轻量级 J2EE 框架应用

E 4　A Simple Controller with handling result view

学号：SA18225433

姓名：杨帆

报告撰写时间：2018/12/01

1.主题概述

1、 Xml。

Xml 为可扩展标记语言，是互联网数据传输的重要工具，它可以跨越互联网任何的平台，不受编程语言和操作系统的限制。Xml 能够自定义扩展标签，有助于在服务器之间传输结构化数据，使得开发人员更好的控制数据的存储和传输。Xml 多用作应用程序的配置文件，数据交换的载体。

2、 XSLT。

XSL 指扩展样式表语言（EXtensible Stylesheet Language），XSLT 指 XSL 的转换，即 XSLT 工具用于将 xml 文件转化为 XHTML 或 HTML 文件，XSLT 样式表本身就是一个 XML 文档，因此它也符合 XML 文档的规则，同时它还具有自己的标签语法，通过相应的标签语法获得 xml 文件中的值，拼接为 html 格式的文件。

3、 请描述 Struts 框架中视图组件的工作方式，如<s:form>/<s:submit>

Struts truts 视图组件的工作方式，

1.DTO 数据传输对象

利用 JavaBean 创建数据传输对象，DTO 用于在不同的层之间传递数据。

2.Struts 框架提供的 DTO：ActionFrom Bean

控制层可以从 ActionForm Bean 读取用户输入数据，也可以把来自模型层 的数据存放到 ActionForm Bean 中，返回给视图。

2.假设

本次作业能够使用 IDEA 实现一个具有结果处理的 Controller，即使用 success_view.xml 定义一个视图，视图中可以自定义不同的组件，同时自定义不同组件的属性，然后将此 xml 作为 login_action 返回结果 result 为 success 的视图，然后定义 xslt 文档，当我们在访问 success_view.xml 时，能解析为 html 文档。

3.实现或证明

1. 实现成果

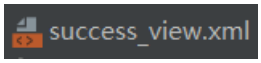
e4：https://github.com/saaaaaail/J2eee4

2. 在 UserSC 工程中，定义一个 success_view.xml 视图。该视图分成<header>与<body>两部分；<header>定义视图标题，<body>定义视图内容。格式可参考如下：

2.1 在 success_view.xml 的<body>中定义一个表单<form>，表单属性有 name\action\method。<form>内部可以嵌套视图组件，如<textView>\<buttonView>\<checkBoxView>等。

2.2 每个视图组件根据用途不同，定义属性 name\label\value\method 等。

首先定义一个 success_view.xml，



创建 view 标签，在里面创建<header>和<body>子标签，在<body>标签中定义一个<form>标签，然后定义这个标签的属性 name 表示表单的名字，action 表示表单提交数据的 action，method 表示表单以何种方式提交数据，

```
<form>
    <name>logoutForm</name>
    <action>logout.action</action>
    <method>post</method>
```

然后在表单中定义子标签两个 textView 和一个 buttonView 分别表示登录名和年龄框以及登录按钮，分别定义其属性包括 name，label，value，

```
<textView>
    <name>userName</name>
    <label>Login Name:</label>
    <value>sail</value>
</textView>
<textView>
    <name>userAge</name>
    <label>Age</label>
    <value>23</value>
</textView>
<buttonView>
    <name>logoutButton</name>
    <label>Logout</label>
</buttonView>
```

3. 基于 E3，将 success_view.xml 作为 login action 的 success 结果视图。参考代码如下：

修改 controller.xml 文件，将 result 标签 name 为 success 的 value 值改为指向 success_view.xml 页面，

```
<action name="login" class="water.ustc.action.LoginAction" method="handleLogin">
    <interceptor-ref name="log"></interceptor-ref>
    <result name="success" type="forward" value="pages/success_view.xml"></result>
    <result name="failure" type="redirect" value="pages/failure.jsp"></result>
</action>
```

4．修改 SimpleController 工程中控制器的功能，当客户端请求 action 返回 result 的资源后缀为“*_view.xml”时，由控制器动态生成推送至客户端的视图。即，根据“*_view.xml”的定义，生成浏览器可以执行的 html 页面（可以参考 XSLT 的使用）。如，将 success_view.xml 中的<buttonView>节点翻译成 html 中的<input>节点。

即将返回的 xml 文档解析为 html 文档，通过定义 success_view.xsl 文档，并与 success_view.xml 文档关联起来，

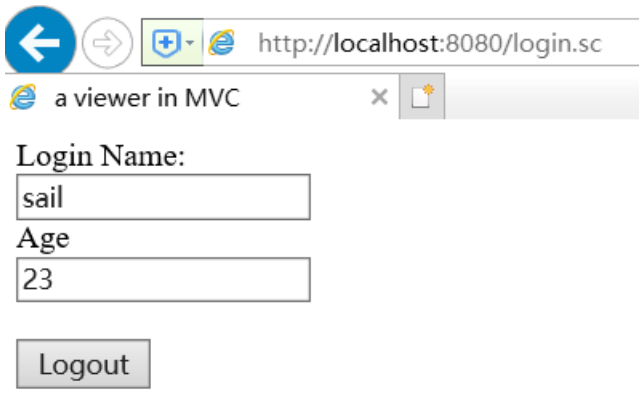
```
<?xml-stylesheet type="text/xsl" href="pages/success_view.xsl"?>
```

在 xsl 文档中解析 xml 文档中的自定义节点，通过<xsl:value-of select=...>标签来获得 select 后面选择的 xml 标签路径的 value 值，通过<xsl:attribute name="...">标签来设置 xsl 文档标签的属性，

```
<form>
    <xsl:attribute name="name"><xsl:value-of select="view/body/form/name"/></xsl:attribute>
    <xsl:attribute name="action"><xsl:value-of select="view/body/form/action"/></xsl:attribute>
    <xsl:attribute name="method"><xsl:value-of select="view/body/form/method"/></xsl:attribute>
    <xsl:for-each select="view/body/form/textView">
        <xsl:value-of select="label"/><br/>
        <input type="text">
            <xsl:attribute name="name"><xsl:value-of select="name"/></xsl:attribute>
            <xsl:attribute name="value"><xsl:value-of select="value"/></xsl:attribute>
        </input><br/>
    </xsl:for-each>
    <br/>
    <input type="submit">
        <xsl:attribute name="name"><xsl:value-of select="view/body/form/buttonView/name"/></xsl:attribute>
        <xsl:attribute name="value"><xsl:value-of select="view/body/form/buttonView/label"/></xsl:attribute>
    </input><br/>
</form>
```

5．自主添加辅助类或工具类，以使控制器代码简洁易读。重新打包导出 simple-controller.jar，测试 UserSC 工程以上任务实现结果，直到调试通过。

请求 url：http://localhost:8080/login.sc



5.结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。

本次作业学习了如何构建返回的 xml 视图，以及如何使用 xslt 解析 xml 视图为 html 文档。

6.参考文献

1.参阅 github 学长代码编写风格
2.XML 通过 XSL 格式化的那点事(XML 到自定义节点折叠显示)
3.修改 html 中 button 显示的文字
4. HTML <form> 标签
5.XML 文件详解以及解析