MongoDB @CreateByShadow

一、是什么?

> 面向文档的NoSQL数据库，用于大量数据存储
>
> 专用名词：
>
> 1、数据库 - show dbs（databases） 包含多个集合
>
> 1、集合 collection 包含多个文档
>
> 2、文档： 集合中的记录，文档包含多个字段名称和值
>
> 3、字段：JSON的名称
>
> 4、游标：执行查询结果集的指针
>
> 5、JSON 文档中存储的数据格式

####二、为什么用?

> 1、非常灵活，可以适应实际的业务环境和需求
>
> 2、支持多种查询
>
> 3、支持索引，提高搜索性能
>
> 4、副本集、高可用
>
> 5、负载均衡、分片

二、与 RDBMS的对照

RDBMS	MongoDB	
Table	Collection	RDBMS - 表；MongoDB - 集合
Row	Document	RDBMS - 数据行；MongoDB - 文档
Column	Field	RDBMS - 数据列；MongoDB - 字段
JOIN	Embedded Document	RDBMS - 关联查询；MongoDB - 内嵌文档

三、数据库类型

类型	实现
关系型数据库	MySQL、Oracle、SQL Server
基于键值对	Redis、DynamoDB、Memcached
大数据存储(聚合查询高性能)	Cassandra、Hbase、Hypertable
基于Hadoop生态	Hive、Spark
面向文档(JSON/XML)	MongoDB、CouchDB、Amazon SimpleDB
基于图形(社交网络、物流、空间数据)	Neo4J、Infinite Graph、OrientDB、FlockDB
快速检索	lucence、Solr、Elasticsearch

四、3V + 3高

- 3V
  - 海量（Volume）
  - 多样（Variety）
  - 实时（Velocity）
- 3高
  - 高并发
  - 高可扩
  - 高性能

五、ACID、CAP 、BASE

> ACID：
>
> - 原子性（Atomicity）
> - 一致性（Consistency）
> - 隔离性（Isolation）
> - 持久性（Durability）
>
> CAP：
>
> - 强一致性（Consistency）
> - 可用性（Availability）
> - 分区容错性（Partition tolerance）
>
> CA - 单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。
> CP - 满足一致性，分区容忍的系统，通常性能不是特别高。（ZK集群）
> AP - 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。（Redis集群）
>
> BASE：
>
> - 基本可用（Basically Available）
> - 软状态（Soft state）
> - 最终一致性（Eventually consistent）

六、安装及连接

- 下载安装包
- 解压安装包

```
tar -zxvf mongodb-linux-x86_64-rhel62-4.0.23.tgz
```

- 创建mongodb数据文件夹

```
mkdir data
mkdir logs
mkdir conf
```

- 创建文件

```
cd ./logs
touch mongdb.log
cd ../conf/
touch mongodb.conf
```

- 编写配置文件

```
vim mongodb.conf
```

输入一下内容

```
#数据库路径
dbpath=/xx/xxx/mongodb/mongo/data
#日志输出文件路径
logpath=/xx/xxx/mongodb/mongo/logs/mongodb.log
#错误日志采用追加模式
logappend=true
#启用日志文件，默认启用
journal=true
#这个选项可以过滤掉一些无用的日志信息，若需要调试使用请设置为false
quiet=true
#端口号 默认为27017
port=27017
#允许远程访问
bind_ip=0.0.0.0
#开启子进程
fork=true
#开启认证，必选先添加用户
auth=true
```

- 配置环境变量

```
vim /etc/profile
```

添加内容

```
export PATH=$PATH:/xx/xxx/mongodb/mogon/bin
```

- 生效配置文件

  ```
  source /etc/profile
  ```

- 启动 mongo服务

  ```
  ./mongod -f /xx/xxx/mongodb/mongo/conf/mongodb.conf
  或者
  ./mongod --config /xx/xxx/mongodb/mongo/conf/mongodb.conf
  ```

- 查看服务是否正常启动

  ```
  netstat -lanp | grep 27017
  或者
  ps -ef | grep mongo
  ```

- 连接mongo服务

  ```
  ./mongo
  ./mongo --host=127.0.0.1 --port=27017
  ```

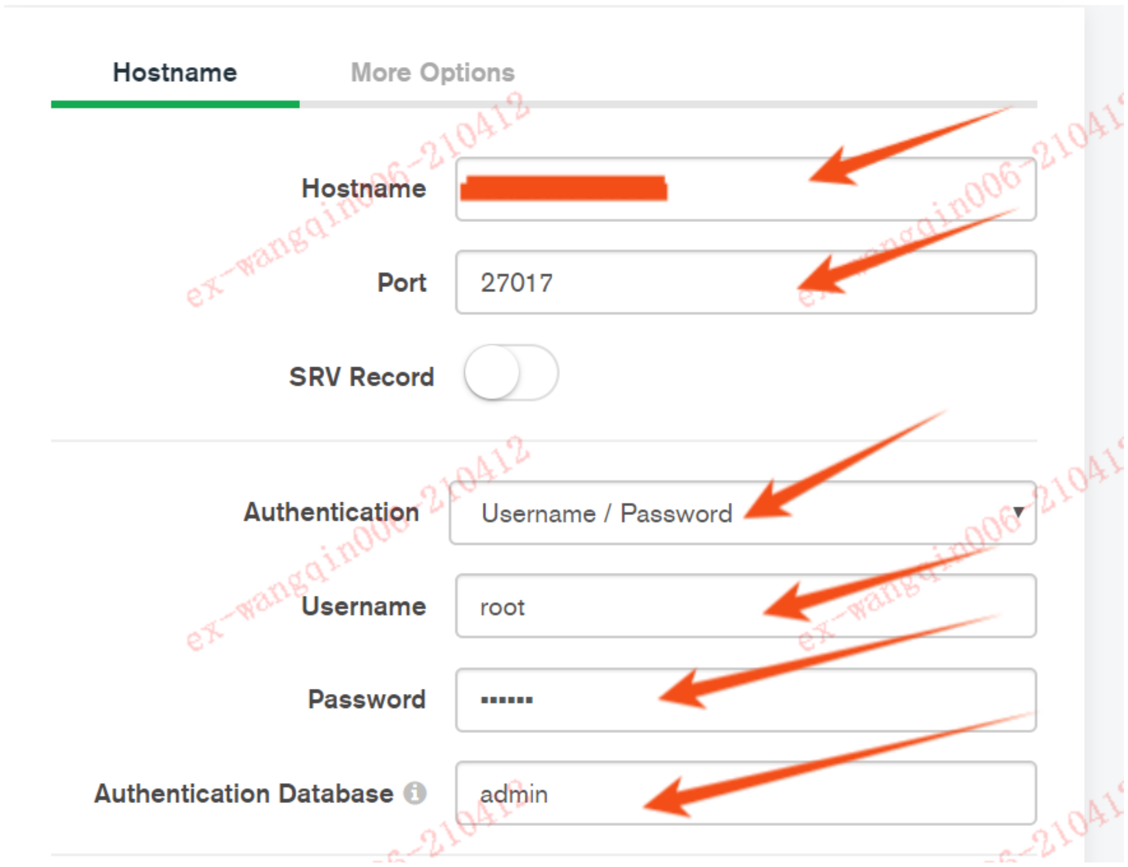
- 添加用户认证

  ```
  >use admin
  >db.createUser({user:"root",pwd:"123456",roles[{role:"root",db:"admin"}]})
  >db.shutdowServer()
  ```

- 认证

  ```
  >use admin
  >db.auth('root','123456')
  >show dbs
  ```

- 工具连接
  - MongoDBCompass
  - Robo 3T
  - Studio 3T
  - NoSQLbooster


Hostname
More Options

Hostname [redacted]

Port 27017

SRV Record

Authentication Username / Password

Username root

Password ••••••

Authentication Database admin

七、基础操作

1、数据库操作

[操作手册]()

- 创建（不存在会自动创建并切换到该库，存在则切换到该库）

  use 数据库名

  ```
  >use shadow
  ```

- 查看

  ```
  >db
  >show dbs
  >show databases
  ```

- 删除（use 到当前库然后执行）

  ```
  >db.dropDatabase()
  ```

- 修复

  ```
  ./mongod --repair --dbpath=./data
  ```

2、集合操作

操作手册

- 创建
  - 显式

    db.createCollection(集合名称)

    ```
    >db.createCollection("shop")
    ```

  - 隐式（没有shop集合会自动创建）

    ```
    >db.shop.insert({x:1})
    ```

- 删除 （db.集合名称.drop()）

  ```
  >db.shop.drop()
  ```

- 查看

  ```
  >show collections
  ```

3、文档操作

操作手册

- 新增

  db.集合名称.insert()

  db.集合名称.insertOne()

  db.集合名称.save() #通过传入的文档来替换已有文档，_id 主键存在就更新，不存在就插入

  db.集合名称.insertMany([])

  ```
  >db.shop.insert({name:"华为",price:6666})
  >db.shop.insertOne({name:"伏地魔"})
  >db.shop.save({name:"马士兵教育"})
  >db.shop.insertMany([{name:"华为",price:1111},{name:"小米",price:2222}])
  ```

- 查询

  db.集合名称.find(query,[projection])

  query： 可选，查询筛选器 JSON 对象

  projection：可选，结果字段 JSON 对象

  ```
  >db.shop.find().pretty() #查询全部
  >db.shop.find({name:"华为"}) #带条件查询
  >var c = db.shop.find() #得到游标
  >while(c.hasNext()){print(tojson(c.next()));} #游标迭代
  >db.shop.find().limit(2).forEach(printjson) #limit()个数限制,forEach()迭代
  >db.shop.find().sort({age:-1}) #sort()排序 按照age字段排序 -1表示降序，1表示升序
  ```

- 删除

db.集合名称.remove({})

```
>db.shop.remove({}) # 删除所有
>db.shop.remove({name:"伏地魔"}) # name是伏地魔的
>db.shop.remove({price:{$lte:2000}}) # price小于等于2000的
```

- 更新

  db.集合名称.update(query,update)

  db.集合名称.updateOne(query,update)

  db.集合名称.updateMany(query,update)

  query：条件，JSON对象

  update: 更新字段 JSON对象

```
>db.shop.update({name:"小米"},{$set:{price:3333}})
>db.shop.updateOne({y:111},{$set:{price:3333}})
>db.shop.updateMany({price:3333},{$set:{age:18}})
```

- 聚合

  相当于 SQL 查询的 GROUP BY，LEFT JOIN等操作

  db.集合名称.count()

  db.集合名称.distinct(Field) -- Field字段名称必须

```
>db.shop.count() #统计文档数量
>db.shop.distinct("price") #字段去重并返回去重后的值(数组)
```

4、常用的操作符

关系	格式	例子
等于	{< key > : < value >}	db.shop.find({name:"小米"}).pretty()
小于	{< key > : {$lt : < value >}}	db.shop.find({price:{$lt:100}}).pretty()
小于等于	{< key > : {$lte : < value >}}	db.shop.find({price:{$lte:100}}).pretty()
大于	{< key > : {$gt : < value >}}	db.shop.find({price:{$gt:100}}).pretty()
大于等于	{< key > : {$gte : < value >}}	db.shop.find({price:{$gte:100}}).pretty()
不等于	{< key > : {$ne : < value >}}	db.shop.find({price:{$ne:100}}).pretty()
and	{key1:value1,key2:value2} {$and:[{key1:value1},{key2:value2}]}	db.shop.find({name:"小米",price:100}).pretty() db.shop.find({$and:[{name:"小米"},{price:100}]}).pretty()
or	{$or:[{key1:value1},{key2:value2}]}	db.shop.find({$or:[{name:"小米"},{price:100}]}).pretty()
and or 联合	{key1:{$lte : value1},$or:[{key2 : value2},{key3 : value3}]}	db.shop.find({price:{$lte:100},$or:[{name:"小米"},{name:"z"}]}).pretty()
$type	{key1 : {$type :'string'}}	db.shop.find({addr:{$type:'string'}}).pretty()
$set 更新或添加字段	{$set : {key : value}}	db.shop.update({name:"zzz"},{$set:{email:111}})
$unset 删除字段	{$unset : {key : 1}}	db.shop.update({name:"zzz"},{$unset:{email:1}})

关系	格式	例子
$inc 数字增减	{$inc : {key:value}}	db.shop.update({name:"zzz"},{$inc:{price:10}})
$pull 数组删除元素	{$pull : {key : value }}	db.shop.update({name:"zzz"},{$pull:{addr:"xx"}})
$pop 删除数组 firts/last	{$pop : {key : -1/1}}	db.shop.update({name:"zzz"},{$pop:{addr:-1}})
$rename 修改字段名称	{$rename : {oldName : newName}}	db.shop.update({name:"zzz"},{$rename:{addr : address}})
$bit 位操作 integer类型	{$bit : {and/or : 5}}	db.shop.insert({name:"qqq",bits:NumberInt(2)}) db.shop.update({name:"qqq"},{$bit :{bits: {or: NumberInt(1)}}})

5、常用的方法

用途	方法	例子
分页查询	limit(n)	db.shop.find().limit(2).pretty()
跳越查询	skip(n)	db.shop.find().skip(2).pretty()
排序(1:升序 -1:降序)	sort({key1:1/-1})	db.shop.find({},{name:1,_id:0,price:1}).sort({price:-1}).pretty()
统计	count()	db.shop.count()
去重	distinct(key)	db.shop.distinct("name")

6、聚合操作

表达式	功能	例子
$sum	求和	db.shop.aggregate([{$group:{_id:"$name",num_price:{$sum:"$price"}}}])
$avg	求平均值	db.shop.aggregate([{$group:{_id:"$name",num_price:{$avg:"$price"}}}])
$min	最小值	db.shop.aggregate([{$group:{_id:"$name",num_price:{$min:"$price"}}}])
$max	最大值	db.shop.aggregate([{$group:{_id:"$name",num_price:{$max:"$price"}}}])
$push	结果文档中插入值到一个数组中	db.shop.aggregate([{$group:{_id:"$name",price:{$push:"$price"}}}])
$addToSet	结果文档中插入值到一个数组中,但不创建副本	db.shop.aggregate([{$group:{_id:"$name",price:{$addToSet:"$price"}}}])
$first	获取第一个文档数据	db.shop.aggregate([{$group:{_id:"$name",price:{$first:"$price"}}}])
$last	获取最后一个文档数据	db.shop.aggregate([{$group:{_id:"$name",price:{$last:"$price"}}}])

7、管道操作

表达式	功能	例子
$project	修改输入文档的结构。可以用来重命名、增加或删除域，也可以用于创建计算结果以及嵌套文档	db.shop.aggregate({$project : {name:1,price:1,_id:0}})
$match	用于过滤数据，只输出符合条件的文档	db.shop.aggregate([{$match:{name:"zzz"}}])
$limit	用来限制MongoDB聚合管道返回的文档数	db.shop.aggregate({$limit: 2})
$skip	跳过指定数量的文档，并返回余下的文档	db.shop.aggregate({$skip: 5})
$unwind	将文档中的某一个数组类型字段拆分成多条，每条包含数组中的一个值 addrs是数组	db.shop.aggregate([{$unwind:"$addrs"}])
$group	将集合中的文档分组，可用于统计	db.shop.aggregate([{$group:

表达式	功能	例子
$group	结果	{_id:"$name"}}])
$sort	将输入文档排序后输出	db.shop.aggregate([{$sort:{price:-1}}])
$geoNear	输出接近某一地理位置的有序文档	##

8、备份与恢复

- 备份

```
# 导出当前机器上的 test 库到当前位置
./mongodump -h localhost:27017 -uroot -p123456 --
authenticationDatabase=admin -d test -o .
```

> - -h:
>
> MongoDB 所在服务器地址，例如：127.0.0.1，当然也可以指定端口号：127.0.0.1:27017
>
> - -d:
>
> 需要备份的数据库实例，例如：test
>
> - -o:
>
> 备份的数据存放位置，例如：c:\data\dump，当然该目录需要提前建立，在备份完成后，系统自动在dump目录下建立一个test目录，这个目录里面存放该数据库实例的备份数据。

- 恢复

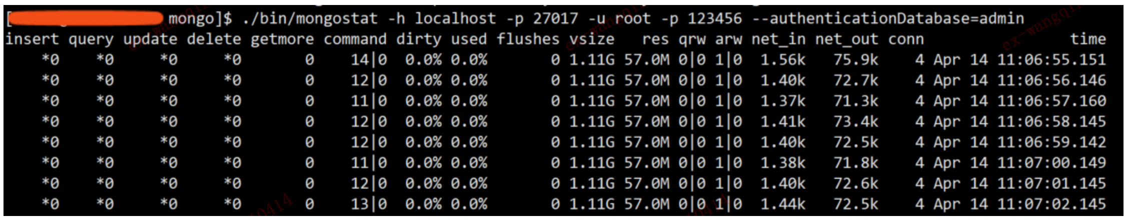
```
#导入/shadow目录下的库
./mongorestore -h localhost:27017 -d shadow -dir ./shadow/
```

> - --host <:port>, -h <:port>:
>
> MongoDB所在服务器地址，默认为： localhost:27017
>
> - --db , -d :
>
> 需要恢复的数据库实例，例如：test，当然这个名称也可以和备份时候的不一样，比如test2
>
> - --drop:
>
> 恢复的时候，先删除当前数据，然后恢复备份的数据。就是说，恢复后，备份后添加修改的数据都会被删除，慎用哦!
>
> - < path >:
>
> mongorestore 最后的一个参数，设置备份数据所在位置，例如：c:\data\dump\test。
>
> 你不能同时指定 和 --dir 选项，--dir也可以设置备份目录。
>
> - --dir:
>
> 指定备份的目录
>
> 你不能同时指定 < path > 和 --dir 选项。

9、监控

- mongostat

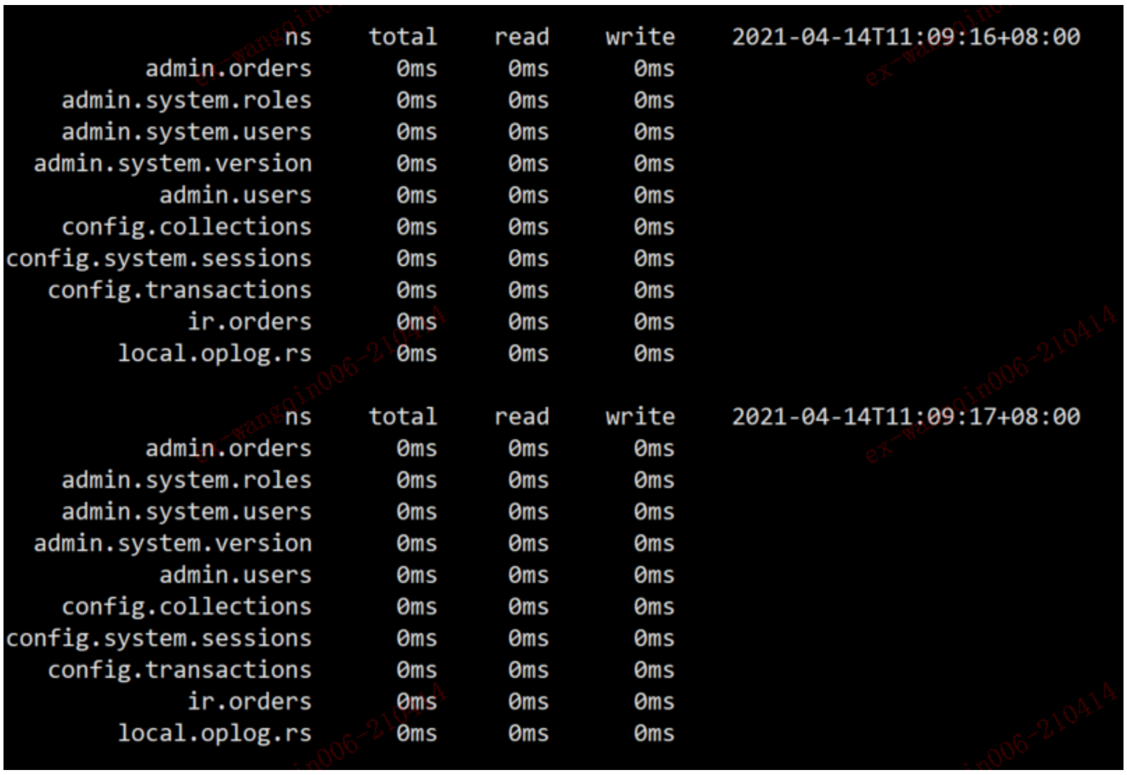
```
./bin/mongostat -h localhost -p 27017 -uroot -p123456 --
authenticationDatabase=admin
```



```
[[redacted] mongo]$ ./bin/mongostat -h localhost -p 27017 -u root -p 123456 --authenticationDatabase=admin
insert query update delete getmore command dirty used flushes vsize   res qrw arw net_in net_out conn                time
    *0    *0     *0     *0       0    14|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.56k   75.9k    4 Apr 14 11:06:55.151
    *0    *0     *0     *0       0    12|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.40k   72.7k    4 Apr 14 11:06:56.146
    *0    *0     *0     *0       0    11|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.37k   71.3k    4 Apr 14 11:06:57.160
    *0    *0     *0     *0       0    12|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.41k   73.4k    4 Apr 14 11:06:58.145
    *0    *0     *0     *0       0    12|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.40k   72.5k    4 Apr 14 11:06:59.142
    *0    *0     *0     *0       0    11|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.38k   71.8k    4 Apr 14 11:07:00.149
    *0    *0     *0     *0       0    12|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.40k   72.6k    4 Apr 14 11:07:01.145
    *0    *0     *0     *0       0    13|0  0.0% 0.0%       0 1.11G 57.0M 0|0 1|0  1.44k   72.5k    4 Apr 14 11:07:02.145
```

- mongotop

```
./bin/mongotop -h localhost -p 27017 -uroot -p123456 --
authenticationDatabase=admin 10
```



```
                      ns    total    read    write    2021-04-14T11:09:16+08:00
             admin.orders      0ms     0ms      0ms
       admin.system.roles      0ms     0ms      0ms
       admin.system.users      0ms     0ms      0ms
     admin.system.version      0ms     0ms      0ms
              admin.users      0ms     0ms      0ms
       config.collections      0ms     0ms      0ms
   config.system.sessions      0ms     0ms      0ms
      config.transactions      0ms     0ms      0ms
                ir.orders      0ms     0ms      0ms
           local.oplog.rs      0ms     0ms      0ms

                      ns    total    read    write    2021-04-14T11:09:17+08:00
             admin.orders      0ms     0ms      0ms
       admin.system.roles      0ms     0ms      0ms
       admin.system.users      0ms     0ms      0ms
     admin.system.version      0ms     0ms      0ms
              admin.users      0ms     0ms      0ms
       config.collections      0ms     0ms      0ms
   config.system.sessions      0ms     0ms      0ms
      config.transactions      0ms     0ms      0ms
                ir.orders      0ms     0ms      0ms
           local.oplog.rs      0ms     0ms      0ms
```

八、Java-MongoDB - 单机

1、pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.70</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-core</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

2、yml

```
server:
  port: 8888
spring:
  data:
    mongodb:
      # 连接 admin 库
      uri: mongodb://root:123456@10.25.175.108:27017/admin
      # 需要用户名和密码认证
      #uri:  mongodb://username:password@ip:port/admin
   #不需要用户名和密码认证
   #uri:  mongodb://ip:port/admin
logging:
  level:
    org:
      springframework:
        data:
          mongodb: DEBUG
```

3、java

- **User.java**

```
@Data
@Accessors(chain = true)
@Document("users") // 集合名称 users
public class User {

    private String name;

    private int age;

    private List<Address> addresses;

```

```java
    private List<String> emails;

    private Map<String,String> phones;
}
```

- **Address.java**

```java
@Data
@Accessors(chain = true)
public class Address {

    private String province;

    private String city;

    private String town;

}
```

- **UserService.java**

```java
public interface UserService {

    User insert(User user);

    long update(String uname);

    long updateChild(String uname,int index,String email);

    long delete(String uname);

    List<User> list();

    List<User> list(String uname);

    List<User> find(String province);
}
```

- **UserServiceImpl.java**

```java
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Override
    public User insert(User user) {
        // 添加记录
        return mongoTemplate.insert(user);
    }

    @Override
    public long update(String uname) {
        Query query = new Query();
```

```
 query.addCriteria(Criteria.where("name").is(uname).and("addresses.province"
).is("新疆"));
        // .$ 更新查询出的数组中的元素字段值，并设置新字段 street、ss(数组)
        Update update = Update.update("addresses.$.province", "武汉") //
child 属性
                              .set("age",30)
                              .set("addresses.$.street","Thanks♪(·ω·)/街道")
                              .addToSet("addresses.$.ss",
Arrays.asList("q","w","e"));
        UpdateResult result = mongoTemplate.updateMulti(query, update,
User.class);
        return result.getModifiedCount();
    }

    @Override
    public long updateChild(String uname,int index,String email) {
        Query query = new Query();
        query.addCriteria(Criteria.where("name").is(uname));
        Update update = Update.update("emails." + index, email); // .index
指定数组下标更新
        UpdateResult result = mongoTemplate.updateMulti(query, update,
User.class);
        return result.getModifiedCount();
    }

    @Override
    public long delete(String uname) {
        Query query = new Query();
        query.addCriteria(Criteria.where("name").is(uname));
        DeleteResult result = mongoTemplate.remove(query,User.class);
        return result.getDeletedCount();
    }

    @Override
    public List<User> list() {
        Query query = new Query();
        return mongoTemplate.find(query, User.class);
    }

    @Override
    public List<User> list(String uname) {
        Query query = new Query();
        query.addCriteria(Criteria.where("name").is(uname));
        return mongoTemplate.find(query, User.class);
    }

    @Override
    public List<User> find(String province) {
        Query query = new Query();

 query.addCriteria(Criteria.where("addresses.0.province").is(province));
        return mongoTemplate.find(query,User.class);
    }
}
```

- **UserController.java**

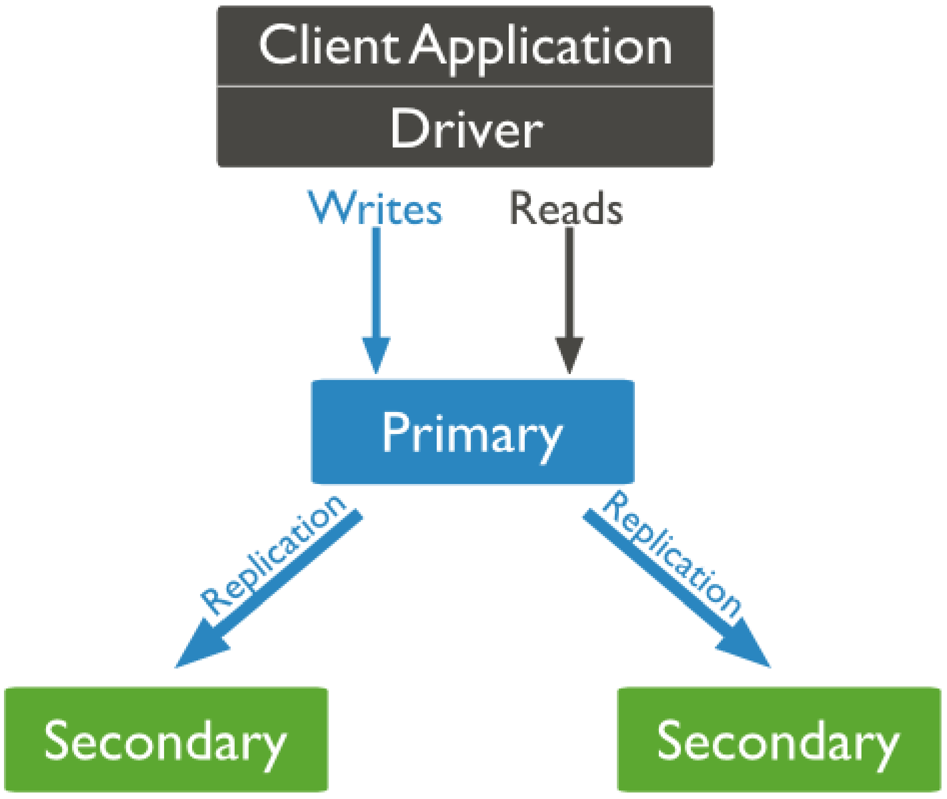
```java
@RestController
@RequestMapping("/mongo")
public class UserController {

    @Autowired
    private UserService userService;

    /**
     * 添加文档
     *{
     *  "name":"shadow",
     *  "age":18,
     *  "addresses":[
     *                 {
     *    "province":"湖南",
     *    "city":"张家界",
     *    "town":"永定区"
     *        },
     *        {
     *    "province":"广东",
     *    "city":"深圳",
     *    "town":"龙华区"
     *        }
     *   ],
     *  "emails":["438111969@qq.com","shadow@163.com"],
     *  "phones":{
     *   "1":"110",
     *   "2":"119",
     *   "3":"114"*  }
     * }
     */
    @PostMapping("/insert")
    public User insert(@RequestBody User user) {
        return userService.insert(user);
    }

    /**
     * 更新文档
     * /mongo/update/shadow
     */
    @PutMapping("/update/{uname}")
    public long update(@PathVariable("uname")String uname) {
        return userService.update(uname);
    }

    /**
     * 更新文档子内容
     * /mongo/update/child/shadow/438111969@qq.com
     */
    @PutMapping("/update/child/{uname}/{index}/{email}")
    public long updateChild(@PathVariable("uname")String
uname,@PathVariable("index")int index,@PathVariable("email")String email) {
        return userService.updateChild(uname,index,email);
    }

    /**
     * 删除文档
     * /mongo/delete/shadow
     */
    @DeleteMapping("/delete/{uname}")
    public long delete(@PathVariable("uname")String uname) {
        return userService.delete(uname);
    }

    /**
     * 查询文档
     * /mongo/list
     */
    @GetMapping("/list")
    public List<User> list() {
        return userService.list();
    }

    /**
     * 条件查询
     * /mongo/find/condition/shadow
     */
    @GetMapping("/find/condition/{uname}")
    public List<User> list(@PathVariable("uname")String uname) {
        return userService.list(uname);
    }

    /**
     * 内部条件查询
     * /mongo/find/inner/condition?province=湖南
     */
    @GetMapping("/find/inner/condition")
    public List<User> find(@RequestParam("province")String province) {
        return userService.find(province);
    }
}
```

- **postman.json**

```
{
 "id": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
 "name": "mongoDB",
 "description": "",
 "order": [
  "907d983b-80a1-54a8-a4e5-429b58337ef5",
  "ae8480f4-7d84-438a-96d3-e0886d62ceca",
  "422eefa1-ddb3-16ae-9f6d-64fba2c35738",
  "32107e4b-4476-9833-bf6c-331b45fcd855",
  "736dd113-f406-b66a-fae8-9042a073297f",
  "31f6a2d3-c359-b239-c942-f96cd42e3a0e",
  "902c88af-9d54-771d-b051-114a2fd9560b"
 ],
 "folders": [],
 "folders_order": [],
 "timestamp": 1618284064028,
 "owner": 0,
 "public": false,
 "requests": [
  {
    "id": "31f6a2d3-c359-b239-c942-f96cd42e3a0e",
    "headers": "",
    "headerData": [],
    "url": "localhost:8888/mongo/find/condition/shadow1",
    "queryParams": [],
    "pathVariables": {},
    "pathVariableData": [],
    "preRequestScript": null,
    "method": "GET",
    "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
    "data": null,
    "dataMode": "params",
    "name": "查询文档-带条件",
    "description": "",
    "descriptionFormat": "html",
    "time": 1618297217476,
    "version": 2,
    "responses": [],
    "tests": null,
    "currentHelper": "normal",
    "helperAttributes": {}
   },
   {
    "id": "32107e4b-4476-9833-bf6c-331b45fcd855",
    "headers": "",
    "headerData": [],
    "url": "localhost:8888/mongo/delete/shadow",
    "queryParams": [],
    "pathVariables": {},
    "pathVariableData": [],
    "preRequestScript": null,
    "method": "DELETE",
    "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
    "data": null,
    "dataMode": "params",
    "name": "删除文档",
    "description": "",
    "descriptionFormat": "html",
    "time": 1618296656680,
    "version": 2,
    "responses": [],
    "tests": null,
    "currentHelper": "normal",
    "helperAttributes": {}
   },
   {
    "id": "422eefa1-ddb3-16ae-9f6d-64fba2c35738",
    "headers": "",
    "headerData": [],
    "url": "localhost:8888/mongo/update/child/shadow/0/qqq@qq.com",
    "queryParams": [],
    "pathVariables": {},
    "pathVariableData": [],
    "preRequestScript": null,
    "method": "PUT",
    "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
    "data": null,
    "dataMode": "params",
    "name": "指定条件及数组下标更新文档",
    "description": "",
    "descriptionFormat": "html",
    "time": 1618295075331,
    "version": 2,
    "responses": [],
    "tests": null,
    "currentHelper": "normal",
    "helperAttributes": {}
  },
  {
    "id": "736dd113-f406-b66a-fae8-9042a073297f",
    "headers": "",
    "headerData": [],
    "url": "localhost:8888/mongo/list",
    "queryParams": [],
    "pathVariables": {},
    "pathVariableData": [],
    "preRequestScript": null,
    "method": "GET",
    "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
    "data": null,
    "dataMode": "params",
    "name": "查询全部文档",
    "description": "",
    "descriptionFormat": "html",
    "time": 1618296896751,
    "version": 2,
    "responses": [],
    "tests": null,
    "currentHelper": "normal",
    "helperAttributes": {}
  },
  {
    "id": "902c88af-9d54-771d-b051-114a2fd9560b",
    "headers": "",
    "headerData": [],
    "url": "localhost:8888/mongo/find/inner/condition?province=湖南",
    "queryParams": [
      {
        "key": "province",
        "value": "湖南",
        "equals": true,
        "description": "",
        "enabled": true
      }
    ],
    "pathVariables": {},
    "pathVariableData": [],
    "preRequestScript": null,
    "method": "GET",
    "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
    "data": null,
    "dataMode": "params",
    "name": "查询文档-带内部条件",
    "description": "",
    "descriptionFormat": "html",
    "time": 1618298214423,
   "version": 2,
   "responses": [],
   "tests": null,
   "currentHelper": "normal",
   "helperAttributes": {}
  },
  {
   "id": "907d983b-80a1-54a8-a4e5-429b58337ef5",
   "headers": "Content-Type: application/json\n",
   "headerData": [
    {
     "key": "Content-Type",
     "value": "application/json",
     "description": "",
     "enabled": true
    }
   ],
   "url": "localhost:8888/mongo/insert",
   "queryParams": [],
   "preRequestScript": null,
   "pathVariables": {},
   "pathVariableData": [],
   "method": "POST",
   "data": [],
   "dataMode": "raw",
   "tests": null,
   "currentHelper": "normal",
   "helperAttributes": {},
   "time": 1618297220124,
   "name": "添加文档",
   "description": "",
   "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
   "responses": [],
   "rawModeData": "
{\n\t\"name\":\"shadow1\",\n\t\"age\":18,\n\t\"addresses\":
[\n\t\t{\n\t\t\t\"province\":\"湖南\",\n\t\t\t\"city\":\"张家界
\",\n\t\t\t\"town\":\"永定区\"\n\t\t},\n\t\t{\n\t\t\t\"province\":\"广东
\",\n\t\t\t\"city\":\"深圳\",\n\t\t\t\"town\":\"龙华区
\"\n\t\t}\n\t\t],\n\t\"emails\":
[\"438111969@qq.com\",\"shadow@163.com\"],\n\t\"phones\":
{\n\t\t\"1\":\"110\",\n\t\t\"2\":\"119\",\n\t\t\"3\":\"114\"\n\t}\n}"
  },
  {
   "id": "ae8480f4-7d84-438a-96d3-e0886d62ceca",
   "headers": "",
   "headerData": [],
   "url": "localhost:8888/mongo/update/shadow",
   "queryParams": [],
   "pathVariables": {},
   "pathVariableData": [],
   "preRequestScript": null,
   "method": "PUT",
   "collectionId": "87c81fb4-d538-0608-7bf1-b2a2b286ffb1",
   "data": null,
   "dataMode": "params",
   "name": "更新文档",
   "description": "",
   "descriptionFormat": "html",
      "time": 1618291236895,
      "version": 2,
      "responses": [],
      "tests": null,
      "currentHelper": "normal",
      "helperAttributes": {}
    }
  ]
}
```

九、副本集

> MongoDB中的副本集是一组维护相同数据集的mongod进程。复制集提供冗余和高可用性，是所有生产部署的基础
>
> Oplog(operations log)是一个特殊的集合，记录所有的对于修改数据库（**新增，修改，删除**）的行为日志，这些日志，被称为Oplog
>
> 两种数据同步方式：
>
> ① **初始同步**： 复制全量的数据从副本集当中的另一个成员哪里
>
> ② **复制**： 从节点在初始同步后连续复制数据
>
> 选举机制：从节点在初始同步后连续复制数据
>
> ```
> 具有投票权的节点之间两两互相发送心跳
> 当5次心跳未收到时判断为节点失联
> 如果主节点失联，从节点会发起选举，选出新的主节点
> 如果从节点失联，不发生新的选举
> 选举算法：RAFT一致性算法。成功的必要条件是大多数投票节点存活
> 副本集中最多可以有50个节点，但具有投票权的节点最多7个
> ```
>
> 副本集特征：
>
> - N 个节点的集群
> - 任何节点可作为主节点
> - **所有写入操作都在主节点上**
> - 自动故障转移
> - 自动恢复



1、一主二从环境搭建(单机不同端口号区分)

- 主机规划

ip	port	role
10.25.175.108	28017	primary
10.25.175.108	28018	secondary
10.25.175.108	28019	secondary

- 下载安装包
- 解压安装包

```
tar -zxvf mongodb-linux-x86_64-rhel62-4.0.23.tgz
```

- 拷贝三份到17/18/19文件夹中

```
cp -r mongodb-linux-x86_64-rhel62-4.0.23 mongo17
cp -r mongodb-linux-x86_64-rhel62-4.0.23 mongo18
cp -r mongodb-linux-x86_64-rhel62-4.0.23 mongo19
```

- 在17/18/19文件夹中创建三个目录

```
cd mongo17/
mkdir data
mkdir logs
mkdir conf
cd ../mongo18/
mkdir data
mkdir logs
mkdir conf
cd ../mongo19/
mkdir data
mkdir logs
mkdir conf
```

- 创建 log 文件

```
touch mongo.log
cp mongo.log ./mongo17/logs/
cp mongo.log ./mongo18/logs/
cp mongo.log ./mongo19/logs/
rm -rf mongo.log
```

- 创建配置文件并配置

```
touch mongod.conf
cp mongod.conf ./mongo17/conf/
cp mongod.conf ./mongo18/conf/
cp mongod.conf ./mongo19/conf/
rm -rf mongod.conf
```

  - 主节点 - 17

    ```
    vim ./mongo17/conf/mongod.conf
    ```

    ```
    systemLog:
      destination: file
      path: "/xx/xxx/mongodb/mongo17/logs/mongo.log"
      logAppend: true
    storage:
      dbPath: "/xx/xxx/mongodb/mongo17/data"
      journal:
        enabled: true
    processManagement:
      fork: true
      pidFilePath: "/xx/xxx/mongodb/mongo17/logs/mongod.pid"
    net:
      bindIp: 0.0.0.0
      port: 28017
    replication:
      replSetName: rs0
    ```

  - 从节点01 - 18

    ```
    vim ./mongo18/conf/mongod.conf
    ```

```
systemLog:
  destination: file
  path: "/xx/xxx/mongodb/mongo18/logs/mongo.log"
  logAppend: true
storage:
  dbPath: "/xx/xxx/mongodb/mongo18/data"
  journal:
    enabled: true
processManagement:
  fork: true
  pidFilePath: "/xx/xxx/mongodb/mongo18/logs/mongod.pid"
net:
  bindIp: 0.0.0.0
  port: 28018
replication:
  replSetName: rs0
```

  - 从节点02 - 19

    ```
    vim ./mongo19/conf/mongod.conf
    ```

    ```
    systemLog:
      destination: file
      path: "/xx/xxx/mongodb/mongo19/logs/mongo.log"
      logAppend: true
    storage:
      dbPath: "/xx/xxx/mongodb/mongo19/data"
      journal:
        enabled: true
    processManagement:
      fork: true
      pidFilePath: "/xx/xxx/mongodb/mongo19/logs/mongod.pid"
    net:
      bindIp: 0.0.0.0
      port: 28019
    replication:
      replSetName: rs0
    ```

- 启动服务

  ```
  ./mongo17/bin/mongod -f ./mongo17/conf/mongod.conf
  ./mongo18/bin/mongod -f ./mongo18/conf/mongod.conf
  ./mongo19/bin/mongod -f ./mongo19/conf/mongod.conf
  ```

- 连接服务

  ```
  ./mongo17/bin/mongo --prot 28017
  ```

- 初始化副本集（**_id 的值 rs0 必须和配置文件中的replication.replSetName保持一致**）

```
>use admin
>rs.initiate({
 _id:"rs0",
 members:[
  {_id:0,host:"10.25.175.108:28017",priority:1},
        {_id:1,host:"10.25.175.108:28018",priority:1},
        {_id:2,host:"10.25.175.108:28019",priority:1}
      ]
})
>rs.status()
>use shadow #切换到shadow库
>db.user.insert({name:"shadow"}) #创建user集合并存储文档
>db.user.find()
```

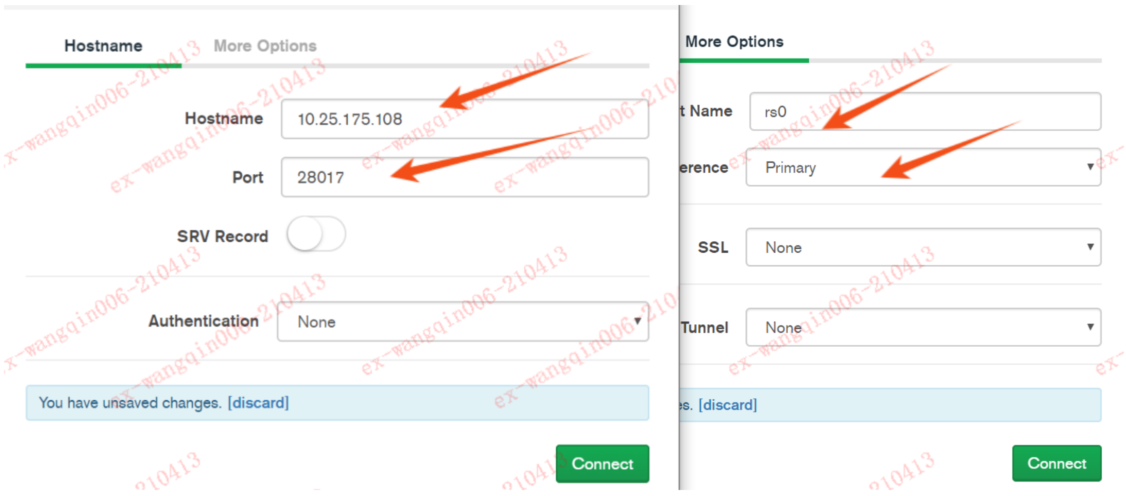
- 连接其他服务，可看见主从情况

```
./mongo18/bin/mongo --port 28018
./mongo19/bin/mongo --port 28019
```

- 分别执行以下命令

```
>db.getMongo().setSecondaryOk() #设置副本节点可以读 或者 rs.salveOk()
>use shadow #切换到shadow库
>db.user.find() #查询user集合中的文档
```

- Compass连接



Hostname: 10.25.175.108
Port: 28017
SRV Record
Authentication: None
You have unsaved changes. [discard]

t Name: rs0
erence: Primary
SSL: None
Tunnel: None
s. [discard]

2、副本集操作

[操作手册]

- 初始化副本集

```
rs.initiate({
 _id:"rs0", #这里和配置文件必须保持一致
 members:[
  {_id:0,host:"10.25.175.108:28017",priority:1},
        {_id:1,host:"10.25.175.108:28018",priority:1},
        {_id:2,host:"10.25.175.108:28019",priority:1}
      ]
})
```

- 添加副本集

```
rs.add({_id:3,host:"10.25.175.108:28020",priority:1})
#添加仲裁节点 rs.addArb(_id:3,host:"10.25.175.108:28020")
```

- 移除副本集

```
rs.remove("10.25.175.108:28018")
```

- 查看副本集配置

```
rs.conf()
```

- 修改副本集配置

```
conf = rs.conf()
conf.members[0].priority=2
rs.reconfig(conf)
```

- 查看副本集状态

```
rs.status()
```

十、事务

> 原子性(A)：事务是最小单位，不可再分（更多关注多行）
> 一致性(C)：事务要求所有的DML语句操作的时候，必须保证同时成功或者同时失败
> 隔离性(I)：事务A和事务B之间具有隔离性
> 持久性(D)：是事务的保证，事务终结的标志(内存的数据持久到硬盘文件中)

- 写事务

> **writeConcern**：
>
> 写关注描述了一次写请求的确认级别，写：包括 向独立的mongod进程，副本集或分片集群 { w: <, j: <, wtimeout: < }
>
> w：决定一个写操作落到多少个节点上才算成功，值包括：
>
> - 0：发起写操作，不关心是否成功
> - 1~集群最大数据节点数： 写操作数据需要全部复制到配置节点上才算成功
> - majority：数据需要被复制到大多数节点上才算成功
> - 默认：0

```
>conf = rs.conf()
>conf.members[1].priority=0 #优先级
>conf.members[1].slaveDelay=10 #延迟10s同步
>rs.reconfig(conf) #重新加载配置
>rs.conf() #查看配置情况
>db.shop.insert({name:"x"}) #马上返回，执行成功
>db.shop.insert({name:"x"},{writeConcern:{w:1}}) #马上返回，执行成功
>db.shop.insert({name:"x"},{writeConcern:{w:2}}) #马上返回，执行成功
>db.shop.insert({name:"x"},{writeConcern:{w:3}}) #等待10s返回，执行成功
>db.shop.insert({name:"x"},{writeConcern:{w:"majority"}}) #马上返回，执行成功
>db.shop.insert({name:"x"},{writeConcern:{w:4}}) #警告，没有足够的数据节点，执行成功
```

- 读事务

> 有别于传统的关系型数据库，mongo天生就是分布式数据库，所以mongo读取数据的时候更关注以下两点：
>
> 1、从哪里读？关注数据节点的位置 - 由 **readPreference** 解决
> 2、什么样的数据可以读？关注数据的隔离性 - 由 **readConcern** 解决
>
> - **readPreference** 决定读取的数据来自那个数据节点，可选值：
> - primary: 只读取主节点
> - primaryPreferred： 优先读取主节点，如果不可用则选择从节点
> - secondary：只选择从节点
> - secondaryPreferred： 优先读取从节点，如果从节点不可用则选择主节点
> - nearest： 选择最近的节点
> - 默认：primary
>
> - **readConcern** 决定这个节点上的数据那些是可读取的，类似关系数据库的隔离级别。可选值包括：
> - availavle: 读取所有可用的数据
> - local: 读取所有可用且数据当前分片的数据
> - majority: 读取在大多数节点上提交完成的数据
> - linearizable: 线性化读取文档
> - snapshot: 读取最近快照中心的数据
> - 默认：local

- 事务操作

```
>s = db.getMongo().startSession()
>s.startTransaction() #开启事务
>s.getDatabase("shadow").shop.insert({name:"wq"})
>s.getDatabase("shadow").user.insert({age:10})
>s.commitTransaction() # 提交或者回滚 s.abortTransaction()
```

十一、Java-MongoDB - 副本集集群

1、pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.70</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-core</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

2、yml

```
server:
  port: 8888
spring:
  data:
    mongodb:
      uri:
mongodb://10.25.175.108:28017,10.25.175.108:28018,10.25.175.108:28019/shadow?
connect=replicaSet&slaveOk=true&replicaSet=rs0&readPreference=secondary
logging:
  level:
    org:
      springframework:
        data:
          mongodb: debug
```

3、java

- TransactionConfig.java

```java
@Configuration
public class TransactionConfig {

    @Bean
    MongoTransactionManager transactionManager(MongoDatabaseFactory factory)
{
        return new MongoTransactionManager(factory);
    }

}
```

- Apple.java

```java
@Data
@AllArgsConstructor
@Document("apples") // 集合名称 apples，需要先创建好
public class Apple {

    private String color;

    private Integer price;

}
```

- Fruit.java

```java
@Data
@AllArgsConstructor
@Document("fruits") // 集合名称 fruits，需要先创建好
public class Fruit {

    private String color;

    private String name;

}
```

- BusinessService.java

```java
public interface BusinessService {

    String insert();
}
```

- BusinessServiceImpl.java

```java
@Service
public class BusinessServiceImpl implements BusinessService {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Transactional
    public String insert() {
        Apple apple = new Apple("红色", 10);
        Fruit fruit = new Fruit("黄色", "香蕉");
        Apple apple1 = mongoTemplate.insert(apple);
        Fruit fruit1 = mongoTemplate.insert(fruit);
        // System.out.println(1 / 0);
        return apple1 + " | " + fruit1;
    }
}
```

- MongoTransactionController.java

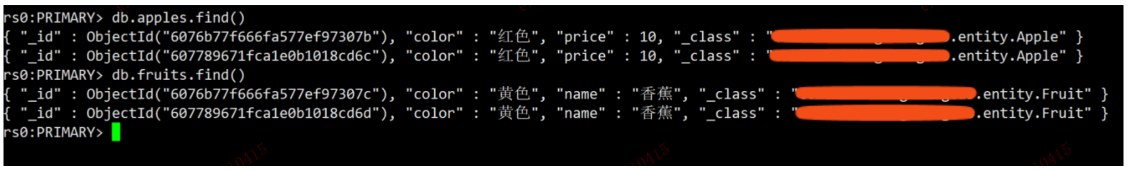
```
@RestController
@RequestMapping("/mongo")
public class MongoTransactionController {

    @Autowired
    private BusinessService businessService;

    /**
     * 执行 Mongo 事务
     */
    @GetMapping("/transaction/insert")
    public String insert() {
        return businessService.insert();
    }

}
```

- 发送请求

```
curl "localhost:8888/mongo/transaction/insert"
```

- 查看 mongodb 数据库存储信息

```
# 连接 mongo 集群中的一台服务
./bin/mongo --port 28017
>use shadow
>show collections
>db.apples.find()
>db.fruits.find()
```



```
rs0:PRIMARY> db.apples.find()
{ "_id" : ObjectId("6076b77f666fa577ef97307b"), "color" : "红色", "price" : 10, "_class" : "[redacted].entity.Apple" }
{ "_id" : ObjectId("607789671fca1e0b1018cd6c"), "color" : "红色", "price" : 10, "_class" : "[redacted].entity.Apple" }
rs0:PRIMARY> db.fruits.find()
{ "_id" : ObjectId("6076b77f666fa577ef97307c"), "color" : "黄色", "name" : "香蕉", "_class" : "[redacted].entity.Fruit" }
{ "_id" : ObjectId("607789671fca1e0b1018cd6d"), "color" : "黄色", "name" : "香蕉", "_class" : "[redacted].entity.Fruit" }
rs0:PRIMARY>
```

十二、索引

> 索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或多列的值进行排序的一种结构

- 创建索引语法

> 语法中 Key 值为你要创建的索引字段，1 为指定按升序创建索引，如果你想按降序来创建索引指定为 -1 即可

```
db.集合名称.createIndex(keys,options)
或者
db.集合名称.ensureIndex(keys,options)
```

options:

参数	类型	描述
background	boolean	是否后台创建索引
unique	boolean	是否唯一索引
name	string	索引名称
expireAfterSeconds	integer	TTL 过期时间

- 案例

```
db.shop.createIndex({price:1,name:-1},{background:
true,expireAfterSeconds:10})
```

- 其他操作

操作	描述	例子
db.集合名称.getIndexes()	查看集合索引	db.shop.getIndexes()
db.集合名称.totalIndexSize()	查看集合索引大小	db.shop.totalIndexSize()
db.集合名称.dropIndexes()	删除集合所有索引	db.shop.dropIndexes()
db.集合名称.dropIndex("索引名称")	删除集合指定索引	db.shop.dropIndex("idx_name")

- 查看查询语句是否使用了索引 **explain()**

> db.集合名称.find({}).explain(verbose)
>
> verbose:
>
> - queryPlanner 默认
> - executionStats
> - allPlansExecution

```
>db.shop.find({name:"zzz",price:{$lte:100}}).explain("allPlansExecution")
```

- 强制使用指定索引 **hint()**

```
>db.shop.createIndex({name:1,price:1},{background:true}) #创建索引
>db.shop.find({name:"zzz"},{_id:0,name:1}).hint({name:1,price:1}).explain()#
强制使用索引
```

- 索引限制
  - 集合中索引不能超过64个
  - 索引名的长度不能超过128个字符
  - 一个复合索引最多可以有31个字段

十三、分片集群

1、分片集群搭建

- 分片角色

角色	作用
Config Server	配置服务
Shard（replica set）	分片服务（副本集）
Router（mongos）	路由服务

- 主机规划

ip	port	role
10.25.175.108	28001	Config Server-1
10.25.175.108	28002	Config Server-2
10.25.175.108	28003	Config Server-3
10.25.175.108	28011	Router-1
10.25.175.108	28012	Router-2
10.25.175.108	28013	Router-3
10.25.175.108	28021	Shard-1-primary
10.25.175.108	28022	Shard-1-secondary
10.25.175.108	28023	Shard-1-secondary
10.25.175.108	28031	Shadr-2-primary
10.25.175.108	28032	Shard-2-secondary
10.25.175.108	28033	Shard-2-secondary
10.25.175.108	28041	Shard-3-primary
10.25.175.108	28042	Shard-3-secondary
10.25.175.108	28043	Shard-3-secondary

- 下载安装包
- 解压安装包

```
tar -zxvf mongodb-linux-x86_64-rhel62-4.0.23.tgz
```

- 重命名并创建文件夹

```
mv mongodb-linux-x86_64-rhel62-4.0.23 mongo
mkdir ./mongo/data
mkdir ./mongo/conf
mkdir ./mongo/logs
touch mongod.conf
cp mongod.conf ./mongo/conf/
rm -rf mongod.conf
touch mongo.log
cp mongo.log ./mongo/logs/
rm -rf mongo.log
```

- 创建目录

```
# configServer
mkdir -p configServer/cfServer28001
mkdir -p configServer/cfServer28002
mkdir -p configServer/cfServer28003
# routerServer
mkdir -p routerServer/rServer28011
mkdir -p routerServer/rServer28012
mkdir -p routerServer/rServer28013
# shard-1
mkdir -p shardServer/shard01/sServer28021
mkdir -p shardServer/shard01/sServer28022
mkdir -p shardServer/shard01/sServer28023
# shard-2
mkdir -p shardServer/shard02/sServer28031
mkdir -p shardServer/shard02/sServer28032
mkdir -p shardServer/shard02/sServer28033
# shard-3
mkdir -p shardServer/shard03/sServer28041
mkdir -p shardServer/shard03/sServer28042
mkdir -p shardServer/shard03/sServer28043
```

- 拷贝文件

```
# configServer
cp -r mongo ./configServer/cfServer28001/
cp -r mongo ./configServer/cfServer28002/
cp -r mongo ./configServer/cfServer28003/
# routerServer
cp -r mongo ./routerServer/rServer28011/
cp -r mongo ./routerServer/rServer28012/
cp -r mongo ./routerServer/rServer28013/
# shard-1
cp -r mongo ./shardServer/shard01/sServer28021
cp -r mongo ./shardServer/shard01/sServer28022
cp -r mongo ./shardServer/shard01/sServer28023
# shard-2
cp -r mongo ./shardServer/shard02/sServer28031
cp -r mongo ./shardServer/shard02/sServer28032
cp -r mongo ./shardServer/shard02/sServer28033
# shard-3
cp -r mongo ./shardServer/shard03/sServer28041
cp -r mongo ./shardServer/shard03/sServer28042
cp -r mongo ./shardServer/shard03/sServer28043
```

- 查看目录结构

  ```
  tree -d ./configServer
  tree -d ./routerServer
  tree -d ./shardServer
  ```

- **配置 configServer**
  - cfServer28001 配置

    ```
    # 28001
    cd configServer/cfServer28001/mongo/conf
    vim mongod.conf
    ```

    配置以下内容

    ```
    pidfilepath =
    /xx/xxx/mongodb/configServer/cfServer28001/mongo/logs/configsrv.pid
    dbpath = /xx/xxx/mongodb/configServer/cfServer28001/mongo/data
    logpath =
    /xx/xxx/mongodb/configServer/cfServer28001/mongo/logs/mongo.log
    logappend = true
    bind_ip = 0.0.0.0
    port = 28001
    fork = true
    #declare this is a config db of a cluster;
    configsvr = true
    #副本集名称
    replSet=rss
    #设置最大连接数
    maxConns=20000
    ```

  - cfServer28002 配置

    ```
    # 28002
    cd configServer/cfServer28002/mongo/conf
    vim mongod.conf
    ```

    配置以下内容

```
pidfilepath =
/xx/xxx/mongodb/configServer/cfServer28002/mongo/logs/configsrv.pid
dbpath = /xx/xxx/mongodb/configServer/cfServer28002/mongo/data
logpath =
/xx/xxx/mongodb/configServer/cfServer28002/mongo/logs/mongo.log
logappend = true
bind_ip = 0.0.0.0
port = 28002
fork = true
#declare this is a config db of a cluster;
configsvr = true
#副本集名称
replSet=rss
#设置最大连接数
maxConns=20000
```

- cfServer28003 配置

  ```
  # 28003
  cd configServer/cfServer28003/mongo/conf
  vim mongod.conf
  ```

  配置以下内容

  ```
  pidfilepath =
  /xx/xxx/mongodb/configServer/cfServer28003/mongo/logs/configsrv.pid
  dbpath = /xx/xxx/mongodb/configServer/cfServer28003/mongo/data
  logpath =
  /xx/xxx/mongodb/configServer/cfServer28003/mongo/logs/mongo.log
  logappend = true
  bind_ip = 0.0.0.0
  port = 28003
  fork = true
  #declare this is a config db of a cluster;
  configsvr = true
  #副本集名称
  replSet=rss
  #设置最大连接数
  maxConns=20000
  ```

- 启动 三台 configServer

  ```
  ./cfServer28001/mongo/bin/mongod -f
  ./cfServer28001/mongo/conf/mongod.conf
  ./cfServer28002/mongo/bin/mongod -f
  ./cfServer28002/mongo/conf/mongod.conf
  ./cfServer28003/mongo/bin/mongod -f
  ./cfServer28003/mongo/conf/mongod.conf
  ```

- 连接 configServer 并初始化集群

```
./cfServer28001/mongo/bin/mongo --port 28001
>use admin
>rs.initiate({
 _id:"rss",
 members:[
  {_id:0,host:"10.25.175.108:28001"},
  {_id:1,host:"10.25.175.108:28002"},
  {_id:2,host:"10.25.175.108:28003"}
    ]
})
```

- **配置分片集群**
  - 配置 shard-1
    - shard01-28021 配置

      ```
      # 28021
      cd shardServer/shard01/sServer28021/mongo/conf
      vim mongod.conf
      ```

      配置以下内容

      ```
      pidfilepath =
      /xx/xxx/mongodb/shardServer/shard01/sServer28021/mongo/logs/shard1.
      pid
      dbpath =
      /xx/xxx/mongodb/shardServer/shard01/sServer28021/mongo/data
      logpath =
      /xx/xxx/mongodb/shardServer/shard01/sServer28021/mongo/logs/mongo.l
      og
      logappend = true
      bind_ip = 0.0.0.0
      port = 28021
      fork = true
      #打开web监控
      #httpinterface=true
      #rest=true
      #副本集名称
      replSet=shard1
      #declare this is a shard db of a cluster;
      shardsvr = true
      #设置最大连接数
      maxConns=20000
      ```

    - shard01-28022 配置

      ```
      # 28022
      cd shardServer/shard01/sServer28022/mongo/conf
      vim mongod.conf
      ```

      配置以下内容

      ```
      pidfilepath =
      /xx/xxx/mongodb/shardServer/shard01/sServer28022/mongo/logs/shard1.
      pid
      ```

```
dbpath =
/xx/xxx/mongodb/shardServer/shard01/sServer28022/mongo/data
logpath =
/xx/xxx/mongodb/shardServer/shard01/sServer28022/mongo/logs/mongo.l
og
logappend = true
bind_ip = 0.0.0.0
port = 28022
fork = true
#打开web监控
#httpinterface=true
#rest=true
#副本集名称
replSet=shard1
#declare this is a shard db of a cluster;
shardsvr = true
#设置最大连接数
maxConns=20000
```

- shard01-28023

  ```
  # 28023
  cd shardServer/shard01/sServer28023/mongo/conf
  vim mongod.conf
  ```

  配置以下内容

  ```
  pidfilepath =
  /xx/xxx/mongodb/shardServer/shard01/sServer28023/mongo/logs/shard1.
  pid
  dbpath =
  /xx/xxx/mongodb/shardServer/shard01/sServer28023/mongo/data
  logpath =
  /xx/xxx/mongodb/shardServer/shard01/sServer28023/mongo/logs/mongo.l
  og
  logappend = true
  bind_ip = 0.0.0.0
  port = 28023
  fork = true
  #打开web监控
  #httpinterface=true
  #rest=true
  #副本集名称
  replSet=shard1
  #declare this is a shard db of a cluster;
  shardsvr = true
  #设置最大连接数
  maxConns=20000
  ```

- 启动三台 shard-1 服务

```
./sServer28021/mongo/bin/mongod -f
./sServer28021/mongo/conf/mongod.conf
./sServer28022/mongo/bin/mongod -f
./sServer28022/mongo/conf/mongod.conf
./sServer28023/mongo/bin/mongod -f
./sServer28023/mongo/conf/mongod.conf
```

- 连接 shard-1集群并初始化集群

```
./sServer28021/mongo/bin/mongo --port 28021
>use admin
>rs.initiate({
 _id:"shard1",
 members:[
 {_id:0,host:"10.25.175.108:28021"},
 {_id:1,host:"10.25.175.108:28022"},
 {_id:2,host:"10.25.175.108:28023"}
 ]
})
```

- 配置 shard-2
  - shard02-28031

```
# 28031
cd shardServer/shard02/sServer28031/mongo/conf
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/xx/xxx/mongodb/shardServer/shard02/sServer28031/mongo/logs/shard2.
pid
dbpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28031/mongo/data
logpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28031/mongo/logs/mongo.l
og
logappend = true
bind_ip = 0.0.0.0
port = 28031
fork = true
#打开web监控
#httpinterface=true
#rest=true
#副本集名称
replSet=shard2
#declare this is a shard db of a cluster;
shardsvr = true
#设置最大连接数
maxConns=20000
```

  - shard02-28032

```
# 28032
cd shardServer/shard02/sServer28032/mongo/conf
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/xx/xxx/mongodb/shardServer/shard02/sServer28032/mongo/logs/shard2.
pid
dbpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28032/mongo/data
logpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28032/mongo/logs/mongo.l
og
logappend = true
bind_ip = 0.0.0.0
port = 28032
fork = true
#打开web监控
#httpinterface=true
#rest=true
#副本集名称
replSet=shard2
#declare this is a shard db of a cluster;
shardsvr = true
#设置最大连接数
maxConns=20000
```

- shard02-28033

```
# 28033
cd shardServer/shard02/sServer28033/mongo/conf
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/xx/xxx/mongodb/shardServer/shard02/sServer28033/mongo/logs/shard2.
pid
dbpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28033/mongo/data
logpath =
/xx/xxx/mongodb/shardServer/shard02/sServer28033/mongo/logs/mongo.l
og
logappend = true
bind_ip = 0.0.0.0
port = 28033
fork = true
#打开web监控
#httpinterface=true
#rest=true
#副本集名称
replSet=shard2
#declare this is a shard db of a cluster;
shardsvr = true
```

```
#设置最大连接数
maxConns=20000
```

- 启动三台 shard-2 服务

```
./sServer28031/mongo/bin/mongod -f
./sServer28031/mongo/conf/mongod.conf
./sServer28032/mongo/bin/mongod -f
./sServer28032/mongo/conf/mongod.conf
./sServer28033/mongo/bin/mongod -f
./sServer28033/mongo/conf/mongod.conf
```

- 连接 shard-2 集群并初始化集群

```
./sServer28031/mongo/bin/mongo --port 28031
>use admin
>rs.initiate({
 _id:"shard2",
 members:[
  {_id:0,host:"10.25.175.108:28031"},
  {_id:1,host:"10.25.175.108:28032"},
  {_id:2,host:"10.25.175.108:28033"}
 ]
})
```

- 配置 shard-3
    - shard03-28041

```
# 28041
cd shardServer/shard03/sServer28041/mongo/conf
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/wls/wls81/mongodb/shardServer/shard03/sServer28041/mongo/logs/shar
d3.pid
dbpath =
/wls/wls81/mongodb/shardServer/shard03/sServer28041/mongo/data
logpath =
/wls/wls81/mongodb/shardServer/shard03/sServer28041/mongo/logs/mong
o.log
logappend = true
bind_ip = 0.0.0.0
port = 28041
fork = true
#打开web监控
#httpinterface=true
#rest=true
#副本集名称
replSet=shard3
#declare this is a shard db of a cluster;
shardsvr = true
```

```
#设置最大连接数
maxConns=20000
```

- shard03-28042

  ```
  # 28042
  cd shardServer/shard03/sServer28042/mongo/conf
  vim mongod.conf
  ```

  配置以下内容

  ```
  pidfilepath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28042/mongo/logs/shard3.
  pid
  dbpath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28042/mongo/data
  logpath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28042/mongo/logs/mongo.l
  og
  logappend = true
  bind_ip = 0.0.0.0
  port = 28042
  fork = true
  #打开web监控
  ###httpinterface=true
  ###rest=true
  ###副本集名称
  replSet=shard3
  ###declare this is a shard db of a cluster;
  shardsvr = true
  ###设置最大连接数
  maxConns=20000
  ```

- shard03-28043

  ```
  # 28043
  cd shardServer/shard03/sServer28043/mongo/conf
  vim mongod.conf
  ```

  配置以下内容

  ```
  pidfilepath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28043/mongo/logs/shard3.
  pid
  dbpath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28043/mongo/data
  logpath =
  /xx/xxx/mongodb/shardServer/shard03/sServer28043/mongo/logs/mongo.l
  og
  logappend = true
  bind_ip = 0.0.0.0
  port = 28043
  fork = true
  #打开web监控
  ###httpinterface=true
  ```

```
###rest=true
###副本集名称
replSet=shard3
###declare this is a shard db of a cluster;
shardsvr = true
###设置最大连接数
maxConns=20000
```

- 启动三台 shard-3 服务

```
./sServer28041/mongo/bin/mongod -f
./sServer28041/mongo/conf/mongod.conf
./sServer28042/mongo/bin/mongod -f
./sServer28042/mongo/conf/mongod.conf
./sServer28043/mongo/bin/mongod -f
./sServer28043/mongo/conf/mongod.conf
```

- 连接 shard-3 集群并初始化集群

```
./sServer28041/mongo/bin/mongo --port 28041
>use admin
>rs.initiate({
 _id:"shard3",
 members:[
  {_id:0,host:"10.25.175.108:28041"},
  {_id:1,host:"10.25.175.108:28042"},
  {_id:2,host:"10.25.175.108:28043"}
 ]
})
```

- **配置 router （mongos）**
  - rServer- 28011 配置

```
# 28011
cd routerServer/rServer28011/mongo/conf/
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/xx/xxx/mongodb/routerServer/rServer28011/mongo/logs/mongos.pid
logpath =
/xx/xxx/mongodb/routerServer/rServer28011/mongo/logs/mongos.log
logappend = true
bind_ip = 0.0.0.0
port = 28011
fork = true
#监听的配置服务器,只能有1个或者3个 rss 为配置服务器的副本集名字
configdb =
rss/10.25.175.108:28001,10.25.175.108:28002,10.25.175.108:28003
##设置最大连接数
maxConns=20000
```

  - rServer-28012 配置

```
# 28012
cd routerServer/rServer28012/mongo/conf/
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/wls/wls81/mongodb/routerServer/rServer28012/mongo/logs/mongos.pid
logpath =
/wls/wls81/mongodb/routerServer/rServer28012/mongo/logs/mongos.log
logappend = true
bind_ip = 0.0.0.0
port = 28012
fork = true
#监听的配置服务器,只能有1个或者3个 rss 为配置服务器的副本集名字
configdb =
rss/10.25.175.108:28001,10.25.175.108:28002,10.25.175.108:28003
##设置最大连接数
maxConns=20000
```

- rServer-28013 配置

```
# 28013
cd routerServer/rServer28013/mongo/conf/
vim mongod.conf
```

配置以下内容

```
pidfilepath =
/wls/wls81/mongodb/routerServer/rServer28013/mongo/logs/mongos.pid
logpath =
/wls/wls81/mongodb/routerServer/rServer28013/mongo/logs/mongos.log
logappend = true
bind_ip = 0.0.0.0
port = 28013
fork = true
#监听的配置服务器,只能有1个或者3个 rss 为配置服务器的副本集名字
configdb =
rss/10.25.175.108:28001,10.25.175.108:28002,10.25.175.108:28003
##设置最大连接数
maxConns=20000
```

- 启动三台 mongos 服务

```
./rServer28011/mongo/bin/mongos -f
./rServer28011/mongo/conf/mongod.conf
./rServer28012/mongo/bin/mongos -f
./rServer28012/mongo/conf/mongod.conf
./rServer28013/mongo/bin/mongos -f
./rServer28013/mongo/conf/mongod.conf
```

- 启用分片

```
./rServer28011/mongo/bin/mongo --port 28011
mongos>use admin
mongos>sh.addShard("shard1/10.25.175.108:28021,10.25.175.108:28022,10.2
5.175.108:28023")
mongos>sh.addShard("shard2/10.25.175.108:28031,10.25.175.108:28032,10.2
5.175.108:28033")
mongos>sh.addShard("shard3/10.25.175.108:28041,10.25.175.108:28042,10.2
5.175.108:28043")
mongos>sh.status()
```

- 测试

```
./rServer28011/mongo/bin/mongo --port 28011
mongos>use admin
mongos>db.runCommand({enablesharding : "shadow"}) #指定shadow分片生效
mongos>db.runCommand({shardcollection : "shadow.shop",key :
{name:"hashed"}}) #指定数据库里需要分片的集合和片键 hash 或 范围 {id:1}
mongos>use shadow
mongos>for(var i = 1;i <= 10000) db.shop.save({id:i,name:"wq"+i})
mongos>db.shop.stats()
```

十四、Java-MongoDB - 分片集群

1、pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
    <version>2.3.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.70</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-core</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

2、yml

```yaml
server:
  port: 8888
spring:
  data:
    mongodb:
      # 连接的是 mongos 的路由服务
      uri: 
mongodb://10.25.175.108:28011,10.25.175.108:28012,10.25.175.108:28013/shadow
logging:
  level:
    org:
      springframework:
        data:
          mongodb: debug
```

3、java

- Shop.java

  ```java
  @Data
  @Accessors(chain = true)
  @Document(collection = "shop")
  public class Shop {

      private int id;

      private String name;

  }
  ```

- ShopRepositiry.java

  ```java
  // Shop类型，主键 Integer 类型
  public interface ShopRepository extends MongoRepository<Shop,Integer> {
  }
  ```

- BusinessService.java

  ```java
  public interface BusinessService {

      void insert();
  }
  ```

- BusinessServiceImpl.java

  ```java
  @Service
  public class BusinessServiceImpl implements BusinessService {

      @Autowired
  ```

```
    private ShopRepository shopRepository;

    @Autowired
    private MongoTemplate mongoTemplate;

    public void insert() {
        Shop shop;
        List<Shop> list = new ArrayList<>();
        for (int i = 1; i <= 1000; i++) {
            shop = new Shop();
            shop.setId(i).setName("shadow" + i);
            list.add(shop);
        }
        shopRepository.saveAll(list);
    }
}
```

- MongoShardController.java

```
@RestController
@RequestMapping("/mongo")
public class MongoShardController {

    @Autowired
    private BusinessService businessService;

    /**
     * 分片集群测试
     */
    @GetMapping("/shard/insert")
    public void insert() {
        businessService.insert();
    }

}
```

- 测试

```
curl "localhost:8888/mongo/shard/insert"
```

- 查看数据分布情况

```
# 路由服务
./rServer28011/mongo/bin/mongo --port 28011
>use shadow
>db.shop.count() # 总记录 = shard1 + shard2 + shard3
```

```
# 分片服务shard-1
# 如果连接的是 Secondary，先执行 rs.slaveOk()
./bin/mongo --port 28021
shard1:PRIMARY>use shadow
shard1:PRIMARY>db.shop.count()
```

```
# 分片服务shard-2
# 如果连接的是 Secondary，先执行 rs.slaveOk()
./bin/mongo --port 28031
shard2:PRIMARY>use shadow
shard2:PRIMARY>db.shop.count()
```

```
# 分片服务shard-3
# 如果连接的是 Secondary，先执行 rs.slaveOk()
./bin/mongo --port 28041
shard3:PRIMARY>use shadow
shard3:PRIMARY>db.shop.count()
```