《Spring Boot 基础教程》

第 18 节

使用 NoSQL 数据库-mongodb

安装：mongodb 下载链接：https://www.mongodb.com/download-center#community

下载版本：mongodb-win32-x86_64-2008plus-ssl-3.2.9-signed.msi

安装出现 2502、2503 错误解决办法：

http://jingyan.baidu.com/article/a501d80cec07daec630f5e18.html

启动命令：mongod.exe --dbpath d:\roncoo_mongodb

指定路径：--dbpath

注：要先创建文件夹

一、添加依赖

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

二、配置文件：

```
# MONGODB (MongoProperties)
spring.data.mongodb.uri=mongodb://localhost/test
spring.data.mongodb.port=27017
#spring.data.mongodb.authentication-database=
#spring.data.mongodb.database=test
#spring.data.mongodb.field-naming-strategy=
#spring.data.mongodb.grid-fs-database=
#spring.data.mongodb.host=localhost
#spring.data.mongodb.password=
#spring.data.mongodb.repositories.enabled=true
#spring.data.mongodb.username=
```

三、代码

```
/**
 * @author wujing
 */
@Component
public class RoncooMongodbComponent {

 @Autowired
 private MongoTemplate mongoTemplate;

 public void insert(RoncooUser roncooUser) {
     mongoTemplate.insert(roncooUser);
```

```
    }

    public void deleteById(int id) {
        Criteria criteria = Criteria.where("id").in(id);
        Query query = new Query(criteria);
        mongoTemplate.remove(query, RoncooUser.class);
    }

    public void updateById(RoncooUser roncooUser) {
        Criteria criteria = Criteria.where("id").in(roncooUser.getId());
        Query query = new Query(criteria);
        Update update = new Update();
        update.set("name", roncooUser.getName());
        update.set("createTime", roncooUser.getCreateTime());
        mongoTemplate.updateMulti(query, update, RoncooUser.class);
    }

    public RoncooUser selectById(int id) {
        Criteria criteria = Criteria.where("id").in(id);
        Query query = new Query(criteria);
        return mongoTemplate.findOne(query, RoncooUser.class);
    }
}
```

设置日志打印：

```
<logger name="org.springframework.data.mongodb.core.MongoTemplate" level="debug"/>
```

```
@Autowired
    private RoncooMongodbComponent roncooMongodbComponent;

    @Test
    public void set() {
        RoncooUser roncooUser = new RoncooUser();
        roncooUser.setId(1);
        roncooUser.setName("无境1");
        roncooUser.setCreateTime(new Date());
        roncooMongodbComponent.insert(roncooUser);
    }

    @Test
    public void select() {

    System.out.println(roncooMongodbComponent.selectById(1));
    }
```

```
    @Test
    public void update() {
        RoncooUser roncooUser = new RoncooUser();
        roncooUser.setId(1);
        roncooUser.setName("测试修改");
        roncooUser.setCreateTime(new Date());
        roncooMongodbComponent.updateById(roncooUser);

    System.out.println(roncooMongodbComponent.selectById(1));
    }

    @Test
    public void delete() {
        roncooMongodbComponent.deleteById(1);
    }
```

四、使用：MongoRepository

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.mongodb.repository.MongoRepository;

import com.roncoo.example.bean.RoncooUserLog;

public interface RoncooUserLogMongoDao extends MongoRepository<RoncooUserLog, Integer>{

    RoncooUserLog findByUserName(String string);

    RoncooUserLog findByUserNameAndUserIp(String string, String ip);

    Page<RoncooUserLog> findByUserName(String string, Pageable pageable);
}
```

测试

```
    @Autowired
    private RoncooUserLogMongoDao roncooUserLogMongoDao;

    @Test
    public void insert() {
        RoncooUserLog entity = new RoncooUserLog();
        entity.setId(1);
        entity.setUserName("无境");
        entity.setUserIp("192.168.0.1");
        entity.setCreateTime(new Date());
```

```
        roncooUserLogMongoDao.save(entity);
    }

    @Test
    public void delete() {
        roncooUserLogMongoDao.delete(1);
    }

    @Test
    public void update() {
        RoncooUserLog entity = new RoncooUserLog();
        entity.setId(1);
        entity.setUserName("无境2");
        entity.setUserIp("192.168.0.1");
        entity.setCreateTime(new Date());
        roncooUserLogMongoDao.save(entity);
    }

    @Test
    public void select() {
        RoncooUserLog result = roncooUserLogMongoDao.findOne(1);
        System.out.println(result);
    }

    @Test
    public void select2() {
        RoncooUserLog result = roncooUserLogMongoDao.findByUserName("
无境2");
        System.out.println(result);
    }

    // 分页
    @Test
    public void queryForPage() {
        Pageable pageable = new PageRequest(0, 20, new Sort(new
Order(Direction.DESC, "id")));
        // Page<RoncooUserLog> result =
roncooUserLogDao.findByUserName("无境2", pageable);
        Page<RoncooUserLog> result =
roncooUserLogMongoDao.findAll(pageable);
        System.out.println(result.getContent());
    }
```

五、 使用嵌入式的 mongo

```
<dependency>
        <groupId>de.flapdoodle.embed</groupId>
        <artifactId>de.flapdoodle.embed.mongo</artifactId>
</dependency>
```

注意：

1. 加入嵌入式的 mongo 之后，首次启动会进行下载，时间会比较久，请耐心等待

```
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [54] -| Download 3.2.2:Windows:B64 : starting...
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [59] -| Download 3.2.2:Windows:B64 : DownloadSize: 148367823
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 0 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 10 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 20 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 30 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 40 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 50 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 60 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 70 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 80 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 90 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [42] -| Download 3.2.2:Windows:B64 : 100 %
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [59] -| Download 3.2.2:Windows:B64 : downloaded with 285kb/s
org.springframework.boot.autoconfigure.mongo.embedded.EmbeddedMongo [49] -| Download 3.2.2:Windows:B64 : finished
```

2. 下载完成，启动之后，默认情况下数据会在内存里面，重启会丢失

更多课程信息，请关注 龙果学院 官方网站 http://www.roncoo.com/

或关注 龙果 微信公众号 RonCoo_com