

SpringBoot 入门教程 (十八) | 整合 MongoDB

作者: [JavaFish](#)

原文链接: <https://ld246.com/article/1621930822628>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

01 前言

如题，今天介绍下 SpringBoot 是如何整合 MongoDB 的。

02 MongoDB 简介

MongoDB 是由 C++ 编写的非关系型数据库，是一个基于分布式文件存储的开源数据库系统，它将数据存储为一个文档，数据结构由键值 (key=>value) 对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组，非常灵活。存储结构如下：

```
{
  "studentId": "201311611405",
  "age":24,
  "gender":"男",
  "name":"一个优秀的废人"
}
```

03 准备工作

- SpringBoot 2.1.3 RELEASE
- MongoDB 2.1.3 RELEASE
- MongoDB 4.0
- IDEA
- JDK8
- 创建一个名为 test 的数据库，不会建的。参考菜鸟教程：

<http://www.runoob.com/mongodb/mongodb-tutorial.html>

04 配置数据源

```
spring:
  data:
    mongodb:
      uri: mongodb://localhost:27017/test
```

以上是无密码写法，如果 MongoDB 设置了密码应这样设置：

```
spring:
  data:
    mongodb:
      uri: mongodb://name:password@localhost:27017/test
```

05 pom 依赖配置

```
<!-- mongodb 依赖 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

```
<!-- web 依赖 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- lombok 依赖 -->
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<!-- test 依赖 (没用到) -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

06 实体类

```
@Data
public class Student {

  @Id
  private String id;

  @NotNull
  private String studentId;

  private Integer age;

  private String name;

  private String gender;
}
```

07 dao 层

和 JPA 一样，SpringBoot 同样为开发者准备了一套 Repository，只需要继承 MongoRepository 入实体类型以及主键类型即可。

```
@Repository
public interface StudentRepository extends MongoRepository<Student, String> {
}
```

08 service 层

```
public interface StudentService {

  Student addStudent(Student student);
}
```

```

void deleteStudent(String id);

Student updateStudent(Student student);

Student findStudentById(String id);

List<Student> findAllStudent();
}

```

实现类:

```

@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;

    /**
     * 添加学生信息
     * @param student
     * @return
     */
    @Override
    @Transactional(rollbackFor = Exception.class)
    public Student addStudent(Student student) {
        return studentRepository.save(student);
    }

    /**
     * 根据 id 删除学生信息
     * @param id
     */
    @Override
    public void deleteStudent(String id) {
        studentRepository.deleteById(id);
    }

    /**
     * 更新学生信息
     * @param student
     * @return
     */
    @Override
    @Transactional(rollbackFor = Exception.class)
    public Student updateStudent(Student student) {
        Student oldStudent = this.findStudentById(student.getId());
        if (oldStudent != null){
            oldStudent.setStudentId(student.getStudentId());
            oldStudent.setAge(student.getAge());
            oldStudent.setName(student.getName());
            oldStudent.setGender(student.getGender());
            return studentRepository.save(oldStudent);
        } else {

```

```

        return null;
    }
}

/**
 * 根据 id 查询学生信息
 * @param id
 * @return
 */
@Override
public Student findStudentById(String id) {
    return studentRepository.findById(id).get();
}

/**
 * 查询学生信息列表
 * @return
 */
@Override
public List<Student> findAllStudent() {
    return studentRepository.findAll();
}
}

```

09 controller 层

```

@RestController
@RequestMapping("/student")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @PostMapping("/add")
    public Student addStudent(@RequestBody Student student){
        return studentService.addStudent(student);
    }

    @PutMapping("/update")
    public Student updateStudent(@RequestBody Student student){
        return studentService.updateStudent(student);
    }

    @GetMapping("/{id}")
    public Student findStudentById(@PathVariable("id") String id){
        return studentService.findStudentById(id);
    }

    @DeleteMapping("/{id}")
    public void deleteStudentById(@PathVariable("id") String id){
        studentService.deleteStudent(id);
    }

    @GetMapping("/list")

```

```
public List<Student> findAllStudent(){
    return studentService.findAllStudent();
}
}
```

10 测试结果

The screenshot shows a Postman interface for a POST request to `http://localhost:8080/student/add`. The request body is a JSON object:

```
{
  "studentId": "201311611405",
  "age": 24,
  "gender": "男",
  "name": "一个优秀的废人"
}
```

The response body is a JSON object:

```
{
  "id": "5c839172d84c7a2f3c9df54b",
  "studentId": "201311611405",
  "age": 24,
  "name": "一个优秀的废人",
  "gender": "男"
}
```

Status: 200 OK, Time: 23 ms, Size: 245 B

Postman 测试已经全部通过，这里仅展示了保存操作。

The screenshot shows the MongoDB Shell interface. The command executed is `db.getCollection('student').find({})`. The result is a document:

Key	Value	Type
(1) ObjectId("5c839172d84c7a2f3c9df54b")	{ 6 fields }	Object
_id	ObjectId("5c839172d84c7a2f3c9df54b")	ObjectId
studentId	201311611405	String
age	24	Int32
name	一个优秀的废人	String
gender	男	String
_class	com.nasus.mongodb.domain.Student	String

这里推荐一个数据库可视化工具 Robo 3T。下载地址：<https://robomongo.org/download>

11 完整代码

https://github.com/turoDog/Demo/tree/master/springboot_mongodb_demo

如果觉得对你有帮助，请给个 Star 再走呗，非常感谢。

12 大厂面试题

如果本文对你哪怕有一丁点帮助，请帮忙点好看。你的好看是我坚持写作的动力。

初次见面，也不知道送你们啥。干脆就送**几百本电子书**和**2021最新面试资料**吧。微信搜索**JavaFish**复**电子书**送你 1000+ 本编程电子书；回复**面试题**获取 50 套大厂面试题；回复**1024**送你一套完整的 jav 视频教程。

面试题都是有答案的，如下所示：有需要的就来拿吧，**绝对免费，无套路获取**。

The image shows a file explorer window with a sidebar on the left and a main pane on the right. The sidebar contains a list of folders: 大数据, C 语言, C++, Java, Git, Python, Go 语言, Linux, 经典必读, 面试相关, 前端, 人工智能, 设计模式. The main pane displays a list of PDF files under the heading '面试题'. The files are listed with their names and modification dates (all 2021). The list includes various topics such as ActiveMQ, Netty, design patterns, Java fundamentals, Tomcat, multithreading, Kafka, Nginx, Spring Boot, JVM, Spring MVC, English, RabbitMQ, Java multithreading/locks/memory, Spring, MyBatis, Dubbo, Redis, and HR interview questions.

名称	修改日期
7道消息队列ActiveMQ面试题! .pdf	2021,
10道Java高级必备的Netty面试题! .pdf	2021,
10道Java面试必备的设计模式面试题!	2021,
10个Java经典的List面试题! .pdf	2021,
10个Java经典的Main方法面试题! .pdf	2021,
10个Java经典的String面试题! .pdf	2021,
15道经典的Tomcat面试题! .pdf	2021,
15道面试常问的Java多线程面试题! .pdf	2021,
17道消息队列Kafka面试题! .pdf	2021,
18道非常牛逼的Nginx面试题! .pdf	2021,
20道顶尖的Spring Boot面试题! .pdf	2021,
20道面试官常问的JVM面试题! .pdf	2021,
22道面试常问的SpringMVC面试题! .pdf	2021,
24道经典的英语面试题! .pdf	2021,
24道消息队列RabbitMQ面试题! .pdf	2021,
27道顶尖的Java多线程、锁、内存模型...	2021,
29道常见的Spring面试题! .pdf	2021,
30个Java经典的集合面试题! .pdf	2021,
36道面试常问的MyBatis面试题! .pdf	2021,
40道常问的Java多线程面试题! .pdf	2021,
55道BAT精选的Mysql面试题! .pdf	2021,
60道必备的Java核心技术面试题! .pdf	2021,
70道阿里巴巴高级Java面试题! .pdf	2021,
Java 面试题经典 77 问! .pdf	2021,
分布式缓存 Redis + Memcached 经典...	2021,
搞定 HR 面试的 40 个必备问题! .pdf	2021,
精选7道Elastic Search面试题! .pdf	2021,
精选8道Dubbo面试题! .pdf	2021,
精选17道海量数量处理面试题! .pdf	2021,
史上最全40道Dubbo面试题! .pdf	2021,
史上最全50道Redis面试题! .pdf	2021,



JavaFish

微信扫描二维码，关注我的公众号