Practical No.12

Title: Write a program to implement MogoDB database connectivity with PHP/ python/Java Implement Database navigation operations (add, delete,edit etc.) using ODBC/JDBC.

Objective:

To learn MongoDB collections.
To understand the concept of JAVA programming.
To understand connectivity of JAVA with MongoDB.

Requirement:

MongoDB
Java

Theory:

MongoDB is an open-source cross-platform **document database** developed using C++. Some features of MongoDB are:

- High and effective performance
- Easily scalable
- High availability
- It can store high volume of data

It contains data in the form of collections and documents instead of rows and tables. A collection is a set of documents. The collection does not have schemas. It represents data in the form of a hierarchical model with which the storage of arrays and other data structures will be easy.

Components of MongoDB

The essentials components of MongoDB are listed below:

1. **Id**: This field represents a unique field in MongoDB. This field is created by default.
2. **Collection**: It is a set of MongoDB documents. It exists with a single database.
3. **Database**: This is the container for collections. Multiple databases can be stored in a mongoDB server.
4. **Document**: A record in mongoDB is called a document. It contains names and values.
5. **Field**: It is a name-value pair in a document.

Establishing connections to database

For making the connection, you have to mention the database name. MongoDB creates a database by default if no name is mentioned.

1. Firstly, import the required libraries for establishing the connection.

2. Here, “**MongoClient**” is used to create the client for the database.
3. “**MongoCredential**” is used for creating the credentials.
4. And finally, to access the database “**MongoDatabase**” is used.
5. Username will be: “**GFGUser**” and the database name will be “**mongoDb**“.

Creating a MongoDb collection:

To create a collection **com.mongodb.client.MongoDatabase class** and **createCollection() method** is used. Here, “database.createCollection()” creates a collection named as “GFGCollection”.

```
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
database.createCollection("sampleCollection");
  System.out.println("Collection created successfully");
```

Getting a Collection

To get/select a collection from the database, getCollection() method of com.mongodb.client.MongoDatabase class is used.Inserting Values into MongoDb**.**

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
// Creating a collection
      System.out.println("Collection created successfully");
      // Retrieving a collection
      MongoCollection<Document> collection = database.getCollection("myCollection");
      System.out.println("Collection myCollection selected successfully");
```

Insert a Document

Only a document type of data can be inserted a MongoDB. Therefore, either we can create a document with the values to be inserted using **append () method** or pass a document directly into the MongoDB using **.insert ()** method. Here, first, we created a new document as “title” and then append the “about” section. Then, we have given the respective values to the documents. To insert a document into MongoDB, **insert ()** method of **com.mongodb.client.MongoCollection** class is used.

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

	// Creating a Mongo client
	MongoClient mongo = new MongoClient( "localhost" , 27017 );

	// Accessing the database
	MongoDatabase database = mongo.getDatabase("myDb");

	// Creating a collection
	database.createCollection("sampleCollection");
	System.out.println("Collection created successfully");

	// Retrieving a collection
	MongoCollection<Document> collection =
database.getCollection("sampleCollection");
	System.out.println("Collection sampleCollection selected successfully");
	Document document = new Document("title", "MongoDB")
	.append("description", "database")
	.append("likes", 100)
	.append("url", "http://www.tutorialspoint.com/mongodb/")
	.append("by", "tutorials point");

	//Inserting document into the collection
	collection.insertOne(document);
	System.out.println("Document inserted successfully");
}
```

Displaying the list of all Documents

For displaying all documents of collection, **find ()** method is used. Here, the database has two documents namely “document” and “document1”, which are retrieved using find () method. We use an iterator since it will iterate over each document present in the list and display it to us.

Dropping of a Collection

To drop a collection from a database, you need to use the **drop()** method of the **com.mongodb.client.MongoCollection** class.

```
// Creating a collection
    System.out.println("Collections created successfully");
    // Retrieving a collection
    MongoCollection<Document> collection = database.getCollection("sampleCollection");
    // Dropping a Collection
    collection.drop();
    System.out.println("Collection dropped successfully");
```

Displaying all the collections

For displaying the list of all collections, **listCollectionNames ()** method is used. Here, we iterate over all the collections we created with the help of "for ()" statement. Database. ListCollectionNames () is used to display the list of all collections present in the database.

```
// Accessing the database
    MongoDatabase database = mongo.getDatabase("myDb");
    System.out.println("Collection created successfully");
    for (String name : database.listCollectionNames()) {
      System.out.println(name);
```

Connection Steps:

Before you start using MongoDB in your Java programs, you need to make sure that you have MongoDB CLIENT and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now, let us check how to set up MongoDB CLIENT.

- You need to download the jar **mongodb-driver-3.11.2.jar and its dependency mongodb-driver-core-3.11.2.jar.**Make sure to download the latest release of these jar files.
- You need to include the downloaded jar files into your class path.

1. Connect to MongoDB.
2. View a List of Connection Options.
3. Specify Connection Behavior with the MongoClient Class.
4. Enable Network Compression.
5. Enable TLS/SSL on a Connection.
6. Connect to MongoDB Using a JNDI Datasource.

Conclusion:

In this way we have successfully connected Java and MongoDB.