

# 《项目开发综合实训》

## 指 导 手 册

(V1.0)

Author: GJQ

2019 年 8 月

## 目录

1. 实训概况.....	2
1.1. 实训目的.....	2
1.2. 实训安排.....	2
1.3. 实训要求.....	2
2. 功能需求与模块设计.....	3
2.1. 功能需求.....	3
2.2. 模块设计.....	3
3. 数据需求与数据库设计.....	3
3.1. 数据需求.....	3
3.2. 数据库设计.....	4
3.2.1. ER 图.....	4
3.2.2. 表设计.....	4
3.2.3. 数据导入.....	6
4. SpringBoot2 开发环境搭建.....	8
4.1. 安装 JDK.....	8
4.2. STS 环境（集成 spring 版本的 eclipse）.....	8
4.2.1. 安装 STS.....	8
4.2.2. 配置 STS.....	9
4.3. 项目测试.....	16
4.3.1. 创建 springboot 项目.....	16
4.3.2. 编写代码.....	18
4.3.3. 运行测试.....	19
5. 项目搭建.....	21
5.1. 创建项目.....	21
5.2. 项目配置.....	24
6. 模块实现.....	29
6.1. 用户管理模块.....	29
6.2. 权限管理模块.....	71
6.3. 角色管理模块.....	72
6.4. 登录控制模块.....	72
7. 项目部署.....	72

# 1. 实训概况

## 1.1. 实训目的

为了进一步巩固 JSP 和 SSH 等 WEB 编程技术，提高项目的综合开发能力，为将来学生做毕业设计或者从事 WEB 开发方面的工作奠定基础，本实训采用较为流行的 Spring Boot2.0 框架结合 JQuery 和 Bootstrap 前端技术来设计一个比较通用的信息管理系统。

## 1.2. 实训安排

实训时间	节次	实训内容	地点	形式
第 1 天	1-4	物联网监控平台功能介绍、数据库设计与项目搭建	6B209	讲授
第 1 天	7-10	用户管理与登录模块实现	6B209	讲授+上机
第 2 天	1-4	设备类型管理模块实现	6B209	上机
第 2 天	7-10	设备管理模块实现	6B209	上机
第 3 天	1-4	项目管理模块实现	6B209	上机
第 3 天	7-10	网关管理模块实现	6B209	上机
第 4 天	1-4	网关设备管理模块实现	6B209	上机
第 4 天	7-10	实训报告撰写	6B209	上机

## 1.3. 实训要求

分组进行实训，每组人数不超过 4 人，基于实训的内容进行扩展与创新，最后以小组为单位提交实训报告和进行现场演示答辩。

## 2. 功能需求与模块设计

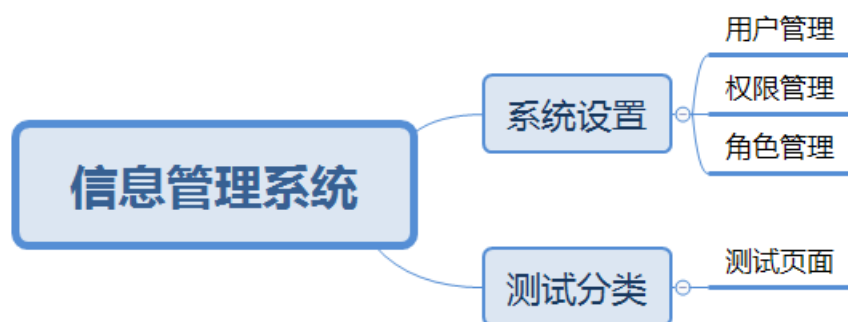
### 2.1. 功能需求

本项目主要实现信息管理系统中的用户与权限管理的通用模块，功能要求如下：

- (1) 用户管理：实现用户的增删改查和用户的角色分配。
- (2) 权限管理：实现权限的增删改查。
- (3) 角色管理：实现角色的增删改查和角色的权限分配。
- (4) 用户登录：实现用户登录验证、动态菜单加载、登录拦截。

### 2.2. 模块设计

系统的总体框架图如下（“测试分类”用来测试，不作为系统功能，画图可以用 xmind、visio、亿图，或者在线的 <https://www.processon.com/>等）：



## 3. 数据需求与数据库设计

### 3.1. 数据需求

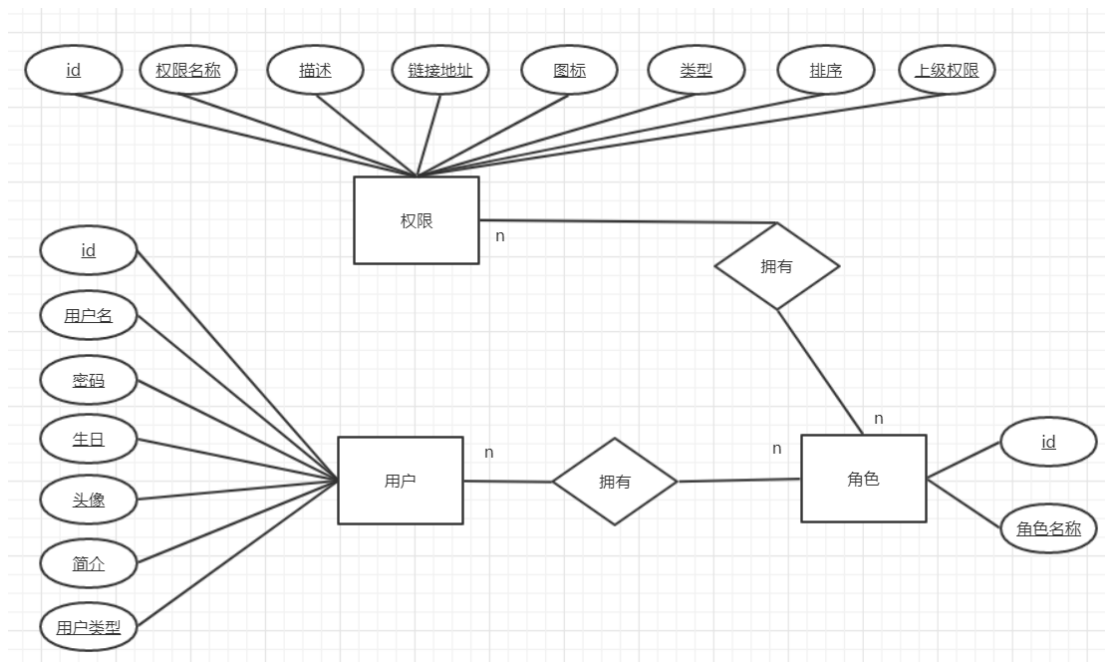
项目中主要包含了用户、权限（菜单）、角色三种类型的数据，各种数据包含的数据项如下：

- (1) 用户：用户名、密码、生日、头像、简介、用户类型
- (2) 权限：权限名称、描述、链接地址、图标、类型、排序、上级权限
- (3) 角色：角色名称

## 3.2. 数据库设计

### 3.2.1. ER 图

数据库总体 er 图如下：



### 3.2.2. 表设计

(1) 用户表：

名	类型	长度	小数点	允许空值 (	
► id	varchar	255	0	<input type="checkbox"/>	🔑 1
username	varchar	255	0	<input checked="" type="checkbox"/>	
password	varchar	255	0	<input checked="" type="checkbox"/>	
birthday	datetime	0	0	<input checked="" type="checkbox"/>	
photo	varchar	255	0	<input checked="" type="checkbox"/>	
introduce	text	0	0	<input checked="" type="checkbox"/>	
usertype	varchar	255	0	<input checked="" type="checkbox"/>	

```

CREATE TABLE `t_sys_user` (
  `id` varchar(255) COLLATE utf8_bin NOT NULL,
  `username` varchar(255) CHARACTER SET utf8 DEFAULT NULL COMMENT '用户账号',
  `password` varchar(255) CHARACTER SET utf8 DEFAULT NULL COMMENT '用户密码',
  `birthday` datetime DEFAULT NULL,
  `photo` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `introduce` text COLLATE utf8_bin,
  `usertype` varchar(255) COLLATE utf8_bin DEFAULT '普通用户' COMMENT '1.普通用户；2.超级管理员（不用授权）',
  PRIMARY KEY (`id`)
)
  
```

```
PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin COMMENT='用户表';
```

(2) 权限表:

名	类型	长度	小数点	允许空值	
▶ id	varchar	255	0	<input type="checkbox"/>	1
name	varchar	255	0	<input checked="" type="checkbox"/>	
description	varchar	255	0	<input checked="" type="checkbox"/>	
url	varchar	255	0	<input checked="" type="checkbox"/>	
pid	varchar	255	0	<input checked="" type="checkbox"/>	
perms	varchar	255	0	<input checked="" type="checkbox"/>	
type	int	11	0	<input checked="" type="checkbox"/>	
icon	varchar	255	0	<input checked="" type="checkbox"/>	
order_num	int	11	0	<input checked="" type="checkbox"/>	

```
CREATE TABLE `t_sys_permission` (
  `id` varchar(255) NOT NULL COMMENT 'id',
  `name` varchar(255) DEFAULT NULL COMMENT '权限名称',
  `description` varchar(255) DEFAULT NULL COMMENT '权限描述',
  `url` varchar(255) DEFAULT NULL COMMENT '授权链接',
  `pid` varchar(255) DEFAULT NULL COMMENT '父节点id',
  `perms` varchar(255) DEFAULT NULL COMMENT '权限标识',
  `type` int(11) DEFAULT '1' COMMENT '类型:1.菜单;2.按钮',
  `icon` varchar(255) DEFAULT NULL COMMENT '菜单图标',
  `order_num` int(11) DEFAULT NULL COMMENT '排序',
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='权限表';
```

(3) 角色表:

名	类型	长度	小数点	允许空值	
▶ id	varchar	255	0	<input type="checkbox"/>	1
name	varchar	255	0	<input checked="" type="checkbox"/>	

```
CREATE TABLE `t_sys_role` (
  `id` varchar(255) NOT NULL COMMENT 'id',
  `name` varchar(255) DEFAULT NULL COMMENT '角色名称',
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='角色表';
```

(3) 用户角色表:

名	类型	长度	小数点	允许空值	
▶ id	varchar	255	0	<input type="checkbox"/>	1
sys_user_id	varchar	255	0	<input checked="" type="checkbox"/>	
sys_role_id	varchar	255	0	<input checked="" type="checkbox"/>	

```
CREATE TABLE `t_sys_role_user` (
  `id` varchar(255) NOT NULL,
  `sys_user_id` varchar(255) DEFAULT NULL COMMENT '用户id',
```

```
`sys_role_id` varchar(255) DEFAULT NULL COMMENT '角色id',
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户角色中间表';
```

(5) 角色权限表:

名	类型	长度	小数点	允许空值 (	
id	varchar	255	0	<input type="checkbox"/>	1
sys_role_id	varchar	255	0	<input checked="" type="checkbox"/>	
sys_permission_id	varchar	255	0	<input checked="" type="checkbox"/>	

```
CREATE TABLE `t_sys_permission_role` (
  `id` varchar(255) NOT NULL,
  `sys_role_id` varchar(255) DEFAULT NULL COMMENT '角色id',
  `sys_permission_id` varchar(255) DEFAULT NULL COMMENT '权限id',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='角色权限中间表';
```

### 3.2.3. 数据导入

#### (1) 安装 MySQL

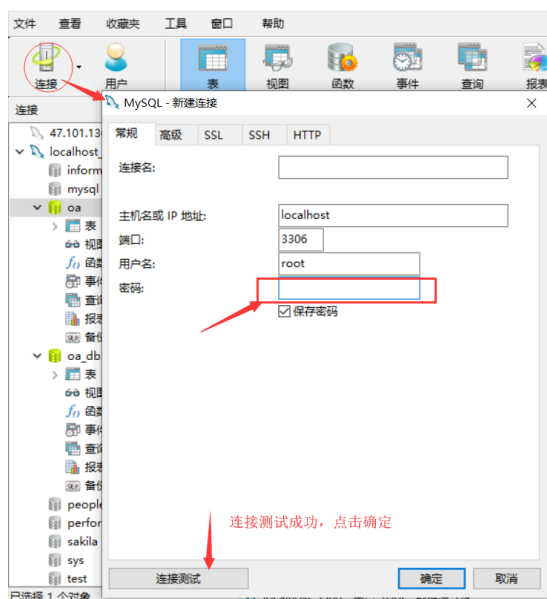
通过安装包“mysql-installer-community-5.7.13.0.exe”，安装 mysql5.7，也可以去官网下载安装，但是由于兼容性问题，建议先不要用 8.0 以上版本。

#### (2) 安装 navicat

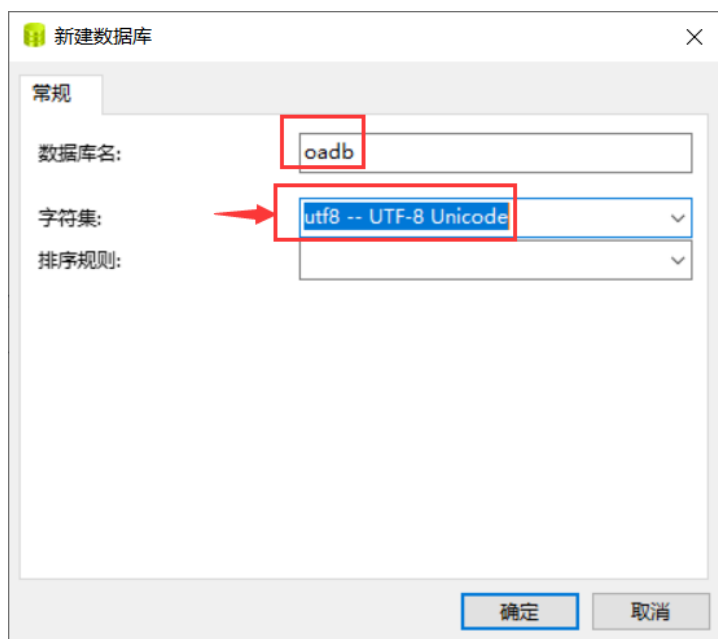
通过安装包“navicat091\_lite\_cs.exe”安装数据库管理可视化工具，这个是免费版，也可以自己去下载破解版。

#### (4) 创建数据库

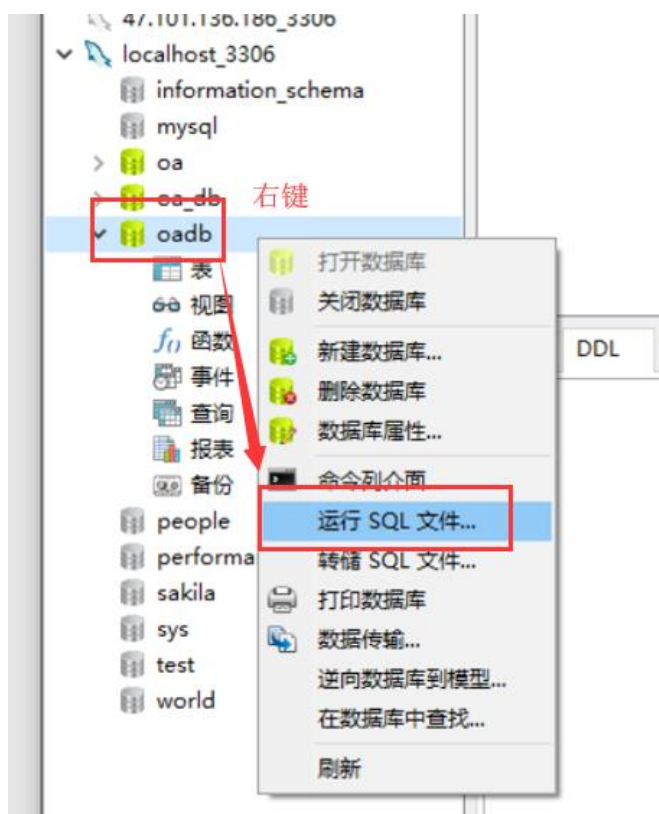
用 naviat 软件连接数据库:



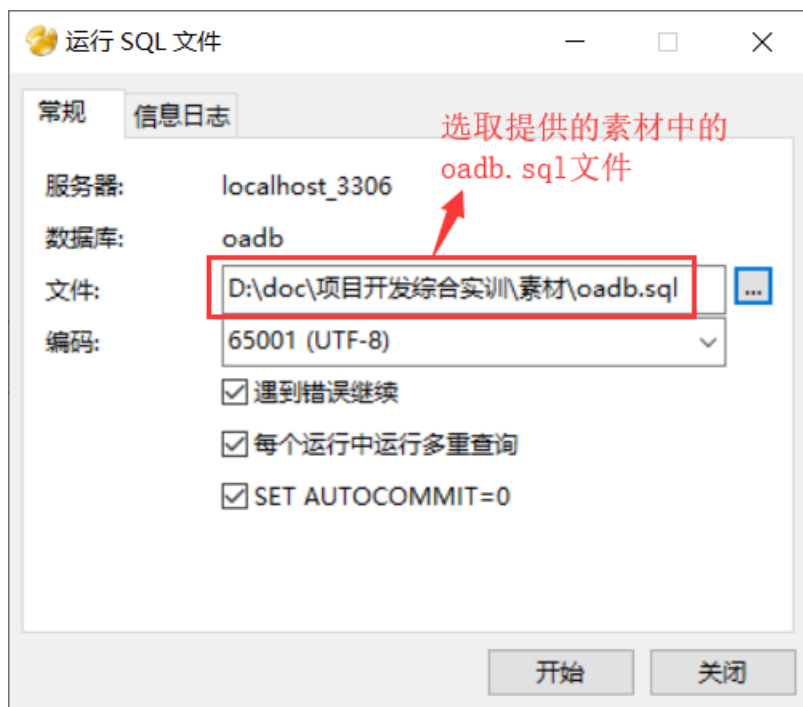
新建一个名称为“oadb”的数据库：



(4) 导入表结构和数据：







导入成功后，刷新就能看到表。



## 4. SpringBoot2 开发环境搭建

### 4.1. 安装 JDK

双击“jdk-8u162-windows-x64.exe”，启动安装程序进行安装，JDK1.8 以上版本可以不用配置环境变量。


### 4.2. STS 环境（集成 spring 版本的 eclipse）

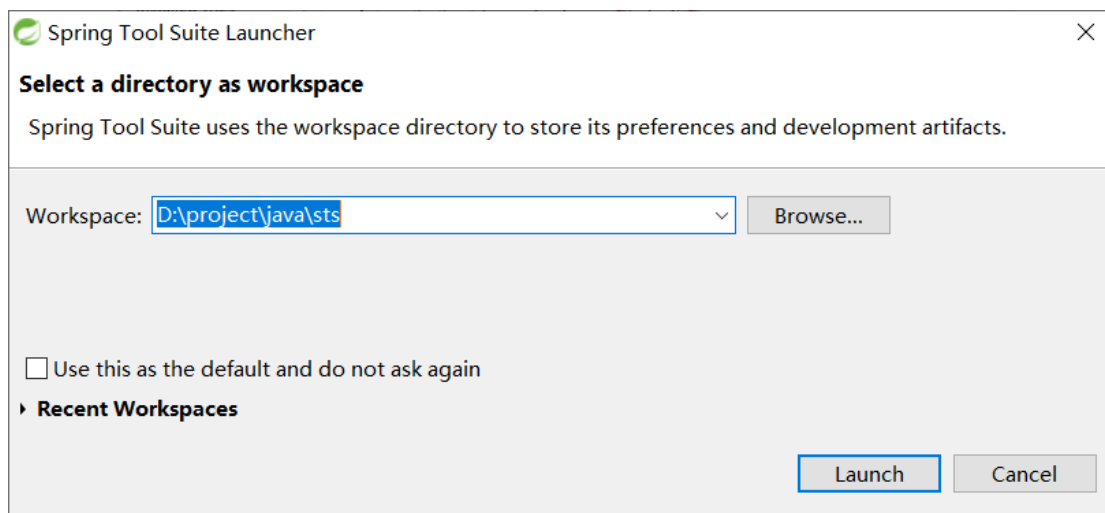
#### 4.2.1. 安装 STS

STS 不用安装，直接解压就能运行。建议在 D 盘或者 E 盘新建一个文件夹“sts”，然后把“spring-tool-suite-3.9.6.RELEASE-e4.9.0-win32-x86\_64.zip”解压到该目录。

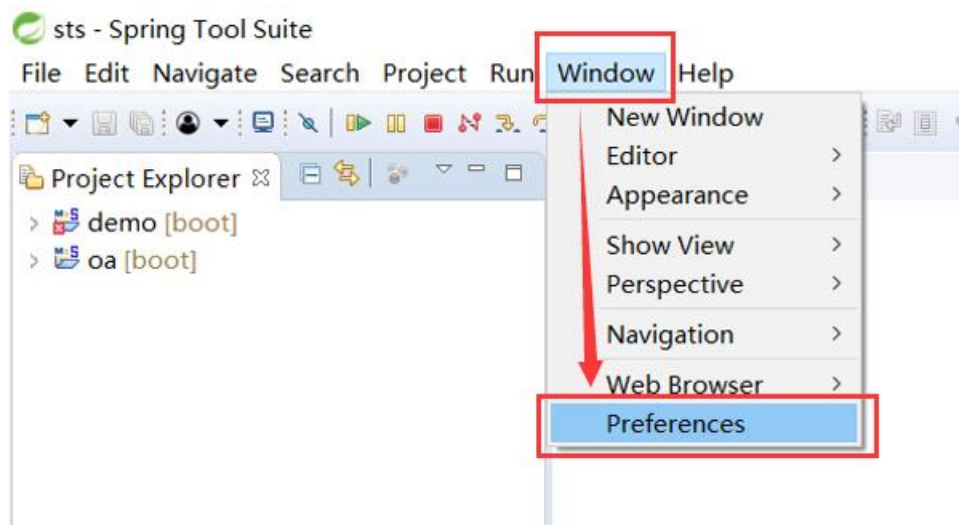
## 4.2.2. 配置 STS

### (1) 设置工作区

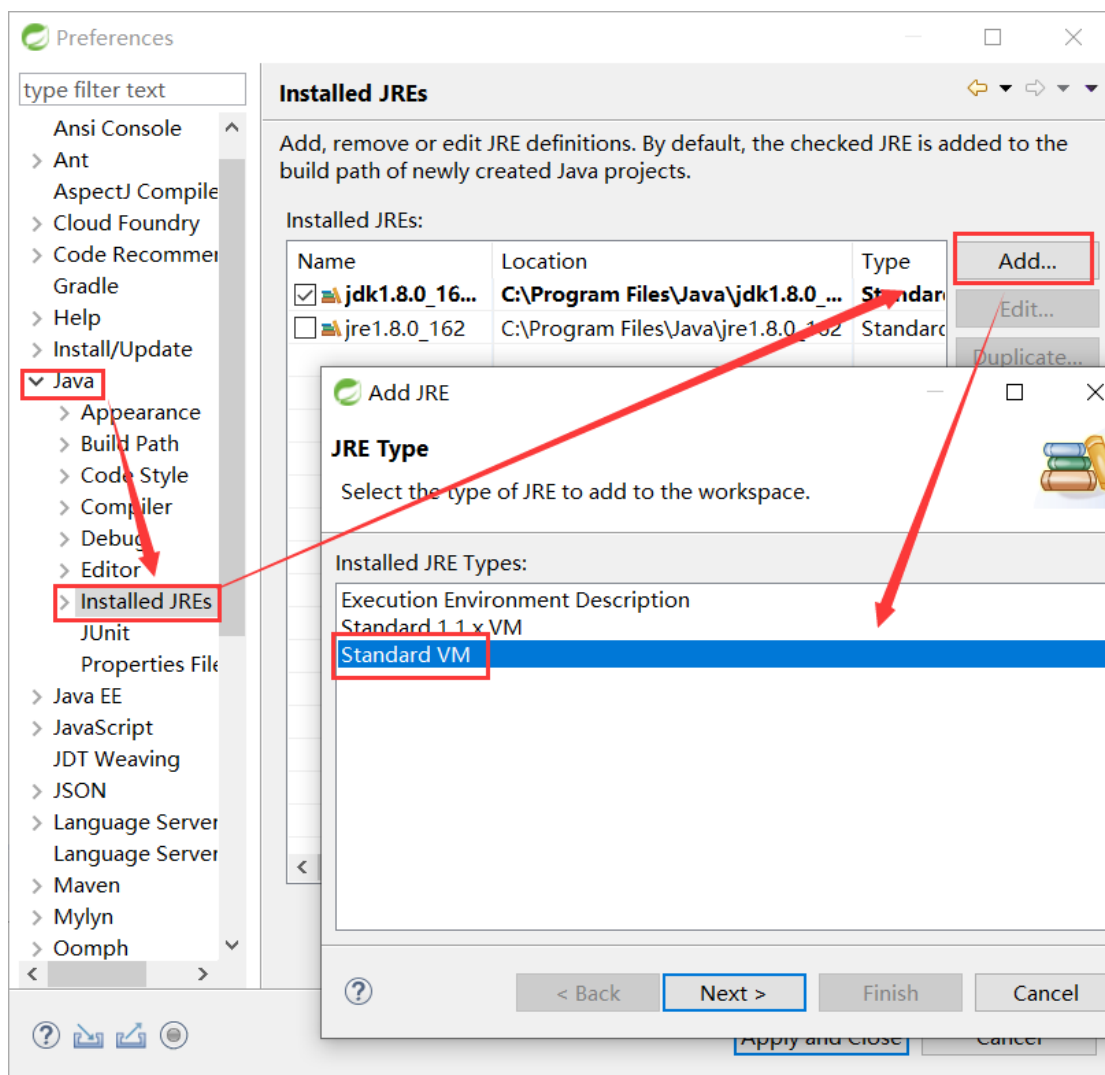
运行解压目录下的“STS.exe” |  STS，初次运行会让你确定自己的工作区，工作区是你创建 project 工程的地方（**在机房使用建议选择在 E 盘的文件夹**）。



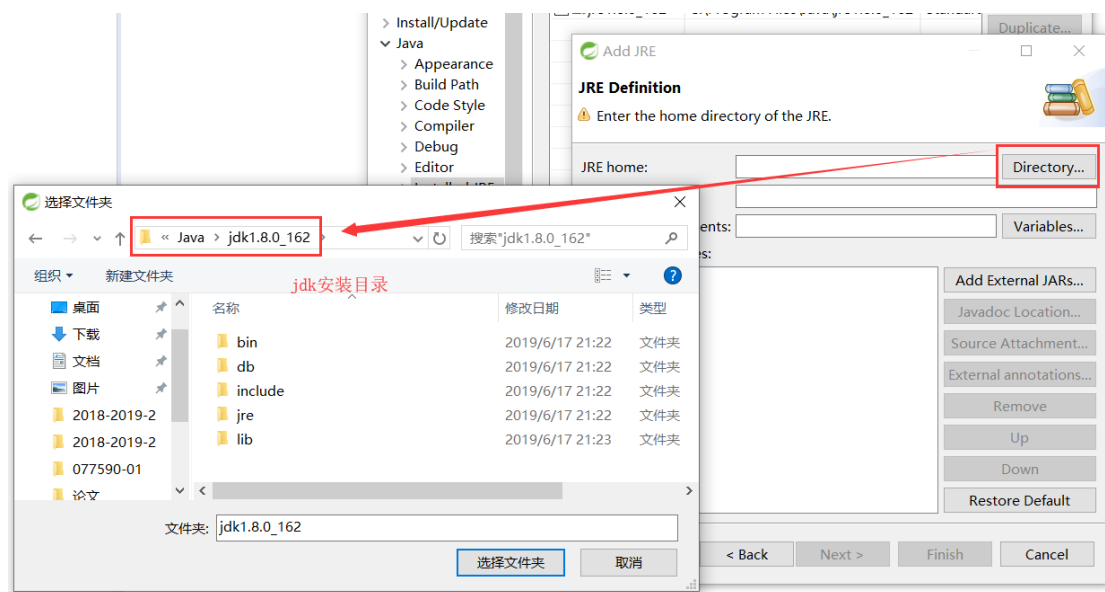
### (2) Java 环境配置

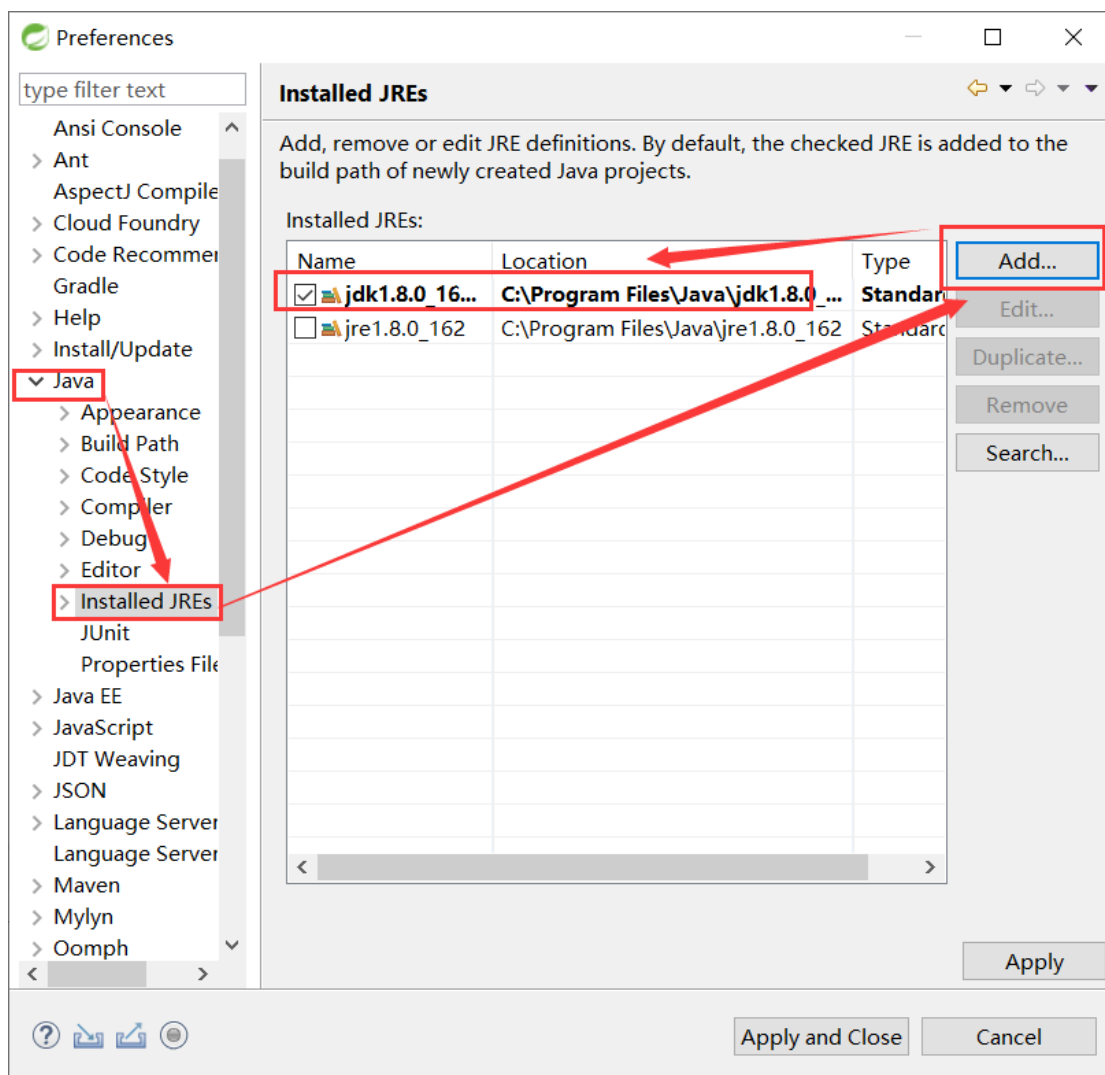


选择 Java 运行环境：

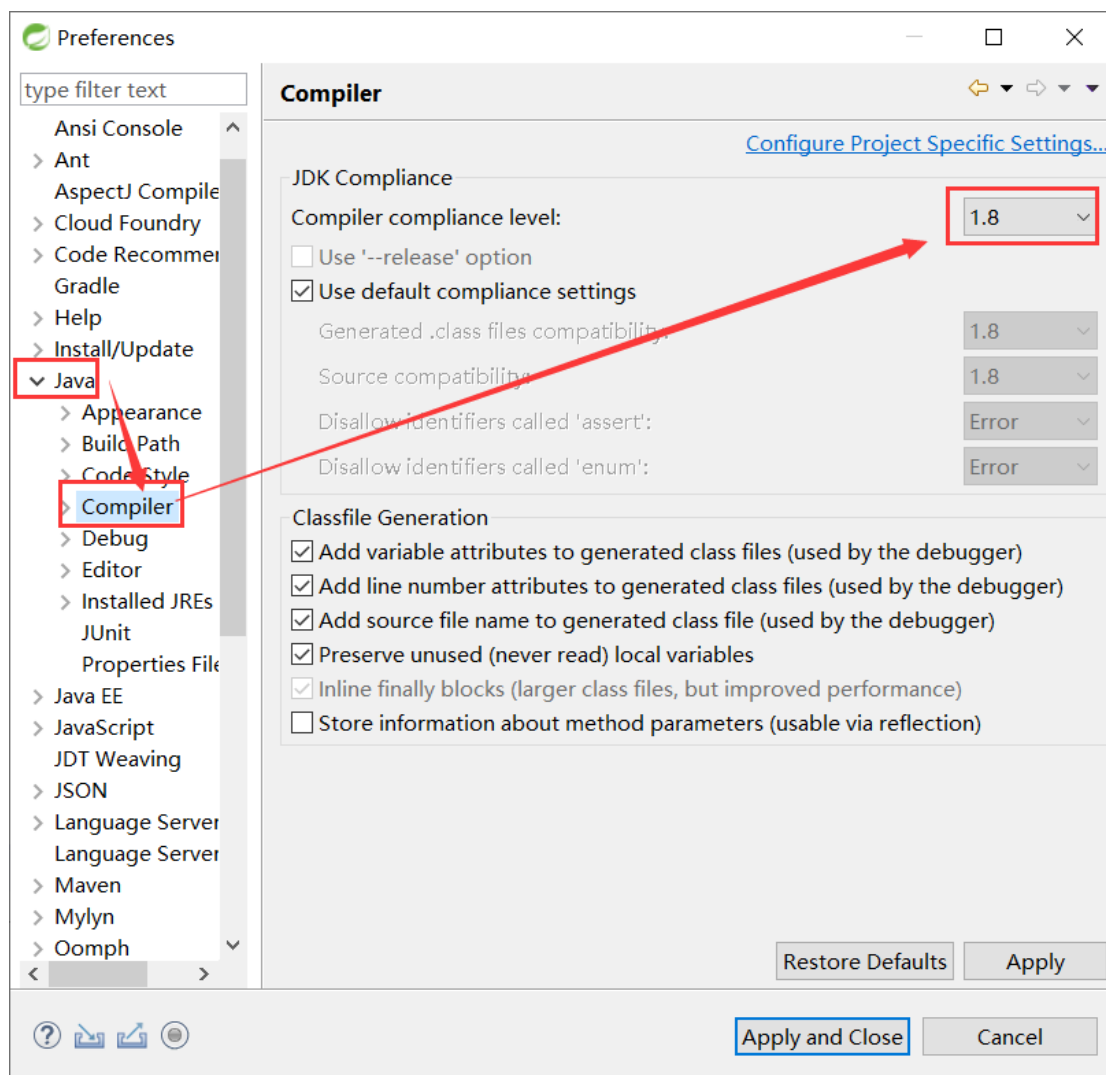


JDK 默认安装在: C:\Program Files\Java\jdk1.8.0\_162



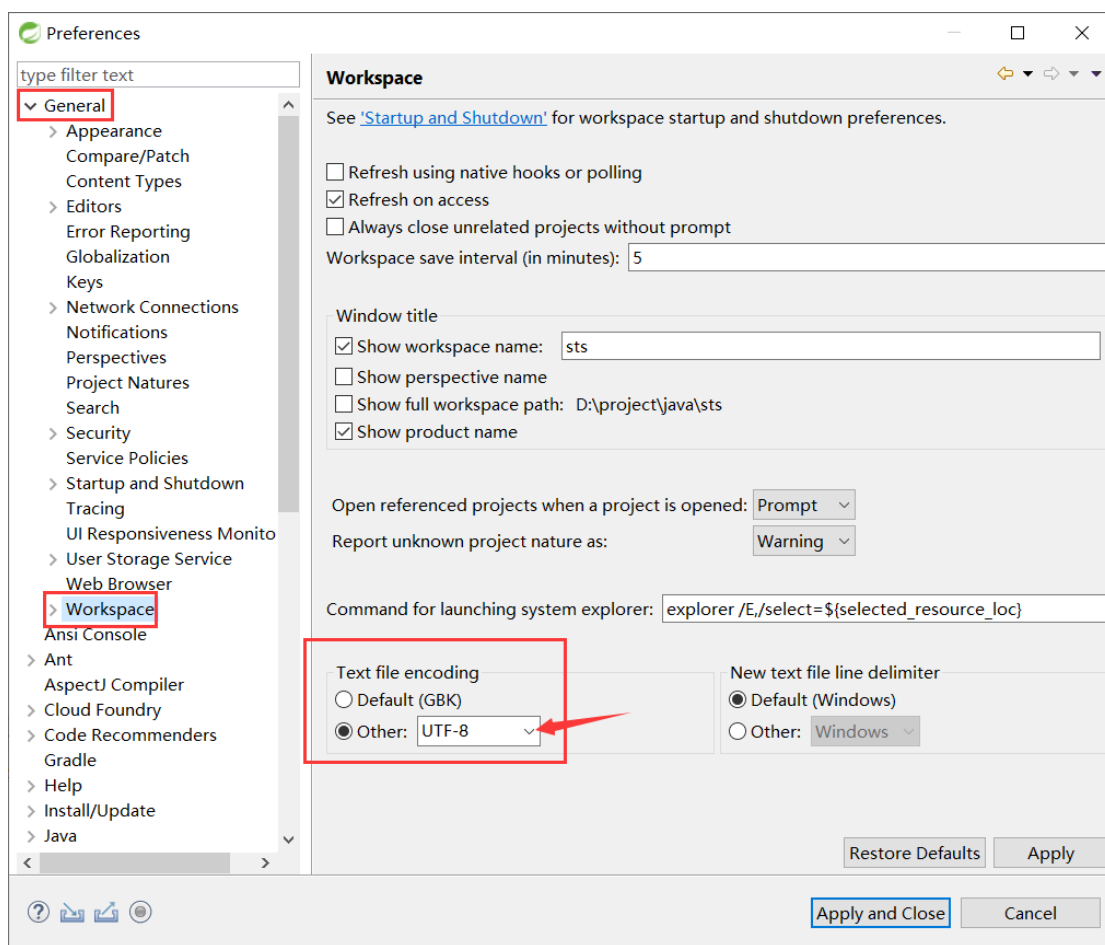


选择 Java 编译版本:

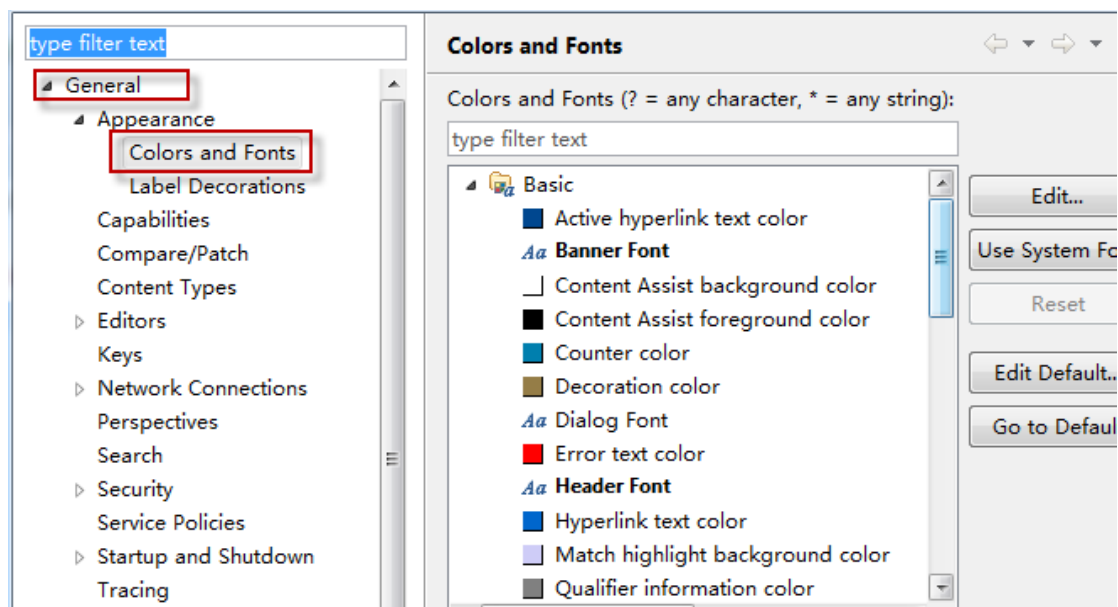


### (3) 设置默认字符集

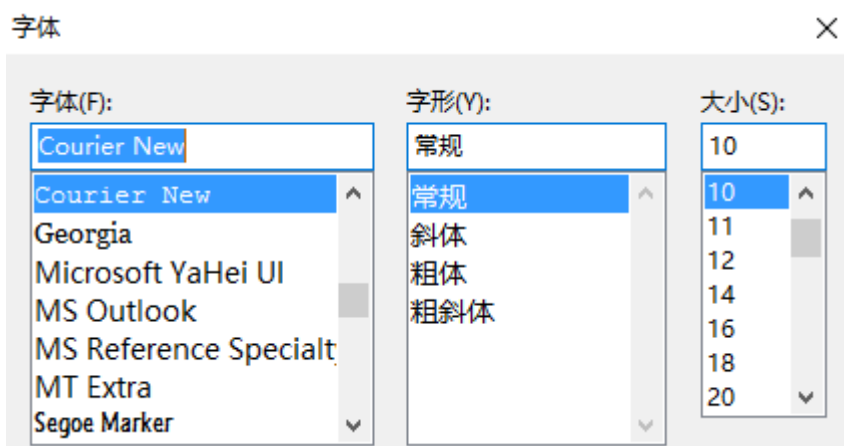
将默认字符集更改为 utf-8



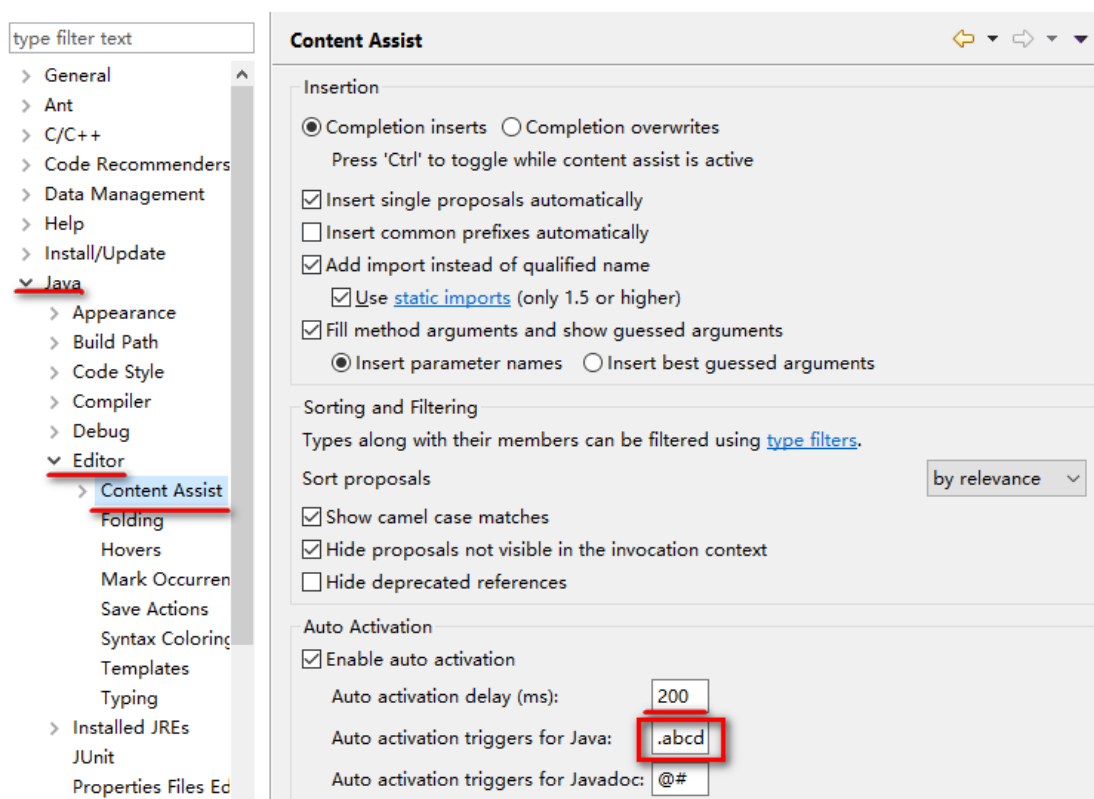
(4) 字体修改



选择 Basic 下的“Text Font”，然后点击右侧的“Edit”按钮，在弹出的字符对话框中选择“Courier New”字体。



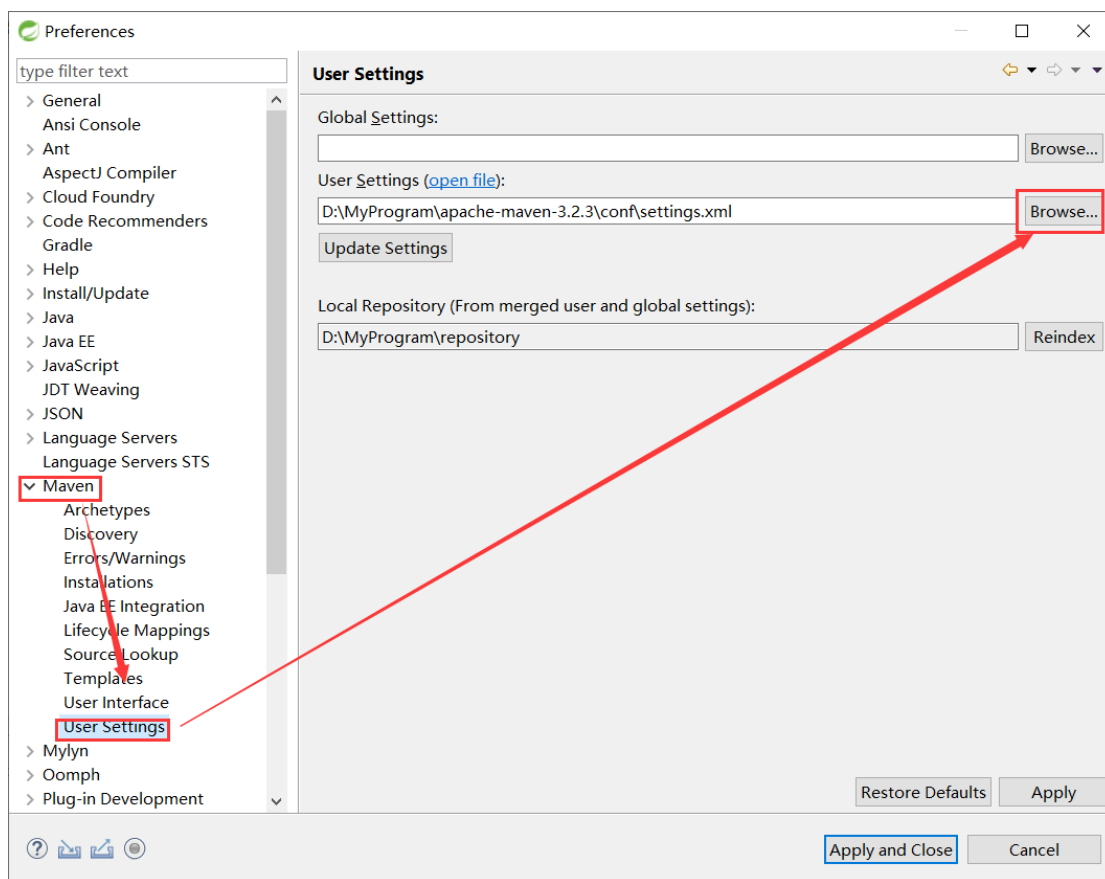
(5) 添加代码提示



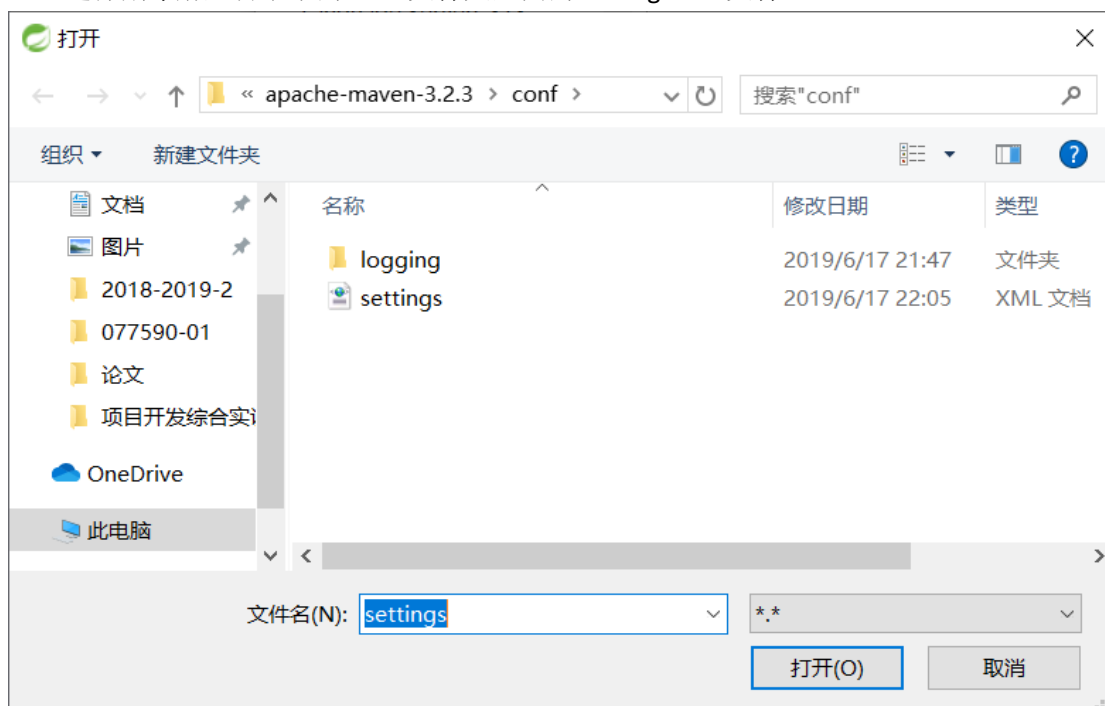
在上边红色框内输入：.abcdefghijklmnopqrstuvwxyz

(6) Maven 环境配置

首先将“apache-maven-3.2.3.zip”解压，例如解压到“E:\maven”下。  
然后，在 STS 中进行如下设置。

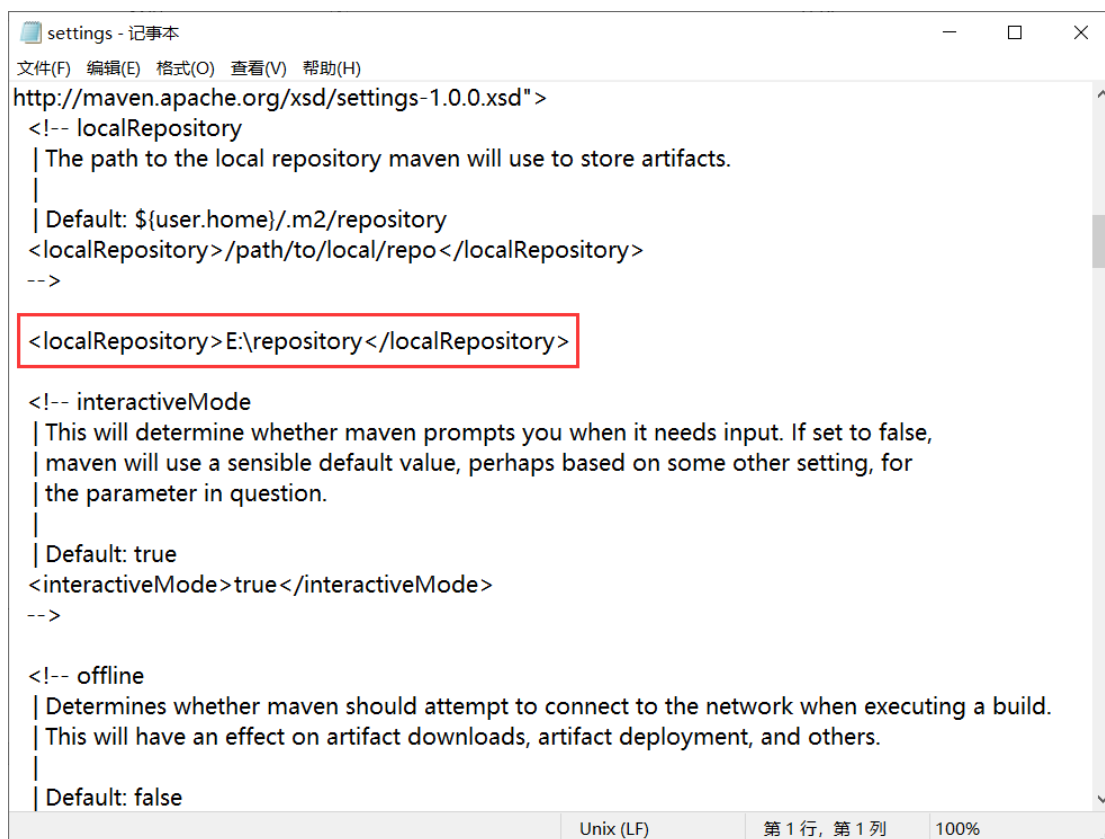


选择刚才解压目录下的 `conf` 文件夹里面的 `settings.xml` 文件。



注意在 `settings.xml` 文件里面有一个配置 `maven` 本地仓库的目录，这个目录在自动被读取出来，如果要修改，请用文本工具修改“`settings.xml`”，并建立好仓库文件夹。



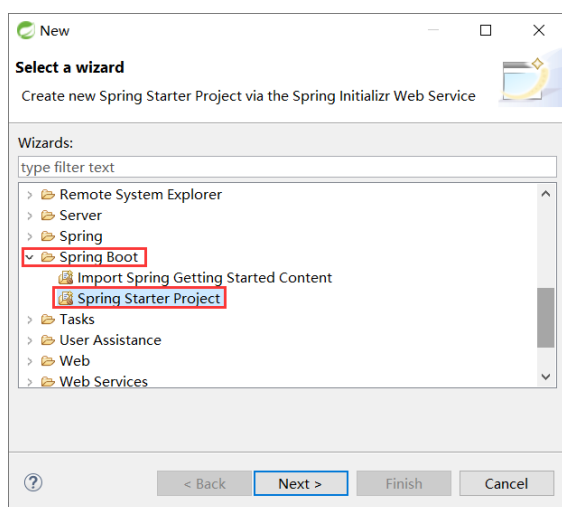


## 4.3. 项目测试

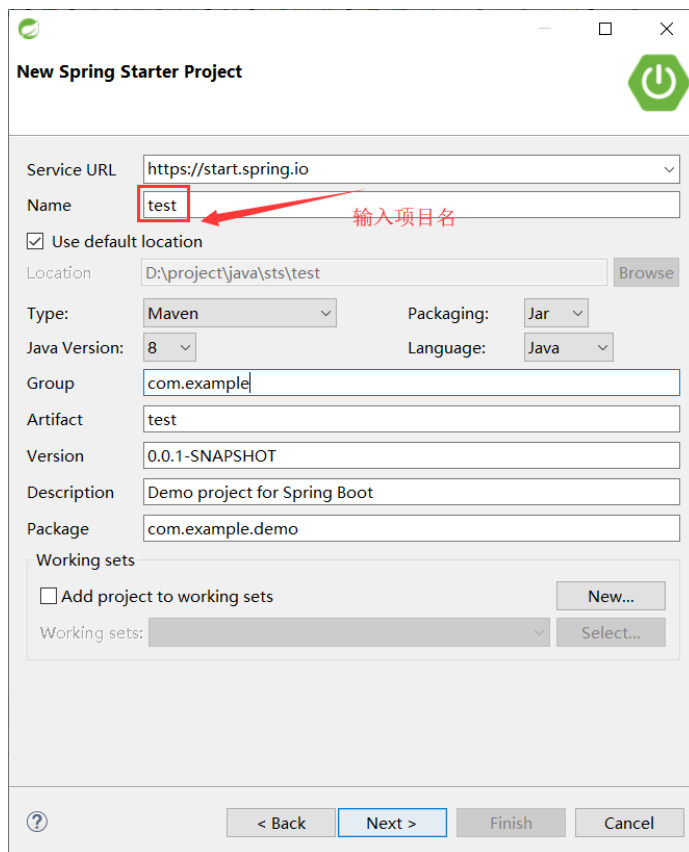
### 4.3.1. 创建 springboot 项目

菜单：File-》new-》other。

选择 starter 项目。

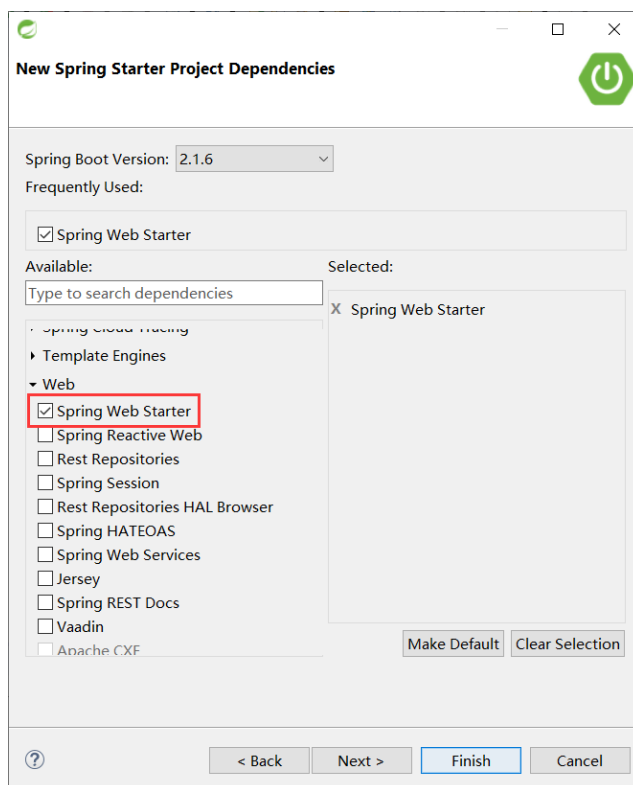


下一步，要保证能连接互联网，输入项目名称。



The 'New Spring Starter Project' dialog box is shown. The 'Name' field is highlighted with a red box and a red arrow pointing to it, with the text '输入项目名' (Enter project name) next to it. The 'Name' field contains the text 'test'. Other fields include 'Service URL' (https://start.spring.io), 'Location' (D:\project\java\sts\test), 'Type' (Maven), 'Packaging' (Jar), 'Java Version' (8), 'Language' (Java), 'Group' (com.example), 'Artifact' (test), 'Version' (0.0.1-SNAPSHOT), 'Description' (Demo project for Spring Boot), and 'Package' (com.example.demo). The 'Working sets' section has an unchecked checkbox 'Add project to working sets' and buttons 'New...' and 'Select...'. At the bottom are buttons '< Back', 'Next >', 'Finish', and 'Cancel'.

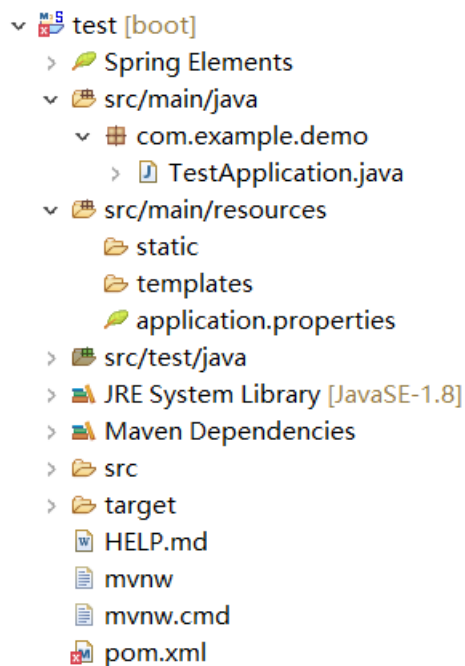
下一步。



The 'New Spring Starter Project Dependencies' dialog box is shown. The 'Spring Boot Version' is set to 2.1.6. Under 'Frequently Used:', 'Spring Web Starter' is checked. In the 'Available:' list, 'Spring Web Starter' is also checked and highlighted with a red box. The 'Selected:' list shows 'Spring Web Starter' with an 'X' icon. At the bottom are buttons '< Back', 'Next >', 'Finish', and 'Cancel'.

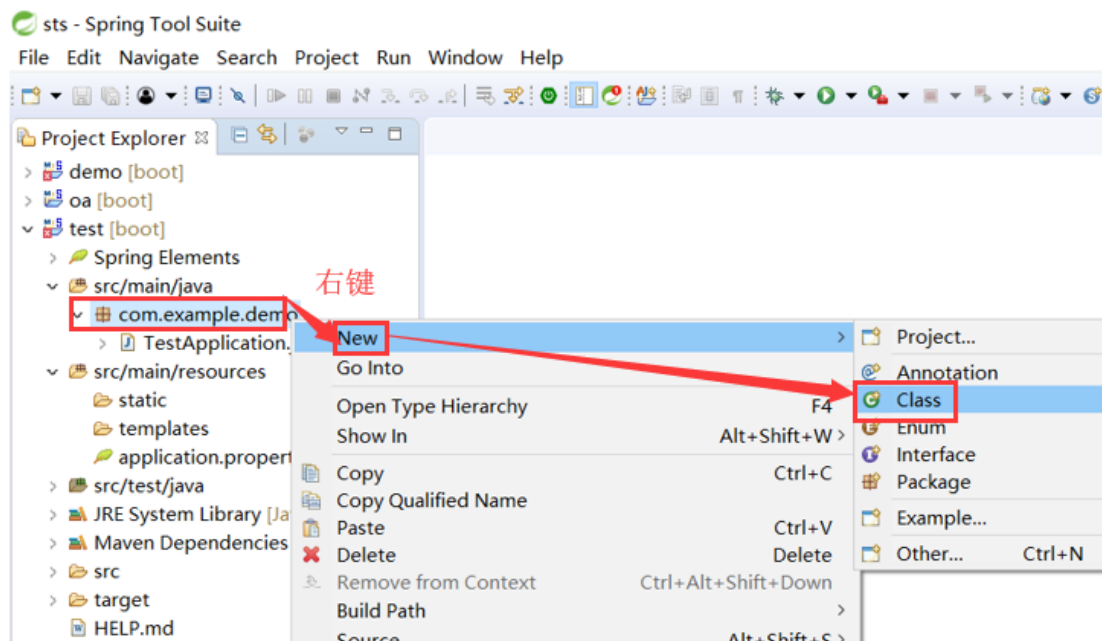
勾选 Spring Web Starter, Finish。

项目结构如下：



### 4.3.2. 编写代码

新建一个 Controller。



输入类名：HelloController

**New Java Class**

Create a new Java class.

Source folder:  Browse...

Package:  Browse...

☐ Enclosing type:  Browse...

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:  Browse...

Interfaces:  Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

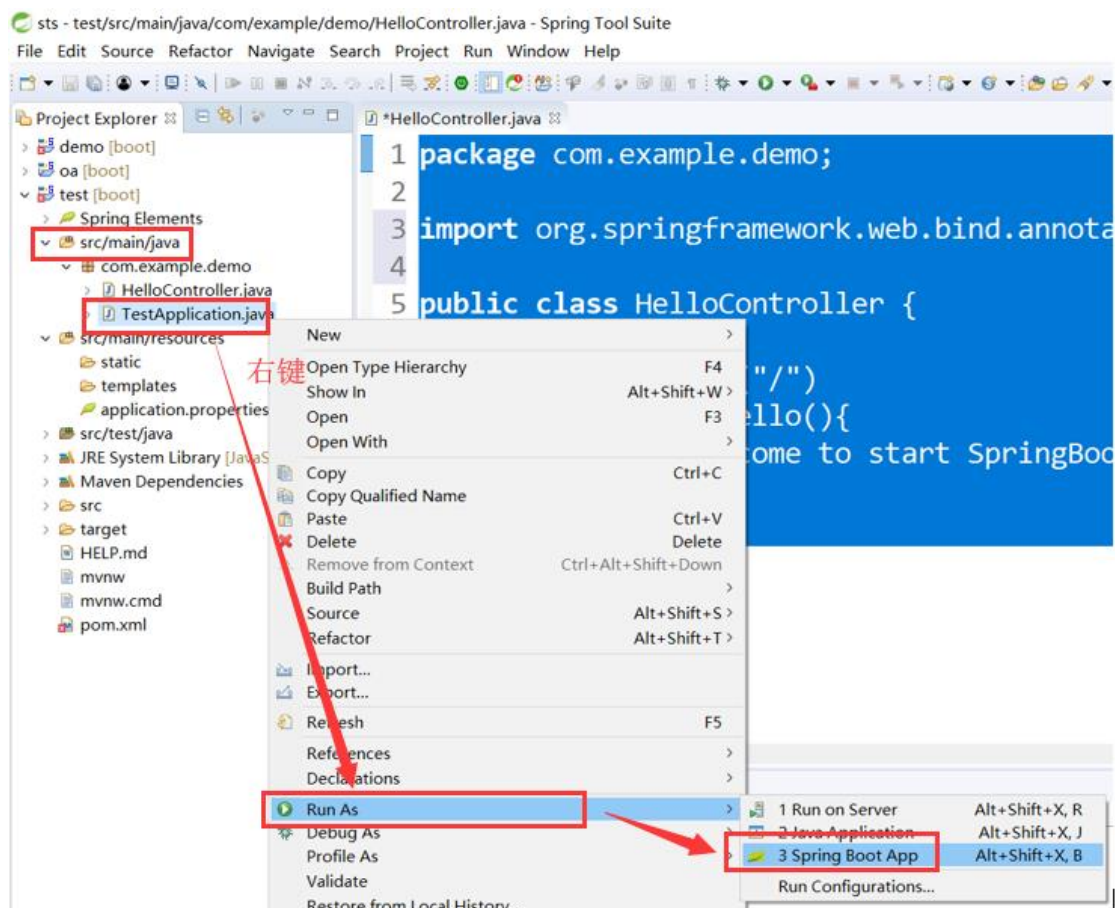
Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

输入如下代码（可以复制）：

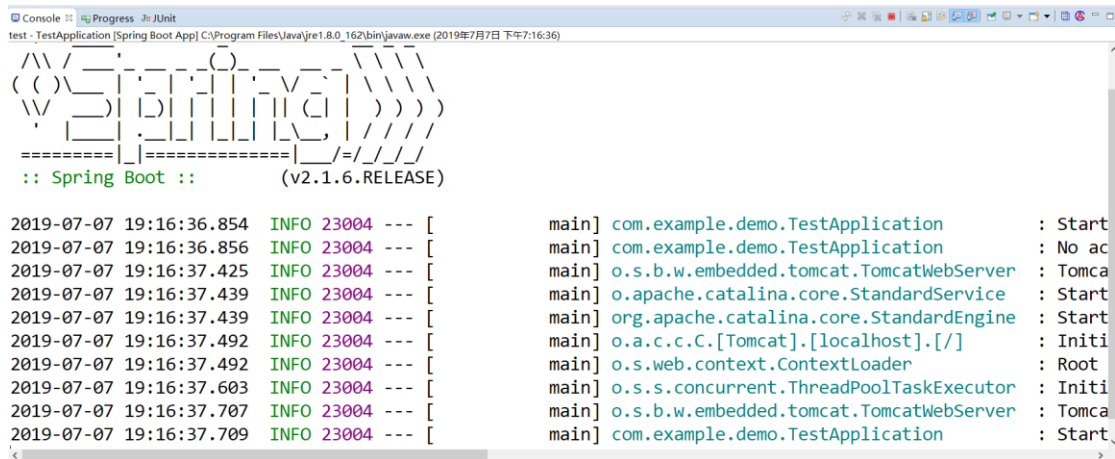
```
package com.example.demo;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
    @RequestMapping("/")
    public String hello(){
        return "Welcome to start SpringBoot!";
    }
}
```

### 4.3.3. 运行测试

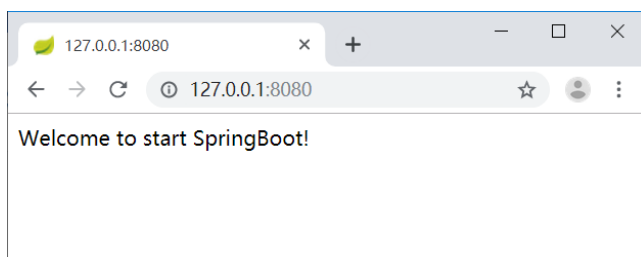
启动项目：



启动成功:



打开浏览器，输入地址 “http://127.0.0.1:8080/” 进行测试:



## 5. 项目搭建

**Maven**，各种包的定义，前端框架，配置文件（基本的）

### 5.1. 创建项目

#### （1）新建 springboot 项目

项目名为“mis”，group 和 package 为“com.example.demo”（用其他包名时，注意后面的代码需要自己修改包名，建议用此包名）：

**New Spring Starter Project**

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

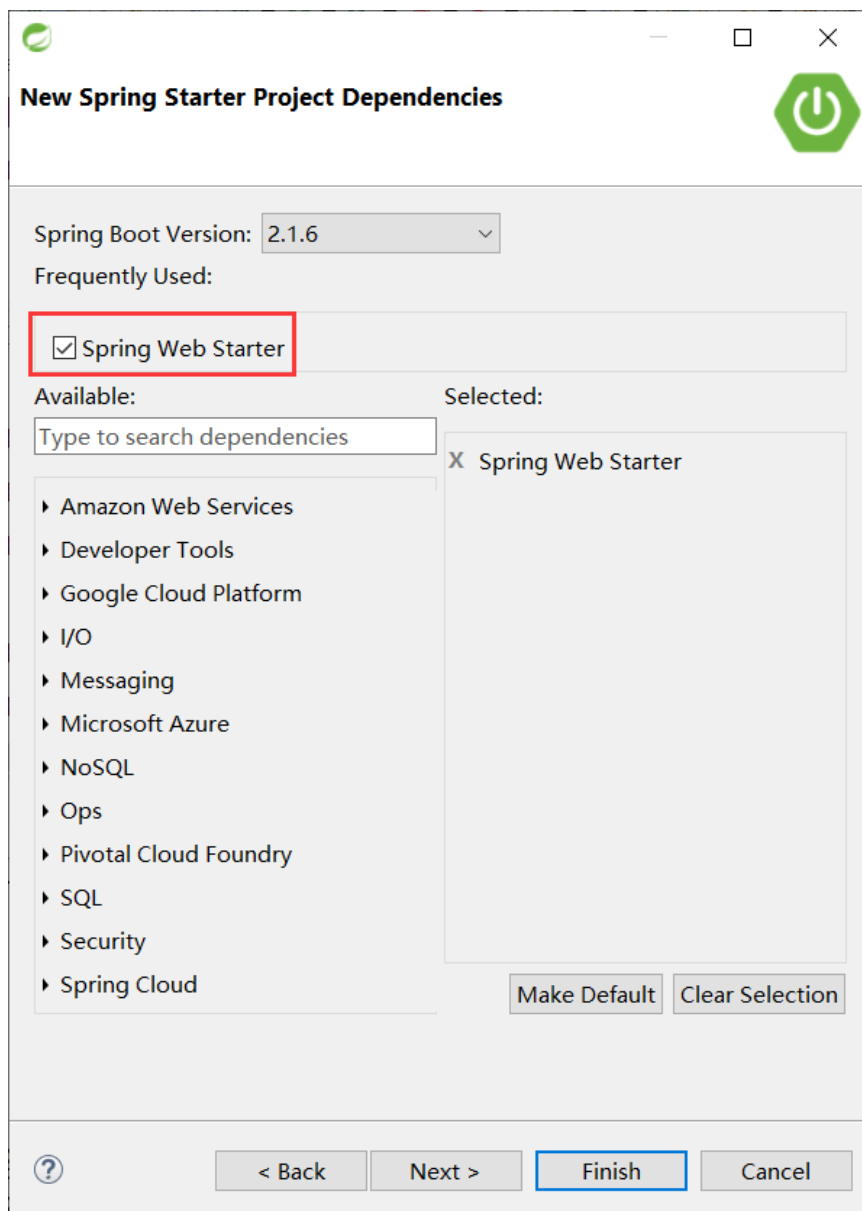
Description:

Package:

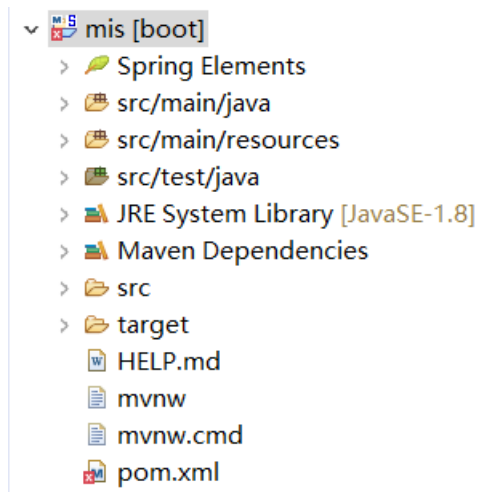
Working sets

☐ Add project to working sets

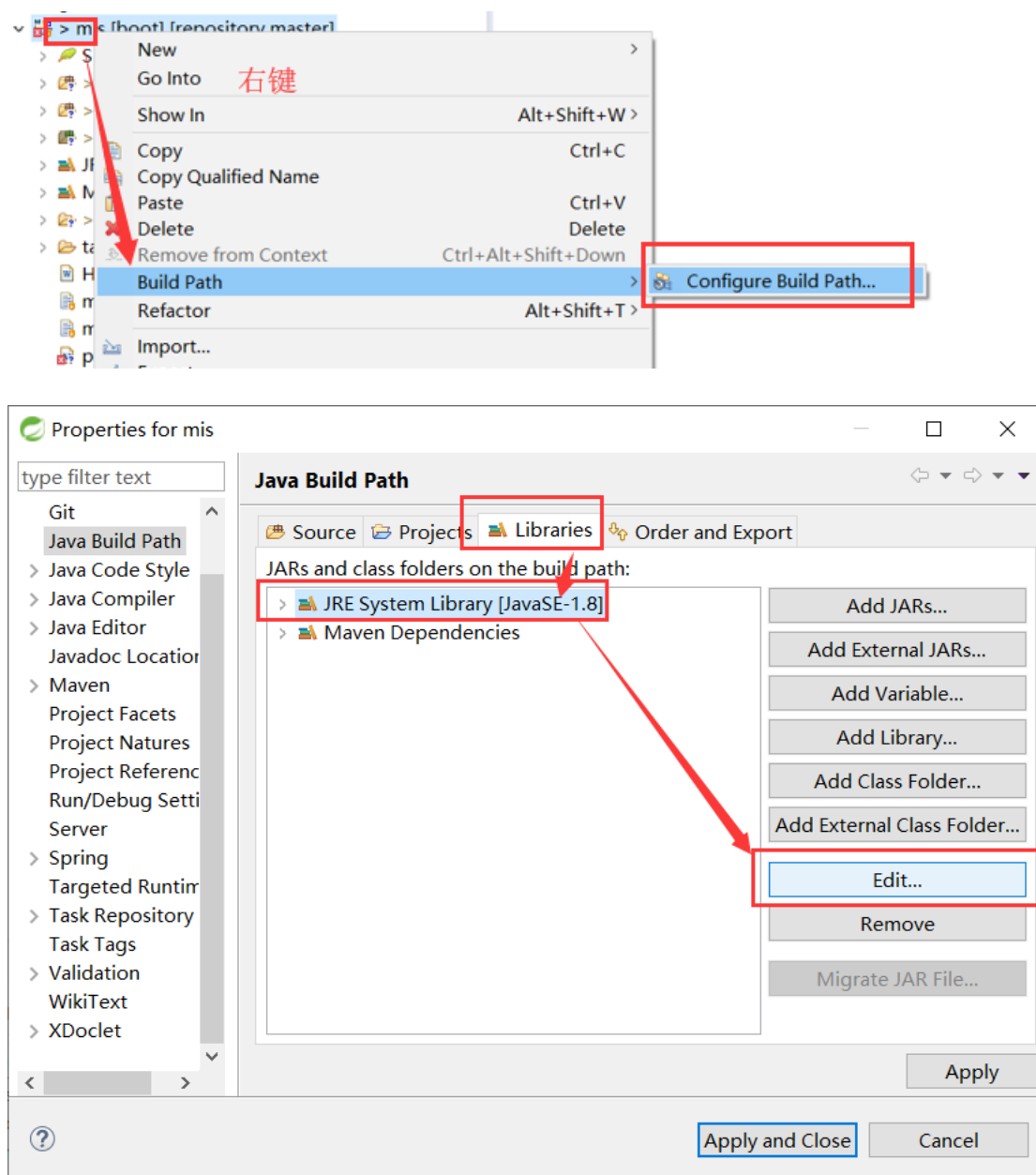
Working sets:



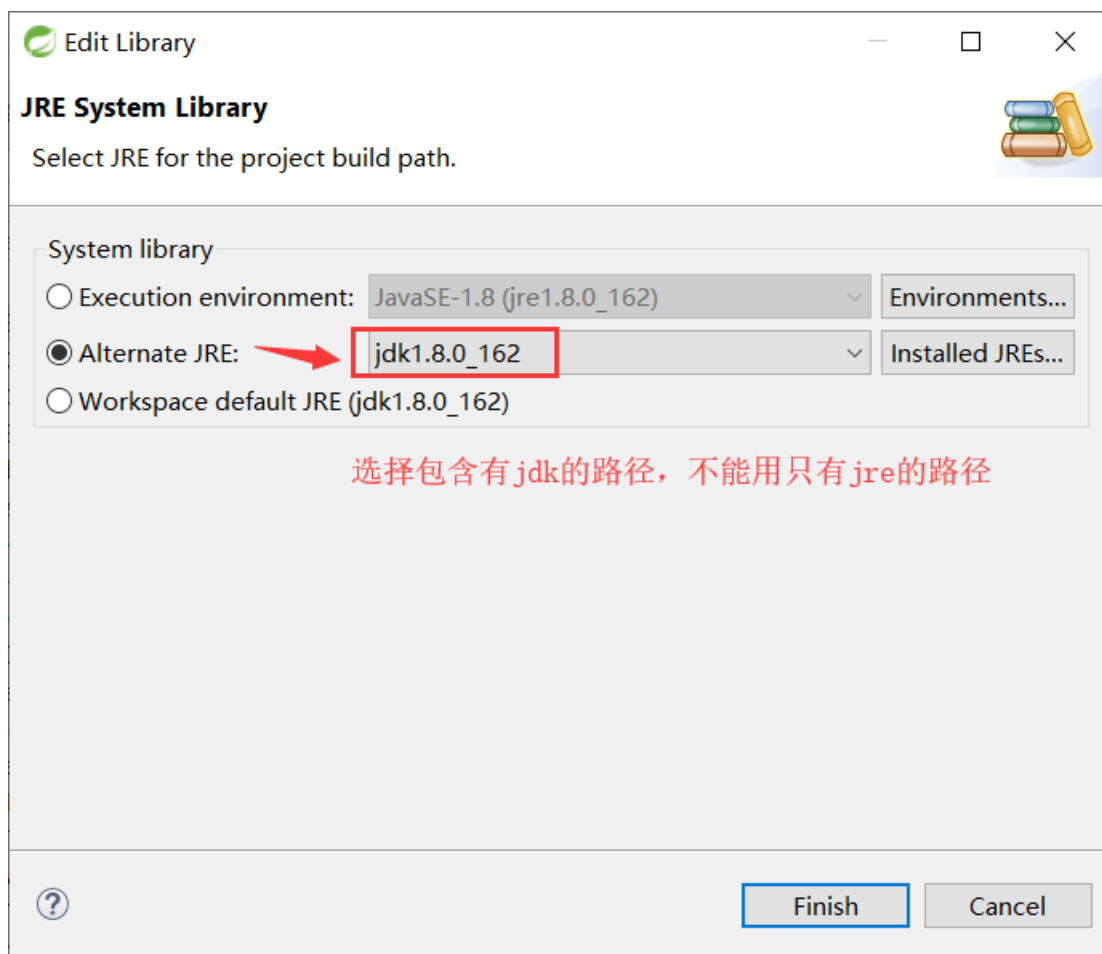
项目结构如下：



## (2) 修改编译环境



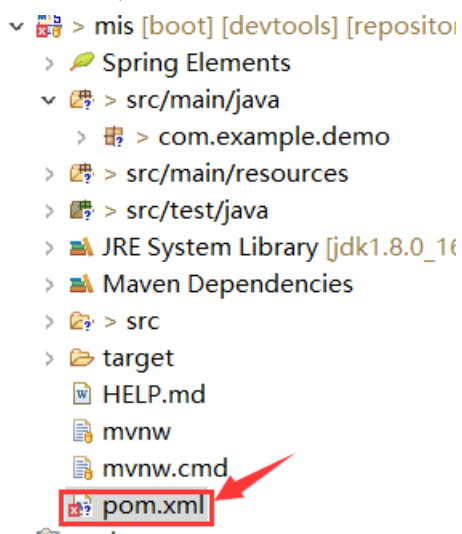




## 5.2. 项目配置

### (1) maven 配置

打开 pom 文件：



在“dependencies”的子节点中，加入以下依赖：



```

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.30</version>
</dependency>
<dependency>
  <groupId>com.baomidou</groupId>
  <artifactId>mybatis-plus-boot-starter</artifactId>
  <version>2.2.0</version>
</dependency>

<!-- thymeleaf模版 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>

<!-- 验证码 -->
<dependency>
  <groupId>com.github.axet</groupId>
  <artifactId>kaptcha</artifactId>
  <version>0.0.9</version>
</dependency>

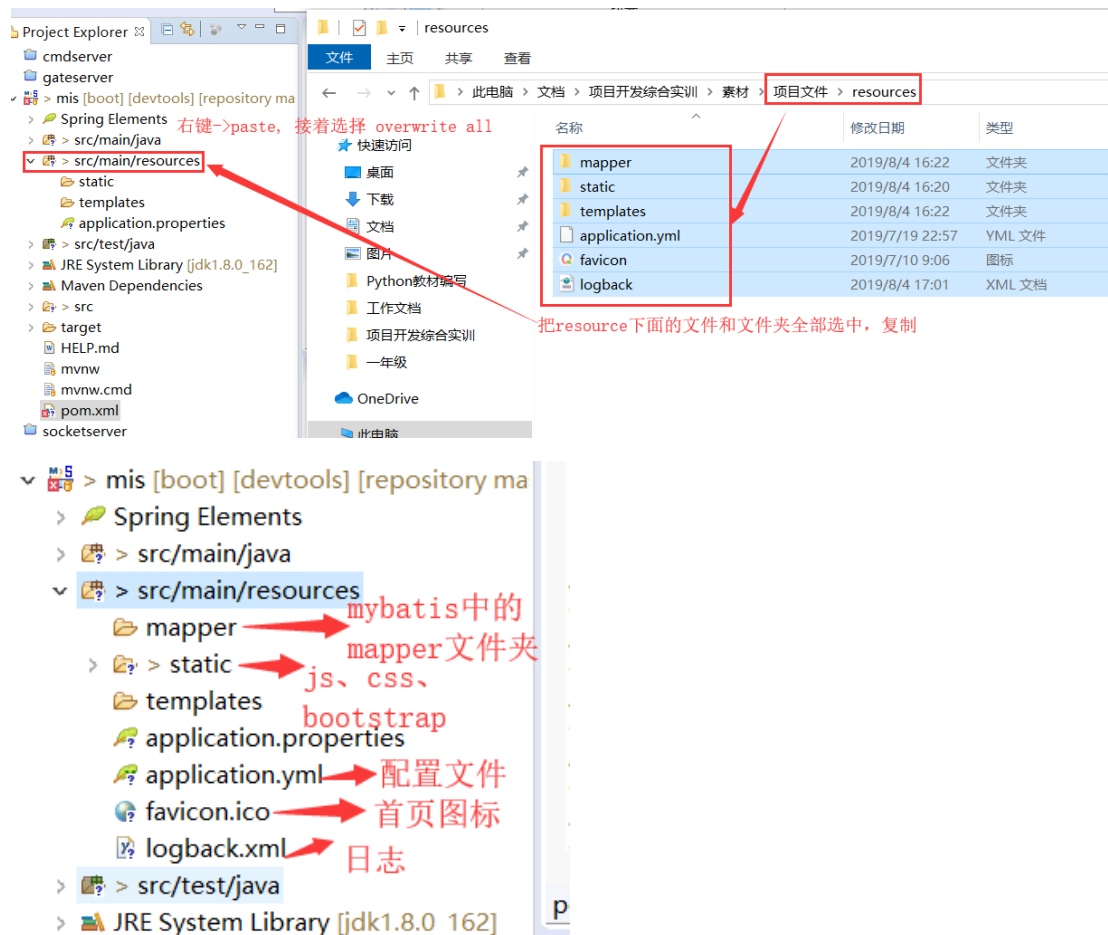
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>

```

```
<optional>true</optional> <!-- 这个需要为 true 热部署才有效 -->
</dependency>
```

## (2) 静态资源文件设置

复制素材目录下的静态文件素材到项目中：



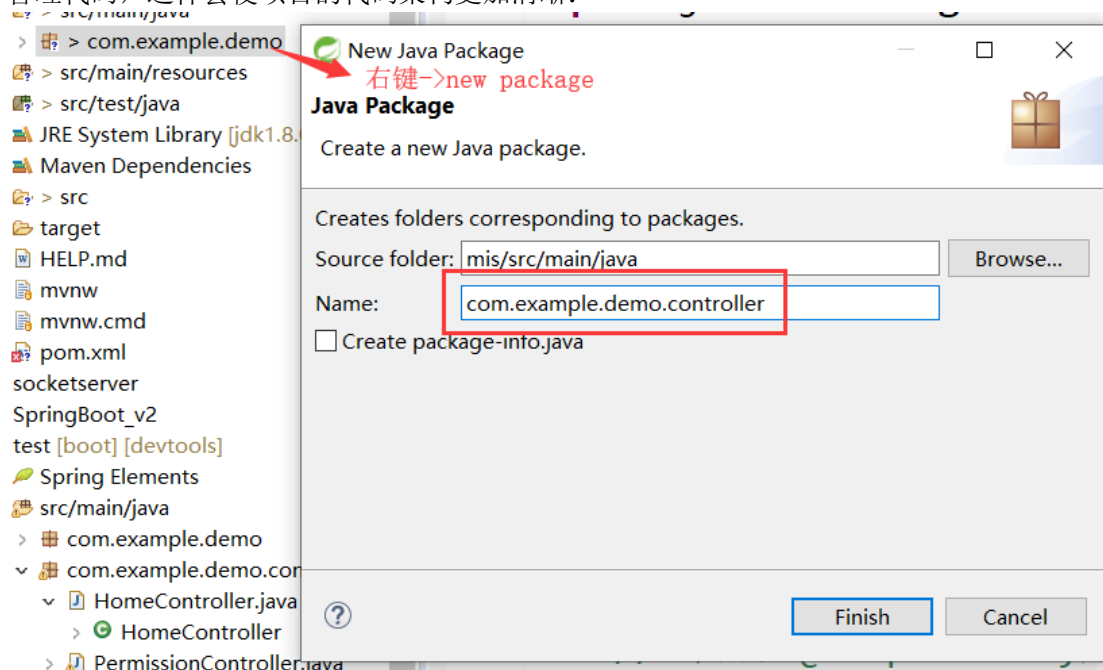
## (3) 配置文件设置

删除自动生成的配置文件“application.properties”这种格式不能输入中文，不好注释，保留“application.yml”文件，数据库名以及用户名和密码根据实际情况修改：

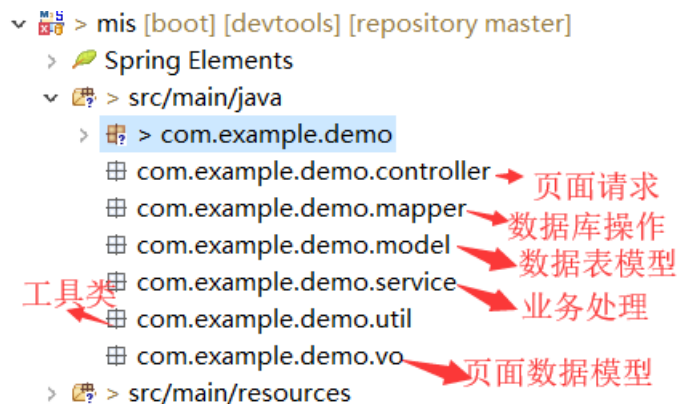
```
1 spring:
2   datasource:
3     driverClassName: com.mysql.jdbc.Driver #冒号后面要有空格，下同
4     url: jdbc:mysql://localhost:3306/oadb?useUnicode=true&characterEncoding=utf-8
5     username: root
6     password: 123456 #数据库用户名和密码
7   mvc:
8     static-path-pattern: /static/** #静态资源映射路径
9   thymeleaf:
10    prefix: classpath:/templates/
11 mybatis-plus:
12   mapper-locations: classpath*:./mapper/**Mapper.xml
13   type-aliases-package: com.example.demo.model #model的包名，在后台源码中必须有这个包名
14   configuration:
15     log-impl: org.apache.ibatis.logging.stdout.StdoutImpl #打印SQL日志，便于调试
16 upload: C:/upload/ #自定义文件上传路径 自定义的属性
```

#### (4) 包的设置

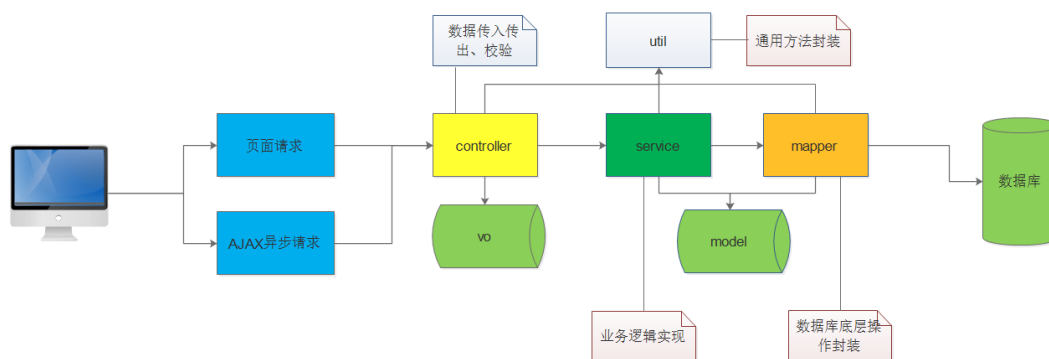
新建 “ com.example.demo.controller ” 、 “ com.example.demo.mapper ” 、 “ com.example.demo.model ” （注意与配置文件 application.yml 中的包名一致）、 “ com.example.demo.service ” 、 “ com.example.demo.vo ” 、 “ com.example.demo.util ” 5 个包来管理代码，这样会使项目的代码架构更加清晰：



包结构如下：



各个包之间关系如下图：



## (5) 添加通用类

复制“util”文件夹下的源码文件到 util 包中（字符串和 list 转换、md5 加密、雪花主键生成），复制方法同上：

文档 > 项目开发综合实训 > 素材 > 项目文件 > util

名称	修改日期
Convert.java	2019/7/8 16
MD5Util.java	2019/7/22 2
SnowflakeldWorker.java	2019/7/22 1

复制“vo”文件夹下的源码文件到 vo 包中（datatableresult 表格展示数据格式、json 异步请求数据返回格式、menu 菜单数据格式、ztreencode 树形结构数据格式，主要与前端页面配合使用），复制方法同上：

文档 > 项目开发综合实训 > 素材 > 项目文件 > vo

名称	修改日期	类型
DataTableResult.java	2019/7/9 8:42	JAVA 文
Json.java	2019/7/8 18:22	JAVA 文
Menu.java	2019/7/22 23:16	JAVA 文
MenuList.java	2019/7/22 23:17	JAVA 文
ZTreeNode.java	2019/7/9 14:54	JAVA 文

mis > [boot] [devtools] [repository master]

> Spring Elements

> src/main/java

> com.example.demo

> MisApplication.java

com.example.demo.controller

com.example.demo.mapper

com.example.demo.model

com.example.demo.service

> com.example.demo.util

> Convert.java

> MD5Util.java

> SnowflakeldWorker.java

> com.example.demo.vo

> DataTableResult.java

> Json.java

> Menu.java

> MenuList.java

> ZTreeNode.java

> src/main/resources

## 6. 模块实现

### 6.1. 用户管理模块

#### 6.1.1. 配置类

首先，在包“com.example.demo”中新建一个“MybatisPlusConfig”类（在这个类中还可以对 mybatis-plus 进一步进行配置，本例中启用了分页插件），代码如下：

```
package com.example.demo;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import com.baomidou.mybatisplus.plugins.PaginationInterceptor;

@EnableTransactionManagement
@Configuration
public class MybatisPlusConfig {

    /**
     * 分页插件
     */
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        return new PaginationInterceptor();
    }
}
```

相关知识点：

（1）@EnableTransactionManagement 注解

@EnableTransactionManagement 开启事务支持后，然后在访问数据库的 Service 方法上添加注解 @Transactional 便可。

（2）@Configuration 注解

@Configuration 用于类，表明这个类是 beans 定义的源，可以定义 bean。在 @Component 注解的类中不能定义类内依赖的@Bean 注解的方法，而@Configuration 可以。

（3）@Bean 注解

用@Bean 注解的方法：会实例化、配置并初始化一个新的对象，这个对象会由 spring IoC 容器管理。

接着，在包“com.example.demo”中新建一个“WebConfig”类（用来映射用户上传的文件以及后面实现登录拦截器的设置），代码如下：

```

package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    //图片存放根路径，从application.yml中读取upload
    @Value("${upload}")
    private String UPLOAD_PATH;

    /**
     * 文件上传
     */
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        //外部静态资源映射路径，用来上传文件
        String filePath = "file:" + UPLOAD_PATH;
        registry.addResourceHandler("/upload/**").addResourceLocations(filePath);
    }
}

```

@Value("\${upload}")表示从配置文件 application.yml 中读入配置信息，“upload”为配置的属性名称。registry.addResourceHandler("/upload/\*\*")，把上传路径映射到“/upload/”的 url 路径中，可以通过 url 地址访问到文件。

### 6.1.2. 用户 model 类

在包“com.example.demo.model”中新建一个“User”类（如果自己添加 model 类时，只须定义好属性，然后利用 eclipse 的生成 getter 和 setter 功能，直接生成代码），代码如下：

```

package com.example.demo.model;

import java.sql.Timestamp;

/**
 * 与数据库中的物理表映射，增删改一般都用这个模型实现
 */
import com.baomidou.mybatisplus.annotations.TableId;
import com.baomidou.mybatisplus.annotations.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;

```

```
@TableName("t_sys_user")//数据库对应表名
public class User {

    @TableId
    private String id;//@TableId表示主键，与字段名称大小写一致

    private String username;//与字段名称大小写一致
    private String password;//与字段名称大小写一致

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Timestamp birthday;//与字段名称大小写一致

    private String photo;//与字段名称大小写一致
    private String introduce;//与字段名称大小写一致
    private String usertype;//与字段名称大小写一致

    public String getUsertype() {
        return usertype;
    }
    public void setUsertype(String usertype) {
        this.usertype = usertype;
    }
    public Timestamp getBirthday() {
        return birthday;
    }
    public void setBirthday(Timestamp birthday) {
        this.birthday = birthday;
    }
    public String getPhoto() {
        return photo;
    }
    public void setPhoto(String photo) {
        this.photo = photo;
    }
    public String getIntroduce() {
        return introduce;
    }
    public void setIntroduce(String introduce) {
        this.introduce = introduce;
    }
    public String getId() {
        return id;
    }
}
```



```

    public void setId(String id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

相关知识点:

(1) @TableName 注解

Mybatis plus 插件中的注解，指明 model 与数据库中关联的表名。

(2) @TableId 注解

@TableId 表示主键，与字段名称大小写一致。在设计表时字段名称建议不要使用大写字母，否则定义 model 属性时，生成 getter 和 setter 后看起来比较乱。

(3) @JsonFormat 注解

@JsonFormat 注解来源于 jackson，Jackson 是一个简单基于 Java 应用库，Jackson 可以轻松地将 Java 对象转换成 json 对象和 xml 文档，同样也可以将 json、xml 转换成 Java 对象。

### 6.1.3. 用户 vo 类

在包“com.example.demo.vo”中新建一个“UserVO”类（如果自己添加类时，只须定义好属性，然后利用 eclipse 的生成 getter 和 setter 功能，直接生成代码），代码如下：

```

package com.example.demo.vo;

import java.sql.Timestamp;

/**
 * 展示模型（一般不做修改操作），不一定与物理表字段一致，可以是几张关联表的字段组合
 * @author gjq
 *
 */
public class UserVO{

    private String id;

```

```
private String username;
private String password;

private Timestamp birthday;

private String photo;
private String introduce;
private String usertype;

// private String rolename;//关联角色表中的name字段，用来存放角色名

public String getUsertype() {
    return usertype;
}

public void setUsertype(String usertype) {
    this.usertype = usertype;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public Timestamp getBirthday() {
```

```

        return birthday;
    }

    public void setBirthday(Timestamp birthday) {
        this.birthday = birthday;
    }

    public String getPhoto() {
        return photo;
    }

    public void setPhoto(String photo) {
        this.photo = photo;
    }

    public String getIntroduce() {
        return introduce;
    }

    public void setIntroduce(String introduce) {
        this.introduce = introduce;
    }

//    public String getRolename() {
//        return rolename;
//    }
//
//    public void setRolename(String rolename) {
//        this.rolename = rolename;
//    }
}

```

VO 类与 model 类的区别：VO 主要与前端展示结合使用，可以是多张表的组合数据，以显示功能为主；而 model 与表是一一对应的，model 一般属性与表中的字段一致，实现表数据的增删改查功能，是更底层的数据模型。

#### 6.1.4. 用户 mapper 类以及 xml 文件

在包 “com.example.demo.mapper” 中新建一个 “UserMapper” 接口，代码如下：

```

package com.example.demo.mapper;

import java.util.List;

```

```

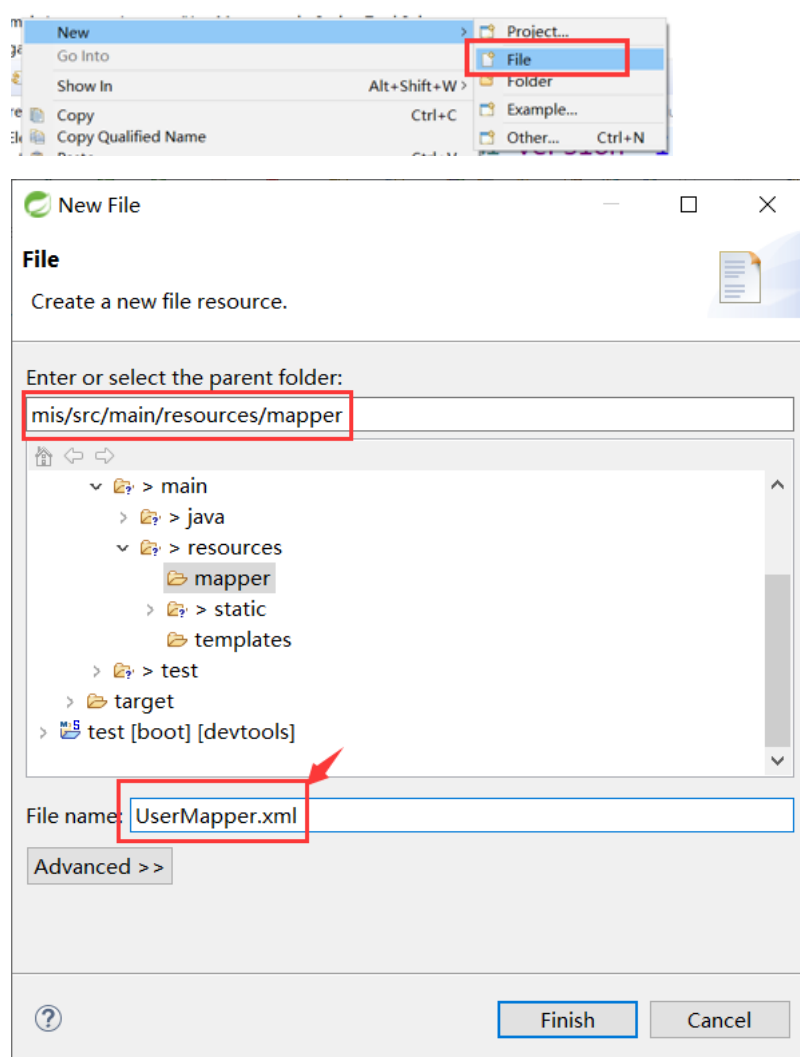
import org.apache.ibatis.annotations.Mapper;
import com.baomidou.mybatisplus.mapper.BaseMapper;
import com.baomidou.mybatisplus.plugins.pagination.Pagination;
import com.example.demo.model.User;
import com.example.demo.vo.UserVO;

@Mapper
public interface UserMapper extends BaseMapper<User>{
    // Integer listCount();
    // User findUserByUsername(String username);
    List<UserVO> selectUserListPage(Pagination page ,UserVO userVO);
}

```

通过继承 `BaseMapper<User>`，拥有了通用 mapper 的基本增删改查功能，因此用户表的基本增删改查功能不必再实现，这里定义的“`selectUserListPage`”是调用 xml 文件中的自定义查询，主要是可以实现多表连接查询和多条件查询（本例中较为简单，没有用到连接查询），扩展通用 mapper 没有的功能。

在“src/main/resources”中的“mapper”文件夹中，新建一个文件，命名为“UserMapper.xml”，代码如下：



```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<!-- 指明当前xml对应的Mapper -->
<mapper namespace="com.example.demo.mapper.UserMapper">
    <!-- 多表查询,Select 的as后面的名称必须与UserVO属性名大小写一致 -->
    <select id="selectUserListPage" parameterType="com.example.demo.vo.UserVO"
resultType="com.example.demo.vo.UserVO">
        SELECT
            u.id as id,
            u.username as username,
            u.password as password,
            u.birthday as birthday,
            u.photo as photo,
            u.usertype as usertype,
            u.introduce as introduce
    <!--         r.name as rolename -->
        FROM
            t_sys_user u
    <!--         LEFT JOIN t_sys_role_user ru ON u.id = ru.sys_user_id
        LEFT JOIN t_sys_role r ON ru.sys_role_id = r.id -->
        WHERE 1=1
        <if test="username != null">
            and u.username like concat('',{username},'%')
        </if>
    <!--         <if test="rolename != null">
            and r.name like concat('',{rolename},'%')
        </if> -->

    </select>
    <!-- 以下为示例 -->
    <!--         <select id="listCount" resultType="Integer">
            SELECT COUNT(*) FROM t_sys_user;
        </select>
        <select id="findUserByUsername" parameterType="String" resultType="User">
            SELECT * FROM t_sys_user WHERE username=#{username}
        </select> -->
    </mapper>

```

Xml 文件采用 mybatis 语法格式来定义查询语句，namespace="com.example.demo.mapper.UserMapper"表示关联的 mapper 类，包名必须与自己源码中的包名一致，id="selectUserListPage"、parameterType="com.example.demo.vo.UserVO"、resultType="com.example.demo.vo.UserVO"

分别与 mapper 中自定义的 selectUserListPage 函数名称、参数类型(Pagination 是分页参数)、返回值类型一致, as 语句后面的名称必须与返回值类型 UserVO 的属性名称大小写一致, test="username != null"中的“username”是参数类型 UserVO 中的属性名称,如果“username”值不为空,则按照该值进行模糊查询。其他查询语句的定义可以参考注释部分的代码,也可以访问官方网址: <http://www.mybatis.org/mybatis-3/zh/sqlmap-xml.html>

### 6.1.5. 用户 service 类

在包 “com.example.demo.service” 中新建一个 “UserService” 类,代码如下:

```
package com.example.demo.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.StringUtils;
import com.baomidou.mybatisplus.mapper.EntityWrapper;
import com.baomidou.mybatisplus.plugins.Page;
import com.example.demo.mapper.UserMapper;
import com.example.demo.model.User;
import com.example.demo.util.Convert;
import com.example.demo.util.SnowflakeIdWorker;
import com.example.demo.vo.DataTableResult;
import com.example.demo.vo.UserVO;

@Service
public class UserService {

    @Autowired
    private UserMapper userMapper; //注入mapper进行数据操作

    /**
     * 自定义xml形式的查询
     * @return
     */
    // public Integer listCount() {
    //
    //     return userMapper.listCount();
    //
    // }
}
```

```
    * 自定义xml形式的查询
    * @param username
    * @return
    */
// public User findUserByUsername(String username) {
//     return userMapper.findUserByUsername(username);
// }

//对于执行数据修改的方法加上事务处理
@Transactional
public int delete(String ids) {
    //ids,逗号隔开的主键
    List<String> listid=Convert.toListStrArray(ids);
    return userMapper.deleteBatchIds(listid);
}

public User selectById(String id) {
    //userMapper.selectOne(user),selectOne可以按照其他字段来查询一条记录
    return userMapper.selectById(id);
}

public User selectByUser(User user) {
    return userMapper.selectOne(user);
}

//对于执行数据修改的方法加上事务处理
@Transactional
public int updateById(User user) {
    return userMapper.updateById(user);
}

//对于执行数据修改的方法加上事务处理
@Transactional
public int insert(User user) {
    //添加雪花主键id
    user.setId(SnowflakeIdWorker.getUUID());
    int n = userMapper.insert(user);
    //密码安全一点的话，不能原文保存，应该用MD5，也可以加盐处理
    // n=1/0; //事务测试
    return n;
}
/**
 * 采用集成的查询方法
 * @param username
```

```

    * @return
    */
    public List<User> selectList(String username) {
        EntityWrapper<User> wrapper = new EntityWrapper<User>();
        wrapper.like("username", username);
        return userMapper.selectList(wrapper);
    }

    /**
     * 分页查询
     * @param user
     * @return
     */
    public DataTableResult selectUserListPage(UserVO userVO,int start,int
length,String orderField,String orderDir) {
        Page<UserVO> page = null;
        //排序
        if(!StringUtils.isEmpty(orderDir)&&!StringUtils.isEmpty(orderField)) {
            if(orderDir.equals("asc")) {
                page = new Page<>(start/length + 1, length,orderField,true);// 当前
页, 每页总条数 构造 page 对象
            }else {
                page = new Page<>(start/length + 1, length,orderField,false);// 当
前页, 每页总条数 构造 page 对象
            }

        }else {
            page = new Page<>(start/length + 1, length,"id",false);//默认id降序
        }
        page.setRecords(userMapper.selectUserListPage(page, userVO));

        DataTableResult result = new DataTableResult();
        result.setRecordsTotal(page.getTotal());
        result.setRecordsFiltered(page.getTotal());
        result.setData(page.getRecords());
        return result;
    }
}

```

Service 类实现业务层的代码,是最为复杂的部分,通过调用 mapper 类的方法实现对数据的增删改查操作,基本方法都比较固定,学会套用即可,类中用到的注解如下:

(1) @Service 注解: 如果一个类带了@Service 注解,将自动注册到 Spring 容器,默认名称是类名(头字母小写),可以@Service("xxxx")这样来指定。

(2) @Autowired 注解: @Autowired 表示被修饰的类需要注入对象, spring 会扫描所有被



@Autowired 标注的类，然后根据类型在 ioc 容器中找到匹配的类注入。

(3) @Transactional 注解: @Transactional 可以作用于接口、接口方法、类以及类方法上；当作用于类上时，该类的所有 public 方法将都具有该类型的事务属性，同时，我们也可以方法级别使用该标注来覆盖类级别的定义。

### 6.1.6. 用户 controller 类

在包 “com.example.demo.controller” 中新建一个 “UserController” 类，代码如下：

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

import java.io.File;
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

import com.example.demo.model.User;
import com.example.demo.service.UserService;
import com.example.demo.util.SnowflakeIdWorker;
import com.example.demo.vo.DataTableResult;
import com.example.demo.vo.Json;
import com.example.demo.vo.UserVO;
import com.google.code.kaptcha.Constants;

@Controller
@RequestMapping("/UserController")
public class UserController {

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    private String prefix = "admin/user";// 页面的路径，注意admin前面不要有/

    // 图片存放根路径，从application.yml中读取upload
```

```

@Value("${upload}")
private String UPLOAD_PATH;

@Autowired
private UserService userService; // 注入业务层的service

// 未加入@ResponseBody用来返回数据给页面
@RequestMapping("view")
public String view(Model model) {
    return prefix + "/view";
}

// @ResponseBody, 直接通过js异步返回数据给页面
@RequestMapping("list")
@ResponseBody
public DataTableResult list(HttpServletRequest request, UserVO userVO) {
    // DataTableResult返回给datatables控件的数据格式
    DataTableResult result = new DataTableResult();
    // 获取分页参数
    int start = Integer.parseInt(request.getParameter("start"));
    int length = Integer.parseInt(request.getParameter("length"));
    // 获取排序字段
    String orderIdx = request.getParameter("order[0][column]");
    // 获取排序字段名
    String orderField = request.getParameter("columns[" + orderIdx +
        "][name]");
    // 获取排序方式, 降序desc或者升序asc
    String orderDir = request.getParameter("order[0][dir]");
    // 调用分页查询方法
    result = userService.selectUserListPage(userVO, start, length, orderField,
        orderDir);
    // result.setDraw(userVO.getDraw());
    return result;
}

// @ResponseBody, 直接通过js异步返回数据给页面
@RequestMapping("insert")
@ResponseBody
public Json insert(User user) {
    Json j = new Json();
    if (userService.insert(user) > 0) {
        j.setSuccess(true);
        j.setMsg("添加成功!");
    } else {
        j.setSuccess(false);
    }
}

```

```
        j.setMsg("添加失败! ");
    }
    return j;
}

// @ResponseBody, 直接通过js异步返回数据给页面
@RequestMapping("update")
@ResponseBody
public Json updateById(User user) {
    Json j = new Json();
    if (userService.updateById(user) > 0) {
        j.setSuccess(true);
        j.setMsg("修改成功! ");
    } else {
        j.setSuccess(false);
        j.setMsg("修改失败! ");
    }
    return j;
}

// @ResponseBody, 直接通过js异步返回数据给页面
@RequestMapping("select")
@ResponseBody
public Json selectById(User user) {
    Json j = new Json();
    j.setSuccess(true);
    j.setObj(userService.selectById(user.getId()));
    return j;
}

// @ResponseBody, 直接通过js异步返回数据给页面
@RequestMapping("delete")
@ResponseBody
public Json delete(HttpServletRequest request) {
    Json j = new Json();
    String ids = request.getParameter("ids");
    if (!StringUtils.isEmpty(ids)) {
        j.setSuccess(true);
        j.setObj("成功删除" + userService.delete(ids) + "条记录");
    } else {
        j.setSuccess(false);
        j.setMsg("没有需要删除的记录!");
    }
    return j;
}
```

```

    }

    // @ResponseBody, 直接通过js异步返回数据给页面
    // @RequestParam("file[]") MultipartFile[] file 多个文件
    @RequestMapping("upload")
    @ResponseBody
    public Json upload(HttpServletRequest request, @RequestParam("file")
MultipartFile file) {
        Json j = new Json();
        if (!file.isEmpty()) {
            try {
                String originalFilename = file.getOriginalFilename();
                // 随机文件名
                String newFileName = SnowflakeIdWorker.getUUID()
                    +
originalFilename.substring(originalFilename.lastIndexOf("."));
                // 上传文件路径
                File upload = new File(UPLOAD_PATH, "images/");
                if (!upload.exists())
                    upload.mkdirs();
                String uploadPath = upload + "\\";
                logger.info("uploadPath = " + uploadPath);
                File uploadfile = new File(uploadPath + newFileName);
                // 将上传文件保存到一个目标文件当中
                file.transferTo(uploadfile);
                j.setSuccess(true);
                j.setObj("/upload/images/" + newFileName);
            } catch (IllegalStateException | IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                j.setSuccess(false);
                j.setObj("上传异常");
            }

        } else {
            j.setSuccess(false);
            j.setObj("上传失败");
        }
        return j;
    }
}

```

Controller 类负责数据的传入与传出, 可以做一些数据校验, 但尽量不涉及底层的数据库操作, 应该交给 service 来实现, 相关知识点如下:

(1) `@Controller` 用于标记在一个类上，使用它标记的类就是一个 SpringMVC Controller 对象。

(2) `@RequestMapping` 是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

(3) `@ResponseBody` 这个注解通常使用在控制层（controller）的方法上，其作用是将方法的返回值以特定的格式写入到 response 的 body 区域，进而将数据返回给客户端，例如 JSON 数据，常用在 AJAX 中的异步请求调用。当方法上面没有写 `ResponseBody`，底层会将方法的返回值封装为 `ModelAndView` 对象，返回页面地址。

(4) `@Value("${upload}")` 表示从配置文件 `application.yml` 中读入配置信息，“upload”为配置的属性名称。

(5) `logger` 用来打印日志，在控制台或者 logs 文件夹下面的日志文件中显示信息，常用来调试和跟踪变量的值，要学会使用，例如 `logger.info()` 方法等。

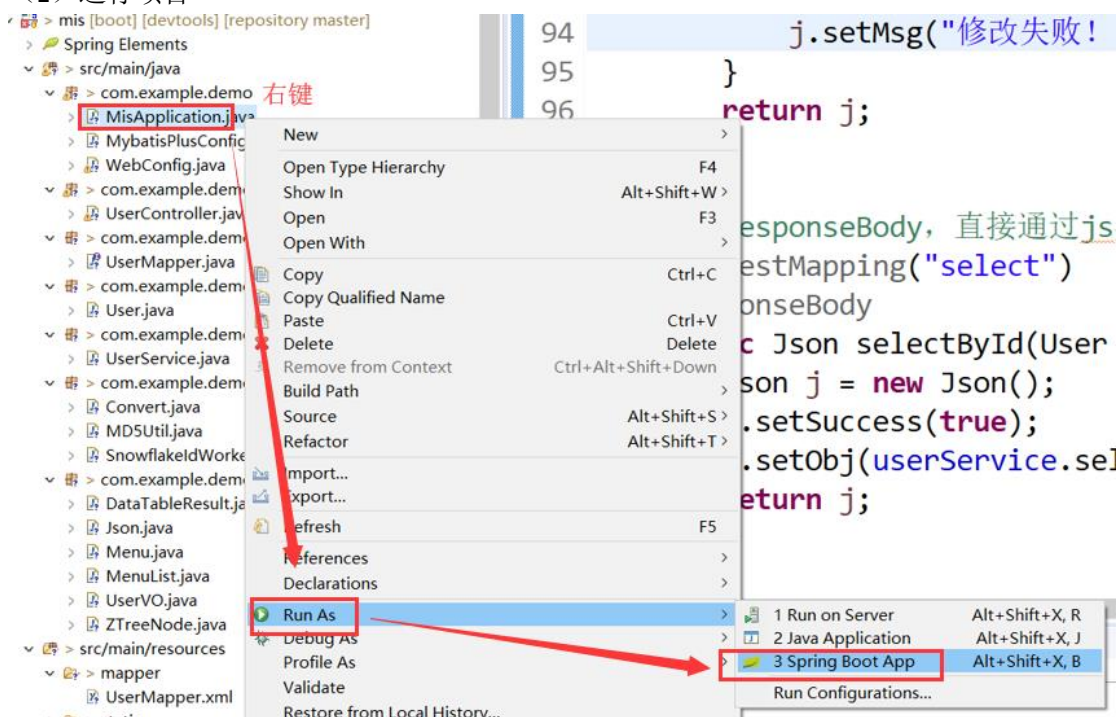
(6) 方法中如何获取页面传递过来的参数：一种是通过定义 `HttpServletRequest` 参数 `request`，来获取参数，该方法可以获取各种参数，但是需要自己一一获取，常用在一些特殊参数值的获取；另一种是通过对象，如 `User` 类型参数 `user`，如果参数名与类中的属性名大小写相同，则直接自动赋值，该方法比较常用。

(7) 返回值：一种是 `String` 类型，一般是用来返回页面地址；另一种是对象类型，如 `Json`，是自定义的数据类型，用来封装异步返回的数据。

(8) 文件上传：利用了 `MultipartFile` 类型的参数 `file` 获取文件对象，通过 `file.transferTo()` 方法保存到配置文件的上传路径中

简单测试一下接口：

(1) 运行项目





```

:: Spring Boot ::                (v2.1.6.RELEASE)

2019-08-04 23:25:54,956 INFO (StartupInfoLogger.java:50)- Starting MisApplication on DESKTOP-8330D8S
2019-08-04 23:25:54,957 INFO (SpringApplication.java:646)- No active profile set, falling back to de
2019-08-04 23:25:54,983 INFO (DeferredLog.java:225)- Devtools property defaults active! Set 'spring.
2019-08-04 23:25:54,984 INFO (DeferredLog.java:225)- For additional web related logging consider set
2019-08-04 23:25:55,733 INFO (TomcatWebServer.java:90)- Tomcat initialized with port(s): 8080 (http)
  
```

## (2) 测试

打开浏览器，建议用 chrome 或者 360 浏览器的极速模式，输入地址 <http://127.0.0.1:8080/UserController/select?id=1>，看到数据就表示访问成功：



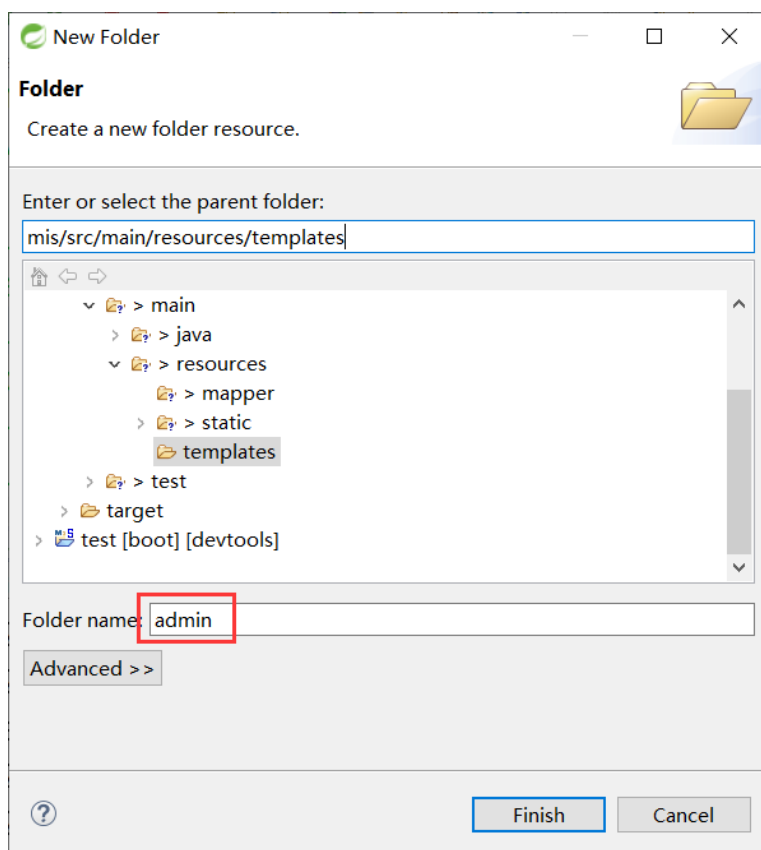
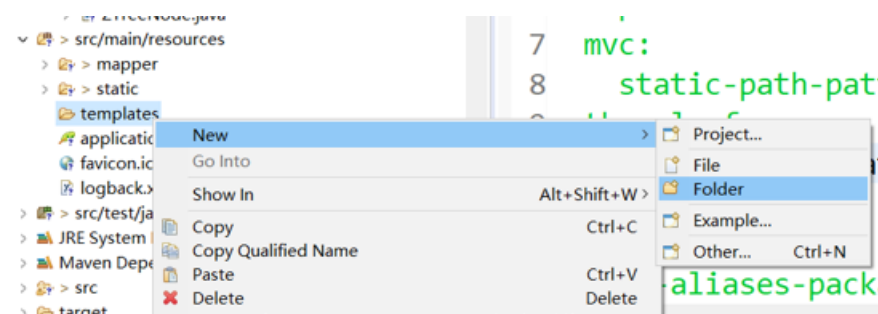
## 6.1.7. 后台管理主界面及首页

后台管理界面采用了开源项目“hAdmin”，GitHub 地址：<https://github.com/huangyaoxin/hAdmin>。hAdmin 是一个免费的后台管理模版，该模版基于 bootstrap（<https://v3.bootcss.com/components/>）与 jQuery（<https://www.runoob.com/jquery/jquery-tutorial.html>）制作，集成了众多常用插件。

Spring Boot 中推荐使用 Thymeleaf 作为模板引擎，因为 Thymeleaf 提供了完美的 Spring MVC 支持。Thymeleaf 是一个跟 Velocity、FreeMarker 类似的模板引擎，它可以完全替代 JSP。官方的使用教程地址：<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>。中文版教程：<https://www.e-learn.cn/thymeleaf>。

后台页面采用 bootstrap 与 jQuery 框架，结合 Thymeleaf 模板引擎实现，模板文件为带 Thymeleaf 标签的 html 文件，统一放在 templates 文件夹里面。

在 templates 文件夹新建一个文件夹“admin”，因为 controller 的转发路径都以 admin 开头。



添加主界面文件，在“admin”文件夹-> 右键->new->file，输入“index.html”，代码如下：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="renderer" content="webkit">
  <title> hAdmin- 主页</title>
  <meta name="keywords" content="">
  <meta name="description" content="">
  <!--[if lt IE 9]>
  <meta http-equiv="refresh" content="0;ie.html" />
  <![endif]-->
  <link rel="shortcut icon" href="favicon.ico">
  <link th:href="@{/static/css/bootstrap.min.css?v=3.3.6}" rel="stylesheet">
```

```

<link th:href="@{/static/css/font-awesome.min.css?v=4.4.0}" rel="stylesheet">
<link th:href="@{/static/css/animate.css}" rel="stylesheet">
<link th:href="@{/static/css/style.css?v=4.1.0}" rel="stylesheet">
</head>
<body class="fixed-sidebar full-height-layout gray-bg" style="overflow:hidden">
  <div id="wrapper">
    <!--左侧导航开始-->
    <nav class="navbar-default navbar-static-side" role="navigation">
      <div class="nav-close"><i class="fa fa-times-circle"></i>
    </div>
    <div class="sidebar-collapse">
      <ul class="nav" id="side-menu">
        <li class="nav-header">
          <div class="dropdown profile-element">
            <a data-toggle="dropdown" class="dropdown-toggle"
href="#">

              <span class="clear">
                <span class="block m-t-xs" style="font-size:20px;">
                  <i class="fa fa-bank"></i>
                  <strong class="font-bold">管理系统</strong>
                </span>
              </span>
            </a>
          </div>
          <div class="logo-element">管理系统
        </div>
      </li>
      <li>
        <a class="J_menuItem" href="/home">
          <i class="fa fa-home"></i>
          <span class="nav-label">主页</span>
        </a>
      </li>
      <li>
        <a href="#">
          <i class="fa fa-desktop"></i>
          <span class="nav-label">系统设置</span>
          <span class="fa arrow"></span>
        </a>
        <ul class="nav nav-second-level">
          <li>
            <a class="J_menuItem" href="/UserController/view">用户
管理</a>
          </li>

```



```

        <li>
            <a class="J_menuItem"
href="/PermissionController/view">权限管理</a>
        </li>
        <li>
            <a class="J_menuItem" href="/RoleController/view">角色
管理</a>
        </li>
    </ul>
</li>

    <li>
        <a href="#">
            <i class="fa fa-file-o"></i>
            <span class="nav-label">测试分类</span>
            <span class="fa arrow"></span>
        </a>
        <ul class="nav nav-second-level">
            <li>
                <a class="J_menuItem" href="graph_echarts.html">测试页
面</a>
            </li>
        </ul>
    </li>
</ul>
</div>
</nav>
<!--左侧导航结束-->
<!--右侧部分开始-->
<div id="page-wrapper" class="gray-bg dashbard-1">
    <div class="row border-bottom">
        <nav class="navbar navbar-static-top" role="navigation"
style="margin-bottom: 0">
            <div class="navbar-header"><a class="navbar-minimalize
minimalize-styl-2 btn btn-info " href="#"><i class="fa fa-bars"></i> </a>
            <form role="search" class="navbar-form-custom" method="post"
action="search_results.html">
                <div class="form-group">
                    <input type="text" placeholder="请输入您需要查找的内
容 ..." class="form-control" name="top-search" id="top-search">
                </div>
            </form>

```

```

        </div>
        <ul class="nav navbar-top-links navbar-right">
            <li>
                <span th:text="{username}"></span><span><a
href="/UserController/logout">注销</a></span>
            </li>
            <li class="dropdown">
                <a class="dropdown-toggle count-info" data-
toggle="dropdown" href="#">
                    <i class="fa fa-envelope"></i> <span class="label
label-warning">16</span>
                </a>
                <ul class="dropdown-menu dropdown-messages">
                    <li class="m-t-xs">
                        <div class="dropdown-messages-box">
                            <a href="profile.html" class="pull-left">
                                
                            </a>
                            <div class="media-body">
                                <small class="pull-right">46小时前</small>
                                <strong>小四</strong> 是不是只有我死了,你们才
不骂爵迹
                                <br>
                                <small class="text-muted">3天前
2014.11.8</small>
                            </div>
                        </div>
                    </li>
                    <li class="divider"></li>
                    <li>
                        <div class="dropdown-messages-box">
                            <a href="profile.html" class="pull-left">
                                
                            </a>
                            <div class="media-body ">
                                <small class="pull-right text-navy">25小时前
</small>
                                <strong>二愣子</strong> 呵呵
                                <br>
                                <small class="text-muted">昨天</small>
                            </div>
                        </div>
                    </li>
                </ul>
            </li>
        </ul>

```

```

        </li>
        <li class="divider"></li>
        <li>
            <div class="text-center link-block">
                <a class="J_menuItem" href="mailbox.html">
                    <i class="fa fa-envelope"></i> <strong> 查看
所有消息</strong>

                </a>
            </div>
        </li>
    </ul>
</li>
<li class="dropdown">
    <a class="dropdown-toggle count-info" data-
toggle="dropdown" href="#">
        <i class="fa fa-bell"></i> <span class="label label-
primary">8</span>

    </a>
    <ul class="dropdown-menu dropdown-alerts">
        <li>
            <a href="mailbox.html">
                <div>
                    <i class="fa fa-envelope fa-fw"></i> 您有16
条未读消息

                    <span class="pull-right text-muted small">4
分钟前</span>

                </div>
            </a>
        </li>
        <li class="divider"></li>
        <li>
            <a href="profile.html">
                <div>
                    <i class="fa fa-qq fa-fw"></i> 3条新回复
                    <span class="pull-right text-muted
small">12分钟前</span>

                </div>
            </a>
        </li>
        <li class="divider"></li>
        <li>
            <div class="text-center link-block">
                <a class="J_menuItem"
href="notifications.html">

```

```

                <strong>查看所有 </strong>
                <i class="fa fa-angle-right"></i>
            </a>
        </div>
    </li>
</ul>
</li>
</ul>
</nav>
</div>
<div class="row J_mainContent" id="content-main">
    <iframe id="J_iframe" width="100%" height="100%" src="/home"
frameborder="0" data-id="/home" seamless></iframe>
</div>
</div>
<!-- 右侧部分结束-->
</div>

<!-- 全局js -->
<script th:src="@{/static/js/jquery.min.js?v=2.1.4}"></script>
<script th:src="@{/static/js/bootstrap.min.js?v=3.3.6}"></script>
<script th:src="@{/static/js/plugins/metisMenu/jquery.metisMenu.js}"></script>
<script
th:src="@{/static/js/plugins/slimscroll/jquery.slimscroll.min.js}"></script>
<script th:src="@{/static/js/plugins/layer/layer.min.js}"></script>

<!-- 自定义js -->
<script th:src="@{/static/js/hAdmin.js?v=4.1.0}"></script>
<script type="text/javascript" th:src="@{/static/js/index.js}"></script>

<!-- 第三方插件 -->
<script th:src="@{/static/js/plugins/pace/pace.min.js}"></script>
</body>
</html>

```

在 html 标签中需要加入“<html xmlns:th="http://www.thymeleaf.org">”表示是 Thymeleaf 模板文件，在页面中以“th:”开头的也都是 Thymeleaf 标签，本项目中只是简单用了一些标签，大部分效果还是通过 jquery 来控制。

添加首页文件，在“admin”文件夹-> 右键->new->file，输入“home.html”，代码如下：

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="renderer" content="webkit">
<title>主页</title>
<meta name="keywords" content="">
<meta name="description" content="">
<!--[if lt IE 9]>
<meta http-equiv="refresh" content="0;ie.html" />
<![endif]-->
<link rel="shortcut icon" href="favicon.ico">
<link th:href="@{/static/css/bootstrap.min.css?v=3.3.6}" rel="stylesheet">
<link th:href="@{/static/css/font-awesome.min.css?v=4.4.0}" rel="stylesheet">
<link th:href="@{/static/css/animate.css}" rel="stylesheet">
<link th:href="@{/static/css/style.css?v=4.1.0}" rel="stylesheet">
</head>

<body class="gray-bg">
  <div class="row wrapper border-bottom white-bg page-heading">
    <div class="col-sm-4">
      <h2>主页</h2>
    </div>
  </div>

  <div class="wrapper wrapper-content">
    <div class="row">
      <div class="col-sm-12">
        <div class="middle-box text-center animated fadeInRightBig">
          <h3 class="font-bold">这里是主页</h3>
        </div>
      </div>
    </div>
  </div>

  <!-- 全局js -->
  <script th:src="@{/static/js/jquery.min.js?v=2.1.4}"></script>
  <script th:src="@{/static/js/bootstrap.min.js?v=3.3.6}"></script>
  <script th:src="@{/static/js/plugins/layer/layer.min.js}"></script>
  <!-- 自定义js -->
  <script th:src="@{/static/js/content.js}"></script>
</body>
</html>

```

Home 页面以 iframe 形式嵌入到 index.html 页面中。

模板文件不能直接访问，必须通过后台转发，因此在包“com.example.demo.controller”中新建一个“HomeController”类来转发上面的地址，代码如下：

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

```

import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

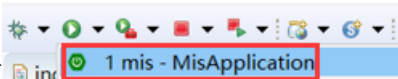
@Controller
public class HomeController {

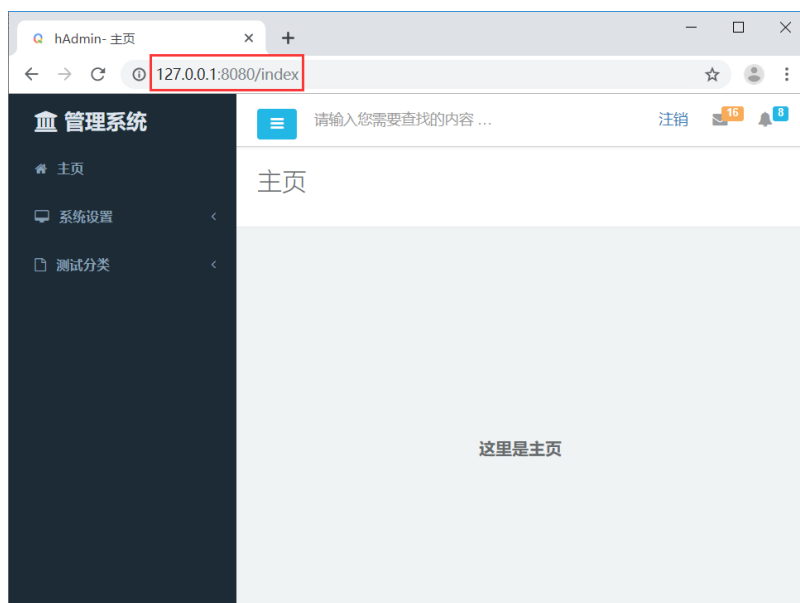
    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    // 未加入@ResponseBody用来返回数据给页面
    @RequestMapping("/index")
    public String index(HttpServletRequest request, Model model) {
        return "admin/index";
    }

    // 未加入@ResponseBody用来返回数据给页面
    @RequestMapping("/home")
    public String home(Model model) {
        return "admin/home";
    }
}

```

确保项目在运行中，如果停止了，通过  启动，打开浏览器，输入地址：<http://127.0.0.1:8080/index>，可以看到以下效果：



url 地址中的 index 是 @RequestMapping("/index") 映射的地址，会按照返回值转化为 templates 路径中的 “admin/index.html”，扩展名可以省略。

## 6.1.8. 用户管理界面

添加用户管理界面，在 “admin” 文件夹中新建一个文件夹 “user”，在 “user” 文件夹 -> 右键->new->file，输入 “view.html”（这里的路径主要是由 controller 的转发方法决定，保持一致即可），代码如下：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="renderer" content="webkit">
<title>用户管理</title>
<meta name="keywords" content="">
<meta name="description" content="">
<!--[if lt IE 9]>
    <meta http-equiv="refresh" content="0;ie.html" />
<![endif]-->
<link rel="shortcut icon" href="favicon.ico">
<link th:href="@{/static/css/bootstrap.min.css?v=3.3.6}"
    rel="stylesheet">
<link th:href="@{/static/css/font-awesome.min.css?v=4.4.0}"
    rel="stylesheet">
<link th:href="@{/static/css/animate.css}" rel="stylesheet">
<link th:href="@{/static/css/style.css?v=4.1.0}" rel="stylesheet">
<!-- Data Tables -->
<link
    th:href="@{/static/css/plugins/dataTables/dataTables.bootstrap.css}"
    rel="stylesheet">
<!-- Ztree -->
<link rel="stylesheet"
th:href="@{/static/css/plugins/ztree/zTreeStyle/zTreeStyle.css}" type="text/css">
<!-- toastr -->
<link th:href="@{/static/css/plugins/toastr/toastr.min.css}" rel="stylesheet">
<!-- summernote -->
<link th:href="@{/static/css/plugins/summernote/summernote.css}" rel="stylesheet">
<link th:href="@{/static/css/plugins/summernote/summernote-bs3.css}"
rel="stylesheet">
<!-- dropzone -->
<link th:href="@{/static/css/plugins/dropzone/basic.css}" rel="stylesheet">
<link th:href="@{/static/css/plugins/dropzone/dropzone.css}" rel="stylesheet">
```

```

</head>

<body class="gray-bg">
  <div class="wrapper wrapper-content animated fadeInRight">
    <div class="row">
      <div class="col-sm-12">
        <div class="ibox float-e-margins">
          <div class="ibox-title">
            <h5>用户管理</h5>
            <div class="ibox-tools">
              <a class="collapse-link">
                <i class="fa fa-chevron-up"></i>
              </a>
              <a class="dropdown-toggle" data-toggle="dropdown"
href="table_data_tables.html#">
                <i class="fa fa-wrench"></i>
              </a>
              <ul class="dropdown-menu dropdown-user">
                <li><a href="table_data_tables.html#">选项1</a>
                </li>
                <li><a href="table_data_tables.html#">选项2</a>
                </li>
              </ul>
              <a class="close-link">
                <i class="fa fa-times"></i>
              </a>
            </div>
          </div>
          <div class="ibox-content">
            <div class="form-horizontal">
              <div class="form-group">
                <div class="col-sm-4">
                  <button id="btn_add" class="btn btn-
primary btn-sm" ><i class="fa fa-plus"></i>&nbsp;添加</button>
                  <button id="btn_del" class="btn btn-
danger btn-sm m-l-sm" ><i class="fa fa-remove"></i>&nbsp;删除</button>
                  <button id="btn_export" class="btn
btn-primary btn-sm m-l-sm"
onclick="$('#table').tableExport({type:'excel',escape:'false',tableName:'导出表格
',ignoreColumn:[0,4]});"><i class="fa fa-file-excel-o"></i>&nbsp;导出</button>
                </div>
                <label class="col-sm-1 control-label">搜
索: </label>

                <div class="col-sm-2">

```



```

        <select id="searchfield" class="form-
control">

        <!-- value为查找字段名称 -->
        <option value="username">用户名
</option>

        </select>

    </div>
    <div class="col-sm-3">
        <input id="keyword" type="search"
class="form-control" placeholder="关键字" />
    </div>
    <div class="col-sm-2">
        <button id="btn_search" class="btn
btn-primary btn-sm m-l-sm" ><i class="fa fa-search"></i>&nbsp;搜索</button>
    </div>

</div>
</div>
<table class="table table-striped table-bordered table-hover"
id="table">
    <thead>
        <tr>
            <th style="padding-left: 10px;">
                <input type="checkbox"
id="cb_selectAll" class="input-lg" style="width:20px;height:20px;" />
            </th>
            <th>id</th>
            <th>用户名</th>
            <th>用户类型</th>
            <th>操作</th>
        </tr>
    </thead>
</table>
</div>
</div>
</div>
</div>

<!-- 模态窗口 -->
<div class="modal fade" data-backdrop="static" id="modal" tabindex="-1"

```

```

role="dialog" aria-hidden="true">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">
          <span aria-hidden="true">&times;</span><span class="sr-
only">Close</span>
        </button>
        <h2 class="modal-title" id="modalttitle"></h2>
      </div>
      <div class="modal-body">
        <div class="row">
          <!-- 表单布局 -->
          <form id="form">
            <div class="col-md-12">
              <div class="form-group">
                <label class="col-sm-3 control-label"><font
color="#FF0000">*</font>用户名称: </label>
                <div class="col-sm-9">
                  <!-- 保存修改的主键值 -->
                  <input type="hidden" name="id" id="id">
                  <input class="form-control" type="text"
name="username" id="username"
                    placeholder="请输入用户名" >
                  <span class="help-block m-b-none"></span>
                </div>
              </div>
            </div>
            <div class="col-md-12">
              <div class="form-group">
                <label class="col-sm-3 control-label"><font
color="#FF0000">*</font>用户密码: </label>
                <div class="col-sm-3">
                  <input class="form-control" type="password"
name="password" id="password"
                    placeholder="请输入用户密码" >
                  <span class="help-block m-b-none"></span>
                </div>
                <label class="col-sm-3 control-label"><font
color="#FF0000">*</font>确认密码: </label>
                <div class="col-sm-3">
                  <input class="form-control" type="password"
name="repassword" id="repassword"
                    placeholder="确认密码" >

```

```

        <span class="help-block m-b-none"></span>
    </div>
</div>
</div>
<div class="col-md-12">
    <div class="form-group">
        <label class="col-sm-3 control-label">用户类
型: </label>

        <div class="col-sm-9">
            <select id="usertype" class="form-
control">

                <option value="普通用户">普通用户
</option>

                <option value="超级管理员">超级管理员
</option>

            </select>
            <span class="help-block m-b-none"></span>
        </div>
    </div>
</div>
<div class="col-md-12">
    <div class="form-group">
        <label class="col-sm-3 control-label">出生日
期: </label>

        <div class="col-sm-9">
            <input class="form-control layer-date"
name="birthday" id="birthday" placeholder="YYYY-MM-DD hh:mm:ss"
onclick="laydate({istime: true, format: 'YYYY-MM-DD hh:mm:ss'})">
            <label class="laydate-icon"></label>
            <span class="help-block m-b-none"></span>
        </div>
    </div>
</div>
<div class="col-md-12">
    <div class="form-group">
        <label class="col-sm-3 control-label">头像:
</label>

        <div class="col-sm-9">
            
        </div>
    </div>
</div>
<div class="col-md-12">

```

```

        <div id="fileupload" class="dropzone">

        </div>
    </div>
    <div class="col-md-12">
        <div class="form-group">
            <label class="col-sm-3 control-label">个人简介: </label>

            <div class="col-sm-9">
                <div class="summernote"
id="introduce">

                </div>

                <span class="help-block m-b-none"></span>
            </div>
        </div>
    </div>
</form>
<!-- 表单布局结束 -->
</div>
</div>
<div class="modal-footer">
    <button type="button" class="btn btn-white"
        data-dismiss="modal">关闭</button>
    <button type="button" class="btn btn-primary" id="btn_save">保存
</button>
</div>
</div>
</div>
</div>
<!-- 模态窗口结束 -->

<!-- 分配角色模态窗口 -->
<div class="modal fade" data-backdrop="static" id="rolemodal" tabindex="-1"
role="dialog" aria-hidden="true">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">
                    <span aria-hidden="true">&times;</span><span class="sr-
only">Close</span>
                </button>

```

```

        <h2 class="modal-title" id="rolemodaltitle">分配角色</h2>
    </div>
    <div class="modal-body">
        <div class="row">
            <!-- 表单布局 -->
            <form id="roleform">
                <div class="col-md-12">
                    <!-- 保存修改的主键值 -->
                    <input type="hidden" name="userid" id="userid">
                    <ul id="tree" class="ztree"></ul>
                </div>
            </form>
            <!-- 表单布局结束 -->
        </div>
    </div>
    <div class="modal-footer">
        <button type="button" class="btn btn-white"
            data-dismiss="modal">关闭</button>
        <button type="button" class="btn btn-primary" id="btn_rolesave">保
存</button>
    </div>
</div>
<!-- 模态窗口结束 -->

<!-- 全局js -->
<script th:src="@{/static/js/jquery.min.js?v=2.1.4}"></script>
<script th:src="@{/static/js/bootstrap.min.js?v=3.3.6}"></script>
<script th:src="@{/static/js/plugins/layer/layer.min.js}"></script>

<!-- 自定义js -->
<script th:src="@{/static/js/content.js}"></script>

<!-- Data Tables -->
<script
th:src="@{/static/js/plugins/dataTables/jquery.dataTables.js}"></script>
<script
    th:src="@{/static/js/plugins/dataTables/dataTables.bootstrap.js}"></script>

<!-- tableexport -->
<!-- 前端直接导出excel有一定的缺陷，使用后台导出功能更强大
    后台导出：阿里巴巴项目组提供了easyexcel工具类，github地址：
https://github.com/alibaba/easyexcel -->

```

```

<script th:src="@{/static/js/plugins/tableexport/tableExport.js}"></script>
<script th:src="@{/static/js/plugins/tableexport/jquery.base64.js}"></script>

<!-- ztree -->
<script th:src="@{/static/js/plugins/ztree/jquery.ztree.core.min.js}"></script>
<script
th:src="@{/static/js/plugins/ztree/jquery.ztree.excheck.min.js}"></script>
<!-- Toastr script -->
<script th:src="@{/static/js/plugins/toastr/toastr.min.js}"></script>
<!-- jQuery Validation plugin javascript-->
<script th:src="@{/static/js/plugins/validate/jquery.validate.min.js}"></script>
<script th:src="@{/static/js/plugins/validate/messages_zh.min.js}"></script>

<!-- layerDate plugin javascript -->
<script th:src="@{/static/js/plugins/layer/laydate/laydate.js}"></script>

<!-- SUMMERNOTE -->
<script th:src="@{/static/js/plugins/summernote/summernote.min.js}"></script>
<script th:src="@{/static/js/plugins/summernote/summernote-zh-CN.js}"></script>

<!-- DROPZONE -->
<script th:src="@{/static/js/plugins/dropzone/dropzone.js}"></script>

<!-- Page-Level Scripts -->
<script>
    //表格行中的按钮点击事件
    function edit(id){
        //异步获取数据
        $.ajax({
            type: "get",
            data: {
                id:id,//第一个id为参数名，第二个为参数值
            },
            url: "/UserController/select",//后台处理地址
            success: function (data) {
                //console.log(data);
                if(data.success){
                    //设置数据
                    $("#id").val(data.obj.id);//修改数据必须有主键值

                    $("#username").val(data.obj.username);
                    $("#password").val(data.obj.password);
                    $("#repassword").val(data.obj.password);
                    $("#usertype").val(data.obj.usertype);
                }
            }
        });
    }

```

```
        $("#birthday").val(data.obj.birthday);
        $('#photo').attr("src",data.obj.photo);
        $('#introduce').summernote("code", data.obj.introduce);
        $("#modaltitle").text("修改用户");
        $("#modal").modal("show");
    }else{
        toastr.error(data.msg, '错误!');
    }
}
}); // end ajax

}

}

//end edit

//表格行中的按钮点击事件
function role(id){
    //异步获取数据
    $.ajax({
        type: "get",
        data: {
            id:id,//第一个id为参数名，第二个为参数值
        },
        url: "/RoleController/tree",//后台处理地址
        success: function (data) {
            //ztree
            var setting = {
                check: {
                    enable: true
                },
                data: {
                    simpleData: {
                        enable: true
                    }
                }
            };
            $.fn.zTree.init($("#tree"), setting, data);
            $("#userid").val(id);//通过隐藏input保存用户id
            $("#rolemodal").modal("show");
        }
    }); // end ajax

}

$(document).ready(function () {
    //datatable
```

性中

```

var datatable= $('#table').DataTable({
    "processing": true,
    "serverSide": true,
    "ajax": {
        url: "/UserController/list",
        type:"post",
        data: function (d) {
            //把字段名和关键词发送给controller进行查询，自动映射到vo模型的相应属性中
            d['#searchfield'].val()=$('#keyword').val()
        }
    },
    "language": {
        "lengthMenu": "每页 _MENU_ 行",
        "info": "从 _START_ 到 _END_, 共 _TOTAL_ 记录",
        "zeroRecords": "没有找到记录",
        "infoEmpty": "暂无记录",
        "infoFiltered": "(从 _MAX_ 条记录过滤)",
        "paginate": {
            "previous": "上一页",
            "next": "下一页",
        },
        "processing": "正在加载..."
    },
    "autoWidth": false,
    "pageLength": 2,
    "lengthChange": false,
    "searching": false,
    "columns": [

        {
            "data": "id", "orderable": false,
            "render": function ( data, type, full, meta ) {
                return '<input type="checkbox" value="'+data+'"'
class="input-lg" style="width:20px;height:20px;" />';
            }
        },
        {
            "data": "id", "name": "id",
            "data": "username", "name": "username",
            "data": "usertype", "name": "usertype",
            //有排序功能必须指定name为字段名称
            "data": "id", "orderable": false,
            "render": function ( data, type, full, meta ) {
                return '<button id="btn_edit" class="btn btn-

```



```

primary btn-sm" onclick="edit(\''+data+'\')"><i class="fa fa-edit"></i>&nbsp;查看修
改</button> <button id="btn_role" class="btn btn-primary btn-sm"
onclick="role(\''+data+'\')"><i class="fa fa-users"></i>&nbsp;分配角色</button>;
    }
    },
    ],
    "order": [
        [1, 'desc']
    ]//默认排序
});

$("#btn_search").click(function(){
    datatable.ajax.reload();//根据关键词重新加载数据
});

//全选
$("#cb_selectAll").click(function(){
    if ($("#cb_selectAll").get(0).checked) {
        $("#table tbody :checkbox").prop("checked", true);
    }else{
        $("#table tbody :checkbox").prop("checked", false);
    }
});

//toastr选项
toastr.options = {
    "positionClass": "toast-bottom-center",
}

//删除
$("#btn_del").click(function(){
    //获取选中的复选框
    var checkboxlist=$("#table tbody :checked");
    if(checkboxlist.length>0){
        if(!confirm("您确定删除数据吗? "))
        {
            return;
        }
    }else{
        toastr.error("请选择要删除的记录。", '错误!');
        return;
    }
    var ids="";
    $.each(checkboxlist, function(n, cb) {

```

```

        ids+=cb.value+",";
    });
    if(ids.length>0){
        ids=ids.substring(0,ids.length-1);
    }
    //异步删除数据
    $.ajax({
        type: "post",
        data: {
            ids:ids,//第一个ids为参数名,第二个为参数值
        },
        url: "/UserController/delete",//后台处理地址
        success: function (data) {
            if(data.success){
                toastr.success(data.msg, '删除成功! ');
                datatable.ajax.reload(null, false);//刷新当前页
            }else{
                toastr.error(data.msg, '错误!');
            }
        }
    }); // end ajax

});//end btn_del

$("#btn_add").click(function(){
    //清空数据
    $("#username").val("");
    $("#password").val("");
    $("#repassword").val("");
    $("#usertype").val("普通用户");
    $("#birthday").val("");
    $('#photo').attr("src","");
    $('#introduce').summernote("code", "");
    //validator.resetForm();//重置验证
    $("#modaltitle").text("添加用户");
    $("#modal").modal("show");
});//end add

```

//不用改, 以下为修改jQuery Validation插件兼容Bootstrap的方法, 没有直接写在插件中是为了便于插件升级

```

$.validator.setDefaults({
    highlight: function (element) {

```

```

        $(element).closest('.form-group').removeClass('has-
success').addClass('has-error');
    },
    success: function (element) {
        element.closest('.form-group').removeClass('has-
error').addClass('has-success');
    },
    errorElement: "span",
    errorPlacement: function (error, element) {
        if (element.is(":radio") || element.is(":checkbox")) {
            error.appendTo(element.parent().parent().parent());
        } else {
            error.appendTo(element.parent());
        }
    },
    errorClass: "help-block m-b-none",
    validClass: "help-block m-b-none"
});
//end setDefaults

// validate form setting
var icon = "<i class='fa fa-times-circle'></i> ";
validator=$("#form").validate({
    rules: {
        username: {
            required: true,
            minlength: 1
        },
        password: {
            required: true,
            minlength: 1
        },
        repassword: {
            required: true,
            minlength: 1,
            equalTo: "#password"
        },
    },
    messages: {
        username: {
            required: icon + "请输入您的用户名",
            minlength: icon + "用户名必须1个字符以上"
        },
        password: {

```

```

        required: icon + "请输入您的密码",
        minlength: icon + "密码必须1个字符以上"
    },
    repassword: {
        required: icon + "请再次输入密码",
        minlength: icon + "密码必须1个字符以上",
        equalTo: icon + "两次输入的密码不一致"
    },
    }
}); //end validate

$("#btn_save").click(function(){
    if($("#form").valid()){
        //save
        var markupStr = $('#introduce').summernote('code');
        //alert(markupStr);
        if($("#modaltitle").text()=="添加用户"){
            //add
            //异步添加数据
            $.ajax({
                type: "post",
                data: {
                    username:$("#username").val(),
                    password:$("#password").val(),
                    usertype:$("#usertype").val(),
                    birthday:$("#birthday").val(),
                    photo:$('#photo').attr("src"),
                    introduce:$('#introduce').summernote("code"),
                },
                url: "/UserController/insert", //后台处理地址
                success: function (data) {
                    if(data.success){
                        toastr.success(data.msg, '添加成功! ');
                        $("#modal").modal('hide');
                        datatable.ajax.reload(null, false); //刷新当前页
                    }else{
                        toastr.error(data.msg, '错误!');
                    }
                }
            }); // end ajax
        }else{
            //update
            //异步修改数据
            $.ajax({

```

```

        type: "post",
        data: {
            id:$("#id").val(),//主键, 从隐藏input获取到
            username:$("#username").val(),
            password:$("#password").val(),
            usertype:$("#usertype").val(),
            birthday:$("#birthday").val(),
            photo:$('#photo').attr("src"),
            introduce:$('#introduce').summernote("code"),
        },
        url: "/UserController/update",//后台处理地址
        success: function (data) {
            if(data.success){
                toastr.success(data.msg, '修改成功! ');
                $("#modal").modal('hide');
                datatable.ajax.reload(null, false);//刷新当前页
            }else{
                toastr.error(data.msg, '错误!');
            }
        }
    }); // end ajax
}

}

}); //end btn_save

//summernote
$('#introduce').summernote({
    lang: 'zh-CN'
});

//Dropzone
Dropzone.autoDiscover = false;// 禁止对所有元素的自动查找, 由于Dropzone会自动
查找class为dropzone的元素
var myDropzone = new Dropzone("#fileupload", {
    url: "/UserController/upload",
    method:"post", //也可用put
    paramName:"file", //后台接收文件参数名称, 默认为file
    maxFiles:1,//一次性上传的文件数量上限
    maxFileSize: 2, //文件大小, 单位: MB
    acceptedFiles: ".jpg,.gif,.png,.jpeg", //上传的类型
    addRemoveLinks: true,
    dictRemoveFile: "删除",
    dictCancelUpload: "取消",

```

```

dictMaxFilesExceeded: "最多上传一个文件",
dictFallbackMessage: '不好意思, 您的浏览器不支持!', //如果浏览器不支持, 默认消息将被替换为这个文本。默认为 "Your browser does not support drag'n'drop file uploads."。

dictInvalidFileType: '该文件不允许上传', //如果文件类型不匹配时显示的错误消息。

dictResponseError: '上传失败, 请稍后重试', //如果服务器响应是无效的时候显示的错误消息。

autoProcessQueue: true,
//uploadMultiple:true, //允许多个文件上传
clickable: true,
init: function() {
    this.on("addedfile", function(file) {
        //上传文件时触发的事件
        console.log("addedfile");
    });
    this.on("success", function(file, data) {
        //上传成功触发的事件
        console.log("success");
        console.log(data);
        if (data.success) {
            $('#photo').attr("src", data.obj);
            toastr.success("上传头像成功!", '成功');
            myDropzone.removeFile(file); //上传界面的预览删除
        }
    });
    this.on("error", function(file, data) {
        //上传失败触发的事件
        console.log("error");
    });
    this.on("removedfile", function(file) {
        //删除文件时触发的方法
        console.log("removedfile");
    });
},
}); // end Dropzone

$("#btn_rolesave").click(function() {
    var treeObj = $.fn.zTree.getZTreeObj("tree");
    var nodes = treeObj.getCheckedNodes(true);
    ids = "";
    for (i = 0; i < nodes.length; i++) {
        if (i < nodes.length - 1) {

```

```

        ids = ids + nodes[i].id + ",";
    }else{
        ids = ids + nodes[i].id;
    }
}
//console.log(ids);
$.ajax({
    type: "post",
    data: {
        id:$("#userid").val(),
        ids:ids
    },
    url: "/RoleController/role",//后台处理地址
    success: function (data) {
        if(data.success){
            toastr.success("分配角色成功!", '成功');
            $("#rolemodal").modal("hide");
        }
    }
}); // end ajax

});//end rolesave

});//end ready

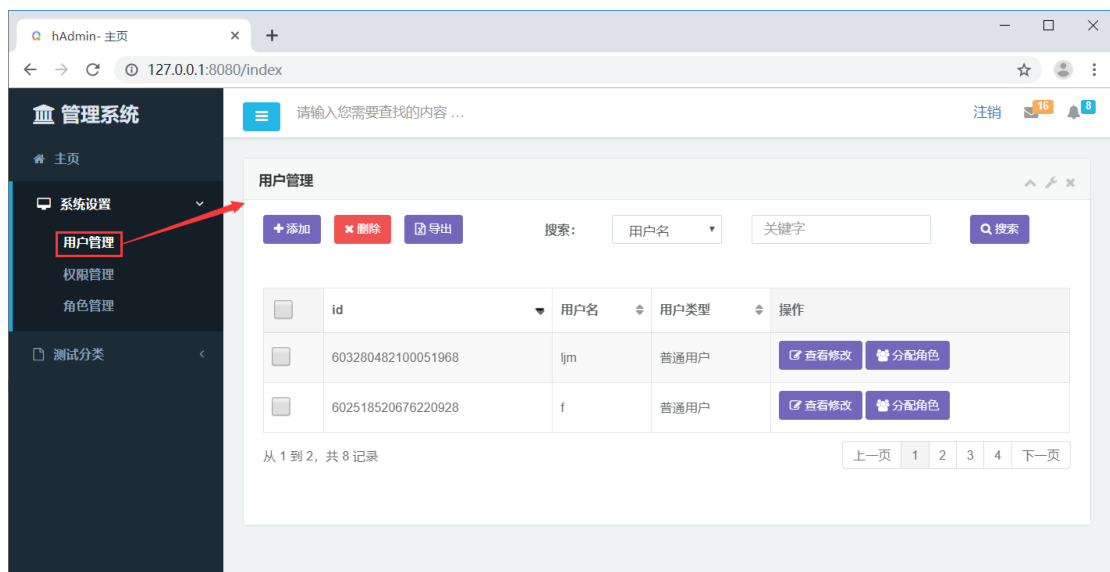
</script>

</body>
</html>

```

数据的增删改查都通过\$.ajax 以 json 格式异步发送到 controller 中进行处理。本页面中还封装了表格、表单验证、在线编辑器、文件上传组件、树形组件（需要调用到角色表，等角色模块完成就可以正常使用）等扩展组件，注意阅读代码中的注释，代码基本以块形式出现，每个扩展都在相应的代码段中。因为涉及的组件较多，重点学会 3 步套用法即可（1、引入 css；2、引入 js；3、编写事件处理代码）。异步调用时，一定要学会用 console.log() 来打印数据进行调试与后台中的 logger.info() 方法类似。

点击用户管理菜单的效果（除了分配角色功能，其他的都已经实现）：



## 6.2. 权限管理模块

### 6.2.1. 权限 model 类



### 6.2.2. 权限 vo 类

### 6.2.3. 权限 mapper 类以及 xml 文件

### 6.2.4. 权限 service 类

### 6.2.5. 权限 controller 类

### 6.2.6. 权限管理界面

## 6.3. 角色管理模块

### 6.3.1. 角色 model 类

### 6.3.2. 角色 vo 类

### 6.3.3. 角色 mapper 类以及 xml 文件

### 6.3.4. 角色 service 类

### 6.3.5. 角色 controller 类

### 6.3.6. 角色管理界面

## 6.4. 登录控制模块

# 7. 项目部署