

# Advanced Distributed Systems

## Higher Institute for Applied Sciences and Technology

Student: ضياء حنا  
Teacher: محمد بشار دسوقي  
Date: 10/4/2026

# Springboot and mongodb integration

**April 10 2026**

This file is divided into tasks each task will talk about specific thing about this homework

### Task 1 (Creating maven basic project) (hello world project)

We can do that using this command

```
Fri 10 Apr - 14:54 > ~/UNI/adistr/MongoProj > | master 2*  
@drnull > mvn archetype:generate \  
-DgroupId=com.mongo \  
-DartifactId=SpringMongo \  
-DarchetypeArtifactId=maven-archetype-quickstart \  
-DinteractiveMode=false  
[INFO] Scanning for projects...
```

This will create a hello world project

```
[INFO] -----  
Fri 10 Apr - 14:58 > ~/UNI/adistr/MongoProj > | master 3*  
@drnull > ls  
↓ README.md  ➤ Report  ➤ SpringMongo  
  
Fri 10 Apr - 14:58 > ~/UNI/adistr/MongoProj > | master 3*  
@drnull > |
```

We navigate to SpringMongo

Then we compile the project

```
Fri 10 Apr - 15:02 ~/UNI/adistr/MongoProj/SpringMongo [master]
$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.mongo:SpringMongo >-----
[INFO] Building SpringMongo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
Downloading from central: https://repo.maven.apache.org/maven2/org/junit/jupiter/junit-jupiter-api/5.10.0/junit-jupiter-api-5.10.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/jupiter/junit-jupiter-api/5.10.0/junit-jupiter-api-5.10.0.pom (3.2 kB at 1.6 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/opentest4j/opentest4j/1.3.0/opentest4j-1.3.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/opentest4j/opentest4j/1.3.0/opentest4j-1.3.0.pom (2.0 kB at 2.6 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-commons/1.10.0/junit-platform-commons-1.10.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-commons/1.10.0/junit-platform-commons-1.10.0.pom (2.8 kB at 3.5 kB/s)
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ SpringMongo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/dnnull/UNI/adistr/MongoProj/SpringMongo/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:compile (default-compile) @ SpringMongo ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 1 source file with javac [debug target 17] to target/classes
[WARNING] system modules path not set in conjunction with -source 17
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 4.653 s
[INFO] Finished at: 2026-04-10T14:02:28+02:00
[INFO]
Fri 10 Apr - 15:02 ~/UNI/adistr/MongoProj/SpringMongo [master]
$ mvn package
```

Then package it

```

Fri 10 Apr - 15:04 ~/UNI/adistr/MongoProj/SpringMongo [root@springmongo]
$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.mongo:SpringMongo >-----
[INFO] Building SpringMongo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ SpringMongo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/drnul/UNI/adistr/MongoProj/SpringMongo/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:compile (default-compile) @ SpringMongo ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ SpringMongo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/drnul/UNI/adistr/MongoProj/SpringMongo/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:testCompile (default-testCompile) @ SpringMongo ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 1 source file with javac [debug target 17] to target/test-classes
[WARNING] system modules path not set in conjunction with -source 17
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ SpringMongo ---
[INFO] Surefire report directory: /home/drnul/UNI/adistr/MongoProj/SpringMongo/target/surefire-reports
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.pom (1.7 kB at 1.0 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-providers/2.12.4/surefire-providers-2.12.4.pom (2.3 kB at 2.9 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.jar (26 kB at 22 kB/s)
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
Running com.mongo.AppTest
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Results :
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ SpringMongo ---
[INFO] Building jar: /home/drnul/UNI/adistr/MongoProj/SpringMongo/target/SpringMongo-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 5.391 s
[INFO] Finished at: 2026-04-10T14:05:02:00
[INFO]
Fri 10 Apr - 15:05 ~/UNI/adistr/MongoProj/SpringMongo [root@springmongo]
$

```

Finally we execute it

```

Fri 10 Apr - 15:07 ~ /UNI/adistr/MongoProj/SpringMongo master 3+
@drnull java -cp target/SpringMongo-1.0-SNAPSHOT.jar com.mongo.App
Hello World!

Fri 10 Apr - 15:07 ~ /UNI/adistr/MongoProj/SpringMongo master 3+
@drnull

```

## Task 2(Integrating with mongodb and springboot)

Turning on monodb locally

```

x Fri 10 Apr - 15:09 ~ /UNI/adistr/MongoProj/SpringMongo master 3+
@drnull sudo systemctl start mongod

Fri 10 Apr - 15:09 ~ /UNI/adistr/MongoProj/SpringMongo master 3+
@drnull nmap localhost -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-04-10 15:09 +03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000074s latency).
Not shown: 65528 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
902/tcp   open  iss-realsecure
8125/tcp  open  unknown
11434/tcp open  unknown
19999/tcp open  dnp-sec
27017/tcp open  mongod
48021/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 1.10 seconds

Fri 10 Apr - 15:09 ~ /UNI/adistr/MongoProj/SpringMongo master 3+
@drnull

```

Port 27017 is open

```
Fri 10 Apr - 15:09 ~/UNI/adistr/MongoProj/SpringMongo } master 3*  
@drnull ss -ltnp | grep 27017  
LISTEN 0 4096 127.0.0.1:27017 0.0.0.0:*  
  
Fri 10 Apr - 15:10 ~/UNI/adistr/MongoProj/SpringMongo } master 3*  
@drnull |
```

and it is only listening on localhost interface

```
MongoshInvalidInputError: [COMMON-10001] 'do  
test> show databases;  
admin 40.00 KiB  
config 72.00 KiB  
local 72.00 KiB  
mydb 72.00 KiB  
test> █
```

We already have mydb from class

I am gonna create a new db (namespace called student\_db)

Creating the database

```

Fri 10 Apr - 15:15 ~ /UNI/adistr/MongoProj/SpringMongo / master 3*
@drnull> mongosh
Current Mongosh Log ID: 69d8ea71b67f1759fe44ba88
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelect
ionTimeoutMS=2000&appName=mongosh+2.8.2
Using MongoDB:      7.0.31
Using Mongosh:      2.8.2

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
The server generated these startup warnings when booting
2026-04-10T15:09:50.364+03:00: Using the XFS filesystem is strongly recommended wi
th the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2026-04-10T15:09:50.850+03:00: Access control is not enabled for the database. Rea
d and write access to data and configuration is unrestricted
-----

test> show collections;

test> show document;
MongoshInvalidInputError: [COMMON-10001] 'document' is not a valid argument for "show
".
test> show documents;
MongoshInvalidInputError: [COMMON-10001] 'documents' is not a valid argument for "sho
w".
test> show databases;
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
mydb     72.00 KiB
test> use student_db;
switched to db student_db
student_db> db.students.insertOne({name:"INIT Student",email:"test@test.com"})
{
  acknowledged: true,
  insertedId: ObjectId('69d8ebb4b67f1759fe44ba89')
}
student_db>

```

```
student_db> show dbs
admin          40.00 KiB
config         108.00 KiB
local          72.00 KiB
mydb           72.00 KiB
student_db     8.00 KiB
student_db> 
```

After some research using chatGPT

It turned out there is a thing called spring-data-mongo plugin

Which weird from what i know spring boot build web application

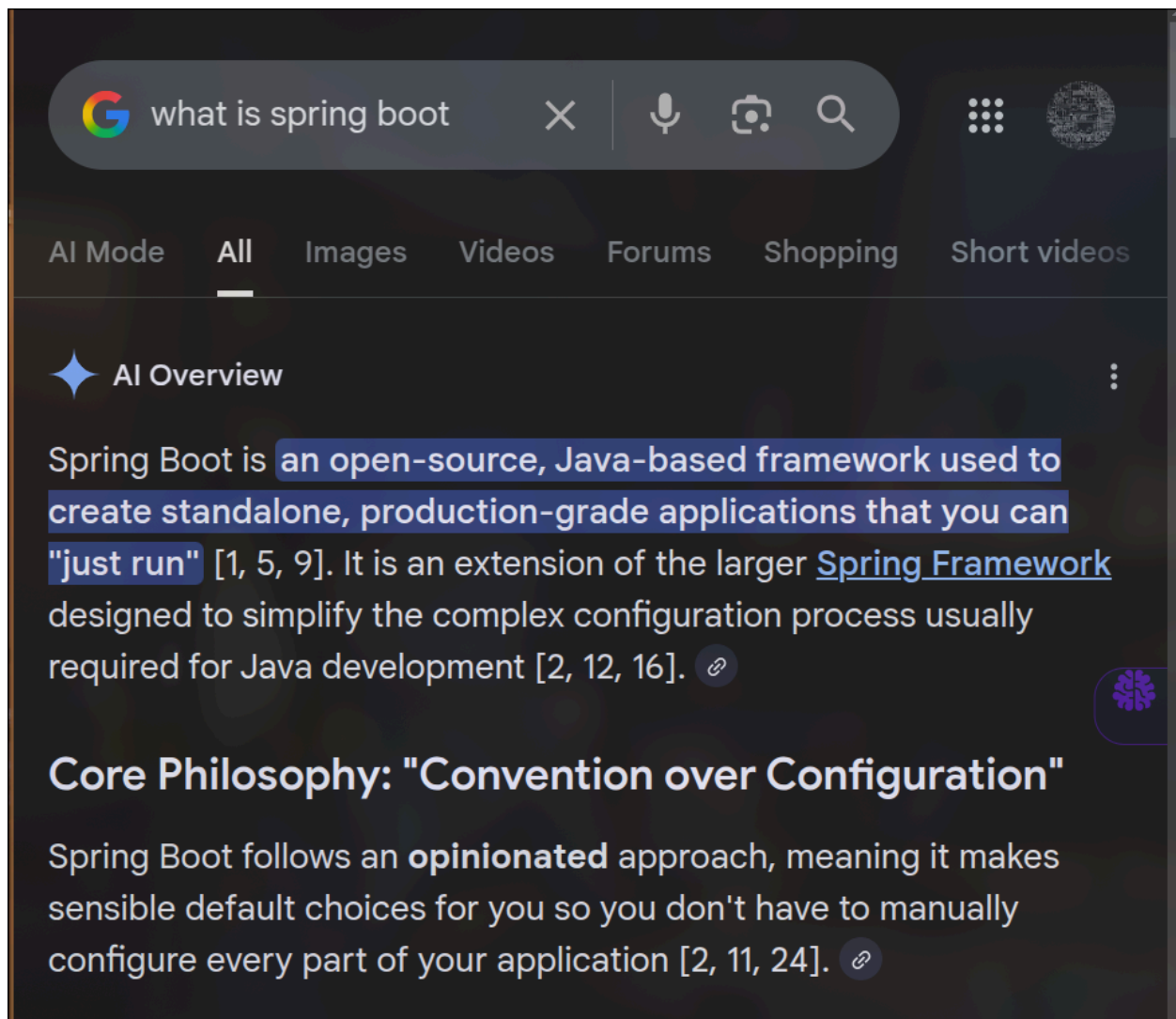
I was expecting downloading two different dependencies

One for mongo db integration and another one for

Springboot web app



Lets revise what springboot is



So it is not just for web applications it is an opinionated designed to ease the programmer from configuring application

We can even write command line applications with it.

And spring-data-mongo is like ORM for mongo db it maps java objects to documents which is exactly what orm does.

### Step 2.1 (editing the pom.xml file)

With the help of chatGPT we can learn how to add spring-data-mongo

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mongo</groupId>
```

```
<artifactId>SpringMongo</artifactId>
<version>1.0-SNAPSHOT</version>

<!-- 1. Add Spring Boot Parent -->
<parent>
    <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.0</version>
    <relativePath/>
</parent>

<properties>
    <java.version>17</java.version>
</properties>

<dependencies>
    <!-- 2. Spring Data MongoDB Starter -->
    <dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifa
```

```
ctId>
    </dependency>

    <dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <!-- 3. Spring Boot Build Plugin →

        <plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
</plugins>
```

```
</build>  
</project>
```

### Step 2.2 (adding the connection string)

This Whole Process is so similar of using .Net with entity framework (ORM)

From the documentation the connection string should be in

src/main/resources/application.properties

```
Fri 10 Apr - 15:42 ~/UNI/adistr/MongoProj/SpringMongo master 3*  
@drnull cat src/main/resources/application.properties  
# Connect to localhost on default port and use student_db  
spring.data.mongodb.uri=mongodb://localhost:27017/student_db
```

Notice that we add the db(namespace in this file).

### Step 2.3 (Creating the Entity File for Student)

In this step we create a student object the maps directly

To a document

```
package com.mongo;  
  
import org.springframework.data.annotation.Id;  
  
import  
org.springframework.data.mongodb.core.mapping.Document;
```

```
@Document(collection = "students")

public class Student {

    @Id

    private String id;

    private String name;

    private String studentId;

    public Student(String name, String studentId) {

        this.name = name;

        this.studentId = studentId;

    }
}
```

```
@Override

    public String toString() {

        return "Student{id='" + id + "', name='" +
name + "', studentId='" + studentId + "'}";

    }

}
```

The File is similar to dotnet entity the only thing that is different is the syntax

#### Step 2.4 (creating command line application using springboot and connecting to mongodb)

```
package com.mongo;
```



```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.data.mongodb.core.MongoTemplate;


@SpringBootApplication

public class App implements CommandLineRunner {

    @Autowired

    private MongoTemplate mongoTemplate;
```

```
public static void main(String[] args) {

    SpringApplication.run(App.class, args);

}

@Override

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

    System.out.println("--- App Started: Connecting to student_db
---");

    //this line create a new student

    Student newStudent = new Student("Alice Smith", "S12345");

    //this line save the student to the database
```

```
mongoTemplate.save(newStudent);

System.out.println("Inserted: " + newStudent);

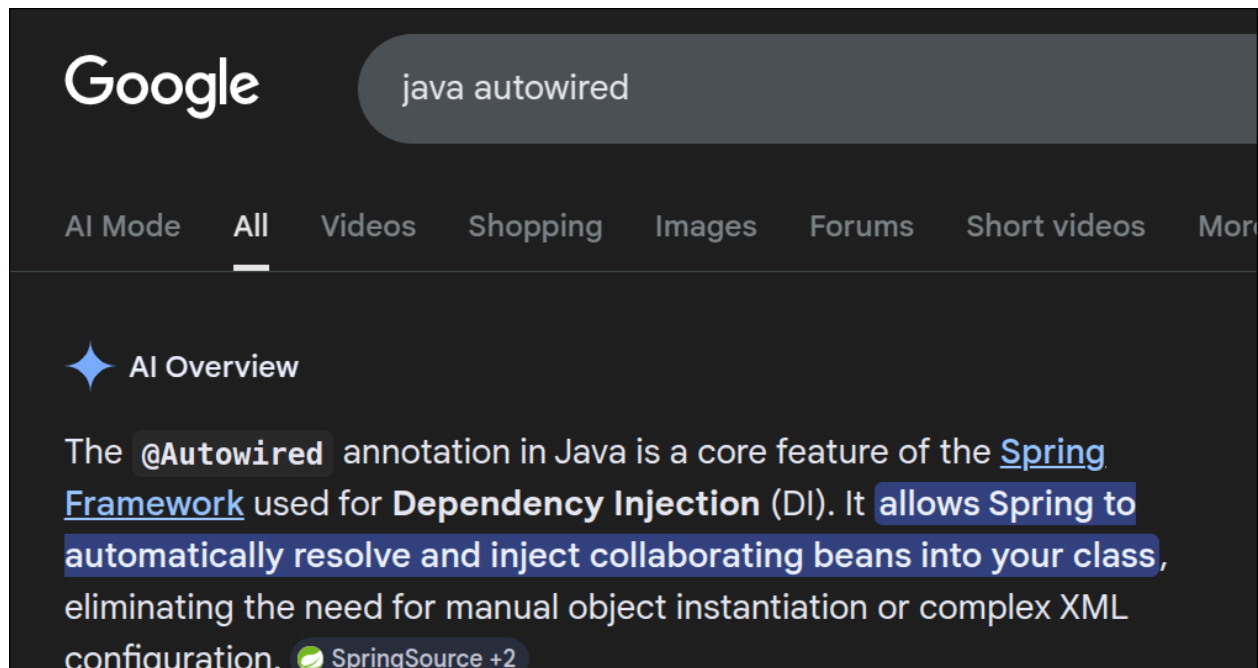
// read all student from the database

System.out.println("Current Students in DB:");

mongoTemplate.findAll(Student.class).forEach(System.out::println);

}

}
```



The code create a student object and save it into the database using monotemplate DI

It also print all the students in the collection

After running

Mvn spring-boot:run

```
inserted. Student{id= '69d8f64ab87d322cff57e0b1', name= 'Alice Smith', studentId= 'S12345'}
Current Students in DB:
Student{id= '69d8ebb4b67f1759fe44ba89', name= 'INIT Student', studentId= 'null'}
Student{id= '69d8ed6459fcce48b344f303', name= 'Alice Smith', studentId= 'S12345'}
Student{id= '69d8f64ab87d322cff57e0b1', name= 'Alice Smith', studentId= 'S12345'}
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.294 s
[INFO] Finished at: 2026-04-10T15:08:26+02:00
[INFO] -----

Fri 10 Apr - 16:08 ~ /UNI/adistr/MongoProj/SpringMongo master 3*
@drnull mvn spring-boot:run
```

Also using mongosh:

```
]
student_db> db.students.find().pretty()
[
  {
    _id: ObjectId('69d8ebb4b67f1759fe44ba89'),
    name: 'INIT Student',
    email: 'test@test.com'
  },
  {
    _id: ObjectId('69d8ed6459fcce48b344f303'),
    name: 'Alice Smith',
    studentId: 'S12345',
    _class: 'com.mongo.Student'
  },
  {
    _id: ObjectId('69d8f64ab87d322cff57e0b1'),
    name: 'Alice Smith',
    studentId: 'S12345',
    _class: 'com.mongo.Student'
  }
]
```

It worked nicely and one note that the app class implements `CommandLineRunner` which tells springboot that we are writing a cmd application and not a web interface

Step 3 (Generate millions of users and store them in MongoDB, read them, and measure the time for both read and write operations.)

First we index the db using the userid to make the read faster

Good video on database indexing is this

["https://www.youtube.com/watch?v=IYh6LrSIDvY"](https://www.youtube.com/watch?v=IYh6LrSIDvY)

This my first time working with indexing with a NoSql database

I thought indexing only works with Sql databases

```
package com.mongo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.data.mongodb.core.MongoTemplate;

import org.springframework.data.mongodb.core.query.Criteria;

import org.springframework.data.mongodb.core.query.Query;

import org.springframework.util.StopWatch;


import java.util.ArrayList;

import java.util.List;

import java.util.stream.Stream;


@SpringBootApplication

public class App implements CommandLineRunner {
```



```
@Autowired

private MongoTemplate mongoTemplate;


public static void main(String[] args) {

    SpringApplication.run(App.class, args);

}


@Override

public void run(String... args) {

    // Configuration

    int totalRecords = 1_000_000;

    int batchSize = 10_000;
```

```
// Clean database before starting

mongoTemplate.dropCollection(Student.class);


// Run Tests

performWriteTest(totalRecords, batchSize);

performReadTest();

}


/**

* Measures batch write performance for 1 million records

*/
```

```
private void performWriteTest(int totalRecords, int batchSize) {

    System.out.println("\n STARTING WRITE BENCHMARK ");

    Stopwatch watch = new Stopwatch();

    watch.start();

    for (int i = 0; i < totalRecords / batchSize; i++) {

        List<Student> batch = new ArrayList<>();

        for (int j = 0; j < batchSize; j++) {

            batch.add(new Student("Student_" + (i * batchSize + j),
"ID_" + (i * batchSize + j)));

        }

    }

}
```

```
        //insert here is faster than save() because we don't worry
        about duplication here

        mongoTemplate.insert(batch, Student.class);

    }

    watch.stop();

    System.out.printf("WRITE COMPLETE: %d records in %.2f seconds (%d
ops/sec)%n",

        totalRecords, watch.getTotalTimeSeconds(),
(int) (totalRecords / watch.getTotalTimeSeconds()));

}

/**

    * Measures smart read performance (Indexed vs Streaming)
```

```
*/

private void performReadTest() {

    System.out.println("\n STARTING SMART READ BENCHMARK ");

    // We pick a random ID from the middle of the set

    String searchId = "ID_543210";

    Stopwatch pointWatch = new Stopwatch();

    pointWatch.start();

    Query query = new Query(Criteria.where("studentId").is(searchId));

    Student found = mongoTemplate.findOne(query, Student.class);
```

```
pointWatch.stop();

System.out.println("1. POINT READ (Indexed Search):");

System.out.println("    - Found: " + found);

System.out.println("    - Time: " + pointWatch.getTotalTimeMillis()
+ " ms");

// Full Stream Read (Testing Retrieval Speed without crashing RAM)

StopWatch streamWatch = new StopWatch();

streamWatch.start();

long count = 0;

// Using Stream ensures we don't load 1M objects into RAM at once
```

```
        try (Stream<Student> studentStream = mongoTemplate.stream(new
Query(), Student.class)) {

            count = studentStream.count();

        }

        streamWatch.stop();

        System.out.println("2. STREAM READ (Full Collection Scan):");

        System.out.println("    - Processed: " + count + " records");

        System.out.println("    - Time: " +
streamWatch.getTotalTimeSeconds() + " seconds");

    }
}
```

We use three notes here

1. We can't write one million record all at once that is why we used batches
2. The read test contains two part the first part is trying to find specific record since we used indexed here it only load one record into memory

We can't load the 1million record into memory so indexing is somewhat necessary here

[3.we](#) used streams to count all the record without loading all of them into memory streaming is clever here

Step 4 (running the program)



```

Fri 10 Apr - 16:53 ~/UNI/adistr/MongoProj/SpringMongo | master 5*
@drnull> mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.mongo:SpringMongo >-----
[INFO] Building SpringMongo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.3.2:clean (default-clean) @ SpringMongo ---
[INFO] Deleting /home/drnull/UNI/adistr/MongoProj/SpringMongo/target
[INFO]
[INFO] --- maven-resources-plugin:3.3.1:resources (default-resources) @ SpringMongo ---
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] Copying 0 resource from src/main/resources to target/classes
[INFO]
[INFO] --- maven-compiler-plugin:3.11.0:compile (default-compile) @ SpringMongo ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 2 source files with javac [debug release 17] to target/classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.126 s
[INFO] Finished at: 2026-04-10T15:53:56+02:00
[INFO]
Fri 10 Apr - 16:54 ~/UNI/adistr/MongoProj/SpringMongo | master 5*
@drnull>

```

```

. Started App in 0.002 seconds (process running for 0.1
STARTING WRITE BENCHMARK
WRITE COMPLETE: 1000000 records in 18.15 seconds (55092 ops/sec)

--- STARTING SMART READ BENCHMARK ---
1. POINT READ (Indexed Search):
  - Found: Student{name='Student_543210', studentId='ID_543210'}
  - Time: 183 ms
2. STREAM READ (Full Collection Scan):
  - Processed: 1000000 records
  - Time: 9.04499219 seconds
[INFO] -----

```

Some good results

### Step 5 (trying different parameters)

2 Million records

```
STARTING WRITE BENCHMARK
WRITE COMPLETE: 2000000 records in 35.91 seconds (55701 ops/sec)

STARTING SMART READ BENCHMARK
1. POINT READ (Indexed Search):
  - Found: Student{name='Student_543210', studentId='ID_543210'}
  - Time: 184 ms
2. STREAM READ (Full Collection Scan):
  - Processed: 2000000 records
  - Time: 18.241512191 seconds
```

Double the amount of the time for 1 million records