【实验】数据库技术

【实验目的】

掌握使用Spring Boot连接MySQL数据库的数据库技术，通过使用对应的数据库技术实现数据查询、插入数据、更新数据和删除数据等基本数据操作

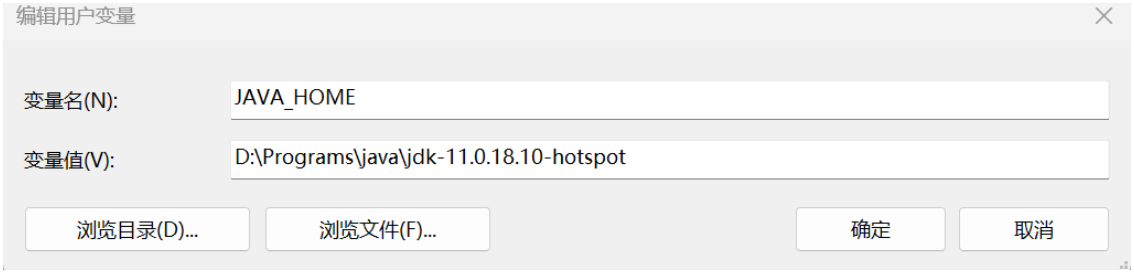
【预备知识】

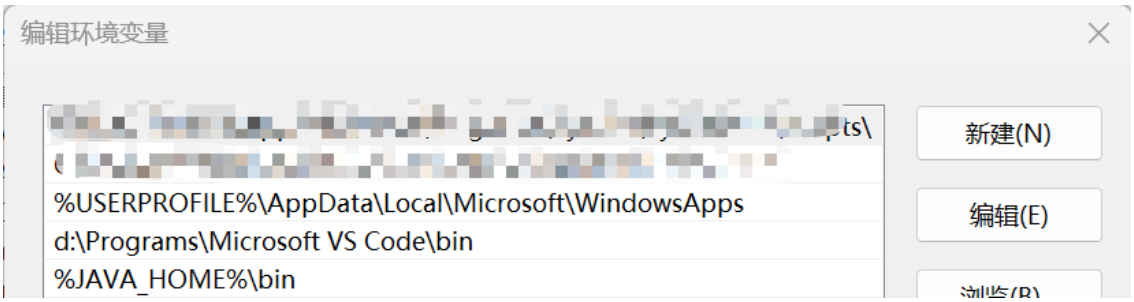
1. MySQL数据库的安装和配置；
2. SQL语言基础知识；

【实验内容】

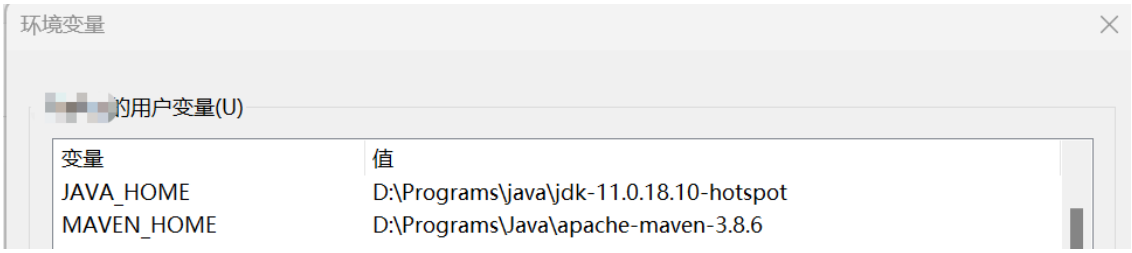
- ### 实验1：环境搭建和参数配置

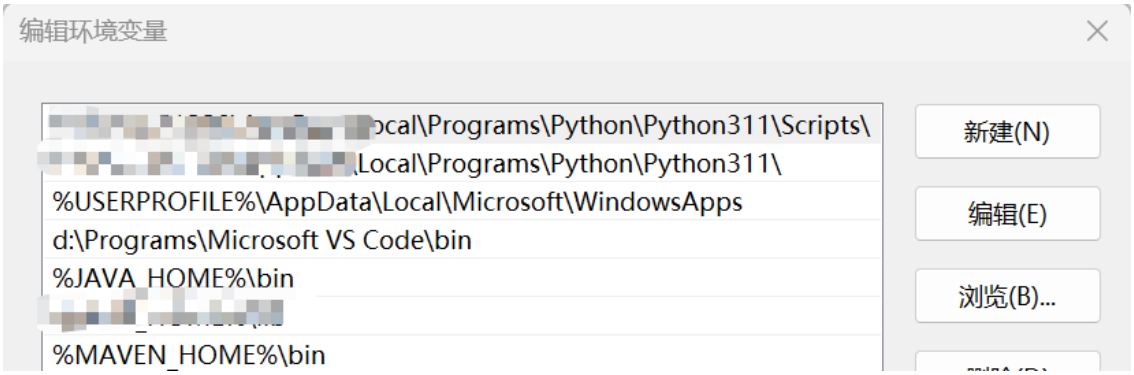
  - JDK的Windows环境配置





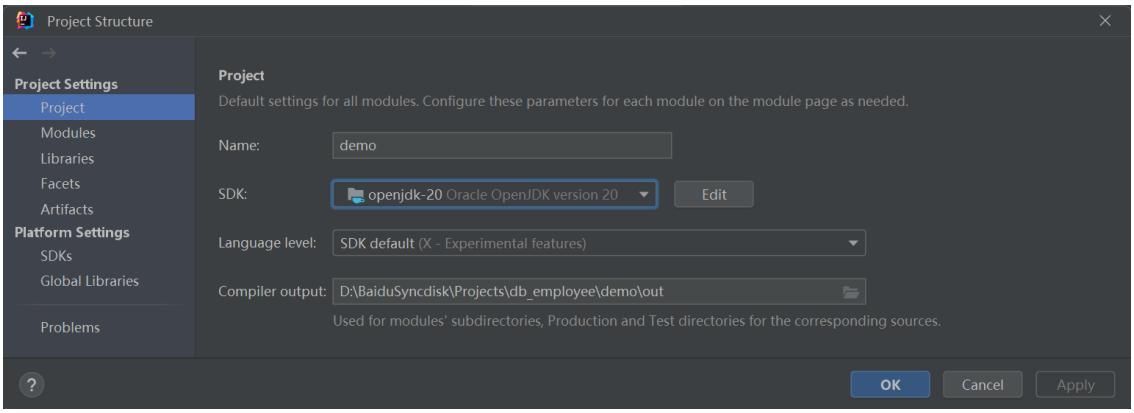
  - Maven的Windows环境配置





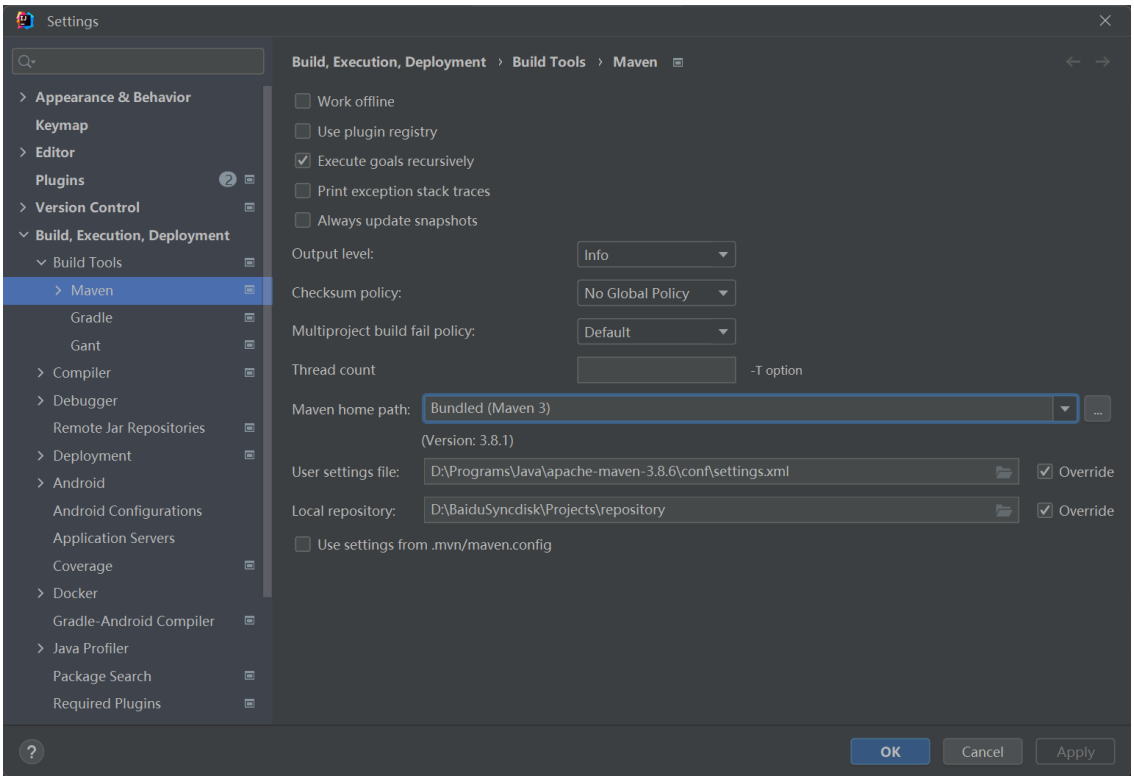
- `IntelliJ IDEA`的`JDK`配置

菜单`File -- Project Structure`

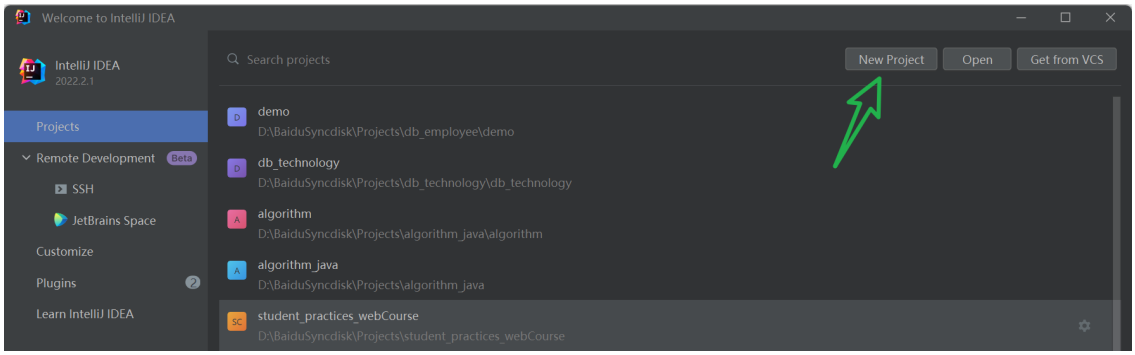


- `IntelliJ IDEA`的`MAVEN`配置

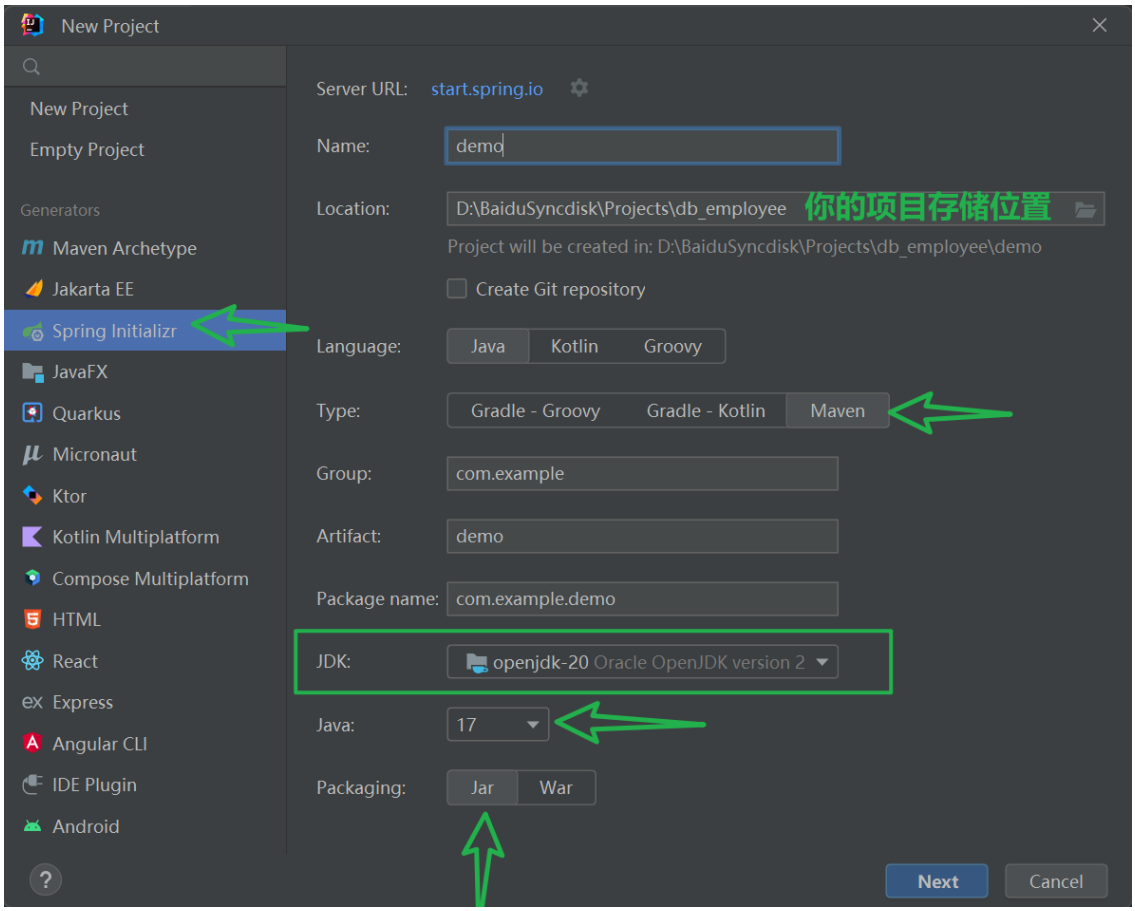
菜单`File -- Settings`



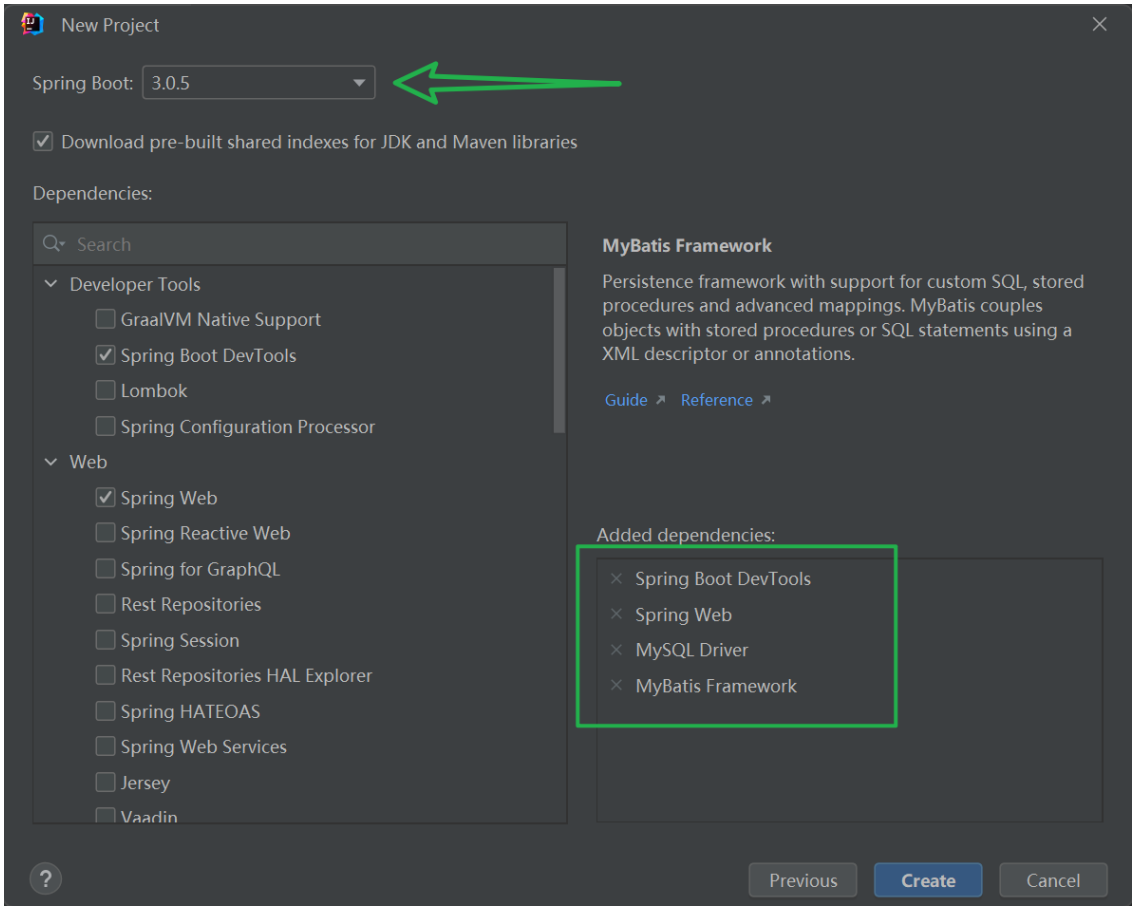
- **实验2：创建`Spring Boot`项目**
  - 新建项目



  - 选择创建Spring Boot项目



  - 选择项目插件



- **实验3：项目配置**
  - 检查`pom.xml`文件，是否引入了相关依赖包

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>3.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
```

```
        </dependency>
    </dependencies>
```

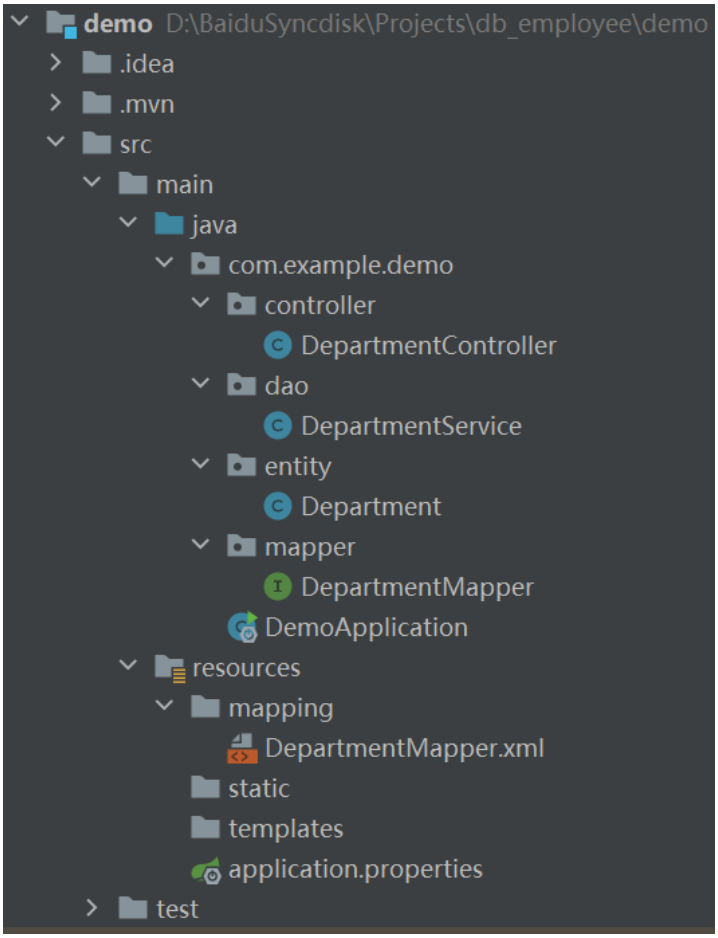
- 修改`application.properties`全局配置文件（也可以使用`application.yml`简化代码实现）

```
server.port=8081

#MySQL配置
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/db_employee?
characterEncoding=UTF-8&userSSL=true&serverTimezone=UTC
spring.datasource.username=admin
spring.datasource.password=123456

#Mybatis配置
mybatis.mapper-locations=classpath:mapping/*Mapper.xml
mybatis.type-aliases-package=com.example.entity.demo

logging.level.com.example.demo.mapper=debug
```

- **实验4：创建实验表结构Department**
  - 在`Navicat`中创建db_employee数据库，并创建表department

```
use db_employee;

drop table if exists department;
create table department(
 id int primary key auto_increment comment'部门编号',
 name varchar(100) not null comment'部门名称'
);
```

- **实验5：项目实现**
  - 项目目录结构，如下图：



- 创建实体类Department

```
package com.example.demo.entity;

public class Department {
    public Department(){}
    public Department(Department entity){
        this.id=entity.id;
        this.name=entity.name;
    }
    // 此处省略get、set方法实现
    @Override
    public String toString() {
        return "Department{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

- 设置映射器：DepartmentMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.demo.mapper.DepartmentMapper">
    <resultMap id="BaseResultMap" type="com.example.demo.entity.Department">
        <result column="id" jdbcType="INTEGER" property="id"/>
        <result column="name" jdbcType="VARCHAR" property="name"/>
```

```
    </resultMap>

    <select id="Sel" resultType="com.example.demo.entity.Department">
        select * from department
        <if test="department.id != null">
            where id = #{department.id}
        </if>
    </select>

    <insert id="Add" parameterType="com.example.demo.entity.Department">
        INSERT INTO department
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="department.name != null">
                name
            </if>
        </trim>
        <trim prefix="VALUES (" suffix=")" suffixOverrides=",">
            <if test="department.name != null">
                #{department.name,jdbcType=VARCHAR}
            </if>
        </trim>
    </insert>

    <!--此处省略更新、删除映射器代码，请模仿进行实现-->

</mapper>
```

MyBatis中，将DAO层的接口与相应的Mapper.xml配置文件进行组合使用，将xxxMapper.xml与xxxMapper.java形成映射关系。resultMap标签是映射输出结果实现

在`application.properties`全局配置文件中，针对配置Mapper.xml的扫描进行了具体实现

```
<!-- 查询多个参数的时候 ,map 的形式  parameterType='map' map为系统定义好的别名-->
    <select id="findByNameAndDept" parameterType="map" resultType="user">
        select * from user where name like concat('%',#{name},'%') and
dept_id=#{id}
    </select>
```

- 映射关系DepartmentMapper.java

```
package com.example.demo.mapper;

import com.example.demo.entity.Department;
import org.apache.ibatis.annotations.*;
import org.springframework.stereotype.Repository;

@Repository
public interface DepartmentMapper {
    Department Sel(@Param("department")Department department);

    int Add(@Param("department")Department department);

    // 此处省略Update，Delete代码，模仿前面编码尝试实现
}
```

- 接口文件DepartmentService.java

```
package com.example.demo.dao;

import com.example.demo.entity.Department;
import com.example.demo.mapper.DepartmentMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class DepartmentService {
    @Autowired
    private DepartmentMapper departmentMapper;

    public Department Sel(Department department) {
        return departmentMapper.Sel(department);
    }

    public String Add(Department department) {
        int a = departmentMapper.Add(department);
        if (a == 1) {
            return "添加成功";
        } else {
            return "添加失败";
        }
    }
// 此处省略Update，Delete编码，请模仿完成
```

- 控制器DepartmentController.java

```
package com.example.demo.controller;

import com.example.demo.dao.DepartmentService;
import com.example.demo.entity.Department;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DepartmentController {
    @Autowired
    private DepartmentService departmentService;

    @RequestMapping(value = "/selectDepartmentById", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
    @ResponseBody
    public String GetDepartment(Department department){
```

```
        return departmentService.Sel(department).toString();
    }

    @RequestMapping(value = "/add", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
    public String Add(Department department){
        return departmentService.Add(department);
    }

    //此处省略Update，Delete，请尝试模仿实现
```

- 在主程序中添加扫描mapper

```
package com.example.demo;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan("com.example.demo.mapper")
public class DemoApplication {

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

• 实验6：项目测试

运行项目，并通过以下URI完成数据新增

http://localhost:8081/add?name=电子信息工程学院

获取数据：

http://localhost:8081/selectDepartmentById?id=2

请完成修改（Update），删除（Delete)的测试。