Inserting, Updating, and Deleting Documents

Dan Geabunea
SENIOR SOFTWARE DEVELOPER

@romaniancoder www.romaniancoder.com

Overview

Insert new documents

Update existing documents

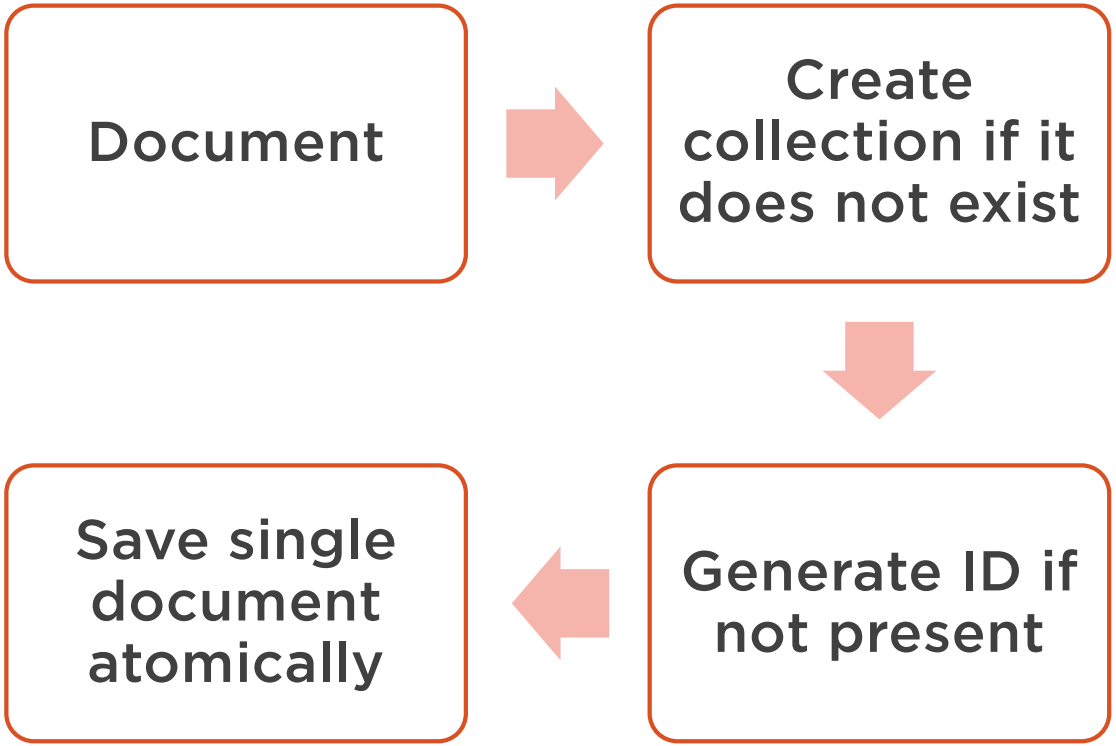
Remove documents

Using converters for custom serialization/deserialization of fields

Demo: Implementing Mongo insert, update and delete operations

Insert New Documents

Insert Process



Transactions

Single document transactions

An operation on a single document is atomic. Because relationships are embedded in a single document, this mostly eliminates the need for multi-document transactions.

Multi-document transactions

For situations that require atomicity to multiple documents (in single or multiple collections), Mongo 4+ supports multi document transactions.

Don’t overuse Mongo multi document transactions.

```
Person p = new Person("Anna", 18);
mongoTemplate.insert(p);
```

Inserting a New Document

_id	name	age

```
Person p = new Person("Anna", 18);

mongoTemplate.insert(p);
```

Inserting a New Document

_id	name	age
3d7745b3-2589-4976-a1fa-9f49ed31a673	Anna	18

```
Person p = new Person("6a791adf-15d9-4ce4-9efa-21ae27f66263", Anna", 18);
mongoTemplate.insert(p);
```

Inserting a New Document with Id

_id	name	age

```
Person p = new Person("6a791adf-15d9-4ce4-9efa-21ae27f66263", Anna", 18);
mongoTemplate.insert(p);
```

Inserting a New Document with Id

_id	name	age
6a791adf-15d9-4ce4-9efa-21ae27f66263	Anna	18

```
Person q = new Person(“6a791adf-15d9-4ce4-9efa-21ae27f66263“, Stan”, 34);
mongoTemplate.insert(q);
```

Inserting a Document with Existing Id

_id	name	age
6a791adf-15d9-4ce4-9efa-21ae27f66263	Anna	18

```
Person q = new Person(“6a791adf-15d9-4ce4-9efa-21ae27f66263“, Stan”, 34);
mongoTemplate.insert(q);

// => ERROR: Duplicate Key
```

Inserting a Document with Existing Id

_id	name	age
6a791adf-15d9-4ce4-9efa-21ae27f66263	Anna	18

```
Person a = new Person("Stan", 34);
Person b = new Person("Anna", 18);
Person c = new Person("John", 40);
mongoTemplate.insert(a);
mongoTemplate.insert(b);
mongoTemplate.insert(c);
```

Inserting Multiple Documents

It is a working solution, but it is inefficient

We have 3 round-trips to the database

```
Person a = new Person("Stan", 34);
Person b = new Person("Anna", 18);
Person c = new Person("John", 40);
List<Person> people = Arrays.asList(a,b,c);
mongoTemplate.insertAll(people);
```

Batch Insert

Same result

One round-trip to the database

Perfect for batch operations

Updating Documents

Update Types

Single Document Updates

Multiple Document Updates

```
Airport otp = mongoTemplate.findById("6a791adf-15d9-4ce4…");

otp.setFlightsPerDay(3000);
otp.setProximityWeather(Weather.RAIN);

mongoTemplate.save(otp);
```

Update Using the Save Method

6a791adf-15d9-4ce4	OTP	2500	CLOUDY

6a791adf-15d9-4ce4	OTP	3000	RAIN

How Save Works

Scans the collection and tries to find a document with a matching ID

If no document is found for the given ID, then Save acts like Insert. A new document is created with the provided ID.

If a document is found, then it is completely replaced with the provided one.

The Save method should really be called InsertOrUpdate

Insert vs. Save

Insert	Save
With no ID, generates one and inserts the document in the collection	With no ID generates one and inserts the document in the collection
With existing ID, if not present, inserts document in collection	With existing ID, if not present, inserts document in collection
With existing ID, throws an error	With existing ID, overwrites whole document with new values
Has batch operation support (insertMany)	Does not have batch operation support

Use insert for new documents and save for updates. Do not use save for both just because it feels easier.

Me, based on experience

Performing Batch Updates

Retrieve the documents you want to update using the Query object

Define the fields that you want to update along with the new values

Update those fields using the ‘updateMulti’ method on the Mongo Template class

Update Multi

```
Query airportsInRomania = Query.query(
                Criteria.where(“country").is(“Romania"));

Update setWeather = Update.update(“proximityWeather", Weather.CLOUDS);

mongoTemplate.updateMulti(
    airportsInRomania,
    setWeather,
    Airport.class);
```

Update First

```
Query airportsInRomania = Query.query(
                Criteria.where(“country").is(“Romania"));

Update setWeather = Update.update(“proximityWeather", Weather.CLOUDS);

mongoTemplate.updateFirst(
    airportsInRomania,
    setWeather,
    Airport.class);
```

When using update* methods, only the fields defined in the Update definition are affected, not the whole document.

Deleting Documents

Delete Types

Delete single

Delete many

Delete all

```
Airport istanbul = …   // fetch from database
mongoTemplate.remove(istanbul)
```

Delete Single

CODE	COUNTRY	CAPACITY
CDG	France	12000
NIC	France	1300
IST	Turkey	11000

```
Airport istanbul = …   // fetch from database
mongoTemplate.remove(istanbul)
```

Delete Single

CODE	COUNTRY	CAPACITY
CDG	France	12000
NIC	France	1300

```
Query franceAirports = Query.query(
    Criteria.where("country").is("France")
);

mongoTemplate.findAllAndRemove(franceAirports, Airport.class);
```

Delete Many

CODE	COUNTRY	CAPACITY
CDG	France	12000
NIC	France	1300
IST	Turkey	11000

```
Query franceAirports = Query.query(
    Criteria.where("country").is("France")
);

mongoTemplate.findAllAndRemove(franceAirports, Airport.class);
```

Delete Many

CODE	COUNTRY	CAPACITY
IST	Turkey	11000

```
Query all = Query.query();

mongoTemplate.findAllAndRemove(all, Airport.class);
```

Delete All

CODE	COUNTRY	CAPACITY
CDG	France	12000
NIC	France	1300
IST	Turkey	11000

```
Query all = Query.query();
mongoTemplate.findAllAndRemove(all, Airport.class);
```

Delete All

CODE	COUNTRY	CAPACITY

```
mongoTemplate.dropCollection(Airport.class);
```

Drop Collection

Using Mongo Converters

Mongo Converter

Is a feature used for mapping all Java types to/from DBObjects when storing or retrieving these objects.

Type Conversion

Java	Mongo DB
UUID/String	Object ID
String	String
int	Number
double	Number
boolean	Boolean
Object	Object (embedded)

But what if I want to change the way this conversion is taking place?

Mongo Converter Creation Process

Create write converter (from Java type to Mongo type)

Create read converter (from Mongo type to Java type)

Register converters as a Spring bean

```
@Document class Person{
    @Id private String id;
    private String name;
    private Address address;
}

class Address{
    private String city;
    private String country;
}
```

```
"_id": "6a791adf-15d9-..",

"name": "John Doe",

"address" : {
    "city": "Paris",
    "country": "France"
}
```

```
@Document class Person{
    @Id private String id;
    private String name;
    private Address address;
}

class Address{
    private String city;
    private String country;
}
```

“_id”: “6a791adf-15d9-..”,

“name”: “John Doe”,

“address” : “Paris,France”

Implement the Mongo Converters

AddrWriteConverter.java

```
public class AddrWriteConverter
    implements Converter<Address, String> {

    @Override
    public String convert(Address address) {
        return address.getCity() + ","
                + address.getCountry();
    }
}
```

AddrReadConverter.java

```
public class AddrReadConverter
    implements Converter<String, Address> {

    @Override
    public Address convert(String s) {
        String[] parts = s.split(",");
        return new Address(s[0],s[1]);
    }
}
```

Register the Custom Converters

AirportApplication.java

```
@Bean public MongoCustomConversions customConversions() {
        List<Converter<?, ?>> converters = new ArrayList<>();
        converters.add(new AddrReadConverter()); converters.add(new AddrWriteConverter());
        return new MongoCustomConversions(converters);
    }
```

You can register the converters in any @Configuration class.

In Spring Boot, it can be the main application class; Else define a new class and annotate it with @Configuration

```
@Document class Person{
    @Id private String id;
    private String name;
    private Address address;
}

class Address{
    private String city;
    private String country;
}
```

“_id”: “6a791adf-15d9-..”,

“name”: “John Doe”,

“address” : “Paris,France”

Demo

Implementing insert, update & delete operations

- Single operations
- Batch operations
- Custom converters

Summary

How to insert, update and remove documents in various ways.

The difference between save and insert

How and when to use custom converters for serialization/deserialization of documents

Mongo Template: Great Flexibility

- Execute complex queries
- Insert, update and delete documents in various ways
- Manipulate collections

Flexibility usually comes at a price. For MongoTemplate the price is giving up type safety.