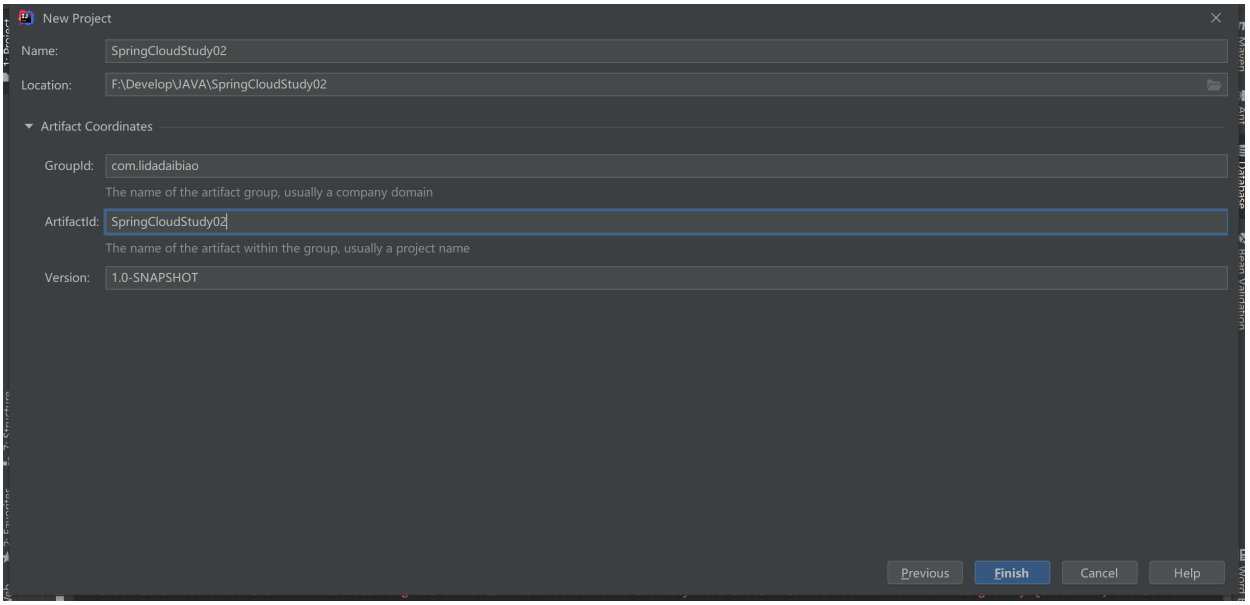
服务提供者

1创建一个Maven父项目



2配置SpringCloudStudy02(父)pom.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.lidadaibiao</groupId>
  <artifactId>SpringCloudStudy02</artifactId>
  <version>1.0-SNAPSHOT</version>
   <packaging>pom</packaging> <!--打包方式改成pom-->
   <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
   <junit.version>4.12</junit.version>
   <lombok.version>1.16.10</lombok.version>
   <log4j.version>1.2.17</log4j.version>
   <logback.version>1.2.3</logback.version>
   </properties> <!--方便存放 依赖包版本号-->
   <dependencyManagement>
   <dependencies>
   <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-dependencies -->
```

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-dependencies</artifactId>
<version>Hoxton.SR1</version>
<type>pom</type>
<scope>runtime</scope>
</dependency>
<!--Springboot-->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
<version>2.1.4.RELEASE</version>
<type>pom</type>
<scope>import</scope>
</dependency>
<!--数据库-->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.47</version>
</dependency>
<!--数据库连接池-->
<dependency>
<groupId>com.alibaba</groupId>
<artifactId>druid</artifactId>
<version>1.1.10</version>
</dependency>
<!--Springboot 启动器-->
<dependency>
<groupId>org.mybatis.spring.boot</groupId>
<artifactId>mybatis-spring-boot-starter</artifactId>
<version>1.3.2</version>
</dependency>
<!--
主要用于日志和测试
-->
<!--junit-->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
```

```
<version>${junit.version}</version>
</dependency>
<!--lombok-->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>${lombok.version}</version>
</dependency>
<!--log4j-->
<dependency>
<groupId>log4j</groupId>
<artifactId>log4j</artifactId>
<version>${log4j.version}</version>
</dependency>
<dependency>
<groupId>ch.qos.logback</groupId>
<artifactId>logback-core</artifactId>
<version>${logback.version}</version>
</dependency>
</dependencies>
</dependencyManagement>
</project>
```

dependencyManagement里只是声明依赖，并不实现引入，因此子项目需要显示的声明需要用的依赖。如果不在子项目中声明依赖，是不会从父项目中继承下来的；只有在子项目中写了该依赖项，并且没有指定具体版本，才会从父项目中继承该项，并且version和scope都读取自父pom;另外如果子项目中指定了版本号，那么会使用子项目中指定的jar版本

dependencies即使在子项目中不写该依赖项，那么子项目仍然会从父项目中继承该依赖项（全部继承

<dependencies>中的jar直接加到项目中，管理的是依赖关系（如果有父pom,子pom,则子pom中只能被动接受父类的版本）；<dependencyManagement>主要管理版本，对于子类继承同一个父类是很有用的，集中管理依赖版本不添加依赖关系，对于其中定义的版本，子pom不一定要继承父pom所定义的版本

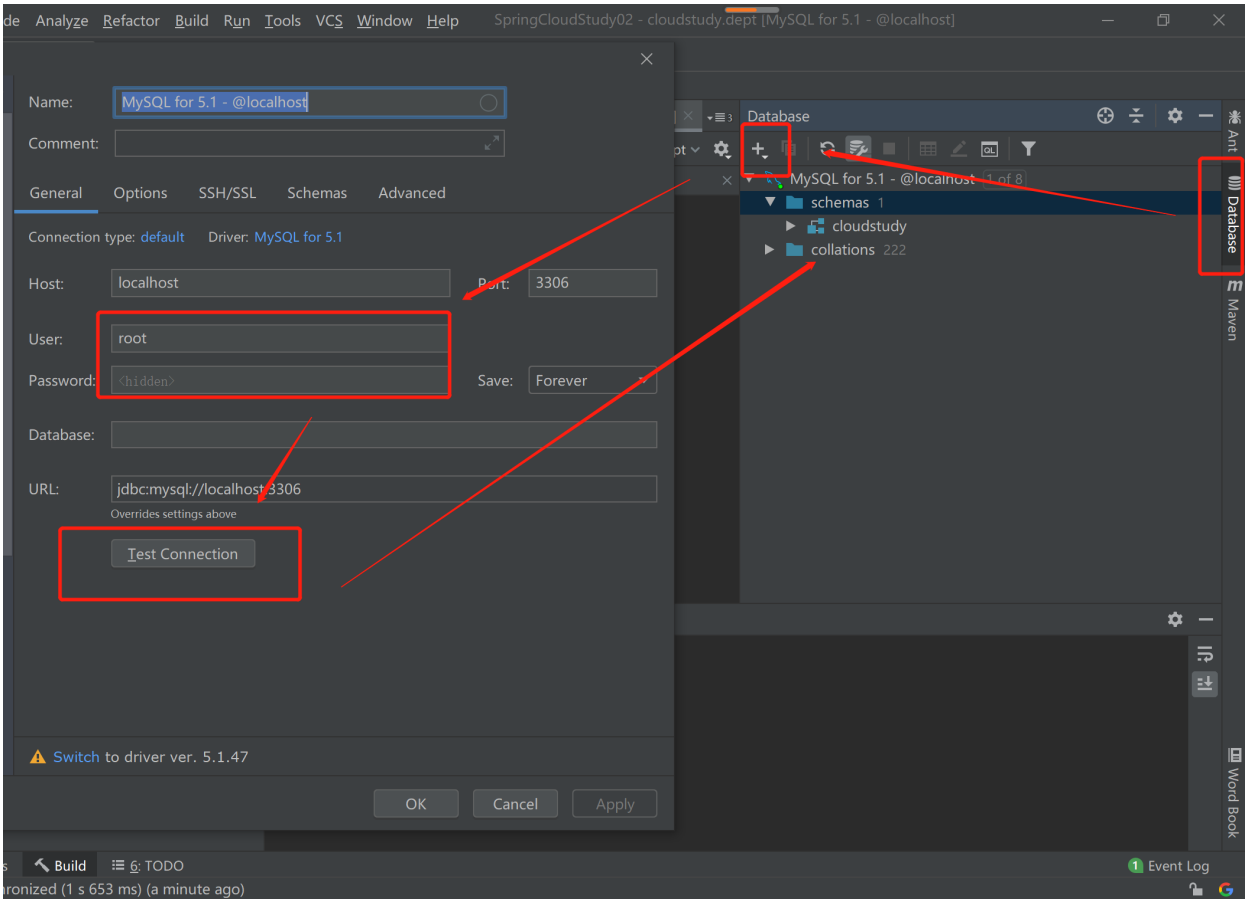
3创建子服务SpringCloud-api（主要是放一些实体类）

3.1编写子服务的pom.xml

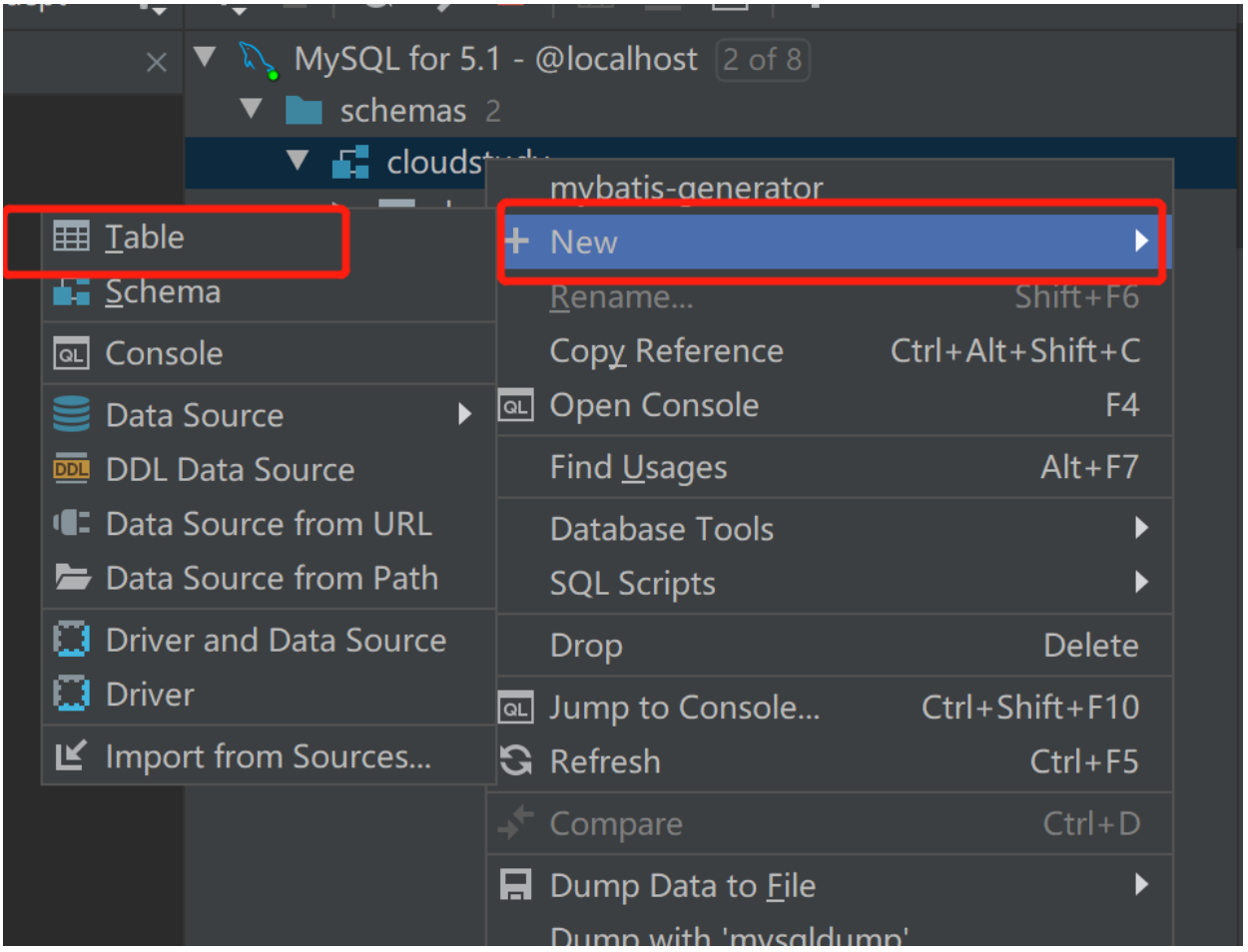
```
<parent>
```

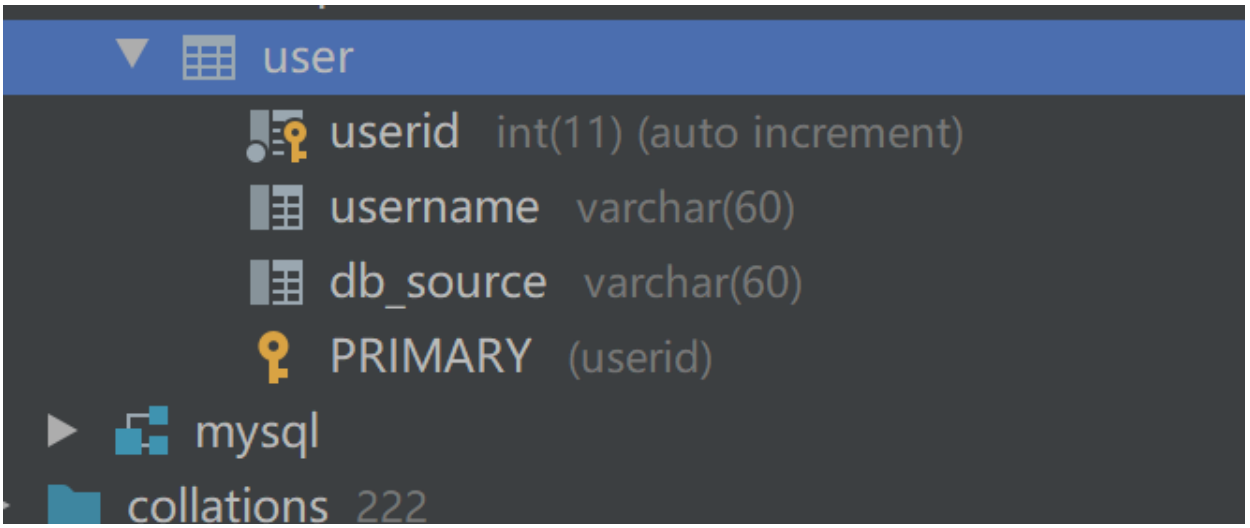
```
<artifactId>SpringCloudStudy02</artifactId> //这里指出继承了SpringCloudStudy02
<groupId>com.lidadaibiao</groupId>
<version>1.0-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>SpringCloud-api</artifactId>
<dependencies>
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 </dependency>
 </dependencies>

</project>
```

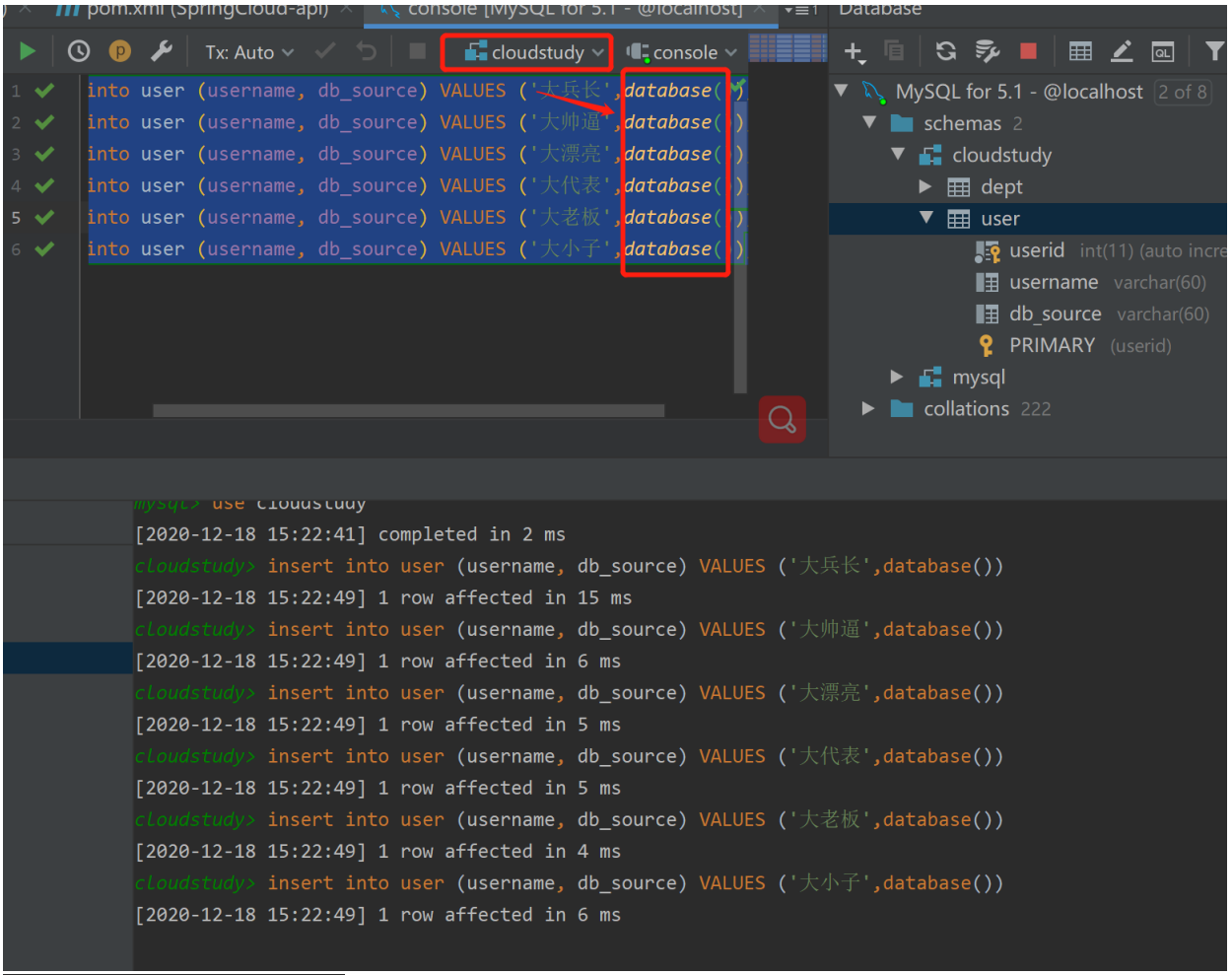
3.2通过idea连接数据库



3.3通过idea创建测试表


MySQL for 5.1 - @localhost 2 of 8
schemas 2
Table
Schema
Console
Data Source
DDL Data Source
Data Source from URL
Data Source from Path
Driver and Data Source
Driver
Import from Sources...
mybatis-generator
New
Rename... Shift+F6
Copy Reference Ctrl+Alt+Shift+C
Open Console F4
Find Usages Alt+F7
Database Tools
SQL Scripts
Drop Delete
Jump to Console... Ctrl+Shift+F10
Refresh Ctrl+F5
Compare Ctrl+D
Dump Data to File


user
userid int(11) (auto increment)
username varchar(60)
db_source varchar(60)
PRIMARY (userid)
mysql
collations 222



database() 意指当前的数据库

3.4编写实体类

```
@Data
@NoArgsConstructor
@Accessors(chain = true) //开启链式编程
public class User implements Serializable {//Dept 实体类 orm 类表关系映射 需要转成二进制字节进行传输
  private int userid; //主键
  private String username;
  //这个数据库存在那个数据库的字段~ 微服务 一个服务对应一个数据库 可能存在不同的数据库
  private String db_source;
  public User(String username){
   this.username = username;
   }
}
```

4创建服务提供者SpringCloud-provider-dept-8001（主要存放dao,service,controller)

4.1编写子服务的pom.xml

```xml
<parent>
 <artifactId>SpringCloudStudy02</artifactId>
 <groupId>com.lidadaibiao</groupId>
 <version>1.0-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>SpringCloud-Provider-Dept-9001</artifactId>
<dependencies>
  <!--我们需要拿到实体类，所以需要配置API module-->
  <dependency>
  <groupId>com.lidadaibiao</groupId>
  <artifactId>SpringCloud-api</artifactId>
  <version>1.0-SNAPSHOT</version>
  </dependency>
  <!--junit-->
  <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  </dependency>
  <!--拿去数据的-->
  <dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  </dependency>
  <dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  </dependency>
  <dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-core</artifactId>
  </dependency>
  <dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  </dependency>
  <dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
```

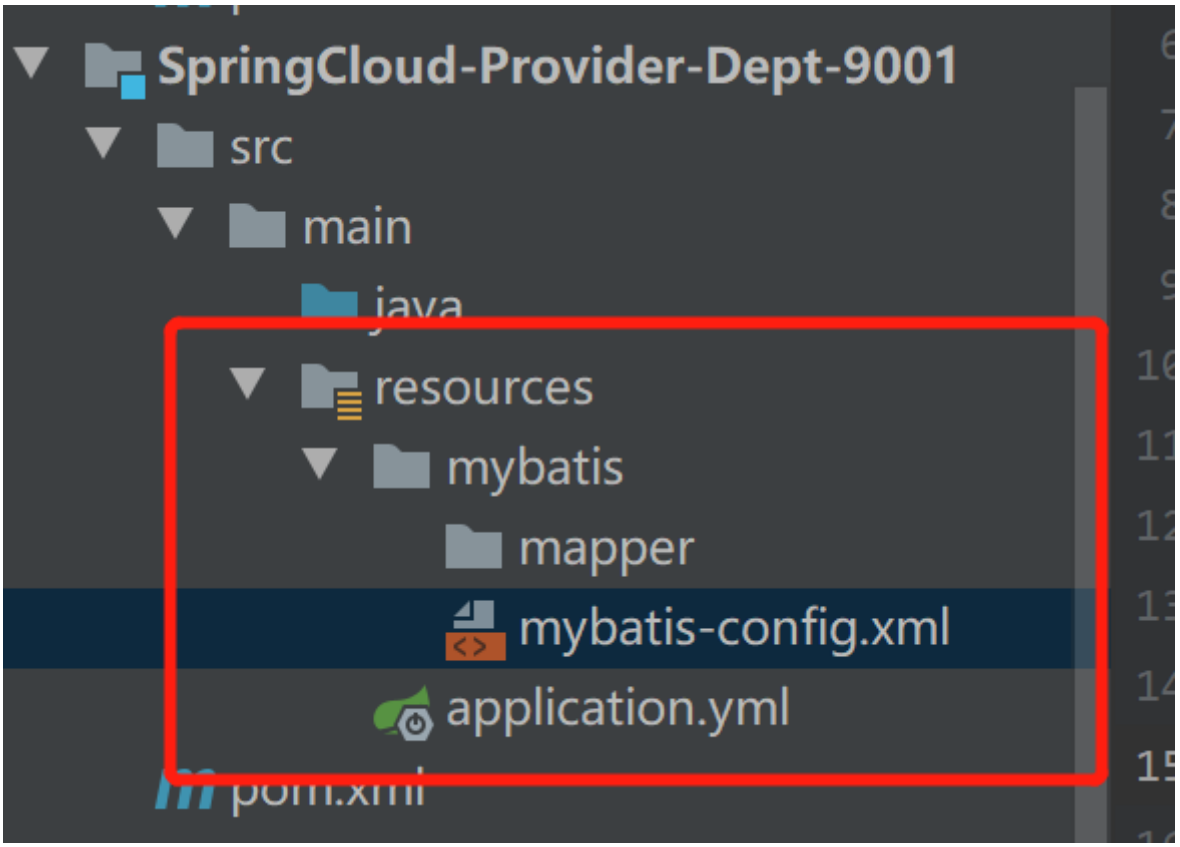
```
  </dependency>
  <!--test-->
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-test</artifactId>
  </dependency>
  <!--web相关-->
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <!--jetty 类似tomcat -->
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jetty</artifactId>
  </dependency>
  <!--热部署-->
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  </dependency>
</dependencies>
```

4.2编写springboot yml文件（application.yml）并创建相关.xml文件、

application.yml基本配置

```
server:
 port: 9001 #服务端口
#mybatis配置
mybatis:
 type-aliases-package: com.lidadaibiao.springcloud.pojo #扫描包
 mapper-locations: classpath:mybatis/mapper/*.xml #扫描mapper文件的地址
 config-location: classpath:mybatis/mybatis-config.xml #扫描加载config文件

#spring配置
spring:
 application:
 name: SpringCloud-Provider-Dept #服务名
 datasource: #数据库相关配置
 type: com.alibaba.druid.pool.DruidDataSource
 driver-class-name: org.gjt.mm.mysql.Driver
```

```
16   url: jdbc:mysql://localhost:3306/cloudstudy?useUnicode=true&characterEncoding=utf-8
17   username: root
18   password: root
```



4.3编写dao层

```
1  @Mapper
2  @Repository
3  public interface UserDao {
4   Boolean addUser(User user);
5   User queryById(int userid);
6   List<User> queryAll();
7  }
8  UserMapper.xml
9  <?xml version="1.0" encoding="UTF-8"?>
10  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
11  <mapper namespace="com.lidadaibiao.springcloud.dao.UserDao">
12   <select id="queryAll" resultType="user">
13   select * from user
14   </select>
15   <select id="queryById" resultType="user" parameterType="int">
16   select * from user where userid = #{userid}
```

```
 </select>
 <select id="addUser" parameterType="user">
 insert into user (username, db_source) values (#(username),database())
 </select>

</mapper>
```

4.4编写service层

```
public interface UserService {
 Boolean addUser(User user);
 User queryById(int userid);
 List<User> queryAll();
}
@Service
public class UserServiceImpl implements UserService {
 @Autowired
 private UserDao userDao;
 @Override
 public Boolean addUser(User user) {
 return userDao.addUser(user);
 }

 @Override
 public User queryById(int userid) {
 return userDao.queryById(userid);
 }

 @Override
 public List<User> queryAll() {
 return userDao.queryAll();
 }
}
```

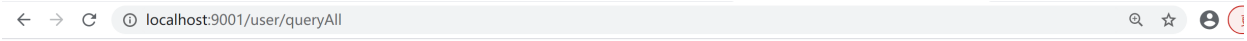
4.5编写Controller层

```
@RestController
public class UserController {
 @Autowired
 private UserService userService;

 @PostMapping("/user/adduser")
 public Boolean addUser(User user){
 return userService.addUser(user);
```

```
9   }
10   @GetMapping("user/query/{id}")
11   public User queryById(@PathVariable("id")int userid){
12   return userService.queryById(userid);
13   }
14   @GetMapping("user/queryAll")
15   public List<User> queryAll(){
16   return userService.queryAll();
17   }
18  }
```

4.6编写启动类，并测试

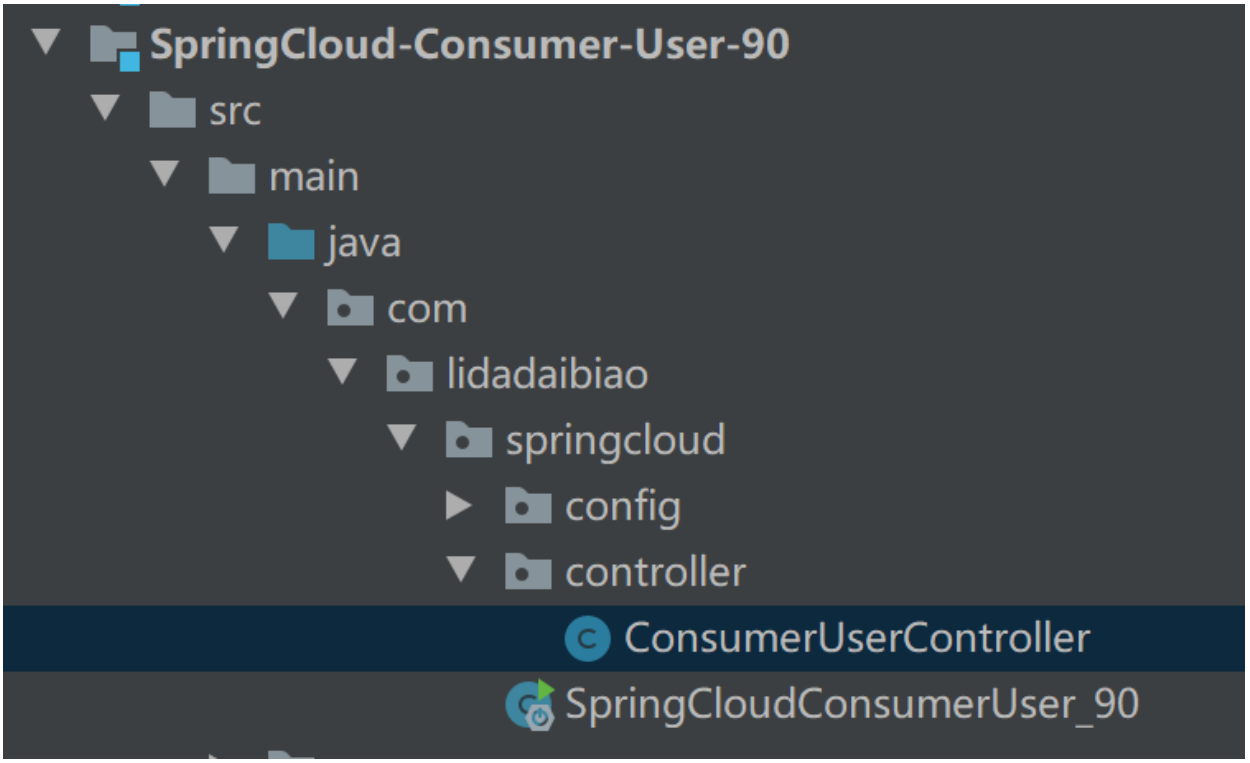
```
1  @SpringBootApplication
2  public class SpringCloudDeptApplication {
3   public static void main(String[] args) {
4   SpringApplication.run(SpringCloudDeptApplication.class,args);
5   }
6  }
```




```
[{"userid":1,"username":"大兵长","db_source":"cloudstudy"},
{"userid":2,"username":"大帅逼","db_source":"cloudstudy"},
{"userid":3,"username":"大漂亮","db_source":"cloudstudy"},
{"userid":4,"username":"大代表","db_source":"cloudstudy"},
{"userid":5,"username":"大老板","db_source":"cloudstudy"},
{"userid":6,"username":"大小子","db_source":"cloudstudy"}]
```


服务消费者（http）（相对dubbo,zookeeper的rpc更加粗暴简单）

1创建子服务（消费）



1.1编写pom.xml文件和application.yml

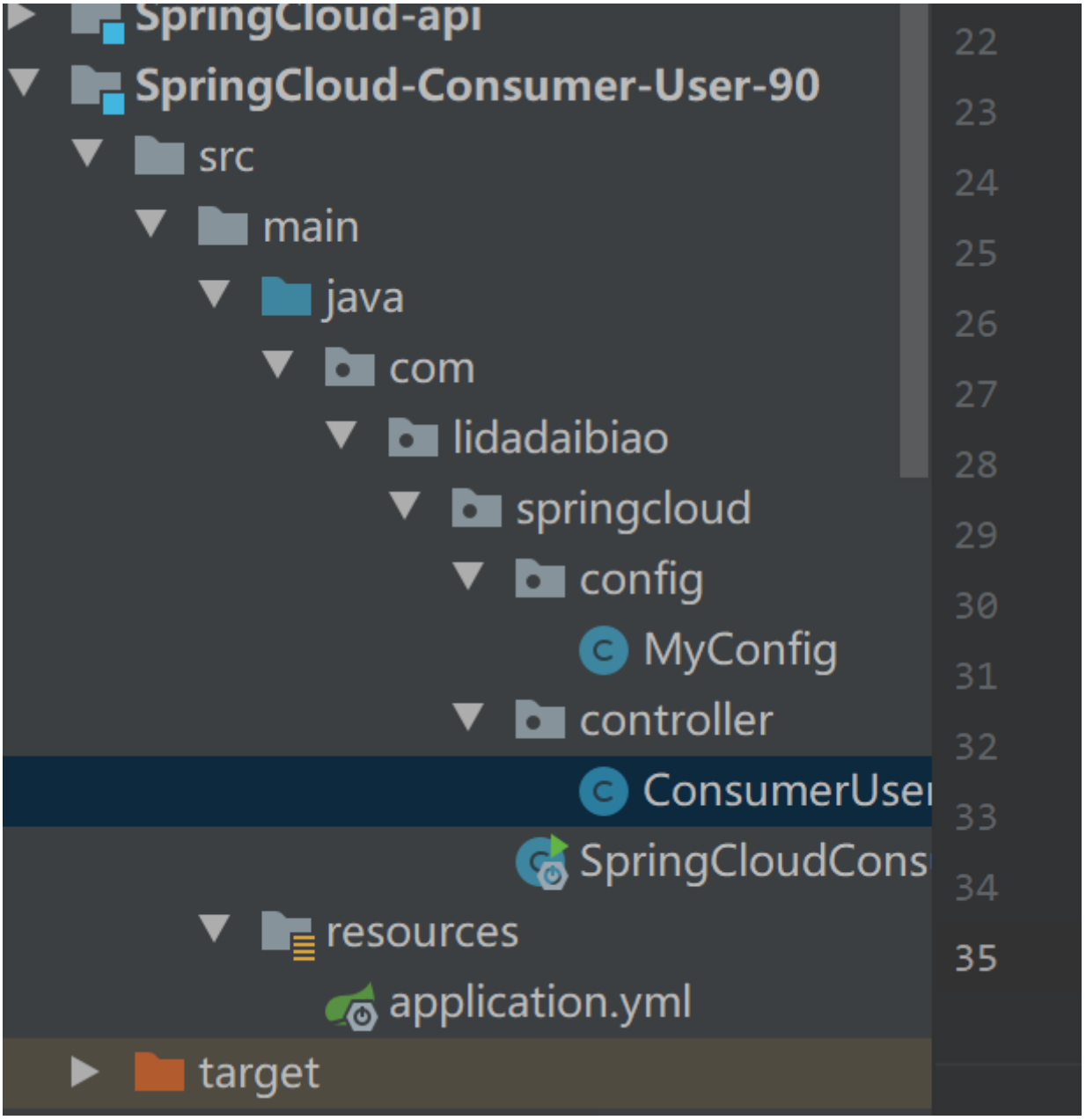
```
server:
 port: 90
```

```
<dependencies>
 <!--实体类和web相关的-->
 <dependency>
 <groupId>com.lidadaibiao</groupId>
 <artifactId>SpringCloud-api</artifactId>
 <version>1.0-SNAPSHOT</version>
 </dependency>
 <!--web相关-->
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <!--热部署-->
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  </dependency>
</dependencies>
```

1.2创建Controller和编写config文件

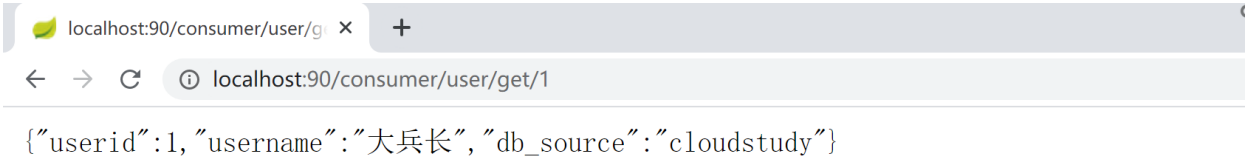
```java
@Configuration
public class MyConfig {
 //这里其实就相当于在spring中applicationContext中配置<bean>一样的道理
 @Bean
 public RestTemplate getRestTemplate(){
 return new RestTemplate();
 }
}
```

```java
@RestController
public class ConsumerUserController {
 //消费者不应该有service层，我们应该直接去调用服务提供层对应的服务
 //这里使用RestTemplate，查看源码我们知道它是没有被注入的spring容器中的，所以我们需要手动注入
 //注意的是三个参数 (url,实体：map,Class<T> responseType)
 @Autowired
 private RestTemplate restTemplate;

 private static final String REST_URL_PREFIX="http://localhost:9001";

 @RequestMapping("/consumer/user/list")
 public List<User> userList(){
 return restTemplate.getForObject(REST_URL_PREFIX+"/user/queryAll",List.class);
 }
 @RequestMapping("/consumer/user/get/{id}")
 public User get(@PathVariable("id")int id){
 return restTemplate.getForObject(REST_URL_PREFIX+"/user/query/"+id,User.class);
 }
 @RequestMapping("/consumer/user/add")
 public boolean add(User user){
 return restTemplate.postForObject(REST_URL_PREFIX+"/user/adduser",user,Boolean.class);
 }
}
```



2编写启动类和测试

```
@SpringBootApplication
public class SpringCloudConsumerUser_90 {
 public static void main(String[] args) {
 SpringApplication.run(SpringCloudConsumerUser_90.class,args);
 }
}
```



{"userid":1,"username":"大兵长","db_source":"cloudstudy"}