KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

MongoDB Java 연동 (심화1)

[KB] IT's Your Life

KB 국민은행

MongoDB Java 연동

- **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) {

    }
}
```

MongoDB Java 연동

- **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) {

    }
}
```

MongoDB Java 연동

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

```
package sec05;

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

    }
}
```

users 컬렉션에 age > 15인 문서를 모두 삭제하세요.

```
package sec05;

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

    }
}
```

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

MongoDB Java 연동 (심화2)

[KB] IT's Your Life

KB 국민은행

MongoDB Java POJO 연동

- **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")
    ...
}
```

MongoDB Java POJO 연동

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 MongoDatabase 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 = MongoClients.create(connectionString);
        database = client.getDatabase("todo_db").withCodecRegistry(pojoCodecRegistry);
    }
```

MongoDB Java POJO 연동

Database.java

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

    public static MongoDatabase 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;
    }
}
```

MongoDB Java POJO 연동

- 앞에서 연습한 Todo 문서의 내용을 가지는 POJO 클래스로 정의하세요.
  - 롬복 사용
  - ObjectId 필드 추가

```
package app.domain;

public class Todo {

}
```

MongoDB Java POJO 연동

- **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();
    }
}
```