Executing mongo-shell-like-query with mongo Java driver

Describe your read queries the same as you do on the mongo client shell, and the library runs it, for both regular queries as well as use of the Mongo aggregation facility

Example

```
String query = ”db.users.find( { ‘name’ : ‘John’} )”;
MongoQueryParser parser = new MongoQueryParser();
MongoQuery mongoQuery = parser.parse(query, new HashMap());
BaiscDBList results = mongoQuery.execute(mongoDB);
```

Features

1. It supports all kinds of read queries (both find as well as aggregation) using various operators
2. It supports query chaining (chain the use of sort, skip, and limit in any order)
3. It tolerates whitespace as: ”db.users.find({ ‘name’ : ‘John’})” is of course, the same as ”db . users . find ({ ‘name’ : ‘John’ })”
4. It supports parameterized queries, wherein values can be specified at runtime

   ```
   String query = ”db.users.find( { ‘name’ : ‘givenName’} )”
   Map[String,String]  params= new HashMap()
   map.add(“givenName”, “John”)
   MongoQueryParser parser = new MongoQueryParser();

   MongoQuery mongoQuery = parser.parse(query, params);
   BaiscDBList results = mongoQuery.execute(mongoDB);
   ```

5. It supports parameterized queries with strongly typed data (Most of BSON types are supported)

   ```
   String query = ”db.users.find( { salary : ‘sal#Long’} )”
   Map[String,String]  params= new HashMap()
   map.add(“sal”, “5000”)
   MongoQueryParser parser = new MongoQueryParser();
   ```

```
MongoQuery mongoQuery = parser.parse(query, params);
BaiscDBList results = mongoQuery.execute(mongoDB);
```

Query looks like mongo query string. Providing placeholders is very simple and have consistent pattern.
ParameterName#DataType
(supported datatypes are String|Boolean|Integer|Double|ObjectId)

For more examples checkout 'src/test/resources' folder and MongoQueryIntegrationTest
(https://github.com/leenabora/mongo-shell-like-query)

6. It supports both scala and java api

Usage (For Maven Users):

1. Add dependency for mongo-shell-like-queries in your project
   a) maven dependacy

```
<dependencies>
  <dependency>
        <groupId>com.ee.mongo.util</groupId>
        <artifactId>mongo-shell-like-query</artifactId>
        <version>1.0-SNAPSHOT</version>
  </dependency>
</dependencies>

<repositories>
  <repository>
      <id>ee-nexus</id>
      <name>thirdparty</name>
      <url>http://49.248.27.91:9090/nexus/content/repositories/snapshots/</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
  </repository>
</repositories>
```

2. Use library:

a) Java API

```
String query = "db.users.find( { 'role' : 'manager'} )";
DB mongoDB = new MongoClient().getDB("your-db-name");
MongoQuery mongoQuery = new MongoQueryParser().parse(query, new
HashMap<String, String>());
BasicDBList results = mongoQuery.execute(mongoDB);
System.out.println(results.size());
```

b) Scala API

```
val query = "db.users.find( { 'role' : 'manager'} )";
val mongoConn = MongoConnection()
val mongoDB = mongoConn("your-db-name")

val mongoQuery: MongoQuery = parser.parse(query, Map())
val results = mongoQuery.execute(mongoDB)
println(results.size)
```