HW3

0. Refs

```
https://dblab.xmu.edu.cn/blog/833/#more-833
```

1. MongoDB下载安装

不建议按照林子雨的教程做，它的教程太老了，现在mongoDB已经有7.0版本了，而且他仅针对ubuntu 14.04，这是ubuntu很老旧的一个release了。

建议按照官网的教程来：

```
https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on
```

2. 启动MongoDB Shell

7.0版本已经把shell启动命令换了，从mongo变成了：

```
mongosh
```

2.1. MongoDB Shell 常用命令

1. `show dbs`:显示数据库列表
2. `show collections`：显示当前数据库中的集合（类似关系数据库中的表table）
3. `show users`：显示所有用户
4. `use yourDB`：切换当前数据库至yourDB/创建数据库yourDB
5. `db.help()` ：显示数据库操作命令
6. `db.yourCollection.help()` ：显示集合操作命令，yourCollection是集合名

2.2. Shell命令试用

1. 创建博客Blog数据库

```
use Blog
```

2. 创建article表/集合

```
db.createCollection('article')
```

3. 展示库中的集合

```
show collection
```

4. 插入数据：

```
db.article.insert({_id:1,title:'article1',view:'10'})
```

此处_id可选。

在林子雨的教程中，有一个save命令，不过那是mongoDB 3.28版本了，我7.0版本这个命令已经没了

```
test> use Blog
switched to db Blog
Blog> db.createCollection('article')
{ ok: 1 }
Blog> show collections
article
Blog> db.article.insert({_id:1,title:'article1',view:'10'})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1 } }
Blog> db.article.find()
[ { _id: 1, title: 'article1', view: '10' } ]
Blog> db.article.save({_id:10,title:'article1',view:'11'})
TypeError: db.article.save is not a function
```

5. 一次性插入多条数据

```
dataSet=[{name:'a2',view:'100'},{view:'0'}]
db.article.insert(dataSet)
```

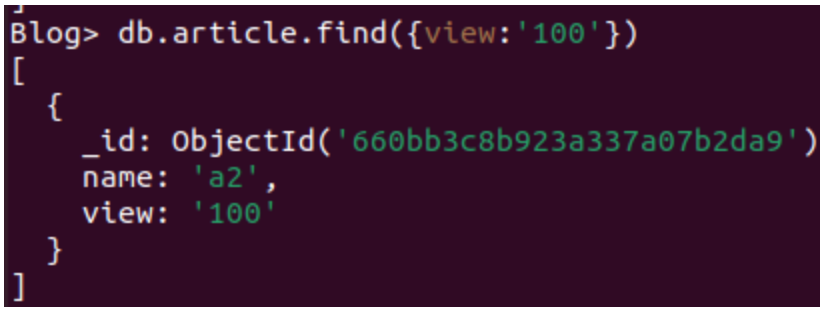
```
Blog> dataSet=[{name:'a2',view:'100'},{view:'0'}]
[ { name: 'a2', view: '100' }, { view: '0' } ]
Blog> db.article.insert(dataSet)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660bb3c8b923a337a07b2da9'),
    '1': ObjectId('660bb3c8b923a337a07b2daa')
  }
}
Blog> db.article.find()
[
  { _id: 1, title: 'article1', view: '10' },
  {
    _id: ObjectId('660bb3c8b923a337a07b2da9'),
    name: 'a2',
    view: '100'
  },
  { _id: ObjectId('660bb3c8b923a337a07b2daa'), view: '0' }
]
```

6. 查找数据

```
db.youCollection.find(criteria, filterDisplay)
```

criteria ：查询条件，可选

filterDisplay：筛选显示部分数据，如显示指定列数据，可选（当选择时，第一个参数不可省略，若查询条件为空，可用{}做占位符，如下例第三句）



7. 修改数据

```
db.youCollection.update(criteria, objNew, upsert, multi )
```

criteria: update的查询条件，类似sql update查询内where后面的

objNew : update的对象和一些更新的操作符（如

```
$set
```

）等，也可以理解为sql update查询内set后面的。

upsert : 如果不存在update的记录，是否插入objNew，true为插入，默认是false，不插入。

multi: mongodb默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。默认false，只修改匹配到的第一条数据。

其中criteria和objNew是必选参数，upsert和multi可选参数

这里相当于sql中的：

```
update article set title='article_dummy' where view = '100'
```

8. 删除数据

```
db.yourCollection.remove(criteria)
```

9. 删除集合

```
db.yourCollection.drop()
```

10. 退出shell

```
exit
```

3. Java API编程

3.1. 导入依赖库

林子雨发的那个maven仓库的mongo-java-driver的链接已经失效了，所以我就没有采用jar包的方式，而是直接用Maven管理了依赖库：

```
  <dependencies>
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver-sync</artifactId>
      <version>5.0.0</version>
    </dependency>
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver-core</artifactId>
      <version>5.0.0</version>
    </dependency>
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver-core</artifactId>
      <version>5.0.0</version>
    </dependency>
  </dependencies>
```

我是最新的MongoDB，7.0的，配套的driver是5.0.0的。

然后在新的driver版本里，林子雨给的那段代码里面有些库函数已经变了，比如MongoClient这个类已经变成了抽象类，官方采用工厂模式，给了一个MongoClients的工厂，通过这个工厂创建客户端与数据库的连接。

改动后的代码如下：

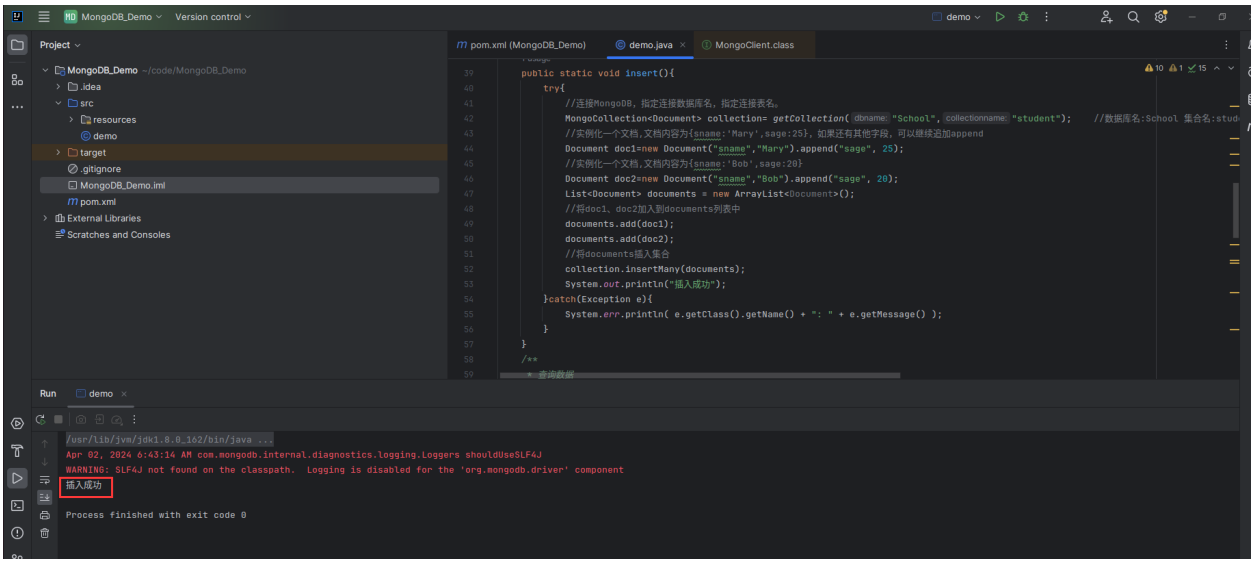
```
import java.util.ArrayList;
import java.util.List;

import com.mongodb.client.*;
import org.bson.Document;
import com.mongodb.client.model.Filters;

public class demo {
```

```
    /**
     * @param args
     */
    public static void main(String[] args) {
      insert();//插入数据。执行插入时，可将其他三句函数调用语句注释，下同
//        find(); //查找数据
//      update();//更新数据
//      delete();//删除数据
    }
    /**
     * 返回指定数据库中的指定集合
     * @param dbname 数据库名
     * @param collectionname 集合名
     * @return
     */
    //MongoDB无需预定义数据库和集合,在使用的时候会自动创建
    public static MongoCollection<Document> getCollection(String
        // MongoDB Java Driver 5.0.0 使用 MongoClients 工厂类来创建
        MongoClient mongoClient = MongoClients.create();
        // 获取数据库实例
        MongoDatabase mongoDatabase = mongoClient.getDatabase(dbn
        // 获取集合实例
        MongoCollection<Document> collection = mongoDatabase.getC
        return collection;
    }
    /**
     * 插入数据
     */
    public static void insert(){
        try{
            //连接MongoDB，指定连接数据库名，指定连接表名。
            MongoCollection<Document> collection= getCollection(
            //实例化一个文档,文档内容为{sname:'Mary',sage:25}，如果还
            Document doc1=new Document("sname","Mary").append("s
            //实例化一个文档,文档内容为{sname:'Bob',sage:20}
            Document doc2=new Document("sname","Bob").append("sa
```

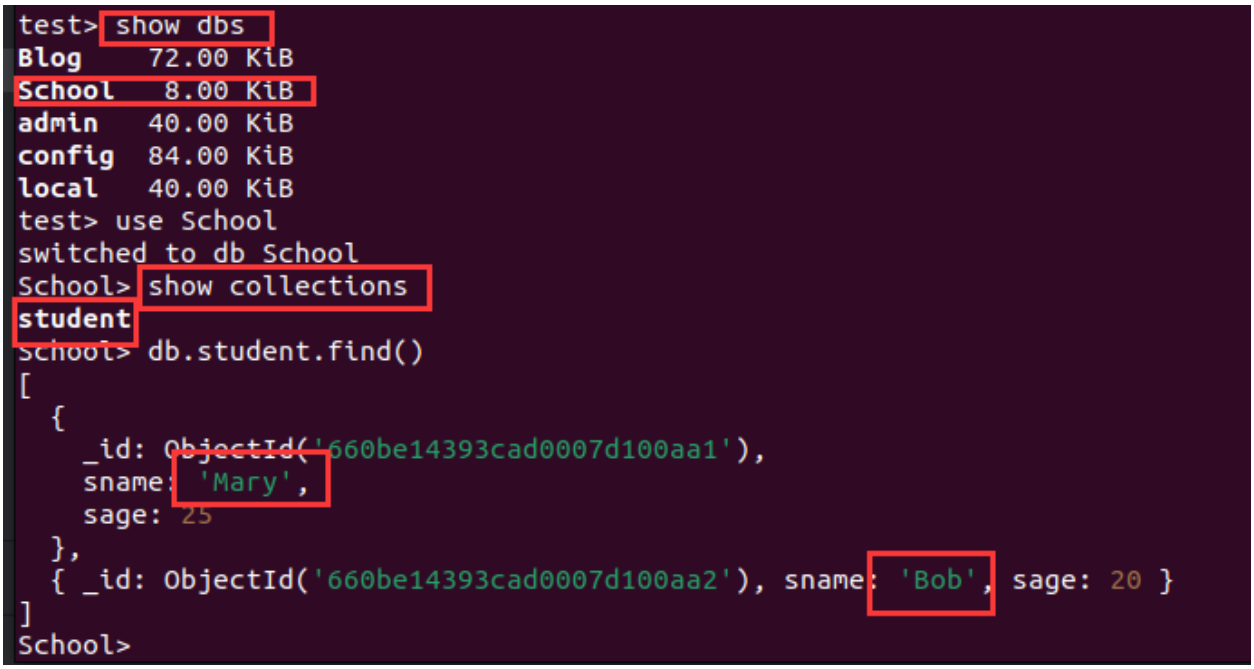
```
            List<Document> documents = new ArrayList<Document>(
            //将doc1、doc2加入到documents列表中
            documents.add(doc1);
            documents.add(doc2);
            //将documents插入集合
            collection.insertMany(documents);
            System.out.println("插入成功");
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " +
        }
    }
    /**
     * 查询数据
     */
    public static void find(){
        try{
            MongoCollection<Document> collection = getCollectio
            //通过游标遍历检索出的文档集合
//          MongoCursor<Document>  cursor= collection.find(new B
            //查询所有数据
            MongoCursor<Document>  cursor= collection.find().ite
            while(cursor.hasNext()){
                System.out.println(cursor.next().toJson());
            }
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " +
        }
    }
    /**
     * 更新数据
     */
    public static void update(){
        try{
            MongoCollection<Document> collection = getCollectio
            //更新文档   将文档中sname='Mary'的文档修改为sage=22
            collection.updateMany(Filters.eq("sname", "Mary"), n
```

```
            System.out.println("更新成功！");
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " +
        }
    }
    /**
     * 删除数据
     */
    public static void delete(){
        try{
            MongoCollection<Document> collection = getCollectio
            //删除符合条件的第一个文档
            collection.deleteOne(Filters.eq("sname", "Bob"));
            //删除所有符合条件的文档
            //collection.deleteMany (Filters.eq("sname", "Bob")
            System.out.println("删除成功！");
        }catch(Exception e){
            System.err.println( e.getClass().getName() + ": " +
        }
    }
}
```

随后就可以解开注释，进行一些curd的操作了。比如下图就是insert



然后我们可以打开shell看看是否真正插入成功了。



插入成功。