**MongoDB Compass Community**

1. **下载**

要从具有图形用户界面的MongoDB开始，MongoDB Compass Community是最好的方法之一。MongoDB Compass Community由MongoDB开发人员开发，这意味着更高的可靠性和兼容性。它为MongoDB提供GUI mongodb工具，以探索数据库交互；具有完整的CRUD功能并提供可视方式。借助内置模式可视化，用户可以分析文档并显示丰富的结构。为了监控服务器的负载，它提供了数据库操作的实时统计信息。就像MongoDB一样，Compass也有两个版本，一个是Enterprise（付费），社区可以免费使用。适用于Linux，Mac或Windows。进入官网的下载页面，下载相应的版本。

下载链接：<https://www.mongodb.com/download-center/compass?jmp=hero>。

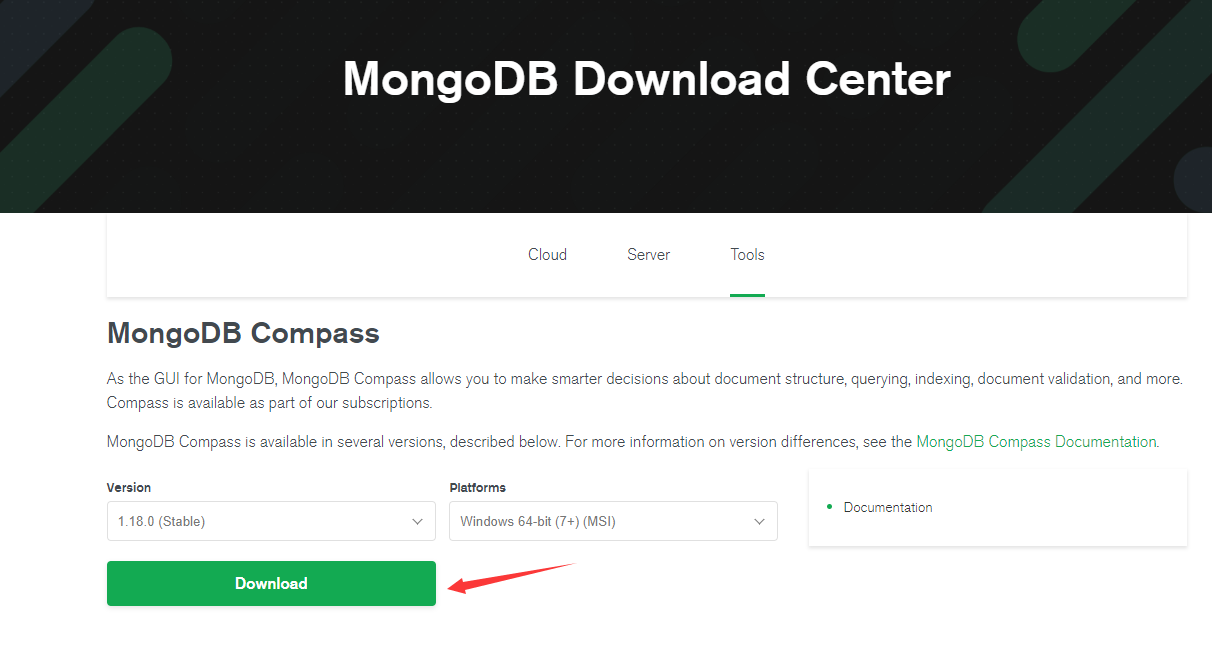


图1 工具下载图

1. **安装**

1）打开下载好的.msi安装包，如图所示，点击next进行安装。

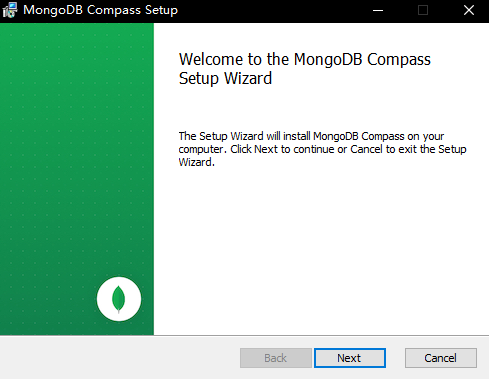


图2 工具安装图

2）选择安装磁盘位置。

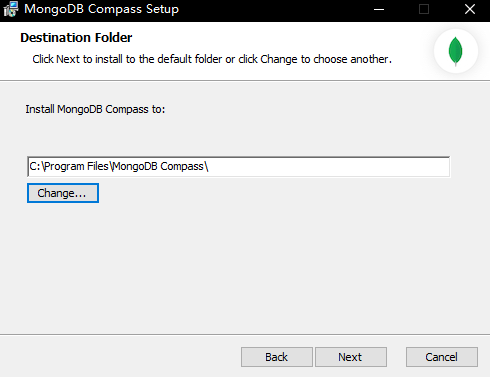


图3 工具安装图

3）安装完成。

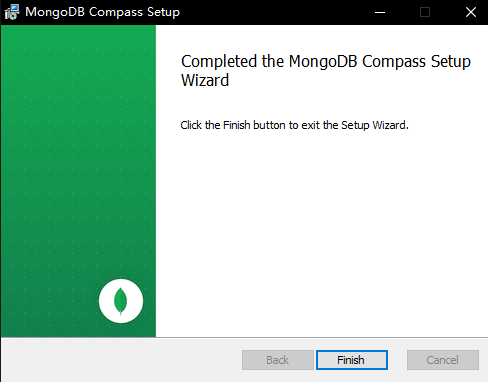


图4 工具安装图

4）打开安装目录，运行可视化工具。



图5 文件图标图

5）打开服务器的mongodb配置文件，修改绑定ip地址为0.0.0.0。

命令：vi /etc/mongod.conf

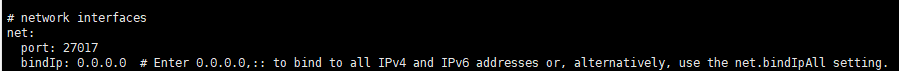


图6 ip地址修改图

6）使用MongoDB Compass Community远程连接，填入相应的ip与port。最

后点击连接。

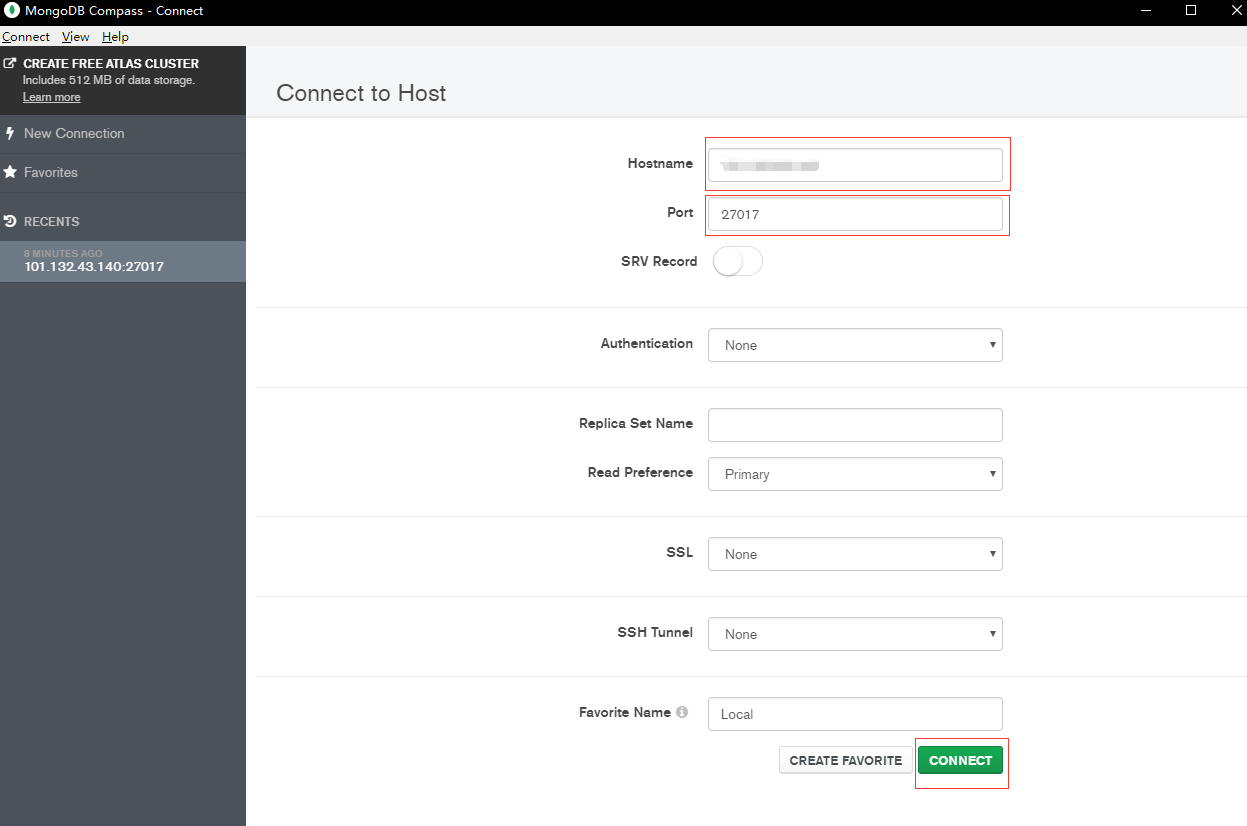


图7 mongodb连接图

1. **使用**

**1）介绍**

连接后的界面如图所示，列出来的为现存的数据库，默认已有admin,

config与local数据库。

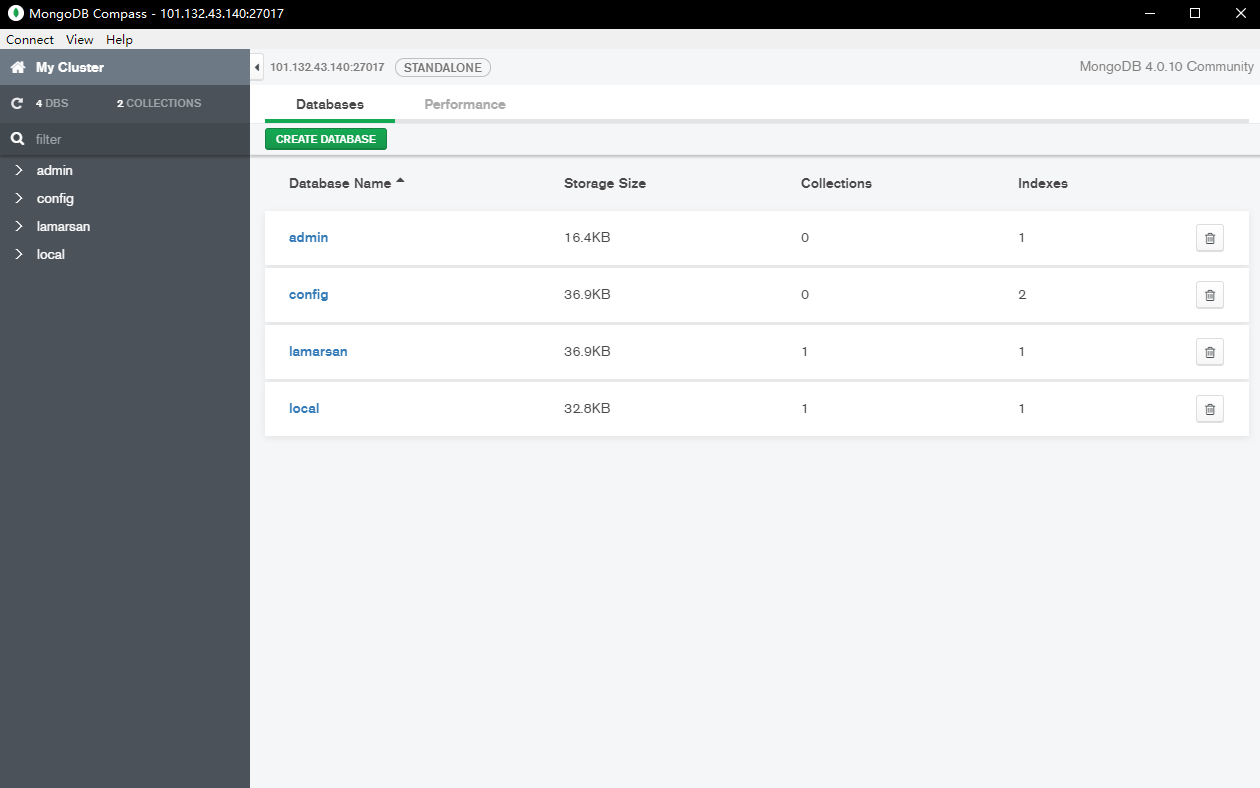


图8 mongodb数据库显示图

点击对应的数据库可以查看该数据库信息，列出该数据库所有的集合。下图所示为lamarsan数据库中的集合——lamarsan。该界面显示了lamarsan集合的平均文档大小，总大小，索引数量以及索引大小等信息。

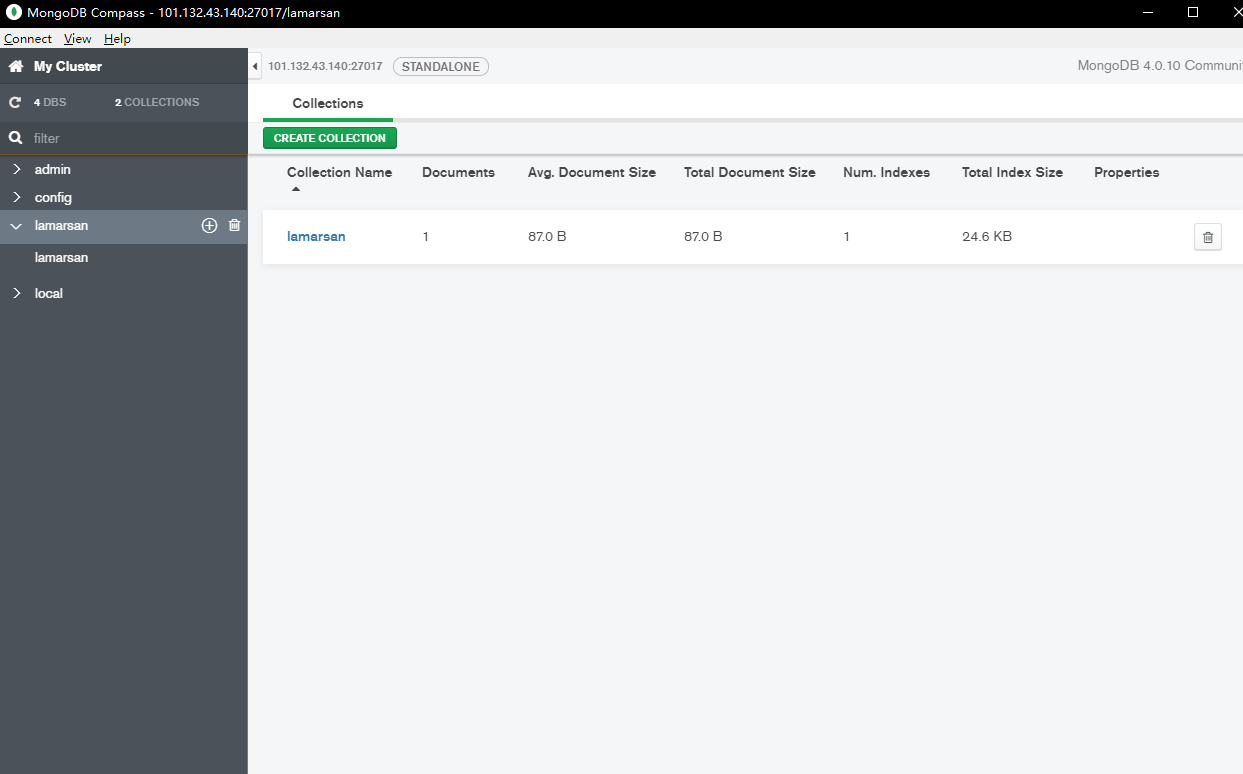


图9 mongodb集合显示图

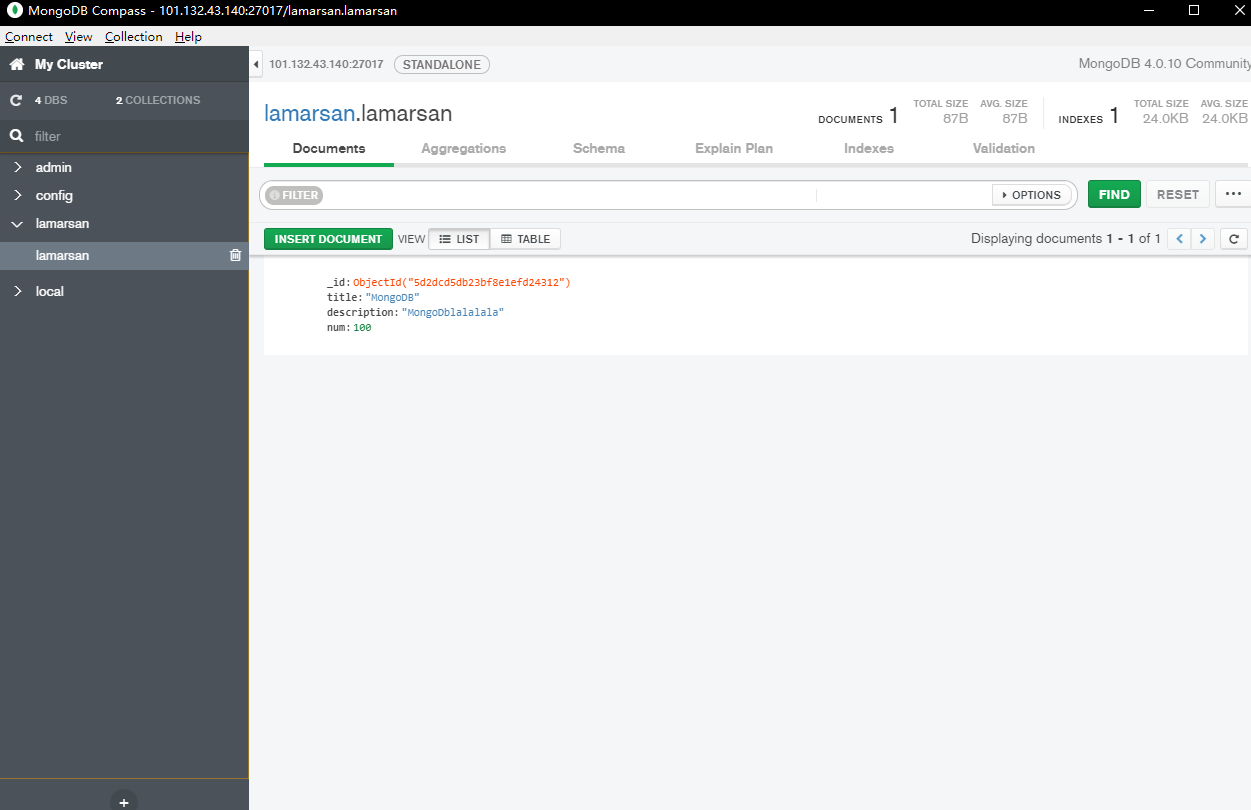
点击对应的集合可以查看该集合信息，如图所示，该集合内只有一个文档。同时，在该页面可以进行文档的增删改查。

图10 mongodb文档显示图

**2）MongoDB概念介绍**

下面列出了mongodb与mysql之间的关系比较。

|  |  |  |
| --- | --- | --- |
| SQL术语/概念 | MongoDB术语/概念 | 解释说明 |
| database | database | 数据库 |
| table | collection | 数据库表/集合 |
| row | document | 数据记录行/文档 |
| column | field | 数据字段/域 |
| index | index | 索引 |
| table joins |  | 表连接，mongodb不支持 |
| primary key | primary key | 主键,MongoDB自动将\_id字段设置为主键 |

**3）新建数据库**

进入首页，点击CREAWTE DATABASE。填入相应的数据库信息和集合信息即可，Capped Collection为固定大小的集合。

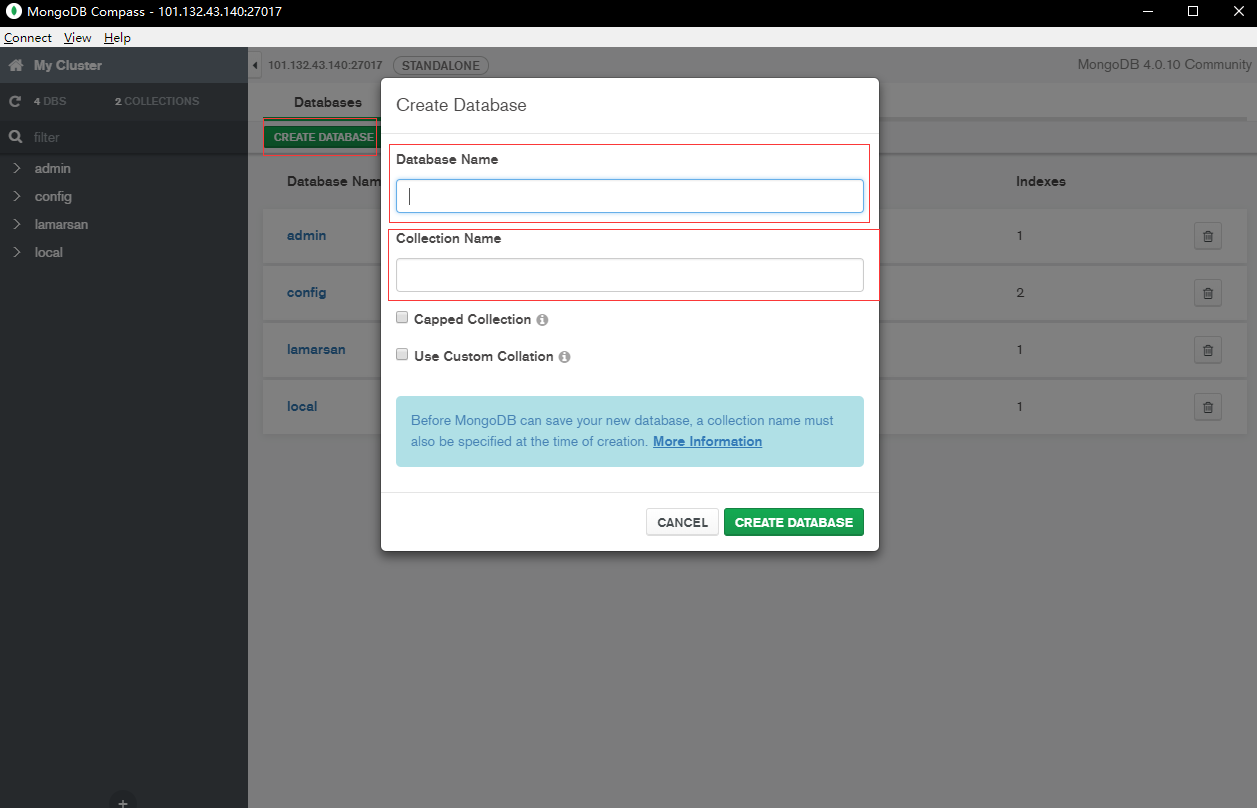


图10 创建数据库图

**4）删除数据库**

点击数据库名后的垃圾图标可进行删除，删除时输入数据库名字，即lama

-r。

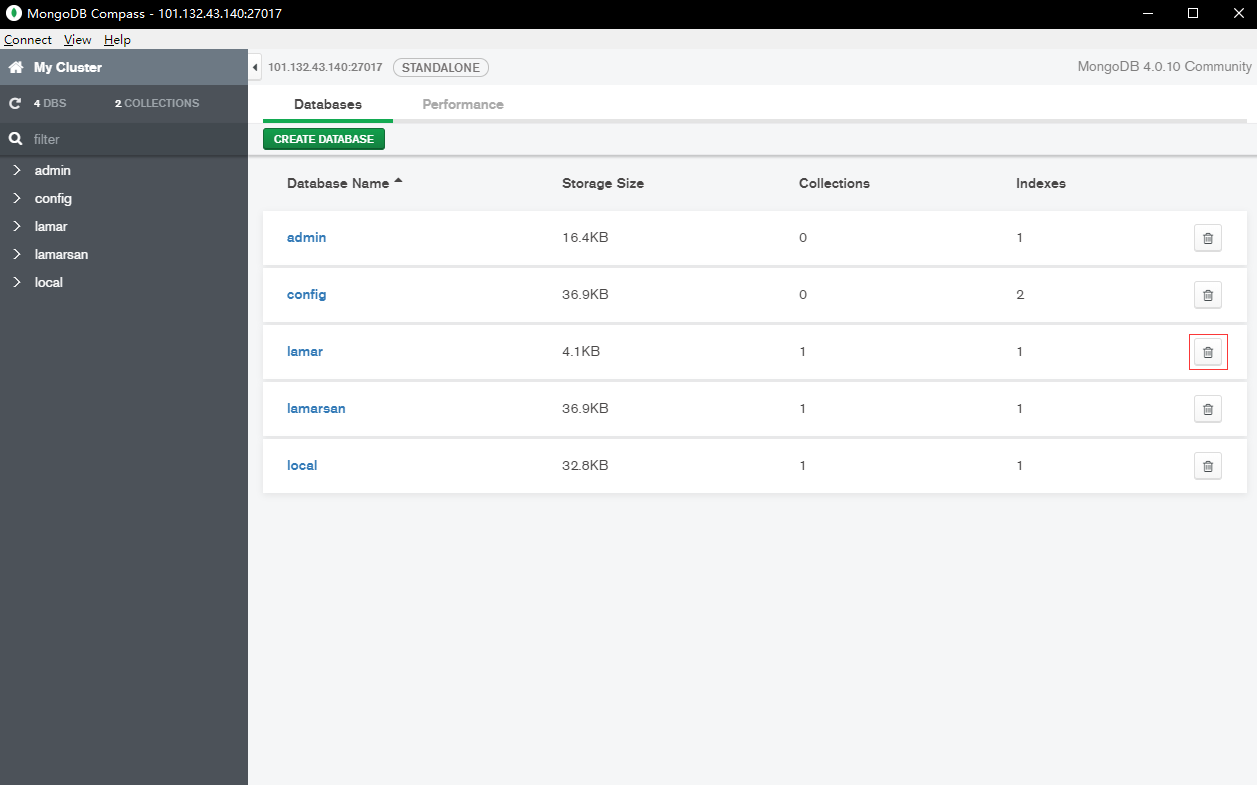


图11 删除数据库图

**5）新建集合**

进入数据库页面，点击集合名字右边的+号或者CREATE COLLECTION,填入相应的集合信息即可，Capped Collection为固定大小的集合。

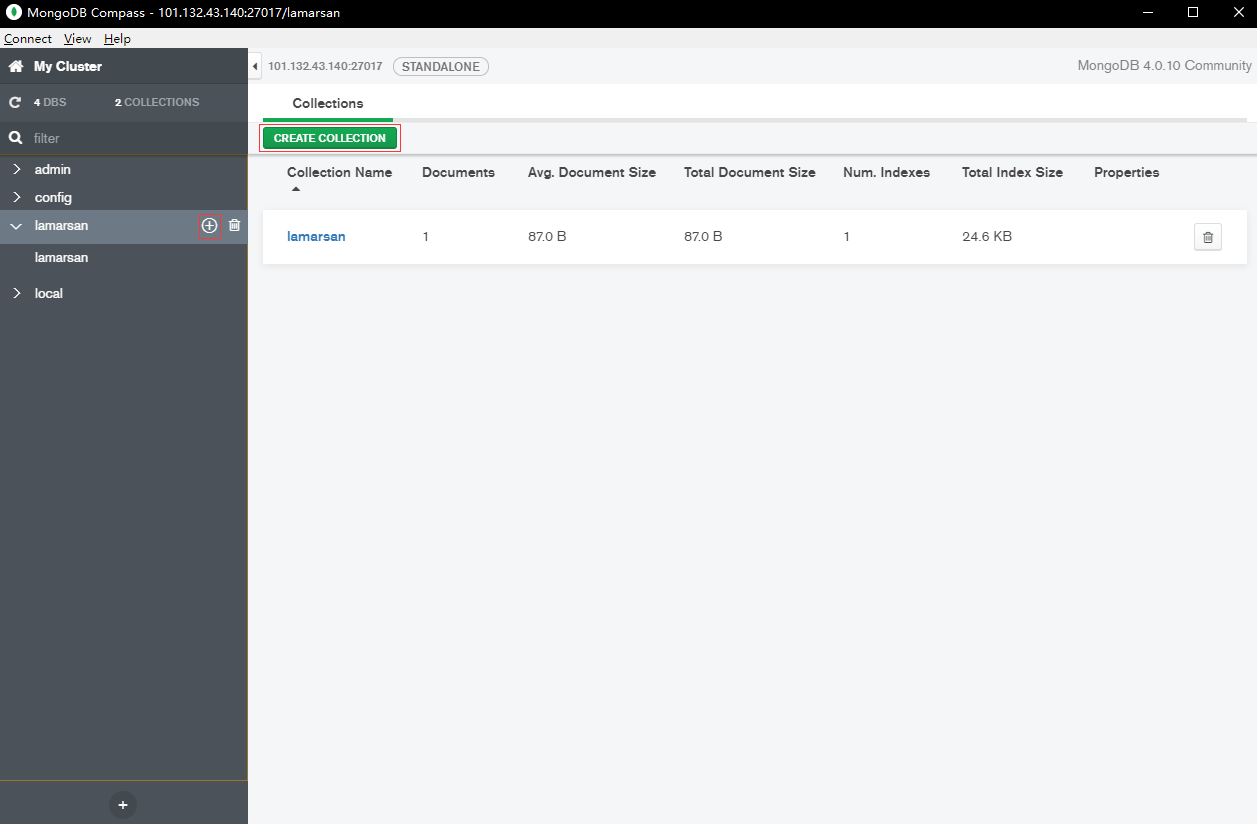


图12 新建集合图

**4）删除集合**

点击集合名后的垃圾图标可进行删除，删除时输入集合名字，即lamar。

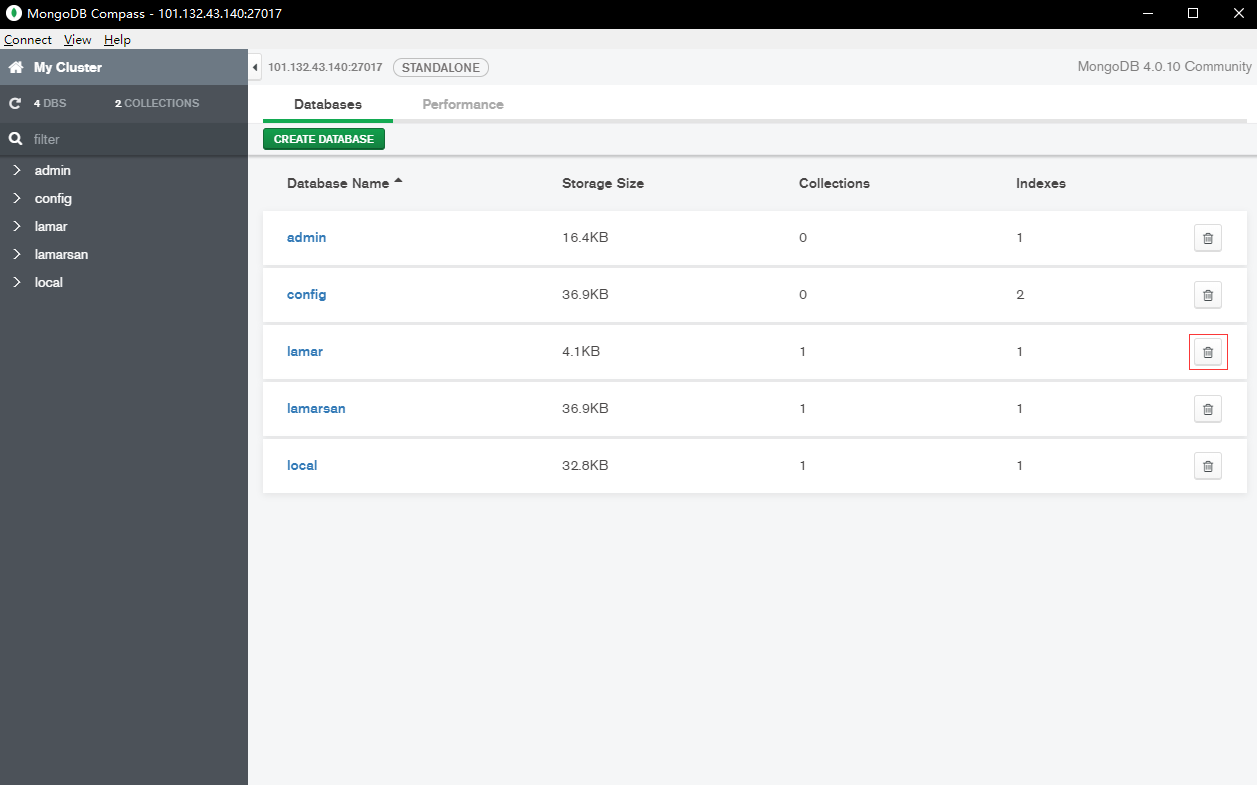


图13 删除集合图

**5）新增文档**

进入集合页面，点击INSERT DOCUMENT，像JSON一样输入键值对，使用回车换行，点击INSERT即可插入。

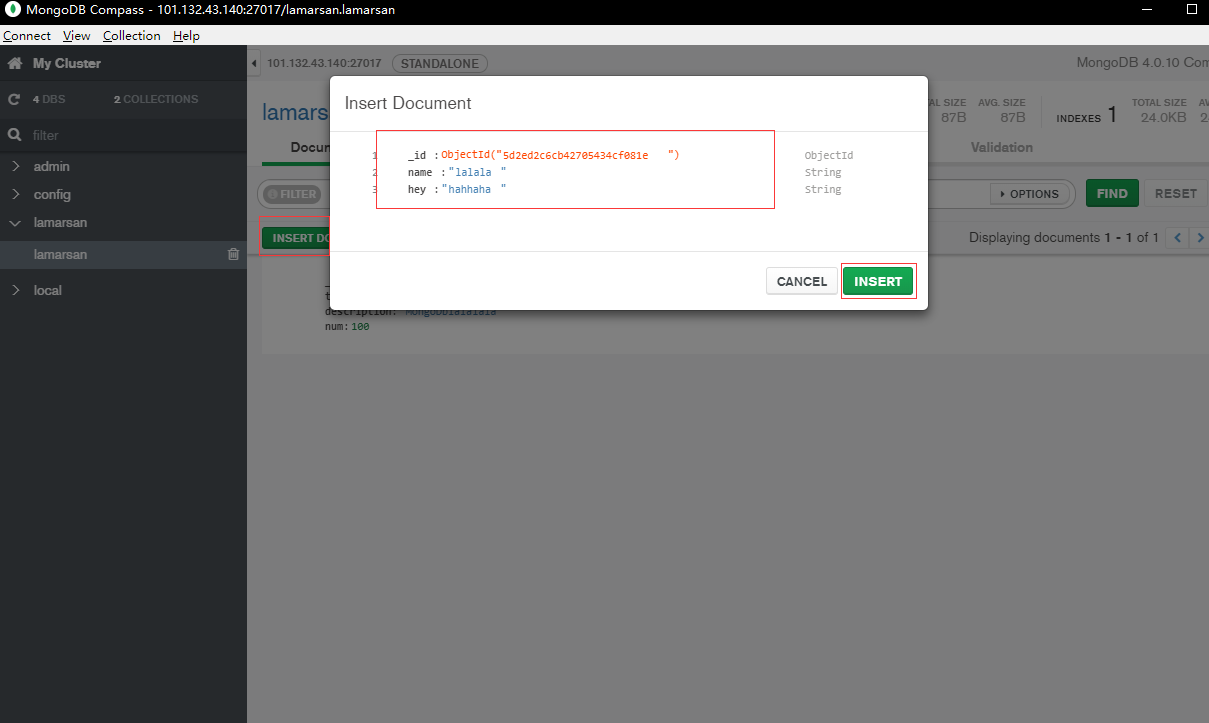


图14 新增文档图

**6）删除文档**

进入集合页面，点击文档右边的垃圾桶标志，即可删除。

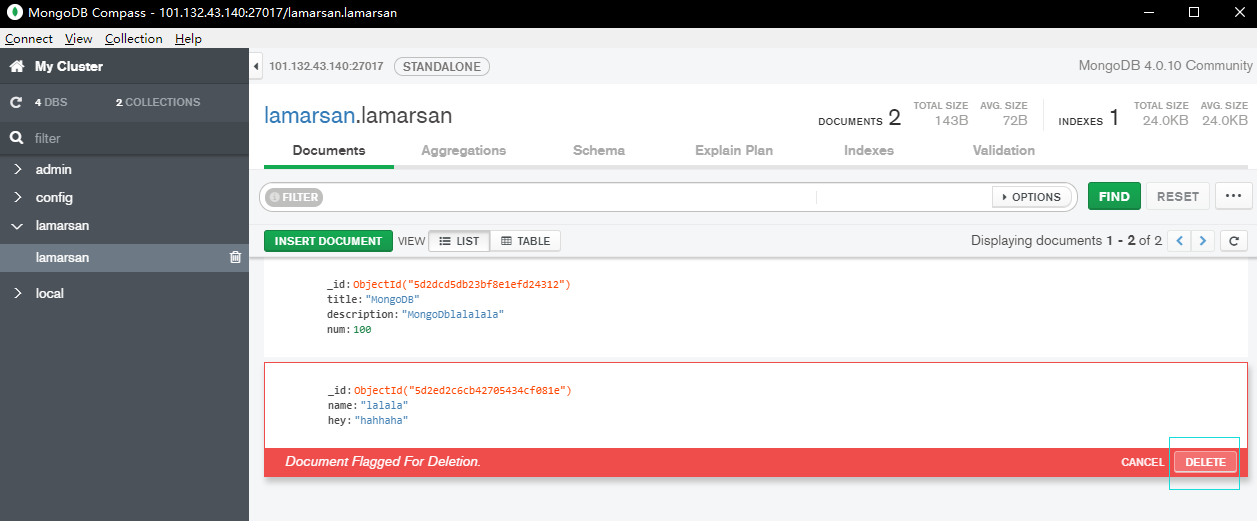


图15 删除文档图

**6）修改文档**

进入集合页面，点击文档右边的笔标志，即可进行对文档的修改。

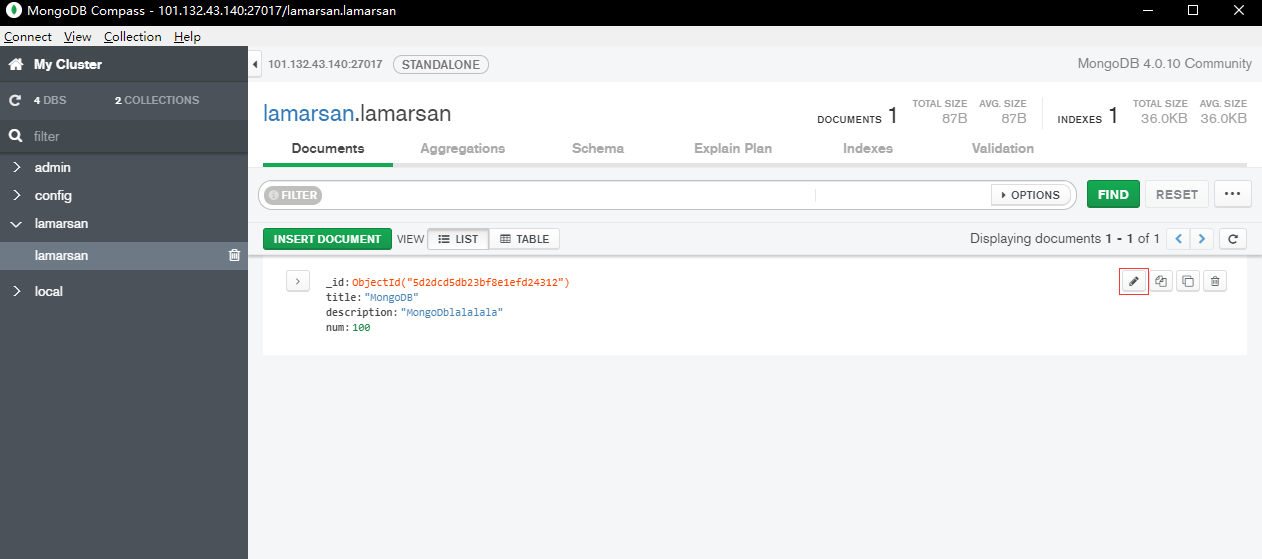


图16 修改文档图

加号为在当前域后加入新域，×为删除当前域。

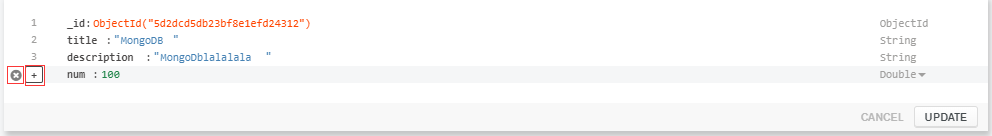


图17 修改文档图

**6）查询文档**

进入集合页面，在FIELTER右边的空框框中输入查询条件：{title:"MongoDB",num:{$lte:100}}

意思是查询title为“MongoDB”，且num小于等于100的文档。点击FIN

-D即可查询。

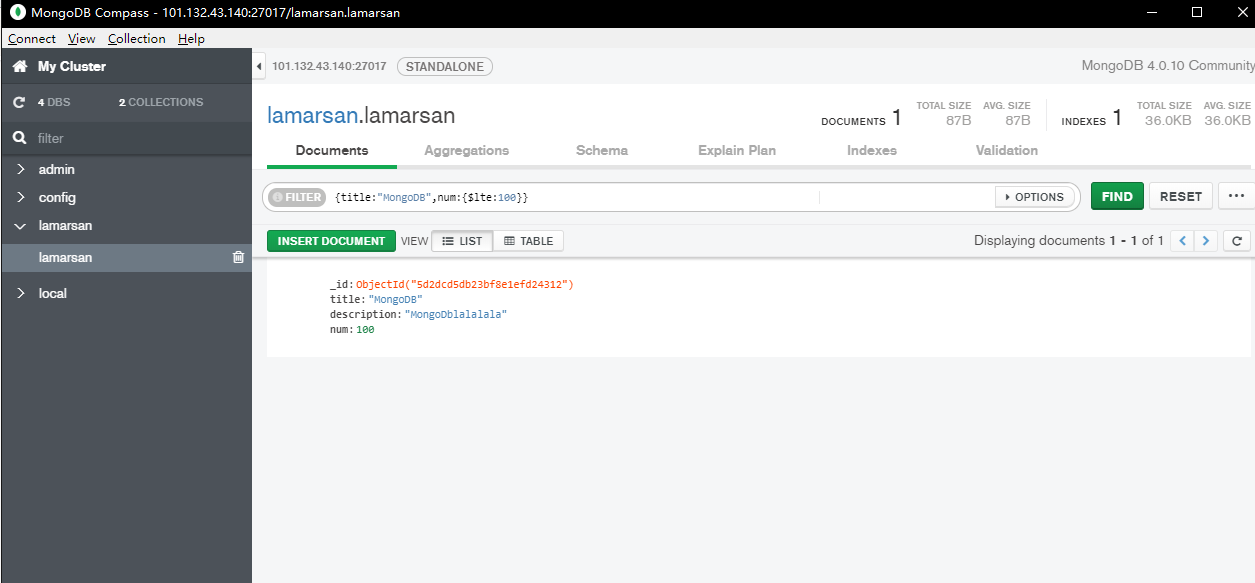


图16 修改文档图

除了上述基本功能外，输入查询语句后，点击右边的…可以获得更多功能，如图所示。

1为查询语句转换图，可以将查询语句转换为java，python等语言，包括需要导入的包，并提供代码复制功能。

2为查询的历史记录。

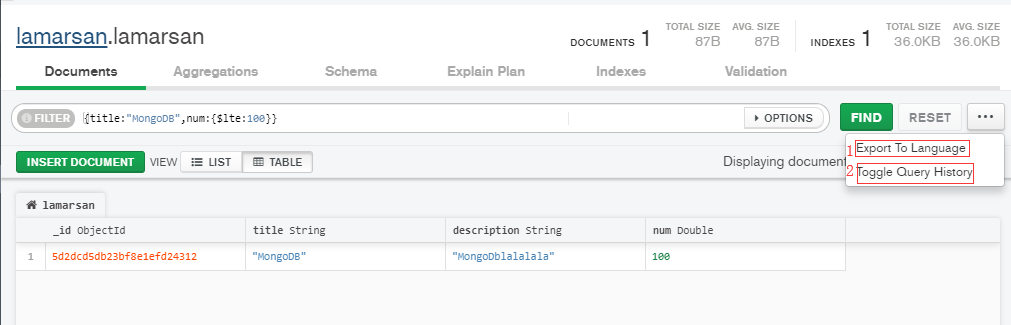


图17 功能扩展图

1. **Java操作mongodb**

1）添加依赖：

<dependency>

<artifactId>baseservice-common-mongodb</artifactId>  
 <groupId>com.ccw.baseservices</groupId>  
 <version>1.0.2-RELEASE</version>  
</dependency>

<dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-mongodb</artifactId>  
 <version>2.0.1.RELEASE</version>  
</dependency>

2）添加配置：

spring.data.mongodb.host=101.132.43.140

spring.data.mongodb.port=27017  
spring.data.mongodb.username=root  
spring.data.mongodb.password=root  
spring.data.mongodb.database=lamarsan

3）编写配置类：

package com.lamarsan.provider.config;

import com.mongodb.MongoClient;  
import com.mongodb.MongoCredential;  
import com.mongodb.ServerAddress;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.PropertySource;  
import org.springframework.data.mongodb.MongoDbFactory;  
import org.springframework.data.mongodb.core.MongoTemplate;  
import org.springframework.data.mongodb.core.SimpleMongoDbFactory;  
import org.springframework.data.mongodb.repository.config.EnableMongoRepositories;  
  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* className: MongDBConf  
 \* description: TODO  
 \*  
 \** ***@author*** *hasee  
 \** ***@version*** *1.0  
 \** ***@date*** *2019/7/17 17:37  
 \*/*@Configuration  
@PropertySource(value = "classpath:application.properties")  
@EnableMongoRepositories(basePackages = "com.lamarsan.provider.mongo.dao")  
public class MongDBConf {  
 @Value("${spring.data.mongodb.username}")  
 private String username;  
 @Value("${spring.data.mongodb.password}")  
 private String password;  
 @Value("${spring.data.mongodb.database}")  
 private String database;  
 @Value("${spring.data.mongodb.host}")  
 private String host;  
 @Value("${spring.data.mongodb.port}")  
 private String port;  
  
 @Bean  
 public MongoClient mongoClient(){  
 String username = this.username;  
 String password = this.password;  
 String database = this.database;  
 String host = this.host;  
 Integer port = Integer.*valueOf*(this.port);  
 ServerAddress serverAddress = new ServerAddress(host,port);  
 MongoCredential mongoCredential = MongoCredential.*createCredential*(username, database, password.toCharArray());  
 List<MongoCredential> mongoCredentialList = new ArrayList<MongoCredential>();  
 mongoCredentialList.add(mongoCredential);  
  
 return new MongoClient(serverAddress,mongoCredentialList);  
 }  
  
 @Bean  
 public MongoDbFactory mongoDbFactory(){  
 String database = this.database;  
 return new SimpleMongoDbFactory(mongoClient(),database);  
 }  
  
 @Bean  
 public MongoTemplate mongoTemplate(){  
  
 MongoTemplate mongoTemplate = new MongoTemplate(mongoDbFactory());  
 return mongoTemplate;  
 }  
}

4）编写实体类，字段加上@Field注释，文档使用@Document注释：

package com.lamarsan.provider.mongo;

import lombok.Data;  
import org.springframework.data.mongodb.core.mapping.Document;  
import org.springframework.data.mongodb.core.mapping.Field;  
  
import java.io.Serializable;  
  
*/\*\*  
 \* className: SongRecord  
 \* description: TODO  
 \*  
 \** ***@author*** *hasee  
 \** ***@version*** *1.0  
 \** ***@date*** *2019/7/17 16:59  
 \*/*@Data  
@Document(collection = "student\_b\_song\_record")  
public class SongRecord implements Serializable {  
 //用户ID  
 @Field("user\_id")  
 private String userId;  
 //歌曲ID  
 @Field("song\_id")  
 private String songId;  
 //音频key  
 @Field("audio\_key")  
 private String audioKey;  
}

5）编写接口类：

public interface SongService {

List<SongRecord> findAllSong();  
}

6）实现接口：

@Service

public class SongServiceImpl implements SongService {  
 @Autowired  
 SongRepository songRepository;  
  
 public List<SongRecord> findAllSong() {  
 return songRepository.findAll();  
 }  
}

7）调用接口：

@RestController

@CrossOrigin  
@RequestMapping("/song")  
public class SongController {  
 @Reference(lazy = true)  
 SongService songService;  
  
 @PostMapping("/allSong")  
 @ResponseBody  
 public List<SongRecord> getAllPerson(){  
 System.*out*.println(songService);  
 return songService.findAllSong();  
 }  
}

8）返回数据：

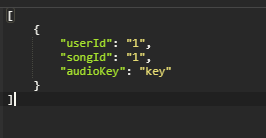


图18 结果图

9）使用图形化工具查看是否正确：

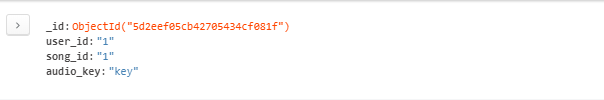


图19 集合结果图