

## 10-书城项目第三阶段

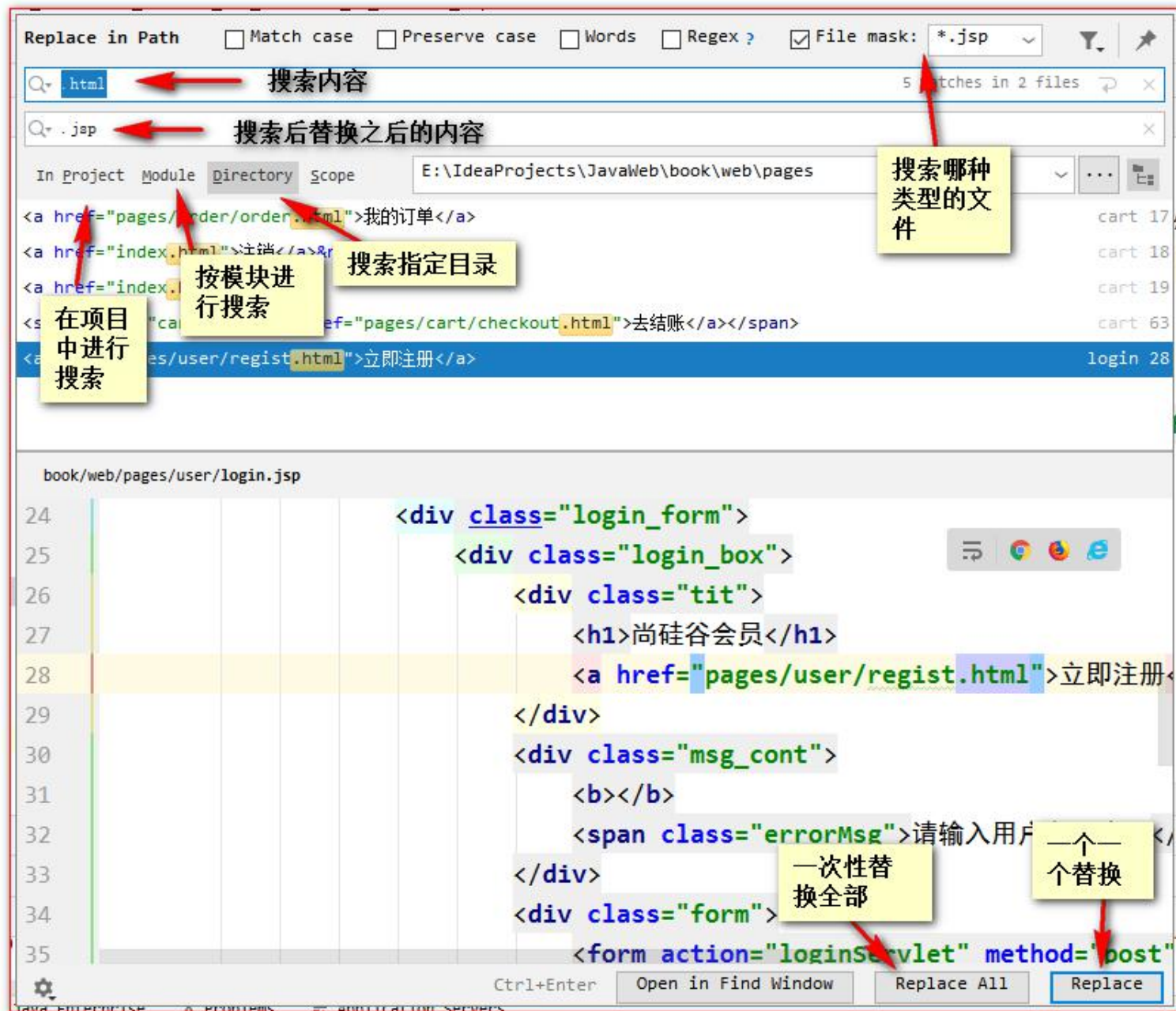
讲师：王振国

### 1.第三阶段：

#### a)页面 jsp 动态化

- 1、在 html 页面顶行添加 page 指令。
- 2、修改文件后缀名为：.jsp
- 3、使用 IDEA 搜索替换.html 为.jsp(快捷键：Ctrl+Shift+R)





## b) 抽取页面中相同的内容

### i. head 中 css、jquery、base 标签

```
<%
String basePath = request.getScheme()
    + "://"
    + request.getServerName()
    + ":"
    + request.getServerPort()
    + request.getContextPath()
    + "/";
```



## ii. 每个页面的页脚

### iii. 登录成功后的菜单

#### iv. manager 模块的菜单

### c) 登录，注册错误提示，及表单回显

Servlet 程序端需要添加回显信息到 Request 域中

```
// 如果等于null,说明登录失败!
if (loginUser == null) {
    // 把错误信息,和回显的表单项信息,保存到Request域中
    req.setAttribute(s: "msg", o: "用户或密码错误!");
    req.setAttribute(s: "username", username);
    // 跳回登录页面
    req.getRequestDispatcher(s: "/pages/user/login.jsp").forward(req, resp);
} else {
```

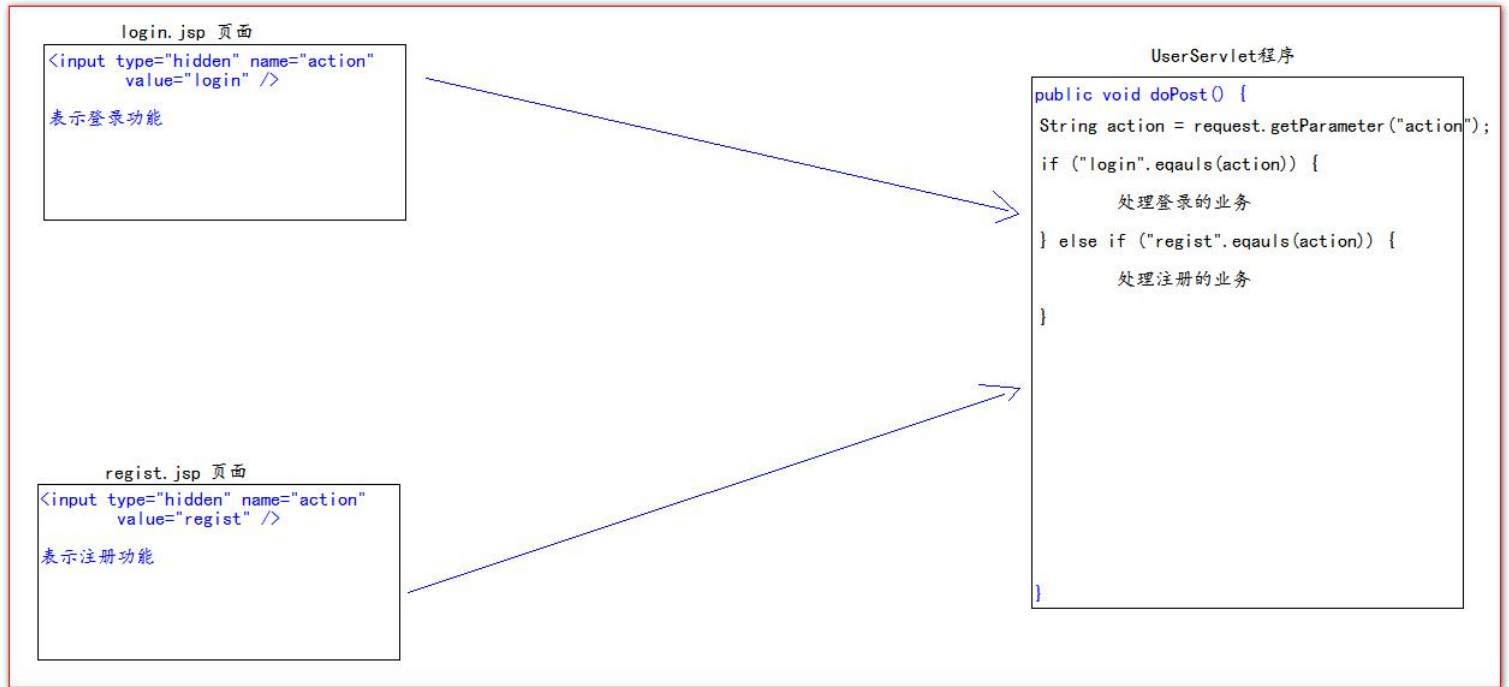
jsp 页面, 需要输出回显信息

```
<b></b>
<span class="errorMsg">
    <%=request.getAttribute(s: "msg")==null?"请输入用户名和密码":request.getAttribute("msg")%>
</span>
</div>
div class="form">
    <form action="loginServlet" method="post">
        <label>用户名称: </label>
        <input class="itxt" type="text" placeholder="请输入用户名"
            autocomplete="off" tabindex="1" name="username"
            value="<%=request.getAttribute(s: "username")==null?"":request.getAttribute(s: "username")%>" />
        <br />
```

## d)BaseServlet 的抽取

在实际的项目开发中, 一个模块, 一般只使用一个 Servlet 程序。

## 代码优化一：代码优化：合并 LoginServlet 和 RegisterServlet 程序为 UserServlet 程序



UserServlet 程序：

```
public class UserServlet extends HttpServlet {

    private UserService userService = new UserServiceImpl();

    /**
     * 处理登录的功能
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    protected void login(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

        // 1、获取请求的参数
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        // 调用 userService.login() 登录处理业务
        User loginUser = userService.login(new User(null, username, password, null));
        // 如果等于 null, 说明登录 失败!
        if (loginUser == null) {
            // 把错误信息, 和回显的表单项信息, 保存到 Request 域中
            req.setAttribute("msg", "用户或密码错误!");
            req.setAttribute("username", username);
            // 跳回登录页面
        }
    }
}
```

```
req.getRequestDispatcher("/pages/user/login.jsp").forward(req, resp);
} else {
    // 登录 成功
    // 跳到成功页面 login_success.html
    req.getRequestDispatcher("/pages/user/login_success.jsp").forward(req, resp);
}

}

/**
 * 处理注册的功能
 * @param req
 * @param resp
 * @throws ServletException
 * @throws IOException
 */
protected void regist(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

    // 1、获取请求的参数
    String username = req.getParameter("username");
    String password = req.getParameter("password");
    String email = req.getParameter("email");
    String code = req.getParameter("code");

    // 2、检查 验证码是否正确 === 写死, 要求验证码为:abcde
    if ("abcde".equalsIgnoreCase(code)) {
    // 3、检查 用户名是否可用
        if (userService.existsUsername(username)) {
            System.out.println("用户名[" + username + "]已存在!");

            // 把回显信息, 保存到Request 域中
            req.setAttribute("msg", "用户名已存在!!");
            req.setAttribute("username", username);
            req.setAttribute("email", email);

            // 跳回注册页面
            req.getRequestDispatcher("/pages/user/regist.jsp").forward(req, resp);
        } else {
            // 可用
            // 调用 Sservice 保存到数据库
            userService.registUser(new User(null, username, password, email));

            // 跳到注册成功页面 regist_success.jsp
            req.getRequestDispatcher("/pages/user/regist_success.jsp").forward(req, resp);
        }
    } else {
        // 把回显信息, 保存到Request 域中
```



```
req.setAttribute("msg", "验证码错误!!");
req.setAttribute("username", username);
req.setAttribute("email", email);

System.out.println("验证码[" + code + "]错误");
req.getRequestDispatcher("/pages/user/regist.jsp").forward(req, resp);
}
```

```
}

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    String action = req.getParameter("action");

    if ("login".equals(action)) {
        login(req, resp);
    } else if ("regist".equals(action)) {
        regist(req, resp);
    }
}
```

还要给 login.jsp 添加隐藏域和修改请求地址

```
</div>
<div class="form">
    <form action="userServlet" method="post">
        <input type="hidden" name="action" value="login" />
        <label>用户名称: </label>
        <input class="itxt" type="text" placeholder="请输入用户名"
            autocomplete="off" tabindex="1" name="username">
```

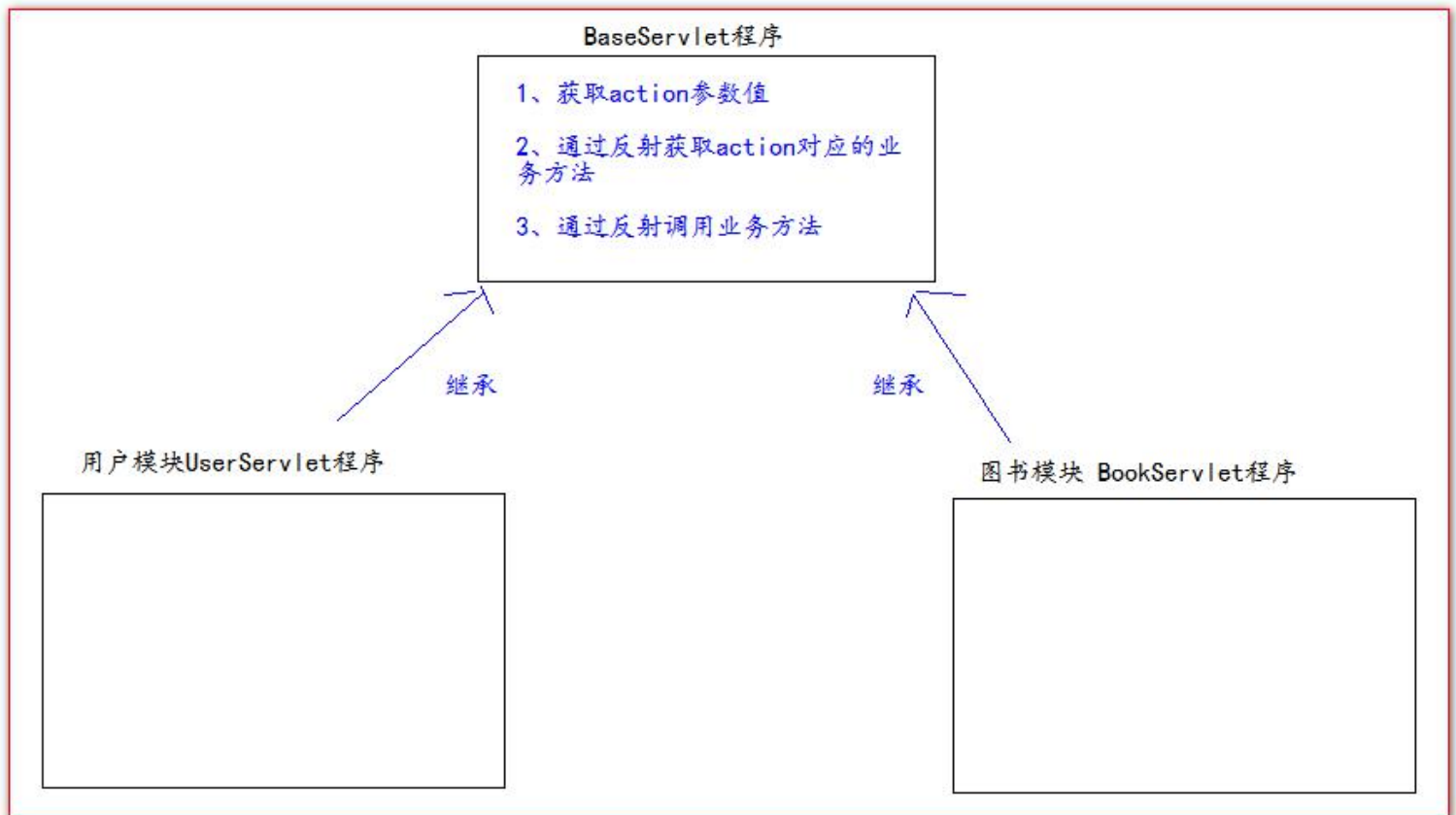
给 regist.jsp 页面添加隐藏域 action，和修改请求地址

```
</div>
<div class="form">
    <form action="userServlet" method="post">
        <input type="hidden" name="action" value="regist">
        <label>用户名称: </label>
        <input class="itxt" type="text" placeholder="请输入用户名">
```

## 优化代码二：使用反射优化大量 else if 代码：

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    String action = req.getParameter("action");
    try {
        // 获取 action 业务鉴别字符串，获取相应的业务 方法反射对象
        Method method = this.getClass().getDeclaredMethod(action, HttpServletRequest.class,
        HttpServletResponse.class);
        // System.out.println(method);
        // 调用目标业务 方法
        method.invoke(this, req, resp);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

## 代码优化三：抽取 BaseServlet 程序。



BaseServlet 程序代码：



```
public abstract class BaseServlet extends HttpServlet {

    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        String action = req.getParameter("action");
        try {
            // 获取 action 业务鉴别字符串, 获取相应的业务 方法反射对象
            Method method = this.getClass().getDeclaredMethod(action, HttpServletRequest.class,
HttpServletResponse.class);
            //      System.out.println(method);
            // 调用目标业务 方法
            method.invoke(this, req, resp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

修改 UserServlet 程序继承 BaseServlet 程序。

```
import java.io.IOException;

public class UserServlet extends BaseServlet {
```

## e)数据的封装和抽取 BeanUtils 的使用

BeanUtils 工具类, 它可以一次性的把所有请求的参数注入到 JavaBean 中。

BeanUtils 工具类, 经常用于把 Map 中的值注入到 JavaBean 中, 或者是对对象属性值的拷贝操作。

BeanUtils 它不是 Jdk 的类。而是第三方的工具类。所以需要导包。

1、导入需要的 jar 包:

commons-beanutils-1.8.0.jar

commons-logging-1.1.1.jar

2、编写 WebUtils 工具类使用:

WebUtils 工具类:

```
public class WebUtils {
    /**
     * 把 Map 中的值注入到对应的 JavaBean 属性中。
     * @param value
```

```

    * @param bean
    */
    public static <T> T copyParamToBean( Map value , T bean ){
        try {
            System.out.println("注入之前: " + bean);
            /**
             * 把所有请求的参数都注入到user 对象中
             */
            BeanUtils.populate(bean, value);
            System.out.println("注入之后: " + bean);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bean;
    }
}

```

## 书城-第四阶段。使用 EL 表达式修改表单回显

以登录为示例：

```

<div class="msg_cont">
    <b></b>
    <span class="errorMsg">
        ${ empty requestScope.msg ? "请输入用户名和密码":requestScope.msg }
    </span>
</div>
<div class="form">
    <form action="userServlet" method="post">
        <input type="hidden" name="action" value="login" />
        <label>用户名称: </label>
        <input class="itxt" type="text" placeholder="请输入用户名"
            autocomplete="off" tabindex="1" name="username"
            value="${requestScope.username}" />
        <br />
        <br />
    </form>
</div>

```



