

hw10: Clustering & Docker

请你根据上课内容，针对你在E-BookStore项目中的数据库设计，完成下列任务：

1.请你参照课程样例，构建你的E-BookStore的集群，它应该至少包含 1 个nginx实例(负载均衡) + 1 个Redis实例(存储session) + 2 个Tomcat实例。(4分)

2.所使用的框架不限，例如可以不使用nginx而选用其他负载均衡器，或不使用Redis而选用其他缓存工具。

3.参照上课演示的案例，将上述系统实现容器化部署，即负载均衡器、缓存、注册中心和服务集群都在容器中部署。(1分)

– 请提交一份Word文档，详细叙述你的实现方式；并提交你的工程代码。

评分标准：

– 能够正确地部署和运行上述系统，在验收时需当面演示。

– 部署方案不满足条件或无法正确运行，则视情况扣分。

数据仓库：大量数据来源多个数据源，读入之后清洗加载，之后按照需求建立主题表，是需求驱动的（OLAP）

OLTP：事务驱动，需要承接这么多的请求

shard server：三备份，share nothing

Session + Redis

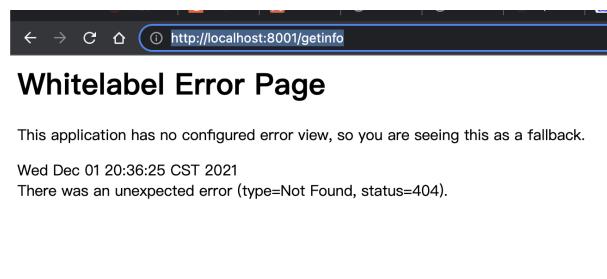
application.properties

```
...
#redis
spring.session.store-type=redis
spring.session.redis.namespace=spring:session
spring.redis.host= localhost
spring.redis.port= 6379
spring.redis.password=
spring.redis.timeout= 6000
spring.redis.jedis.pool.max-active=8
spring.redis.jedis.pool.max-wait= -1ms
spring.redis.jedis.pool.max-idle= 8
spring.redis.jedis.pool.min-idle= 0
spring.redis.database = 0
```

增加依赖

```
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
</dependency>
```

实例项目，登陆之后可以得到信息



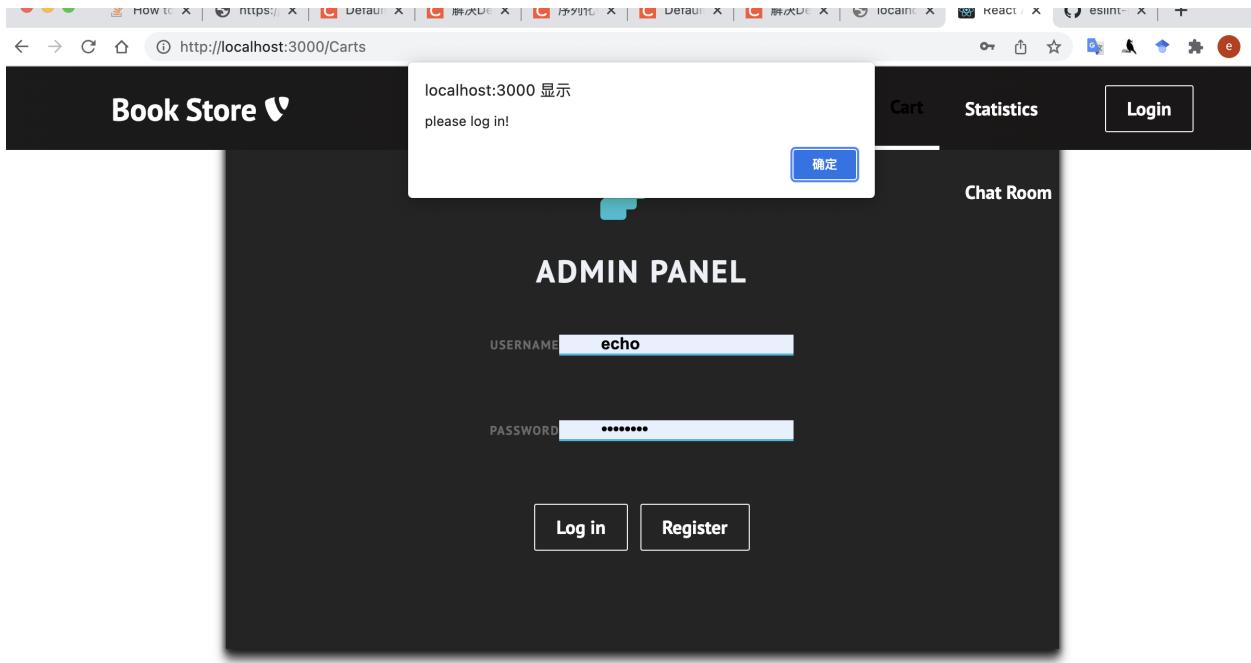
```
// 20211201203748
// http://localhost:8001/getInfo
1 [
2   "sessionId": "6e069454-ee42-4236-bca5-5c1195dee7fc",
3   "requestURI": "/session/getInfo",
4   "username": "zzt"
5 ]
```

自己的项目中，一开始没有登陆时无法获得用户购物车信息，登陆之后

The screenshot shows two separate Postman requests. The first request is a GET to `https://localhost:9090/session/login`. It has a single query parameter `user_id` with value `1`. The response body is `1 login finished`. The second request is a GET to `https://localhost:9090/session/getUserOrders?user_id=1`. It also has a single query parameter `user_id` with value `1`. The response status is `401`.

This screenshot shows a single Postman session. It starts with a GET request to `https://localhost:9090/session/login` with a `user_id` of `1`, resulting in a `200 OK` response with the message `login finished`. Following this, a second GET request is made to `https://localhost:9090/session/getUserOrders?user_id=1` using the same `user_id`. The response body is a JSON array containing three order objects:

```
[{"date": "2021-10-03", "orderId": 102, "order_price": 24687, "userId": 1}, {"date": "2021-10-03", "orderId": 152, "order_price": 2399, "userId": 1}, {"date": "2021-10-05", "orderId": 303, "order_price": 12129, "userId": 1}, {"date": "2021-10-05", "orderId": 352, "order_price": 3299, "userId": 1}]
```



前端在未登陆状态下查看order，会被要求登陆。

```
35) "order303"
36) "spring:session:sessions:09f4183e-52fd-4222-99a8-3a33bf47cbe6"
37) "BookISBN3"
38) "book16"
39) "order902"
40) "order805"
41) "book453"
42) "book5"
43) "book452"
44) "userOrder1"
45) "book10"
46) "BookISBN978-7-18618-3"
47) "book7"
48) "spring:session:expirations:1638717660000"
49) "allOrders"
```

redis中记录的session

Nginx

安装 brew install nginx

启动 brew start/restart nginx

① http://localhost:8080

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

实例程序

nginx.config 位于/usr/local/etc/nginx 编辑nginx.config，增加

```
events {
    worker_connections  1024;
}

http {
    include      mime.types;
    default_type application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #'          '$status $body_bytes_sent "$http_referer" '
    #'          '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;
    sendfile      on;
    #tcp_nopush   on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    upstream pancm{
        #ip_hash;
        #least_conn;
        server 127.0.0.1:8080 weight=3;
        server 127.0.0.1:8090;
    }

    #gzip  on;

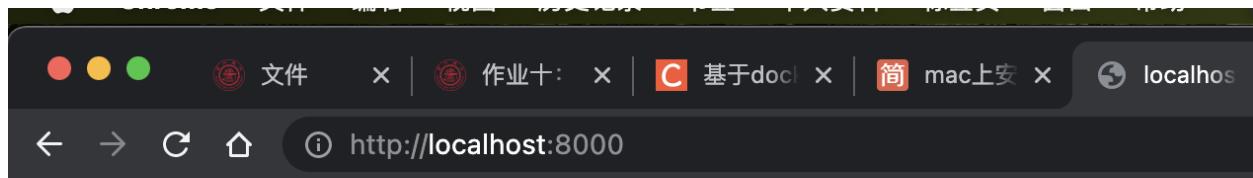
    server {
        listen      8000;
        server_name localhost;

        #charset koi8-r;
        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            proxy_pass http://pancm;
            index  index.html index.htm;
        }
        #error_page  404          /404.html;
        # redirect server error pages to the static page /50x.html
    }
}
```

现在启动样例项目进行测试（8080端口与8090端口）

重启并reload configuration `sudo nginx -s reload`



Let's start! Server One: New FFD6CD77465B3FE692AFE7074DDE2833

接下来更换自己的项目，复制后端工程两份配置为9091和9092，同时修改nginx.config

```
#keepalive_timeout 0;
keepalive_timeout 65;

upstream pancm{
#ip_hash;
#least_conn;
server 127.0.0.1:9091 weight=3;
server 127.0.0.1:9092;

}

#gzip on;

server {
    listen      9090;
    server_name localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
-- INSERT --
```

集群的端口是9091和9092， nginx 的端口为9091

Project Files

UserService

BookstoreApplication.java

application.properties

pom.xml (bookstore)

LoginController.java

OrderCon

BookstoreApplication.java

BookRep

Databases

BookstoreApplication.java

return args -> {

@Bean

CommandLineRunner demo {

bookTypeRepository:

BookType novel =

BookType children =

BookType biography =

BookType autobiography =

BookType science =

List<BookType> before =

log.info("Before: " + before);

for (BookType bookType : bookTypeRepository.findAll()) {

if (bookType.getName().equals("children")) {

bookType.setAgeRange(AgeRange.TEENAGER);

bookTypeRepository.save(bookType);

return bookType;

}

return null;

}

public Connector connector() {

Connector connector = new Connector("org.apache.coyote.http11.Http11NioProtocol");

connector.setScheme("http");

connector.setPort(8787);

connector.setSecure(false);

connector.setRedirectPort(9990);

return connector;

Run: BookstoreApplication

Console Endpoints

***** APPLICATION FAILED TO START *****

Description:

The Tomcat connector configured to listen on port 8787 failed to start. The port may already be in use or the connector may be misconfigured.

Action:

Verify the connector's configuration, identify and stop any process that's listening on port 8787, or configure this application to listen on another port.

Process finished with exit code 1

Invalid VCS root mapping
The directory...
Configure...

Git

Run

TODO

Problems

Terminal

Profiler

Endpoints

Build

Spring

1 hr 48 mins 197:1 main Material Deep Ocean

注意这里因为是listen的同一个端口会报错，所以把8787改为8788

```
70
71 @Bean
72     public Connector connector() {
73         Connector connector = new Connector( protocol: "org.apache.coyote.http11.Http11NioProtocol");
74         connector.setScheme("http");
75         connector.setPort(8788);|
76         connector.setSecure(false);
77         connector.setRedirectPort(9090);
78         return connector;
79     }
80 }
```

这样两个项目就跑完了。

发现会遇到TLS的问题，因为9090端口和9080端口前面都是https，之前做了TLS认证，于是把 proxy_pass 从http://pancm 改为 https://pancm；

成功结果

The image shows two separate Postman requests side-by-side.

Request 1:

- Method: GET
- URL: <https://localhost:9091/getUserCart?userId=1>
- Body tab selected
- Response (Pretty Print):

```

1 [
2   {
3     "cartId": 402,
4     "name": "A Promise Land",
5     "author": "Barack Obama",
6     "price": 1233,
7     "number": 1,
8     "bookId": 5,
9     "userId": 1
10 },
11 {
12   "cartId": 452,
13   "name": "Elon Musk",
14   "author": "Ashlee Vance",
15   "price": 3299,
16   "number": 5,
17   "bookId": 4,
18   "userId": 1
19 }
20 ]

```

Request 2:

- Method: GET
- URL: <https://localhost:9092/getUserCart?userId=1>
- Body tab selected
- Response (Pretty Print):

```

1 [
2   {
3     "cartId": 402,
4     "name": "A Promise Land",
5     "author": "Barack Obama",
6     "price": 1233,
7     "number": 1,
8     "bookId": 5,
9     "userId": 1
10 },
11 {
12   "cartId": 452,
13   "name": "Elon Musk",
14   "author": "Ashlee Vance",
15   "price": 3299,
16   "number": 5,
17   "bookId": 4,
18   "userId": 1
19 }
20 ]

```

The image shows a single Postman request with a successful response.

Request URL: <http://localhost:9090/getUserCart?userId=1>

Request Method: GET

Response Status: 200 OK | Time: 58 ms | Size: 441 B | Save Response

Settings tab selected

Response (Pretty Print):

```

1 [
2   {
3     "cartId": 402,
4     "name": "A Promise Land",
5     "author": "Barack Obama",
6     "price": 1233,
7     "number": 1,
8     "bookId": 5,
9     "userId": 1
10 },
11 {
12   "cartId": 452,
13   "name": "Elon Musk",
14   "author": "Ashlee Vance",
15   "price": 3299,
16   "number": 5,
17   "bookId": 4,
18   "userId": 1
19 }
20 ]

```

注意这里需要把https改为http

前端之前的https端口调用都改为http 端口

```
        return;
    }
    axios({
        method: 'GET',
        url: 'http://localhost:9090/login',
        params: {
            username: values.username,
            password: values.password
        }
    }).then(response => {
        console.log(response)
```

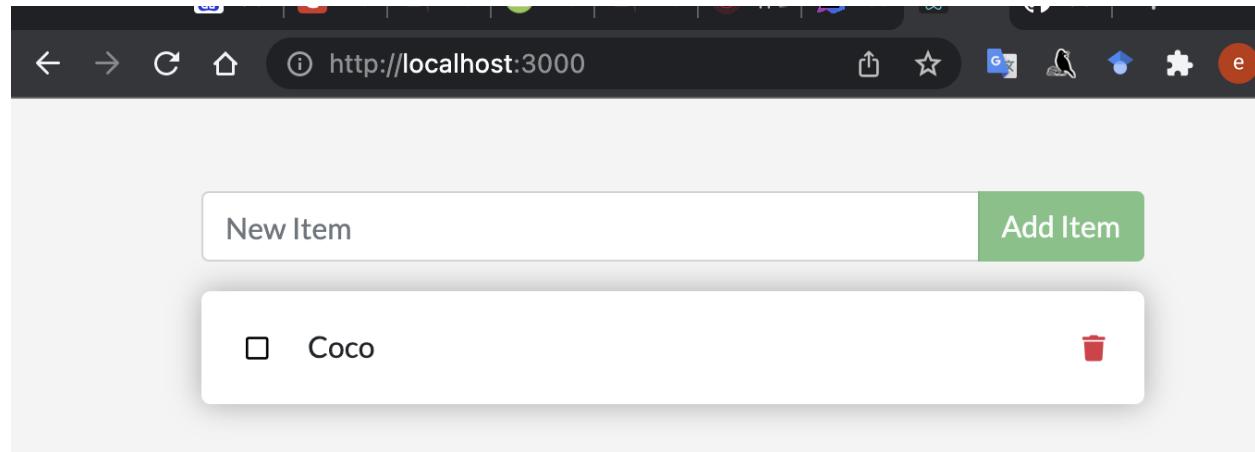
容器化部署

上课样例程序实现

Sample application

Estimated reading time: 5 minutes For the rest of this tutorial, we will be working with a simple todo list manager that is running in Node.js. If you're not familiar with Node.js, don't worry. No real JavaScript experience is needed.

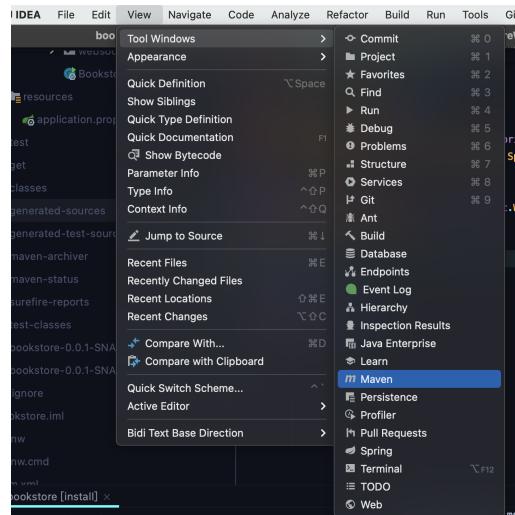
https://docs.docker.com/get-started/02_our_app/



	Named Volumes	Bind Mounts
Host Location	Docker chooses	You control
Mount Example (using <code>-v</code>)	<code>my-volume:/usr/local/data</code>	<code>/path/to/data:/usr/local/data</code>
Populates new volume with container contents	Yes	No
Supports Volume Drivers	Yes	No

docker

1. intelliJ 打包spring boot



springboot打jar包报javax.websocket.server.ServerContainer not available

参考解决

https://blog.csdn.net/qq_41754409/article/details/107997225

原注解 `@SpringBootTest` 改为

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
```

Shutdown in program

```
javax.jms.JMSException Create breakpoint : Could not create Transport. Reason: java.lang.IllegalStateException: Shutdown in progress
    at org.apache.activemq.util.JMSExceptionSupport.create(JMSExceptionSupport.java:36) ~[activemq-client-5.16.2.jar:5.16.2]
    at org.apache.activemq.ActiveMQConnectionFactory.createTransport(ActiveMQConnectionFactory.java:333) ~[activemq-client-5.16.2.jar:5.16.2]
    at org.apache.activemq.ActiveMQConnectionFactory.createActiveMQConnection(ActiveMQConnectionFactory.java:346) ~[activemq-client-5.16.2.jar:5.16.2]
    at org.apache.activemq.ActiveMQConnectionFactory.createActiveMQConnection(ActiveMQConnectionFactory.java:304) ~[activemq-client-5.16.2.jar:5.16.2]
    at org.apache.activemq.ActiveMQConnectionFactory.createConnection(ActiveMQConnectionFactory.java:244) ~[activemq-client-5.16.2.jar:5.16.2]
```

不是大的问题,只是因为测试结束了,Shutdown in progress 释放资源

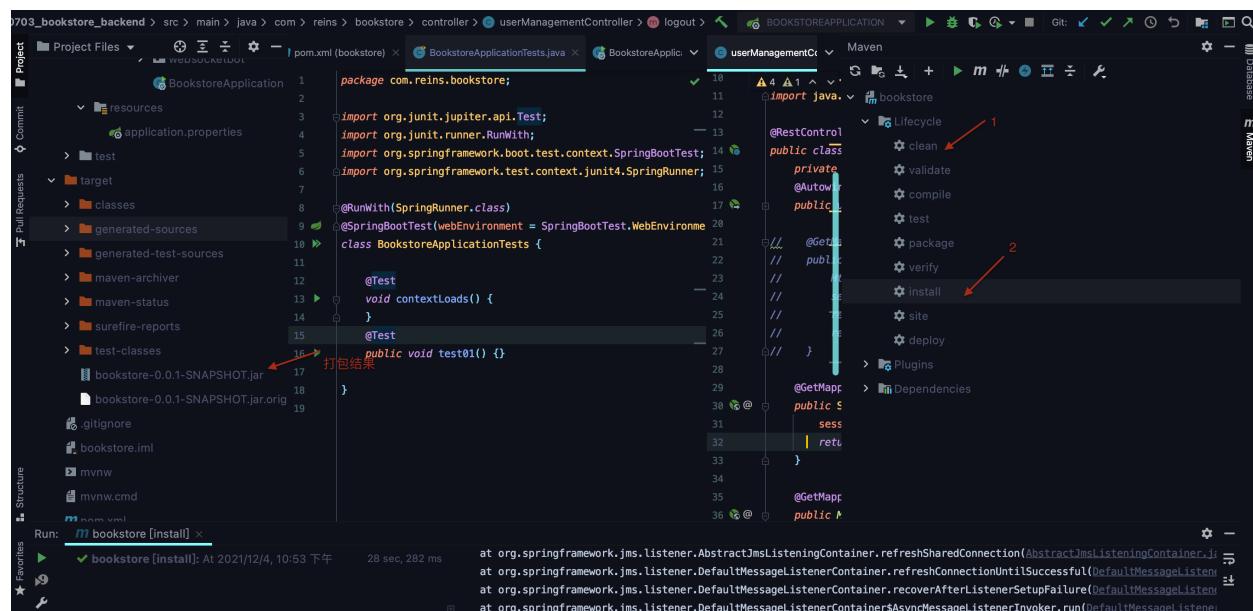
可以在测试方法结尾添加

```
while (true) {
```

```
}
```

就可以完美解决

<https://blog.csdn.net/zhangningniwanle/article/details/80062266>



可以看见左侧打包成功了出现jar包

运行命令

```
java -jar bookstore-0.0.1-SNAPSHOT.jar
```

```

2021-12-05 11:46:10.362 [INFO] 55576 --- [tomcatShutdownHook] C:\Users\kangyixiao\nikarik\nikari.datasource      : HikariPool-1 shutdown completed.
(base) ➜ target java -jar bookstore-0.0.1-SNAPSHOT.jar

.
.
.
:: Spring Boot ::          (v2.5.2)

2021-12-05 12:38:11.926 [INFO] 65463 --- [           main] c.reins.bookstore.BookstoreApplication : Starting BookstoreApplication v0.0.1-SNAPSHOT using Java 17 on 192.168.0.109 with PID 65463 (C:\Users\kangyixiao\Downloads\E-BookStore-aaf2e0701004e47b2c10b2724b4812737f42fd6\20210327_bookStoreWeb\20200703_bookstore_backend\target\bookstore-0.0.1-SNAPSHOT.jar started by kangyixiao in /Users/kangyixiao/Downloads/E-BookStore-aaf2e0701004e47b2c10b2724b4812737f42fd6\20210327_bookStoreWeb\20200703_bookstore_backend\target)
2021-12-05 12:38:11.929 [INFO] 65463 --- [           main] c.reins.bookstore.BookstoreApplication : No active profile set, falling back to default profiles: default
2021-12-05 12:38:13.214 [INFO] 65463 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode!
2021-12-05 12:38:13.217 [INFO] 65463 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-12-05 12:38:13.488 [INFO] 65463 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 257 ms. Found 6 JPA repository interfaces.
2021-12-05 12:38:13.526 [INFO] 65463 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode!
2021-12-05 12:38:13.529 [INFO] 65463 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Redis repositories in DEFAULT mode.
2021-12-05 12:38:13.580 [INFO] 65463 --- [           main] .RepositoryConfigurationExtensionSupport : Spring Data Redis - Could not safely identify store assignment for repository candidate interface com.reins.bookstore.repository.UserRepository. If you want this repository to be a Redis repository, consider annotating your entities with one of these annotations: org.springframework.data.redis.core.RedisHash (preferred), or consider extending one of the following types with your repository: org.springframework.data.keyvalue.repository.KeyValueRepository.
2021-12-05 12:38:13.582 [INFO] 65463 --- [           main] .RepositoryConfigurationExtensionSupport : Spring Data Redis - Could not safely identify store assignment for repository candidate interface com.reins.bookstore.repository.CartRepository. If you want this repository to be a Redis repository, consider annotating your entities with one of these annotations: org.springframework.data.redis.core.RedisHash (preferred), or consider extending one of the following types with your repository: org.springframework.data.keyvalue.repository.KeyValueRepository.
2021-12-05 12:38:13.584 [INFO] 65463 --- [           main] .RepositoryConfigurationExtensionSupport : Spring Data Redis - Could not safely identify store assignment for repository candidate interface com.reins.bookstore.repository.OrderRepository. If you want this repository to be a Redis repository, consider annotating your entities with one of these annotations: org.springframework.data.redis.core.RedisHash (preferred), or consider extending one of the following types with your repository: org.springframework.data.keyvalue.repository.KeyValueRepository.

```

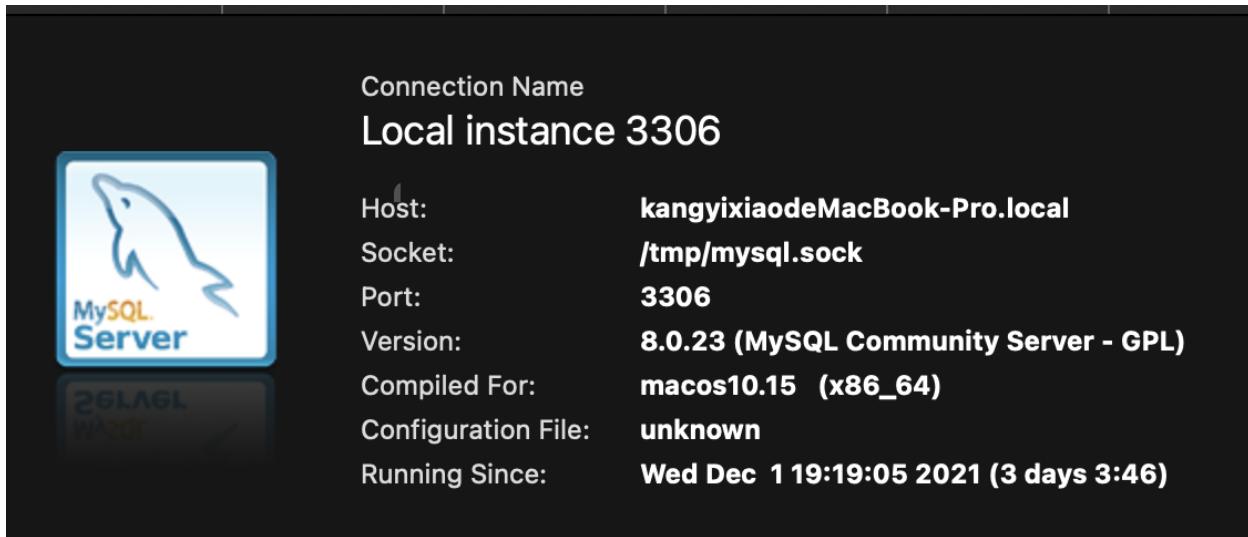
前端也可以用它来服务了。

```

{
  "bookId": 6,
  "isbn": "978-7-18618-6",
  "name": "Becoming",
  "type": "biographies",
  "author": "Michelle Obama",
  "price": 2342,
  "description": "Michelle Robinson Obama served as First Lady of the United States from 2009 to 2017. A graduate of Princeton University and Harvard Law School, Mrs. Obama started her career as an attorney at the Chicago law firm Sidley & Austin, where she met her future husband, Barack Obama. She later worked in the Chicago mayor's office, at the University of Chicago, and at the University of Chicago Medical Center. Mrs. Obama also founded the Chicago chapter of Public Allies, an organization that prepares young people for careers in public service. The Obamas currently live in Washington, DC, and have two daughters, Malia and Sasha. \"A serious work of candid reflection by a singular figure of early-twenty-first-century America . . . Becoming is refined and forthright, gracefully written and at times laugh-out-loud funny.\"-Isabel Wilkerson, The New York Times Book Review\n\n\"Becoming is inspirational without trying to be. From the first words, the very warmth that permeates its author emanates from the pages. . . . Becoming manages to be a coming-of-age tale, a love story and a family saga all in one. More importantly, this book is a reminder that America is still a work-in-progress, and that hope can be an action word if we allow it to be. Becoming is a balm that America needs, from a woman America does not yet deserve.\"-Angie Thomas, Time\n",
  "inventory": 12323,
  "image": "http://r.photo.store.qq.com/psc?V11fv0hk0pCaQ1/TmEUgtj9EK6.7V8ajmQrEC48C9FsUddTxr7k1JFsxXXPvEqCAslLaQE."
}

```

接下来获取各个依赖的版本



```
catelogd stopped
wallTrace: ~ nginx -v
nginx version: nginx/1.21.4
(base) ~ redis-server -v
Redis server v=6.2.6 sha=00000000:0 malloc=libc bits=64 build=c6f3693d1aced7d9
```

```
docker pull java:8
8: Pulling from library/java
5040bd298390: Pull complete
fce5728aad85: Pull complete
76610ec20bf5: Pull complete
60170fec2151: Pull complete
e98f73de8f0d: Pull complete
11f7af24ed9c: Pull complete
49e2d6393f32: Pull complete
bb9cdec9c7f3: Pull complete
Digest: sha256:c1ff613e8ba25833d2e1940da0940c3824f03f802c449f3d1815a66b7f8c0e9d
Status: Downloaded newer image for java:8
docker.io/library/java:8
```

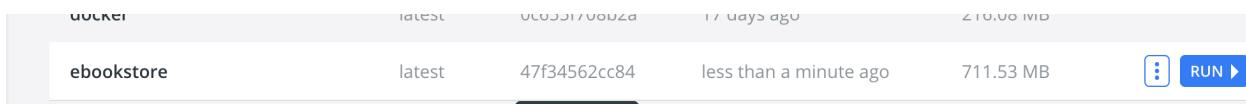
安装对应版本的image

Run 'docker image COMMAND --help' for more information on a command.					
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	
getting-started	latest	eaa5d35481a4	15 hours ago	410MB	
<none>	<none>	86b41bc02a9d	24 hours ago	327MB	
echoxiao99/getting-started	latest	de59840ce7e1	27 hours ago	410MB	
<none>	<none>	4be2d1fd56d9	41 hours ago	410MB	
7-gateway	latest	4a7ace1b26f1	41 hours ago	146MB	
omniapp	latest	407439bfea5b	41 hours ago	166MB	
gr-eureka	latest	26e7e31ca0ff	41 hours ago	149MB	
ich-redis	6.2.6	aea9b698d7d1	2 days ago	113MB	
2-mysql	5.7	738e7101490b	2 days ago	448MB	
nginx	1.21.4	f652ca386ed1	2 days ago	141MB	
nicolaka/netshoot	latest	f4c8dceca780	2 weeks ago	432MB	
AU-docker	latest	0c655f708b2a	2 weeks ago	216MB	
ers-node	12-alpine	2f014773d54a	3 weeks ago	89.5MB	
of-ubuntu	latest	ba6acccecd29	7 weeks ago	72.8MB	
/1-shenjiahuan/cselab_env	1.0	ce422d95bcdd	2 months ago	412MB	
fi-mysql	8.0.23	cbe8815cbea8	7 months ago	546MB	
lme-java	8	d23bdf5b1b1b	4 years ago	643MB	
(base) → ~					

编写Dockerfile, 在jar包同目录touch Dockerfile

```
# 基础镜像使用java
FROM java:8
# 维护人, 作者
MAINTAINER yixiao
# VOLUME 指定了临时文件目录为/tmp,
# 其效果是在主机 /var/lib/docker 目录下创建了一个临时文件, 并链接到容器的/tmp
VOLUME /tmp
# 将jar包添加到容器中并更名为app.jar
ADD bookstore-0.0.1-SNAPSHOT.jar eBookStore.jar
# 暴露运行端口, 但是这个EXPOSE并不会实际起作用, 而只是一个提示一个声明, 比如这里EXPOSE 80, 实际application.yaml中的server.port配置90, 那么容器实际有用的端口是9
# 在docker run -p指定宿主机端口与容器端口映射时容器端口应指定90而不是80
EXPOSE 80
# 指定prod环境运行
ENTRYPOINT ["java", "-Dspring.profiles.active=prod", "-Duser.timezone=GMT+8", "-jar", "eBookStore.jar"]
```

```
(base) ➔ dockerFile docker build -t ebookstore .
[+] Building 3.9s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 835B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/java:8
=> [1/2] FROM docker.io/library/java:8
=> [internal] load build context
=> => transferring context: 68.35MB
=> [2/2] ADD bookstore-0.0.1-SNAPSHOT.jar eBookStore.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:47f34562cc84199b7314099965c3bc7251d69a9a2d8d8db807d82d9079b82380
=> => naming to docker.io/library/ebookstore
```



跑容器

```
docker run -it --net=host --name ebookstor(base)  
docker run -it --net=host --name ebookstore -p 9091:9091 ebookstoree -p 9091:9091 ebookstore
```

```
docker run --name mysql -p 3306:3306 -v mysql_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=password -d mysql:8.0.23
```

于是就在docker环境跑起来了，但是会报mysql等依赖连接的错误

2021-12-05 13:14:30.890 ERROR 1 ---- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Exception during pool initialization.

```
com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
at com.mysql.cj.jdbc.exceptions.SQLException.createCommunicationsException(SQLError.java:174) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:64) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:833) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:453) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:246) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:198) ~[mysql-connector-java-8.0.25.jar!/:8.0.25]
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:138) ~[HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:364) ~[HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:206) ~[HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:476) [HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:561) [HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:115) [HikariCP-4.0.3.jar!:na]
at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:112) [HikariCP-4.0.3.jar!:na]
at org.hibernate.engine.jdbc.connections.internal.DataSourceConnectionProviderImpl.getConnection(DataSourceConnectionProviderImpl.java:122) [hibernate-core-5.4.32.Final.jar!/:5.4.32.Final]
at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess.obtainConnection(JdbcEnvironmentInitiator.java:180) [hibernate-core-5.4.32.Final.jar!/:5.4.32.Final]
at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator.initiateService(JdbcEnvironmentInitiator.java:68) [hibernate-core-5.4.32.Final.jar!/:5.4.32.Final]
at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator.initiateService(JdbcEnvironmentInitiator.java:35) [hibernate-core-5.4.32.Final.jar!/:5.4.32.Final]
```

Structure Favorites Run TODO Problems Terminal Profiler Build Endpoints Spring

```
ERROR 1049 (42000): Unknown database 'password'
root@96fde1534eb9:/# mysql -uroot -ppassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

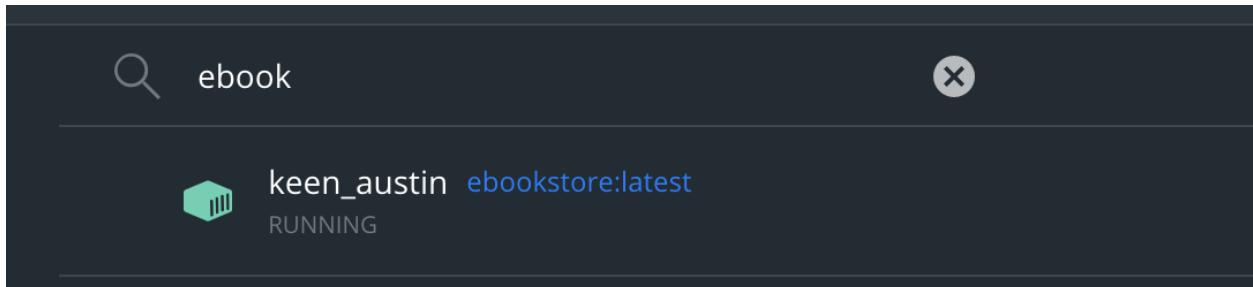
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Structure Favorites

```
docker exec -it mysql bash
mysql -uroot -ppassword
GRANT ALL PRIVILEGES ON . TO 'root'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

```
Bye  
root@84252263fa16:/# cat /etc/hosts  
127.0.0.1      localhost  
::1      localhost ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
172.17.0.2      84252263fa16
```



session 和 cookie 的区别

Cookies and Sessions are used to store information. Cookies are only stored on the **client-side** machine, while sessions get stored on the client as well as a **server**.

Session

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

A session ends when the user closes the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. Server script sends a set of cookies to the browser. For example name, age, or identification number etc. The browser stores this information on a local machine for future use.

When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

reference

<https://blog.csdn.net/jomexiaotao/article/details/83271458>

记一次docker部署springboot项目,mysql以及redis一样是docker中安装_zhs145612zhs的博客-CSDN博客

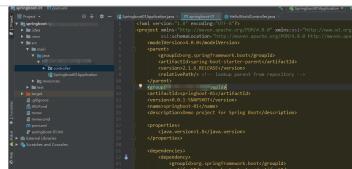
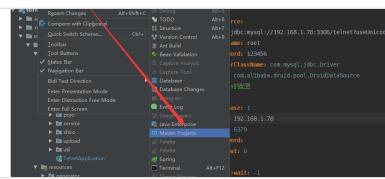
一：首先在idea中的springboot项目打成jar包,具体是方式是如下 这里的数据库密码和账户必须是真确的，数据库和redis也是在docker中安装的 二:在docker中安装mysql的服务 这里的linux的版本是centos,具体安装docker就不介绍了 执行docker search mysql 这里的最新版本是8.0版本了 如果采用的话 项目中使用这个的话就会导致一般的5.X的jar不兼容

[https://blog.csdn.net/zhs145612zhs/article/details/82225591?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link](https://blog.csdn.net/zhs145612zhs/article/details/82225591?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.no_search_link)

在springboot项目中怎么打包成jar，同时怎么解析jar_my_ha_ha的博客-CSDN博客.springboot怎么打包成jar

第一步、创建好一个springboot的项目当然、我们需要在pom文件中导入一个插件 <!--把项目打包成jar包的插件-->
<plugins> <plugin> <groupId>org.springframework.boot</groupId>...

https://blog.csdn.net/my_ha_ha/article/details/94183113



完整的docker+springboot+mysql部署_eric程序之路-CSDN博客

网上文章多如牛毛，但在参照做时，多少遇到坑，故记一次完整的于此！我使用的是centos7。 1、准备阶段 a) 安装docker b) 安装mysql c) springboot打包与Dockerfile 2、开始部署 a) jar、Dockerfile文件 b) 运行mysql并连接navicat c) 运行项目 下载docker [root@localhost ~]# sudo yum install docker-ce 启动docker [root@localhost ~]# service docker start

https://blog.csdn.net/zhangcc233/article/details/96706157?spm=1001.2101.3001.6650.8&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-8.fixedcolumn&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-8.fixedcolumn

