

2025년 상반기 K-디지털 트레이닝

MongoDB Java 연동 (심화1)

[KB] IT's Your Life

- ✓ **users 컬렉션에 저장된 문서 하나의 id 값을 준비하고, 해당 문서를 다음 내용으로 수정하세요.**
 - username: "modified name"
 - lastUpdated: 실행된 실제 시간

```
package sec04;

public class UpdateOneTest {
    public static void main(String[] args) {

    }
}
```

✓ UpdateOneTest.java

```
package sec04;
...
public class UpdateOneTest {
    public static void main(String[] args) {
        MongoClient<Document> collection = Database.getCollection("users");

        String id = "666a6b65a25a4c74fddfedc2";
        Bson query = eq("_id", new ObjectId(id));

        Bson updates = Updates.combine(
            Updates.set("name", "modify name"),
            Updates.currentTimestamp("lastUpdated"));

        UpdateResult result = collection.updateOne(query, updates);
        System.out.println("==> UpdateResult : " + result.getModifiedCount());

        Database.close();
    }
}
```

✓ **users 컬렉션에 age가 16보다 큰 문서들을 다음 내용으로 수정하세요.**

- username: "modified name"
- lastUpdated: 실행된 실제 시간

```
package sec04;
```

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

✓ UpdateManyTest.java

```
package sec04;
...
public class UpdateManyTest {
    public static void main(String[] args) {
        MongoClient<Document> collection = Database.getCollection("users");

        int age = 16;
        Bson query = gt("age", age);

        Bson updates = Updates.combine(
            Updates.set("name", "modify name"),
            Updates.currentTimestamp("lastUpdated"));

        UpdateResult result = collection.updateMany(query, updates);
        System.out.println("==> UpdateManyResult : " + result.getModifiedCount());

        Database.close();
    }
}
```

- ✓ **users 컬렉션에 저장된 문서 하나의 id 값을 준비하고, 해당 문서를 삭제하세요.**

```
package sec05;

public class DeleteOneTest {
    public static void main(String[] args) {

    }
}
```

✓ DeleteOneTest.java

```
package sec05;
...

public class DeleteOneTest {
    public static void main(String[] args) {
        MongoClient<Document> collection = Database.getCollection("users");

        String id = "666a6b65a25a4c74fddfedc2";
        Bson query = eq("_id", new ObjectId(id));

        DeleteResult result = collection.deleteOne(query);
        System.out.println("==> DeleteResult : " + result.getDeletedCount());

        Database.close();
    }
}
```

✓ **users 컬렉션에 age > 15인 문서를 모두 삭제하세요.**

```
package sec05;

public class DeleteManyTest {
    public static void main(String[] args) {

    }
}
```


✓ DeleteManyTest.java

```
package sec05;
...

public class DeleteManyTest {
    public static void main(String[] args) {
        MongoClient<Document> collection = Database.getCollection("users");

        int age = 15;
        Bson query = gt("age", age);

        DeleteResult result = collection.deleteMany(query);
        System.out.println("==> DeleteManyResult : " + result.getDeletedCount());

        Database.close();
    }
}
```

2025년 상반기 K-디지털 트레이닝

MongoDB Java 연동 (심화2)

[KB] IT's Your Life

✓ **build.gradle에 롬복을 다음과 같이 추가하세요.**

```
dependencies {  
    compileOnly("org.projectlombok:lombok:1.18.32")  
    annotationProcessor("org.projectlombok:lombok:1.18.32")  
    testCompileOnly("org.projectlombok:lombok:1.18.32")  
    testAnnotationProcessor("org.projectlombok:lombok:1.18.32")  
    ...  
}
```

다음과 같이 Database.java 파일을 작성하세요.

```
package app;
...

import org.bson.codecs.configuration.CodecProvider;
import org.bson.codecs.configuration.CodecRegistry;
import org.bson.codecs.pojo.PojoCodecProvider;
import static com.mongodb.MongoClientSettings.getDefaultCodecRegistry;
import static org.bson.codecs.configuration.CodecRegistries.fromProviders;
import static org.bson.codecs.configuration.CodecRegistries.fromRegistries;

public class Database {
    static MongoClient client;
    static MongoDB database;

    static {
        CodecProvider pojoCodecProvider = PojoCodecProvider.builder().automatic(true).build();
        CodecRegistry pojoCodecRegistry = fromRegistries(getDefaultCodecRegistry(), fromProviders(pojoCodecProvider));

        ConnectionString connectionString = new ConnectionString("mongodb://127.0.0.1:27017");
        client = MongoClient.create(connectionString);
        database = client.getDatabase("todo_db").withCodecRegistry(pojoCodecRegistry);
    }
}
```

Database.java

```
public static void close() {
    client.close();
}

public static MongoDBDatabase getDatabase() {
    return database;
}

public static MongoCollection<Document> getCollection(String colName) {
    MongoCollection<Document> collection = database.getCollection(colName);
    return collection;
}

public static <T> MongoCollection<T> getCollection(String colName, Class<T> clazz) {
    MongoCollection<T> collection = database.getCollection(colName, clazz);
    return collection;
}
}
```

✓ 앞에서 연습한 Todo 문서의 내용을 가지는 POJO 클래스로 정의하세요.

- 롬복 사용
- ObjectId 필드 추가

```
package app.domain;  
  
public class Todo {  
  
}
```

Todo.java

```
package app.domain;
...
import org.bson.types.ObjectId;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Todo {
    private ObjectId id;
    private String title;
    private String desc;
    private boolean done;
}
```

✓ 다음 코드를 완성하세요.

- Todo 클래스를 이용한 단일 삽입 및 List<Todo> 이용한 다중 삽입
- todo 컬렉션 전체 문서 출력 및 특정 id에 대한 todo 추출 및 출력

```
package app;

public class App {
    public static void main(String[] args) {
        // todo 컬렉션 추출

        // 단일 insert

        // 다중 insert

        // 전체 todo 목록 출력
        // 특정 id값 기반으로 검색 후 출력

        Database.close();
    }
}
```


App.java

```
package app;

import app.domain.TODO;
import com.mongodb.client.MongoCollection;
import org.bson.conversions.Bson;
import org.bson.types.ObjectId;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.mongodb.client.model.Filters.eq;

public class App {
    public static void main(String[] args) {
        MongoCollection<TODO> collection = Database.getCollection("todo", TODO.class);
```

App.java

```
// insertOne
Todo newtodo = new Todo(null, "POJO", "POJO 테스트 확인", false);
collection.insertOne(newtodo);

// insertMany
List <Todo> newTodos = Arrays.asList(
    new Todo(null, "POJO2", "POJO2 테스트 확인", false),
    new Todo(null, "POJO3", "POJO3 테스트 확인", true),
    new Todo(null, "POJO4", "POJO4 테스트 확인", false)
);

collection.insertMany(newTodos);
```

App.java

```
// find()
List<Todo> todos = new ArrayList<>();
collection.find().into(todos);

for(Todo todo : todos) {
    System.out.println(todo);
}

// findOne()
String id = "666a6296f4fe57189cd03eea";
Bson query = eq("_id", new ObjectId(id));

Todo todo = collection.find(query).first();
System.out.println("==> findByIdResult : " + todo);

Database.close();
}
}
```