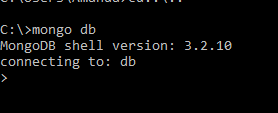
**Mongo Db**

For my project I will be using the MongoDB database which is schema-less so documents can contain fields or types that no other document contains if the developer wishes. I will be showing examples of CRUD operations through the Mongo shell command-line and through a GUI I will develop using Java and the Swing framework as well as the MongoDB Java API. I chose the MongoDB java driver as I could not find any others for the Java language and this seemed the most supported. I will be populating my database with football players from around the world with the fields: name, date of birth, club and achievements with nested fields of club achievements and personal achievements. I hope to gather a better understanding of MongoDB from this project that will benefit my understanding of the different databases available and which databases are best suited to the data I need to store, as well as figuring out the benefits and negatives of MongoDB.

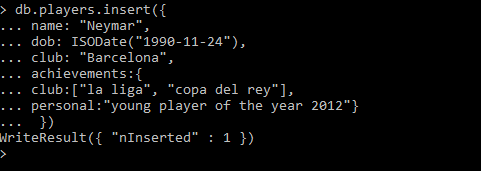
I feel this type of database suits MongoDB as there is potential for a lot of documents considering the vast amount of football players around today, which suits MongoDB as it is known for handling huge amounts of data. MongoDB is very consistent which means that the user always gets the most recent write, which is beneficial to my data as football players can change club regularly and achievements both club and personal can change often and I want the data to always be up to date. MongoDB is also partition tolerant so even if one node goes down all data will still be available to the user.

Create DB



Create Collection

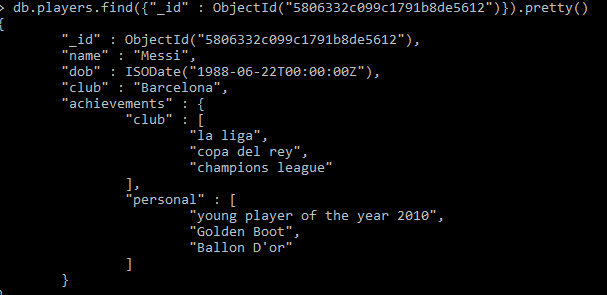
Inserts a player into the database which in turn creates the collection ‘players’



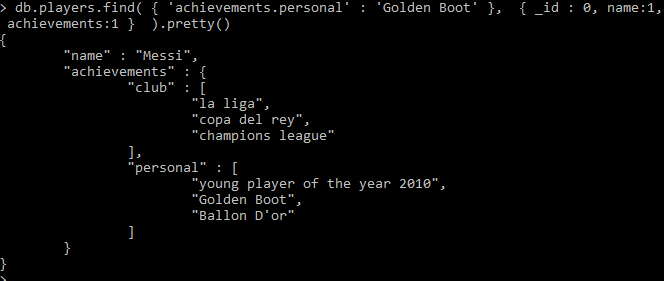
Find



Find by Id

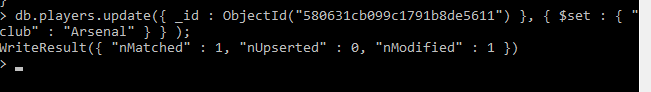


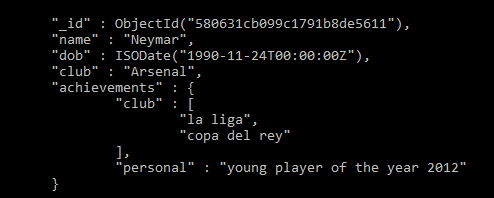
Find by field ‘achievements’



Update

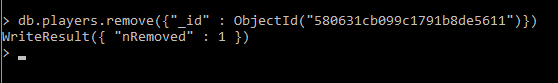
Updates the club field from ‘Barcelona’ to ‘Arsenal’



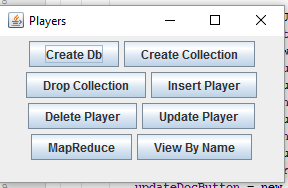


Delete

Can be done by replacing the find() method with remove()

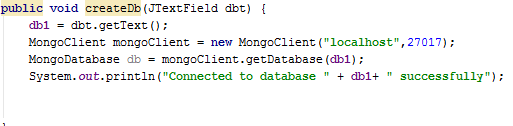


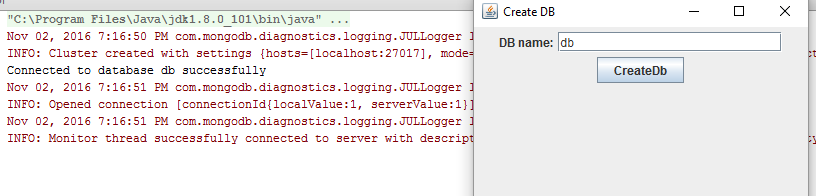
GUI



Create DB

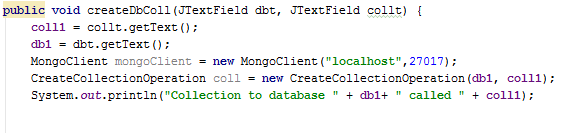
This creates a database called db

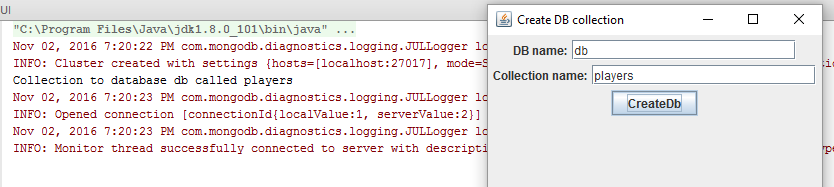




Create Collection

This creates a collection called ‘players’ to the database ‘db’. It also creates the database ‘db’ if it doesn’t already exist.

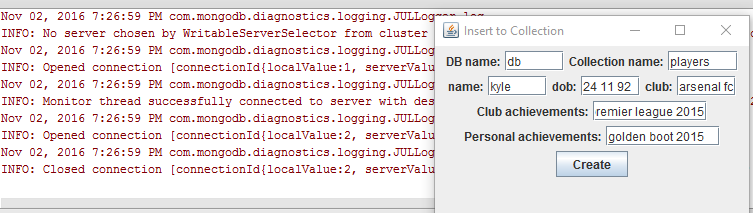


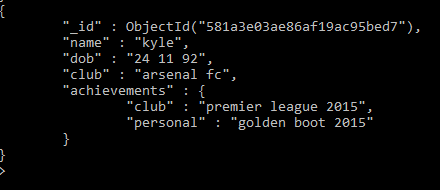


Insert to Collection

This inserts a player to the collection ‘players’.

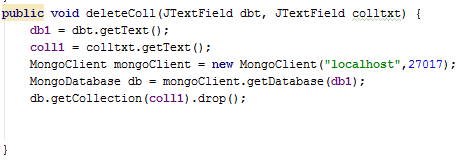


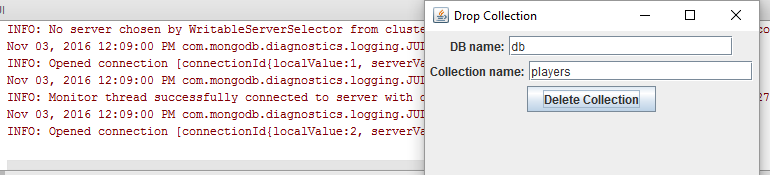


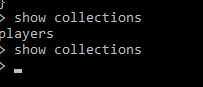


Drop Collection

This code drops a collection and all the documents in it.

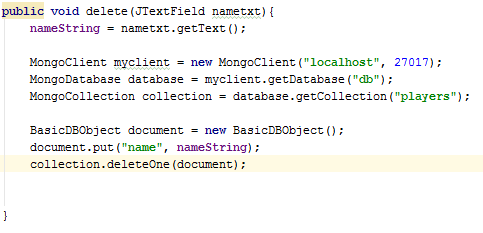


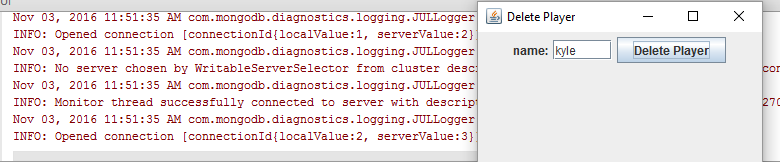




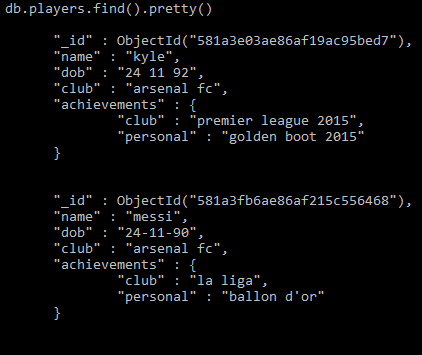
Remove Player from Collection

This removes a player from the collection by there name.

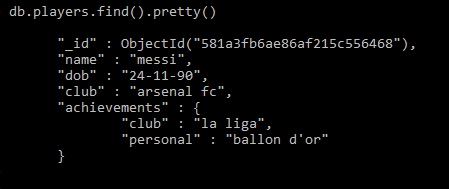




Before delete

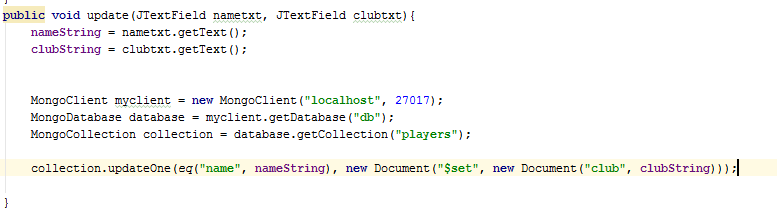


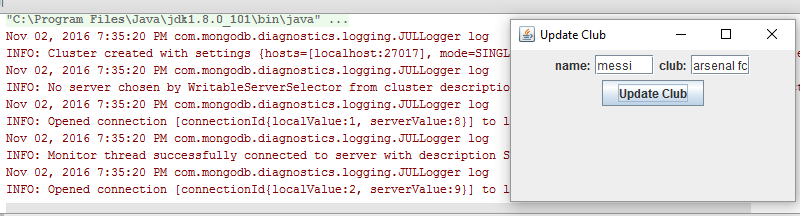
After delete

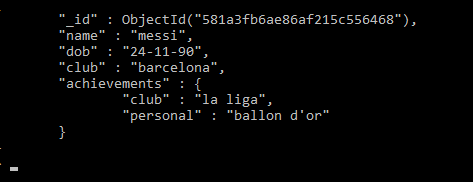


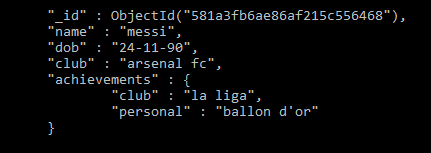
Update Club for a player

This updates a player’s club. In this case it changed messi from Barcelona to arsenal fc.



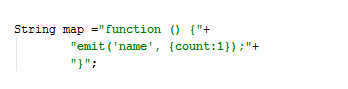






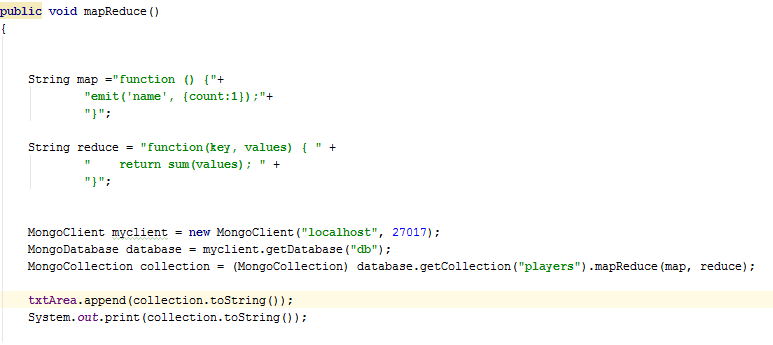
View by Name

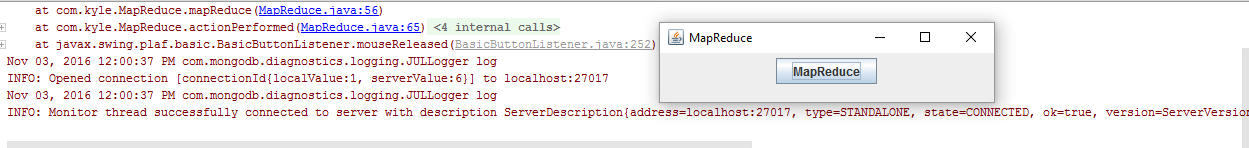
This code views all the documents names.



MapReduce

This code views all the names and gives the number of times each name appears as the value. I had trouble printing the data to the JTextArea.





Conclusion

In conclusion I found MongoDB very easy to use through the command-line, with the commands very easy to follow and remember. I also found the MongoDB java driver very simple to use with most CRUD operations possible with a line or two of code. However due to MongoDB recently upgrading to version 3.0, I ran into a lot of trouble trying to use old and deprecated methods from previous versions. When I realised this I also had trouble finding support online as this version is relatively new and doesn’t yet have the support of older versions.