Day02笔记

知识点：

1. SpringMVC
2. LayUI前端框架

Spring依赖注入：能够保证组件的装配，降低组件和组件之间的耦合性。

SpringMVC的作用：作用点在web端（接受请求和响应信息），是可以替代servlet的一种解决方案。优点;1交互上采用接口方式（restful，json）,

优点2，请求接收信息以及响应更为方便

**SpringMVC应用 （Controller）**

**案例1：SpringMVC的初体验案例**

**package** com.dfrz.edusys3.controller;  
  
**import** com.dfrz.edusys3.bean.User;  
**import** com.dfrz.edusys3.utils.Result;  
**import** com.dfrz.edusys3.utils.ResultUtils;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
*/\*  
\* @RestController=@Controller+@ResponseBody  
\* @Controller 用于表示控制器类  
\* @ResponseBody 返回的对象结果会被转为Json格式数据  
\* \*/*@RestController  
**public class** UserController {  
 */\* @RequestMapping 映射路径\*/* @RequestMapping(**"/user/login"**)  
 **public** Result login(String uname, String upass){  
 System.***out***.println(**"【用户名】"**+uname+**",【密码】"**+upass);  
 User user=**new** User();  
 user.setUname(uname);  
 user.setUname(upass);  
 **return** ResultUtils.*success*(user);  
 }  
}

自定义Result类

**package** com.dfrz.edusys3.utils;  
  
*/\*\*  
 \* 自定义的结果类，用于控制器进行返回结果  
 \** ***@param <T>*** *\*/***public class** Result<T> {  
 **private** Integer **code**;  
 **private** String **msg**;  
 **private** T **data**;  
 **private** Integer **count**;  
  
 **public** Result() {  
 **super**();  
 }  
  
 **public** Result(Integer code, String msg, T data) {  
 **this**.**code** = code;  
 **this**.**msg** = msg;  
 **this**.**data** = data;  
 }  
  
 **public** Integer getCode() {  
 **return code**;  
 }  
  
 **public void** setCode(Integer code) {  
 **this**.**code** = code;  
 }  
  
 **public** String getMsg() {  
 **return msg**;  
 }  
  
 **public void** setMsg(String msg) {  
 **this**.**msg** = msg;  
 }  
  
 **public** T getData() {  
 **return data**;  
 }  
  
 **public void** setData(T data) {  
 **this**.**data** = data;  
 }  
  
 **public** Integer getCount() {  
 **return count**;  
 }  
  
 **public void** setCount(Integer count) {  
 **this**.**count** = count;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Result{"** +  
 **"code="** + **code** +  
 **", msg='"** + **msg** + **'\''** +  
 **", data="** + **data** +  
 **'}'**;  
 }  
}

**package** com.dfrz.edusys3.utils;  
  
**public enum** ResultEnum {  
 *//这里是可以自己定义的，方便与前端交互即可* ***UNKNOWN\_ERROR***(-1,**"未知错误"**),  
 ***SUCCESS***(10000,**"成功"**),  
 ***USER\_NOT\_EXIST***(1,**"用户不存在"**),  
 ***USER\_IS\_EXISTS***(2,**"用户已存在"**),  
 ***DATA\_IS\_NULL***(3,**"数据为空"**),  
 ***LOGIN\_FAILS***(4,**"登陆失败"**),  
 ;  
 **private** Integer **code**;  
 **private** String **msg**;  
   
 ResultEnum(Integer code, String msg) {  
 **this**.**code** = code;  
 **this**.**msg** = msg;  
 }  
   
 **public** Integer getCode() {  
 **return code**;  
 }  
   
 **public** String getMsg() {  
 **return msg**;  
 }  
}

**package** com.dfrz.edusys3.utils;  
  
**public class** ResultUtils {  
 */\*\*  
 \* 返回带数据的信息  
 \** ***@param obj*** *\** ***@return*** *\*/* **public static** Result success(Object obj){  
 Result result=**new** Result();  
 result.setCode(ResultEnum.***SUCCESS***.getCode());  
 result.setMsg(ResultEnum.***SUCCESS***.getMsg());  
 result.setData(obj);  
 **return** result;  
 }  
  
 */\*\*  
 \* 返回成功但不带数据  
 \** ***@return*** *\*/* **public static** Result success(){  
 **return** *success*(**null**);  
 }  
  
 */\*\*  
 \* 错误信息  
 \** ***@param code*** *\** ***@param msg*** *\** ***@return*** *\*/* **public static** Result error(Integer code,String msg){  
 Result result=**new** Result();  
 result.setCode(code);  
 result.setMsg(msg);  
 **return** result;  
 }  
}

SpringMVC对接模板

模板：JSP、freemarker、thymeleaf

页面= 模板+数据

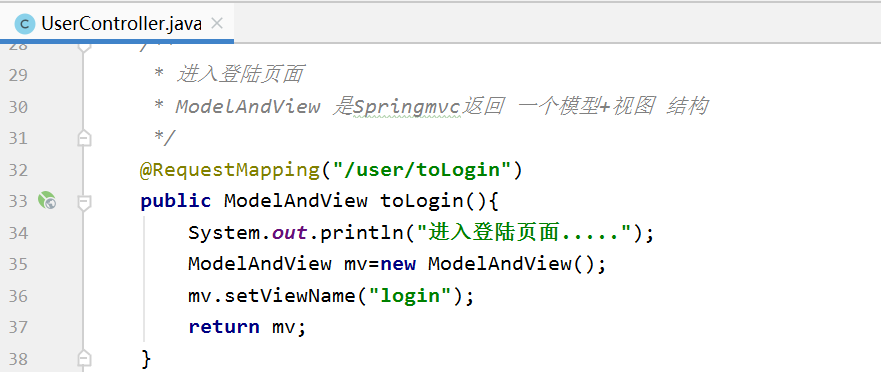
Springboot配置文件 application.properties / application.yml

**server**:  
 **port**: 8083  
**spring**:  
 **thymeleaf**:  
 **enabled**: true *#开启thymeleaf视图解析* **encoding**: utf-8 *#编码* **prefix**: classpath:/templates/ *#前缀* **cache**: false *#是否使用缓存* **mode**: HTML *#严格的HTML语法模式* **suffix**: .html *#后缀名*

Pom.xml 增加thymeleaf的依赖

*<!--引入thymeleaf依赖-->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
</**dependency**>

UserController中添加跳转页面方法toLogin()



模板页面 login.html

<!DOCTYPE **html**>  
<**html lang="en" xmlns:th="http://www.thymeleaf.org"**>  
  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>欢迎页面-DFRZ</**title**>  
 <**meta name="renderer" content="webkit"**>  
 <**meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"**>  
 <**meta name="viewport" content="width=device-width,user-scalable=yes, minimum-scale=0.4, initial-scale=0.8,target-densitydpi=low-dpi"** />  
 <**link rel="shortcut icon" href="/favicon.ico" type="image/x-icon"** />  
 <**link rel="stylesheet" th:href="@{/css/font.css}"**>  
 <**link rel="stylesheet" th:href="@{/css/xadmin.css}"**>  
 <**script type="text/javascript" th:src="@{/plugins/jquery/jquery.min.js}"** ></**script**>  
 <**script th:src="@{/lib/layui/layui.js}" charset="utf-8"**></**script**>  
 <**script type="text/javascript" th:src="@{/js/xadmin.js}"**></**script**>  
 *<!-- 让IE8/9支持媒体查询，从而兼容栅格 -->  
 <!--[if lt IE 9]>  
 <script src="https://cdn.staticfile.org/html5shiv/r29/html5.min.js"></script>  
 <script src="https://cdn.staticfile.org/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->*</**head**>  
  
</**head**>  
<**body class="login-bg"**>  
   
 <**div class="login layui-anim layui-anim-up"**>  
 <**div class="message"**>后台系统-管理登录</**div**>  
 <**div id="darkbannerwrap"**></**div**>  
   
 <**form method="post" class="layui-form"** >  
 <**input name="username" placeholder="用户名" type="text" lay-verify="required" class="layui-input"** >  
 <**hr class="hr15"**>  
 <**input name="password" lay-verify="required" placeholder="密码" type="password" class="layui-input"**>  
 <**hr class="hr15"**>  
 <**input value="登录" lay-submit lay-filter="login" style="width**:100%;**" type="submit"**>  
 <**hr class="hr20"** >  
 </**form**>  
 </**div**>  
  
 <**script**>  
 **$**(**function** () {  
 **layui**.use(**'form'**, **function**(){  
 **var** form = **layui**.**form**;  
 *// layer.msg('玩命卖萌中', function(){  
 // //关闭后的操作  
 // });  
 //监听提交* form.on(**'submit(login)'**, **function**(data){  
 *// alert(888)* layer.msg(***JSON***.stringify(data.field),**function**(){  
 ***location***.**href**=**'index.html'** });  
 **return false**;  
 });  
 });  
 })  
  
   
 </**script**>  
  
   
 *<!-- 底部结束 -->*</**body**>  
</**html**>

用户登陆案例： springmvc+spring+layui+thymeleaf

1. login.html中增加登陆的业务代码

<**script**>  
 **$**(**function** () {  
 **layui**.use(**'form'**, **function**(){  
 **var** form = **layui**.**form**; *//监听提交* form.on(**'submit(login)'**, **function**(data){  
  */\*  
 知识点：Ajax是客户端通过异步通信的方式访问服务端  
 JQuery Ajax访问服务端的示例  
 url：访问地址  
 data：请求数据  
 dataType:期待返回值是JSON数据  
 success：成功回调函数  
 \*/* **$**.ajax({  
 **url**:**"/user/doLogin"**,  
 **data**:data.field,  
 **dataType**:**"json"**,  
 success:**function** (data) {  
 **if**(data.**code**==10000){  
 layer.msg(data.msg,**function**(){  
 *//前端跳转页面* ***location***.**href**=**'index.html'** });  
 }  
 **else**{  
 layer.msg(data.msg);  
 }  
 }  
 });  
 **return false**;  
 });  
 });  
 })  
  
   
</**script**>

1. 更新UserController的login方法

@RestController  
**public class** UserController {  
 @Autowired  
 IUserService **userService**;  
 */\* @RequestMapping 映射路径\*/* @RequestMapping(**"/user/doLogin"**)  
 **public** Result login(String uname, String upass){  
 Result result=**null**;  
 System.***out***.println(**"执行登陆....."**);  
 System.***out***.println(**"【用户名】"**+uname+**",【密码】"**+upass);  
 User user=**new** User();  
 user.setUname(uname);  
 user.setUpass(upass);  
 User loginUser=**userService**.login(user);  
 **if**(loginUser!=**null**){  
 result=ResultUtils.*success*(user);  
 }  
 **else**{  
 result=ResultUtils.*success*(user);  
 result.setCode(ResultEnum.***LOGIN\_FAILS***.getCode());  
 result.setMsg(ResultEnum.***LOGIN\_FAILS***.getMsg());  
 }  
 **return** result;  
  
 }

注意点1：在@RestController 下

ModelAndView返回值 是进行跳转页面

对象返回值 是返回对象转化的JSON数据