链滴

SpringBoot 整合 Docker 和 MongoDB 笔记

作者：marshalby2

原文链接：https://ld246.com/article/1602314571431

来源网站：链滴

前言

一开始我只是想学一下Docker，在了解了Docker的基础知识后，发现Docker安装软件很方便，于是通过Docker安装了MongoDB，顺带着把MongoDB也学习了，最后想着，既然Docker和MongoDB学习了，为什么不顺带着将MongoDB 和 SpringBoot整合一下，再通过Docker部署呢？于是乎就有这篇笔记，记录一下学习的过程。

Docker 学习

Docker的学习主要参考Docker官网 和 菜鸟教程-Docker教程

Docker的安装以及 一些基础的知识，包括 **镜像** 和 **容器** 等概念，看一下Docker官网就可以 。需要练使用 **镜像** 和 **容器** 的操作指令，包括拉取（删除、打包. 镜像，运行（停止、删除）容器等。这些令和git和Linux的指令很像的，很容易掌握。总结如下：

镜像操作指令

以Docker官网提供的hello-world镜像为例

1. 查找镜像

指令： docker search hello-world

```
$ docker search hello-world
NAME                                      DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
hello-world                               Hello World! (an example of minimal Dockeriz…   1300                [OK]
kitematic/hello-world-nginx               A light-weight nginx container that demonstr…   147
tutum/hello-world                         Image to test docker deployments. Has Apache…   73                                      [OK]
dockercloud/hello-world                   Hello World!                                    19                                      [OK]
crccheck/hello-world                      Hello World web server in under 2.5 MB          13                                      [OK]
vad1mo/hello-world-rest                   A simple REST Service that echoes back all t…   4                                       [OK]
ppc64le/hello-world                       Hello World! (an example of minimal Dockeriz…   2
carinamarina/hello-world-app              This is a sample Python web application, run…   1                                       [OK]
ansibleplaybookbundle/hello-world-db-apb  An APB which deploys a sample Hello World! a…   1                                       [OK]
markmnei/hello-world-java-docker          Hello-World-Java-docker                         1                                       [OK]
souravpatnaik/hello-world-go              hello-world in Golang                           1
rancher/hello-world                                                                       1
ansibleplaybookbundle/hello-world-apb     An APB which deploys a sample Hello World! a…   1                                       [OK]
datawire/hello-world                      Hello World! Simple Hello World implementati…   1                                       [OK]
strimzi/hello-world-streams                                                               0
koudaiii/hello-world                                                                      0
strimzi/hello-world-consumer                                                              0
burdz/hello-world-k8s                     To provide a simple webserver that can have …   0                                       [OK]
businessgeeks00/hello-world-nodejs                                                        0
strimzi/hello-world-producer                                                              0
freddiedevops/hello-world-spring-boot                                                     0
kevindockercompany/hello-world                                                            0
nirmata/hello-world                                                                       0                                       [OK]
infrastructureascode/hello-world          A tiny "Hello World" web server with a healt…   0                                       [OK]
okteto/hello-world                                                                        0
```

2. 拉取镜像

指令： docker pull hello-world

```
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
0e03bdcc26d7: Downloading
latest: Pulling from library/hello-world
0e03bdcc26d7: Downloading
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

3. 列出所有镜像

指令： docker image ls 或者 docker images

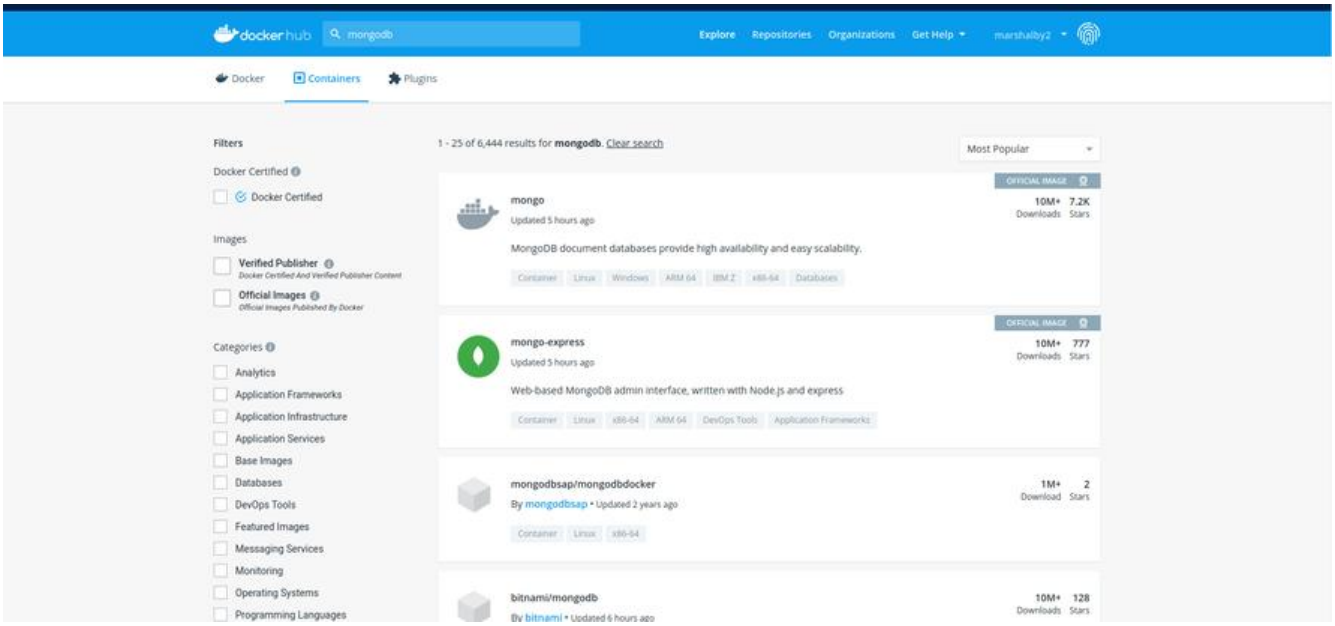
```
$ docker image ls
REPOSITORY                                TAG                 IMAGE ID            CREATED             SIZE
springboot/springboot-mongodb-docker      latest              598fbead8a67        24 hours ago        142MB
springboot/my-server                      latest              56bcce7f8568        2 days ago          147MB
springboot/my-springboot-docker           latest              cd572e40c1f9        2 days ago          474MB
<none>                                    <none>              b2c93728f7f3        2 days ago          121MB
<none>                                    <none>              fab7faa08bfe        2 days ago          105MB
marshalby2/cheers2019                     latest              8d96b6e76394        2 days ago          4.01MB
<none>                                    <none>              5d3bbcef2f5b        2 days ago          357MB
docker.elastic.co/kibana/kibana           7.9.2               ba296c26886a        3 days ago          1.18GB
docker.elastic.co/elasticsearch/elasticsearch   7.9.2         caa7a21ca06e        3 days ago          763MB
postgres                                  latest              d38331307292        8 days ago          314MB
adoptopenjdk/openjdk14                    latest              4d4d7a949e89        8 days ago          458MB
mongo                                     latest              50e17a9fdd96        9 days ago          492MB
mysql                                     latest              e1d7dc9731da        2 weeks ago         544MB
hello-world                               latest              bf756fb1ae65        8 months ago        13.3kB
b3log/solo                                latest              356131c98d4a        9 months ago        152MB
golang                                    1.11-alpine         e116d2efa2ab        13 months ago       312MB
openjdk                                   8-jdk-alpine        a3562aa0b991        16 months ago       105MB
```

4. 删除镜像

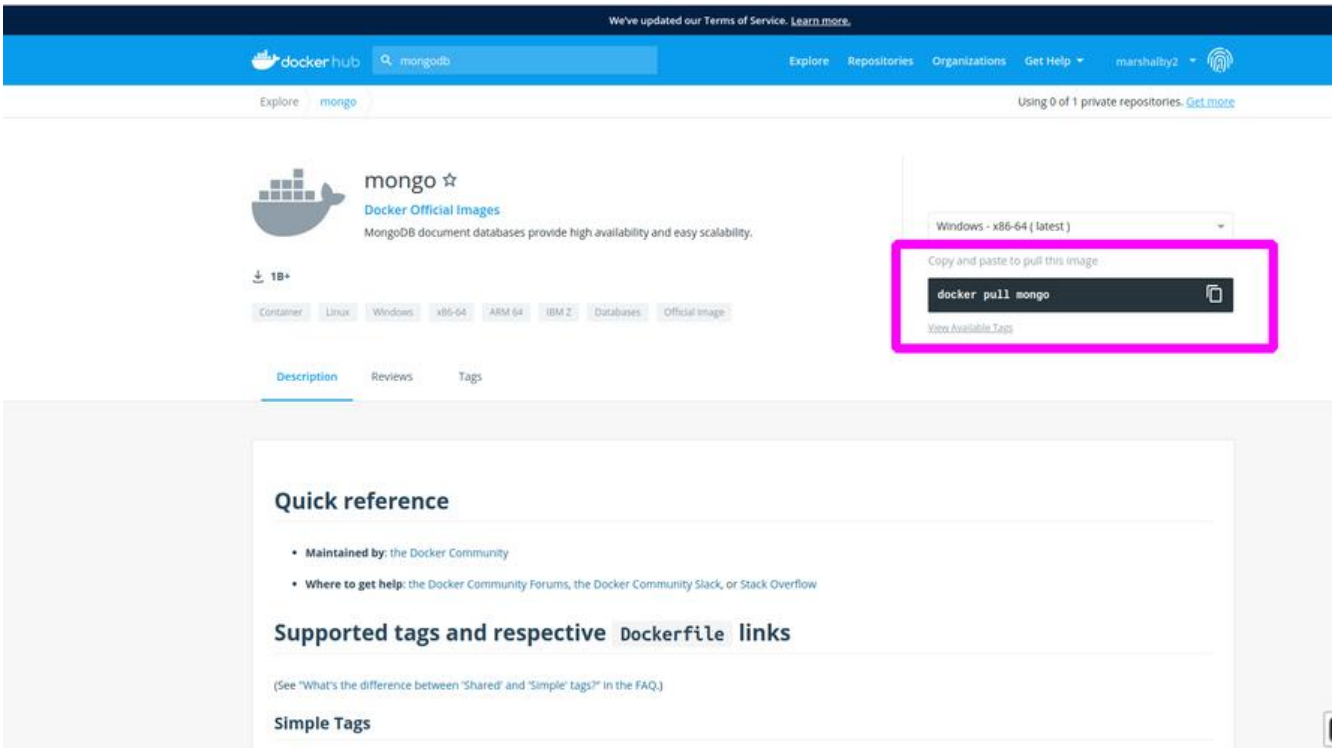
指令： docker rmi hello-world

容器操作指令

容器的操作，就结合安装MongoDB来一起学习，Docker安装软件就是将该软件对应的Docker镜像取下来，然后运行成为一个容器的过程。常用的开发软件都有对应的Docker镜像的，我们可以在DocerHub这个网站去找我们需要的软件镜像。比如我们搜索mongodb，如下图所示：



第一个就是我们要找的镜像，我们点进去，右侧就是我们拉取镜像的指令。



1. 拉取MongoDB镜像

指令： docker pull mongo

2. 运行容器

指令： docker run -itd --name mongo -p 27017:27017 mongo --auth

参数说明：

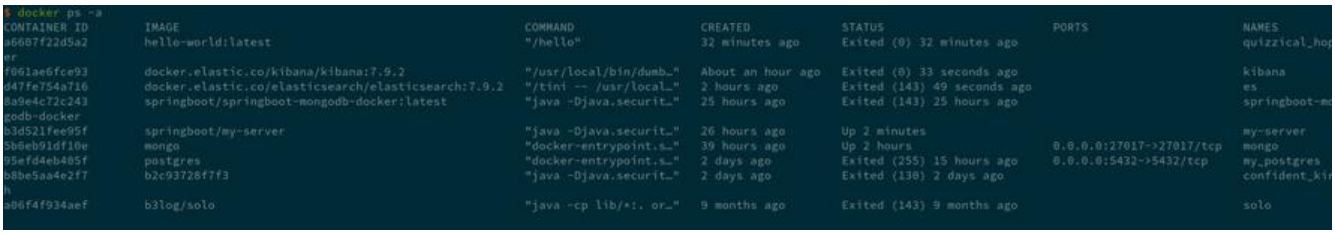
- -i : 交互式操作
- -t : 终端
- -d : 后台运行
- --name : mongo 指定容器的名字
- -p : 指定端口号
- --auth : 指的是 MongoDB数据库运行后，需要密码才能访问容器服务

其实使用Docker安装MongoDB这两步就完成了。

3. 查看容器信息

指令： docker ps -a

这个指令会列出容器ID、镜像、创建时间、状态、端口号、名称等信息，比较重要的就是 **CONTAINER ID(容器ID)**，**NAMES(容器名称)** 这两个，我们可以通过它们来管理镜像



4. 停止容器

指令：docker stop [容器ID] 或者 [容器名称]

5. 启动容器

指令： docker start [容器ID] 或者 [容器名称]

6. 重启容器

指令： docker start [容器ID] 或者 [容器名称]

7. 删除容器

指令： docker rm -f [容器ID] 或者 [容器名称]

MongoDB 学习

在前面我们已经通过Dokcer安装了MongoDB，现在我们通过docker exec -it mongo mongo 指令入MongoDB容器

```
$ docker exec -it mongo mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("225d5060-d245-4ab6-9d05-ce376de94325") }
MongoDB server version: 4.4.1
>
```

初次进入，我们需要创建一个超级用户，执行下面的两个命令：

```
# 1. 切换到admin数据库

 use admin

# 2. 创建超级用户admin

db.createUser({
    user: 'admin',
    pwd: '123456',
    roles: [{
        role: 'userAdminAnyDatabase',
        db: 'admin'
    }, "readWriteAnyDatabase"]
});
```

然后我们切换到test数据库，创建一个test用户，我们所有的学习都基于这个test数据库来进行，与上类似，执行下面两个指令：

```
# 1. 切换到test，说明一下，use指令切换数据库时，如果数据库不存在，会自动创建的

use test

# 2. 创建一个对test数据库具有读写权限的用户test

db.createUser({
    user: "test",
    pwd: passwordPrompt(), // or cleartext password
    roles: [{
        role: "readWrite",
        db: "test"
    }]
})
执行完这条语句后，会提示我们输入该用户的密码，输入完成，用户就创建成功了。
```

关于MongoDB其他知识，参考 MongoDB官网

创建SpringBoot项目，整合MongoDB 和 Docker

我们通过SpringBoot项目来整合MongoDB和Docker,然后通过Swagger 来测试

1. MongoDB和Swagger依赖配置

```
<!--   mongodb     -->
      <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <!--   web     -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--   lombok     -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <!-- swagger -->
        <dependency>
            <groupId>com.spring4all</groupId>
            <artifactId>swagger-spring-boot-starter</artifactId>
            <version>1.7.0.RELEASE</version>
        </dependency>
        <!--   启动类加上@EnableSwagger2Doc注解后,如果不加validation这个依赖,启动会失败,目
原因不明     -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
```

2. Docker插件配置

```
<properties>
        <java.version>1.8</java.version>
    <!-- dcoker 插件版本 -->
        <docker.maven.plugin.version>1.2.2</docker.maven.plugin.version>
        <docker.image.prefix>springboot</docker.image.prefix>
    </properties>

<!--     docker 配置       -->
        <plugin>
            <groupId>com.spotify</groupId>
            <artifactId>docker-maven-plugin</artifactId>
            <version>${docker.maven.plugin.version}</version>
            <configuration>
                <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
        <!--  指定Dockfile文件所在位置  -->
                <dockerDirectory>src/main/docker</dockerDirectory>
                <resources>
                    <resource>
                        <targetPath>/</targetPath>
                        <directory>${project.build.directory}</directory>
                        <include>${project.build.finalName}.jar</include>
                    </resource>
                </resources>
            </configuration>
        </plugin>
```

3. MongoDB连接配置

```
spring:
  data:
    mongodb:
      database: test
      host: 127.0.0.1
      port: 27017
      username: test
      password: test
```

4. 代码

实体类

```
@Data
public class Book {
    private Integer id;
    private String name;
    private String author;
    private String description;
}
```

操作MongoDB数据库主要通过注入MongoTemplate来实现

```
@Service
public class BookService {

    @Autowired
    private MongoTemplate mongoTemplate;

    /**
     *  保存
     *
     * @param book
     */
    public void save(Book book) {
        mongoTemplate.save(book);
    }


    /**
     *  根据ID查询
     *
     * @param id
     * @return
     */
    public Book findById(Integer id) {
       return mongoTemplate.findOne(Query.query(Criteria.where("id").is(id)), Book.class);
    }

    /**
     *  根据名字查询
```

```
     *
     * @param name
     * @return
     */
    public Book findByName(String name) {
        return mongoTemplate.findOne(Query.query(Criteria.where("name").is(name)), Book.class
;
    }

    /**
     *  查询所有
     *
     * @return
     */
    public List<Book> findAll() {
        return mongoTemplate.findAll(Book.class);
    }

    /**
     *  更新
     *
     * @param book
     */
    public void update(Book book) {
        Query query = Query.query(Criteria.where("id").is(book.getId()));
        Update update = Update.update("name", book.getName()).set("author", book.getAuthor
))
                .set("description", book.getDescription());
        mongoTemplate.updateFirst(query, update, Book.class);
    }


    /**
     *  删除
     *
     * @param id
     */
    public void delete(Integer id) {
        mongoTemplate.remove(Query.query(Criteria.where("id").is(id)), Book.class);
    }


}
```

控制层

```
@RestController
@RequestMapping("/book")
@Api(value = "书籍管理", tags = "book")
public class BookController {

    @Autowired
    private BookService bookService;
```

```
    @PostMapping("/save")
    @ApiOperation(value = "保存")
    public void save(@RequestBody Book book) {
        this.bookService.save(book);
    }

    @GetMapping("/getById/{id}")
    @ApiOperation(value = "根据ID查询")
    public Book getById(@PathVariable Integer id) {
        return bookService.findById(id);
    }

    @GetMapping("/getByName/{name}")
    @ApiOperation(value = "根据名称查询")
    public Book getByName(@PathVariable String name) {
        return bookService.findByName(name);
    }


    @GetMapping("/getAll")
    @ApiOperation(value = "查询所有")
    public List<Book> getAll() {
        return bookService.findAll();
    }


    @PutMapping("/update")
    @ApiOperation(value = "更新")
    public void update(@RequestBody Book book) {
        bookService.update(book);
    }

    @DeleteMapping("/delete{id}")
    @ApiOperation(value = "删除")
    public void delete(@PathVariable Integer id) {
        bookService.delete(id);
    }
}
```

5. 编写Dockfile文件

打包之前需要先编写一个Dcokerfile文件，这个文件我放在/src/main/docker/目录下，Dockerfile的体配置，可以看看这个 Docker Dockerfile

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ADD springboot-mongodb-docker-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

6. 将项目打包为Docker镜像

进入项目所在目录，执行mvn package docker:build命令，如果没有报错，就说明打包成功，此时行docker images指令查看镜像，第一个就是我们打包的镜像

```
$ docker images
REPOSITORY                                    TAG                 IMAGE ID            CREATED             SIZE
springboot/springboot-mongodb-docker          latest              598fbead8a67        30 hours ago        142MB
springboot/my-server                          latest              56bcce7f8568        2 days ago          147MB
<none>                                        <none>              b2c93728f7f3        3 days ago          121MB
<none>                                        <none>              fab7faa08bfe        3 days ago          105MB
marshalby2/cheers2019                         latest              8d96b6e76394        3 days ago          4.01MB
<none>                                        <none>              5d3bbcef2f5b        3 days ago          357MB
docker.elastic.co/kibana/kibana               7.9.2               ba296c26886a        3 days ago          1.18GB
docker.elastic.co/elasticsearch/elasticsearch 7.9.2               caa7a21ca06e        3 days ago          763MB
postgres                                      latest              d38331307292        8 days ago          314MB
adoptopenjdk/openjdk14                        latest              4d4d7a949e89        8 days ago          458MB
mongo                                         latest              50e17a9fdd96        9 days ago          492MB
mysql                                         latest              e1d7dc9731da        2 weeks ago         544MB
hello-world                                   latest              bf756fb1ae65        8 months ago        13.3kB
b3log/solo                                    latest              356131c98d4a        9 months ago        152MB
golang                                        1.11-alpine         e116d2efa2ab        13 months ago       312MB
openjdk                                       8-jdk-alpine        a3562aa0b991        16 months ago       105MB
```

7. 启动镜像

指令：docker run -d --net=host -p 8080:8080 --name springboot-mongodb-docker springboot/springboot-mongodb-docker:latest

参数说明：

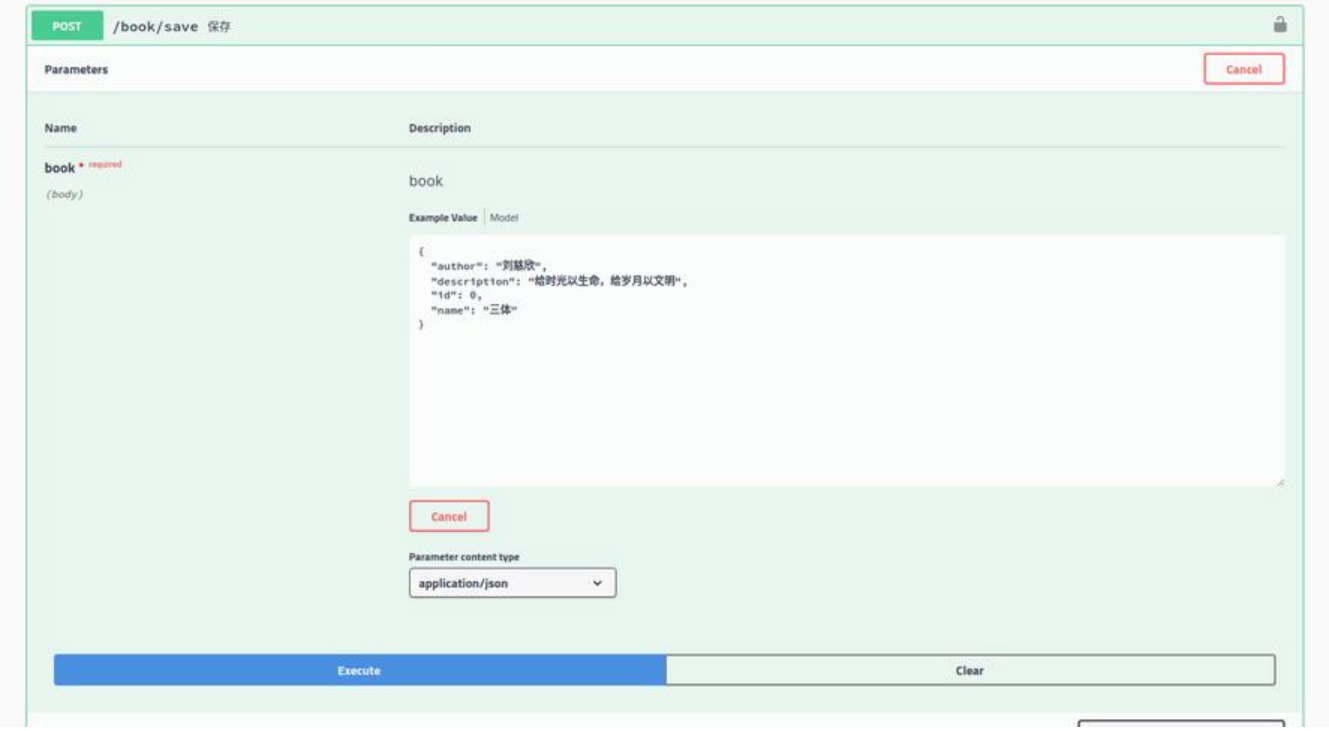
- -d : 后台运行
- --net=host: 使用host模式启动容器，此时容器和宿主机共用一个Network Namespace，这样我的容器才可以访问其他运行着的软件，比如MongoDB
- -p : 指定端口号
- --name: 指定我们容器的名字

如果我们想查看项目的日志，可以通过指令docker logs springboot-mongodb-docker > logs.txt 将日志写到logs.txt文件中

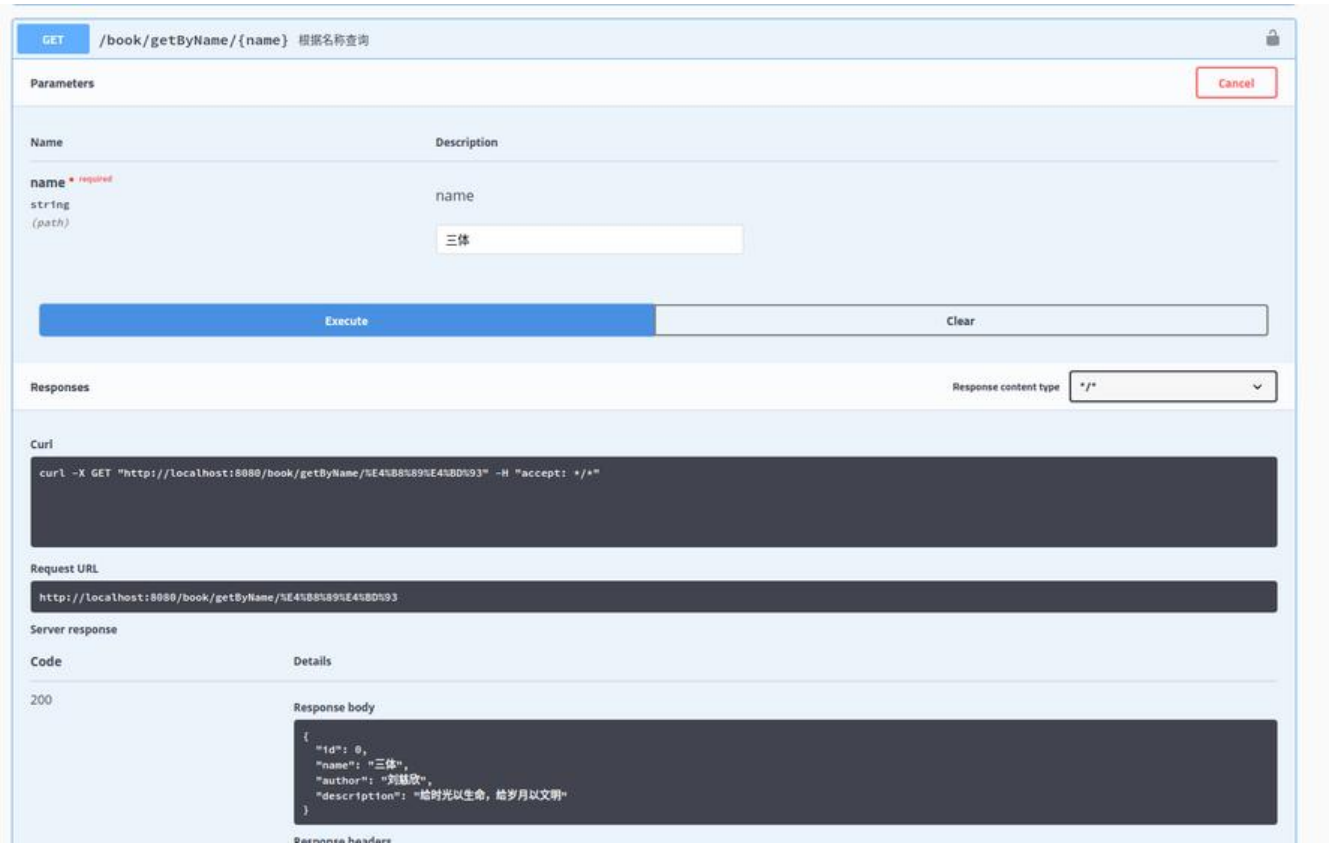
8. 通过Swagger测试

在浏览器输入http://localhost:8080/swagger-ui.html#/

测试保存接口



测试查询接口



进入数据库查看

至此，我们就成功了 smile

总结

通过Docker安装软件，是非常便利的，只需要学会docker pull 和 docker run两个指令就可以了。时，docker和 SpringBoot的整合，使我们部署维护项目也变得很简单。

完整代码 springboot-mongodb-docker