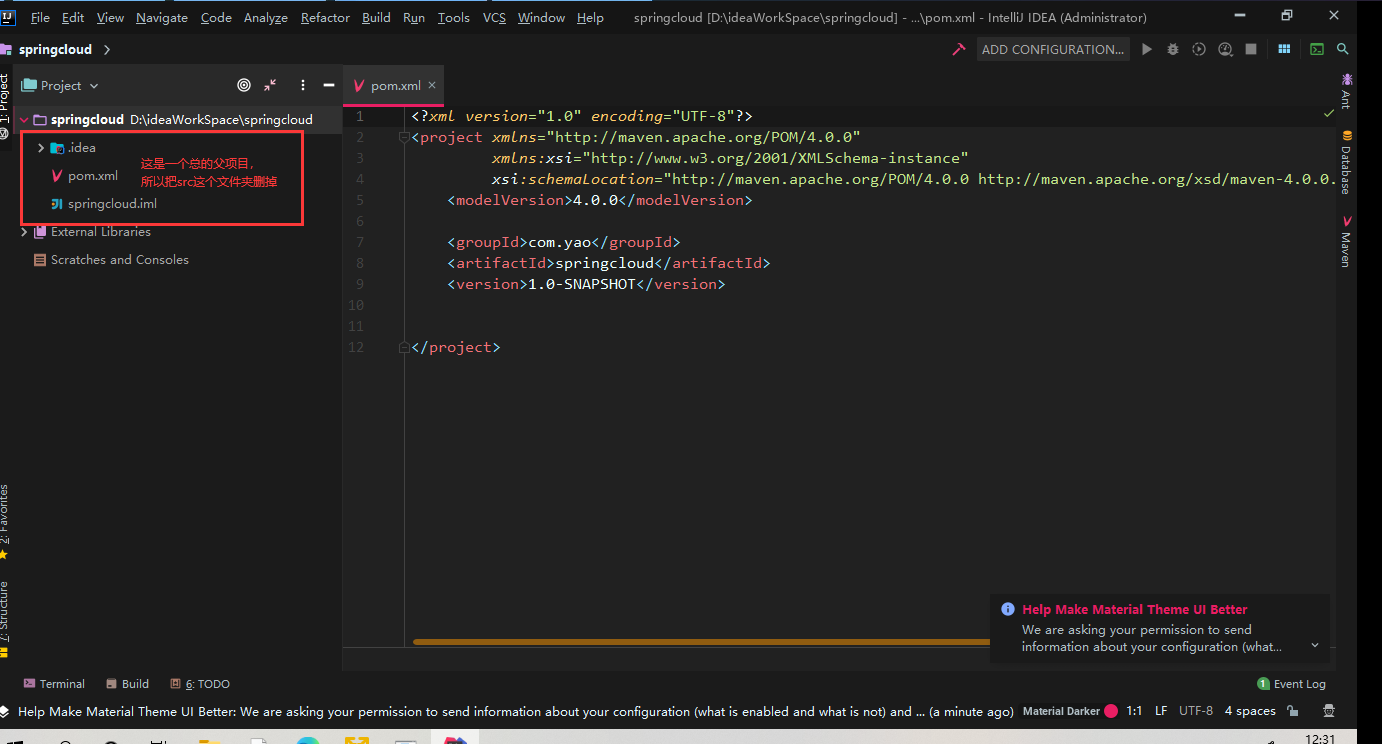
**[Springcloud工程搭建](https://www.cnblogs.com/yaoyaoo/p/14293695.html)**

**1. 创建一个maven项目**

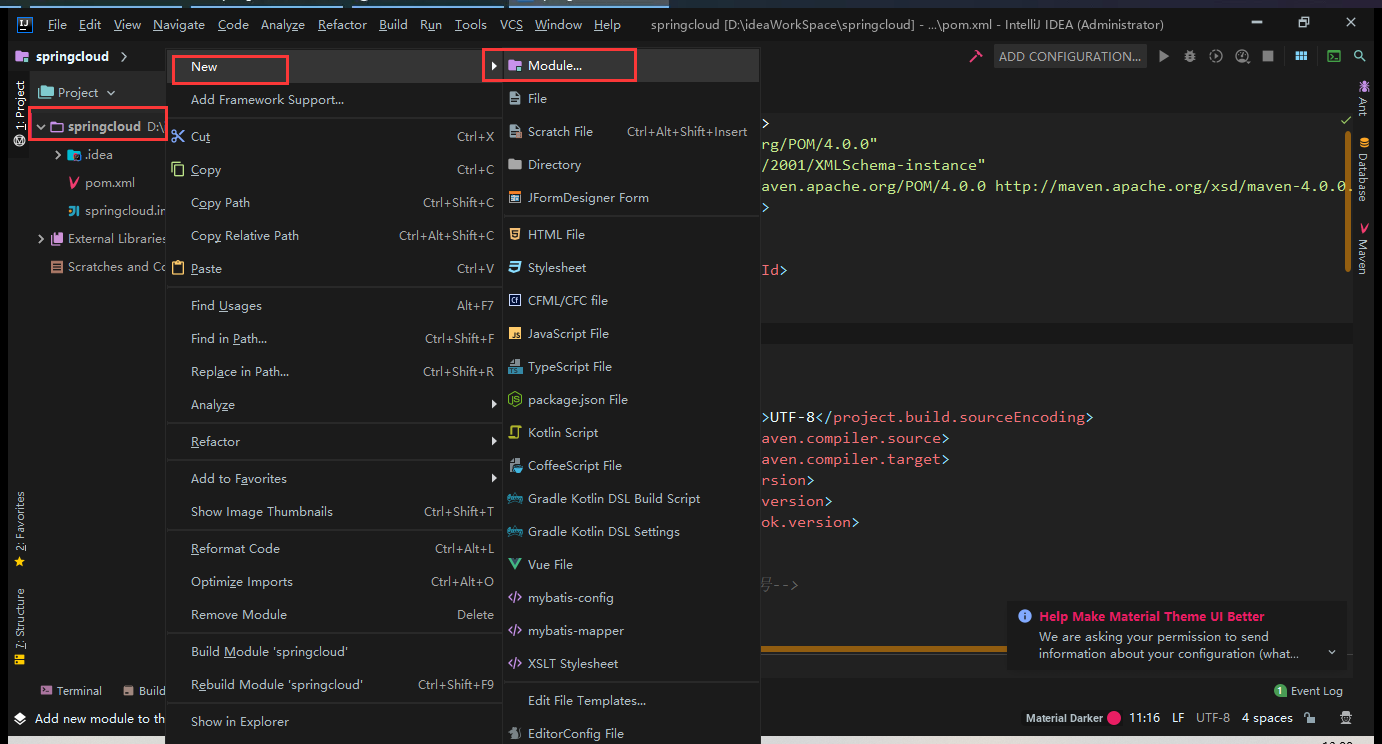
**1.1 删除src**

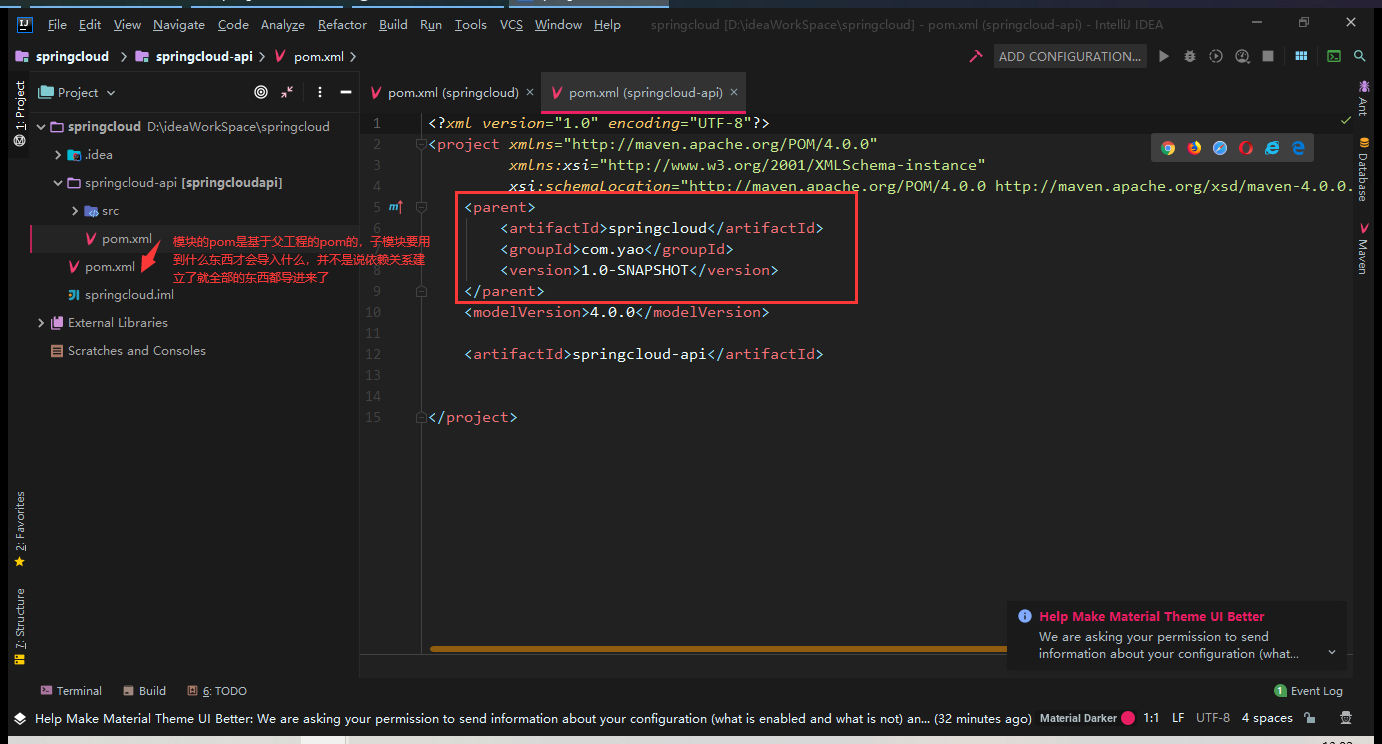


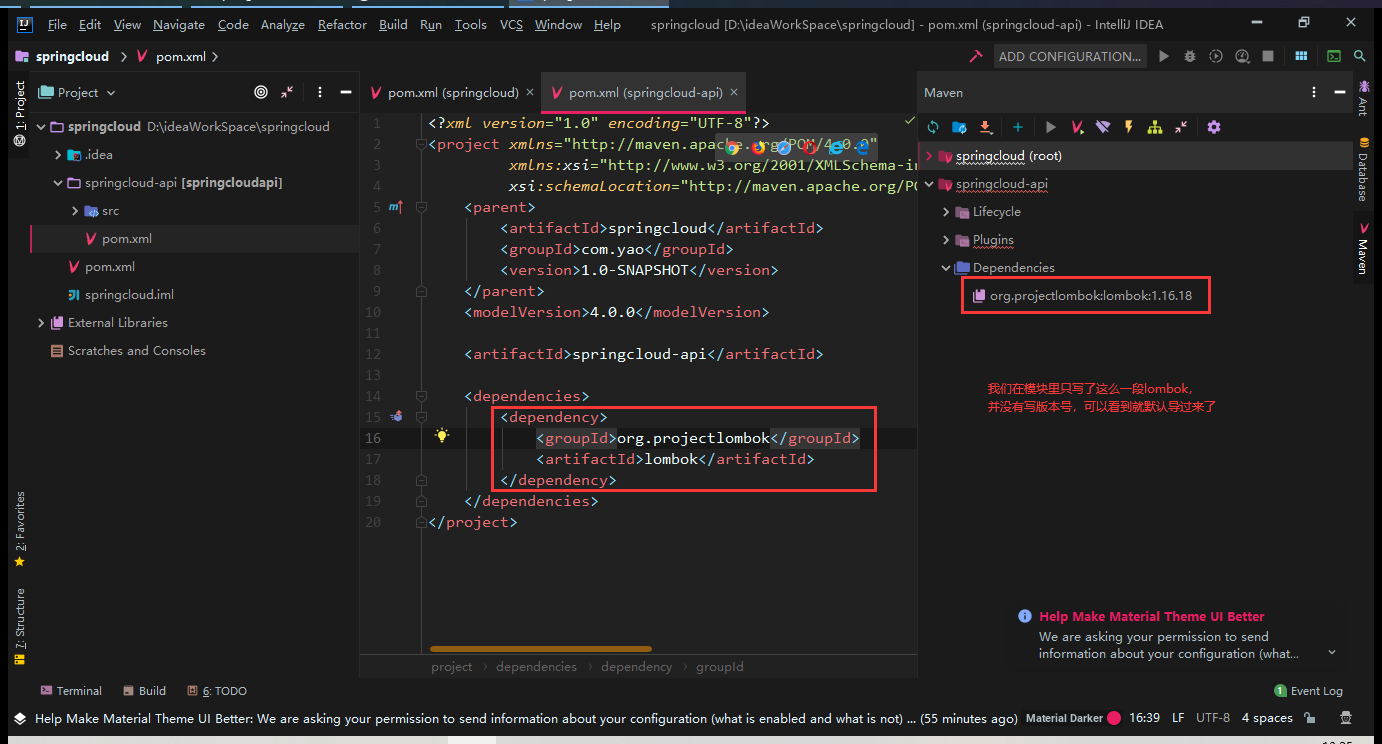
**1.2 父工程pom.xml的配置**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87 | <?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.yao</groupId>      <artifactId>springcloud</artifactId>      <version>1.0-SNAPSHOT</version>        <!--打包方式-->      <packaging>pom</packaging>        <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>          <log4j.version>1.2.17</log4j.version>          <lombok.version>1.16.18</lombok.version>      </properties>        <!--这个只管理版本，子项目不用写版本号-->      <dependencyManagement>          <dependencies>              <!--springCloud的依赖-->              <dependency>                  <groupId>org.springframework.cloud</groupId>                  <artifactId>spring-cloud-dependencies</artifactId>                  <version>Greenwich.SR1</version>                  <type>pom</type>                  <scope>import</scope>              </dependency>              <!--springboot依赖-->              <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>              <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>              <!--日志和测试~-->              <dependency>                  <groupId>ch.qos.logback</groupId>                  <artifactId>logback-core</artifactId>                  <version>1.2.3</version>              </dependency>              <dependency>                  <groupId>junit</groupId>                  <artifactId>junit</artifactId>                  <version>${junit.version}</version>              </dependency>              <dependency>                  <groupId>log4j</groupId>                  <artifactId>log4j</artifactId>                  <version>${log4j.version}</version>              </dependency>              <dependency>                  <groupId>org.projectlombok</groupId>                  <artifactId>lombok</artifactId>                  <version>${lombok.version}</version>              </dependency>          </dependencies>      </dependencyManagement>        <build>        </build>    </project> |

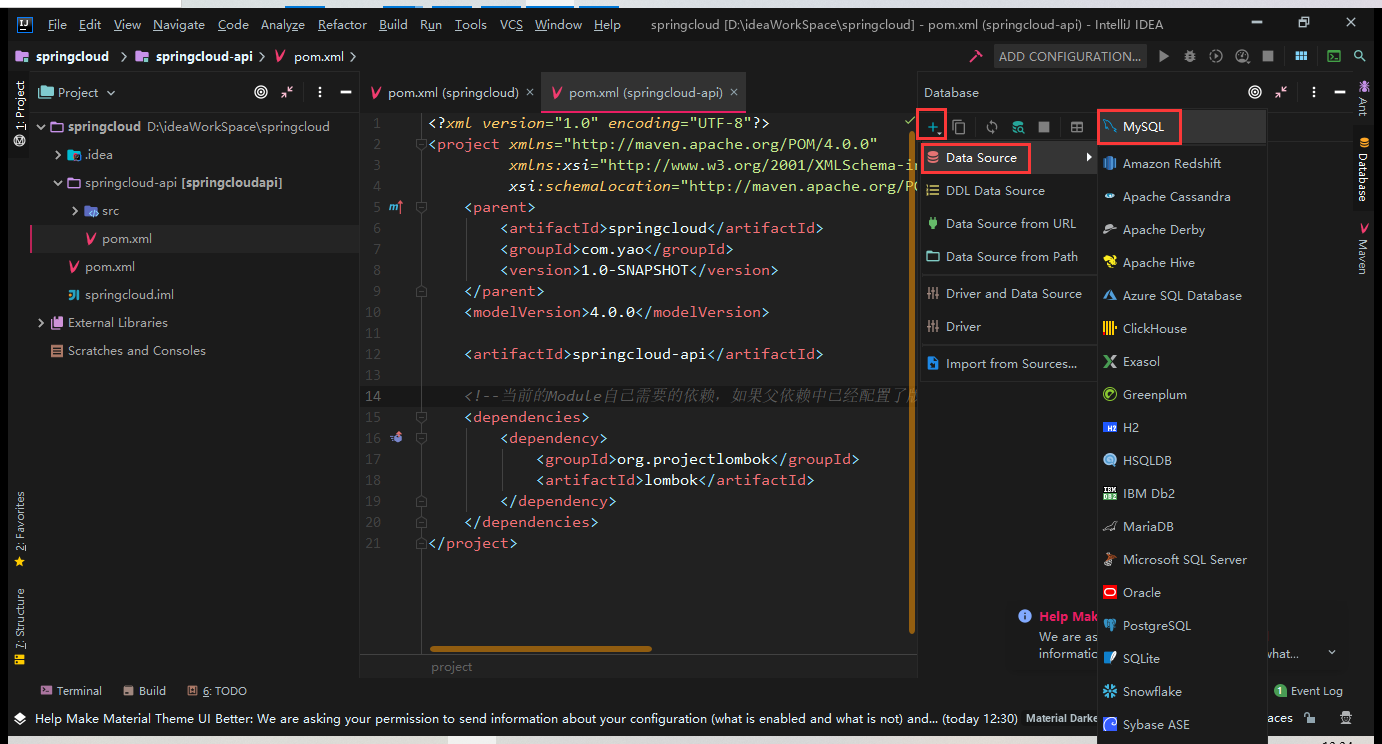
**1.3 创建第一个模块**

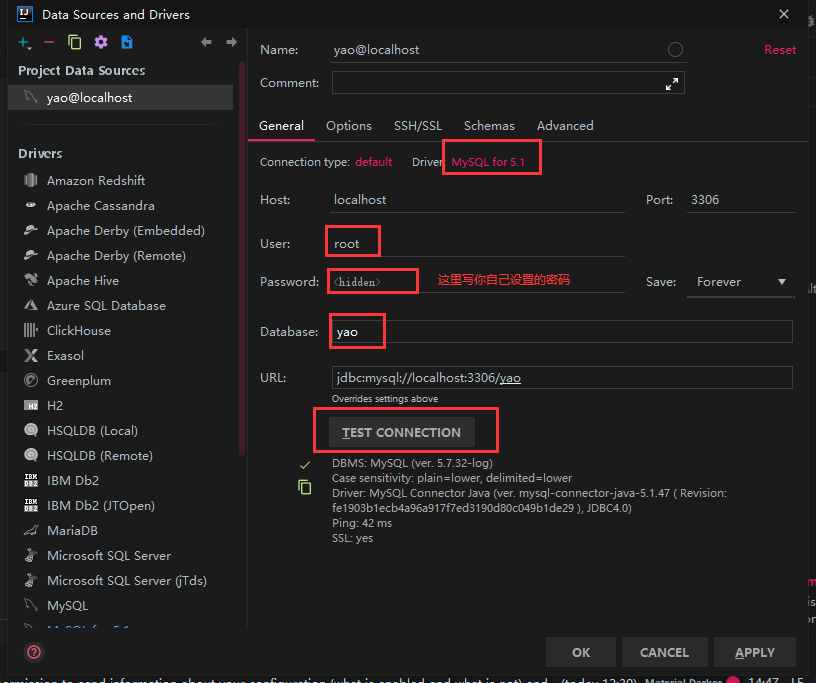




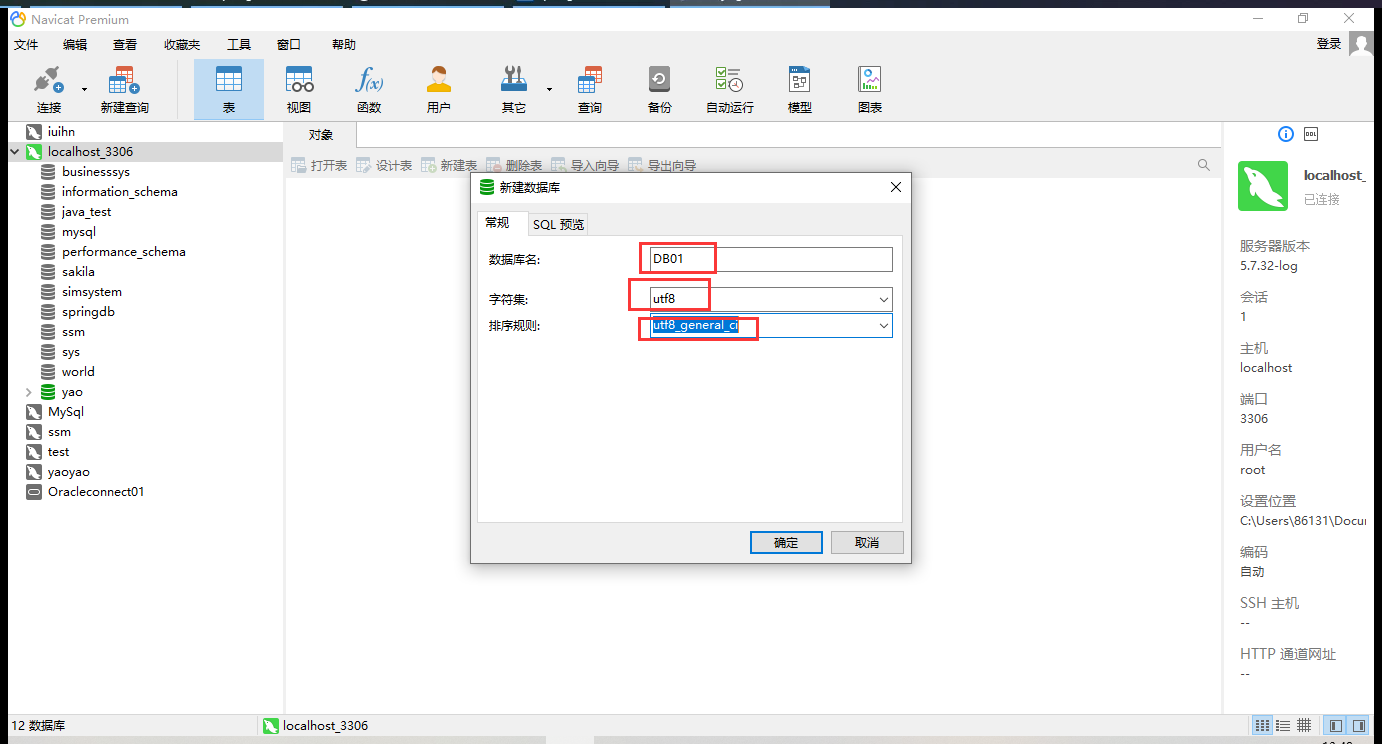


**1.3.1 连接mysql数据库**

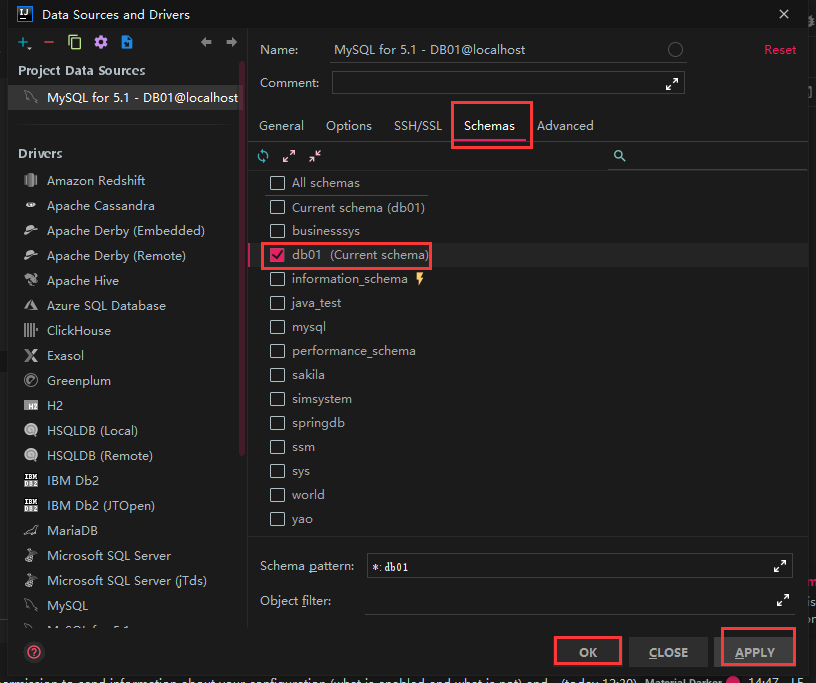


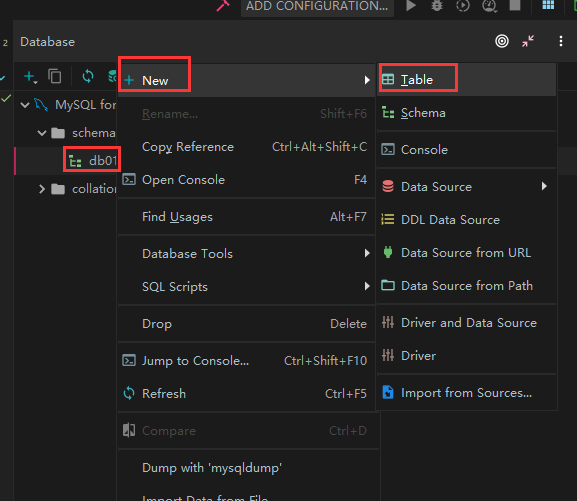


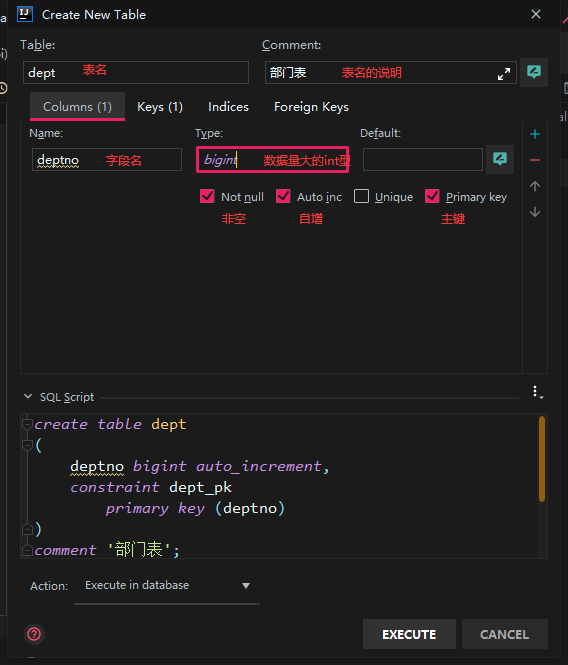
**1.3.1 由于数据库里还没有东西，我们去创建一个数据库**



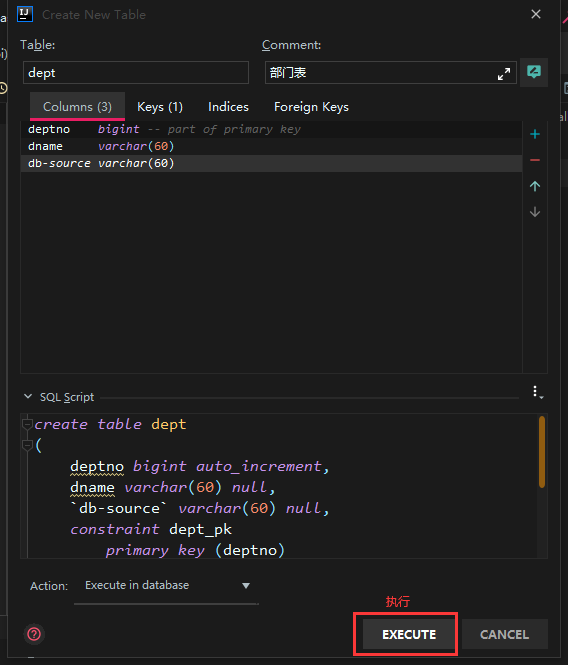
**1.3.2 在idea中连接数据库并创建表**





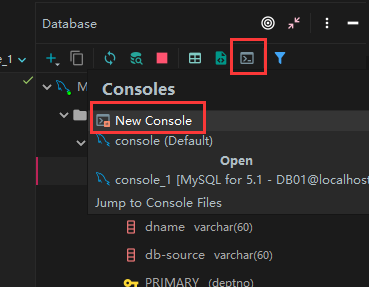


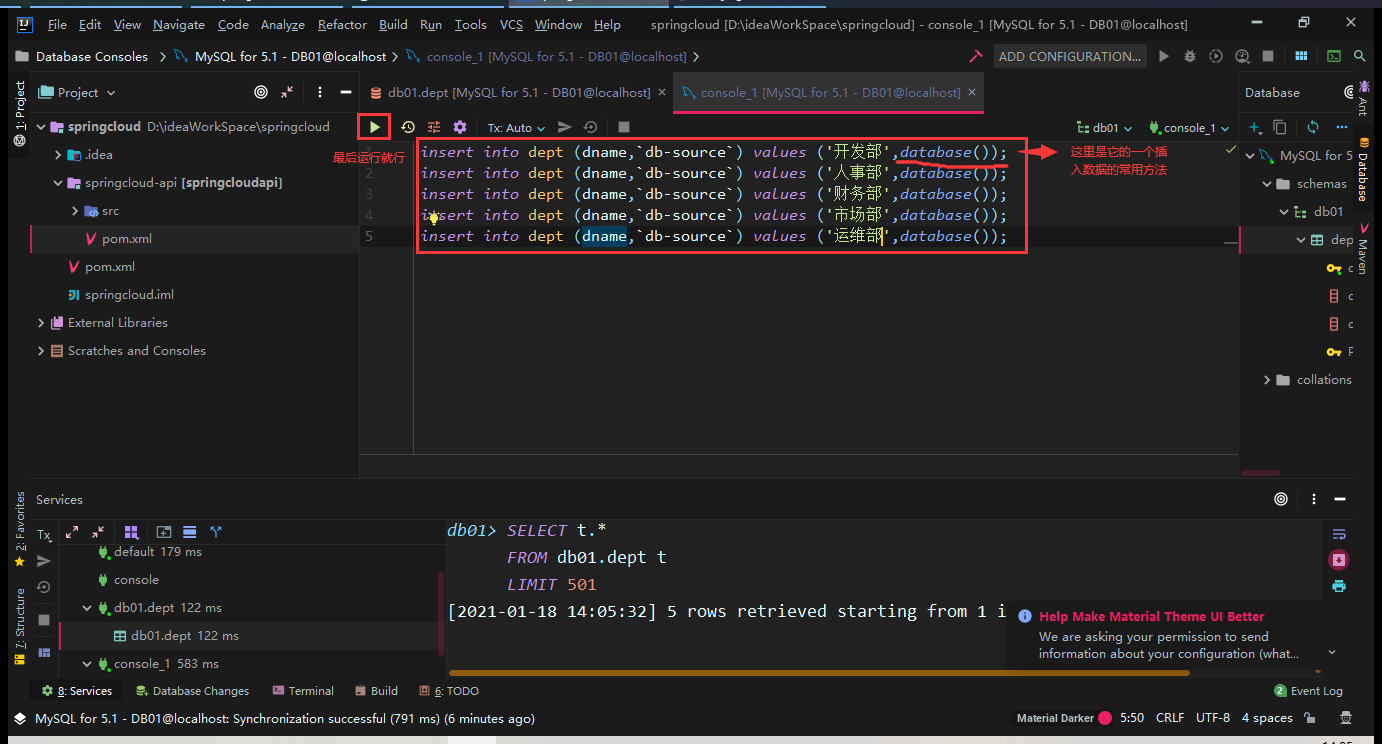
 类似这样去创建一些字段



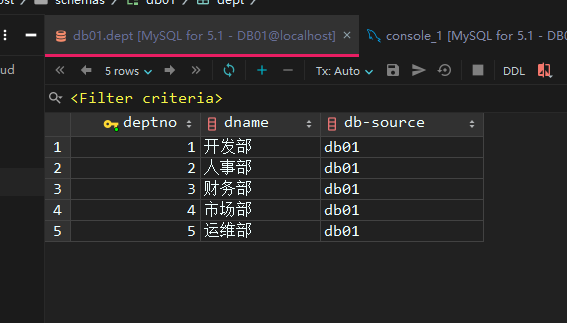
 这样这个表就创建好了

**1.3.3 打开命令行插入一些数据**



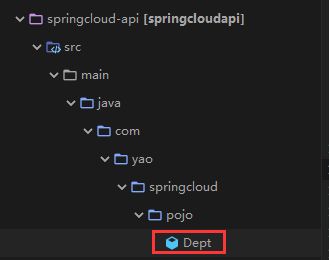


这个函数返回值代表当前数据库



 可以看到插入成功了

**1.3.4 表我们已经创建好了，我们来创建一个实体类**

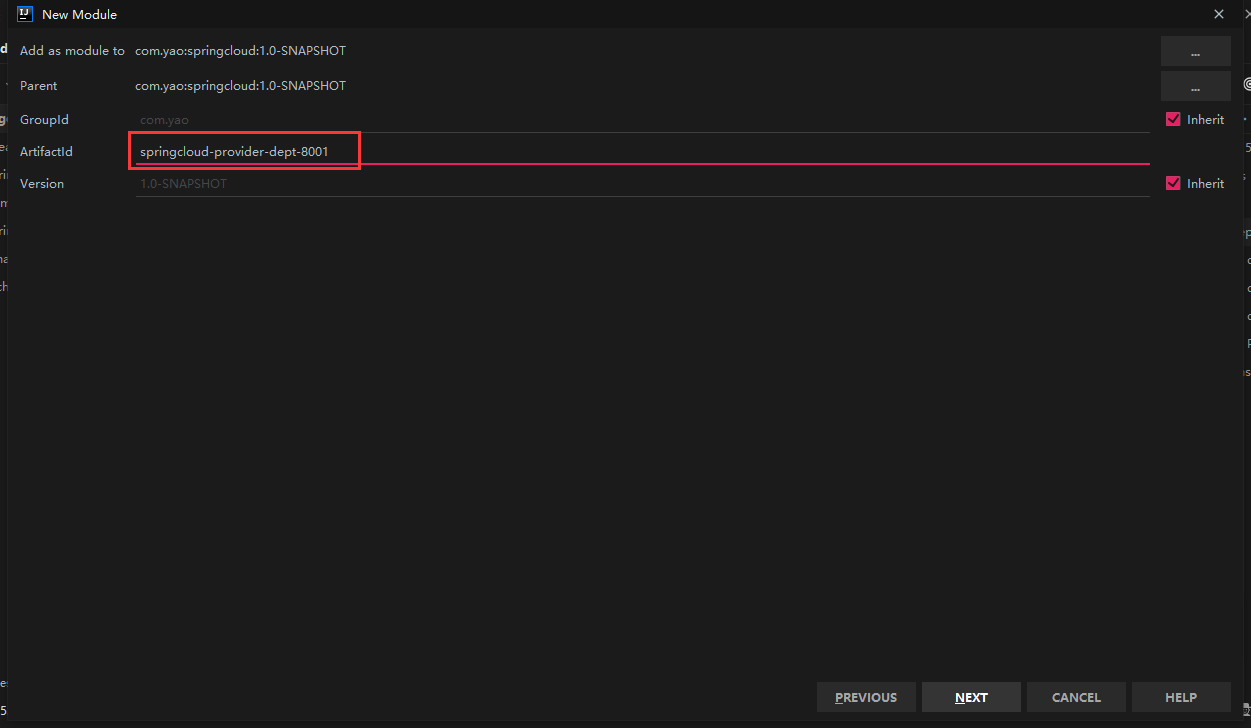


 代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package com.yao.springcloud.pojo;    import lombok.Data;  import lombok.NoArgsConstructor;  import lombok.experimental.Accessors;    import java.io.Serializable;    @Data  @NoArgsConstructor  @Accessors(chain = true) //链式写法,写了这个注解也就是可以用链式的方式来写了  public class Dept implements Serializable { //实体类,orm对象关系映射，类表关系映射      private Long deptno;      private String dname;      //这个数据存在那个数据库的字段，因为微服务是一个服务对应于一个数据库，同一个信息可能存在不同的数据库      private String db\_source;        //构造器我们这里只需要一个就够了      public Dept(String dname) {          this.dname = dname;      }        /\*      链式写法：          Dept dept = new Dept();          dept.setDeptNo(11)              .setDname('222')              .setDb\_source('001')       \*/  } |

以上springcloud-api就写完了，我们接下来写下一个模块

**1.4 创建第二个模块**

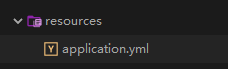


 注意这里要写上端口号

**1.4.1 pom文件：**

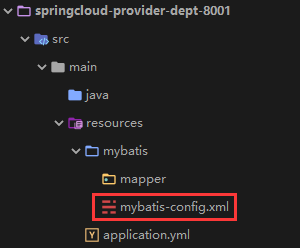
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | <?xml version="1.0" encoding="UTF-8"?>  <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">      <parent>          <artifactId>springcloud</artifactId>          <groupId>com.yao</groupId>          <version>1.0-SNAPSHOT</version>      </parent>      <modelVersion>4.0.0</modelVersion>        <artifactId>springcloud-provider-dept-8001</artifactId>        <dependencies>          <!--因为要拿到实体类，所以要配置api module-->          <dependency>              <groupId>com.yao</groupId>              <artifactId>springcloud-api</artifactId>              <version>1.0-SNAPSHOT</version>          </dependency>          <dependency>              <groupId>junit</groupId>              <artifactId>junit</artifactId>          </dependency>          <dependency>              <groupId>mysql</groupId>              <artifactId>mysql-connector-java</artifactId>          </dependency>          <dependency>              <groupId>com.alibaba</groupId>              <artifactId>druid</artifactId>          </dependency>          <dependency>              <groupId>ch.qos.logback</groupId>              <artifactId>logback-core</artifactId>              <version>1.2.3</version>          </dependency>          <dependency>              <groupId>org.mybatis.spring.boot</groupId>              <artifactId>mybatis-spring-boot-starter</artifactId>          </dependency>          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-test</artifactId>          </dependency>          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-web</artifactId>          </dependency>          <!--jetty-->          <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>    </project> |

**1.4.2 配置文件yml的编写**



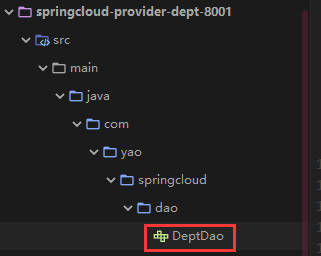
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | server:    port: 8001    #mybatis配置  mybatis:    type-aliases-package: com.yao.springcloud.pojo    config-location: classpath:mybatis/mybatis-config.xml    mapper-locations: classpath:mybatis/mapper/\*.xml    #spring配置  spring:    application:      name: springcloud-provider-dept    datasource:      type: com.alibaba.druid.pool.DruidDataSource      driver-class-name: org.gjt.mm.mysql.Driver      url: jdbc:mysql://localhost:3306/db01?useUnicode=true&characterEncoding=utf-8      username: root      password: 892095368llq |

**1.4.3 mybatis核心配置文件的编写**



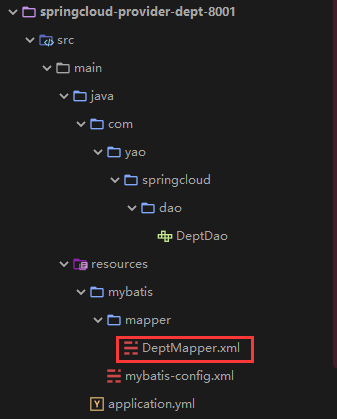
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE configuration          PUBLIC "-//mybatis.org//DTD Config 3.0//EN"          "http://mybatis.org/dtd/mybatis-3-config.dtd">    <configuration>      <settings>          <!--开启二级缓存-->          <setting name="cacheEnabled" value="true"/>      </settings>  </configuration> |

**1.4.4 创建Dao层**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package com.yao.springcloud.dao;    import com.yao.springcloud.pojo.Dept;  import org.apache.ibatis.annotations.Mapper;  import org.springframework.stereotype.Repository;    import java.util.List;    @Mapper  @Repository  public interface DeptDao {      public boolean addDept(Dept dept);        public Dept queryById(Long id);        public List<Dept> queryAll();  } |

**1.4.5 创建并编写mapper文件**

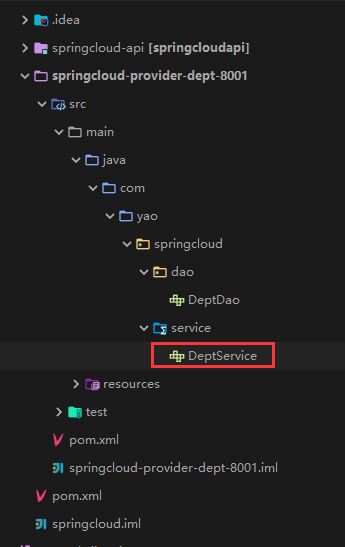


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper          PUBLIC "-//mybatis.org//DTD Config 3.0//EN"          "http://mybatis.org/dtd/mybatis-3-mapper.dtd">    <mapper namespace="com.yao.springcloud.dao.DeptDao">      <insert id="addDept" parameterType="Dept">          insert into dept (dname, "db-source") values (#{dname},DATABASE())      </insert>      <select id="queryById" resultType="Dept" parameterType="Long">          select \* from dept where deptno=#{deptno};      </select>      <select id="queryAll" resultType="Dept">          select \* from dept;      </select>  </mapper> |

以上dao层就写完了。

**1.4.6 创建service层**

**1.4.6.1 写接口**

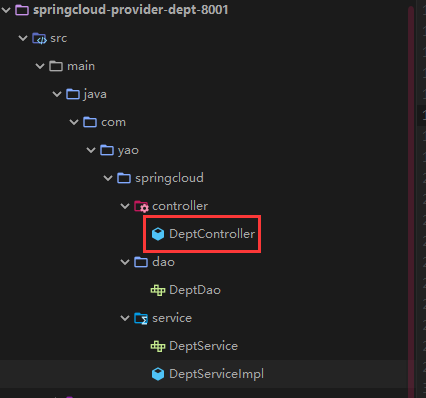


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.yao.springcloud.service;    import com.yao.springcloud.pojo.Dept;    import java.util.List;    public interface DeptService {      public boolean addDept(Dept dept);        public Dept queryById(Long id);        public List<Dept> queryAll();  } |

**1.4.6.2 写实现类**

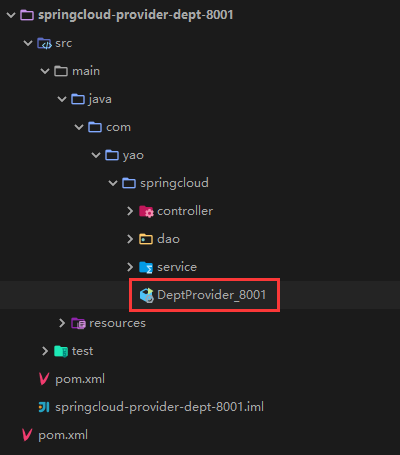
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | package com.yao.springcloud.service;    import com.yao.springcloud.dao.DeptDao;  import com.yao.springcloud.pojo.Dept;  import org.springframework.beans.factory.annotation.Autowired;    import java.util.List;  @Service  public class DeptServiceImpl implements DeptService{        @Autowired      private DeptDao deptDao;        @Override      public boolean addDept(Dept dept) {          return deptDao.addDept(dept);      }        @Override      public Dept queryById(Long id) {          return deptDao.queryById(id);      }        @Override      public List<Dept> queryAll() {          return deptDao.queryAll();      }  } |

**1.4.7 创建controller层**



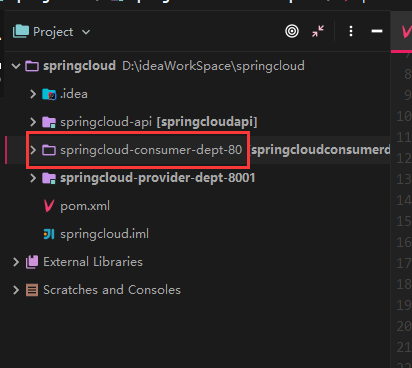
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package com.yao.springcloud.controller;    import com.yao.springcloud.pojo.Dept;  import com.yao.springcloud.service.DeptService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RestController;    import java.util.List;    //提供Restful服务！  @RestController  public class DeptController {      @Autowired      private DeptService deptService;        @PostMapping("/dept/add")      public boolean addDept(Dept dept){          return deptService.addDept(dept);      }        @GetMapping("/dept/get/{id}")      public Dept get(@PathVariable("id") Long id){          return deptService.queryById(id);      }        @GetMapping("/dept/list")      public List<Dept> queryAll(){          return deptService.queryAll();      }        } |

**1.4.8 创建启动类**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package com.yao.springcloud;    import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;    //启动类  @SpringBootApplication  public class DeptProvider\_8001 {      public static void main(String[] args) {          SpringApplication.run(DeptProvider\_8001.class,args);      }  } |

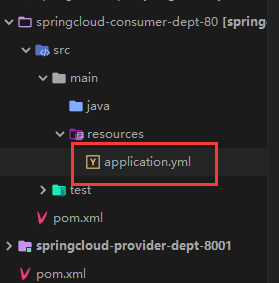
**1.5 写第三个模块（消费者模块）**



**1.5.1 配置pom.xml**

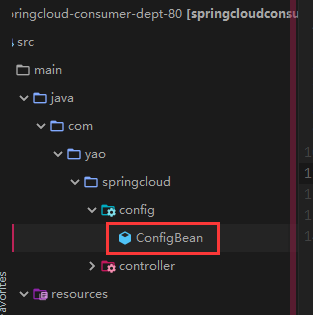
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <parent>          <artifactId>springcloud</artifactId>          <groupId>com.yao</groupId>          <version>1.0-SNAPSHOT</version>      </parent>      <modelVersion>4.0.0</modelVersion>        <artifactId>springcloud-consumer-dept-80</artifactId>        <!--实体类+web-->      <dependencies>          <dependency>              <groupId>com.yao</groupId>              <artifactId>springcloud-api</artifactId>              <version>1.0-SNAPSHOT</version>          </dependency>          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-web</artifactId>          </dependency>          <dependency>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-devtools</artifactId>          </dependency>      </dependencies>  </project> |

**1.5.2 配置application.yml**



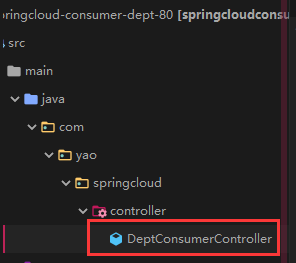
|  |  |
| --- | --- |
| 1  2 | server:    port: 80 |

**1.5.3 创建config**



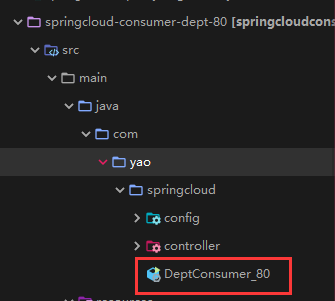
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.yao.springcloud.config;    import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import org.springframework.web.client.RestTemplate;    @Configuration  public class ConfigBean {      @Bean      public RestTemplate getRestTemplate(){          return new RestTemplate();      }  } |

**1.5.4 创建并编写controller**



package com.yao.springcloud.controller;  
  
import com.yao.springcloud.pojo.Dept;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
  
import java.util.List;  
  
@RestController  
public class DeptConsumerController {  
  
 // 理解：消费者没有service层  
 // RestTemplate .... 供我们直接调用就可以了,但是原本的它并没有  
 // 注册到容器里去，所以我们需要刚刚那个config讲它注入  
 //(url,实体：Map,Class<T> responseType)  
 @Autowired  
 private RestTemplate restTemplate;//提供多种便捷访问远程http服务的方法，简单的Restful服务模板  
  
 private static final String REST\_URL\_PREFIX="http://localhost:8001";  
  
 @RequestMapping("/consumer/dept/add")  
 public boolean add(Dept dept){  
 return restTemplate.postForObject(REST\_URL\_PREFIX+"/dept/add",dept,boolean.class);  
 }  
  
 @RequestMapping("/consumer/dept/list")  
 public List queryAll(){  
 return restTemplate.getForObject(REST\_URL\_PREFIX+"/dept/list",List.class);  
 }  
  
 @RequestMapping("/consumer/dept/get/{id}")  
 public Dept get(@PathVariable("id") Long id){  
 return restTemplate.getForObject(REST\_URL\_PREFIX+"/dept/get/"+id,Dept.class);  
 }  
  
}

**1.5.5 编写启动类**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package com.yao.springcloud;    import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;    @SpringBootApplication  public class DeptConsumer\_80 {      public static void main(String[] args) {          SpringApplication.run(DeptConsumer\_80.class,args);      }  } |

**1.6 测试**

先要启动provider也就是服务端8001端口，再启动消费者80端口的那个

