学生院别：计算机与信息工程学院

学生专业：计算机科学与技术

学 号： 1606915084

课程设计成绩：



# JavaWeb 课程设计报告

## 开课学院： 计算机与信息工程学院

## 课程设计地点： 7#504

## 指导老师：

## 学生姓名： 蒋家聪

## 专业班级： 计科1班

　　　　　2018 —— 2019 学年第 2 学期

# JavaWeb 课程设计任务书

## 题目

## 用户信息后台管理系统

## 目的要求

**实现功能要求**：

1. 登录
2. 删除学生信息功能
3. 修改学生信息功能
4. 增加学生信息功能
5. 查找学生信息功能

## 主要内容及技术要求

访问风格：restful

开发工具：IDEA

依赖环境：JDK8,Docker,mysql5.6,chrome

技术：Springboot,Maven,thylemeaf,Bootstrap,Mybatis

## 需求分析

**1.1 功能需求**

常用的后台管理系统需要有如下功能：

１．登录

２．删除学生信息功能

３．修改学生信息功能

４．增加学生信息功能

５．查找学生信息功能

用户可以通过点击按钮或者链接进行不同的操作，用户有登录以及增删改查的操作，如图１.1所示：

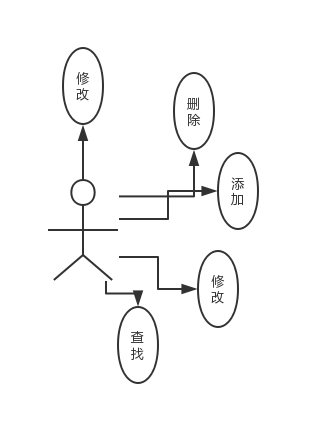
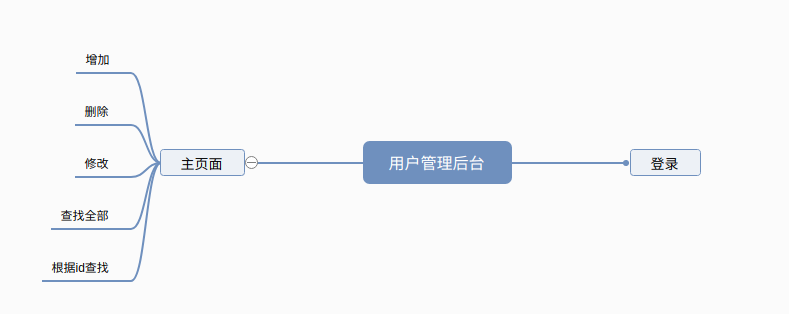


图１.1功能用例图

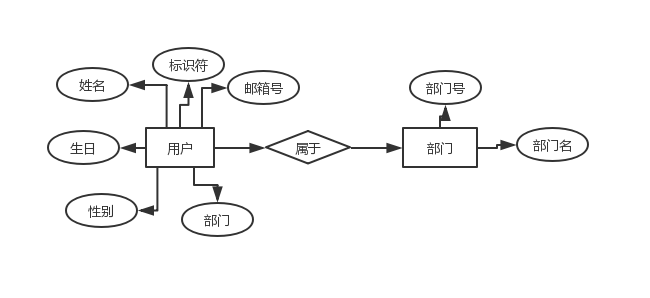
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例名称 | 参与者 | 目标 | 前置条件 | 事件流 |
| 登录 | 用户 | 实现登录以及拦截功能 | 服务后台启用 | 点击“登录”按钮 |
| 增加 | 用户 | 增加用户信息 | 服务后台启用 | 填写表单，点击“增加”按钮 |
| 删除 | 用户 | 根据ｉｄ删除用户信息 | 服务后台启用 | 点击“删除”按钮 |
| 修改 | 用户 | 修改指定用户信息 | 服务后台启用 | 点击“编辑”按钮 |
| 查询 | 用户 | 查询用户信息 | 服务后台运行 | 跳转主页时显示 |

**１.2 模块划分思维导图**

****

**图１.2　模块划分思维导图**

**1.3 数据库中ＥＲ图如图１.3所示**

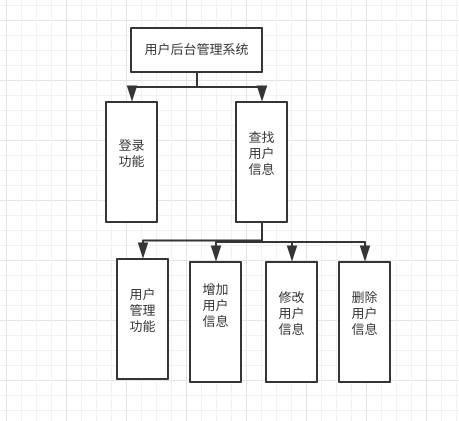
****

**图１.３　数据库用例ＥＲ图**

## 系统设计

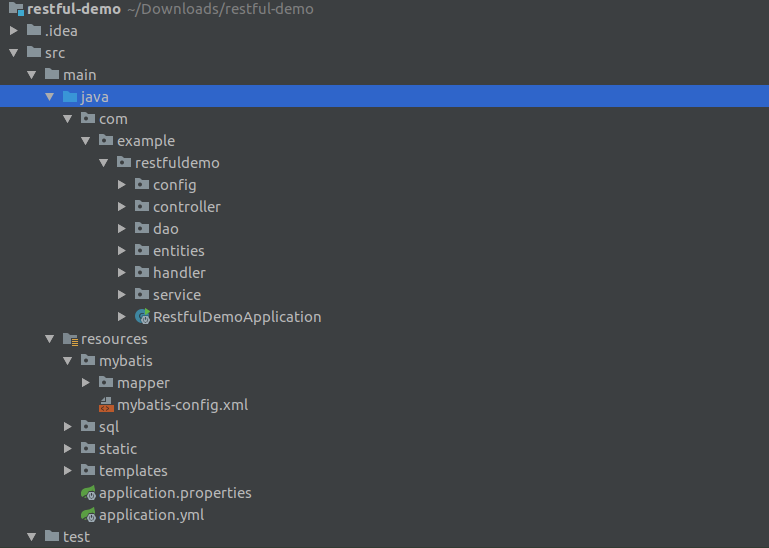
### 2.1总体设计

根据用户后台管理要求，我们要实现登录，增删改查可分为两大模块部分，如图2.1所示：



**图2.1功能结构图**

**2.2 包结构设计**



1. **部署：**

**３.1环境配置：**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.1.1启动docker:

su root

service docker start

netstat -tanlp

kill 1386

3.1.2创建mysql的docker

docker run -p 3306:3306 --name jdbc -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:5.6 --character-set-server=utf8mb4 --collation-server=utf8mb4\_unicode\_ci

3.1.3进入该docker

docker exec -it dfe1291fb0c7 env LANG=C.UTF-8 /bin/bash

mysql -uroot -p

3.1.4创建database:students

3.1.5执行数据库初始化表结构：执行department.sql

3.1.6执行数据库初始化数据：执行dataDo.sql

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3.2启动Springboot**

**3.3访问指定位置**

登录界面：<http://localhost:8451/crud/index>

用户操作界面：<http://localhost:8451/crud/emps>

添加用户页面：<http://localhost:8451/crud/emp>

编辑用户页面：<http://localhost:8451/crud/emp/{id}>

源码我已经放到Github上：

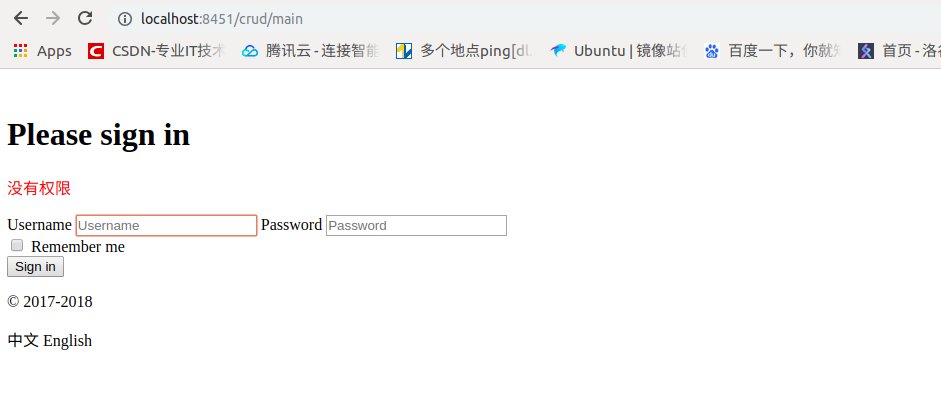
https://github.com/jjc123/Springboot\_restful\_crudDemo

## 效果图

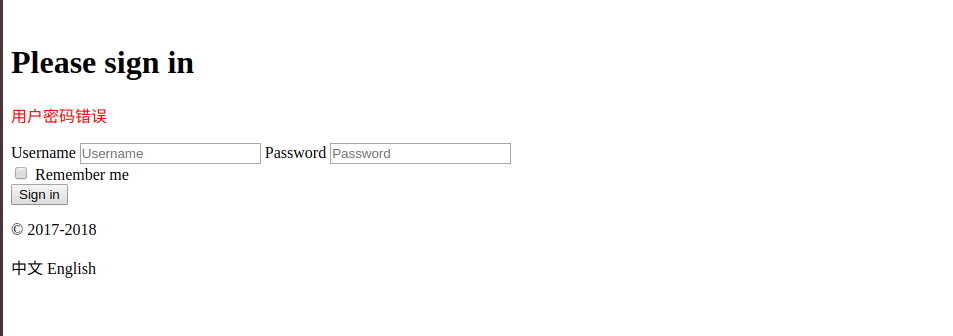
**4.1 登录首页页面：**

## Screenshot from 2019-06-25 19-51-50

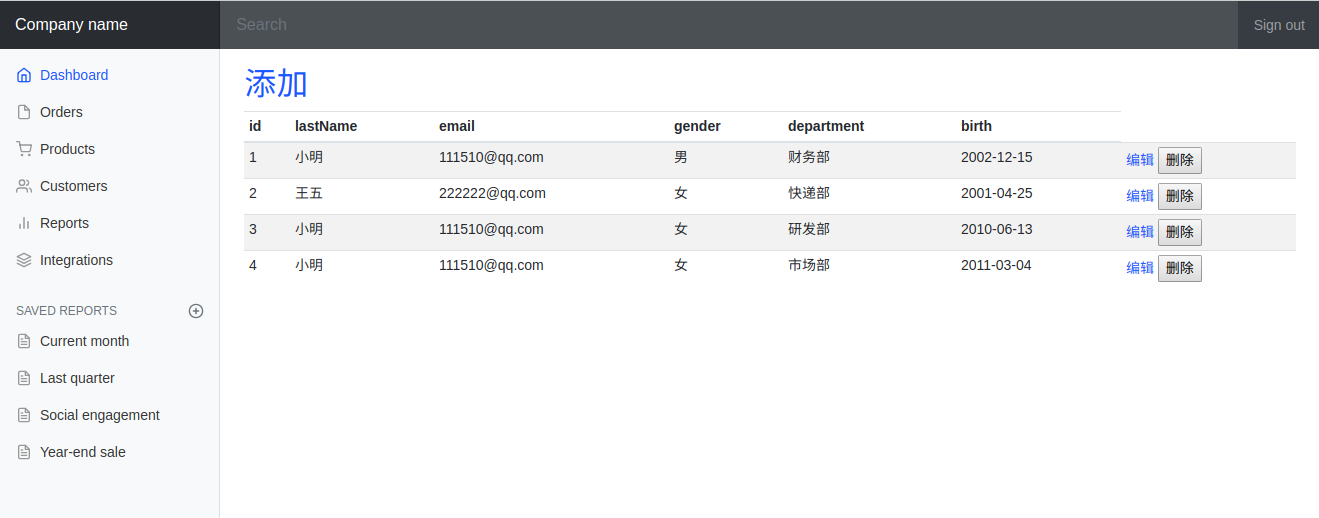
**４.2 没有访问权限跳转登录页面：**



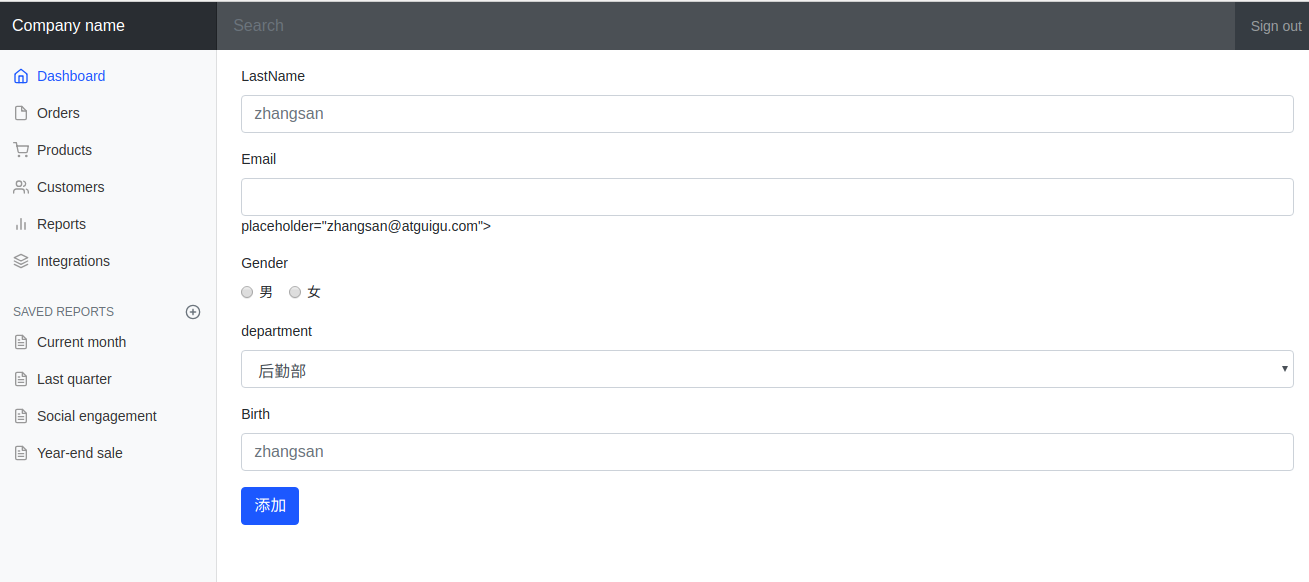
**4.3 账号或密码错误跳转登录页面：**



**4.4 操作主页页面：**



**4.5 添加页面:**



1. **代码实现**

**实体类：**

public class Department {

private Integer id;

private String departmentName;

｝

public class Employee {

private Integer id;

private String lastName;

private String email;

//1 male, 0 female

private Integer gender;

private Department department;

private Date birth;

｝

**登录控制层：**

@Controller

public class Login {

//@RequestMapping(value = "/login",method = RequestMethod.POST)

//@GetMapping

@PostMapping(value = "/login")

public String login(@RequestParam("username") String username, @RequestParam("userpassword") String userpassword,

Map<String,Object> map, HttpSession session){

if(!StringUtils.isEmpty(username) &&"123456".equals(userpassword)){

session.setAttribute("user",username);

return "redirect:/main.html";

}else{

map.put("msg","用户密码错误");

return "index";

}

}

}

**登录拦截器：**

public class LoginHandlerInterceptor implements HandlerInterceptor {

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

Object user = request.getSession().getAttribute("user");

if(user == null){

request.setAttribute("msg","没有权限");

request.getRequestDispatcher("/index.html").forward(request,response);

return false;

}else{

return true;

}

}

@Override

public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {

}

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {

}

}

**Druid配置类注入：**

@Configuration

public class DruidConfig {

@ConfigurationProperties(prefix = "spring.datasource")

@Bean

public DataSource druid(){

return new DruidDataSource();

}

//配置Druid的监控

//1、配置一个管理后台的Servlet

@Bean

public ServletRegistrationBean statViewServlet(){

ServletRegistrationBean bean = new ServletRegistrationBean(new StatViewServlet(), "/druid/\*");

Map<String,String> initParams = new HashMap<>();

initParams.put("loginUsername","admin");

initParams.put("loginPassword","123456");

initParams.put("allow","");//默认就是允许所有访问

initParams.put("deny","192.168.15.21");

bean.setInitParameters(initParams);

return bean;

}

//2、配置一个web监控的filter

@Bean

public FilterRegistrationBean webStatFilter(){

FilterRegistrationBean bean = new FilterRegistrationBean();

bean.setFilter(new WebStatFilter());

Map<String,String> initParams = new HashMap<>();

initParams.put("exclusions","\*.js,\*.css,/druid/\*");

bean.setInitParameters(initParams);

bean.setUrlPatterns(Arrays.asList("/\*"));

return bean;

}

}

**配置资源映射类：**

@Configuration

public class MyConfig implements WebMvcConfigurer {

/\*\*

\* springboot2默认是拦截静态资源的，这里需要给静态资源进行地址映射，

\* 注意如果是"classpath:"会去resources路径下寻找，如果不加classpath无法地址映射

\* @param registry

\*/

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/static/\*\*").addResourceLocations("classpath:/static/");

}

/\*\*

\* 映射控制层地址，这里不需要加classpath

\* @param registry

\*/

@Override

public void addViewControllers(ViewControllerRegistry registry) {

registry.addViewController("/").setViewName("index");

}

@Bean

public WebMvcConfigurer hello(){

WebMvcConfigurer webMvcConfigurer = new WebMvcConfigurerAdapter() {

@Override

public void addViewControllers(ViewControllerRegistry registry) {

registry.addViewController("/index").setViewName("index");

registry.addViewController("/index.html").setViewName("index");

registry.addViewController("/main.html").setViewName("dashboard");

}

};

return webMvcConfigurer;

}

/\*\*

\* 添加拦截器，拦截所有地址，但是过滤指定地址

\* @param registry

\*/

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(new LoginHandlerInterceptor()).addPathPatterns("/\*\*").excludePathPatterns("/","/index"

,"/index.html","/login","/static/\*\*");

}

}

**配置servlet容器类：**

@Configuration

public class MyServletConfig {

/\*\*

\* 配置嵌入式servlet容器并配置

\* @return

\*/

@Bean

public WebServerFactoryCustomizer<ConfigurableWebServerFactory> webServerFactoryCustomizer(){

return new WebServerFactoryCustomizer<ConfigurableWebServerFactory>() {

@Override

public void customize(ConfigurableWebServerFactory factory) {

factory.setPort(8451);

}

};

}

}

**用户增删改查控制类：**

@Controller

public class EmployeeController {

@Autowired

EmployeeService employeeService;

@Autowired

DepartmentService departmentService;

@GetMapping(value = "/emps")

public String getAll(Model model){

Collection<Employee> all = employeeService.getAll();

model.addAttribute("emps",all);

return "emps/list";

}

@GetMapping(value = "/emp")

public String toAdd(Model model){

Collection<Department> departments = departmentService.getDepartments();

model.addAttribute("deps",departments);

return "emps/toadd";

}

@PostMapping(value = "/emp")

public String addEmp(Employee employee){

int count = employeeService.save(employee);

int key = employee.getId();

System.out.println("插入的人员数量:"+count);

System.out.println("插入的人员id:"+key);

//重定向，这里/emps是从上面的控制类进入,而不是资源

return "redirect:/emps";

}

@GetMapping(value = "/emp/{id}")

public String toChange(@PathVariable("id") Integer id,Model model){

Employee employee = employeeService.get(id);

model.addAttribute("emp",employee);

Collection<Department> departments = departmentService.getDepartments();

model.addAttribute("deps",departments);

return "emps/toadd";

}

@PutMapping(value = "/emp")

public String change(Employee employee){

int count = employeeService.update(employee);

System.out.println("修改的人员数量:"+count);

return "redirect:/emps";

}

@DeleteMapping(value = "/emp/{id}")

public String delete(@PathVariable("id") Integer id){

int delete = employeeService.delete(id);

System.out.println("删除的人员数量:"+delete);

return "redirect:/emps";

}

}

**Service层：**

@Service("departmentService")

public class DepartmentServiceImpl implements DepartmentService {

@Autowired

DepartmentDao departmentDao;

@Override

public Collection<Department> getDepartments() {

return departmentDao.getDepartments();

}

}

@Service("employeeService")

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

EmployeeDao employeeDao;

@Override

public Collection<Employee> getAll() {

return employeeDao.getAll();

}

@Override

public int save(Employee employee) {

return employeeDao.save(employee);

}

@Override

public Employee get(Integer id) {

return employeeDao.get(id);

}

@Override

public int delete(Integer id) {

return employeeDao.delete(id);

}

@Override

public int update(Employee employee) {

return employeeDao.update(employee);

}

}

**Dao层接口：**

public interface DepartmentDao {

Collection<Department> getDepartments();

Departmpartment getDepartnt(@Param("id")Integer id);

}

public interface EmployeeDao {

Collection<Employee> getAll();

int save(Employee employee);

Employee get(@Param("id") Integer id);

int delete(@Param("id") Integer id);

int update(Employee employee);

}

**Mapper.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.restfuldemo.dao.DepartmentDao">

<select id="getDepartments" resultType="com.example.restfuldemo.entities.Department">

select \* from department

order by id desc

</select>

<select id="getDepartment" resultType="com.example.restfuldemo.entities.Department">

select \* from department where id =#{id}

</select>

</mapper>

<?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.restfuldemo.dao.EmployeeDao">

<!--

Employee get(@Param("id") Integer id);

int delete(@Param("id") Integer id);

int update(Employee employee);-->

<resultMap type="com.example.restfuldemo.entities.Employee" id="Employee">

<id property="id" column="id"></id>

<!--has-a关系 多对一使用association-->

<association property="department" select="com.example.restfuldemo.dao.DepartmentDao.getDepartment" column="d\_id">

<id property="id"/>

<result column="department\_name" property="departmentName"></result>

</association>

</resultMap>

<select id="getAll" resultMap="Employee">

select \* from employee

order by id

</select>

<select id="get" resultMap="Employee">

select \* from employee where id = #{id}

</select>

<delete id="delete">

delete from employee where id = #{id}

</delete>

<update id="update">

update employee set last\_name=#{lastName},email=#{email},gender=#{gender},d\_id=#{department.id},

birth=#{birth} where id=#{id}

</update>

<!--insert语句默认是不返回记录的主键值，而是返回插入的记录条数； 添加后通过原来的插入类.getId()可获得插入后得到的自增id-->

<insert id="save" useGeneratedKeys="true" keyProperty="id">

insert into employee (last\_name,email,gender,d\_id,birth) values

(#{lastName},#{email},#{gender},#{department.id},#{birth});

</insert>

</mapper>

Mybatis配置xml:

<?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>

<!-- 开启驼峰映射 ，为自定义的SQL语句服务-->

<settings>

<setting name="mapUnderscoreToCamelCase" value="true"/>

</settings>

</configuration>

**初始化表结构：**

SET FOREIGN\_KEY\_CHECKS=0;

-- ----------------------------

-- Table structure for department

-- ----------------------------

DROP TABLE IF EXISTS `department`;

CREATE TABLE `department` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`department\_name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `employee`;

CREATE TABLE `employee` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`last\_name` varchar(255) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL,

`gender` int(11) NOT NULL,

`d\_id` int(11) NOT NULL,

`birth` DATE NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

**初始化表数据：**

insert into department (department\_name) values ('财务部');

insert into department (department\_name) values ('快递部');

insert into department (department\_name) values ('研发部');

insert into department (department\_name) values ('市场部');

insert into department (department\_name) values ('后勤部');

insert into employee (last\_name,email,gender,d\_id,birth) values ('小明','111510@qq.com','1','1','2002-12-15');

insert into employee (last\_name,email,gender,d\_id,birth) values ('王五','222222@qq.com','0','2','2001-4-25');

insert into employee (last\_name,email,gender,d\_id,birth) values ('小明','111510@qq.com','0','3','2010-6-13');

insert into employee (last\_name,email,gender,d\_id,birth) values ('小明','111510@qq.com','0','4','2011-3-4');

insert into employee (last\_name,email,gender,d\_id,birth) values ('小明','111510@qq.com','1','5','2008-4-1');

**配置配置类：**

spring:

datasource:

url: jdbc:mysql://localhost:3306/students

username: root

password: 123456

driver-class-name: com.mysql.cj.jdbc.Driver

type: com.alibaba.druid.pool.DruidDataSource

initialSize: 5

minIdle: 5

maxActive: 20

maxWait: 60000

timeBetweenEvictionRunsMillis: 60000

minEvictableIdleTimeMillis: 300000

validationQuery: SELECT 1 FROM DUAL

testWhileIdle: true

testOnBorrow: false

testOnReturn: false

poolPreparedStatements: true

# 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙

filters: stat,wall,log4j

maxPoolPreparedStatementPerConnectionSize: 20

useGlobalDataSourceStat: true

connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=500

#启动时需要初始化的建表语句

#schema:

#- classpath:sql/department.sql

#data:

#- classpath:sql/dataDo.sql

#initialization-mode: always

#启动时需要初始化的数据

#默认加载schema.sql与data.sql。脚本位置可以通过spring.datasource.schema 与spring.datasource.data 来改变

mybatis:

config-location: classpath:mybatis/mybatis-config.xml

mapper-locations: classpath:mybatis/mapper/\*.xml

## 主要参考资料

<<Java Web整合开发王者归来>>　作者: [刘京华](https://book.douban.com/search/%E5%88%98%E4%BA%AC%E5%8D%8E) 出版社: 清华大学

<<JavaWeb程序开发进阶>> 作者: [传智播客高教产品研发部](https://book.douban.com/search/%E4%BC%A0%E6%99%BA%E6%92%AD%E5%AE%A2%E9%AB%98%E6%95%99%E4%BA%A7%E5%93%81%E7%A0%94%E5%8F%91%E9%83%A8) 出版社: 清华大学出版社

<<Spring实战（第4版）>> 作者: [Craig Walls 沃尔斯](https://book.douban.com/search/Craig Walls %E6%B2%83%E5%B0%94%E6%96%AF) 出版社: 人民邮电出版社

<<Spring Boot实战>> 作者: [[美]克雷格·沃斯](https://book.douban.com/search/%E5%85%8B%E9%9B%B7%E6%A0%BC%C2%B7%E6%B2%83%E6%96%AF)  出版社: 人民邮电出版社

## 结论

项目虽然简单，页面也是从Bootstrap拷贝的模板，但是应用到了多种主流技术，如SpringBoot以及Docker技术．所谓麻雀虽小，五脏俱全，用户的登录功能，以及主要的用户增删改查功能都完好实现.主要的特点如下：

1. 撅弃了JSP技术，改用thylemeaf模板
2. 采用Springboot的组合框架，使用jar包的方式打包项目
3. 采用了Docker容器技术，使得某些配置更加方便，如mysql
4. 实体类Employee中有Department属性类,Mybatis中采用分布查询的方式注入，并且开启了驼峰命名法

项目虽然有点简陋，不过还是有很多可以借鉴的东西的．