Web应用开发实验（四）

姓名：叶倩琳

学号：201706061330

完成日期：2019/5/19

# 实验目的

1. 掌握jsp脚本元素的使用。
2. 掌握静态包含的原理及include指令的使用方法。
3. 掌握动态包含的原理及<jsp:include>动作的使用方法。
4. 掌握请求转发的原理及<jsp:forward>动作的使用方法。
5. 掌握javabean的编写规范和jsp标准动作；
6. 掌握MVC设计模式的体系结构。

# 实验内容

1. 配置运行书本图4-4（P97）的main.jsp（PPT课件第87张）：
   1. 进入Tomcat主目录下work目录查看其最终生成的main\_jsp.java文件的输出内容；
   2. 将main.jsp包含other.jsp的方式改为使用<jsp:include>的动态包含（图4-6，P99），进入Tomcat主目录下work目录查看其最终生成的main\_jsp.java文件的输出内容以及other\_jsp.java的输出内容，并比较静态包含和动态包含的区别。
2. 源代码
3. 静态包含

<%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*

pageEncoding=*"utf-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"utf-8"*>

<title>main</title>

</head>

<body>

<b>Weclome</b>

<%@include file=*"other01.jsp"* %>

<b>Good Bye</b>

</body>

</html>

Other01.jsp:

<pre>

Nothing is impossible!

</pre>

1. 动态包含

<%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*

pageEncoding=*"utf-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"utf-8"*>

<title>main02</title>

</head>

<body>

<br>Hello,World!</br>

<jsp:include page=*"other02.jsp"* />

<b>Good Bye</b>

</body>

</html>

Other02.jsp:

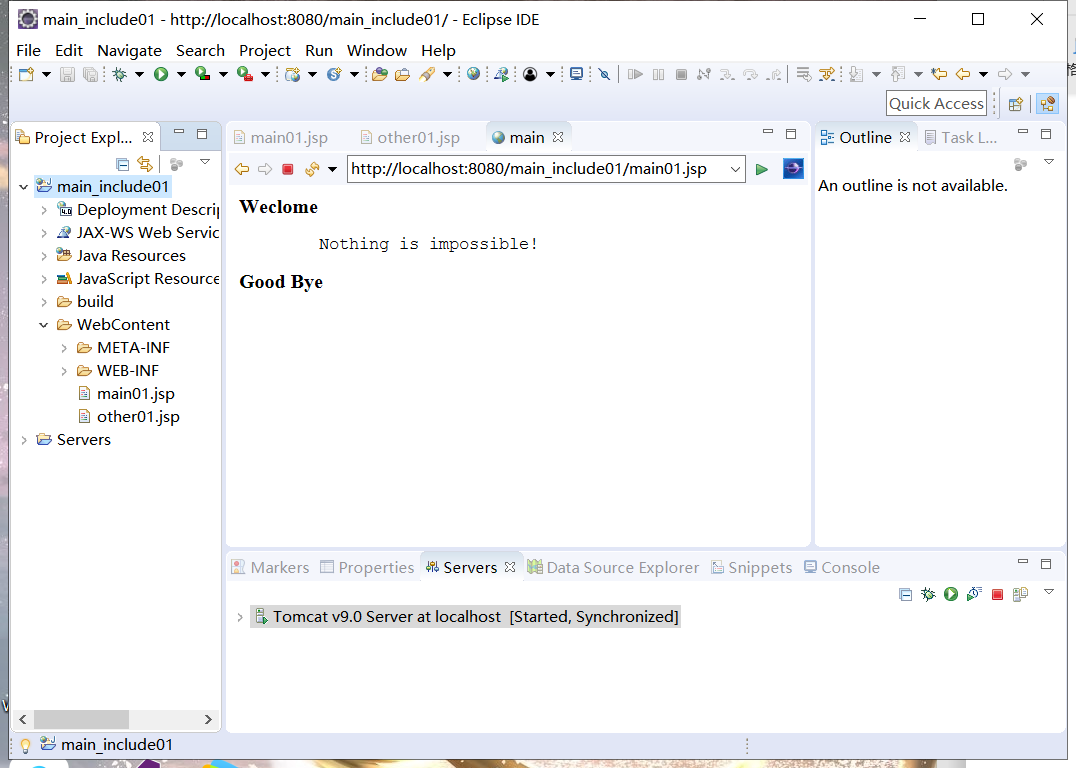
<pre>

Nothing is impossible!

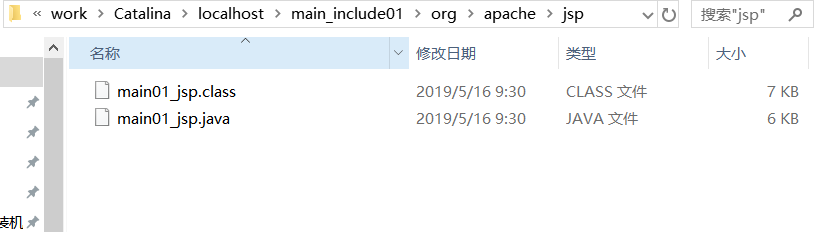
</pre>

1. 实验截屏

1)静态包含



因为我是在eclipse中配置运行的，所以生成的\_jsp.clss和\_jsp.java文件在C:\Users\hp\Desktop\实验\Web应用实验\实验四\main\_include01\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\main\_include01\org\apache\jsp目录下



其中，main01\_jsp.java中的内容如下：

/\*

\* Generated by the Jasper component of Apache Tomcat

\* Version: Apache Tomcat/9.0.14

\* Generated at: 2019-05-16 01:33:24 UTC

\* Note: The last modified time of this file was set to

\* the last modified time of the source file after

\* generation to assist with modification tracking.

\*/

package org.apache.jsp;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.jsp.\*;

public final class main01\_jsp extends org.apache.jasper.runtime.HttpJspBase

implements org.apache.jasper.runtime.JspSourceDependent,

org.apache.jasper.runtime.JspSourceImports {

private static final javax.servlet.jsp.JspFactory \_jspxFactory =

javax.servlet.jsp.JspFactory.getDefaultFactory();

private static java.util.Map<java.lang.String,java.lang.Long> \_jspx\_dependants;

static {

\_jspx\_dependants = new java.util.HashMap<java.lang.String,java.lang.Long>(1);

\_jspx\_dependants.put("/other01.jsp", Long.valueOf(1557970345214L));

}

private static final java.util.Set<java.lang.String> \_jspx\_imports\_packages;

private static final java.util.Set<java.lang.String> \_jspx\_imports\_classes;

static {

\_jspx\_imports\_packages = new java.util.HashSet<>();

\_jspx\_imports\_packages.add("javax.servlet");

\_jspx\_imports\_packages.add("javax.servlet.http");

\_jspx\_imports\_packages.add("javax.servlet.jsp");

\_jspx\_imports\_classes = null;

}

private volatile javax.el.ExpressionFactory \_el\_expressionfactory;

private volatile org.apache.tomcat.InstanceManager \_jsp\_instancemanager;

public java.util.Map<java.lang.String,java.lang.Long> getDependants() {

return \_jspx\_dependants;

}

public java.util.Set<java.lang.String> getPackageImports() {

return \_jspx\_imports\_packages;

}

public java.util.Set<java.lang.String> getClassImports() {

return \_jspx\_imports\_classes;

}

public javax.el.ExpressionFactory \_jsp\_getExpressionFactory() {

if (\_el\_expressionfactory == null) {

synchronized (this) {

if (\_el\_expressionfactory == null) {

\_el\_expressionfactory = \_jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory();

}

}

}

return \_el\_expressionfactory;

}

public org.apache.tomcat.InstanceManager \_jsp\_getInstanceManager() {

if (\_jsp\_instancemanager == null) {

synchronized (this) {

if (\_jsp\_instancemanager == null) {

\_jsp\_instancemanager = org.apache.jasper.runtime.InstanceManagerFactory.getInstanceManager(getServletConfig());

}

}

}

return \_jsp\_instancemanager;

}

public void \_jspInit() {

}

public void \_jspDestroy() {

}

public void \_jspService(final javax.servlet.http.HttpServletRequest request, final javax.servlet.http.HttpServletResponse response)

throws java.io.IOException, javax.servlet.ServletException {

if (!javax.servlet.DispatcherType.ERROR.equals(request.getDispatcherType())) {

final java.lang.String \_jspx\_method = request.getMethod();

if ("OPTIONS".equals(\_jspx\_method)) {

response.setHeader("Allow","GET, HEAD, POST, OPTIONS");

return;

}

if (!"GET".equals(\_jspx\_method) && !"POST".equals(\_jspx\_method) && !"HEAD".equals(\_jspx\_method)) {

response.setHeader("Allow","GET, HEAD, POST, OPTIONS");

response.sendError(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED, "JSP 只允许 GET、POST 或 HEAD。Jasper 还允许 OPTIONS");

return;

}

}

final javax.servlet.jsp.PageContext pageContext;

javax.servlet.http.HttpSession session = null;

final javax.servlet.ServletContext application;

final javax.servlet.ServletConfig config;

javax.servlet.jsp.JspWriter out = null;

final java.lang.Object page = this;

javax.servlet.jsp.JspWriter \_jspx\_out = null;

javax.servlet.jsp.PageContext \_jspx\_page\_context = null;

try {

response.setContentType("text/html; charset=utf-8");

pageContext = \_jspxFactory.getPageContext(this, request, response,

null, true, 8192, true);

\_jspx\_page\_context = pageContext;

application = pageContext.getServletContext();

config = pageContext.getServletConfig();

session = pageContext.getSession();

out = pageContext.getOut();

\_jspx\_out = out;

out.write("\r\n");

out.write("<!DOCTYPE html>\r\n");

out.write("<html>\r\n");

out.write("<head>\r\n");

out.write("<meta charset=\"utf-8\">\r\n");

out.write("<title>main</title>\r\n");

out.write("</head>\r\n");

out.write("<body>\r\n");

out.write("\t<b>Weclome</b>\r\n");

out.write("\t");

out.write("<pre>\r\n");

**out.write("\tNothing is impossible!\r\n");**

out.write("</pre>");

out.write("\r\n");

out.write("\t<b>Good Bye</b>\r\n");

out.write("</body>\r\n");

out.write("</html>");

} catch (java.lang.Throwable t) {

if (!(t instanceof javax.servlet.jsp.SkipPageException)){

out = \_jspx\_out;

if (out != null && out.getBufferSize() != 0)

try {

if (response.isCommitted()) {

out.flush();

} else {

out.clearBuffer();

}

} catch (java.io.IOException e) {}

if (\_jspx\_page\_context != null) \_jspx\_page\_context.handlePageException(t);

else throw new ServletException(t);

}

} finally {

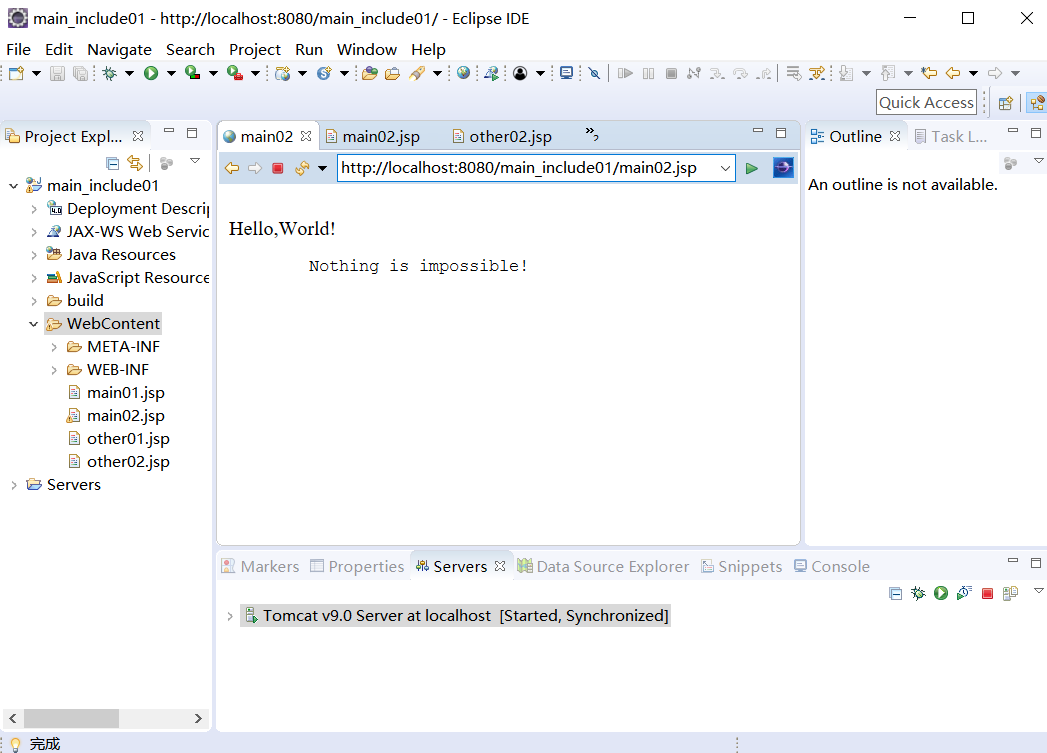
\_jspxFactory.releasePageContext(\_jspx\_page\_context);

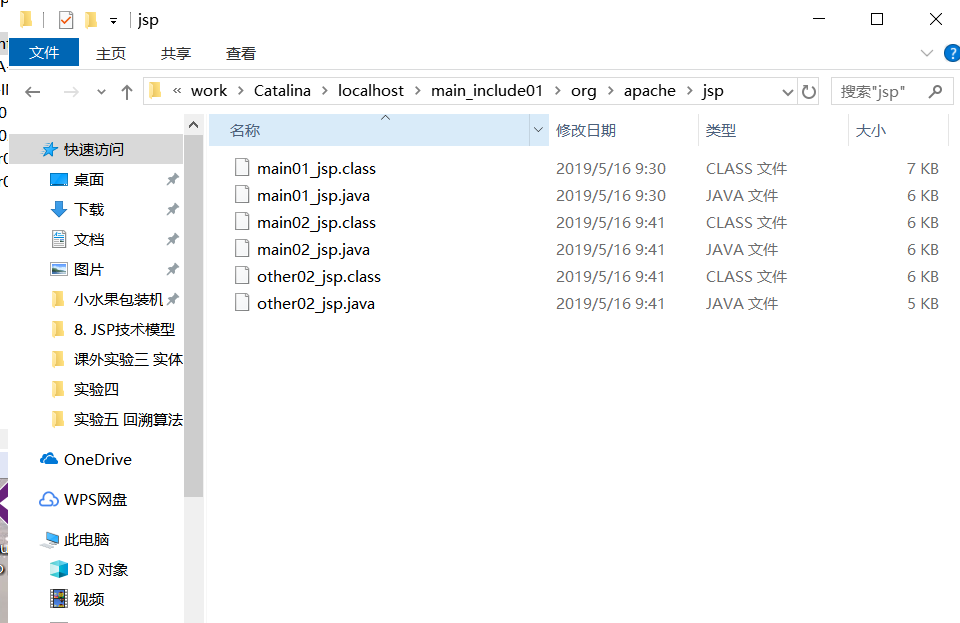
}

}

}

2）动态包含





其中，main02\_jsp.java中的内容如下：

/\*

\* Generated by the Jasper component of Apache Tomcat

\* Version: Apache Tomcat/9.0.14

\* Generated at: 2019-05-16 01:46:00 UTC

\* Note: The last modified time of this file was set to

\* the last modified time of the source file after

\* generation to assist with modification tracking.

\*/

package org.apache.jsp;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.jsp.\*;

public final class main02\_jsp extends org.apache.jasper.runtime.HttpJspBase

implements org.apache.jasper.runtime.JspSourceDependent,

org.apache.jasper.runtime.JspSourceImports {

private static final javax.servlet.jsp.JspFactory \_jspxFactory =

javax.servlet.jsp.JspFactory.getDefaultFactory();

private static java.util.Map<java.lang.String,java.lang.Long> \_jspx\_dependants;

private static final java.util.Set<java.lang.String> \_jspx\_imports\_packages;

private static final java.util.Set<java.lang.String> \_jspx\_imports\_classes;

static {

\_jspx\_imports\_packages = new java.util.HashSet<>();

\_jspx\_imports\_packages.add("javax.servlet");

\_jspx\_imports\_packages.add("javax.servlet.http");

\_jspx\_imports\_packages.add("javax.servlet.jsp");

\_jspx\_imports\_classes = null;

}

private volatile javax.el.ExpressionFactory \_el\_expressionfactory;

private volatile org.apache.tomcat.InstanceManager \_jsp\_instancemanager;

public java.util.Map<java.lang.String,java.lang.Long> getDependants() {

return \_jspx\_dependants;

}

public java.util.Set<java.lang.String> getPackageImports() {

return \_jspx\_imports\_packages;

}

public java.util.Set<java.lang.String> getClassImports() {

return \_jspx\_imports\_classes;

}

public javax.el.ExpressionFactory \_jsp\_getExpressionFactory() {

if (\_el\_expressionfactory == null) {

synchronized (this) {

if (\_el\_expressionfactory == null) {

\_el\_expressionfactory = \_jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory();

}

}

}

return \_el\_expressionfactory;

}

public org.apache.tomcat.InstanceManager \_jsp\_getInstanceManager() {

if (\_jsp\_instancemanager == null) {

synchronized (this) {

if (\_jsp\_instancemanager == null) {

\_jsp\_instancemanager = org.apache.jasper.runtime.InstanceManagerFactory.getInstanceManager(getServletConfig());

}

}

}

return \_jsp\_instancemanager;

}

public void \_jspInit() {

}

public void \_jspDestroy() {

}

public void \_jspService(final javax.servlet.http.HttpServletRequest request, final javax.servlet.http.HttpServletResponse response)

throws java.io.IOException, javax.servlet.ServletException {

if (!javax.servlet.DispatcherType.ERROR.equals(request.getDispatcherType())) {

final java.lang.String \_jspx\_method = request.getMethod();

if ("OPTIONS".equals(\_jspx\_method)) {

response.setHeader("Allow","GET, HEAD, POST, OPTIONS");

return;

}

if (!"GET".equals(\_jspx\_method) && !"POST".equals(\_jspx\_method) && !"HEAD".equals(\_jspx\_method)) {

response.setHeader("Allow","GET, HEAD, POST, OPTIONS");

response.sendError(HttpServletResponse.SC\_METHOD\_NOT\_ALLOWED, "JSP 只允许 GET、POST 或 HEAD。Jasper 还允许 OPTIONS");

return;

}

}

final javax.servlet.jsp.PageContext pageContext;

javax.servlet.http.HttpSession session = null;

final javax.servlet.ServletContext application;

final javax.servlet.ServletConfig config;

javax.servlet.jsp.JspWriter out = null;

final java.lang.Object page = this;

javax.servlet.jsp.JspWriter \_jspx\_out = null;

javax.servlet.jsp.PageContext \_jspx\_page\_context = null;

try {

response.setContentType("text/html; charset=utf-8");

pageContext = \_jspxFactory.getPageContext(this, request, response,

null, true, 8192, true);

\_jspx\_page\_context = pageContext;

application = pageContext.getServletContext();

config = pageContext.getServletConfig();

session = pageContext.getSession();

out = pageContext.getOut();

\_jspx\_out = out;

out.write("\r\n");

out.write("<!DOCTYPE html>\r\n");

out.write("<html>\r\n");

out.write("<head>\r\n");

out.write("<meta charset=\"utf-8\">\r\n");

out.write("<title>main02</title>\r\n");

out.write("</head>\r\n");

out.write("<body>\r\n");

out.write("\t<br>Hello,World!</br>\r\n");

out.write("\t");

**org.apache.jasper.runtime.JspRuntimeLibrary.include(request, response, "other02.jsp", out, false);**

out.write("\r\n");

out.write("</body>\r\n");

out.write("</html>");

} catch (java.lang.Throwable t) {

if (!(t instanceof javax.servlet.jsp.SkipPageException)){

out = \_jspx\_out;

if (out != null && out.getBufferSize() != 0)

try {

if (response.isCommitted()) {

out.flush();

} else {

out.clearBuffer();

}

} catch (java.io.IOException e) {}

if (\_jspx\_page\_context != null) \_jspx\_page\_context.handlePageException(t);

else throw new ServletException(t);

}

} finally {

\_jspxFactory.releasePageContext(\_jspx\_page\_context);

}

}

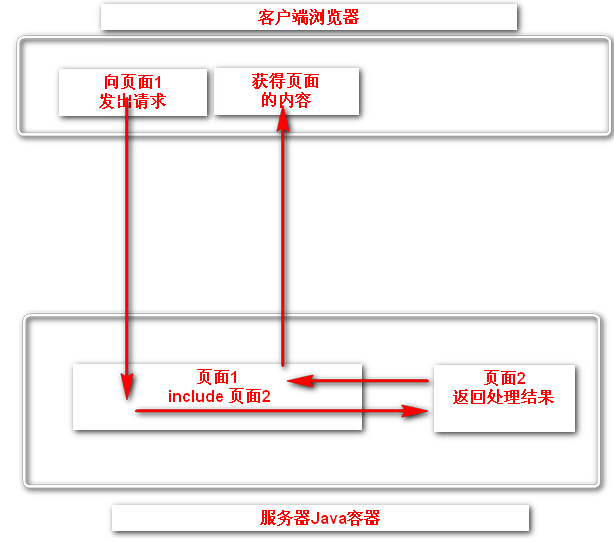
}

1. 比较

通过这两个\_jsp.java文件的比较，可得出include静态和动态包含的区别。

1）静态包含：＜%@ include file=”包含页面”%＞ **静态包含一般用于加载进页面显示后就再也不变的东西**，在页面转化阶段将另一个文件的内容直接包含进来，产生同一个Servlet。本例中**会直接把other01.jsp文件的内容全部内嵌到main01.jsp文件中包含include的语句的位置。**也就是说，**静态include**

* 1. 动态包含：<jsp:include page=" " flush="true"/>。 **动态include与静态include最大的区别在于：包含文件与被包含文件都是先编译执行，再包含，**二者的编译阶段是相互独立的，是不同的Servlet，且变量不通用，但request、response、out隐含变量是同一个。在本例中，main02.jsp先编译成servlet类文件，然后运行，当运行到jsp : include语句处，other02.jsp再编译运行，并把其运行servlet运行结果包含进来。最后把main02.jsp的servlet类运行结果输出到浏览器显示。



1. 在Tomcat服务器上配置一个名称为zjut的Web应用，使用MVC设计模式实现一个简单用户登录系统：
   1. 用户在登陆页面输入用户名和密码；
   2. 如果用户名和密码正确，则转到用户信息显示页面(userinfo.jsp：姓名，性别，年龄和地址；
   3. 如果不正确，则提示用户“用户名或密码”不正确，并重新跳转到登陆页面。
2. 模型(Model)：新建一个名称为User.java的用户JavaBean，具体如下：
3. 包含属性：用户名，密码，姓名，年龄，性别和地址；
4. 包含对应的get和set方法；

（1）源代码

package com.bean;

public class User {

private String username;

private String password;

private String sex;

private int age;

private String address;

public User(String userName, String password, String sex, int age, String address) {

super();

this.username = userName;

this.password = password;

this.sex = sex;

this.age = age;

this.address = address;

}

public User() {

super();

// TODO Auto-generated constructor stub

}

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 String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getAddress() {

return address;

}

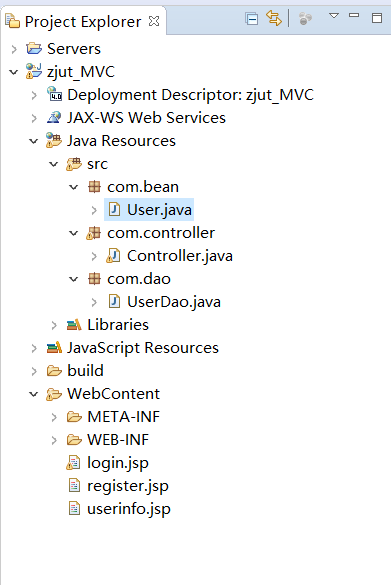
public void setAddress(String address) {

this.address = address;

}

}

1. 实验截屏



1. 视图(View)：
   1. login.jsp页面：用户在登陆页面输入用户名和密码，并具有记住用户名和密码功能；
   2. userinfo.jsp页面：显示姓名，性别，年龄和地址，当用户未登陆直接访问这个页面时，提示用户“请登录”。

控制器(Controller, Controller.java)：用户访问系统的任一资源只能通过Controller.java，映射关系如下（通过查询字符串action参数值来定位资源）：

|  |  |  |
| --- | --- | --- |
| 资源 | urlpattern | 访问形式 |
| login.jsp | /controller?action=login | http://localhost/zjut/controller?action=login |
| userinfo.jsp | /controller?action=userinfo | http://localhost/zjut/controller?action=userinfo |

（1）源代码

com.bean

User.java

package com.bean;

public class User {

private String username;

private String password;

private String sex;

private int age;

private String address;

public User(String username, String password, String sex, int age, String address) {

super();

this.username = username;

this.password = password;

this.sex = sex;

this.age = age;

this.address = address;

}

public User() {

super();

// TODO Auto-generated constructor stub

}

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 String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

Com.controller

Controller.java

package com.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.bean.User;

import com.dao.UserDao;

@WebServlet("/Controller")

public class Controller extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String action=request.getParameter("action");

String location="";

if(action.equals("login")) {

Cookie []cookies=request.getCookies();

String username="123";

String password="";

if(cookies!=null) {

for(int i=0;i<cookies.length;i++) {

Cookie cookie=cookies[i];

if(cookie.getName().equals("username"))

username=cookie.getValue();

if(cookie.getName().equals("password"))

password=cookie.getValue();

/\*

\* System.out.println(username); System.out.println(password);

\*/

}

}

User user=UserDao.verifyAccount(username, password);

//若登录成功

if(user!=null) {

HttpSession session=request.getSession();

request.setAttribute("user", user);

request.setAttribute("message", "恭喜你，登录成功！");

location="userinfo.jsp";

}

else {

HttpSession session=request.getSession();

request.setAttribute("message", "<font color='red'>用户名或密码不正确，请重新登录！</font>");

location="login.jsp";

}

}

else if(action.equals("userinfo01")) {

location="register.jsp";

}

else if(action.equals("userinfo02")) {

location="login.jsp";

}

else if(action.equals("register")) {

String username = request.getParameter("username");

String sex = request.getParameter("sex");

//字符串转整型

int age = Integer.parseInt(request.getParameter("age"));

String address = request.getParameter("address");

String password = request.getParameter("password");

User user = new User(username,password,sex,age,address);

boolean isSuccess=UserDao.addUser(username, password, sex, age, address);

if(isSuccess) {

HttpSession session=request.getSession();

request.setAttribute("message", "恭喜你，注册成功！");

location="login.jsp";

}

else {

HttpSession session=request.getSession();

request.setAttribute("message", "对不起，注册失败！");

location="register.jsp";

}

}

RequestDispatcher rd = request.getRequestDispatcher(location);

rd.forward(request,response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String action=request.getParameter("action");

String location="";

if(action.equals("login")) {

String username="";

String password="";

username=request.getParameter("username");

password=request.getParameter("password");

User user=UserDao.verifyAccount(username, password);

//若登录成功

if(user!=null) {

HttpSession session=request.getSession();

request.setAttribute("user", user);

request.setAttribute("message", "恭喜你，登录成功！");

//如果用户选了自动登录复选框，就向浏览器发送两个Cookie

if((request.getParameter("check"))!=null && (request.getParameter("check")).equals("check")) {

Cookie nameCookie=new Cookie("username",username);

Cookie pswdCookie=new Cookie("password",password);

nameCookie.setMaxAge(60\*60);

pswdCookie.setMaxAge(60\*60);

response.addCookie(nameCookie);

response.addCookie(pswdCookie);

}

location="userinfo.jsp";

}

else {

HttpSession session=request.getSession();

request.setAttribute("message", "<font color='red'>用户名或密码不正确，请重新登录！</font>");

location="login.jsp";

}

}

else if(action.equals("userinfo01")) {

location="register.jsp";

}

else if(action.equals("userinfo02")) {

location="login.jsp";

}

else if(action.equals("register")) {

String username = request.getParameter("username");

String sex = request.getParameter("sex");

//字符串转整型

int age = Integer.parseInt(request.getParameter("age"));

String address = request.getParameter("address");

String password = request.getParameter("password");

User user = new User(username,password,sex,age,address);

boolean isSuccess=UserDao.addUser(username, password, sex, age, address);

if(isSuccess) {

HttpSession session=request.getSession();

request.setAttribute("message", "恭喜你，注册成功！");

location="login.jsp";

}

else {

HttpSession session=request.getSession();

request.setAttribute("message", "对不起，注册失败！");

location="register.jsp";

}

}

RequestDispatcher rd = request.getRequestDispatcher(location);

rd.forward(request,response);

}

}

com.dao

UserDao.java

package com.dao;

import com.bean.User;

import java.util.HashMap;

import java.util.Map;

public class UserDao {

private static Map<String,User> db=new HashMap<String,User>();

public static boolean addUser(String username,String password,

String sex,int age,String address) {

//若用户名已被注册

if(db.containsKey(username)) {

return false;

}

else {

User user= new User(username,password,sex,age,address);

//加入该用户

db.put(username, user);

return true;

}

}

public static User verifyAccount(String username,String password) {

if(db.containsKey(username)) {

//读取该用户名的用户

User user=db.get(username);

if(user.getPassword().equals(password)) {

return user;

}

else {

return null;

}

}

return null;

}

}

Login.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>用户登录页</title>

</head>

<body>

<%

Object msg=request.getAttribute("message");

if(msg!=null)

out.println(msg);

%>

<form action=*"Controller?action=login"* method=*"post"*>

<table>

<h3>用户登录</h3>

<tr>

<td>登录名：</td>

<td><input name=*"username"* type=*"text"* size=*"20"*></td>

</tr>

<tr>

<td>密 码:</td>

<td><input name=*"password"* type=*"password"* size=*"20"*></td>

</tr>

</table>

<br>

<input type=*"checkbox"* name=*"check"* value=*"check"*/>自动登录<br>

<br>

<input type=*"submit"* value=*"登 录"*> &nbsp;&nbsp; <input type=*"reset"*

value=*"重 置"*>

</form>

</body>

</html>

Register.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>用户注册页</title>

</head>

<body>

<%

Object msg=request.getAttribute("message");

if(msg!=null)

out.println(msg);

%>

<fieldset>

<legend>请填写如下内容：</legend>

<form action=*"Controller?action=register"* method=*"post"*>

<table style="margin-left: *40%*">

<caption>用户注册</caption>

<tr>

<td>用户名：</td>

<td><input name=*"username"* type=*"text"* size=*"20"*></td>

</tr>

<tr>

<td>性 别:</td>

<td><input name=*"sex"* type=*"text"* size=*"5"*></td>

</tr>

<tr>

<td>年 龄:</td>

<td><input name=*"age"* type=*"text"* size=*"5"*></td>

</tr>

<tr>

<td>地址:</td>

<td><input name=*"address"* type=*"text"* size=*"20"*></td>

</tr>

<tr>

<td>密 码:</td>

<td><input name=*"password"* type=*"password"* size=*"20"*></td>

</tr>

</table>

<input type=*"submit"* value=*"注 册"*> &nbsp;&nbsp;

<input type=*"reset"* value=*"重 置"*>

</form>

</fieldset>

</body>

</html>

Userifo.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<% request.setCharacterEncoding("UTF-8"); %>

<title>用户登录成功页</title>

</head>

<body>

<%

Object msg=request.getAttribute("message");

if(msg==null){

%>

您还未登录，请先注册登录！<br>

<a href=*"Controller?action=userinfo01"*>注 册</a>&nbsp;&nbsp;

<a href=*"Controller?action=userinfo02"*>登 录</a>

<% }

else{

out.println(msg);

%>

欢迎您：<%=request.getParameter("username") %>

<h3>用户信息如下：</h3><br>

姓名：${user.username}<br>

性别：${user.sex}<br>

年龄：${user.age}<br>

地址：${user.address}<br>

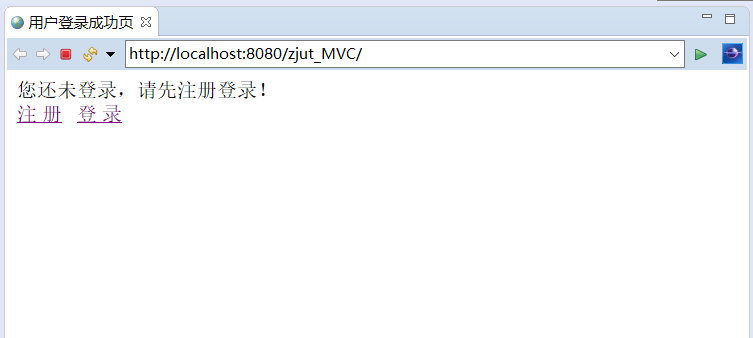
<%} %>

</body>

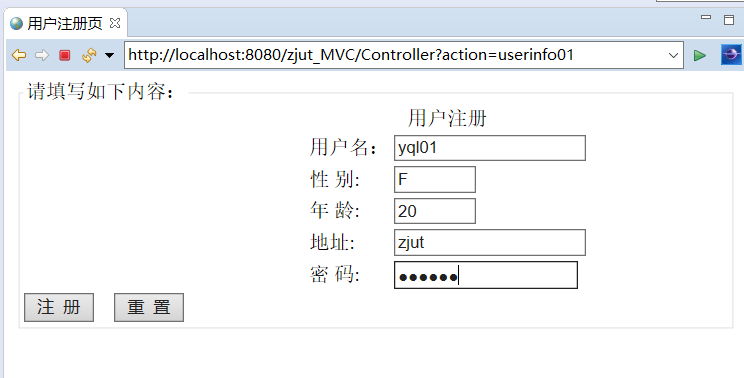
</html>

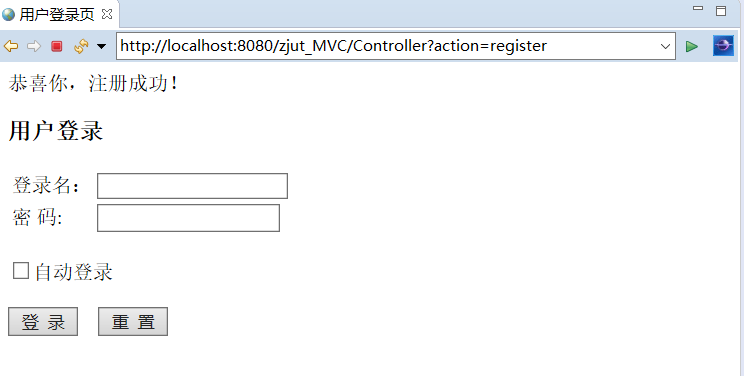
（2）实验截屏

1）先进入userinfo.jsp界面

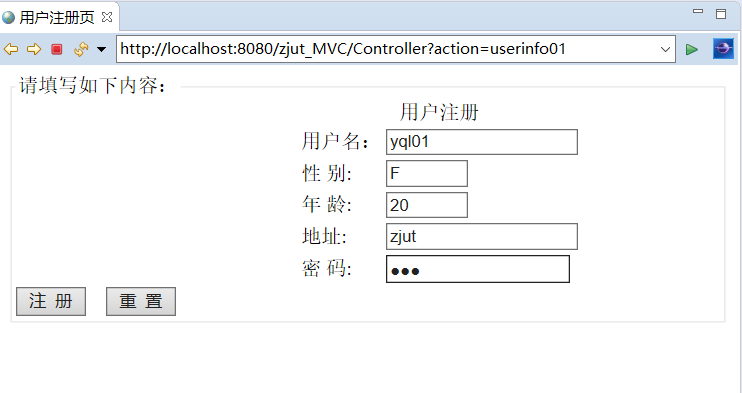


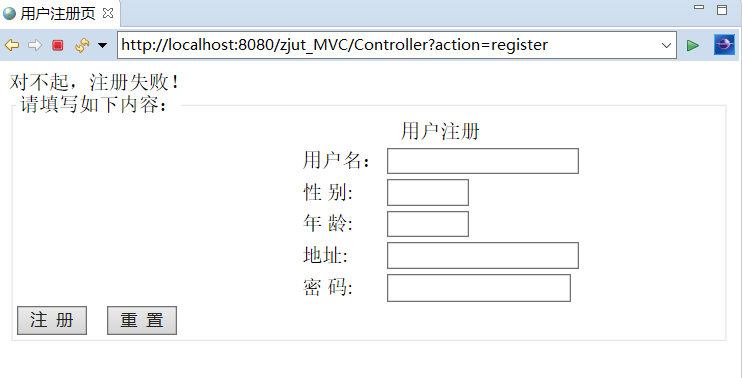
1. 点击注册按钮，Controller转发进入注册界面，填写注册信息。



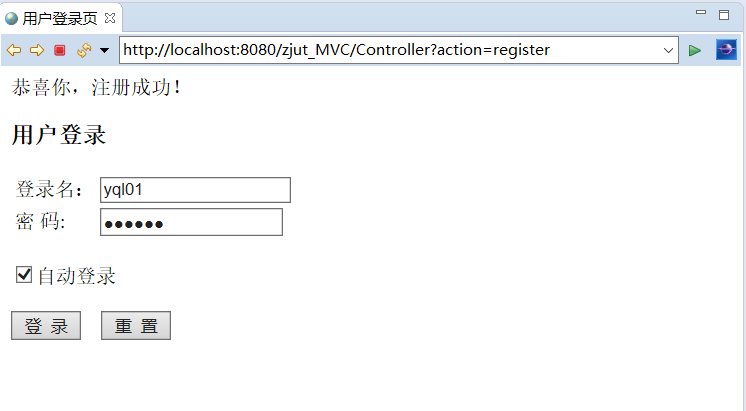


注册时，若该用户名已存在，会注册失败。

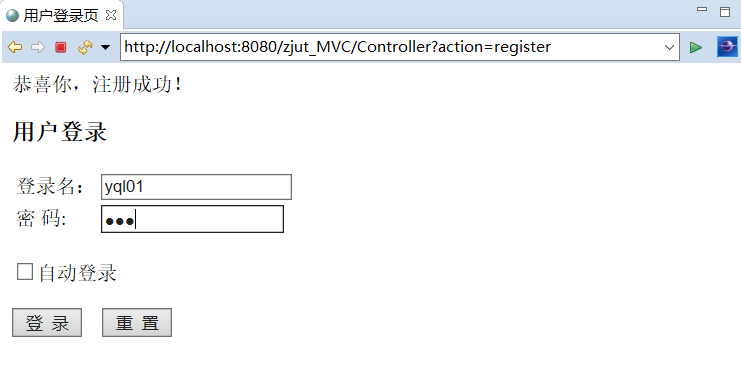




1. 登录成功



1. 登录时，若用户名或密码不正确





6）登录成功，用户名和密码都正确，则跳转到用户信息显示页面(userinfo.jsp：姓名，性别，年龄和地址）



（3）实验体会：

1）通过本次实验，我了解了MVC基本模式，模型、视图和控制部分相对独立。Model（模型）主要包含业务数据和业务逻辑。View(视图），是用户和程序交互的接口，把输入提交给控制器处理。Controller（控制器）主要用来处理用户输入的数据。

2）代码中，servlet负责处理用户请求，jsp负责数据显示给用户，javabean负责封装处理数据，从而实现界面和业务逻辑分离。 Servlet、JSP和JavaBean各个模块之间层次清晰，并且在本题中用户访问系统的任一资源只能通过控制器Controller映射，更加直观。

3）通过MVC实现用户登录注册程序这个案例，可以把之前的学过的XML、Servlet、JSP、MVC模式等知识点都串联起来。

4）虽然我对知识点已有了基本的理解，但我在做实验的过程中依然遇到了两个问题，由此可见实践也是非常重要的。下面是我对我所遇到的两个问题的描述和解决方案。其中一个是对MVC中cotroller控制器的处理遇到了一点困难，通过查阅实验指导书以及老师的指导，解决了通过查询字符串action参数值来定位资源的问题。最终实现了用户访问系统的任一资源只能通过Controller.java进行映射。还有一个问题是通过Cookie实现自动登录的功能，一开始我直接设置读取Cookie，导致登录失败。通过老师的指点，发现Cookie还没有发送，当用户勾选“记住用户名”复选框时，response.addCookie加入新new的用户名和密码的Cookie，这样在下次发送Get请求时可读取Cookie中的值，实现自动登录。