoDB, Voldemort, Oracle BDB, Amazon SimpleDB, Riak.

Wide-column Stores

They store data in tables, rows, and dynamic columns. Here each row doesn't need to have the same columns. Therefore, they are better in flexibility over relational databases. Also, these were created to store and process very large amounts of data distributed over many machines. Wide-column stores are suitable for storing Internet of Things data and user profile data. **Examples:** Cassandra, HBase

Document Databases

They are similar to key-value stores and inspired by Lotus Notes. Document Databases store data in documents like JSON objects. Further JSON can store values in a variety of types like Strings, Numbers, Booleans, Arrays or Objects. They also support powerful query languages. Hence, they can be used as a general purpose database. According to DB-engines ranking, MongoDB is consistently ranked as the world's most popular NoSQL database. **Examples: MongoDB,** CouchDB

Graph Databases

They use a flexible graph model which, again, can scale across multiple machines. Graph Databases store data in nodes and edges. Nodes typically store information about people, places, and things while edges store information about the relationships between the nodes. They are useful in use cases where you need to traverse relationships to look for patterns such as social networks, fraud detection, and recommendation engines. **Examples:** Neo4J, JanusGraph, InfoGrid, Infinite Graph

What are the benefits of using a NoSQL Database?

NoSQL databases are more flexible and scalable, faster, reliable, distributed, schema-free architecture, easy replication support, simple API, support for big data etc. NoSQL databases can often perform better than SQL(relational databases) for your use case. For example, if you're using a document database and are storing all the information about an object in the same document (so that it matches the objects in your code), the database only needs to go to one place for those queries. In a SQL database, the same query would likely involve joining multiple tables and records, which can dramatically impact performance while also slowing down how quickly developers write code.

What are the Drawbacks of a NoSQL Database?

One of the recognized drawbacks of NoSQL databases is that they don't support ACID (atomicity, consistency, isolation, durability) transactions across multiple documents. Single record atomicity is possible for lots of applications with appropriate schema design. However, there are still many applications that may demand ACID across multiple records. Further, MongoDB included support for multi-document ACID transactions in the 4.0 release to address this drawback. As NoSQL data models primarily focused on optimizing for queries rather than reducing data duplication, therefore NoSQL databases can be larger than SQL databases.

How to download & install MongoDB in your system?

In order to work with MongoDB, make sure that you have installed the Mongo DB in your system. If not, please visit our article on how to install MongoDB in your system. Here, you will find step by step tutorial to download & install MongoDB including some required commands to operate on it.

How to create collections, insert, retrieve, update & delete records in MongoDB ?

First Start MongoDB server then start MongoDB client by typing below commands on two separate command prompts respectively.
->Start Mongo Server : cmd > mongod
->Start Mongo Client : cmd > mongo

Now practice running below commands on MongoDB client window to get confidence on this.

1) View all existing Databases : **>** show dbs
2) View current current Database : **>** db
3) Create your own Database : **>** use <DATABASE_NAME>

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

example :

```
> use myTestDB
```

4) View all existing collections : **>** show collections

5) Create a collection and insert data into it

```
> db.<COLLECTION_NAME>.insert({key:value....})
```

example :

```
> db.employee.insert({"eid":137229, "ename":"Anderson", "esal" :96784})
```

6) Display all Documents(JSON) from a collection

```
> db.<COLLECTION_NAME>.find()
> db.<COLLECTION_NAME>.find().pretty() [pretty() method displays the results in a formatted way ]
```

example :

```
> db.employee.find({"ename":"Anderson"}).pretty()
```

7) Delete Document (JSON) from a Collection

```
> db.<COLLECTION_NAME>.deleteOne({k:v})  [deletes one record]
> db.<COLLECTION_NAME>.deleteMany({k:v}) [deletes multiple records]
```

example:

```
> db.employee.deleteOne({"ename":"Anderson"})
```

8) Update document from a collection

```
> db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
> db.employee.update({'ename':'Anderson'},{$set:{'ename':'George'}})
```

9) Drop a Collection

```
> db.<COLLECTION_NAME>.drop()
> db.employee.drop()
```

10) Drop a database

Dropping database is a 2 step process. First use a database then drop it, otherwise it will delete default 'test' database.

```
> use myTestDB
> db.dropDatabase()
```

How to work in Spring Boot with MongoDB?

Let's create an example of Spring Boot test application using MongoDB step by step.

What all Tools & Technologies used in this example project?

♦ STS (Spring Tool Suite) : Version-> 4.7.1.RELEASE
⇒ Dependent Starters : 'Spring Boot Mongodb', and 'Lombok'
♦ JDK8 or later versions (Extremely tested on JDK8, JDK11 and JDK14)
♥ MongoDB 3.6.20 2008R2Plus SSL (64 bit)

Create a Spring Boot Starter Project in STS(Spring Tool Suite)

While creating Starter Project select 'Spring Boot Mongodb', and 'Lombok' as starter project dependencies. Still if you don't know how to create Spring Boot Starter Project, Kindly visit Internal Link. Also, if you are interested to know more about Lombok, please visit internal link. Additionally to know how to install MongoDB use this internal link.

Updating application.properties

application.properties

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=springBootMongoDB
```

Creating an Entity class

We are taking 'Book.java' as an entity to test the application. Here, observe the @Document annotation instead of @Entity.

Book.java

```java
package com.dev.springboot.mongodb.entity;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@NoArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor
@Document            // Maps Entity class objects to JSON formatted Documents
public class Book {

        @Id              // making this variable as ID, will be auto-generated by Mo
        private String id;

        @NonNull
        private Integer bookId;
        @NonNull
        private String bookName;
        @NonNull
        private String bookAuthor;
        @NonNull
        private Double bookCost;
}
```

Creating Repository Interface

As per standard naming convention, Create interface BookRepository.java which will extend from MongoRepository<Book, Integer>as below example code.

BookRepository.java

```
package com.dev.springboot.mongodb.repo;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.dev.springboot.mongodb.entity.Book;

public interface BookRepository extends MongoRepository<Book, Integer> {

}
```

Creating Runner class

Create one runner class named as ‘BookTestRunner.java’ just to test the application once. It implements ‘CommandLineRunner.java’. In this example we are saving 4 books in MongoDB. In addition, updating ID of last book manually(which is allowed) and name of author. MongoDB will consider this as new record. At the last, retrieve all the saved books and display it in the console. For example, below is the code.

BookTestRunner.java

```
package com.dev.springboot.mongodb.runner;

import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import com.dev.springboot.mongodb.entity.Book;
import com.dev.springboot.mongodb.repo.BookRepository;

@Component
public class BookTestRunner implements CommandLineRunner {

        @Autowired
        private BookRepository bookRepo;

        @Override
        public void run(String... args) throws Exception {

                // Removing old data if exists
                bookRepo.deleteAll();

                // Saving 4 books into DB
                bookRepo.saveAll(Arrays.asList(
                                new Book(501, "Core Java", "Kathy Sierra", 1065.5),
                                new Book(502, "Spring in Action", "Craig Walls", 940
                                new Book(503, "Hibernate in Action", "Gavin King", 8
                                new Book(504, "Practical MongoDB", "Shakuntala Gupta
                ));
                System.out.println("All Data saved into MongoDB");

                // Updating ID(PK) manually (allowed) : It will create one new recor
                bookRepo.save(new Book("ISBN10:1484240251", 504,"Practical MongoDB",

                // Printing all books
                List<Book> bookList =  bookRepo.findAll();
                bookList.forEach(System.out::println);

        }

}
```

Running the Application

Right click on the project, then select "Run As' >> 'Spring Boot App'.

Output from STS

On running as Spring Boot App, you will see the below output.

Output from STS console

```
All Data saved into MongoDB
Book(id=5fac078513a1bf3485c961a2, bookId=501, bookName=Core Java, bookAuthor=Kathy S
Book(id=5fac078513a1bf3485c961a3, bookId=502, bookName=Spring in Action, bookAuthor=
Book(id=5fac078513a1bf3485c961a4, bookId=503, bookName=Hibernate in Action, bookAuth
Book(id=5fac078513a1bf3485c961a5, bookId=504, bookName=Practical MongoDB, bookAuthor
Book(id=ISBN10:1484240251, bookId=504, bookName=Practical MongoDB, bookAuthor=Navin
```

Output from MongoDB

Run below command from MongoDB client and observe the output as below
> db.book.find().pretty()

Output from MongoDB

```
{
        "_id" : ObjectId("5fabf2fa4008d81d97a94a83"),
        "bookId" : 501,
        "bookName" : "Core Java",
        "bookAuthor" : "Kathy Sierra",
        "bookCost" : 1065.5,
        "_class" : "com.dev.springboot.mongodb.entity.Book"
}
{
        "_id" : ObjectId("5fabf2fa4008d81d97a94a84"),
        "bookId" : 502,
        "bookName" : "Spring in Action",
        "bookAuthor" : "Craig Walls",
        "bookCost" : 940.75,
        "_class" : "com.dev.springboot.mongodb.entity.Book"
}
{
        "_id" : ObjectId("5fabf2fa4008d81d97a94a85"),
        "bookId" : 503,
        "bookName" : "Hibernate in Action",
        "bookAuthor" : "Gavin King",
        "bookCost" : 889.25,
        "_class" : "com.dev.springboot.mongodb.entity.Book"
}
{
        "_id" : ObjectId("5fabf2fa4008d81d97a94a86"),
        "bookId" : 504,
        "bookName" : "Practical MongoDB",
        "bookAuthor" : "Shakuntala Gupta",
        "bookCost" : 785,
        "_class" : "com.dev.springboot.mongodb.entity.Book"
}
{
        "_id" : "ISBN10:1484240251",
        "bookId" : 504,
        "bookName" : "Practical MongoDB",
        "bookAuthor" : "Navin Sabharwal",
        "bookCost" : 785,
        "_class" : "com.dev.springboot.mongodb.entity.Book"
}
```

Summary

Once you finish going through all the details of this topic ‘How to work in Spring Boot with MongoDB?’, Of course, you will feel comfortable in working with MongoDB. Although, this topic was a foundation to the MongoDB. Further, in upcoming topics of MongoDB, we will learn more complex part of it with examples. We will update this topic as well whenever required to do so. Also, please feel free to provide your comments below.