

MongoDB Query Document using find() with Example

Details

Last Updated: 11 December 2019

The method of fetching or getting data from a MongoDB database is carried out by using queries. While performing a query operation, one can also use criteria's or conditions which can be used to retrieve specific data from the database.

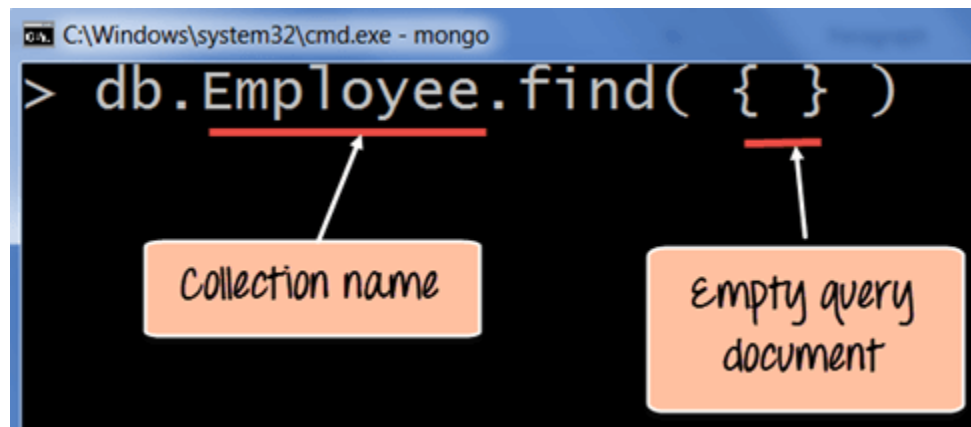
MongoDB provides a function called **db.collection.find()** which is used for retrieval of documents from a MongoDB database.

During the course of this tutorial, you will see how this function is used in various ways to achieve the purpose of document retrieval.

Basic query operations

The basic query operations cover the simple operations such as getting all of the documents in a MongoDB collection. Let's look at an example of how we can accomplish this.

All of our code will be run in the MongoDB [JavaScript](#) command shell. Consider that we have a collection named 'Employee' in our MongoDB database and we execute the below command.



Code Explanation:

1. Employee is the collection name in the MongoDB database
2. The find command is an in-built function which is used to retrieve the documents in the collection.

If the command is executed successfully, the following Output will be shown

Output:

Output:

```
db.Employee.find().forEach(printjson);
```

```
{ "_id" : ObjectId("563479cc8a8a4246bd27d784"),  
  "Employeeid" : 1,  
  "EmployeeName" : "Smith"  
}
```

```
{ "_id" : ObjectId("563479d48a8a4246bd27d785"),  
  "Employeeid" : 2,  
  "EmployeeName" : "Mohan"  
}
```

```
{ "_id" : ObjectId("563479df8a8a4246bd27d786"),  
  "Employeeid" : 3,  
  "EmployeeName" : "Joe"  
}
```

Find command returns all of the documents in the collection

The output shows all the documents which are present in the collection.

We can also add criteria to our queries so that we can fetch documents based on certain conditions.

Example 1

Let's look at a couple of examples of how we can accomplish this.

```
db.Employee.find({EmployeeName : "Smith"}).forEach(printjson);
```

Code Explanation:

1. Here we want to find for an Employee whose name is "Smith" in the collection, hence we enter the filter criteria as EmployeeName : "Smith"

If the command is executed successfully, the following Output will be shown

Output:

Output:

```
C:\Windows\system32\cmd.exe - mongo.exe
```

```
> db.Employee.find({EmployeeName : "Smith"}).forEach(printjson);
```

```
{ "_id" : ObjectId("563479cc8a8a4246bd27d784"),  
  "Employeeid" : 1,  
  "EmployeeName" : "Smith"  
}
```

The document which contains EmployeeName as Smith is returned

The output shows that only the document which contains "Smith" as the Employee Name is returned.

Example 2

Now, let's take a look at another code example which makes use of the greater than search criteria. When this criteria is included, it actually searches those documents where the value of the field is greater than the specified value.

```
db.Employee.find({Employeeid : {$gt:2}}).forEach(printjson);
```

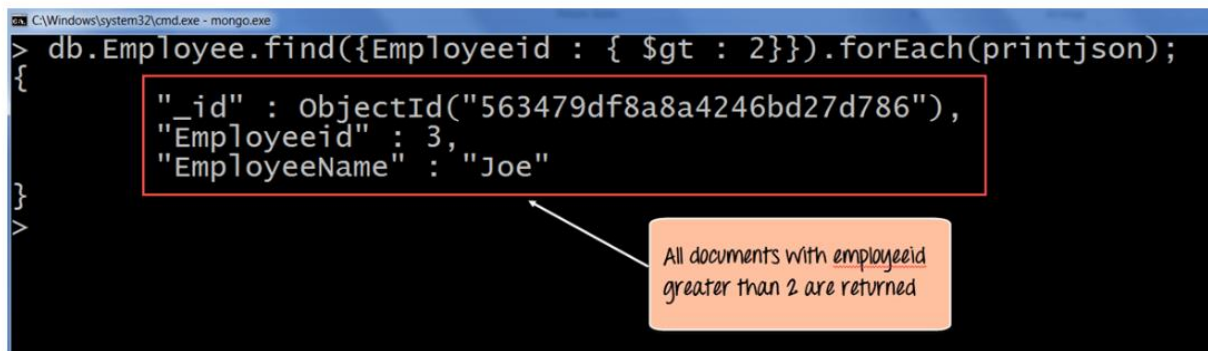
Code Explanation:

1. Here we want to find for all Employee's whose id is greater than 2. The \$gt is called a query selection operator, and what it just means is to use the greater than expression.

If the command is executed successfully, the following Output will be shown

Output:

Output:



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - mongo.exe". The command entered is `db.Employee.find({Employeeid : { $gt : 2}}).forEach(printjson);`. The output is a JSON document: `{"_id" : ObjectId("563479df8a8a4246bd27d786"), "Employeeid" : 3, "EmployeeName" : "Joe"}`. A red box highlights the output document. An arrow points from a text box to the `Employeeid` field in the output. The text box contains the text: "All documents with employeeid greater than 2 are returned".

All of the documents wherein the Employee id is greater than 2 is returned.