

**ARTL for Java**

Java Technical Good

**<C0001-MongoDB >**

*Technical Design*

*Feb 2015 version 1*

# Contents

[1. Contents 2](#_Toc411973997)

[2. < C0001-MongoDB > TECHNICAL DESIGN 3](#_Toc411973998)

[ MongoInsertAndSearch.java 3](#_Toc411974000)

[ MongoAuthentication.java 3](#_Toc411974001)

[ MongoInsertDocs.java 4](#_Toc411974002)

[ MongoSaveImage.java 5](#_Toc411974003)

[ MongoFindTest.java 6](#_Toc411974004)

[ MongoFindCriteriaTest.java 6](#_Toc411974005)

[ MongoFindAndModifyTest.java 7](#_Toc411974006)

[ MongoFieldSelectionTest.java 3](#_Toc411974000)

[ MongoDotNotationTest.java 3](#_Toc411974001)

[ MongoSortSkipLimitTest.java 4](#_Toc411974002)

[ MongoUpdateRemoveTest.java 5](#_Toc411974003)

[ MongoSpringIntegration.java 6](#_Toc411974004)

[3. DOCUMENT CONTROL 9](#_Toc411974007)

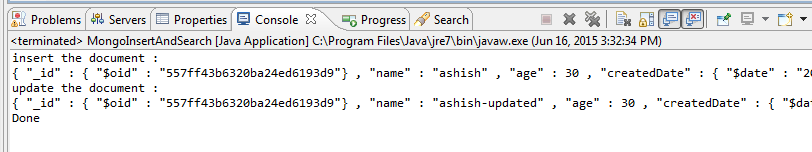
[ Change history 9](#_Toc411974008)

[ Open issues 9](#_Toc411974009)

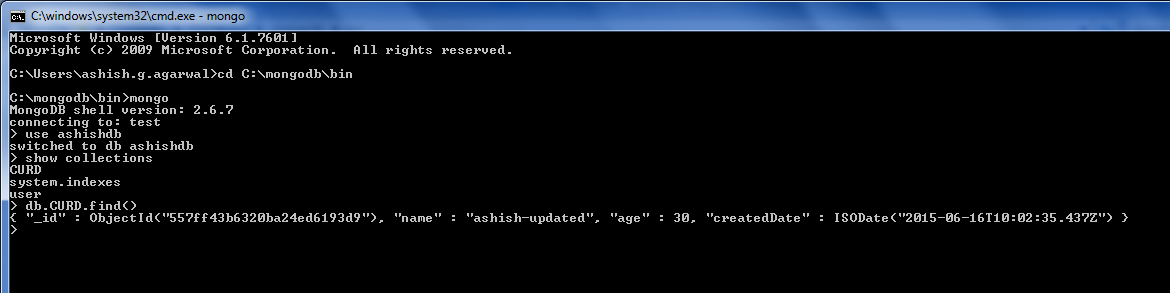
# 2. < C0001-Mongodb > Technical design

## MongoInsertAndSearch.java

This simple example show how to connect, create database, collection and document, save, update, remove, get and display document (data).Please find below output on the console after running the MongoInsertAndSearch.java as java application.



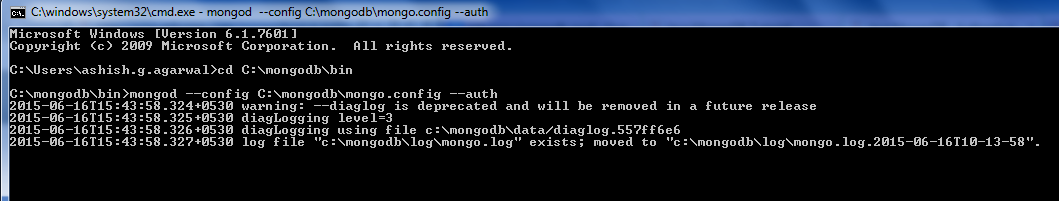
Below snapshot shows the document that is saved into MongoDB.



## MongoAuthentication.java

By default, MongoDB is run in trust environment (authentication with a username and password is NOT required). In this example, we will explain how to start MongoDB in secure mode / enable authentication, and connect with the Java MongoDB driver.

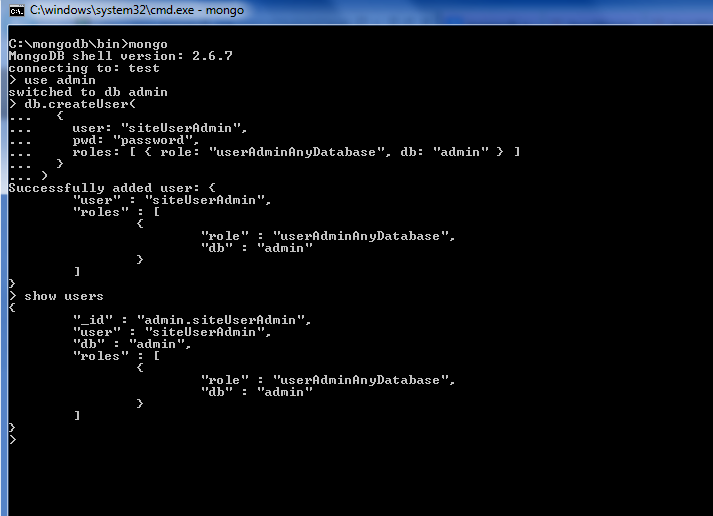
Start MongoDB with --auth option, now, MongoDB need username and password to perform any database / collection operations.



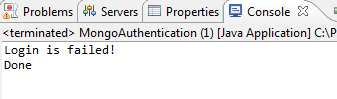
­

Later, we need to connect to the database “ashishdb”, so add a user for testing later.

Now connect to the admin database and create one “siteUserAdmin” user as shown in the below acrennshot.

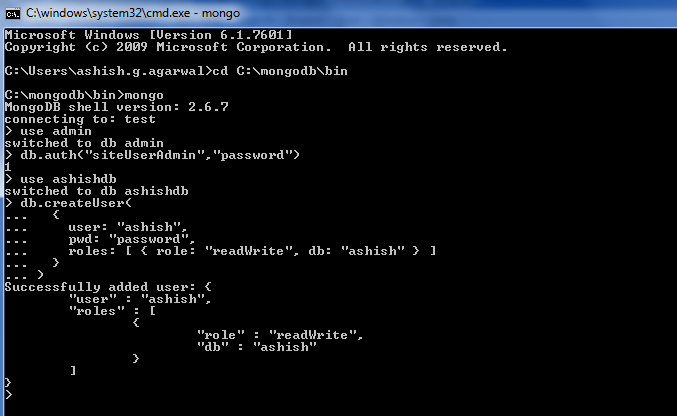


Try to run the example as java application . Following output will come on the console.



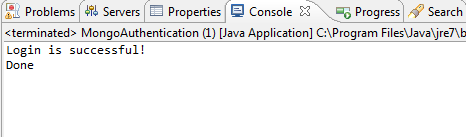
Now connect to the “ashishdb” . Perform the authentication ***db.auth("siteUserAdmin","password")*** (1 means succeed, 0 means failed). Create one user in “ashishdb” .

Below snapshot shows the steps .



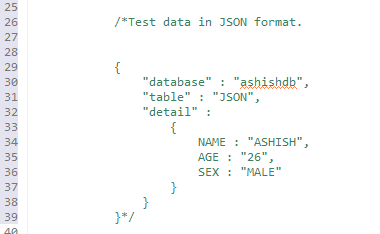
Again run the example as java application . Following output will come on the console. This time login is successful

.

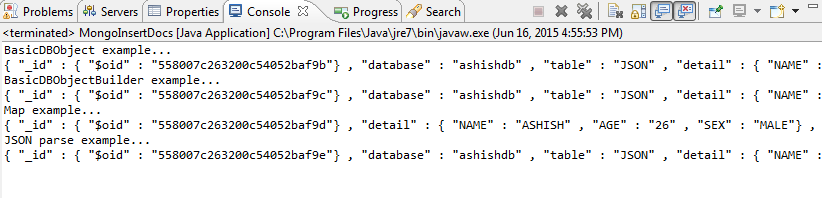


## MongoInsertDocs.java

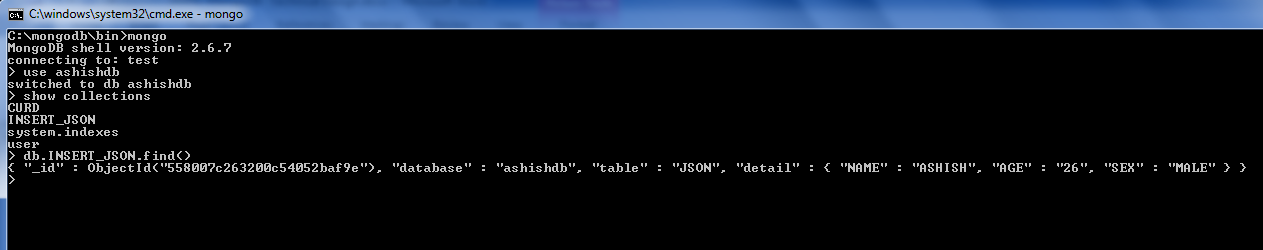
In this example, We have inserted the below JSON data in four ways via Java MongoDB API.



Below is the screenshot of the console output.



Below snapshot shows the document that is saved into MongoDB.

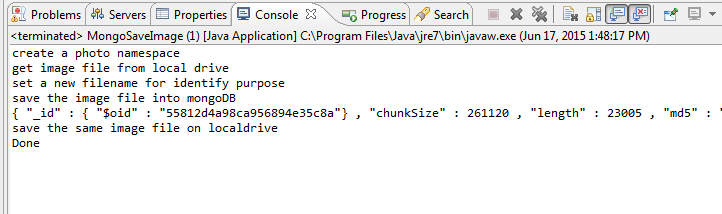


## MongoSaveImage.java

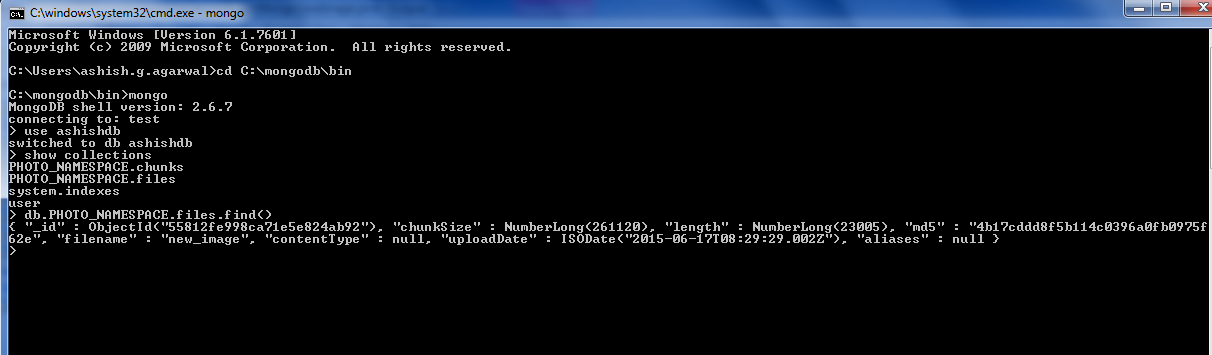
This example explains how to save image file in the MongoDB . We can save image , get the image , print all the saved images , save the image on our local machine from DB and delete the image from the DB .

In this example we have used “**GridFS**” class for creating one name space , “**GridFSInputFile**” class for saving the image under the one name space and “**GridFSDBFile**” class for getting the image file from the same name space .

Below is the console output after running the example.



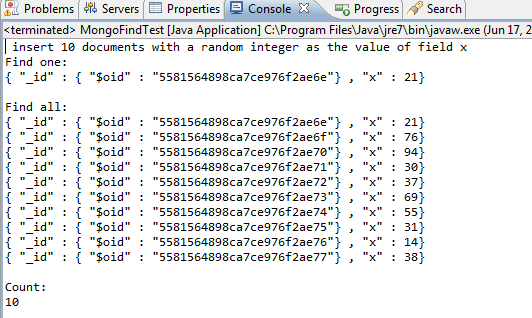
Below snapshot shows the image file that is saved into MongoDB.



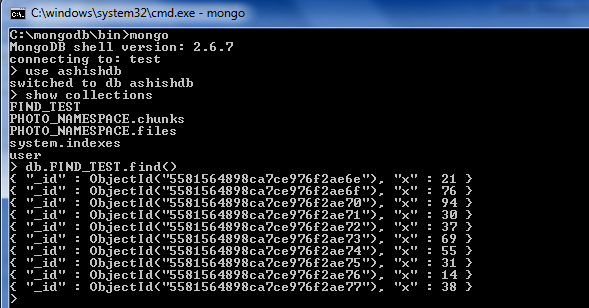
## MongoFindTest.java

This example explains how to insert and find document data in MongoDB.

By running the example below will be console output.



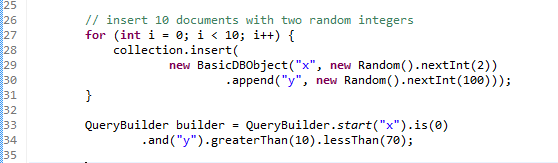
Below snapshot shows the document data with some random values of ‘x’ that is saved into MongoDB.



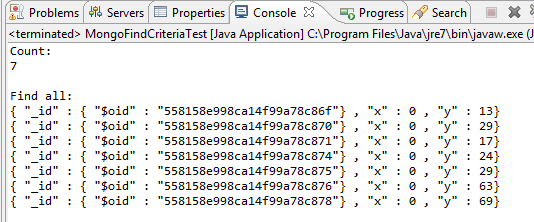
## MongoFindCriteriaTest.java

This example explains how to find the document data based on some criteria from MongoDB.

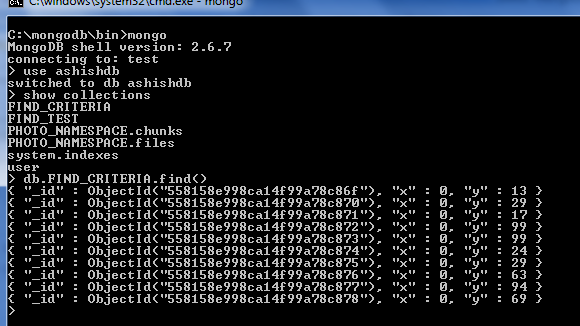
Below is the criteria for inserting and finding the document :



By running the example below will be console output.

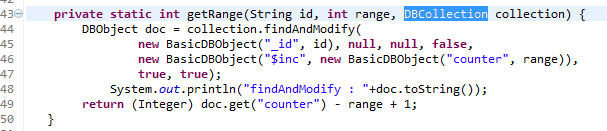


Below snapshot shows the document data with some random values of ‘x’ and ‘y’ that is saved into MongoDB.

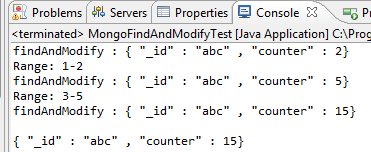


## MongoFindAndModifyTest.java

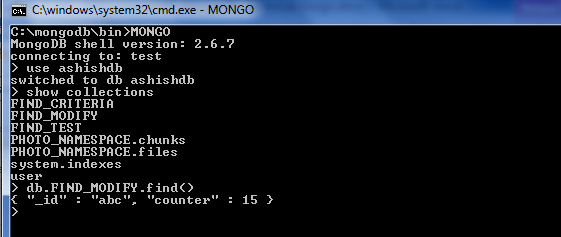
This example explains how to atomically modify and return a single document by using **DBCollection.findAndModify(...)** method. By default, the returned document does not include the modifications made on the update. Below is the code snippet for the same .



By running the example below will be console output.

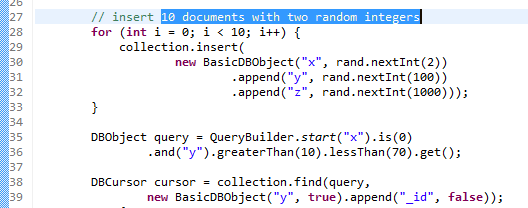


Below snapshot shows the document data with some random values of ‘x’ and ‘y’ that is saved into MongoDB.

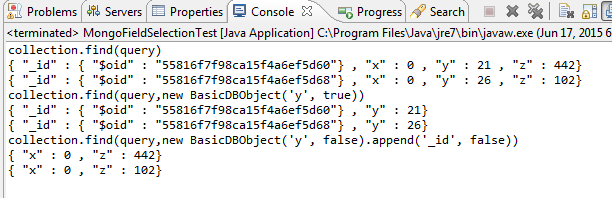


## MongoFieldSelectionTest.java

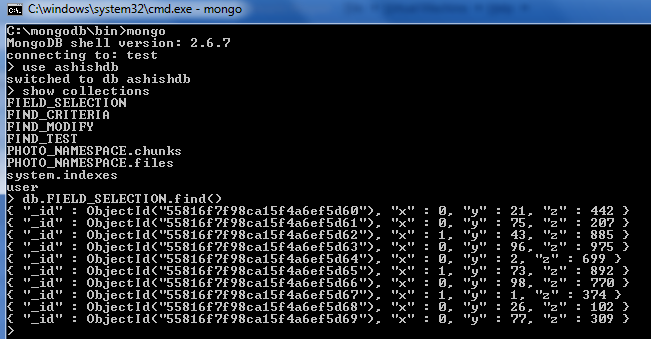
In this example first we will insert 10 documents with two random integers , then select the data with the query builder as per blow selection criteria :



By running the example below will be console output.



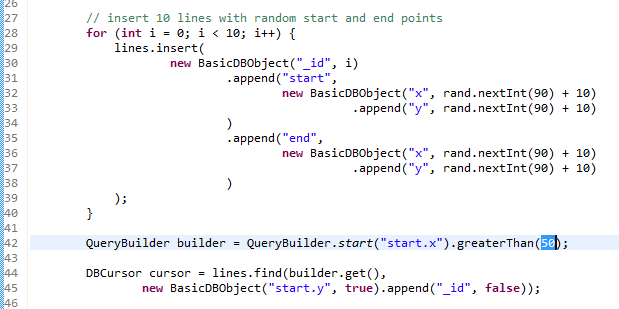
Below snapshot shows the document data with some random values of ‘x’ , ’y’ and ‘z’ that is saved into MongoDB.



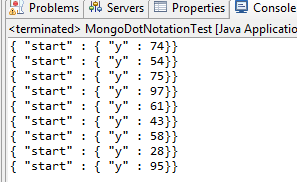
## MongoDotNotationTest.java

This example explains how we can use dot notation for finding some data from MongoDB.

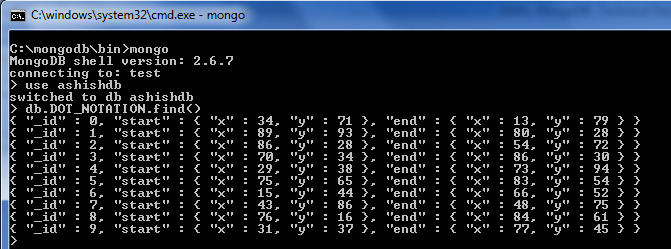
Following is the code snippet for inserting and find the data using dot notation.



By running the example below will be console output.



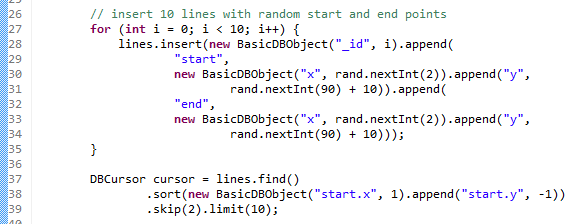
Below snapshot shows the document data with some random values of ‘start.x’ , ’start.y’ and ‘end.x’ , ’end.y’ that is saved into MongoDB.



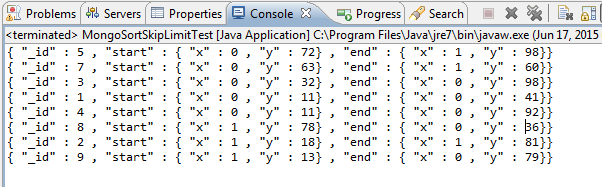
## MongoSortSkipLimitTest.java

This example explains how we can sort , skip and limit the data which we are getting from MongoDB.

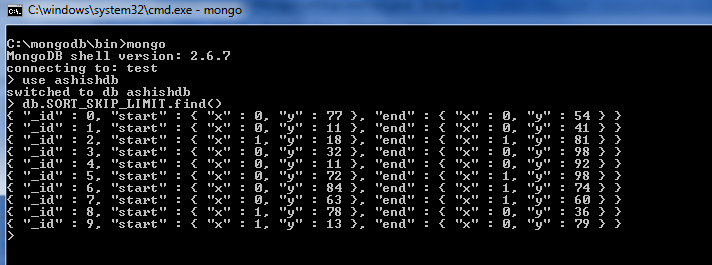
Below is the code snippet for insert , sort , skip and limit the data.



By running the example below will be console output.



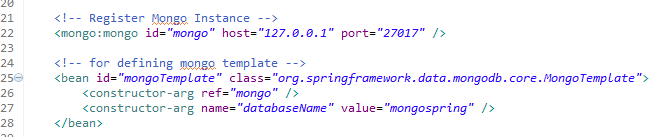
Below snapshot shows the document data with some random values of ‘start.x’ , ’start.y’ and ‘end.x’ , ’end.y’ that is saved into MongoDB.



## MongoSpringIntegration.java

Spring Data provides the developer a set of annotations that could be used for mapping the Business Domain classes into MongoDB collections. MongoDB organized its data through documents; those documents are organized in collections, which are arbitrary containers for a set of documents.

Mongo Template is the central class of the spring’s Mongo DB support providing a rich feature set to interact with the database. We have used below configuration for initializing mongo Template.

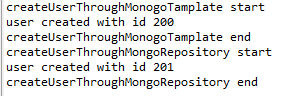


By providing these mentioned lines, you have the ability to locate mongo Template bean from the Spring Context that already initialized.

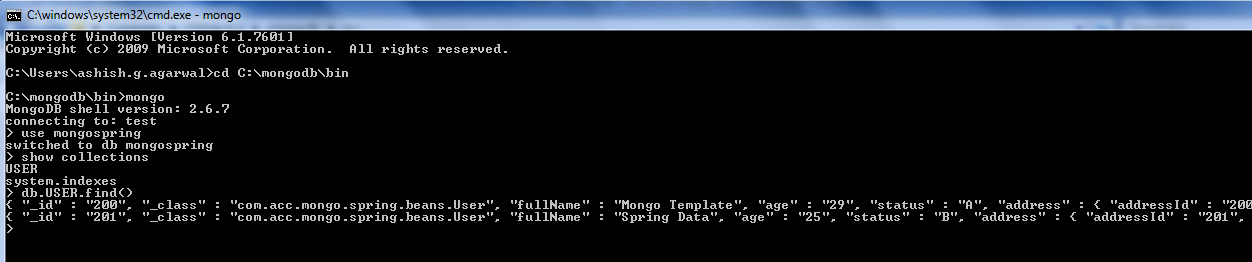
Regardless of using Mongo Template that already provided by spring itself, Spring Data provides a special repository for being used with the MongoDB. Those repositories provide you a full access of inserting, deleting, updating and reading the documents from Mongo DB’s collections without any need for care about the implementation. Mongo namespace provides you an XML element for defining the proper repository for being used later on.



In this example we have inserted the two records (through MonogoTamplate and through MongoRepository) in Mongo DB with the integration of spring data. By running the MongoSpringIntegration.java as java application below output will come on the console.



Below snapshot shows the documents that are saved into MongoDB.



# 3. DOCUMENT CONTROL

## Change history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Doc Version** | **Date** | **Asset Version** | **Author** | **Approver** | **Comment** |
| 1 | 15/06/2015 | 1 | Ashish Agarwal |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Open issues

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Doc Version** | **Date** | **Asset Version** | **Author** | **Action** | **Owner** | **Status** |
|  | mm/dd/yyyy |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |