

Quiz后端-1、spring项目初始化与数据库设置

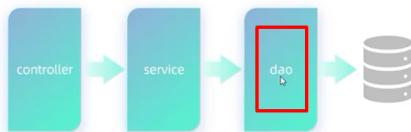
日期：2025.07.19 作者：tfzhang

后端主要提供用户与题目的注册、查询、删除等操作。使用到的开发工具：

- IDEA2024; 关注公众号”青椒工具”，发送”IDEA”，获取windows下的IDEA安装包
- mysql 5.7; 关注公众号”青椒工具”，发送”mysql”，获取windows下的mysql5.7安装包；

后端采用java实现，项目采用spring初始化，并进行数据库设计。数据库采用mysql，涉及到后端技术栈：

- java
- spring
- springmvc
- mybatis：一款优秀的持久层框架，用于简化JDBC的开发；



- springboot

这些工具都可以在spring初始化时安装。

本项目使用mysql数据库，所以我们要先安装mysql数据库，其安装过程及资源可参考如下的视频：

```
1 | https://www.bilibili.com/video/BV1Ra4y1y7Hs/?  
    spm_id_from=333.999.0.0&vd_source=b0e6d0da66db457c6afda440766d8139
```

为了让Maven管理器，下载包更快，可以将当前系统中的maven包源地址设置为阿里云镜像：

```
1 <mirrors>  
2   <mirror>  
3     <id>aliyun</id>  
4     <name>Aliyun Maven Mirror</name>  
5     <url>https://maven.aliyun.com/repository/public</url>  
6     <mirrorOf>central</mirrorOf>  
7   </mirror>  
8 </mirrors>
```

具体地，在Windows: C:\Users\你的用户名\.m2\settings.xml文件中，输入上述内容。

关于Maven的详情，可以参考：[Day04-01_maven-课程介绍哔哩哔哩bilibili](#)

1、spring初始化

直接在IDEA开发工具进行初始化，

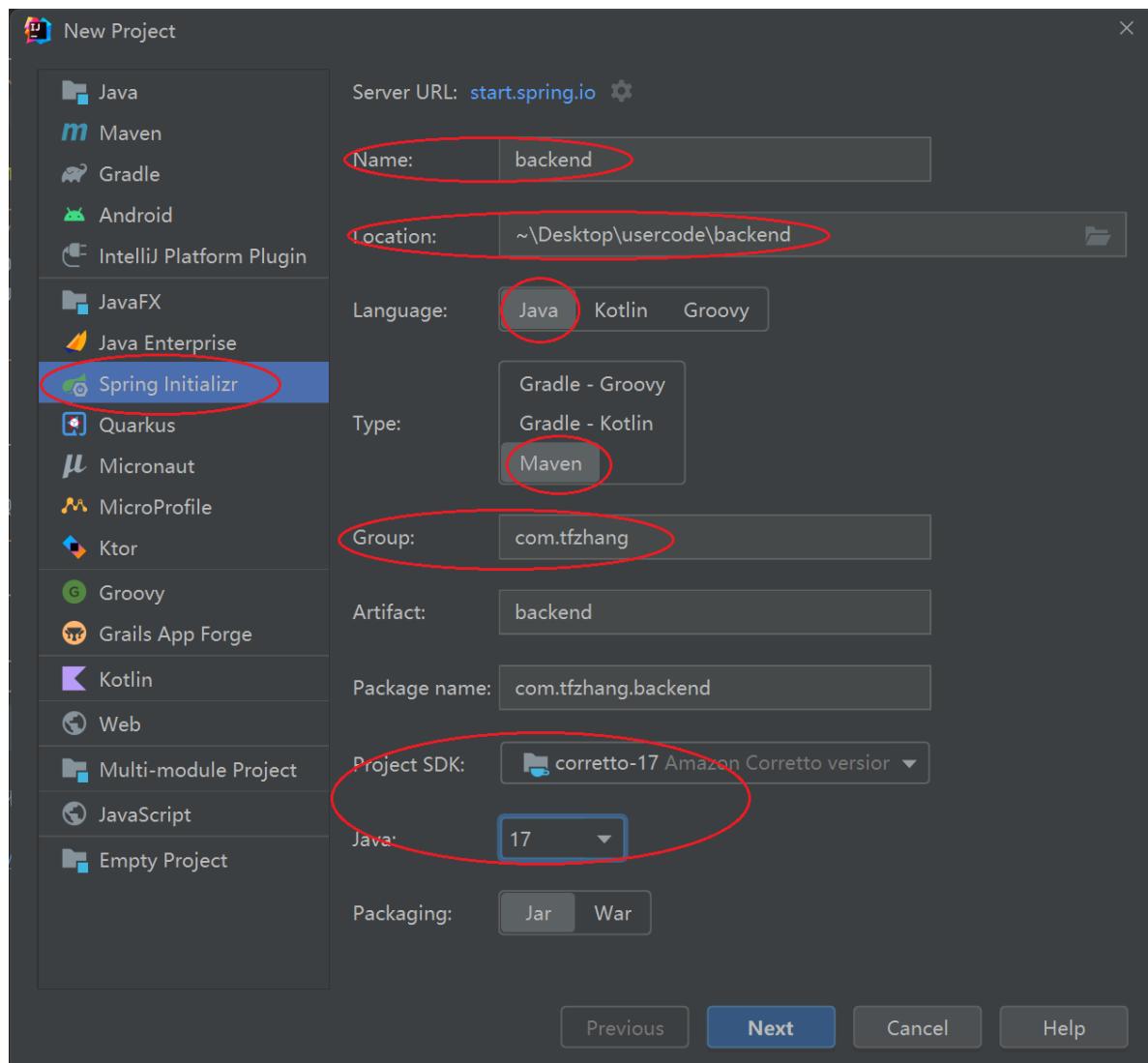


图1 spring项目初始化

此处值得注意的是Project SDK是选择jdk版本，你选择好后，IDEA会自动帮你下载安装，不需要自己额外到系统中安装java jdk。

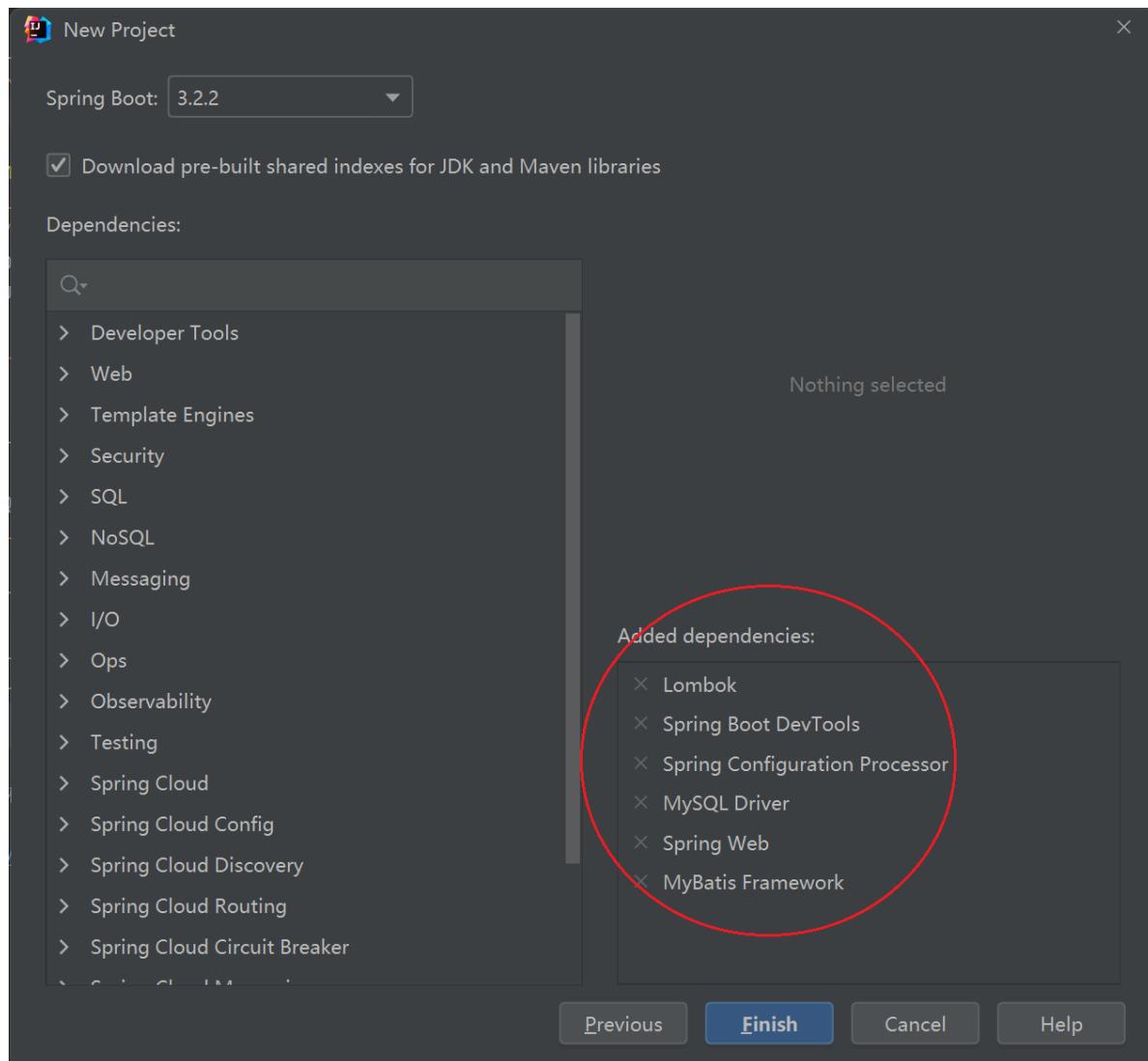


图2 spring boot需要安装的依赖

这些依赖，可以通过左上角的"搜索框"查找，添加。简单介绍下：

- Lombok：自动添加get和set方法；
- Mysql Driver：链接数据库的驱动；
- Spring web：支持网络访问接口；
- MyBatis Framework：操作数据库；

点击Finish后，IDEA会自动去下载需要的依赖包，需要一定的时间，完成下载后如图3所示。

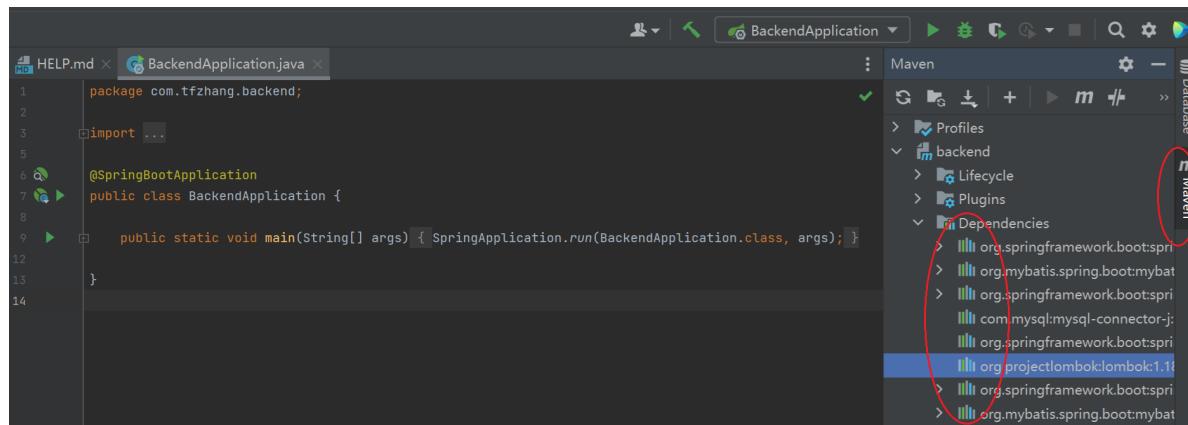


图3 点击右侧的maven展示依赖包

图4是IDEA初始化后的项目目录，简单扼要介绍下：

- src: 主要存放源代码， main是业务代码， test是测试代码；
- pom.xml: 依赖包的配置文件，如果要增加新的依赖包，可以在该文件中配置；

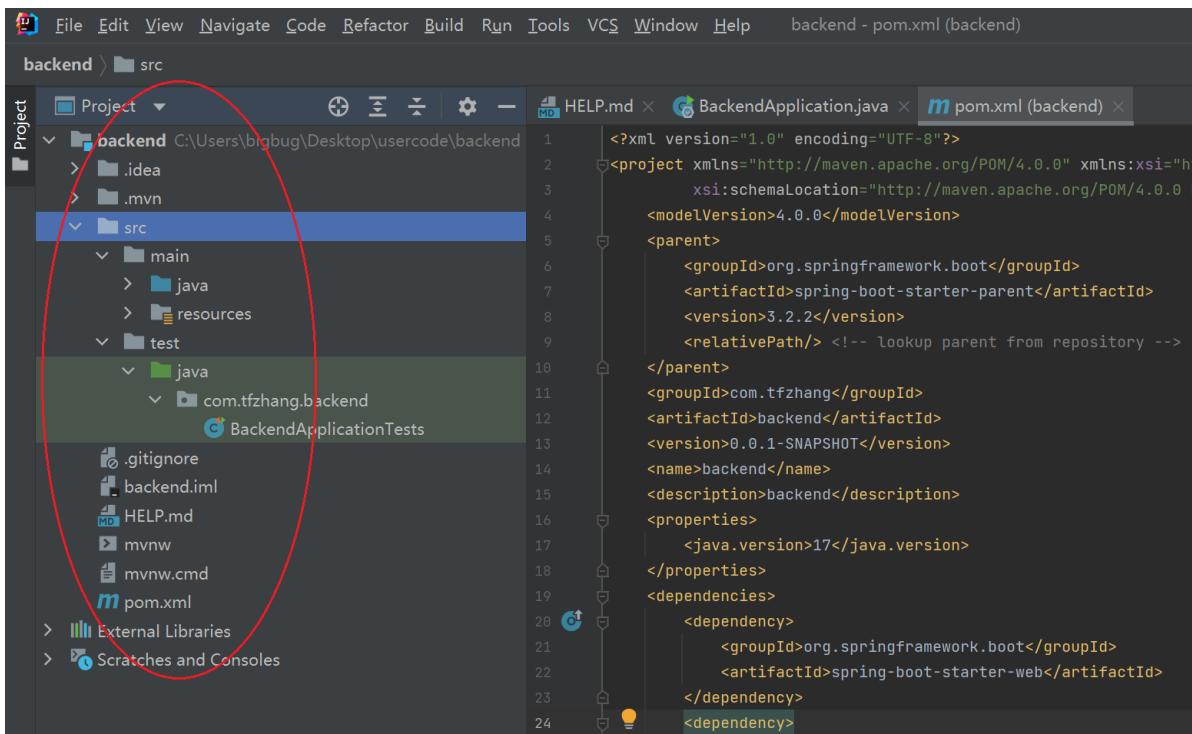


图4 IDEA初始化的项目文件

2、连接数据库及数据库设计

完成数据库安装后，我们使用IDEA连接数据库，并初始化数据库。

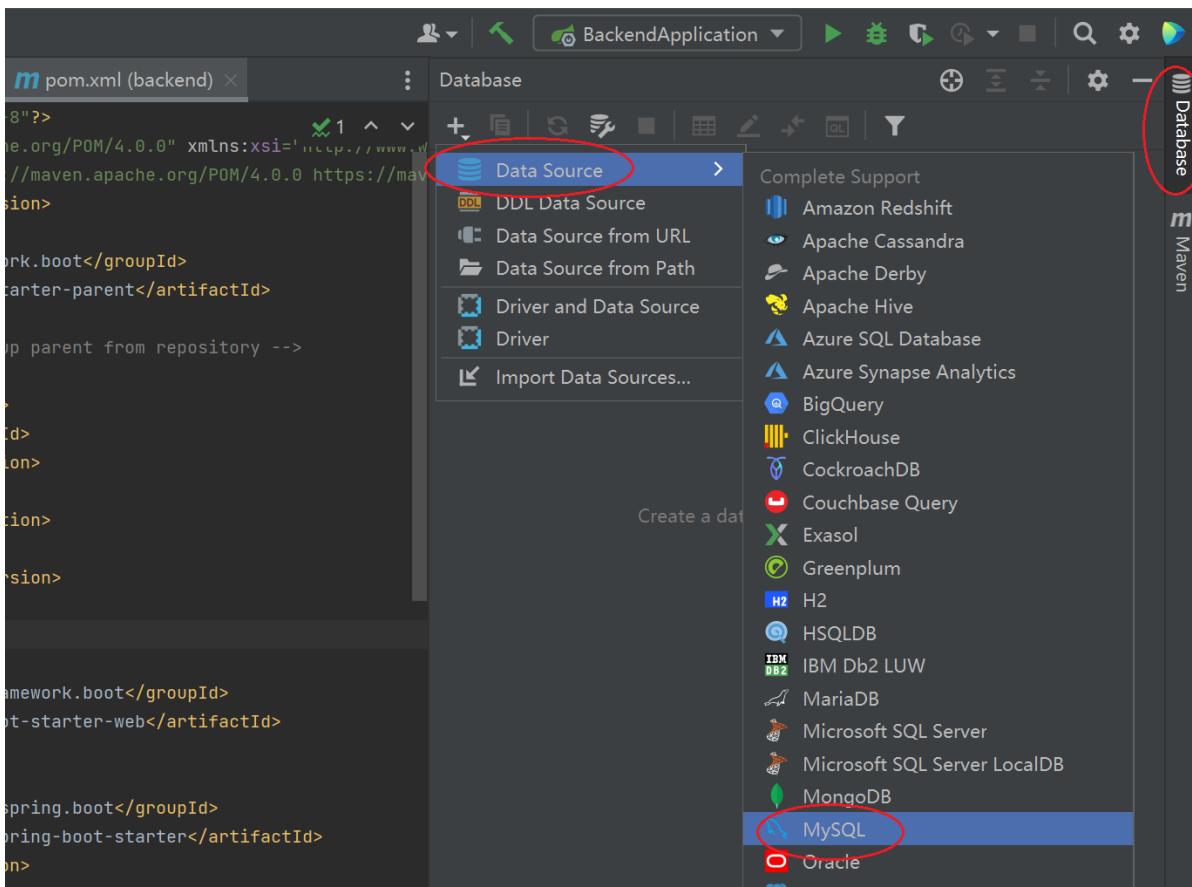


图5 IDEA连接mysql数据库

填写数据库用户和密码，登录数据库，图6所示：

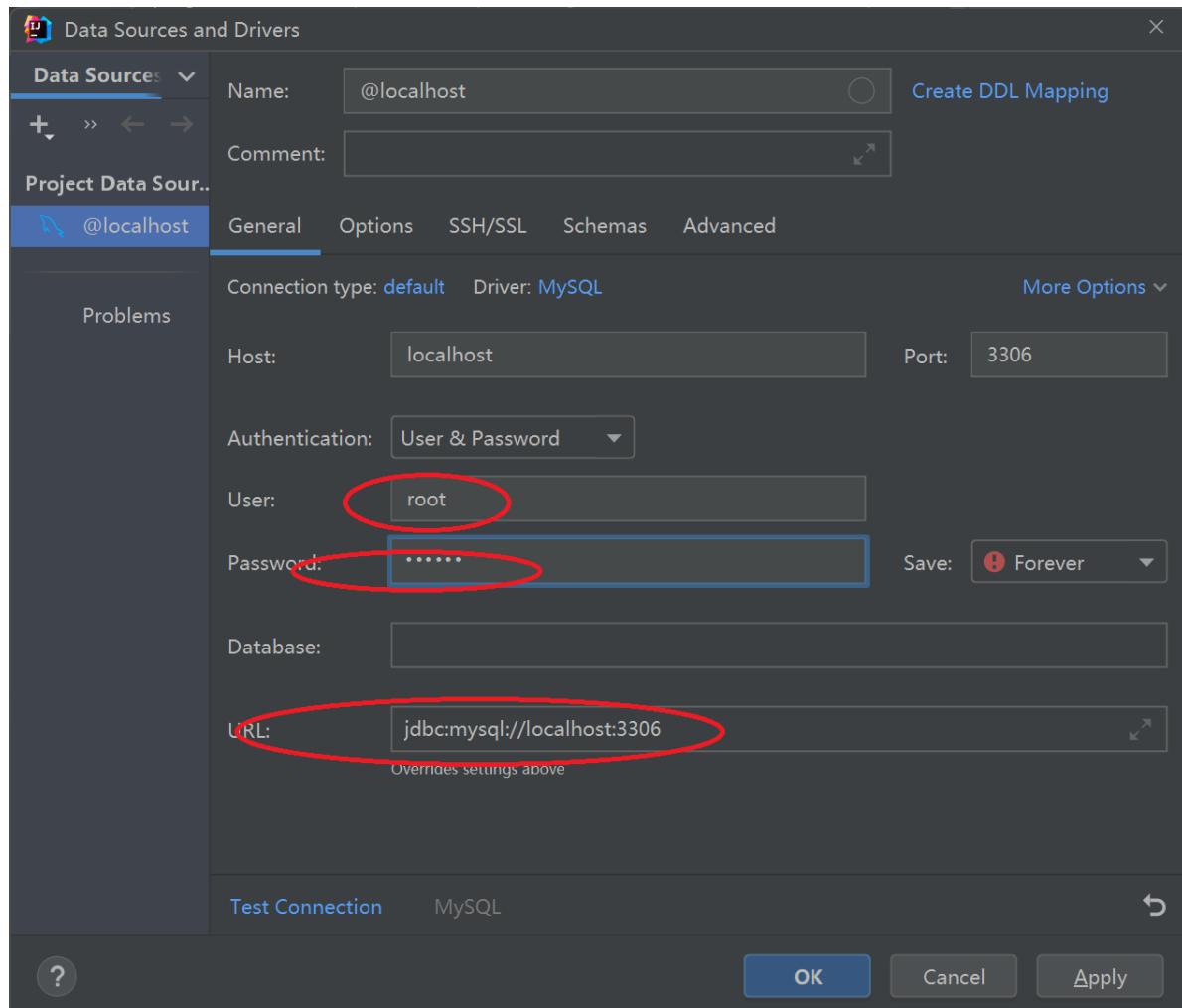


图6 填写数据库账户名和密码

数据库的root用户和密码是你在安装mysql数据库时设定的，另外，由于是本地数据库，所以URL地址默认是：

```
1 | jdbc:mysql://localhost:3306
```

鼠标选中mysql的localhost，new->schema，创建名为quiz的数据库：

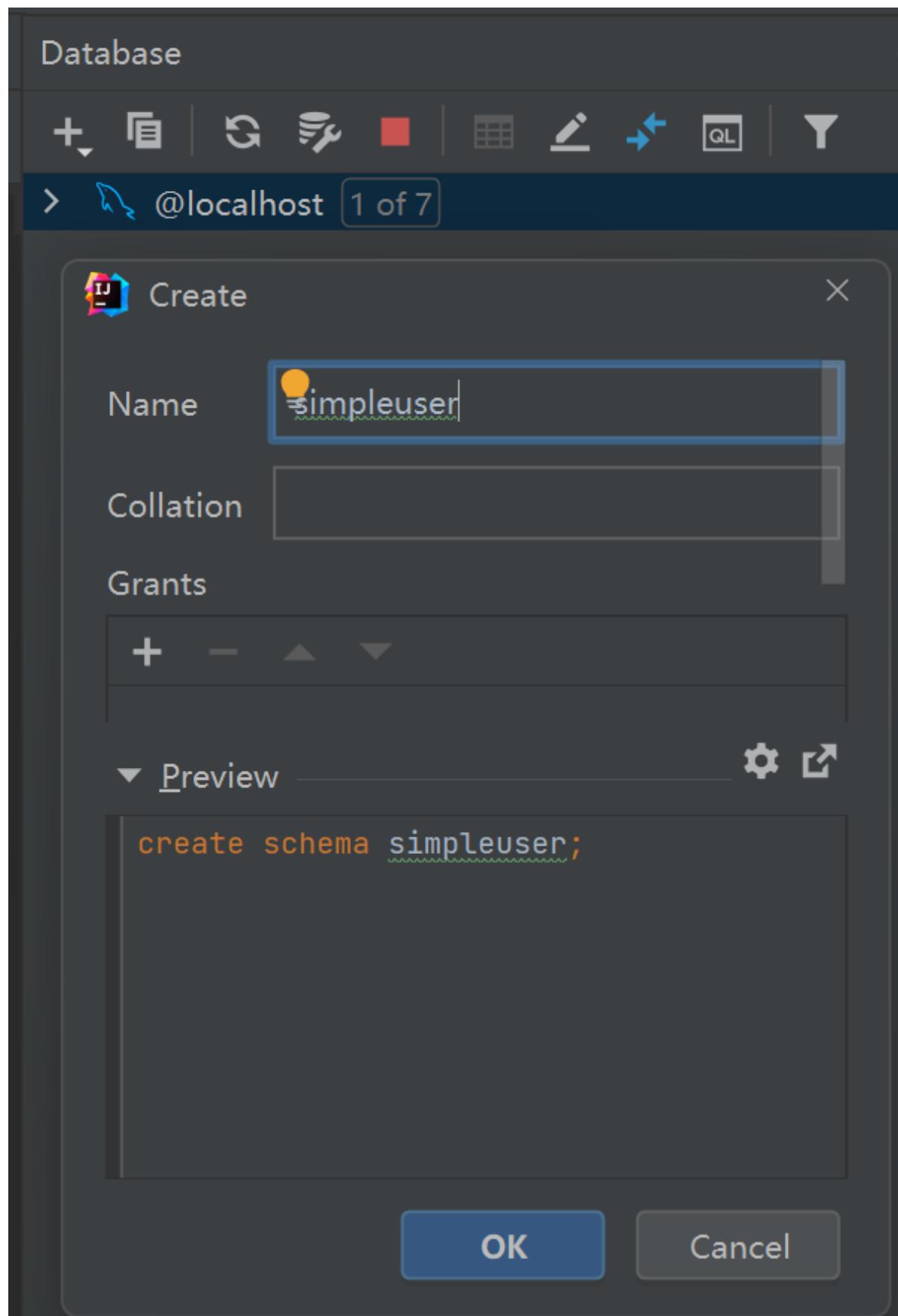


图7 创建数据库

3、MyBatis操作数据库：

步骤1：创建示例数据库

快速地创建一个mytest数据库，然后参照文档创建user表，并插入数据；

```
1 | DROP TABLE IF EXISTS `user`;
```

```

2 CREATE TABLE `user`
3 (
4     id BIGINT NOT NULL COMMENT '主键ID',
5     name VARCHAR(30) NULL DEFAULT NULL COMMENT '姓名',
6     age INT NULL DEFAULT NULL COMMENT '年龄',
7     email VARCHAR(50) NULL DEFAULT NULL COMMENT '邮箱',
8     PRIMARY KEY (id)
9 );
10
11
12 DELETE FROM `user`;
13
14 INSERT INTO `user` (id, name, age, email) VALUES
15 (1, 'Jone', 18, 'test1@baomidou.com'),
16 (2, 'Jack', 20, 'test2@baomidou.com'),
17 (3, 'Tom', 28, 'test3@baomidou.com'),
18 (4, 'Sandy', 21, 'test4@baomidou.com'),
19 (5, 'Billie', 24, 'test5@baomidou.com');

```

步骤2：创建数据库对应的实体类

在com.tfzhang.quiz目录下创建model目录，其中添加User.java文件，并输入文档中的内容：

```

1 package com.tfzhang.quiz.model;
2
3 import lombok.Data;
4
5 @Data
6 public class User {
7     private Long id;
8     private String name;
9     private Integer age;
10    private String email;
11 }

```

步骤3：创建MyBatis配置文件

将项目中的application.properites配置修改为application.yml，即只改后缀，主要原因是yml后缀的配置文件书写更灵活；参照mybatis文档，书写配置文件：

```

1 spring:
2   application:
3     name:quiz
4   datasource:
5     driver-class-name: com.mysql.cj.jdbc.Driver
6     url: jdbc:mysql://localhost:3306/mytest
7     username: root
8     password: 123456
9   server:
10    port: 8080

```

注意：数据库采用我们为本次测试创建的mytest数据库；

步骤4：编写SQL语句

在com.tfzhang.quiz目录下创建mapper目录， 创建UserMapper这个接口；

```
1 package com.tfzhang.quiz.mapper;
2
3 import com.tfzhang.quiz.model.User;
4 import org.apache.ibatis.annotations.Mapper;
5 import org.apache.ibatis.annotations.Select;
6
7 import java.util.List;
8
9 @Mapper //运行时，框架会自动实现实现类对象，并将对象实例交由IOC容器管理;
10 public interface UserMapper{
11
12     //查询全部用户;
13     @Select("select * from user")
14     public List<User> list();
15 }
```

需要注意的问题：

- 这里我们只需要定义interface， 不需要写实现类；因为使用了Mapper标签；
- list()方法与sql语句的“select * from user”绑定；通过@Select标签；

步骤5：编写测试类

将我们的测试list()的代码写在test.com.tfzhang.quiz.QuizApplicationTests.java中；

标签@Autowired， 使用的是依赖注入的概念， 具体参见：[Day05-10. 分层解耦-分层解耦\(IOC-DI引入\)哔哩哔哩bilibili](#)

```
1 package com.tfzhang.quiz;
2
3 import com.tfzhang.quiz.mapper.UserMapper;
4 import com.tfzhang.quiz.model.User;
5 import org.junit.jupiter.api.Test;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.boot.test.context.SpringBootTest;
8
9 import java.util.List;
10
11 @SpringBootTest
12 class QuizApplicationTests {
13
14     @Autowired //依赖注入;
15     private UserMapper userMapper;
16
17     @Test
18     public void testListUser(){
19         List<User> userList = userMapper.list();
20         userList.forEach(user->{
21             System.out.println(user);
22         });
23     }
24 }
```

The screenshot shows a Java IDE interface with the following details:

- UserMapper.java:** The code defines a User entity with fields id, name, age, and email. It includes a generated toString() method.
- QuizApplicationTests:** A test class with a ListUser() method that prints the list of users to the console.
- Terminal Output:** The terminal shows the execution of the test and the output of the ListUser() method. The output is:

```
Tests passed: 1 of 1 test - 1 sec 387 ms
User(id=1, name=Jone, age=18, email=test1@baomidou.com)
User(id=2, name=Jack, age=20, email=test2@baomidou.com)
User(id=3, name=Tom, age=28, email=test3@baomidou.com)
User(id=4, name=Sandy, age=21, email=test4@baomidou.com)
User(id=5, name=Billie, age=24, email=test5@baomidou.com)
```

运行测试类后，可以在终端看到5个数据的输出。至此，MyBatis的测试成功。