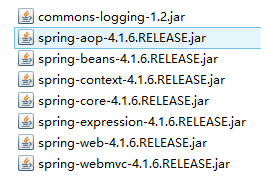
spring mvc学习总结

# 入门程序

## HelloWord程序

1. 首先，导入SpringMVC需要的jar包



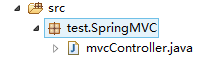
1. 添加Web.xml配置文件中关于SpringMVC的配置

|  |
| --- |
| <!--configure the setting of springmvcDispatcherServlet and configure the mapping-->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc-servlet.xml</param-value>  </init-param>  <!-- <load-on-startup>1</load-on-startup> -->  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping> |

1. 在src下添加springmvc-servlet.xml配置文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">  <!-- scan the package and the sub package -->  <context:component-scan base-package="test.SpringMVC"/>  <!-- don't handle the static resource -->  <mvc:default-servlet-handler />  <!-- if you use annotation you must configure following setting -->  <mvc:annotation-driven />    <!-- configure the InternalResourceViewResolver -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"  id="internalResourceViewResolver">  <!-- 前缀 -->  <property name="prefix" value="/WEB-INF/jsp/" />  <!-- 后缀 -->  <property name="suffix" value=".jsp" />  </bean>  </beans> |

1. 在WEB-INF文件夹下创建名为jsp的文件夹，用来存放jsp视图。创建一个hello.jsp，在body中添加“Hello World”。
2. .建立包及Controller，如下所示



1. 编写Controller代码

|  |
| --- |
| @Controller  @RequestMapping("/mvc")  public class mvcController {  @RequestMapping("/hello")  public String hello(){  return "hello";  }  } |

1. 启动服务器，键入 http://localhost:8080/项目名/mvc/hello

# @Controller

使用@Controller 定义一个 Controller 控制器

|  |
| --- |
| @Controller  public class EmployeeController {  } |

# @RequestMapping

@RequestMapping 除了可以使用请求 URL 映射请求外，还可以使用请求方法、请求参数及请求头映射请求。

## 使用在方法上

|  |
| --- |
| @RequestMapping("/hello")  public String hello(){  return "hello";  }  前台调用：${pageContext.request.contextPath}/hello |

## 使用在类上面

|  |
| --- |
| @RequestMapping("/mvc")  public class mvcController {  @RequestMapping("/hello")  public String hello(){  return "hello";  }  }  前台调用：${pageContext.request.contextPath}/mvc/hello |

# 设置访问方式(RequestMethod)

|  |
| --- |
| @RequestMapping(value = "/emp", method = RequestMethod.POST)  public String hello () {  return "hello";  } |

# @PathVariable

通过 @PathVariable 可以将URL中占位符参数绑定到控制器处理方法的入参中。

|  |
| --- |
| @RequestMapping(value = "/emp/{id}", method = RequestMethod.GET)  public String input(@PathVariable("id") Integer id, Map<String, Object> map) {  map.put("employee", employeeDao.get(id));  map.put("departments", departmentDao.getDepartments());  return "input";  } |

# @RequestParam

在处理方法入参处使用@RequestParam可以把请求参数传递给请求方法

|  |
| --- |
| @RequestMapping("/testRequestParm")  Public String testRequestParm(@RequestParam(value="name")String name,@RequestParam(value="age")Integer age){  System.out.println("testRequestParm name:"+name +";age:"+age);  return SUCCESS;  } |

# @RequestHeader

使用 @RequestHeader 绑定请求报头的属性值请求头包含了若干个属性，服务器可据此获知客户端的信息，通过 @RequestHeader 即可将请求头中的属性值绑定到处理方法的入参中。

|  |
| --- |
| @RequestMapping("/testRequestHeader")  public String testRequestHeader(@RequestHeader(value="Accept-Language")String wn){  System.out.println("testRequestHeader,Accept-Language:"+wn);  return SUCCESS;  } |

# @CookieValue

使用 @CookieValue 绑定请求中的 Cookie 值，@CookieValue 可• 让处理方法入参绑定某个 Cookie 值。

|  |
| --- |
| @RequestMapping("/testCookievalue")  public String testCookievalue(@CookieValue("JSESSIONID")String sessionId ){  System.out.println("testCookievalue JSESSIONID:"+sessionId);  return SUCCESS;  } |

# 使用 POJO 对象绑定请求参数值

|  |
| --- |
| **Controller类**  @RequestMapping("/testPojo")  public String testPojo(User user){  System.out.println("testPojo user:"+user);  return SUCCESS;  }  **User类**  public class User {  private String username;  private String password;  private String email;  private Integer age;  private Address address;  public String getUsername() {  return username;  }  **Jsp页面**  <form action="springmvc/testPojo" method="post">  用户名：<input type="text" name="username"><br>  密码：<input type="password" name="password"><br>  邮箱：<input type="text" name="email"><br>  年龄：<input type="text" name="age"><br>  <input type="submit" value="提交">  </form> |

# 使用 Servlet API 作为入参

|  |
| --- |
| @RequestMapping("/testServletApi")  public String testServletApi(HttpServletRequest request,HttpServletResponse response){  System.out.println("testServletApi:"+request+","+response);  return SUCCESS;  } |

MVC 的 Handler 方法可以接受如下ServletAPI 类型的参数

HttpServletRequest •

HttpServletResponse •

HttpSession •

java.security.Principal •

Locale •

InputStream •

OutputStream •

Reader •

Writer

# 处理模型数据

Spring MVC 提供了以下几种途径输出模型数据。

## ModelAndView

处理方法返回值类型为 ModelAndView 时, 方法体即可通过该对象添加模型数据。

|  |
| --- |
| @RequestMapping("/testModelAndView")  private ModelAndView testModelAndView(){  String viewName=SUCCESS;  ModelAndView modelAndView=new ModelAndView(viewName);  modelAndView.addObject("time",new Date().toLocaleString());  return modelAndView;  } |

页面获取：${ time }

## Map 及 Model

入参为org.springframework.ui.Model、org.springframework.ui.

ModelMap 或 java.uti.Map 时，处理方法返回时，Map 中的数据会自动添加到模型中。

|  |
| --- |
| @RequestMapping("/testMap")  public String testMap(Map<String,Object> map){  map.put("names", Arrays.asList("zhangsan","lis","wangwu"));  return SUCCESS;  } |

页面获取：${ names }

## @SessionAttributes

将模型中的某个属性暂存到HttpSession 中，以便多个请求之间可以共享这个属性。

|  |
| --- |
| /\*\*  \* @SessionAttributes:除了可以通过属性名指定需要放到会话中的属性外（实际上使用的是value属性值），  \* 还可以通过模型属性的对象类型指定哪些模型属性需要放到会话中（实际上使用的是types属性值）  \* 只能放在类上面，而不能放到方法上面  \*/  @Controller  @RequestMapping("/springmvc")  @SessionAttributes(value={"user"},types={String.class})  public class SpringmvcTest {  private final static String SUCCESS="success";    @RequestMapping("/testSessionAttributtes")  public String testSessionAttributtes(Map<String,Object>map){  User user=new User("zhangsan","123456","zhangsan@qq.com",25);  map.put("user", user);  map.put("comapny", "huawei");  return SUCCESS;  }  } |

前台获取：${sessionScope.user}

## @ModelAttribute

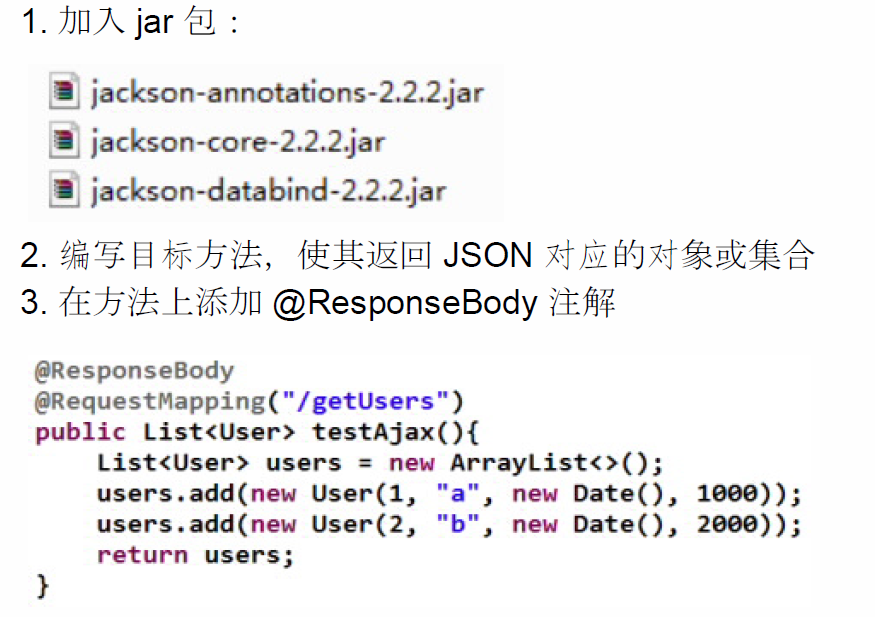
方法入参标注该注解后, 入参的对象就会放到数据模型中

|  |
| --- |
| /\*\*  \* @ModelAttribute标记的方法，会在每个目标方法执行之前被springmvc调用  \*/  @Controller  @RequestMapping("/springmvc")  public class SpringmvcTest {  private final static String SUCCESS="success";    @ModelAttribute  public void getUser(@RequestParam(value="id",required=false)Integer id,Map<String,Object>map){  if(id!=null){  //模拟从数据库获取对象  User user=new User(1,"zhangsan","123456","zhangsan@qq.com",12);  System.out.println("从数据库获取一个对象："+user);  map.put("user", user);  }    }  /\*\*  \* \* 运行流程:  \* 1. 执行 @ModelAttribute 注解修饰的方法: 从数据库中取出对象, 把对象放入到了 Map 中. 键为: user  \* 2. SpringMVC 从 Map 中取出 User 对象, 并把表单的请求参数赋给该 User 对象的对应属性.  \* 3. SpringMVC 把上述对象传入目标方法的参数.  \*/  @RequestMapping("/testModelAttribute")  public String testModelAttribute(User user){  System.out.println("修改："+user);  return SUCCESS;  }  } |

# 重定向

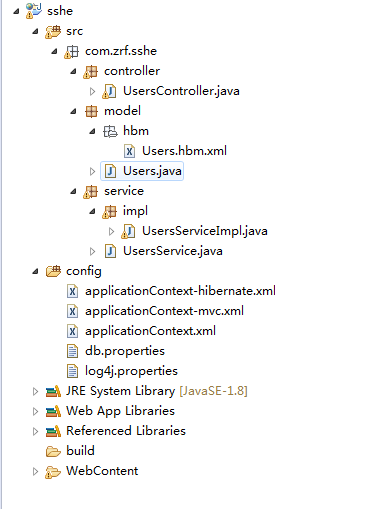
如果返回的字符串中带 forward: 或 redirect: • 前缀时，SpringMVC 会对他们进行特殊处理：将 forward: 和redirect: 当成指示符，其后的字符串作为 URL 来处理redirect:success.jsp：会完成一个到 success.jsp 的重定向的操作 –forward:success.jsp：会完成一个到 success.jsp 的转发操作。

# 返回json数据



# 整合示例

## Springmvc+spring+hibernate整合



### 配置文件

#### web.xml

|  |
| --- |
| <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext\*.xml  </param-value>  </context-param>  <servlet>  <servlet-name>springMvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>springMvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <filter>  <filter-name>SpringOpenSessionInViewFilter</filter-name>  <filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>SpringOpenSessionInViewFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

**注意：获取当前的session要加上OpenSessionInViewFilter**

#### applicationContext-hibernate.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:c="http://www.springframework.org/schema/c" xmlns:cache="http://www.springframework.org/schema/cache"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:jdbc="http://www.springframework.org/schema/jdbc" xmlns:jee="http://www.springframework.org/schema/jee"  xmlns:lang="http://www.springframework.org/schema/lang" xmlns:p="http://www.springframework.org/schema/p"  xmlns:task="http://www.springframework.org/schema/task" xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/cache http://www.springframework.org/schema/cache/spring-cache.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc.xsd  http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee.xsd  http://www.springframework.org/schema/lang http://www.springframework.org/schema/lang/spring-lang.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd">  <!-- 配置数据源 -->  <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">  <property name="url" value="${jdbc\_url}" />  <property name="username" value="${jdbc\_username}" />  <property name="password" value="${jdbc\_password}" />  <!-- 初始化连接大小 -->  <property name="initialSize" value="0" />  <!-- 连接池最大使用连接数量 -->  <property name="maxActive" value="30" />  <!-- 连接池最小空闲 -->  <property name="minIdle" value="0" />  <!-- 获取连接最大等待时间 -->  <property name="maxWait" value="60000" />  <property name="validationQuery" value="${validationQuery}" />  <property name="testOnBorrow" value="false" />  <property name="testOnReturn" value="false" />  <property name="testWhileIdle" value="true" />  <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->  <property name="timeBetweenEvictionRunsMillis" value="60000" />  <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->  <property name="minEvictableIdleTimeMillis" value="25200000" />  <!-- 打开removeAbandoned功能 -->  <property name="removeAbandoned" value="true" />  <!-- 1800秒，也就是30分钟 -->  <property name="removeAbandonedTimeout" value="1800" />  <!-- 关闭abanded连接时输出错误日志 -->  <property name="logAbandoned" value="true" />  <!-- 监控数据库 -->  <!-- <property name="filters" value="mergeStat" /> -->  <property name="filters" value="stat" />  </bean>  <!-- 配置sessionFactory -->  <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">  <property name="dataSource" ref="dataSource" />  <!-- 配置hibernate相关属性 -->  <property name="hibernateProperties">  <props>  <prop key="hibernate.hbm2ddl.auto">${hibernate.hbm2ddl.auto}</prop>  <prop key="hibernate.dialect">${hibernate.dialect}</prop>  <prop key="hibernate.show\_sql">${hibernate.show\_sql}</prop>  <prop key="hibernate.format\_sql">${hibernate.format\_sql}</prop>  <prop key="hibernate.use\_sql\_comments">${hibernate.use\_sql\_comments}</prop>  <prop key="hibernate.connection.release\_mode">${hibernate.connection.release\_mode}</prop>  </props>  </property>  <!-- 自动扫描hbm方式配置的hibernate文件和.hbm文件 -->  <property name="mappingDirectoryLocations">  <list>  <value>classpath:com/zrf/sshe/model/hbm</value>  </list>  </property>  </bean>  <!-- 配置事务管理 -->  <bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">  <property name="sessionFactory" ref="sessionFactory"></property>  </bean>    <!-- 拦截器方式配置事物 -->  <tx:advice id="transactionAdvice" transaction-manager="transactionManager">  <tx:attributes>  <tx:method name="get\*" propagation="REQUIRED" />  <tx:method name="\*\*" propagation="REQUIRED" />  </tx:attributes>  </tx:advice>  <!-- AOP切面配置 -->  <aop:config>  <aop:pointcut id="transactionPointcut" expression="execution(\* com.zrf.sshe.service.impl.\*.\*(..))" />  <aop:advisor pointcut-ref="transactionPointcut" advice-ref="transactionAdvice" />  </aop:config>  </beans> |

#### applicationContext-mvc.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">  <context:component-scan base-package="com.zrf.sshe"></context:component-scan>  <mvc:annotation-driven></mvc:annotation-driven>  <mvc:default-servlet-handler />  <bean id="viewResolver"  class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <!-- 前缀 -->  <property name="prefix" value="/"></property>  <!-- 后缀 -->  <property name="suffix" value=".jsp"></property>  </bean>  </beans> |

#### applicationContext.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd">  <!-- 加载配置文件 -->  <context:property-placeholder location="classpath:db.properties" />    <!-- 自动扫描目录 -->  <context:component-scan base-package="com.zrf.sshe" />    <bean id="objectMapper" class="com.fasterxml.jackson.databind.ObjectMapper" />    <!-- 自动注解支持 -->  <context:annotation-config />    </beans> |

#### db.properties

|  |
| --- |
| hibernate.dialect=org.hibernate.dialect.MySQLDialect  validationQuery=SELECT 1  driverClassName = com.mysql.jdbc.Driver  jdbc\_url = jdbc\:mysql\://localhost\:3306/easyui?useUnicode\=true&characterEncoding\=UTF-8  jdbc\_username = root  jdbc\_password = 000000  hibernate.hbm2ddl.auto=update  hibernate.show\_sql=false  hibernate.format\_sql=false  hibernate.use\_sql\_comments=true  hibernate.connection.release\_mode=on\_close |

### 代码

#### Service

|  |
| --- |
| public interface UsersService {  public List<Users>getUsers();  public void addUser(Users users);  public void deleteUser(Integer id);  public void updateUser(Users users);  public Users getUser(Integer id);  } |

#### Impl

|  |
| --- |
| package com.zrf.sshe.service.impl;  import java.util.List;  import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.zrf.sshe.model.Users;  import com.zrf.sshe.service.UsersService;  @Service  public class UsersServiceImpl implements UsersService {  @Autowired  private SessionFactory sessionFactory;  public Session getSession(){  return sessionFactory.getCurrentSession();  }  @Override  public List<Users> getUsers() {  String hq1="from Users";  return getSession().createQuery(hq1).list();  }  @Override  public void addUser(Users users) {  getSession().save(users);  }  @Override  public void deleteUser(Integer id) {  String hq1="delete from Users where id="+id;  getSession().createQuery(hq1).executeUpdate();  }  @Override  public void updateUser(Users users) {  String hq1="update Users set firstname=?,lastname=?,phone=?,email=? where id=?";  Query query=getSession().createQuery(hq1);  query.setString(0, users.getFirstname());  query.setString(1, users.getLastname());  query.setString(2, users.getPhone());  query.setString(3, users.getEmail());  query.setInteger(4, users.getId());  query.executeUpdate();  }  @Override  public Users getUser(Integer id) {  String hq1="from Users where id="+id;  Query query=getSession().createQuery(hq1);  Users users=(Users) query.uniqueResult();  return users;  }  } |

#### Model

|  |
| --- |
| public class Users {  private Integer id;  private String firstname;  private String lastname;  private String phone;  private String email;  public Users() {  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getFirstname() {  return firstname;  }  public void setFirstname(String firstname) {  this.firstname = firstname;  }  public String getLastname() {  return lastname;  }  public void setLastname(String lastname) {  this.lastname = lastname;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }      } |

对象映射文件

|  |
| --- |
| <?xml version="1.0" ?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <hibernate-mapping package="com.zrf.sshe.model" default-access="field">  <class name="Users" table="Users">  <id name="id" type="java.lang.Integer">  <column name="ID" />  <generator class="native" />  </id>  <property name="firstname" column="firstname" length="50" type="java.lang.String"/>  <property name="lastname" column="lastname" length="50" type="java.lang.String"/>  <property name="phone" column="phone" length="200" type="java.lang.String"/>  <property name="email" column="email" length="200" type="java.lang.String"/>  </class>  </hibernate-mapping> |

#### Controller

|  |
| --- |
| package com.zrf.sshe.controller;  import java.util.List;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.ResponseBody;  import com.zrf.sshe.model.Users;  import com.zrf.sshe.service.UsersService;  @Controller  @RequestMapping("/users")  public class UsersController {  @Autowired  private UsersService usersService;  @RequestMapping("/getUsers")  @ResponseBody  public List<Users>getUsers(){  return usersService.getUsers();  }  @RequestMapping("/deleteUsers")  public String deleteUsers(Integer id,HttpServletResponse response){  usersService.deleteUser(id);  return "redirect:/users/getUsers";  }  @RequestMapping("/addUser")  public String addUser(Users users){  usersService.addUser(users);  return "redirect:/users/getUsers";  }  @RequestMapping("/getUser")  public String getUser(Integer id,HttpServletRequest request){  Users users = usersService.getUser(id);  request.setAttribute("users", users);  return "/editUser";  }  @RequestMapping("/updateUsers")  public String updateUsers(Users users){  usersService.updateUser(users);  return "redirect:/users/getUsers";  }  @RequestMapping("/toAddUser")  public String toAddUser(){  return "/addUser";  }  } |

#### 页面

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Insert title here</title>  <link rel="stylesheet" type="text/css"  href="easyui/themes/default/easyui.css">  <link rel="stylesheet" type="text/css" href="easyui/themes/icon.css" />  <script type="text/javascript" src="easyui/js/jquery.min.js"></script>  <script type="text/javascript" src="easyui/js/jquery.easyui.min.js"></script>  </head>  <body>  <table id="dg"></table>  <div id="dlg" class="easyui-dialog"  style="width: 500px; height: 300px; padding: 10px 20px;" closed="true"  buttons="#dlg-buttons">  <form id="fm" method="post">  <div class="fitem">  首次登陆名<input id="firstname" name="firstname"  class="easyui-validatebox" />  </div>  <div class="fitem">  最后登录名<input id="lastname" name="lastname" class="easyui-validatebox" />  </div>  <div class="fitem">  手机&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input  id="phone" name="phone" value="${users.phone}"  class="easyui-validatebox" />  </div>  <div class="fitem">  邮箱&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input  id="email" name="email" value="${users.email}"  class="easyui-vlidatebox" />  </div>  </form>  </div>  <div id="dlg-buttons">  <a href="javascript:void(0)" class="easyui-linkbutton"  onclick="save()" iconcls="icon-save">保存</a> <a  href="javascript:void(0)" class="easyui-linkbutton"  onclick="javascript:$('#dlg').dialog('close')" iconcls="icon-cancel">取消</a>  </div>  <div id="dllg" class="easyui-dialog"  style="width: 500px; height: 300px; padding: 10px 20px;" closed="true"  buttons="#dlg-buttons">  <form id="fmm" method="post">  <div class="fitem">  首次登陆名<input id="firstname" name="firstname"  class="easyui-validatebox" />  </div>  <div class="fitem">  最后登录名<input id="lastname" name="lastname" class="easyui-validatebox" />  </div>  <div class="fitem">  手机&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input  id="phone" name="phone" value="${users.phone}"  class="easyui-validatebox" />  </div>  <div class="fitem">  邮箱&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input  id="email" name="email" value="${users.email}"  class="easyui-vlidatebox" />  </div>  <input type="hidden" name="id" id="id" />  </form>  </div>  <div id="dlg-buttons">  <a href="javascript:void(0)" class="easyui-linkbutton"  onclick="saveuser()" iconcls="icon-save">保存</a> <a  href="javascript:void(0)" class="easyui-linkbutton"  onclick="javascript:$('#dllg').dialog('close')" iconcls="icon-cancel">取消</a>  </div>  </body>  <script type="text/javascript">  $(function() {  $('#dg').datagrid({  url : "${pageContext.request.contextPath}/users/getUsers",  nowrap : true,  fit : true,  fitColumns : false,  border : false,  rownumbers : true,  idField : 'id',  remoteSort : false,  singleSelect : true,  checkOnSelect : true,  selectOnCheck : true,  showPageList : false,  columns : [ [ {  field : 'id',  title : '编号',  width : 100  }, {  field : 'firstname',  title : '首次名',  width : 100  }, {  field : 'lastname',  title : '最后名',  width : 100  }, {  field : 'phone',  title : '手机',  width : 100  }, {  field : 'email',  title : '邮箱',  width : 150  } ] ],  toolbar : [ {  text : '新增',  iconCls : 'icon-add',  handler : function() {  add();  }  }, '-', {  text : '修改',  iconCls : 'icon-edit',  handler : function() {  update();  }  }, '-', {  text : '删除',  iconCls : 'icon-remove',  handler : function() {  destroyUser();  }  } ]  });  });  function add() {  $("#dlg").dialog("open").dialog('setTitle', '新增用户');  $("#fm").form("clear");  }  function save() {  $("#fm").form("submit", {  url : "${pageContext.request.contextPath}/users/addUser",  onsubmit : function() {  return $(this).form("validate");  },  success : function(result) {  if (result) {  $.messager.alert("提示信息", "操作成功");  $("#dlg").dialog("close");  $("#dg").datagrid("load");  } else {  $.messager.alert("提示信息", "操作失败");  }  }  });  }  function saveuser() {  $("#fmm")  .form(  "submit",  {  url : "${pageContext.request.contextPath}/users/updateUsers",  onsubmit : function() {  return $(this).form("validate");  },  success : function(result) {  if (result) {  $.messager.alert("提示信息", "修改成功");  $("#dllg").dialog("close");  $("#dg").datagrid("load");  } else {  $.messager.alert("提示信息", "操作失败");  }  }  });  }  function update() {  var row = $('#dg').datagrid('getSelected');  if (row) {  $("#dllg").dialog("open").dialog('setTitle', '更新用户');  $("#fmm").form("clear");  $("#fmm").form("load", row);  url = "${pageContext.request.contextPath}/users/getUser?id="  + row.id;  }  }  function destroyUser() {  var row = $('#dg').datagrid('getSelected');  if (row) {  $.messager.confirm('Confirm', '您确认删除吗?', function(r) {  if (r) {  $.post(  '${pageContext.request.contextPath}/users/deleteUsers?id='+row.id,function(json) {  $.messager.alert("提示信息", "删除成功");  $("#dlg").dialog("close");  $("#dg").datagrid("load");  });  }  });  } else {  $.messager.alert("提示信息", "请选择要删除的数据");  }  }  </script>  </html> |

# 示例代码

