Kotlin MongoDB Tutorial

This guide demonstrates how to use MongoDB with Kotlin for basic CRUD operations (Create, Read, Update, Delete), querying, and searching. We'll use the `KMongo` library, a Kotlin idiomatic MongoDB driver.

Prerequisites

1. **MongoDB installed** on your machine or use a cloud service like MongoDB Atlas.
2. **Kotlin project** setup with `KMongo` dependency.

Step 1: Add KMongo to your project

In your `build.gradle.kts` file, add the following dependency:

```
dependencies {
    implementation("org.litote.kmongo:kmongo:4.7.1") // Latest version
}
```

Step 2: Create a simple Kotlin data class

For this example, we'll work with a simple `User` class.

```
data class User(val name: String, val age: Int, val email: String)
```

Step 3: Setup MongoDB connection

Connect to MongoDB using the `KMongo` driver.

```
import org.litote.kmongo.KMongo
import org.litote.kmongo.getCollection

// Create a connection to the database
val client = KMongo.createClient() // Use KMongo to create a MongoDB client
val database = client.getDatabase("demo_db") // Connect to the demo_db database
val userCollection = database.getCollection<User>() // Access the User collection
```

Step 4: Insert documents (Create)

You can insert new documents into your MongoDB collection using the `insertOne()` function.

```
fun addUser() {
    val user = User("John Doe", 25, "john@example.com")
    userCollection.insertOne(user)
    println("User inserted: $user")
}

addUser()
```

Step 5: Read documents (Find / Query)

To read or query documents, use `find()` or `findOne()`.

Find all users:

```
fun getAllUsers() {
    val users = userCollection.find().toList()
    println("All users: $users")
}

getAllUsers()
```

Find a single user by field:

```
fun findUserByName(name: String) {
    val user = userCollection.findOne("{name: '$name'}")
    println("User found: $user")
}

findUserByName("John Doe")
```

Step 6: Update documents

You can update documents using the `updateOne()` function.

```
fun updateUserEmail(name: String, newEmail: String) {
    val result = userCollection.updateOne("{name: '$name'}", "{\$set: {email: '$newEmai
    if (result.matchedCount > 0) {
        println("User updated successfully.")
    } else {
        println("No user found with the name $name")
    }
}

updateUserEmail("John Doe", "newjohn@example.com")
```

Step 7: Delete documents

You can delete documents using the `deleteOne()` or `deleteMany()` functions.

```
fun deleteUser(name: String) {
    val result = userCollection.deleteOne("{name: '$name'}")
    if (result.deletedCount > 0) {
        println("User deleted.")
    } else {
        println("No user found with the name $name")
    }
}

deleteUser("John Doe")
```

Step 8: More advanced queries (Search)

Find users by a specific condition (age > 20):

```
fun findUsersByAge(age: Int) {
    val users = userCollection.find("{age: {\$gt: $age}}").toList()
    println("Users older than $age: $users")
}

findUsersByAge(20)
```

Use regex to search by name:

```
fun searchUserByName(pattern: String) {
    val users = userCollection.find("{name: {\$regex: '$pattern'}}").toList()
    println("Users matching pattern '$pattern': $users")
}

searchUserByName("Jo.*") // Finds names starting with "Jo"
```

Step 9: Close the database connection

Don't forget to close your MongoDB client once you're done with your operations.

```
client.close()
```

Full Example in One Program

```
import org.litote.kmongo.*

data class User(val name: String, val age: Int, val email: String)

fun main() {
    // Connect to MongoDB
    val client = KMongo.createClient()
    val database = client.getDatabase("demo_db")
    val userCollection = database.getCollection<User>()

    // Insert a new user
    val user = User("John Doe", 25, "john@example.com")
    userCollection.insertOne(user)

    // Find all users
    val users = userCollection.find().toList()
    println("All users: $users")

    // Find a user by name
    val foundUser = userCollection.findOne("{name: 'John Doe'}")
    println("User found: $foundUser")

    // Update user's email
    userCollection.updateOne("{name: 'John Doe'}", "{\$set: {email: 'newjohn@example.com

    // Delete the user
    userCollection.deleteOne("{name: 'John Doe'}")

    // Close the MongoDB connection
    client.close()
}
```

Explanation:

- **Insert:** Adds a new document to the collection.

- **Find:** Retrieves all documents or specific ones based on a condition.
- **Update:** Modifies fields in existing documents.
- **Delete:** Removes documents from the collection.
- **Query:** Filters documents based on conditions like `age > 20` or regex patterns.

This tutorial should give you a solid foundation to work with MongoDB in Kotlin. You can expand this to include more complex queries, aggregation pipelines, and indexing as you progress!