

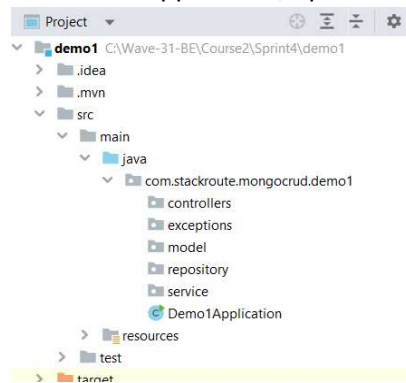
Step 1 Create spring boot application in spring initializr

The screenshot shows the Spring Initializr web application with the following configuration:

- Project:** ☒ Maven Project, ☐ Gradle Project
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.0.0 (SNAPSHOT), ☐ 3.0.0 (M5), ☐ 2.7.5 (SNAPSHOT), ☒ 2.7.4, ☐ 2.6.13 (SNAPSHOT), ☐ 2.6.12
- Project Metadata:**
 - Group: `com.stackroute.mongocrud`
 - Artifact: `demo1`
 - Name: `demo1`
 - Description: `Demo project for Spring Boot`
 - Package name: `com.stackroute.mongocrud.demo1`
 - Packaging: ☒ Jar, ☐ War
 - Java: ☐ 19, ☒ 17, ☐ 11, ☐ 8
- Dependencies:**
 - Spring Web (WEB):** Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Data MongoDB (MONGODB):** Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.
 - Lombok (DEVELOPER TOOLS):** Java annotation library which helps to reduce boilerplate code.

At the bottom, there is a status bar indicating "You are screen sharing" and a "Stop Share" button.

Step 2 download application, open in intellij, create required packages



Step 3 Define model classes
Customer HAS_A Address

```
10  @NoArgsConstructor
11  @AllArgsConstructor
12  @Data
13  @Document
14  public class Customer {
15      @Id
16      private String customerId;
17      private String name, email, mobile;
18      private Address address;
19  }

7  @NoArgsConstructor
8  @AllArgsConstructor
9  @Data
10 public class Address {
11     private String doorNo, street, area, city;
12 }
```

Step 4 Create custom exceptions

```
6  @ResponseStatus(code= HttpStatus.NOT_FOUND, reason="Customer not found")
7  public class CustomerNotFoundException extends Exception {
8  }

6  @ResponseStatus(code= HttpStatus.CONFLICT, reason="Customer already exists")
7  public class CustomerAlreadyExistingException extends Exception{
8  }
```

Step 5 Create repository

```
6  public interface CustomerRepository extends MongoRepository<Customer, String> {
7  }
```

Step 6 Create service layer

```
9 public interface CustomerService {
10     public abstract List<Customer> getAllCustomers();
11     public abstract Customer addCustomer(Customer customer) throws CustomerAlreadyExistingException;
12     public abstract Customer getCustomerById(String customerId) throws CustomerNotFoundException;
13 }

23 @Override
24 public Customer addCustomer(Customer customer) throws CustomerAlreadyExistingException {
25     // if new customer customerid existing in db, throw except
26     // else add record and return response
27     if( customerRepository.findById(customer.getId()).isPresent()){
28         throw new CustomerAlreadyExistingException();
29     }
30     else {
31         return customerRepository.insert(customer);
32     }
33 }

35 @Override
36 public Customer getCustomerById(String customerId) throws CustomerNotFoundException {
37     //
38     if(customerRepository.findById(customerId).isPresent()){
39         return customerRepository.findById(customerId).get();
40     }
41     else{
42         throw new CustomerNotFoundException();
43     }
44 }
```

Step 7 Define controller

```
13 @RequestMapping("/customerapp/v1")
14 public class CustomerController {
15     3 usages
16     @Autowired
17     private CustomerService customerService;
18
19     @GetMapping("/customer")
20     public ResponseEntity<?> getAllCustomers(){
21         return new ResponseEntity<>(customerService.getAllCustomers(), HttpStatus.OK);
22     }
23
24     @PostMapping("/customer")
25     public ResponseEntity<?> addCustomer(@RequestBody Customer customer) throws CustomerAlreadyExistingException{
26         try{
27             return new ResponseEntity<>(customerService.addCustomer(customer), HttpStatus.OK);
28         }
29         catch(CustomerAlreadyExistingException ex){
30             throw new CustomerAlreadyExistingException();
31         }
32     }
33
34     @GetMapping("/get-customer-by-id/{customerId}")
35     public ResponseEntity<?> getCustomerById(@PathVariable String customerId) throws CustomerNotFoundException {
36         try{
37             return new ResponseEntity<>(customerService.getCustomerById(customerId), HttpStatus.OK);
38         }
39         catch(CustomerNotFoundException ex){
40             throw new CustomerNotFoundException();
41         }
42     }
43 }
```

Activate Window:
Go to Settings to activate

Step 8 edit application.properties

```
1 server.port=9999
2 spring.data.mongodb.database=sprint4
3 spring.data.mongodb.uri=mongodb://localhost:27017/
4 spring.error.include-message=always
```

customer query for getting customers by city

in repository

```
2 usages
9 public interface CustomerRepository extends MongoRepository<Customer, String> {
10
11     1 usage
12     @Query("{ 'address.city' : {$in:[?0]}}")
13     public abstract List<Customer> getCustomersByCity(String city);
}
```

in service

```
public abstract List<Customer> getCustomersByCity(String city);
```

```
46 @Override
47 public List<Customer> getCustomersByCity(String city) {
48     return customerRepository.getCustomersByCity(city);
49 }
```

Controller

```
49 // GET
50 // http://localhost:9999/customerapp/v1/get-customers-by-city/Chennai
51 @GetMapping("/get-customers-by-city/{city}")
52 public ResponseEntity<?> getCustomersByCity(@PathVariable String city){
53     return new ResponseEntity<>(customerService.getCustomersByCity(city), HttpStatus.OK);
54 }
```