Unit testing/Integration Testing

During Development I Created a program to handle a couple restful messages. The following shows the commands and testing that was done for them and how to replicate on a users machine. Also note this was testing done on the reactive version.

First install mongodb and make sure to create a database called EPAMBlog

We could use postman for sending the restful request but for simplicity once the code is downloaded from the github repository(https://github.com/gpawelczyk9382/EPAM-Project) and ran you can run the following requests in the following document. Please note for testing id's are generated and you may need to update appropriately if testing locally.

Testing

Submit a blog post with their name (unique nickhandle), title and content with the publish date

Even though nickhandle is unique a user may enter more then one blog so I used a unique identifier for all blogs. I proceeded to have only 1 collection and have a field for parent that would potentially allow for limitless levels of comments like a real blog. Comments can hypothetically have parents. Users can potentially comment on comments or the blog itself, hence the type of Blog collection can be a B for blog itself or C for a comment type. A recursive method can potentially be created to traverse the blog structure.

I created the following restful commands to enter some sample data into the mongodb.

http://localhost:8080/SubmitBlog?BlogCreator=Greg&BlogTitle=Building%20new%20Sites&BlogText=General%20Data

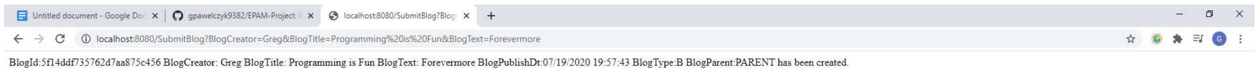
http://localhost:8080/SubmitBlog?BlogCreator=Jon&BlogTitle=Working%20with%20Spring&BlogText=General%20Data

http://localhost:8080/SubmitBlog?BlogCreator=Jim&BlogTitle=.Net%20is%20Fun&BlogText=General%20Data

http://localhost:8080/SubmitBlog?BlogCreator=Jon&BlogTitle=Almost%20got%20Webflux%20Working&BlogText=General%20Data

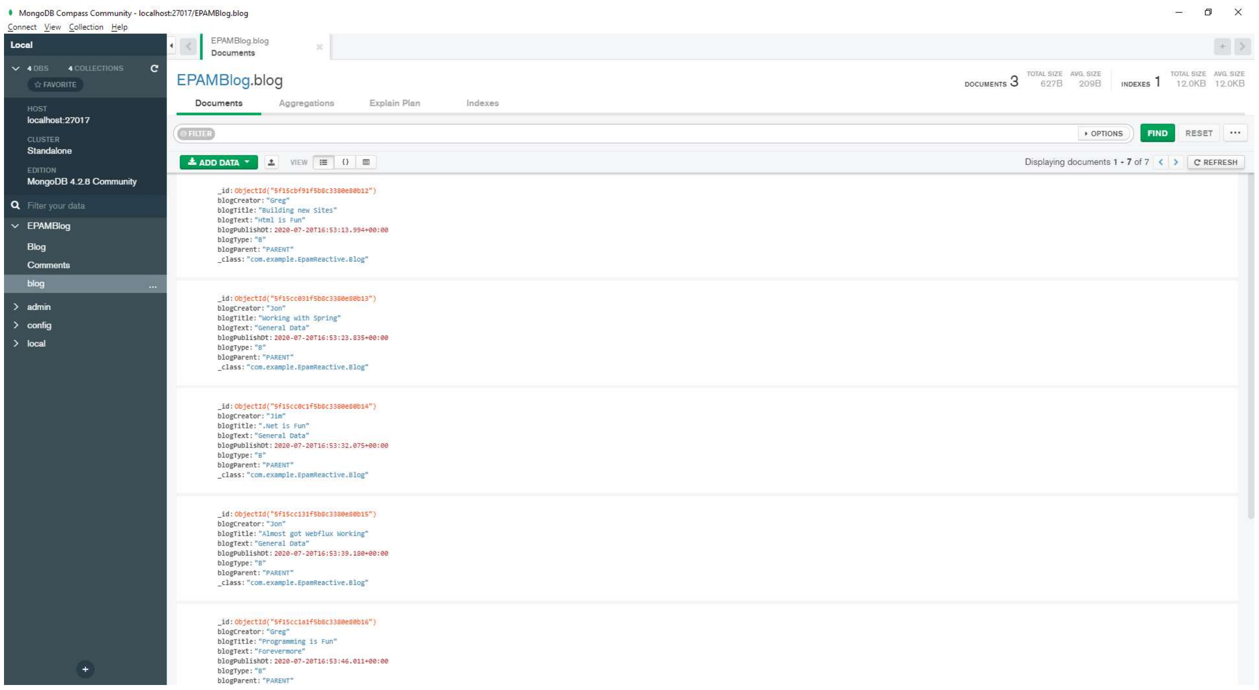
http://localhost:8080/SubmitBlog?BlogCreator=Greg&BlogTitle=Programming%20is%20Fun&BlogText=Forevermore

Note for purposes of testing when you add records I send a return of the record that was created and if it was successful.



```
localhost:8080/SubmitBlog?BlogCreator=Greg&BlogTitle=Programming%20is%20Fun&BlogText=Forevermore
BlogId:5f14ddf735762d7aa875c456 BlogCreator: Greg BlogTitle: Programming is Fun BlogText: Forevermore BlogPublishDt:07/19/2020 19:57:43 BlogType:B BlogParent:PARENT has been created.
```

We can see the data in local mongdb as in printscreen below or we can go and run the following call that will display all data that was created for testing.

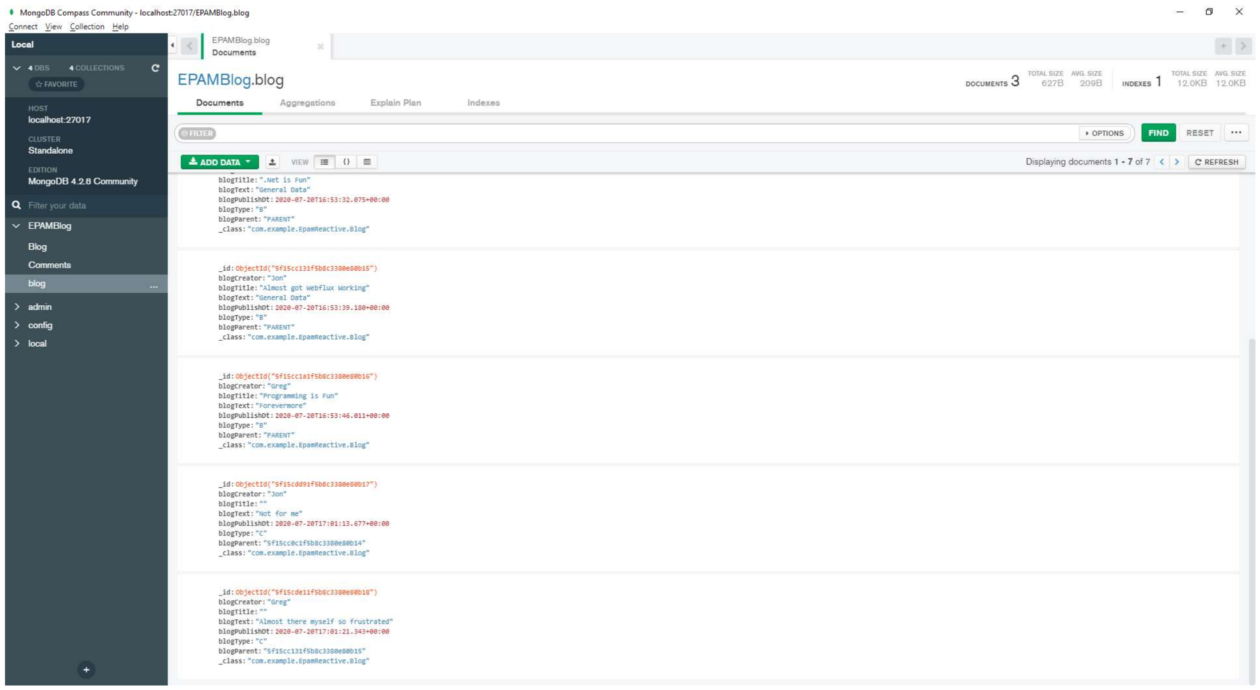


```
_id: ObjectId("5f15cbf91f5b8c3380e80b12")
blogCreator: "Greg"
blogTitle: "Building new Sites"
blogText: "Html is Fun"
blogPublishDt: 2020-07-20T16:53:13.994+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc031f5b8c3380e80b13")
blogCreator: "Jon"
blogTitle: "Working with Spring"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:23.835+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc0c1f5b8c3380e80b14")
blogCreator: "Jim"
blogTitle: ".Net is Fun"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:32.075+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc131f5b8c3380e80b15")
blogCreator: "Jon"
blogTitle: "Almost got Webflux Working"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:39.180+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc1a1f5b8c3380e80b16")
blogCreator: "Greg"
blogTitle: "Programming is Fun"
blogText: "Forevermore"
blogPublishDt: 2020-07-20T16:53:46.011+00:00
blogType: "B"
blogParent: "PARENT"
```

Update the blog post

We can update any post we want technically as long as we have the id of the blog records. Since users can add more than one blog post I did not want to create it by nickname. A users if utilizing this should use a unique identifier to update the correct blog. An example of the command to update a blog is below.

http://localhost:8080/UpdateBlog?Id=5f15cbf91f5b8c3380e80b12&BlogTitle=Building%20new%20Sites&BlogText=Html%20is%20Fun


MongoDB Compass Community - localhost:27017/EPAMBlog.blog
Connect View Collection Help
Local
4 DBS 4 COLLECTIONS
FAVORITE
HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 4.2.8 Community
Filter your data
EPAMBlog
Blog
Comments
blog
admin
config
local
EPAMBlog.blog Documents
EPAMBlog.blog
Documents Aggregations Explain Plan Indexes
DOCUMENTS 3 TOTAL SIZE 627B AVG. SIZE 209B
INDEXES 1 TOTAL SIZE 12.0KB AVG. SIZE 12.0KB
FILTER
OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 7 of 7
REFRESH

```
blogTitle: ".Net is Fun"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:32.075+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc131f5b8c3380e80b15")
blogCreator: "Jon"
blogTitle: "Almost got Webflux Working"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:39.180+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc1a1f5b8c3380e80b16")
blogCreator: "Greg"
blogTitle: "Programming is Fun"
blogText: "Forevermore"
blogPublishDt: 2020-07-20T16:53:46.011+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cdd91f5b8c3380e80b17")
blogCreator: "Jon"
blogTitle: ""
blogText: "Not for me"
blogPublishDt: 2020-07-20T17:01:13.677+00:00
blogType: "C"
blogParent: "5f15cc0c1f5b8c3380e80b14"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cde11f5b8c3380e80b18")
blogCreator: "Greg"
blogTitle: ""
blogText: "Almost there myself so frustrated"
blogPublishDt: 2020-07-20T17:01:21.343+00:00
blogType: "C"
blogParent: "5f15cc131f5b8c3380e80b15"
_class: "com.example.EpamReactive.Blog"
```

In the above requirement I created the ability to see all comments and/or blogs. Since it wasn't defined anything that is created whether comment or blog will be displayed in order of newest to oldest.

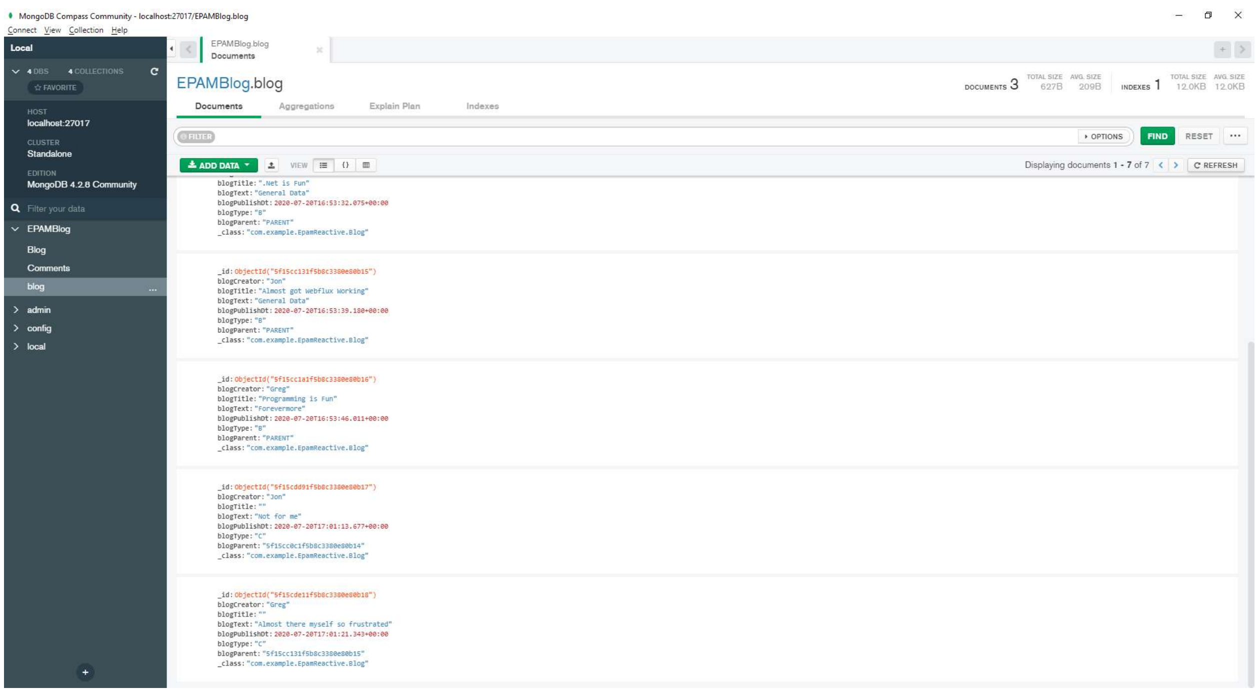
Comment on any post in a flat structure.

Creation of some commented data for 2 of the blogs below. We pass the blogid we want the comment to be associated to and we create a comment.

http://localhost:8080/CommentBlog?Id=5f15cc0c1f5b8c3380e80b14&BlogCommentCreator=Jon&BlogCommentText=Not%20for%20me

http://localhost:8080/CommentBlog?Id=5f15cc131f5b8c3380e80b15&BlogCommentCreator=Greg&BlogCommentText=Almost%20there%20myself%20so%20frustrated

Comments were successfully added. Shown below:



```
blogTitle: ".Net is Fun"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:32.075+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc131f5b8c3380e80b15")
blogCreator: "Jon"
blogTitle: "Almost got Webflux Working"
blogText: "General Data"
blogPublishDt: 2020-07-20T16:53:39.180+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cc1a1f5b8c3380e80b16")
blogCreator: "Greg"
blogTitle: "Programming is Fun"
blogText: "Forevermore"
blogPublishDt: 2020-07-20T16:53:46.011+00:00
blogType: "B"
blogParent: "PARENT"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cdd91f5b8c3380e80b17")
blogCreator: "Jon"
blogTitle: ""
blogText: "Not for me"
blogPublishDt: 2020-07-20T17:01:13.677+00:00
blogType: "C"
blogParent: "5f15cc0c1f5b8c3380e80b14"
_class: "com.example.EpamReactive.Blog"

_id: ObjectId("5f15cde11f5b8c3380e80b18")
blogCreator: "Greg"
blogTitle: ""
blogText: "Almost there myself so frustrated"
blogPublishDt: 2020-07-20T17:01:21.343+00:00
blogType: "C"
blogParent: "5f15cc131f5b8c3380e80b15"
_class: "com.example.EpamReactive.Blog"
```

Since comments and blogs are essentially the same and stored in the same collection it is easy to add a comment to a blog. We simply make sure we capture the id of the parent that the comment is for and save it in the record.

Finally we are tasked with View content posted by other users by their nicks ordered by creation time (newest first). I have the following command that will display all data for the user. When I put in Greg it will pull all the users data.

http://localhost:8080/ViewContent?BlogCreator=Greg

Used to help build initial dependencies:

https://start.spring.io/