

Spring MongoDB

~Add the following dependency:-

*Spring Web

*Spring Boot DevTools

*Lombok

*Spring Data MongoDB

Book.java

```
package com.example.demo;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Document(collection="Book")

public class Book {

    @Id
    private int id;
    private String bookName;
    private String authorName;

    public Book()
    {
    }

    public Book(int id, String bookName, String authorName)
    {
    }
```

```
super();
this.id=id;
this.bookName=bookName;
this.authorName=authorName;
}
public int getId() {

return id;
}
public void setId(int id) { this.id = id;
}
public String getBookName() { return bookName;
}
public void setBookName(String bookName) { this.bookName = bookName;
} public String getAuthorName() { return authorName;
}
public void setAuthorName(String authorName) { this.authorName =
authorName;
}
}
```

BookRepo.java

```
package com.example.demo;

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

@Repository
public interface BookRepo extends MongoRepository<Book, Integer> {
```

```
// No custom methods needed for basic CRUD operations.  
// MongoRepository already provides methods like save(), findAll(),  
findById(), and deleteById().  
}
```

DemoApplication.java

```
package com.example.demo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DemoApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(DemoApplication.class, args);  
    }  
  
}
```

MainController.java

```
package com.example.demo;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
```

```
@RestController
```

```
public class MainController {
```

```
@Autowired
```

```
private BookRepo repo;
```

```
@PostMapping("/addbook")
```

```
public String saveBook(@RequestBody Book book)
```

```
{
```

```
repo.save(book);
```

```
return "Added Successfully";
```

```
}
```

```
@GetMapping("/findAllBooks")
```

```
public List<Book>getBooks()
```

```
{
```

```
return repo.findAll();
```

```
}
```

```
@DeleteMapping("/delete/{id}")
```

```
public String deleteBook(@PathVariable int id)
```

```
{
```

```
repo.deleteById(id);
```

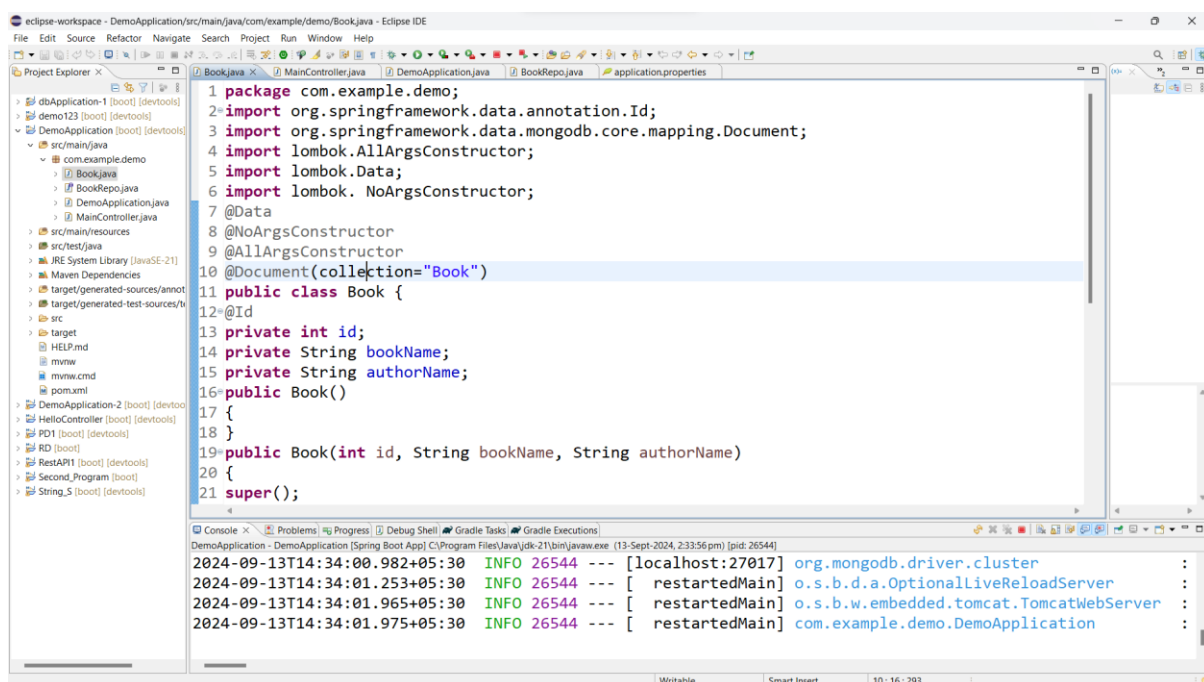
```
return"deleted Successfully";
```

```
}
```

```
@PutMapping("")
```

```
+ "")
```

```
public String updateBook(@RequestBody Book book)
{
    repo.save(book);
    return "update Successfully";
}
}
```



The screenshot shows the Eclipse IDE with the `Book.java` file open. The code defines a `Book` class with attributes `id`, `bookName`, and `authorName`, and methods `getId()`, `setId()`, `getBookName()`, `setBookName()`, `getAuthorName()`, and `setAuthorName()`.

```
20 {
21     super();
22     this.id=id;
23     this.bookName=bookName;
24     this.authorName=authorName;
25 }
26 public int getId() {
27
28     return id;
29 }
30 public void setId(int id) { this.id = id;
31 }
32 public String getBookName() { return bookName;
33 }
34 public void setBookName(String bookName) { this.bookName = bookName;
35 } public String getAuthorName() { return authorName;
36 }
37 public void setAuthorName(String authorName) { this.authorName = authorName;
38 }
39 }
40 }
```

The console output shows the application starting successfully on `localhost:27017`.

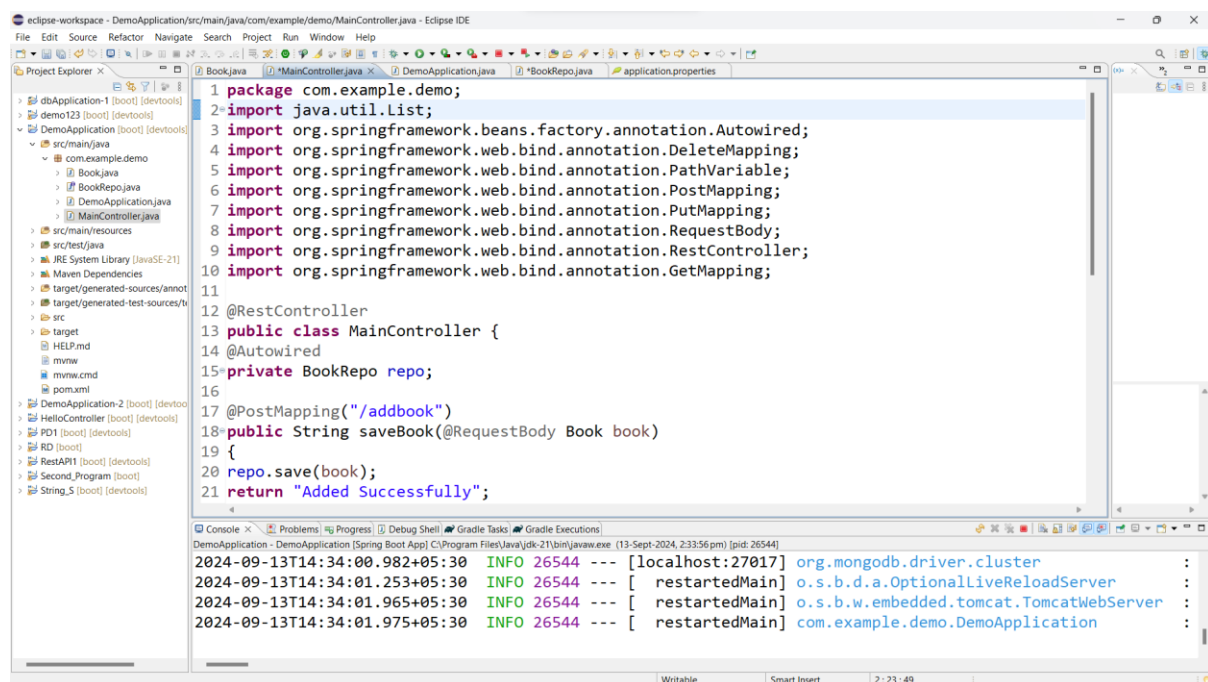
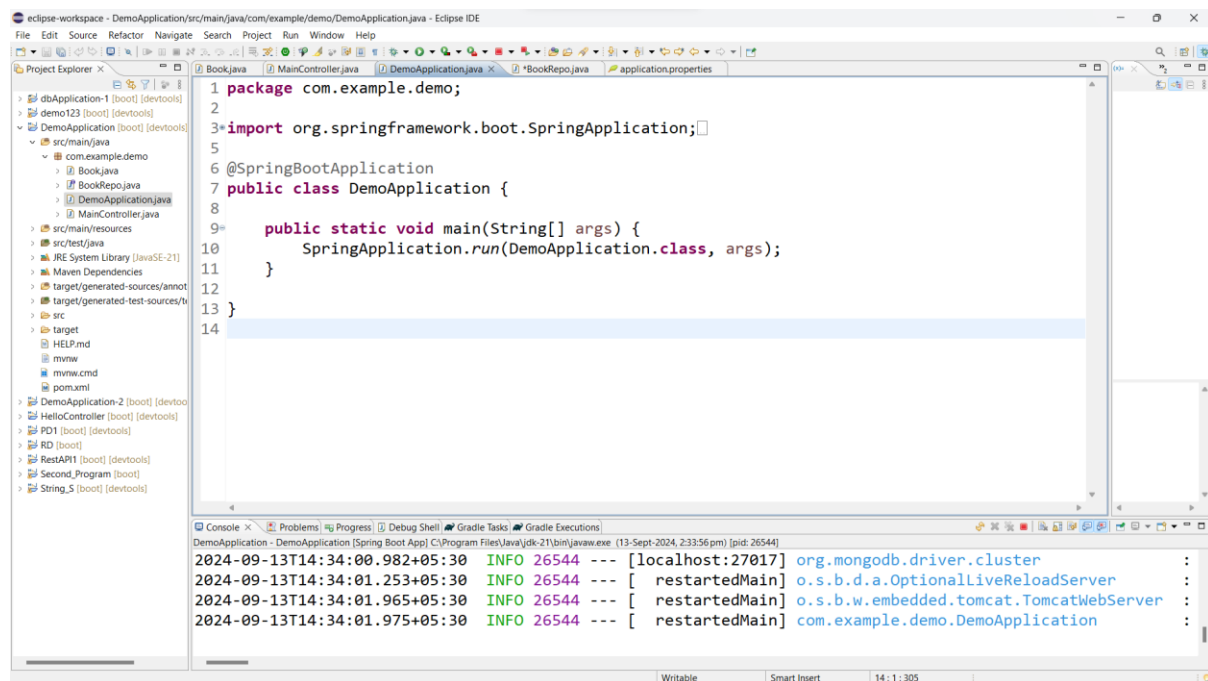
```
2024-09-13T14:34:00.982+05:30 INFO 26544 --- [localhost:27017] org.mongodb.driver.cluster :
2024-09-13T14:34:01.253+05:30 INFO 26544 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
2024-09-13T14:34:01.965+05:30 INFO 26544 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer :
2024-09-13T14:34:01.975+05:30 INFO 26544 --- [ restartedMain] com.example.demo.DemoApplication :
```

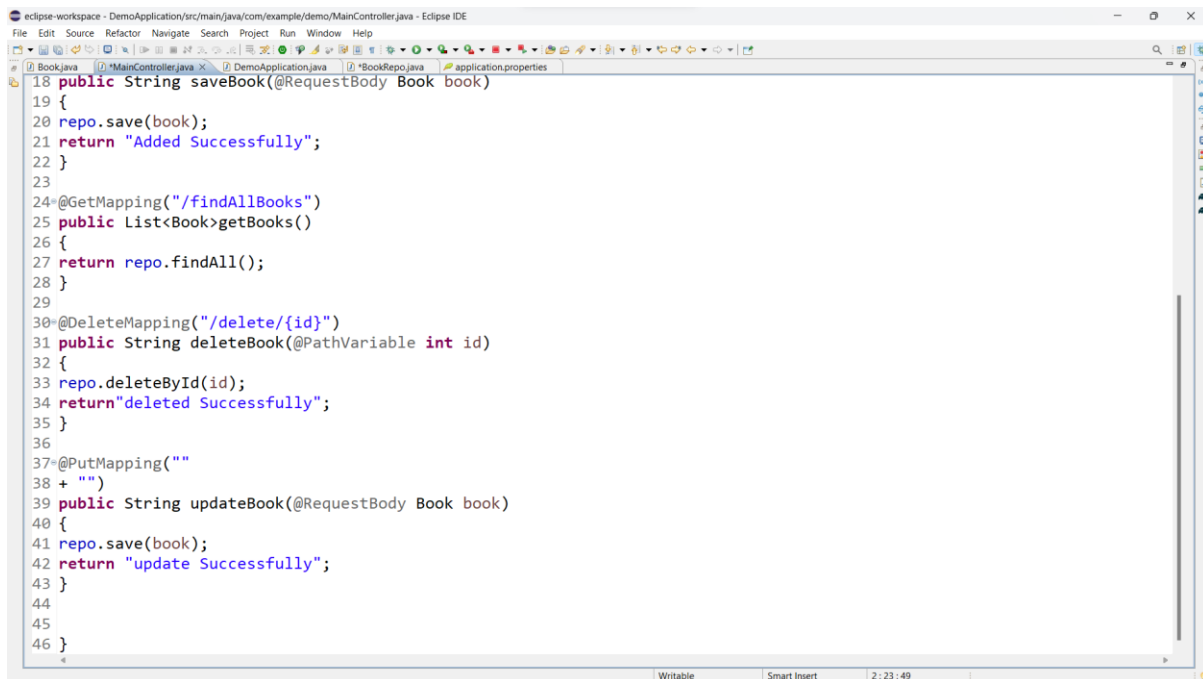
The screenshot shows the Eclipse IDE with the `BookRepo.java` file open. The code defines a `BookRepo` interface that extends `MongoRepository<Book, Integer>`.

```
1 package com.example.demo;
2
3 import org.springframework.data.mongodb.repository.MongoRepository;
4
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface BookRepo extends MongoRepository<Book, Integer> {
9     // No custom methods needed for basic CRUD operations.
10    // MongoRepository already provides methods like save(), findAll(), findById(), and del
11 }
12 }
```

The console output is identical to the previous screenshot, showing the application starting successfully on `localhost:27017`.

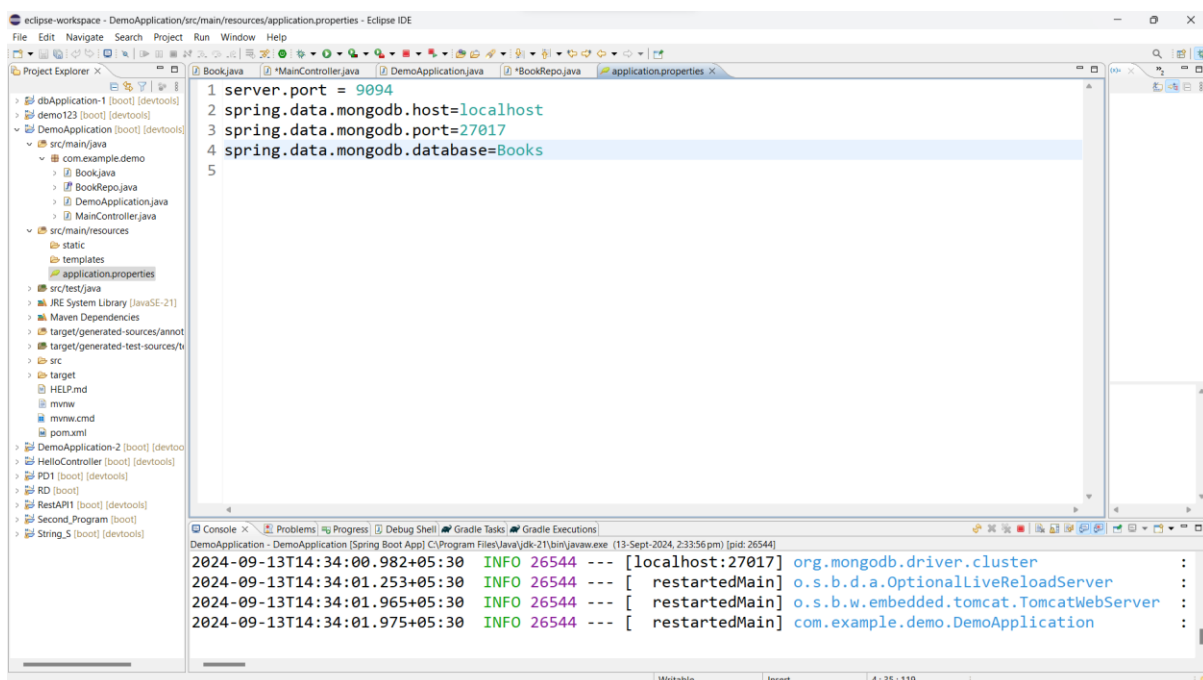
```
2024-09-13T14:34:00.982+05:30 INFO 26544 --- [localhost:27017] org.mongodb.driver.cluster :
2024-09-13T14:34:01.253+05:30 INFO 26544 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
2024-09-13T14:34:01.965+05:30 INFO 26544 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer :
2024-09-13T14:34:01.975+05:30 INFO 26544 --- [ restartedMain] com.example.demo.DemoApplication :
```





The screenshot shows the Eclipse IDE with the `MainController.java` file open. The code defines a Spring Boot controller for a book management application. It includes methods for saving, finding, deleting, and updating books, each returning a success message. The controller is annotated with `@RestController` and `@RequestMapping("/api/books")`.

```
18 public String saveBook(@RequestBody Book book)
19 {
20     repo.save(book);
21     return "Added Successfully";
22 }
23
24 @GetMapping("/findAllBooks")
25 public List<Book> getBooks()
26 {
27     return repo.findAll();
28 }
29
30 @DeleteMapping("/delete/{id}")
31 public String deleteBook(@PathVariable int id)
32 {
33     repo.deleteById(id);
34     return "deleted Successfully";
35 }
36
37 @PutMapping("/{id}")
38 public String updateBook(@RequestBody Book book)
39 {
40     repo.save(book);
41     return "update Successfully";
42 }
43 }
44
45 }
```



The screenshot shows the Eclipse IDE with the `application.properties` file open. The properties are configured for a Spring Boot application running on port 9094, using a MongoDB database named 'Books' on localhost. The console output shows the application starting successfully and connecting to the database.

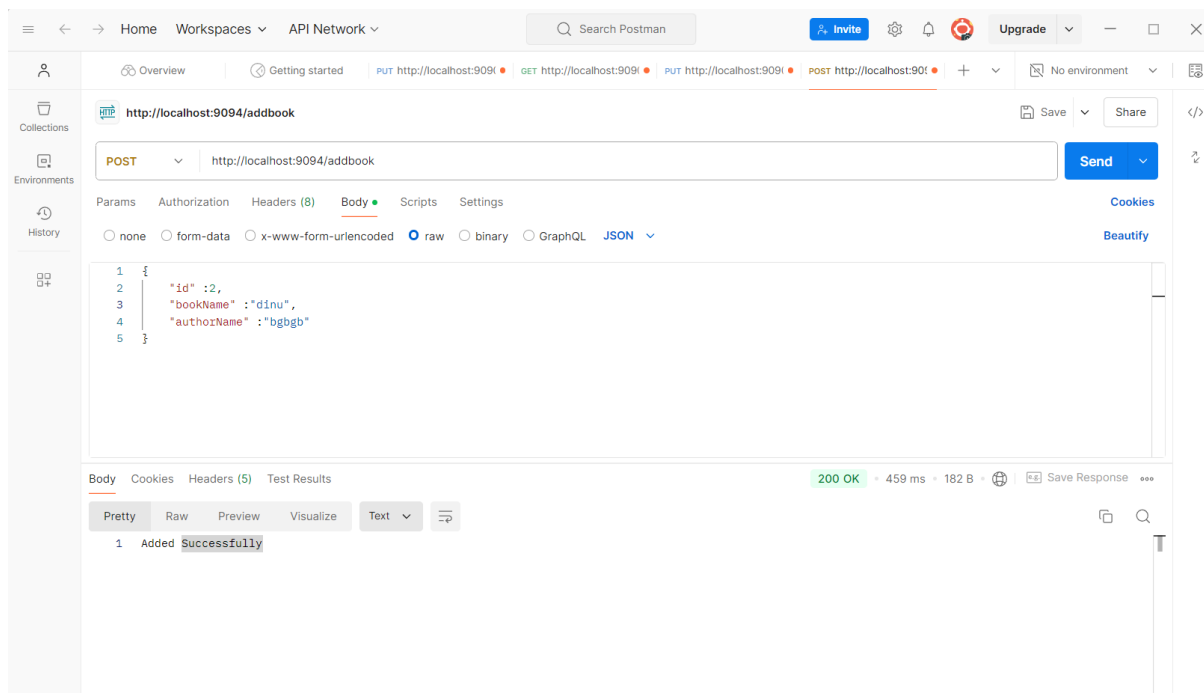
```
1 server.port = 9094
2 spring.data.mongodb.host=localhost
3 spring.data.mongodb.port=27017
4 spring.data.mongodb.database=Books
5
```

Console Output:

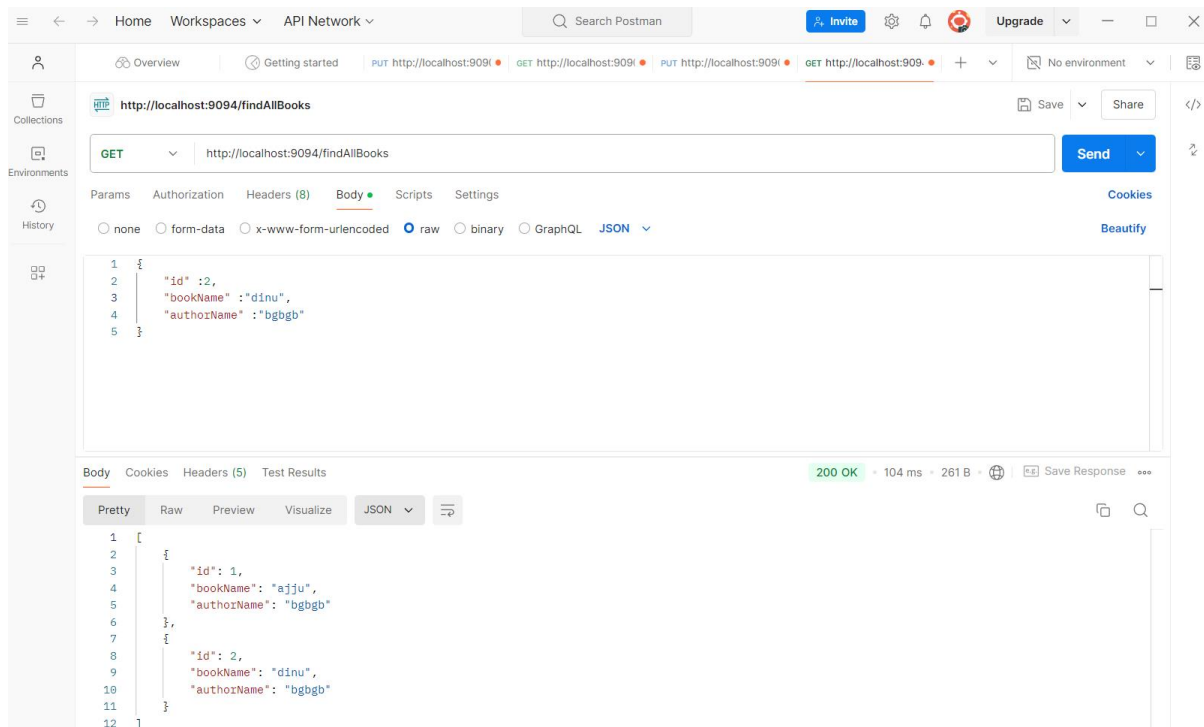
```
2024-09-13T14:34:00.982+05:30 INFO 26544 --- [localhost:27017] org.mongodb.driver.cluster :
2024-09-13T14:34:01.253+05:30 INFO 26544 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
2024-09-13T14:34:01.965+05:30 INFO 26544 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer :
2024-09-13T14:34:01.975+05:30 INFO 26544 --- [ restartedMain] com.example.demo.DemoApplication :
```

Output:-

POST= <http://localhost:9094/addbook>



PUT= <http://localhost:9094/findAllBooks>



Delete= <http://localhost:9094/delete/1>

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar labeled 'Search Postman' is present. The left sidebar shows 'Collections', 'Environments', 'History', and a 'Send' button. The main area shows a DELETE request to 'http://localhost:9094/delete/1'. The request body is a JSON object:

```
{  "id" : 2,  "bookName" : "dīnu",  "authozName" : "bggbg"}
```

. The response is a 200 OK status with a response time of 33 ms and a body of 'deleted Successfully'.

Home Workspaces API Network Search Postman

Overview Getting started PUT http://localhost:9094/ GET http://localhost:9094/ PUT http://localhost:9094/ DEL http://localhost:9094/ No environment

http://localhost:9094/delete/1 Save Share

DELETE http://localhost:9094/delete/1 Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

```
1 {
2   "id" : 2,
3   "bookName" : "dīnu",
4   "authozName" : "bggbg"
5 }
```

Body Cookies Headers (5) Test Results 200 OK • 33 ms • 184 B • Save Response

Pretty Raw Preview Visualize Text

```
1 deleted Successfully
```