Advanced Database Programming Portfolio 2 - MongoDB

Niall Stack T00174406

2016

# Q1 - Implementing My Own MongoDB Database

## Introduction:

For the first question of my Advanced Database Programming project part two, I was asked to document the creation of a MongoDB database. I had used CouchDB in the first part so I had gotten used to using a NoSQL database although there was no simple web interface like futon for MongoDB, so everything had to be done via the command line(Although it is possible to download some interfaces they have to be installed and set up unlike futon). I used Mongo to create a car registration database that could be used to store all the cars in a country.

## Vision:

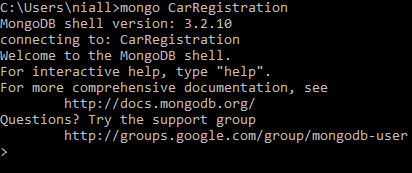
MongoDB is very good at handling big data so I wanted to pick a project that would suit this ability. I decided on choosing a car registration database that could be used for both a relatively small scale (such as for a town) and a large scale (like a national database), taking advantage of Mongo’s scalability. Using Mongo also gives the ability to have fields that are in some document and no others, such as if I needed different fields for an electric car because Mongo does not enforce a schema. MongoDB is also very efficient at ad hoc queries so it is fast at retrieving documents which is very important for a large scale document based application like a car registration database.

## Creating the Database:

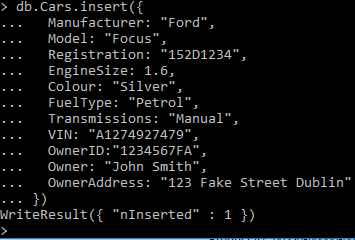
For the car registration database I initially decided to have 12 fields, although I knew it was likely I was going to add more but luckily that is very simple to do with the schema-less MongoDB:

* \_id
* Manufacturer
* Model
* Registration
* EngineSize
* Colour
* FuelType
* Transmissions
* VIN
* OwnerID
* Owner
* OwnerAddress

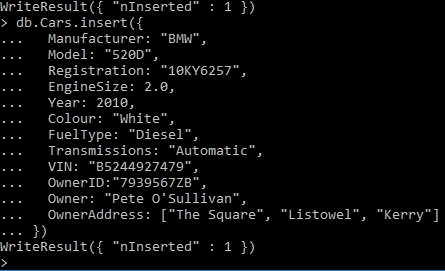
To create the Database, the same command is used as to connect to it which either connects if the database exists or creates a new database with the chosen name and connects to it.



Creating a collection also uses the same command as is used to add a document to a collection which like the above command to connect creates a collection if it does not already exist. The command creates the collection called “Cars” as there was no collection of Cars previously in the database and also inserts the document into it.



I inserted another document but instead of having a single string for address I created an array.

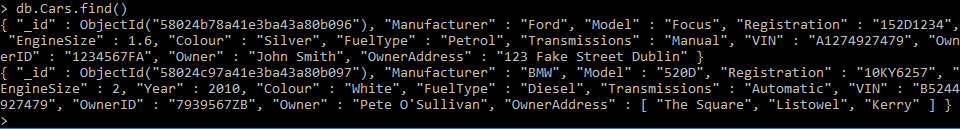


## Reading From the Database:

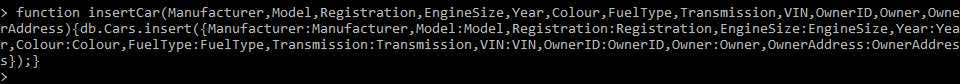
To view all collections in the database just use the simple command – “show collections”.



Then to view all the documents in a collection.



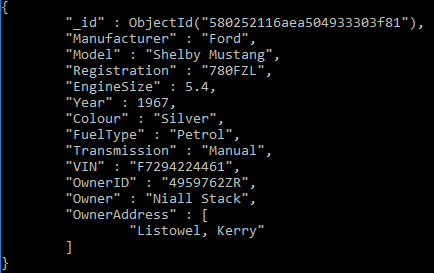
MongoDB also allows for functions to be created that can speed up the population process which is very useful for a big data application.



Using this function I can add a document to the collection with a more simplified insert.



Which could then be seen in the collection.



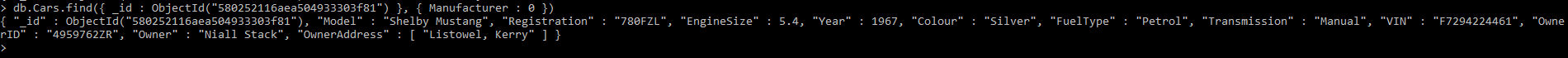
It is also possible make it easier to read add by adding to the find command.



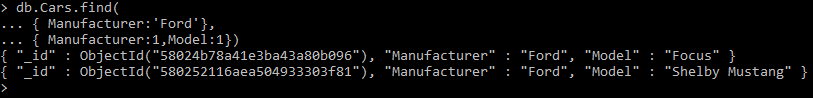
Mongo allows to view just one field along with the ID by setting the specific field to 1.



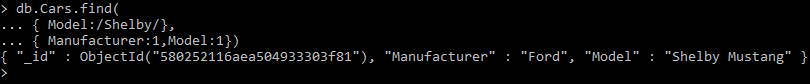
Then by turning changing it to zero you get all fields except that one.



You can also search for exact values in a field.



Also by partial values in a field using “/…/”.

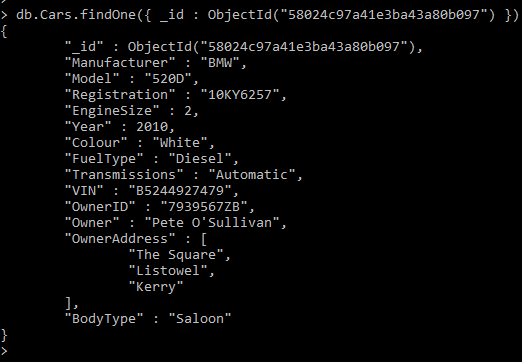


## Updating the Database

In Mongo it is simple to update an existing document for example by adding a field you just need the \_id and then use the $set command to alter the document.



We can check this with the findOne command



Not only can we use $set we can use $inc to increment a number, like the year the car was manufactured in.

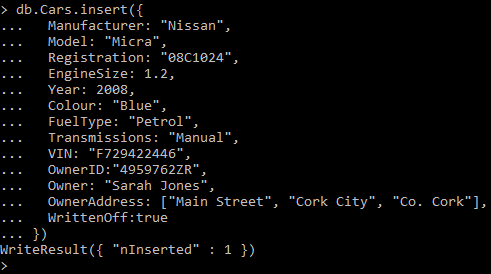


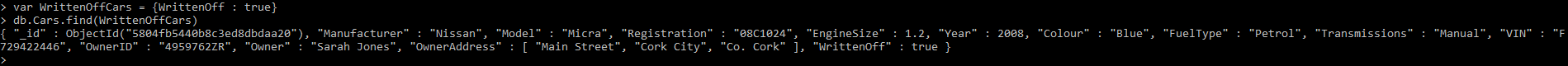
You can do many other operations with the update directives even removing with unset which I used after accidently creating a field called 2010 when I was meant to increment the Year.



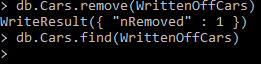
## Deleting from the Database

To delete a document from mongo there is a simple remove function but we can use an $elemMatch to search for certain queries and then remove them based on that. For this I decided to add a field about if a car was written off in a crash when I added a new vehicle.



I then created a “WrittenOffCars” variable that I saved all cars where the field “WrittenOff = true”.

Then to remove all cars that are written off I used the “WrittenOffCars” variable with the remove command.



## Conclusion:

Using Mongo’s CRUD commands I was able to create the document I had envisioned with relative ease and was able to do it quickly enough considering I was doing it all through the command line. Although I didn’t discuss is in the first question as I was not asked to, using map reduces was also very simple and would be of good help when creating a GUI to connect with the car registration database that I created here. MongoDB would definitely be an option I would consider when choosing a database for any future projects.

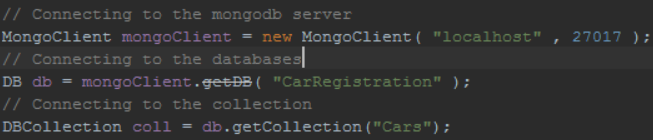
# Q2 - Implementing the Database into a GUI

## Introduction:

For question two I had to connect to the database that I created in question one with a user interface in a language of my choice. CouchDB was not as popular so finding useful resources for it on Java was very difficult so I had chosen Python instead for that but I found creating the UI in that vary tedious and time consuming so I wanted to use something else instead for my Mongo database. Luckily there are better resources online for connecting MongoDB and Java together since MongoDB is a lot more popular. I wanted to use Java as it is the language I am most comfortable with and creating the UI would be made a lot simpler by the NetBeans UI designer so I wouldn’t have to waste too much time trying doing unimportant things like aligning widgets. Although we were asked to compare different drivers, there was one official Java driver for Mongo that was in Mongo’s own documentation so it was obvious to use the official driver as there would be a lot more documentation and help available to me than if I were to use some unofficial driver.

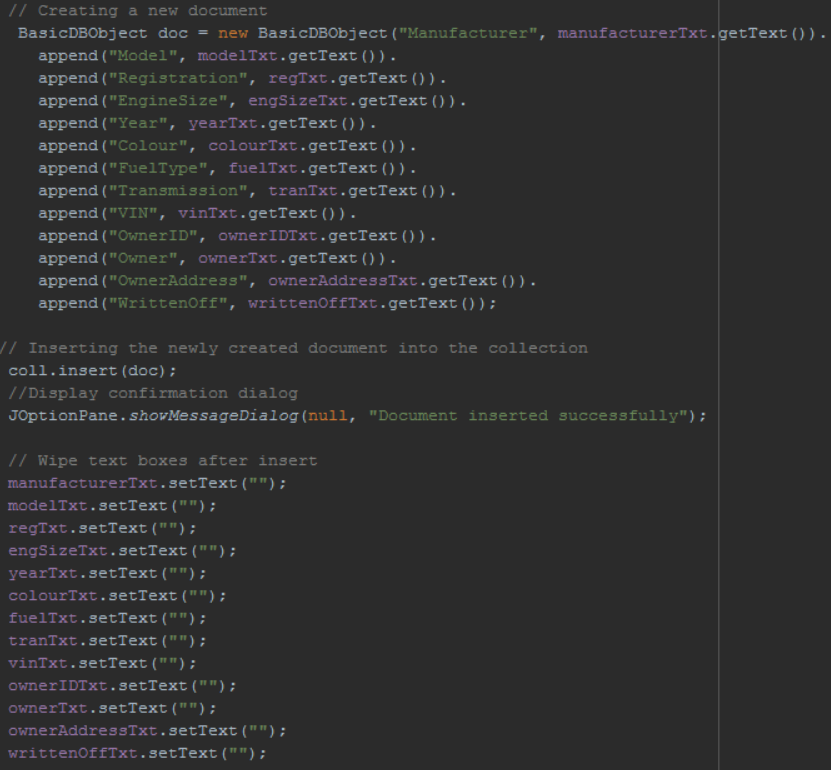
## Code Snippets Showing CRUD:

Connecting to my Mongo database only requires three lines of code. One to connect to the server, another to connect to the actual database and finally one to connect to the collection.



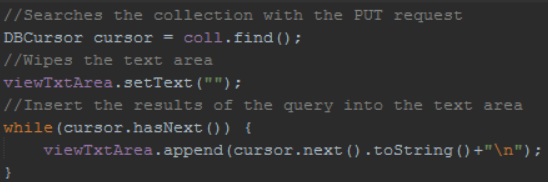
### Create:

To create a new document, I used the “append” method to add the fields and variables which were got from text fields. Then I inserted the data into the collection as a document and to show the user this had worked I had a message dialog box appear to inform them that the insert had been successful and then the text fields get wiped so more documents can be added.

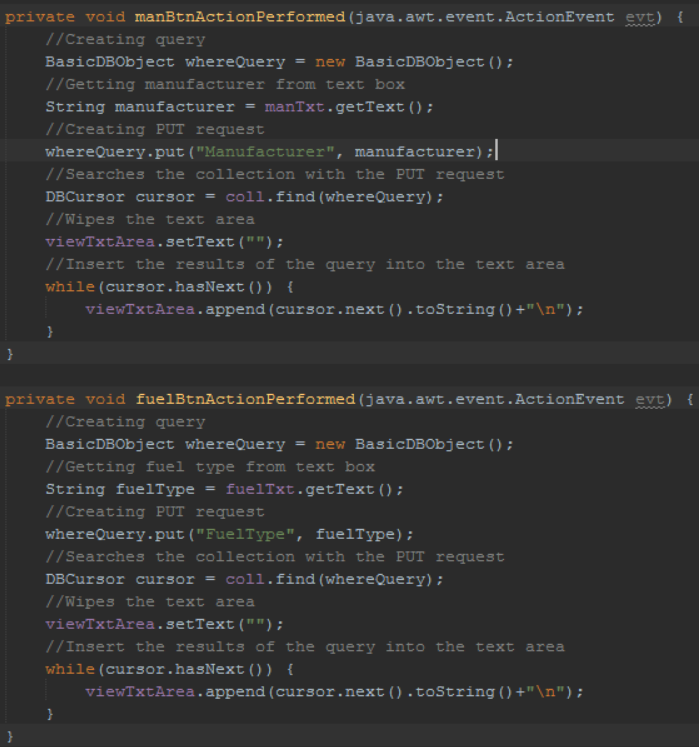


### Read:

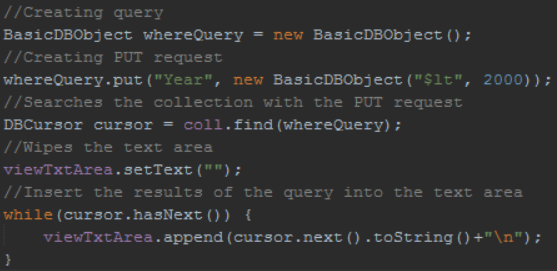
I created many different views of the database, such as a very simple one to view all documents in the collection and print them into a text box.



I also used buttons and text fields on the form to display by keyword like by manufacturer or by fuel type.



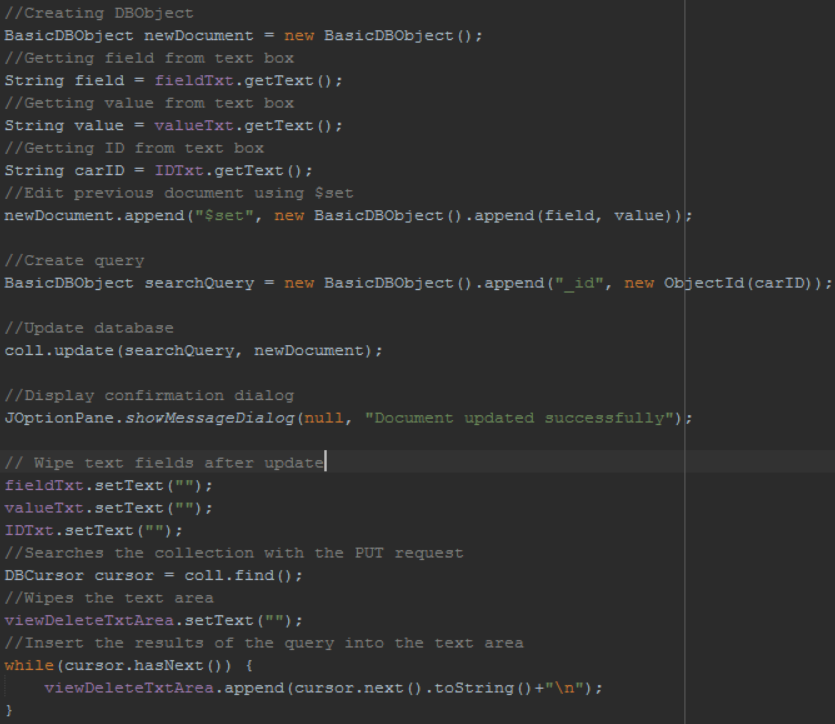
I used the “$lt” and “$gt” commands to display years greater than or less than the year 2000.



I used many other views in my application but they varied little in the code used from the ones I have just displayed.

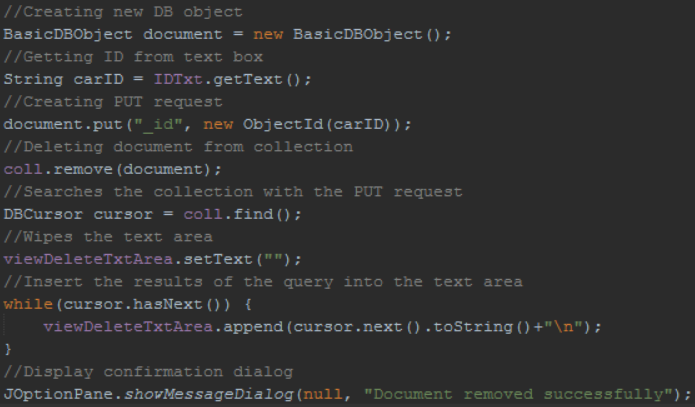
### Update:

Updating is very similar to creating and instead of having to make the user type in all fields again, I allowed them to supply the ID and change a single field of their choice and use the “$set” command to add the change. Then I updated the collection and again added a dialog box to confirm that the update had been successful and wiped the fields for more updates. I also refreshed to textbox displaying all the documents in the collection so the user could see the change they made.



### Delete:

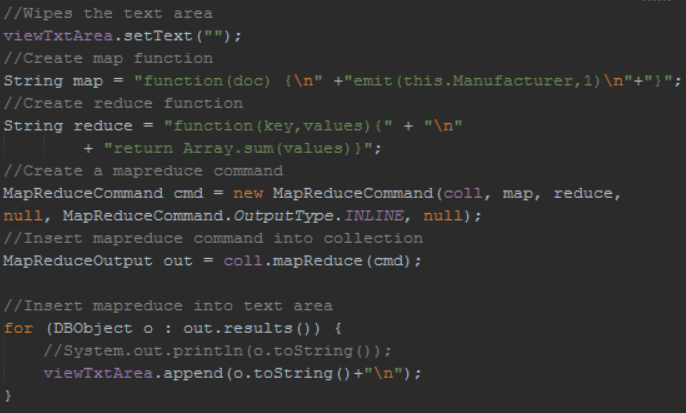
I allowed the user to delete a document by proving the ID and then used removed all documents matching that ID number. Then I wiped the field and displayed a confirmation message confirming the deletion of the document.



There was little code needed for all these CRUD operations showing how simple it is to connect a Java application to a MongoDB database.

## Code Snippet Showing Map Reduce:

Like I did for my CouchDB project I was able to take the JavaScript code used for a map reduce and insert it directly into my code. I could execute the map reduce by a button and then display the results in a textbox.

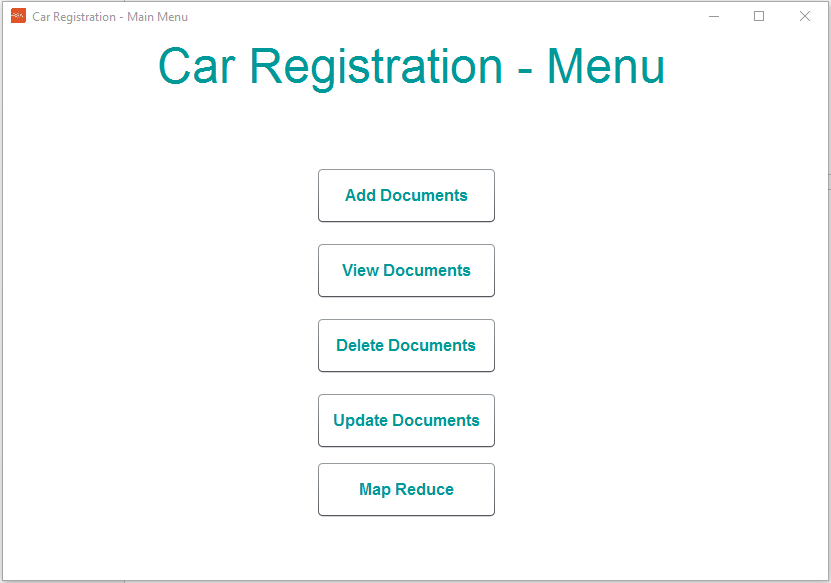


The map function I made emitted the manufacturers and a value of one for each in all my documents and then the reduce function added up all the different manufacturers and summed up the amount of cars in the collection that were created by the same manufacturer.

## User Interface:

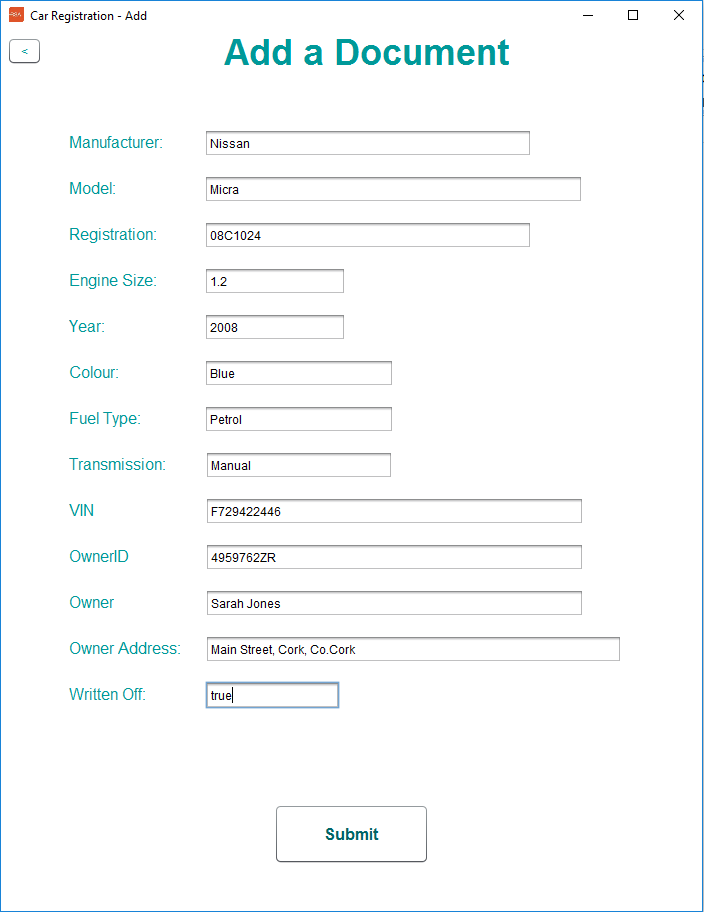
### Main Menu:

I created a basic main menu to help navigate between the different pages.

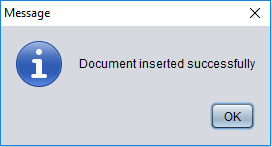


### Create:

Next I have the add documents page



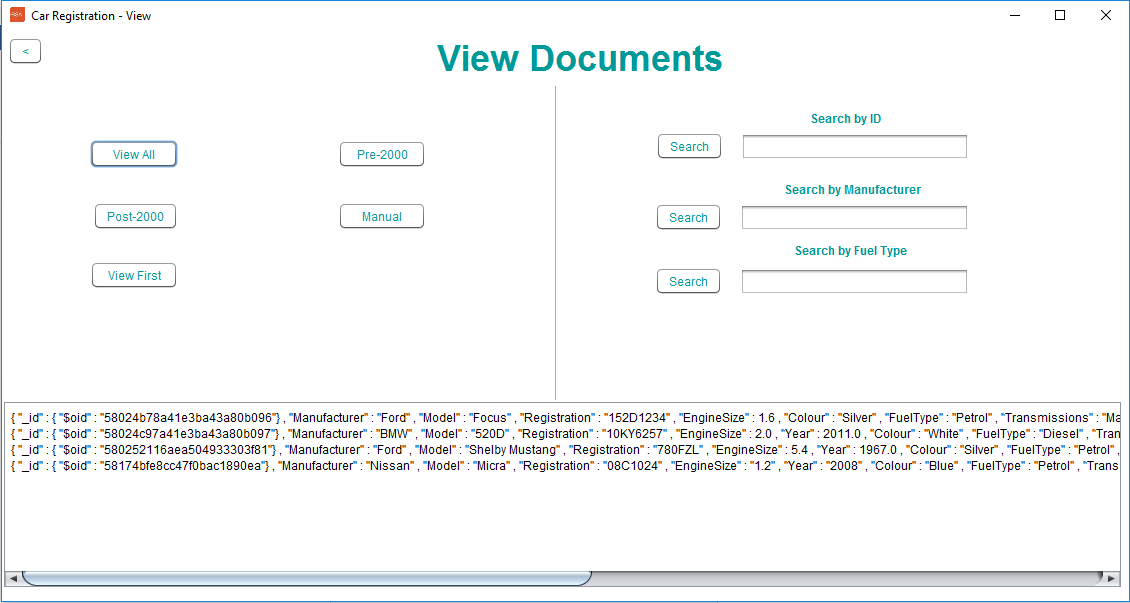
And then the dialog box appears to show that the document has been created successfully



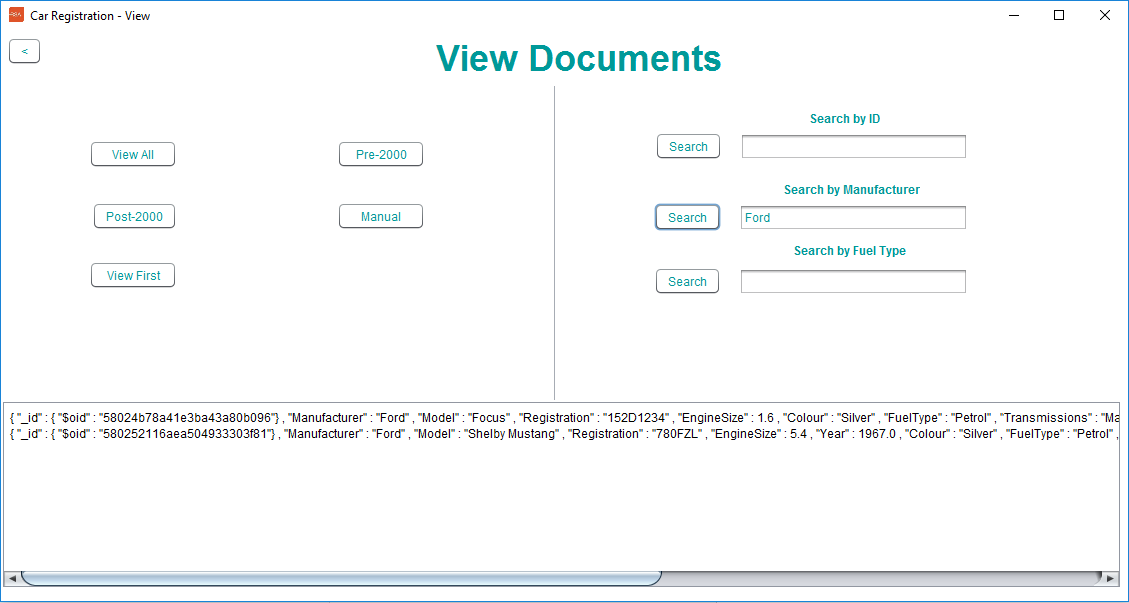
### Read:

The next page was the view documents page where the user was given the option of several different views

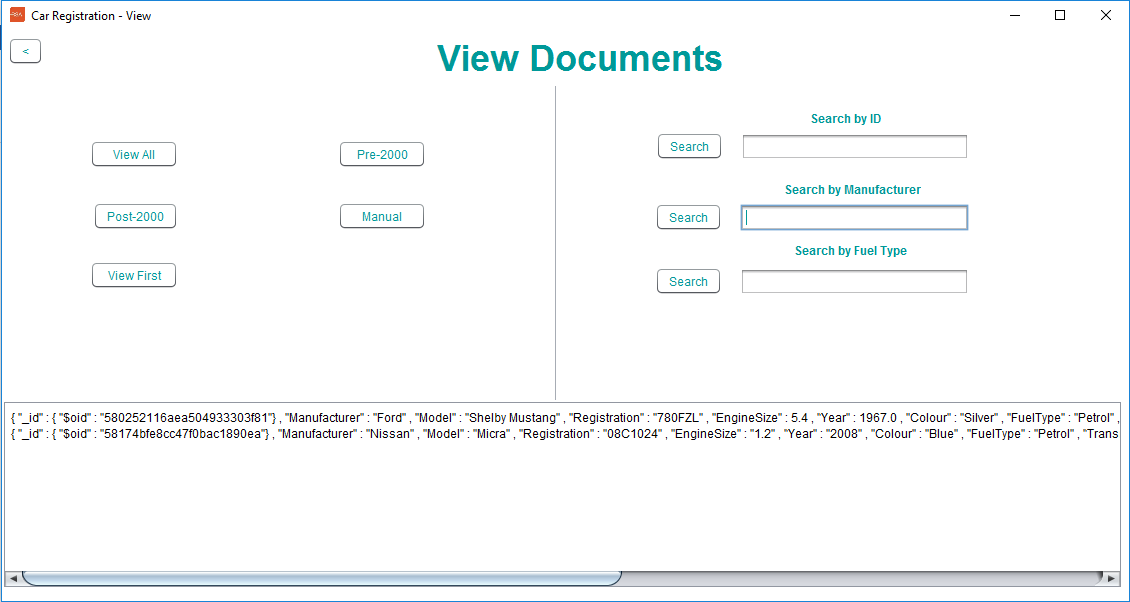
(View All)



(View by Manufacturer)

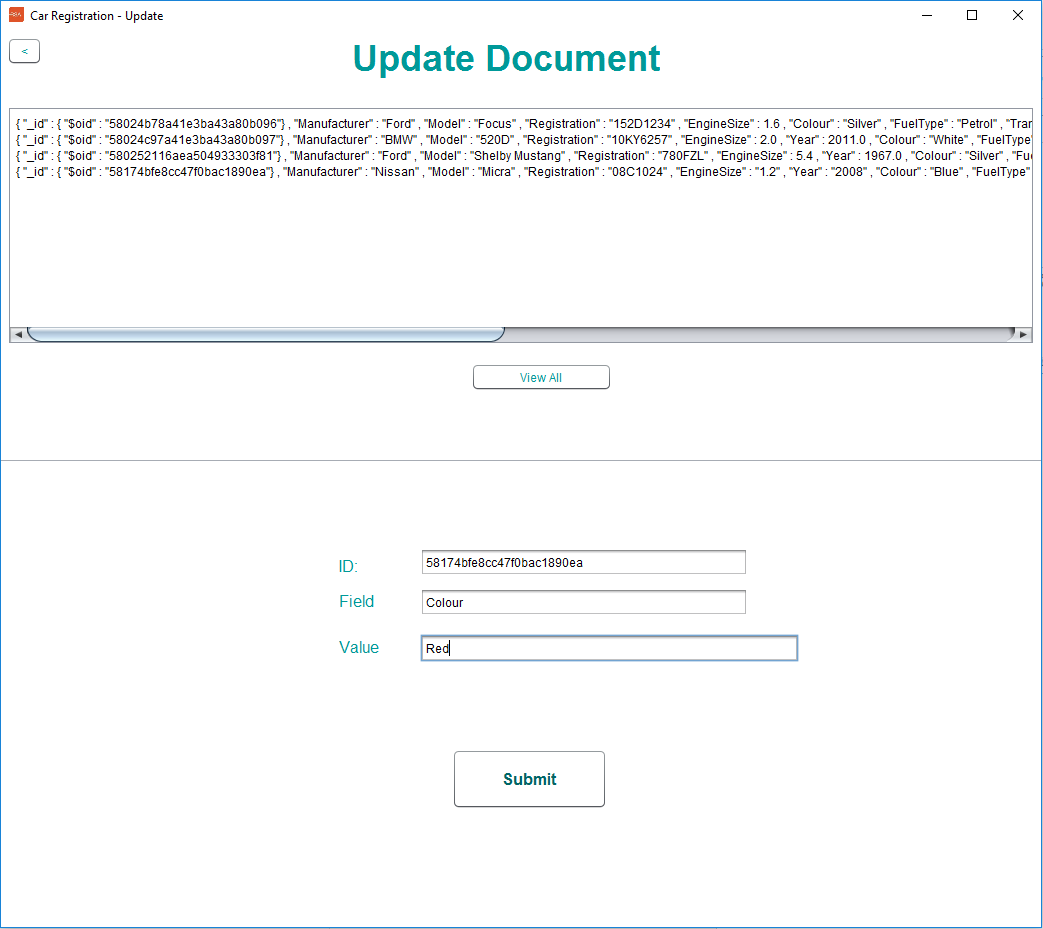


(View by Transmission)



### Update:

I reused the view element for my update so the user could see all the records while updating one.

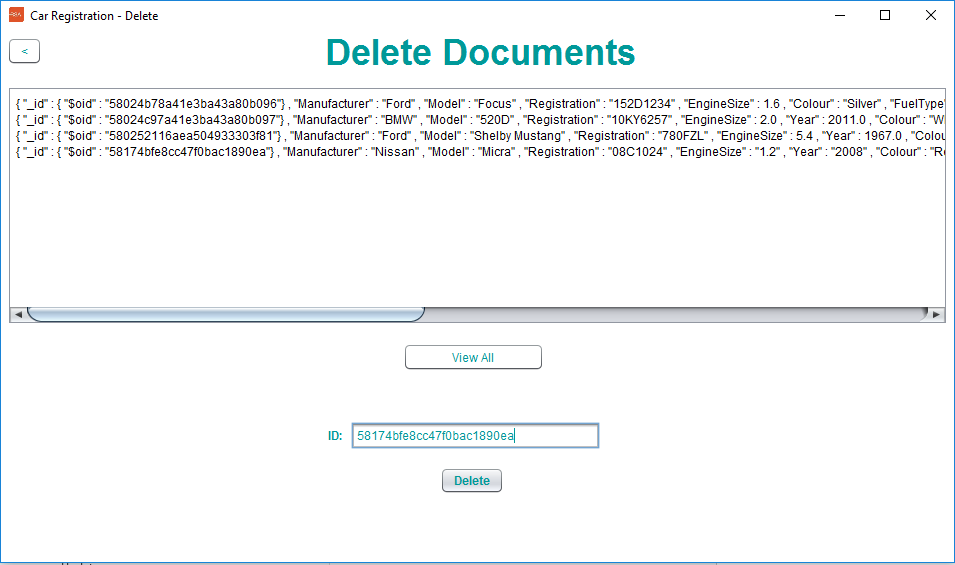


The updated document can then be seen in the text box

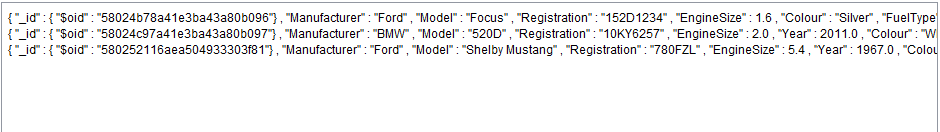


### Delete:

I also used the same view text box in my delete page to make it easier to see what you are deleting.

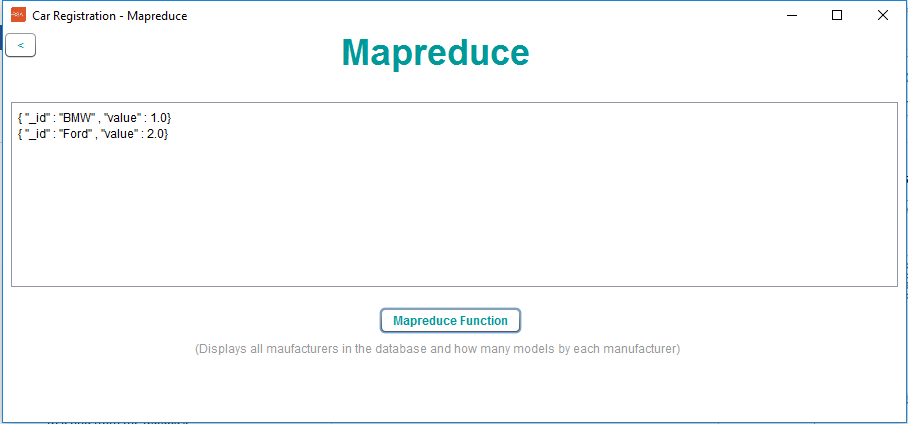


After a confirmation dialog, the text box view resets.



### Map Reduce:

Those were all the CRUD operations that I separated into their own pages, then I created a page for the map reduce.



This was a very basic page that simply used the map reduce function I showed earlier that total the number of cars by each manufacturer in the collection. Here it shows that there is only one BMW in the system but there are two Fords.

## Advantages and Disadvantages of Using Java:

Connecting to the database was very simple and there was a lot of help online, even more so than for Python and CouchDB due to the greater popularity of Java and MongoDB. Building a UI was very simple as I could use the NetBeans UI designer to do all the tedious work for me and I was actually able to create a fully functional application that all linked together perfectly and in the end I could build it into a single .jar file that could run anywhere. Having only one official Java Mongo driver made it much easier as I didn’t have to compare and sort through various different drivers and hope that all the documentation was applicable to my driver.

There was only one very slight disadvantage to using Java and that was that it was slightly simpler to connect and perform CRUD operations in Python but only very slightly. It was hard to find disadvantages as Java and MongoDB are often used together and they are very well documented so I had little difficulty creating my UI and database.

## Conclusion:

Working with MongoDB has been a lot easier than working with SQL databases with their schemas that are very difficult to alter as Mongo is a lot more fluid and can be changed very easily. Being able to update a single field without having to retype in all the others was a lot more user friendly than in my CouchDB project and overall I think using Java and MongoDB allowed me to create a much more user friendly application than my Python and CouchDB project. I personally would rather use MongoDB than CouchDB if only for how well it connects with Java.