Full Stack Development
MongoDB Lab

Creating a MongoDB data service

1. Lab objectives

In this lab, you will use Spring Boot and docker to implement a microservice type architecture.

You will not be writing any code in this lab since it is:

1. Similar to the code in the JPA lab
2. Just too much to accomplish in time we have

You will be using a demo from the Spring Boot Playground repository. This project has already been created using Spring Boot – check out the POM file to see the started dependencies.

It is recommended that you also try out some of the other projects in this repo and give credit to the repo owner who has done an amazing job of curating this set of projects

2. Clone the repo

In a clean directory, clone the repo from the URL

https://github.com/atharvasiddhabhatti/Springboot-Playground

```
D:\>git clone https://github.com/atharvasiddhabhatti/Springboot-Playground.git
Cloning into 'Springboot-Playground'...
remote: Enumerating objects: 665, done.
remote: Counting objects: 100% (665/665), done.
remote: Compressing objects: 100% (434/434), done.
Receiving objects:  90% (599/665)used 537 (delta 89), pack-reused 0
Receiving objects: 100% (665/665), 240.39 KiB | 2.29 MiB/s, done.
Resolving deltas: 100% (214/214), done.
```

Go into the directory **springboot-demo-mongodb-crud**

3. Examine the Data Service

The service is made up of two docker containers. Since you don't have MongoDB installed on your computer, you will use a docker container that contains an instance of the MongoDB.

The other container is the MVC frontend that has the same functionality as the MVC project you did earlier.

The two containers are started together a single application using a utility called "docker-compose." Docker will start up special private network these two containers will use to communicate, otherwise the pair of containers will work much like the containers you have already seen.

This service is described in a special configuration file shown here:

```
services:
    #service 1: definition of mongo database
    mongo_db:
      image: mongo
      container_name: mongoDB
      restart: always
      ports:
        - 27017:27017


    #service 2: definition of your spring-boot app
    productservice:                         #it is just a name, which will be used only in this file.
      image: product-service                #name of the image after dockerfile executes
      container_name: product-service-app   #name of the container created from docker image
      build:
        context: .                          #docker file path (. means root directory)
        dockerfile: Dockerfile              #docker file name
      ports:
        - "8080:8080"                       #docker containter port with your os port
      restart: always
      depends_on:                           #define dependencies of this app
        - mongo_db                                #dependency name (which is defined with this name 'db' in this file earlier)
```

The MongoDB service is an interface to an infrastructure component, which in this case is a MongoDB container.

The product service is a container that is build with the Dockerfile shown on the next page.

```
FROM openjdk:11 as mongoDbDemo
EXPOSE 8081
WORKDIR /app

# Copy maven executable to the image
COPY mvnw .
COPY .mvn .mvn

# Copy the pom.xml file
COPY pom.xml .

# Copy the project source
COPY ./src ./src
COPY ./pom.xml ./pom.xml

RUN chmod 755 /app/mvnw

RUN ./mvnw clean package -DskipTests
ENTRYPOINT ["java","-jar","target/springboot-demo-mongodb-crud-0.0.1-SNAPSHOT.jar"]
```

What we are doing here is copying the maven project to the container and building the executable jar in the container. This packaging allows you to run your executable inside the container.

However, this particular Dockerfile assumes you are coping from a Linux system into the Linux based container.

However, the Maven executable is a Windows executable and will not run in the container. Trying to run a docker build with this file will produce an error on the last line. It will work without error on a Mac or Linux based system. If you are running this lab on a Mac or Linux machine, you should not have to make this change. This has been tested on a Linux machine and it works perfectly as is

Just for this lab, you will rewrite the Dockerfile so that Docker build will work on a Windows system.

In the new version of the Dockerfile, you will do the maven build and then put the resulting jar file in the image rather than executing the whole build inside the image.

The new Dockerfile is in the directory MongoDB Lab directory and is shown on the next page

```
FROM openjdk:11 as mongoDbDemo
EXPOSE 8081
WORKDIR /app

# Copy maven executable to the image
# COPY mvnw .
# COPY .mvn .mvn

# Copy the pom.xml file
#COPY pom.xml .

# Copy the project source
COPY ./target ./target
#COPY ./pom.xml ./pom.xml

#RUN chmod 755 /app/mvnw

#RUN ./mvnw clean package -DskipTests
ENTRYPOINT ["java","-jar","target/springboot-demo-mongodb-crud-0.0.1-SNAPSHOT.jar"]
```

To create the executable, first run “mvn package” in the same directory as the POM.xml file.

Make sure that your Docker desktop is running.

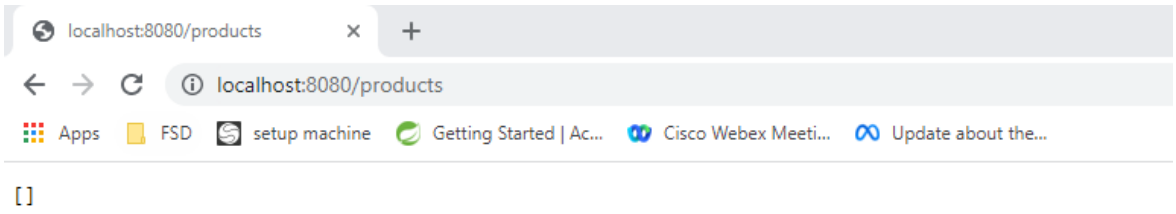
Now you can create start the service with the “docker-compose up” command as show below

```
D:\Springboot-Playground\springboot-demo-mongodb-crud>docker-compose up
Creating network "springboot-demo-mongodb-crud_default" with the default driver
Creating mongoDB ... done
Creating product-service-app ... done
Attaching to mongoDB, product-service-app
mongoDB             | {"t":{"$date":"2022-12-06T02:16:39.001+00:00"},"s":"I",  "c":"CONTROL",  "id":23285
g":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
```

```
onitor thread successfully connected to server with description ServerDescription{address=mongo_db:27017, type=
E, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeo
=30, roundTripTimeNanos=20180773}
product-service-app | 2022-12-06 02:16:42.381  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebSe
omcat started on port(s): 8080 (http) with context path ''
product-service-app | 2022-12-06 02:16:42.391  INFO 1 --- [           main] c.s.SpringbootDemoMongodbCrudApplic
tarted SpringbootDemoMongodbCrudApplication in 1.905 seconds (JVM running for 2.219)
```

4. Confirm the service is running

Confirm that the service is running by accessing its URL. Notice that there is no data in the database… yet.



5. Add some data

You are going to have to add some data using the POST method. In most interfaces, there would be some form to be filled out and submitted. In this case you will emulate this using the curl command.

Add three products using the command in the Commands.txt file in the Mongo Lab directory and then refresh the browser to see the three data items.

6. Modify some of the data

Use the put command in the Commands.txt file to change the description of the Apple product Refresh the browser to see the change.

7. Delete the data

Use the delete command in the Commands.txt file to delete the iPhone. Refresh the browser to see the change

8. Shut down the App

In a command prompt in the same directory where you did the docker-compose up command, check the running app with “docker ps”. Now execute the “docker-compose down” command to stop the service.

End Lab