

## Lesson 01 Demo 06

### Reading Documents in MongoDB CRUD Operations

**Objective:** To read documents from MongoDB by executing the application and fetching a document based on the e-mail

**Tools required:** Eclipse IDE

**Prerequisites:** None

Steps to be followed:

1. Execute the App.java file
2. Fetch the document

#### Step 1: Execute the App.java file

- 1.1 Go to the browser, navigate to the **MongoDB** database, and select the **Browse Collections** option

The screenshot shows the MongoDB Atlas interface for a database deployment named 'Cluster0'. On the left, there's a sidebar with sections like 'DEPLOYMENT', 'DATA SERVICES', and 'SECURITY'. The main area displays metrics for the cluster, including connection counts and data sizes. At the top right, there's a green 'Create' button. Below the metrics, there's a section for 'Enhance Your Experience' with a 'Upgrade' button. The 'Browse Collections' button is highlighted with a yellow box.

## 1.2 Under the **MongoDBCRUDOperations** project, open the **DBOperations.java** file

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudoperations/dao/DBOperations.java - Eclipse Java EE IDE Platform". The left side features the "Package Explorer" view, which lists the project structure:

- MongoDBCRUDOperations
- src/main/java
  - com.example.mongodbcrudoperations
  - com.example.mongodbcrudoperations.dao
    - DBOperations.java
  - com.example.mongodbcrudoperations.model
  - com.example.mongodbcrudoperations.util
- src/test/java
- JRE System Library [j2SE-1.5]
- Maven Dependencies
- src
- target
- pom.xml

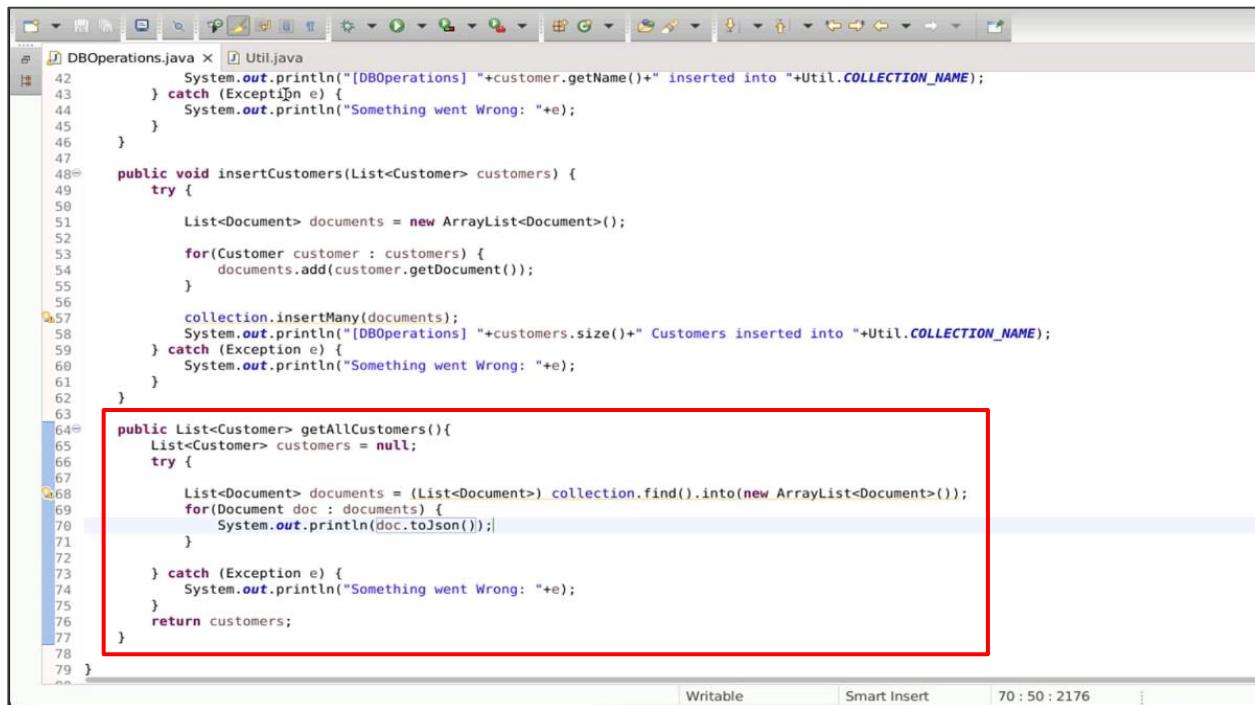
The right side shows the code editor with the file "DBOperations.java" open. The code is as follows:

```
1 package com.example.mongodbcrudoperations.dao;
2
3 import java.util.ArrayList;
4
5 public class DBOperations {
6
7     MongoClient client;
8     MongoDatabase database;
9     MongoCollection collection;
10
11    public DBOperations() {
12
13        try {
14            String connectionString = "mongodb+srv://john:john12345@cluster0.acllp.m";
15            client = MongoClients.create(connectionString);
16            System.out.println("[DBOperations] Connection Created");
17            database = client.getDatabase(Util.DB_NAME);
18            System.out.println("[DBOperations] Database Selected as eStore");
19            //collection = database.getCollection(Util.COLLECTION_NAME);
20        } catch (Exception e) {
21            e.printStackTrace();
22        }
23    }
24}
```

The "Console" tab in the bottom right corner displays the message "No consoles to display at this time."

**Note:** Please refer to the previous demo on how to create the **MongoDBCRUDOperations** project

1.3 Write the following lines of code (from lines 64 to 76) to create a method **Customer** to return the list of customers:



```
42         System.out.println("[DBOperations] "+customer.getName()+" inserted into "+Util.COLLECTION_NAME);
43     } catch (Exception e) {
44         System.out.println("Something went Wrong: "+e);
45     }
46
47
48     public void insertCustomers(List<Customer> customers) {
49         try {
50             List<Document> documents = new ArrayList<Document>();
51
52             for(Customer customer : customers) {
53                 documents.add(customer.getDocument());
54             }
55
56             collection.insertMany(documents);
57             System.out.println("[DBOperations] "+customers.size()+" Customers inserted into "+Util.COLLECTION_NAME);
58         } catch (Exception e) {
59             System.out.println("Something went Wrong: "+e);
60         }
61     }
62
63
64     public List<Customer> getAllCustomers(){
65         List<Customer> customers = null;
66         try {
67
68             List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
69             for(Document doc : documents) {
70                 System.out.println(doc.toJson());
71             }
72
73         } catch (Exception e) {
74             System.out.println("Something went Wrong: "+e);
75         }
76         return customers;
77     }
78
79 }
```

The code block shows the `getAllCustomers()` method highlighted with a red box. This method uses MongoDB's `find()` operation to retrieve all documents from the collection and converts them into a list of `Customer` objects. The `toJson()` method is used to print each document as a JSON string.

1.4 Go to the **App.java** file, write the following line of code (line 34) to call the **getAllCustomers** method, and then comment out the following lines of code (from lines 21 to 32):

```
1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5 /**
6  * Hello world!
7  *
8  */
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println("MongoDB CRUD Operations App");
14         DBOperations operations = new DBOperations();
15
16         /*Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com", 98.4f, new Date(), new Date());
17         System.out.println("Customer Details: ");
18         System.out.println(customer);
19         operations.insertCustomer(customer);*/
20
21
22         /*List<Customer> customers = new ArrayList<Customer>();
23         customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date()));
24         customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date()));
25         customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date()));
26
27         operations.insertCustomers(customers);*/
28
29         operations.getAllCustomers();
30
31     }
32
33
34
35
36
37
38 }
```

1.5 Now, run the code. You should see the following output with all the document data:

```

1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5 /**
6  * Hello world!
7  */
8
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println("MongoDB CRUD Operations App" );
14         DBOperations operations = new DBOperations();
15
16         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
17         System.out.println("Customer Details: ");
18         System.out.println(customer);
19         operations.insertCustomer(customer);*/
20
21
22         /*List<Customer> customers = new ArrayList<Customer>();
23         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
24         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date());
25         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
26
27         operations.insertCustomers(customers);*/
28
29         operations.getAllCustomers();
30
31
32     }
33
34
35
36
37 }
38
39

```

```

<terminated> App [3] [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.host:
mongodiagnostics.logging.Loggers shouldUseSLF4J
the classpath. Logging is disabled for the 'org.mongodb.driver' component
atted as eStore
eStore selected as customers
d47568840122c", "name": "John Watson", "phone": "+91 999999 11111", "email": "5da6c1fa283e9"}, "name": "Fionna", "phone": "+91 999999 22222", "email": "5da6c1fa283ea"}, "name": "Mike", "phone": "+91 999999 33333", "email": "5da6c1fa283eb"}, "name": "Anna", "phone": "+91 999999 44444", "email": "5da6c1fa283ec"}]

```

1.6 Go to the **DBOperations.java** file and write the following lines of code (lines 71, 72) to convert the documents into the **customer** data type:

```

42         System.out.println("[DBOperations] "+customer.getName()+" inserted into "+Util.COLLECTION_NAME);
43     } catch (Exception e) {
44         System.out.println("Something went Wrong: "+e);
45     }
46 }
47
48 public void insertCustomers(List<Customer> customers) {
49     try {
50
51         List<Document> documents = new ArrayList<Document>();
52
53         for(Customer customer : customers) {
54             documents.add(customer.getDocument());
55         }
56
57         collection.insertMany(documents);
58         System.out.println("[DBOperations] "+customers.size()+" Customers inserted into "+Util.COLLECTION_NAME);
59     } catch (Exception e) {
60         System.out.println("Something went Wrong: "+e);
61     }
62 }
63
64 public List<Customer> getAllCustomers(){
65     List<Customer> customers = null;
66     try {
67
68         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
69         for(Document doc : documents) {
70             //System.out.println(doc.toJson());
71             System.out.println(doc.get("name"));
72             System.out.println(doc.get("intime"));
73         }
74
75     } catch (Exception e) {
76         System.out.println("Something went Wrong: "+e);
77     }
78     return customers;
79 }

```

1.7 Go to the **App.java** file and run the application. You can see the entire details of customers.

```

public class App {
    public static void main( String[] args ) {
        operations.insertCustomer(customer);
        System.out.println("Customer Details: ");
        System.out.println(customer);
        insertCustomer(customer);
    }
}

```

Console Output:

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspace - Mo... Data | Cloud: MongoDB ...
eclipse-workspace - MongoCRUDOperations/src/main/java/com/example/mongodbcrudoperations/App.java - Eclipse IDE
Navigate Search Project Run Window Help
Util.java App.java
Problems Javadoc Declaration Console X
MongoDB CRUD Operations App
Jan 04, 2022 2:12:25 PM com.mongodb.diagnostics.logging.Loggers shouldUse5
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.*' package.
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
John Watson
Tue Jan 04 10:01:41 UTC 2022 I
Fionna
Tue Jan 04 10:08:24 UTC 2022
Mike
Tue Jan 04 10:08:24 UTC 2022
Anna
Tue Jan 04 10:08:24 UTC 2022

```

1.8 Go to the **DBOperations.java** file and write the following lines of code (from lines 66 to 70) to can define a document into a **Customer** object:

```

49
50
51     List<Document> documents = new ArrayList<Document>();
52
53     for(Customer customer : customers) {
54         documents.add(customer.getDocument());
55     }
56
57     collection.insertMany(documents);
58     System.out.println("[DBOperations] "+customers.size()+" Customers inserted into "+Util.COLLECTION_NAME);
59 } catch (Exception e) {
60     System.out.println("Something went Wrong: "+e);
61 }
62 }
63
64 public Customer convertDocumentToCustomer(Document doc) {
65     Customer customer = new Customer();
66
67     customer.setName(doc.get("name").toString());
68     customer.setPhone(doc.get("phone").toString());
69     customer.setEmail(doc.get("email").toString());
70
71     return customer;
72 }
73
74 public List<Customer> getAllCustomers(){
75     List<Customer> customers = new ArrayList<Customer>();
76     try {
77
78         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79         for(Document doc : documents) {
80             customers.add(convertDocumentToCustomer(doc));
81         }
82
83     } catch (Exception e) {
84         System.out.println("Something went Wrong: "+e);
85     }
86     return customers;
87 }
```

1.9 Now, go to the **App.java** file, call the **getAllCustomers()** method (lines 34 to 36), and run the code. You can see the list of customers, and the other details such as temperature and in-and-out time as **null**.

```

1 package com.example.mongodboperations;
2
3 import java.util.ArrayList;
4
5 /**
6 * Hello world!
7 */
8
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println("MongoDB CRUD Operations App" );
14         DBOperations operations = new DBOperations();
15
16         /*Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com");
17         operations.insertCustomer(customer);*/
18         System.out.println("Customer Details: ");
19         System.out.println(customer);
20         operations.insertCustomer(customer);*/
21     }
22 }

```

<terminated> App [3] [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot/jre/lib/server/libjvm.so
MongoDB CRUD Operations App
Jan 04, 2022 2:16:29 PM com.mongodb.diagnostics.logging.Logger\$ shouldUse5
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.\*' logger.
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from esstore selected as customers
Customer [name=John Watson, phone=+91 999999 1111, email=john@example.com]
Customer [name=Fionna, phone=+91 999999 2222, email=fionna@example.com, temperature=null]
Customer [name=Mike, phone=+91 999999 3333, email=mike@example.com, temperature=null]
Customer [name=Anna, phone=+91 999999 4444, email=anna@example.com, temperature=null]

```

eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudoperations/App.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
DBOperations.java Util.java App.java Customer.java
1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5 /**
6  * Hello world!
7  */
8
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println("MongoDB CRUD Operations App" );
14         DBOperations operations = new DBOperations();
15
16         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", "1234567890", "1234567890", "1234567890");
17         System.out.println("Customer Details: ");
18         System.out.println(customer);
19         operations.insertCustomer(customer);*/
20
21
22         /*List<Customer> customers = new ArrayList<Customer>();
23         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date());
24         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date());
25         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));*/
26
27
28
29
30

```

s.logging.Loggers shouldUseSLF4J  
logging is disabled for the 'org.mongodb.driver' component

as customers  
1111, email=john@example.com, temperature=null, intime=null, outtime=null  
email=fionna@example.com, temperature=null, intime=null, outtime=null  
mail=mike@example.com, temperature=null, intime=null, outtime=null  
mail=anna@example.com, temperature=null, intime=null, outtime=null

## Step 2: Fetch the document

2.1 To fetch a document based on email, go to the **DBOperations.java** file and create a method **getCustomerByEmail** (from lines 90 to 94):

```

DBOperations.java Util.java App.java Customer.java
60     }
61 }
62
63
64 public Customer convertDocumentToCustomer(Document doc) {
65     Customer customer = new Customer();
66
67     customer.setName(doc.get("name").toString());
68     customer.setPhone(doc.get("phone").toString());
69     customer.setEmail(doc.get("email").toString());
70
71     return customer;
72 }
73
74 public List<Customer> getAllCustomers(){
75     List<Customer> customers = new ArrayList<Customer>();
76     try {
77
78         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79         for(Document doc : documents) {
80             customers.add(convertDocumentToCustomer(doc));
81         }
82
83     } catch (Exception e) {
84         System.out.println("Something went Wrong: "+e);
85     }
86     return customers;
87 }
88
89
90 public Customer getCustomerByEmail(String email) {
91     Document filter = new Document("email", email);
92     Document document = (Document) collection.find(filter).first();
93     Customer customer = convertDocumentToCustomer(document);
94     return customer;
95 }
96
97

```

2.2 Go to the **App.java** file, add a print statement to display the customer email (from lines 42 to 45), and run the code. The following output can be seen when the customer details are fetched using the email **fionna@example.com**:

```
10=/*
11 * Hello world!
12 *
13 */
14 public class App
15 {
16=    public static void main( String[] args )
17    {
18        System.out.println("MongoDB CRUD Operations App");
19        DBOperations operations = new DBOperations();
20
21        /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22        System.out.println("Customer Details: ");
23        System.out.println(customer);
24        operations.insertCustomer(customer);*/
25
26
27        /*List<Customer> customers = new ArrayList<Customer>();
28        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date());
29        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date());
30        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32        operations.insertCustomers(customers);*/
33
34        List<Customer> customers = operations.getAllCustomers();
35        /*for(Customer customer : customers) {
36            System.out.println(customer);
37        }*/
38        customers.forEach(customer -> {
39            System.out.println(customer);
40        });
41
42        System.out.println("~~~~~");
43        System.out.println("Fetching customer with email: fionna@example.com");
44        Customer customer = operations.getCustomerByEmail("fionna@example.com");
45        System.out.println(customer);*/
46
47 }
```

2.3 Go to the **DBOperations.java** file, and mark the following lines of code as comments  
(lines 91, 92):

```

60         System.out.println("Something went wrong: " + e);
61     }
62 }
63
64 public Customer convertDocumentToCustomer(Document doc) {
65     Customer customer = new Customer();
66
67     customer.setName(doc.get("name").toString());
68     customer.setPhone(doc.get("phone").toString());
69     customer.setEmail(doc.get("email").toString());
70
71     return customer;
72 }
73
74 public List<Customer> getAllCustomers(){
75     List<Customer> customers = new ArrayList<Customer>();
76     try {
77
78         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79         for(Document doc : documents) {
80             customers.add(convertDocumentToCustomer(doc));
81         }
82
83     } catch (Exception e) {
84         System.out.println("Something went Wrong: " + e);
85     }
86     return customers;
87 }
88
89
90 public Customer getCustomerByEmail(String email) {
91     /*Document filter = new Document("email", email);
92     Document document = (Document) collection.find(filter).first();*/
93     Customer customer = convertDocumentToCustomer(document);
94     return customer;
95 }
96
97

```

Unexpected end of comment

Writable

Smart Insert

92 : 74 : 2774

2.4 Write the code (line 94) to pass the equal function as an input to the filters, go to the **App.java** file, and run the code. The output shown will be similar to the previous output, but it is a different way of executing them.

```

60         System.out.println("Something went wrong: " + e);
61     }
62 }
63
64 public Customer convertDocumentToCustomer(Document doc) {
65     Customer customer = new Customer();
66
67     customer.setName(doc.get("name").toString());
68     customer.setPhone(doc.get("phone").toString());
69     customer.setEmail(doc.get("email").toString());
70
71     return customer;
72 }
73
74 public List<Customer> getAllCustomers(){
75     List<Customer> customers = new ArrayList<Customer>();
76     try {
77
78         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79         for(Document doc : documents) {
80             customers.add(convertDocumentToCustomer(doc));
81         }
82
83     } catch (Exception e) {
84         System.out.println("Something went Wrong: " + e);
85     }
86     return customers;
87 }
88
89
90 public Customer getCustomerByEmail(String email) {
91     /*Document filter = new Document("email", email);
92     Document document = (Document) collection.find(filter).first();*/
93     Document document = (Document) collection.find(Filters.eq("email", email)).first();
94     Customer customer = convertDocumentToCustomer(document);
95     return customer;
96 }
97

```

Writable

Smart Insert

The screenshot shows the Eclipse IDE interface with two tabs open: `App.java` and `Customer.java`. The `App.java` tab contains the following code:

```

public class App {
    public static void main(String[] args) {
        System.out.println("MongoDB CRUD Operations App");
        DBOperations operations = new DBOperations();
        Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com");
        operations.insertCustomer(customer);
        System.out.println("Customer Details: ");
        System.out.println(customer);
        List<Customer> customers = operations.getAllCustomers();
        for (Customer customer : customers) {
            System.out.println(customer);
        }
    }
}

```

The `Customer.java` tab contains the definition of the `Customer` class and its methods.

In the `Console` tab, the application's output is displayed:

```

terminated> App [3] [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspace
MongoDB CRUD Operations App
Jan 04, 2022 2:30:55 PM com.mongodb.diagnostics.logging.Loggers shouldUseSystemDefaultLogger
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.*' package.
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=John Watson, phone=+91 999999 1111, email=john@example.com]
Customer [name=Fionna, phone=+91 999999 2222, email=fionna@example.com, tempId=1]
Customer [name=Mike, phone=+91 999999 3333, email=mike@example.com, tempId=2]
Customer [name=Anna, phone=+91 999999 4444, email=anna@example.com, tempId=3]
Fetching customer with email: fionna@example.com
Customer [name=Fionna, phone=+91 999999 2222, email=fionna@example.com, tempId=1]

```

2.5 Go to the `DBOperations.java` file and write the following lines of code (lines 81 & 82) to sort the documents in ascending order:

The screenshot shows the `DBOperations.java` code editor. The `getAllCustomers()` method is highlighted with a red box around the sorting logic:

```

public List<Customer> getAllCustomers() {
    List<Customer> customers = new ArrayList<Customer>();
    try {
        List<Document> documents = (List<Document>) collection
            .find()
            .sort(Sorts.ascending("name"))
            .into(new ArrayList<Document>());
        for(Document doc : documents) {
            customers.add(convertDocumentToCustomer(doc));
        }
    } catch (Exception e) {
        System.out.println("Something went Wrong: "+e);
    }
    return customers;
}

```

## 2.6 Go to the App.java file and run the code

The screenshot shows the Eclipse IDE interface with the 'App.java' file open in the editor. The code implements a MongoDB CRUD application. It starts by printing 'Hello world!', creating a 'Customer' object, and inserting it into the database. Then, it retrieves all customers and prints them out. Finally, it fetches a customer by email and prints that customer's details. A yellow box highlights the 'Run App (3)' button in the toolbar.

```

10 /**
11 * Hello world!
12 *
13 */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         System.out.println("MongoDB CRUD Operations App");
19         DBOperations operations = new DBOperations();
20
21         /*Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com", 98.4f, new Date(), new Date());
22         System.out.println("Customer Details: ");
23         System.out.println(customer);
24         operations.insertCustomer(customer);*/
25
26
27         /*List<Customer> customers = new ArrayList<Customer>();
28         customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date()));
29         customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date()));
30         customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32         operations.insertCustomers(customers);*/
33
34         List<Customer> customers = operations.getAllCustomers();
35         /*for(Customer customer : customers) {
36             System.out.println(customer);
37         }*/
38         customers.forEach(customer -> {
39             System.out.println(customer);
40         });
41
42         System.out.println("-----");
43         System.out.println("Fetching customer with email: fionna@example.com");
44         Customer customer = operations.getCustomerByEmail("fionna@example.com");
45         System.out.println(customer);
46     }
47 }

```

The following output shows the data sorted in ascending order of name:

The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the execution of the 'App' class. It shows the application starting, connecting to MongoDB, and listing all customers. The customers are listed in ascending order of name: Fionna, Anna, John Watson, and Mike. The console output ends with a message indicating the process completed at 2:42:07 PM on Jan 4, 2022.

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_16.0.2.v20210721-1149/re/bin/java (Jan 4, 2022, 2:42:07 PM - 2:42:10 PM)
MongoDB CRUD Operations App
Jan 4, 2022 2:42:07 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=Anna, phone=+91 999999 4444, email=anna@example.com, temperature=null, intime=null, outtime=null]
Customer [name=Fionna, phone=+91 999999 2222, email=fionna@example.com, temperature=null, intime=null, outtime=null]
Customer [name=John Watson, phone=+91 999999 1111, email=john@example.com, temperature=null, intime=null, outtime=null]
Customer [name=Mike, phone=+91 999999 3333, email=mike@example.com, temperature=null, intime=null, outtime=null]

Fetching customer with email: fionna@example.com
Customer [name=Fionna, phone=+91 999999 2222, email=fionna@example.com, temperature=null, intime=null, outtime=null]

```

To explore more, similar operations can be performed with different data and conditions. This is how we can read the documents in MongoDB by fetching details based on email.

By following these steps, you have successfully read documents from MongoDB by executing the application and fetching a document based on the e-mail.