MongoDB - Java

In this chapter, we will learn how to set up MongoDB JDBC driver.

Installation

Before you start using MongoDB in your Java programs, you need to make sure that you have MongoDB JDBC driver 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 JDBC driver.

- 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 classpath.

Connect to Database

To connect database, you need to specify the database name, if the database doesn't exist then MongoDB creates it automatically.

Following is the code snippet to connect to the database −

```
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class ConnectToDB {

   public static void main( String args[] ) {

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

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");
      System.out.println("Credentials ::"+ credential);
   }
}
```

Now, let's compile and run the above program to create our database myDb as shown below.

```
$javac ConnectToDB.java
$java ConnectToDB
```

On executing, the above program gives you the following output.

```
Connected to the database successfully
Credentials ::MongoCredential{
   mechanism = null,
   userName = 'sampleUser',
   source = 'myDb',
   password = <hidden>,
   mechanismProperties = {}
}
```

Create a Collection

To create a collection, **createCollection()** method of **com.mongodb.client.MongoDatabase** class is used.

Following is the code snippet to create a collection −

```
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class CreatingCollection {

   public static void main( String args[] ) {

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

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

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

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

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection created successfully
```

Getting/Selecting a Collection

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

Following is the program to get/select a collection −

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class selectingCollection {

   public static void main( String args[] ) {

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

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

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

      // 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");
   }
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection created successfully
Collection myCollection selected successfully
```

Insert a Document

To insert a document into MongoDB, **insert()** method of **com.mongodb.client.MongoCollection** class is used.

Following is the code snippet to insert a document −

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
public class InsertingDocument {
        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");
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection sampleCollection selected successfully
Document inserted successfully
```

Retrieve All Documents

To select all documents from the collection, **find()** method of **com.mongodb.client.MongoCollection** class is used. This method returns a cursor, so you need to iterate this cursor.

Following is the program to select all documents −

```
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class RetrievingAllDocuments {
        public static void main( String args[] ) {

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

                // Creating Credentials
                MongoCredential credential;
                credential = MongoCredential.createCredential("sampleUser", "myDb", "password".t
                System.out.println("Connected to the database successfully");

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

                // Retrieving a collection
                MongoCollection<Document> collection = database.getCollection("sampleCollection'
                System.out.println("Collection sampleCollection selected successfully");
                Document document1 = new Document("title", "MongoDB")
                .append("description", "database")
                .append("likes", 100)
                .append("url", "http://www.tutorialspoint.com/mongodb/")
                .append("by", "tutorials point");
                Document document2 = new Document("title", "RethinkDB")
                .append("description", "database")
                .append("likes", 200)
                .append("url", "http://www.tutorialspoint.com/rethinkdb/")
                .append("by", "tutorials point");
                List<Document> list = new ArrayList<Document>();
                list.add(document1);
                list.add(document2);
                collection.insertMany(list);
                // Getting the iterable object
                FindIterable<Document> iterDoc = collection.find();
                int i = 1;
                // Getting the iterator
                Iterator it = iterDoc.iterator();
                while (it.hasNext()) {
                        System.out.println(it.next());
                        i++;
                }
        }
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection sampleCollection selected successfully
Document{{_id=5dce4e9ff68a9c2449e197b2, title=MongoDB, description=database, likes=100, url=http
Document{{_id=5dce4e9ff68a9c2449e197b3, title=RethinkDB, description=database, likes=200, url=ht
```

Update Document

To update a document from the collection, **updateOne()** method of **com.mongodb.client.MongoCollection** class is used.

Following is the program to select the first document −

```java
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Updates;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class UpdatingDocuments {

   public static void main( String args[] ) {

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

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");
      // Retrieving a collection
      MongoCollection<Document> collection = database.getCollection("sampleCollection");
      System.out.println("Collection myCollection selected successfully");
      collection.updateOne(Filters.eq("title", 1), Updates.set("likes", 150));
      System.out.println("Document update successfully...");

      // Retrieving the documents after updation
      // Getting the iterable object
      FindIterable<Document> iterDoc = collection.find();
      int i = 1;
      // Getting the iterator
      Iterator it = iterDoc.iterator();
      while (it.hasNext()) {
         System.out.println(it.next());
         i++;
      }
   }
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection myCollection selected successfully
Document update successfully...
Document{{_id=5dce4e9ff68a9c2449e197b2, title=MongoDB, description=database, likes=100, url=http
Document{{_id=5dce4e9ff68a9c2449e197b3, title=RethinkDB, description=database, likes=200, url=ht
```

Delete a Document

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

Following is the program to delete a document −

```
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class DeletingDocuments {

   public static void main( String args[] ) {

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

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");
      // Retrieving a collection
      MongoCollection<Document> collection = database.getCollection("sampleCollection");
      System.out.println("Collection sampleCollection selected successfully");
      // Deleting the documents
      collection.deleteOne(Filters.eq("title", "MongoDB"));
      System.out.println("Document deleted successfully...");

      // Retrieving the documents after updation
      // Getting the iterable object
      FindIterable<Document> iterDoc = collection.find();
      int i = 1;
      // Getting the iterator
      Iterator it = iterDoc.iterator();
      while (it.hasNext()) {
         System.out.println(it.next());
         i++;
      }
   }
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection sampleCollection selected successfully
Document deleted successfully...
Document{{_id=5dce4e9ff68a9c2449e197b3, title=RethinkDB, description=database, likes=200, url=ht
```

Dropping a Collection

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

Following is the program to delete a collection −

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class DropingCollection {

   public static void main( String args[] ) {
      // Creating a Mongo client
      MongoClient mongo = new MongoClient( "localhost" , 27017 );
      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");
      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");

      // 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");
   }
}
```

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection sampleCollection selected successfully
Collection dropped successfully
```

Listing All the Collections

To list all the collections in a database, you need to use the **listCollectionNames()** method of the **com.mongodb.client.MongoDatabase** class.

Following is the program to list all the collections of a database −

```
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class ListOfCollection {

   public static void main( String args[] ) {

      // Creating a Mongo client
      MongoClient mongo = new MongoClient( "localhost" , 27017 );
      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

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

On compiling, the above program gives you the following result −

```
Connected to the database successfully
Collection created successfully
myCollection
myCollection1
myCollection5
```

Remaining MongoDB methods **save(), limit(), skip(), sort()** etc. work same as explained in the subsequent tutorial.