《Spring Boot 基础教程》

第 13 节

使用 SQL 关系型数据库-JdbcTemplate

一、配置数据源：

嵌入式数据库的支持：Spring Boot 可以自动配置 H2, HSQL and Derby 数据库，不需要提供任何的链接 URLs，只需要加入相应的 jar 包，Spring boot 可以自动发现装配

```xml
<!-- 数据库 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

mysql

```
spring.datasource.url=jdbc:mysql://localhost/spring_boot_demo?useUnicode=true&characterEncoding=utf-8
spring.datasource.username=root
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

注：

1.可以不指定 driver-class-name，spring boot 会自动识别 url。

2.数据连接池默认使用 tomcat-jdbc

连接池的配置： spring.datasource.tomcat.*

二、JdbcTemplate 模板

```java
// 自动注册
@Autowired
private JdbcTemplate jdbcTemplate;
```

三、脚本

```sql
CREATE TABLE `roncoo_user` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255) DEFAULT NULL,
    `create_time` datetime DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='用户表';
```

四、实体类

```
/**
 * 实体类
 *
 * @author wujing
 */
public class RoncooUser {
 private int id;
 private String name;
 private Date createTime;

 public int getId() {
     return id;
 }

 public void setId(int id) {
     this.id = id;
 }

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

 public Date getCreateTime() {
     return createTime;
 }

 public void setCreateTime(Date createTime) {
     this.createTime = createTime;
 }

 @Override
 public String toString() {
     return "RoncooUser [id=" + id + ", name=" + name + ", createTime=" + createTime
+ "]";
 }

}
```

五、接口

```
    int insert(RoncooUser roncooUser);

    int deleteById(int id);

    int updateById(RoncooUser roncooUser);

    RoncooUser selectById(int id);
```

六、实现类代码

```
    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public int insert(RoncooUser roncooUser) {
        String sql = "insert into roncoo_user (name, create_time) values (?, ?)";
        return          jdbcTemplate.update(sql,          roncooUser.getName(),
roncooUser.getCreateTime());
    }

    @Override
    public int deleteById(int id) {
        String sql = "delete from roncoo_user where id=?";
        return jdbcTemplate.update(sql, id);
    }

    @Override
    public int updateById(RoncooUser roncooUser) {
        String sql = "update roncoo_user set name=?, create_time=? where id=?";
        return          jdbcTemplate.update(sql,          roncooUser.getName(),
roncooUser.getCreateTime(), roncooUser.getId());
    }

    @Override
    public RoncooUser selectById(int id) {
        String sql = "select * from roncoo_user where id=?";
        return jdbcTemplate.queryForObject(sql, new RowMapper<RoncooUser>() {
            @Override
            public RoncooUser mapRow(ResultSet rs, int rowNum) throws SQLException {
                RoncooUser roncooUser = new RoncooUser();
                roncooUser.setId(rs.getInt("id"));
                roncooUser.setName(rs.getString("name"));
                roncooUser.setCreateTime(rs.getDate("create_time"));
                return roncooUser;
            }
            }, id);
    }
```

七、测试类代码

```
        @Autowired
        private RoncooUserDao roncooUserDao;

        @Test
        public void insert() {
                RoncooUser roncooUser = new RoncooUser();
                roncooUser.setName("测试");
                roncooUser.setCreateTime(new Date());
                int result = roncooUserDao.insert(roncooUser);
                System.out.println(result);
        }

        @Test
        public void delete() {
                int result = roncooUserDao.deleteById(1);
                System.out.println(result);
        }

        @Test
        public void update() {
                RoncooUser roncooUser = new RoncooUser();
                roncooUser.setId(2);
                roncooUser.setName("测试 2");
                roncooUser.setCreateTime(new Date());
                int result = roncooUserDao.updateById(roncooUser);
                System.out.println(result);
        }

        @Test
        public void select() {
                RoncooUser result = roncooUserDao.selectById(2);
                System.out.println(result);
    }
```

打印 sql 语句，添加如下

```
<logger name="org.springframework.jdbc.core.JdbcTemplate" level="debug"/>
```

八、封装 spring jdbc，带分页

```
/**
     * 获取当前事务最后一次更新的主键值
     */
    public Long getLastId() {
        return jdbcTemplate.queryForObject("select last_insert_id() as id", Long.class);
    }

    /**
     * 获取对象信息
     */
    public <T> T queryForObject(String sql, Class<T> clazz, Object... args) {
        Assert.hasText(sql, "sql 语句不能为空");
        return jdbcTemplate.queryForObject(sql, new BeanPropertyRowMapper<T>(clazz), args);
    }

    /**
     * 获取对象集合信息
     */
    public <T> List<T> queryForObjectList(String sql, Class<T> clazz, Object... args) {
        Assert.hasText(sql, "sql 语句不能为空");
        return jdbcTemplate.query(sql, args, new BeanPropertyRowMapper<T>(clazz));
    }

    /**
     * 分页，jdbcTemplate 不支持 like 自定义，只能拼装
     */
    public Page<Map<String, Object>> queryForPage(String sql, int pageCurrent, int
pageSize, Object... args) {
        Assert.hasText(sql, "sql 语句不能为空");
        Assert.isTrue(pageCurrent >= 1, "pageNo 必须大于等于 1");
        String sqlCount = Sql.countSql(sql);
        int count = jdbcTemplate.queryForObject(sqlCount, Integer.class, args);
        pageCurrent = Sql.checkPageCurrent(count, pageSize, pageCurrent);
        pageSize = Sql.checkPageSize(pageSize);
        int totalPage = Sql.countTotalPage(count, pageSize);
        String sqlList = sql + Sql.limitSql(count, pageCurrent, pageSize);
        List<Map<String, Object>> list = jdbcTemplate.queryForList(sqlList, args);
        return new Page<Map<String, Object>>(count, totalPage, pageCurrent, pageSize,
list);
    }

    /**
     * 分页，jdbcTemplate 不支持 like 是定义，只能拼装
     */
    public <T> Page<T> queryForPage(String sql, int pageCurrent, int pageSize, Class<T>
clazz, Object... args) {
        Assert.hasText(sql, "sql 语句不能为空");
        Assert.isTrue(pageCurrent >= 1, "pageNo 必须大于等于 1");
        Assert.isTrue(clazz != null, "clazz 不能为空");
        String sqlCount = Sql.countSql(sql);
        int count = jdbcTemplate.queryForObject(sqlCount, Integer.class, args);
        pageCurrent = Sql.checkPageCurrent(count, pageSize, pageCurrent);
        pageSize = Sql.checkPageSize(pageSize);
        int totalPage = Sql.countTotalPage(count, pageSize);
        String sqlList = sql + Sql.limitSql(count, pageCurrent, pageSize);
        List<T> list = jdbcTemplate.query(sqlList, new BeanPropertyRowMapper<T>(clazz),
args);
        return new Page<T>(count, totalPage, pageCurrent, pageSize, list);
    }
```

测试

```
@Test
    public void select2() {
        RoncooUser result = roncooUserDao.selectById(7);
        System.out.println(result);
    }

    // 分页测试
    @Test
    public void queryForPage(){
        Page<RoncooUser> result = roncooUserDao.queryForPage(1,
20, "测试");
        System.out.println(result.getList());
    }
```

实现类

```
@Override
    public Page<RoncooUser> queryForPage(int pageCurrent, int
pageSize, String name){
        // 确定参数
        /*String sql = "select * from roncoo_user where name=?";
        return queryForPage(sql.toString(), pageCurrent,
pageSize, RoncooUser.class, name);*/

        // 若name可能为空，则要进行判定，如下
        /*StringBuffer sql = new StringBuffer("select * from
roncoo_user where 1");
```

```
        if(!StringUtils.isNullOrEmpty(name)){
            // Sql.checkSql 的作用是防止sql注入
            sql.append(" and name = 
'").append(Sql.checkSql(name)).append("' ");
        }
        return queryForPage(sql.toString(), pageCurrent, 
pageSize, RoncooUser.class);*/

        // 若要like查询，如下
        StringBuffer sql = new StringBuffer("select * from 
roncoo_user where 1");
        if(!StringUtils.isNullOrEmpty(name)){
            // Sql.checkSql 的作用是防止sql注入
            sql.append(" and name like 
'%").append(Sql.checkSql(name)).append("%' ");
        }
        return queryForPage(sql.toString(), pageCurrent, 
pageSize, RoncooUser.class);
    }   }
```

Mysql-Front 下载地址： http://www.mysqlfront.de/