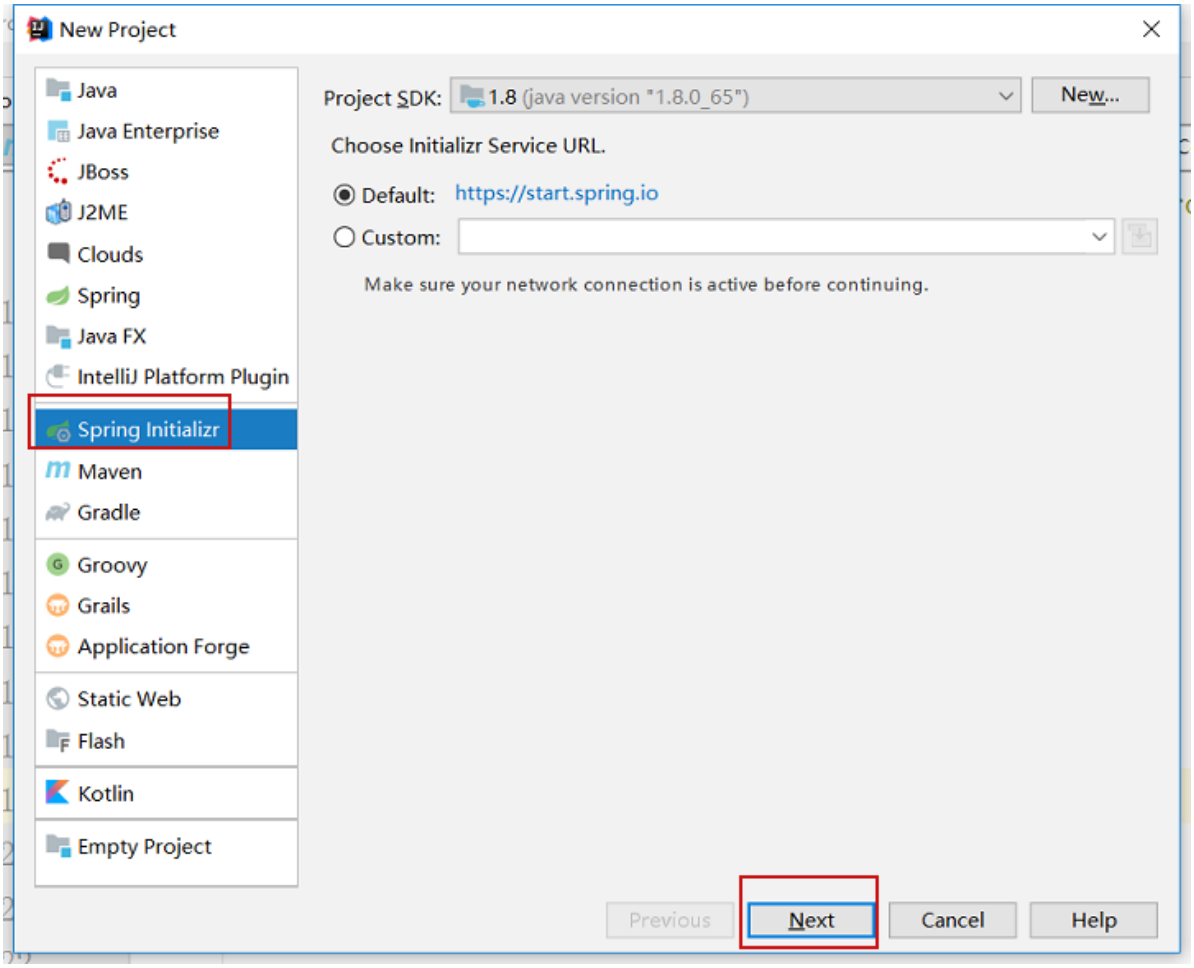
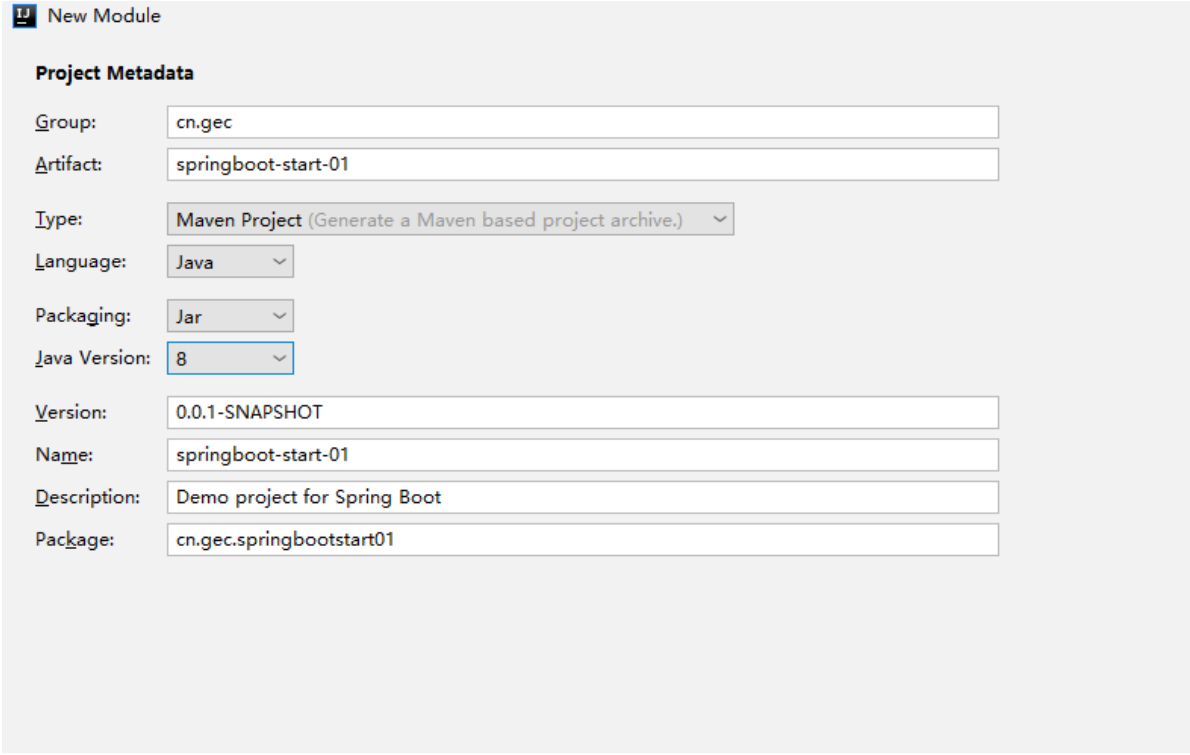
# 第一个springboot程序

## 配置入门

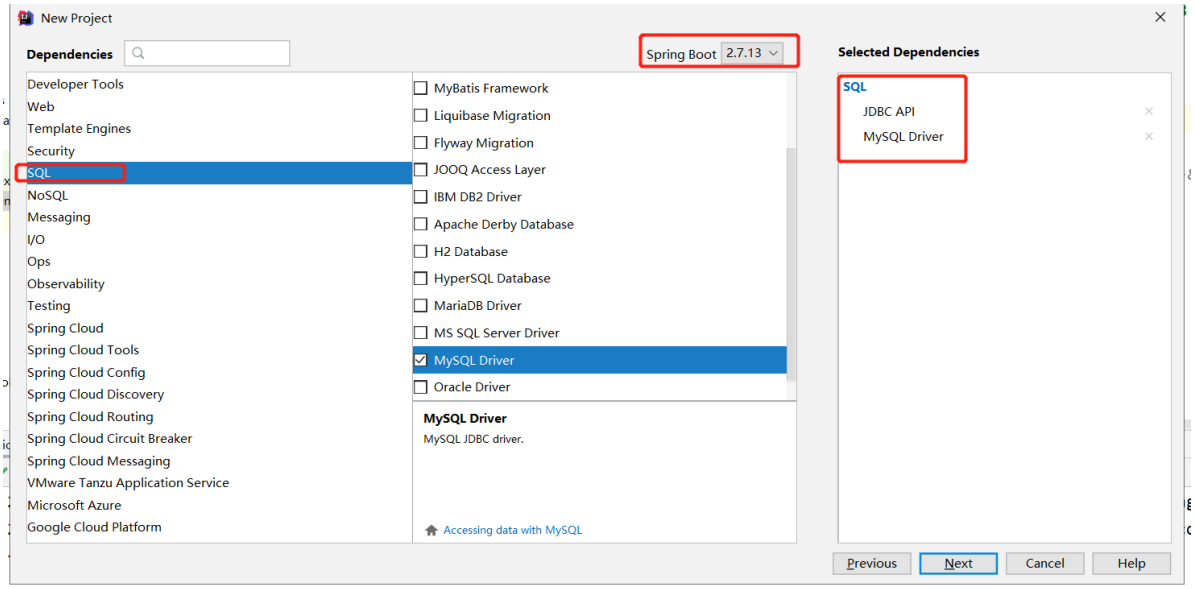
### 新建一个项目




### 填写创建项目的信息，注意使用Maven的方式创建项目，以jar方式打包




## 配置依赖pom.xml文件（其实就是从官网获取到依赖的pom配置）




> 注意：
>
> springboot的版本设置为2.7.13

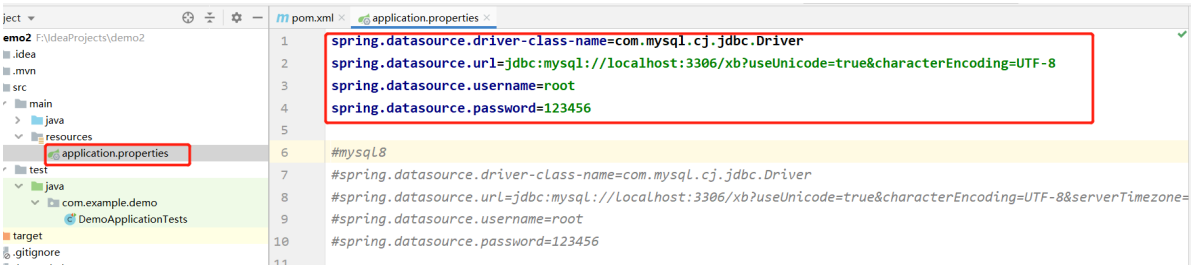
## 配置数据库连接参数

打开application.properties,加入数据库连接参数:

```
#mysql5
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/xb?
useUnicode=true&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=123456

#mysql8
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.datasource.url=jdbc:mysql://localhost:3306/xb?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Hongkong
#spring.datasource.username=root
#spring.datasource.password=123456

```




## 测试

在测试类中加入如下代码:

```
@Autowired
    DataSource dataSource;

    @Test
    void contextLoads() throws SQLException {
        System.out.println(dataSource.getConnection().getClientInfo());

    }
```


> 注意，项目不同，测试类名字可能不同，不用改名字

# JdbcTemplate

## JdbcTemplate简介

JdbcTemplate是对数据库操作的封装，让我们操作数据库更加方便，功能更加强大。能帮我们实现对数据库的增、删、改、查、调用存储过程、存储函数等。对结果集自动封装、预编译sql、动态传值等

创建表语句：

```sql
create table test(id int primary key auto_increment, name varchar(30));
CREATE TABLE `user`  (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `username` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '用户名',
  `password` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '密码',
  `email` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '邮箱',
  `qq_openid` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT 'QQ登录标识符',
  `wx_openid` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '微信登录标识符',
  `real_name` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '真实姓名',
  `age` int(11) NULL DEFAULT NULL COMMENT '年龄',
  `phone` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '手机号',
  `gender` varchar(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '性别 1:男 0:女',
  `desc` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '简介',
  `register_time` datetime(0) NULL DEFAULT NULL COMMENT '注册时间',
  `login_time` datetime(0) NULL DEFAULT NULL COMMENT '上次登录时间',
  `pic` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '头像',
  `look` int(11) NULL DEFAULT NULL COMMENT '查看数',
  `is_secret` varchar(1) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '是否私密 0：私密 1：公开',
  `dept_name` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '部门名称',
  `dept_id` int(11) NULL DEFAULT NULL COMMENT '部门id',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 22 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Compact;

-- ----------------------------
-- Records of user
-- ----------------------------
INSERT INTO `user` VALUES (1, 'admin', 'admin', 'zijun1024@aliyun.com', NULL, NULL, '5', 20, '110', NULL, NULL, '2019-12-01 10:51:50', '2020-01-03 00:00:00', 'http://localhost:8080/upload/def.png', 48, NULL, '', 5);
```

```
INSERT INTO `user` VALUES (2, 'xiaodong', 'admin', 'xiaobiao@dfbz.com', NULL, NULL, '小东', 18, '110', '1', '我很帅啊', '2019-12-02 12:35:10', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 32, '1', '研发部', 1);
INSERT INTO `user` VALUES (3, 'xiaofang', 'admin', 'xiaofang@dfbz.com', NULL, NULL, '小方', 18, '110', '1', '我很帅啊', '2019-12-02 12:35:10', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 29, '1', '研发部', 1);
INSERT INTO `user` VALUES (4, 'xiaobiao', 'admin', 'xiaobiao@dfbz.com', NULL, NULL, '小标', 18, '110', '1', '我很帅啊', '2019-12-02 12:35:10', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 27, '1', '研发部', 1);
INSERT INTO `user` VALUES (5, 'xiaozhun', 'admin', 'xiaozhun@dfbz.com', NULL, NULL, '小准', 18, '110', '1', '我很帅啊', '2019-12-02 12:35:10', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 31, '1', '研发部', 1);
INSERT INTO `user` VALUES (6, 'dfbz', 'admin', 'dfbz@dfbz.com', NULL, NULL, '东方标准', 18, '110', '1', '我很帅啊', '2019-11-28 11:30:24', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 20, '1', '研发部', 1);
INSERT INTO `user` VALUES (7, 'xiaoming', 'admin', 'xm@dfbz.com', NULL, NULL, '小明', 18, '110', '1', '我很帅啊', '2019-12-04 07:30:28', '2019-11-20 10:51:53', 'http://localhost:8080/upload/def.png', 21, '1', '研发部', 1);
INSERT INTO `user` VALUES (8, 'root', 'admin', 'root@dfbz.com', NULL, NULL, '管理员', 28, '110', '0', NULL, '2019-12-06 12:33:41', '2019-12-06 00:00:00', 'http://localhost:8080/upload/22892836-ed70-4039-8558-69dc81dd676b.png', 3, '0', '', 3);
INSERT INTO `user` VALUES (9, 'root_1', 'admin', 'root_1@dfbz.com', NULL, NULL, '管理员1号', 18, '119', '0', NULL, '2019-12-06 12:37:29', '2019-12-06 00:00:00', 'http://localhost:8080/upload/def.png', 0, '0', NULL, NULL);
INSERT INTO `user` VALUES (17, 'cc-475fd36f42ef', NULL, NULL, '3584F8F99BE67A5254A122D123B2BE24', NULL, 'Cool', 3332, '111', NULL, NULL, '2019-12-27 12:07:54', '2019-12-27 00:00:00', 'http://localhost:8080/upload/f1a3139d-d51a-4376-9ad7-90a4f2aa28de.png', 2, NULL, '秘书部', 7);
INSERT INTO `user` VALUES (18, 'af-845c0bfa13d5', NULL, NULL, '80303D26E1091C58F41FFAB08123F427', NULL, 'qq', 11, '110', NULL, NULL, '2019-12-27 15:30:53', '2020-01-03 00:00:00', 'http://thirdqq.qlogo.cn/g?b=oidb&k=7ywAfU4U8EOKpibrQ2UXibkw&s=100&t=1556885708', 2, NULL, '', 4);
INSERT INTO `user` VALUES (19, 'a9-4b9ef510e8a9', NULL, NULL, NULL, 'oQ4QeuPW7WL6ZwKt4Dc9wlc8WjRY', 'For', NULL, NULL, '1', NULL, '2019-12-31 10:49:37', '2020-01-03 00:00:00', 'http://thirdwx.qlogo.cn/mmopen/vi_32/icxlG7XRNqvQ4gUhgicXlzOoH4icR2E2MH8TOqic2X2KdbIiaKGlp9FQ91qnHjr2KW2uBBOZyfia7WOasvgIjgh3yK4A/132', 0, '0', NULL, NULL);
INSERT INTO `user` VALUES (20, '7f-3ea444301d8d', NULL, NULL, '01CC173B24CFE00ABD613E6AB58CED54', NULL, '嘿嘿嘿', NULL, NULL, NULL, NULL, '2019-12-31 16:45:47', '2019-12-31 00:00:00', 'http://thirdqq.qlogo.cn/g?b=oidb&k=IMauKBZYyia9kUL43DEiaMOQ&s=100&t=1557174653', 0, '0', NULL, NULL);
INSERT INTO `user` VALUES (21, '8a-6e107a887a55', NULL, NULL, NULL, 'oQ4QeuNRR9S_4IyRzkJbNRoPNSH4', '嘿嘿嘿', 1, '234234234', NULL, NULL, '2019-12-31 16:46:20', '2019-12-31 00:00:00', 'http://localhost:8080/upload/a3fb2503-b1b1-46e1-854a-8051c7368b41.jpg', 1, '1', '财务部', 4);
```

# JdbcTemplate的DML

API介绍：

| 方法 | 说明 |
| --- | --- |
| **public int update(final String sql, Object...args)** | 作用：实现增删改操作<br>参数：<br>1）SQL语句<br>2）替换占位符值<br>返回值：影响的行数 |

```
//创建数据库操作模板对象
@Autowired
    JdbcTemplate jdbcTemplate;

// 新增
@Test
public void test1() throws Exception{

    jdbcTemplate.update("insert into test values(null,'张三');");

}

// 修改
@Test
public void test2() throws Exception{

    // 传递占位符
    jdbcTemplate.update("update user set username=? where id=?","李四",1);

}

// 删除
@Test
public void test3() throws Exception{

    jdbcTemplate.update("delete from user where id=?",1);

}
```

# JdbcTemplate 的DQL

## 查询多条结果集返回`List<T>`

API介绍：

| 方法 | 说明 |
| --- | --- |
| **List query(String sql, BeanPropertyRowMapper rowMapper, Object... args)** | 作用： 查询多条记录封装成List类型的对象<br>参数：<br>1) SQL语句<br>2) 接口：用于将一条记录封装(映射)成实体类对象<br>3) 占位符替换值<br>返回值： 一个封装好数据的对象 |
| **T query(String sql, ResultSetExtractor rse, @Nullable Object... args)** | 作用： 查询多条记录封装成类型的对象<br>参数：<br>1) SQL语句<br>2) 接口：用于将多条记录映射成指定类型的对象<br>3) 占位符替换值<br>返回值： 一个封装好数据的对象 |

查询全部结果集

**BeanPropertyRowMapper 自动映射**

```
@Test
public void test3() throws Exception {

    // 使用BeanPropertyRowMapper 自动映射
    List<User> userList = jdbcTemplate.query("select * from user", new
BeanPropertyRowMapper<>(User.class));

    for (User user : userList) {
        System.out.println(user);
    }
}
```

自动映射必须实体属性名和数据库列名一致或者实体属性驼峰和数据库列名下划线匹配。

BeanPropertyRowMapper 自动映射图：

自动映射规则:
1、实体类属性名和数据库结果集列名一致可实现自动映射
2、实体类驼峰命名和数据库结果集列名下划线匹配也可以实现自动映射

```
public class User {

    private Long id;
    private String username;
    private String password;
    private String email;
    private String qqOpenid;
    private String wxOpenid;
    private String realName;
    private Long age;
    private String phone;
    private String gender;
    private String desc;
    private Date registerTime;
    private Date loginTime;
    private String pic;
    private Long look;
    private String isSecret;
    private String deptName;
    private Long deptId;
```



| 键 | 值 |
| --- | --- |
| id | 1 |
| username | xiaodong |
| password | admin |
| email | xiaodong@dfbz.com |
| register_time | 2019-10-20 |
| login_time | 2019-10-20 |
| real_name | 小东 |
| dept_id | 1 |

## 查询单条记录，返回T

API:

| 方法 | 说明 |
| --- | --- |
| **T queryForObject(String sql, Class requiredType)** | 作用： 查询单条记录返回指定类型<br>参数:<br>1) SQL语句<br>2) 要查询的结果类型<br>返回值： T |
| **T queryForObject(String sql, BeanPropertyRowMapper rowMapper, Object... args)** | 作用： 查询单条记录返回指定类型<br>参数:<br>1) SQL语句<br>2) BeanPropertyRowMapper映射接口，用于映射结果集与返回类型的关系<br>返回值： T |

```
1  @Test
2  public void test3() throws Exception {
3
4      // 指定返回类型为Long类型
5      Long count = jdbcTemplate.queryForObject("select count(1) from
   user",Long.class );
6      System.out.println(count);
7  }
8
9  @Test
10 public void test4() throws Exception {
11
12     // 指定返回类型为User类型,并使用BeanPropertyRowMapper完成自动映射
13     User user = jdbcTemplate.queryForObject(
14             "select * from user where id =?",
```

```
            new BeanPropertyRowMapper<>(User.class),
            "1");
    System.out.println(user);
}
```

注意：**queryForObject只能查询单条数据，不能是0条（没查询到），也不能是多条。**

如果没有查询到结果集，则抛出：`org.springframework.dao.EmptyResultDataAccessException`异常

如果查询到多条结果集，则抛出：`org.springframework.dao.IncorrectResultSizeDataAccessException`异常

# 小结

## DML

使用update方法执行原生sql语句，一般用于执行DML语句，返回受影响的行数

## DQL

- query：用于查询多条结果集，结果集类型由用户自己指定
  - BeanPropertyRowMapper映射：传入要映射成实体的字节码对象，实现自动映射
    - 规则1：实体类属性名和查询结果集的列名一致
    - 规则2：实体类属性驼峰命名匹配查询结果集的列名
- queryForList：用于查询多条结果集，结果集类型为Map，数据库结果集的列名就是Map的键
- queryForObject：用于查询单挑结果集，结果集类型为T，可以使用RowMapper进行关系映射。不能查询到多条，也不能查询0条