第 3 章　Struts2 的高级组件

本章介绍 Struts2 中比较常用的高级组件。通过这些高级组件的学习，有助于进一步了解和使用 Struts2 框架。

本章主要内容如下。

（1）Struts2 对国际化的支持。

（2）Struts2 常用拦截器的使用。

（3）Struts2 的数据验证功能。

（4）Struts2 对文件上传和下载的支持。

3.1　Struts2 的国际化


Struts2 的国际化

国际化是指一个应用程序在运行时能够根据客户端请求所来自国家或地区语言的不同而显示不同的用户界面。例如，请求来自于一台中文操作系统的客户端计算机，则应用程序响应界面中的各种标签、错误提示和帮助信息均使用中文文字；如果客户端计算机采用英文操作系统，则应用程序也应能识别并自动以英文界面进行响应。

引入国际化机制的目的在于提供自适应的、更友好的用户界面，而不必改变程序的其他功能或者业务逻辑。人们常用 I18N 这个词作为国际化的简称，其来源是英文单词 Internationalization 的首末字母 I 和 N 及它们之间的字符数 18。

3.1.1　Struts2 实现国际化的流程

Struts2 国际化是建立在 Java 国际化基础上的，Java 对国际化进行了优化和封装，从而简化了国际化的实现过程。Struts2 国际化流程如图 3-1 所示。

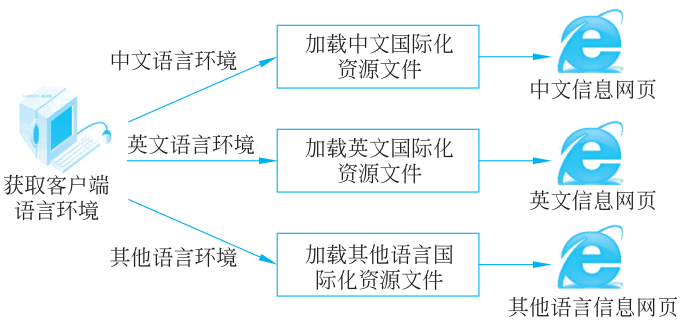

图 3-1　Struts2 国际化流程

具体流程如下。

（1）不同地区使用的操作系统环境不同，如中文操作系统、英文操作系统等。获得客户端地区的语言环境后，在 struts.xml 文件中会找到相应的国际化资源文件，如果操作系统环

境是中文语言环境，就加载中文国际化资源文件。所以国际化需要编写支持多个语言的国际化资源文件，并且在 struts.xml 文件中配置。

(2) 根据选择的语言加载相应的国际化资源文件，视图通过 Struts2 标签读取国际化资源文件并把数据输出到页面上，完成页面的显示。

下面介绍在国际化流程中用到的文件。

1. 国际化资源文件或者资源文件

国际化资源文件又称资源文件，是以 properties 为扩展名的文件，新建一个文本文件把扩展名改为 properties 即可，该文件以“键＝值”的形式存储资源数据。

例如：

```
key=value
loginName=用户名称
loginPassword=用户密码
```

当需要多个资源文件为不同语言版本提供国际化服务时，可以为资源文件命名，命名的格式有以下两种形式：

```
资源文件名.properties
资源文件名_语言种类.properties
```

文件扩展名必须是 properties，语言种类必须是有效的 ISO(International Organization for Standardization，国际标准化组织)语言代码，ISO-639 标准定义的这些代码格式为英文小写、双字符，具体如表 3-1 所示。

表 3-1　常用标准语言代码

语　言	编码	语　言	编码	语　言	编码
汉语(Chinese)	zh	法语(French)	fr	日语(Japanese)	ja
英语(English)	en	德语(German)	de	意大利语(Italian)	it

资源文件如果使用第一种命名方式，即默认语言代码，当系统找不到与客户端请求的语言环境匹配的资源文件时，就使用默认的属性文件。

例如，如果要对前面介绍的登录系统进行国际化处理，要求根据不同的语言环境显示中文和英文用户界面，那么就需要创建中文和英文的资源文件，分别取名为 globalMessages_GBK.properties 和 globalMessages_en_US.properties，内容分别如例 3-1 和例 3-2 所示。

【例 3-1】　中文资源文件(globalMessages_GBK.properties)。

```
loginTitle=用户登录
loginName=用户名称
loginPassword=用户密码
loginSubmit=登录
```

【例 3-2】　英文资源文件(globalMessages_en_US.properties)。

```
loginTitle=UserLogin
loginName=UserName
loginPassword=UserPassword
loginSubmit=Login
```

从例 3-1 和例 3-2 中可以看出，国际化资源文件的内容都是以 key＝value 形式存在的。资源文件中 key 部分是相同的，即等号左边部分相同，value 部分不同，即等号右边部分不同。

在国际化时，所有的字符都要使用标准的编码方式，需要把中文字符转换为 Unicode 代码，否则在国际化处理时页面将会出现乱码。如例 3-1 中的中文国际化资源文件是不能直接使用的，必须转换为指定的编码方式。可以使用 JDK 自带的 native2ascii 工具进行中文国际化资源文件编码方式的转换。具体操作如下：选择“开始”→“运行”菜单，输入 cmd 命令，单击“确定”按钮后出现如图 3-2 所示的命令行窗口。

图 3-2　命令行窗口

如果开发平台使用的是 NetBeans 8.2，将资源文件放在“D:\Web 框架技术(Struts2＋Hibernate5＋Spring5)教程(第 3 版)\ch03\src\java”，即和 struts.xml 文件一起放在默认的资源包中，其中“D:\Web 框架技术(Struts2＋Hibernate5＋Spring5)教程(第 3 版)”为工作区，\ch03 为项目名，\src\java 为资源文件存放的目录；如果使用的是 MyEclipse 2017 或 Eclipse，将资源文件放在 struts.xml 文件所在的文件夹，位置是“D:\Web 框架技术(Struts2＋Hibernate5＋Spring5)教程(第 3 版)\ch03\src”。

在图 3-2 中通过 cd 命令进入资源文件所在的文件夹下，输入 native2ascii -encoding UTF-8 globalMessages_GBK.properties globalMessages_zh_CN.properties，按 Enter 键后，如图 3-3 所示。命令执行后在该文件夹下会生成一个 globalMessages_zh_CN.properties 文件，该文件的内容如例 3-3 所示。

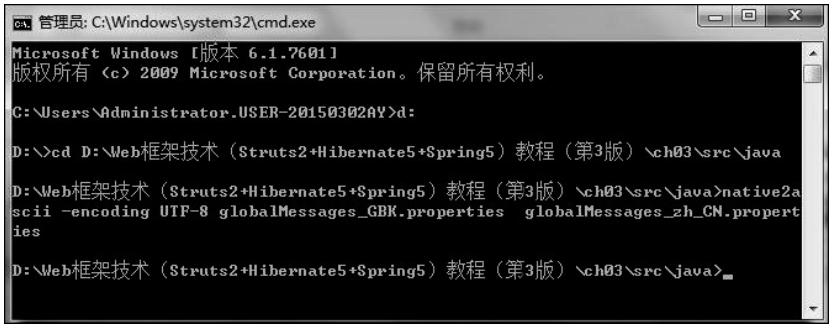

图 3-3　编码转换命令

【例 3-3】　编译后对应的 Unicode 代码(globalMessages_zh_CN.properties)。

```
loginTitle=\u7528\u6237\u767b\u5f55
loginName=\u7528\u6237\u540d\u79f0
loginPassword=\u7528\u6237\u5bc6\u7801
loginSubmit=\u767b\u5f55
```

2. 在 struts.xml 文件中配置

编写完国际化资源文件后，需要在 struts.xml 配置文件中配置国际化资源文件的名字，从而使 Struts2 的 I18N 拦截器在需要加载国际化资源文件的时候能找到这些国际化资源文件，在 struts.xml 中的配置很简单，代码如例 3-4 所示。

【例 3-4】 在 struts.xml 中配置国际化资源文件。

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <!--使用 Struts2 中的 I18N 拦截器，并通过<constant>元素配置常量，指定国际化资源文
    件的名字，value 的值就是常量值，即国际化资源文件的名字-->
    <constant name="struts.custom.i18n.resources" value="globalMessages" />
    <constant name="struts.i18n.encoding" value="UTF-8" />
    <package name="I18N" extends="struts-default">
        <action name="checkLogin" class="loginAction.LoginAction">
            <result name="success">/I18N/loginSuccess.jsp</result>
            <result name="error">/I18N/login.jsp</result>
        </action>
    </package>
</struts>
```

3. 输出国际化信息

国际化资源文件中的 value 值要根据语言环境通过 Struts2 标签输出到页面上。可以在页面上输出国际化信息，也可以在 Struts2 表单标签的 label 标签上输出国际化信息。

3.1.2 Struts2 国际化应用实例

本项目开发一个中英文登录页面，通过该项目的开发可以帮助读者了解 Struts2 国际化应用的开发过程。

1. 项目介绍

该项目为中英文登录系统，项目有一个登录页面(login.jsp)，代码如例 3-5 所示；对应的资源文件如例 3-1 和例 3-2 所示；登录页面对应的业务控制器是 LoginAction 类，代码如例 3-7 所示；如果登录成功(用户名或密码正确)转到 loginSuccess.jsp 页面，代码如例 3-6 所示；如果登录失败(用户名或密码不正确)则重新回到登录页面(login.jsp)。此外，还需要配置 web.xml，代码如例 1-3 所示；配置 struts.xml，代码如例 3-4 所示。项目的文件结构如图 3-4 所示。

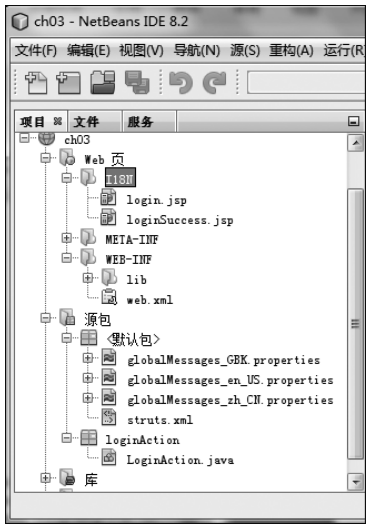

图 3-4 项目的文件结构

2. 在 web.xml 中配置核心控制器 StrutsPrepareAndExecuteFilter

参考 1.3.1 节中的例 1-3。

3. 编写国际化资源文件并进行编码转换

编写中英文国际化资源文件，参考例 3-1 和例 3-2；编码转换参考图 3-3。

4. 编写视图组件(JSP 页面)输出国际化消息

编写一个如图 3-5 或者图 3-6 所示的登录页面。

图 3-5　中文登录页面

图 3-6　英文登录页面

备注：因为 360 安全浏览器不太支持国际化，所以本节例子使用的是 IE 浏览器。

登录页面是一个 JSP 页面，代码如例 3-5 所示。

【例 3-5】　中英文登录页面(login.jsp)。

```
<%@page contentType="text/html; charset=UTF-8" %>
<%@taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <!--使用 text 标签输出国际化消息 -->
        <title><s:text name="loginTitle"/></title>
    </head>
    <body>
        <s:form action="checkLogin" method="post">
            <!--表单元素的 key 值与资源文件的 key 对应-->
            <s:textfield name="UserName" key="loginName" size="20"/>
            <s:password name="UserPassWord" key="loginPassword" size="20"/>
            <s:submit key="loginSubmit"/>
        </s:form>
    </body>
</html>
```

登录成功页面的代码如例 3-6 所示。

【例 3-6】　登录成功页面(loginSuccess.jsp)。

```
<%@page contentType="text/html; charset=UTF-8" %>
<%@taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <!--使用 text 标签输出国际化消息 -->
        <title><s:text name="successPage"/></title>
    </head>
    <body>
      <hr>
      <s:text name="loginName"/>:<s:property value="userName"/><br>
      <s:text name="loginPassword"/>:<s:property value="userPassWord"/>
    </body>
</html>
```

5. 编写业务控制器 Action

登录页面(login.jsp)对应的业务控制器是如例 3-7 所示的 LoginAction 类。

【例 3-7】 中英文登录页面对应的业务控制器(LoginAction.java)。

```
package loginAction;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;

public class LoginAction extends ActionSupport{
    private String userName;
    private String userPassWord;
    private String tip;
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName =userName;
    }
    public String getUserPassWord() {
        return userPassWord;
    }
    public void setUserPassWord(String userPassWord) {
        this.userPassWord =userPassWord;
    }
    public String getTip() {
        return tip;
    }
    public void setTip(String tip) {
        this.tip =tip;
    }
    public String execute() throws Exception
    {
        ActionContext ac=ActionContext.getContext();
        if (getUserName().equals("QQ")&&getUserPassWord().equals("123") )
        {
            ac.getSession().put("userName",getUserName());
            this.tip =getText("login.success",new String[] { this.userName });
            return SUCCESS;
        }
```

```
            else
            {
                return ERROR;
            }
        }
    }
```

6. 在 struts.xml 中配置 Action 与国际资源文件

修改配置文件 struts.xml，在配置文件中配置 Action 与国际化资源文件，参考例 3-4。

7. 项目部署和运行

项目部署后运行，如果操作系统是中文系统，运行 login.jsp 后，出现如图 3-5 所示的页面，在“用户名称”中输入 QQ、“用户密码”中输入 123 后，单击“登录”按钮，将跳转到如图 3-7所示的登录成功页面。

图 3-7　中文操作系统下的登录成功页面

如果使用的是英文操作系统或者设置浏览器语言为英文，运行后将出现如图 3-6 所示的英文登录页面。把浏览器设置为英文语言的步骤如下。

（1）选择“IE 浏览器”→“工具”→“Internet 选项”命令，如图 3-8 所示。

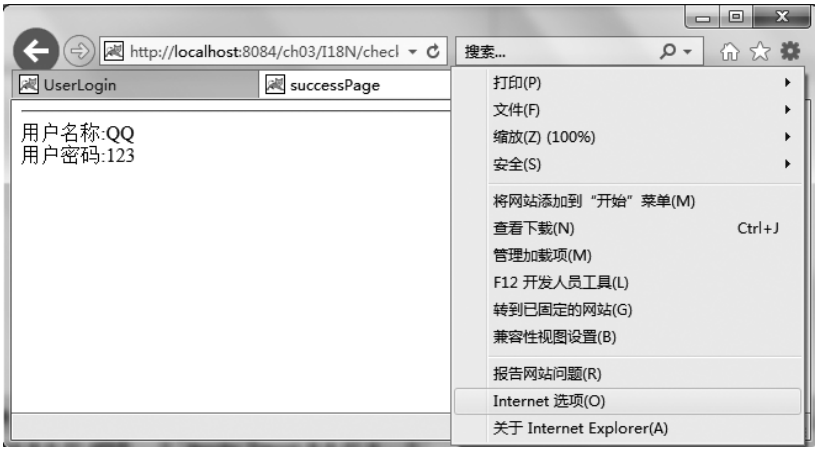

图 3-8　选择“Internet 选项”命令

（2）选择图 3-8 中的“Internet 选项”命令后，出现如图 3-9 所示的“Internet 属性”对话框。

（3）单击图 3-9 中的“语言”按钮，出现如图 3-10 所示的“语言首选项”对话框，其中“语

言”区域默认为“中文(简体,中国)[zh-CN]”,即在中文操作系统下,IE 浏览器默认的首选语言是中文,也可以添加其他语言。

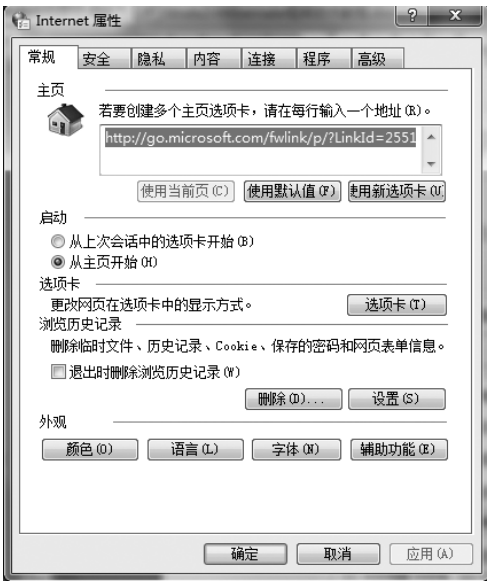

图 3-9 “Internet 属性”对话框

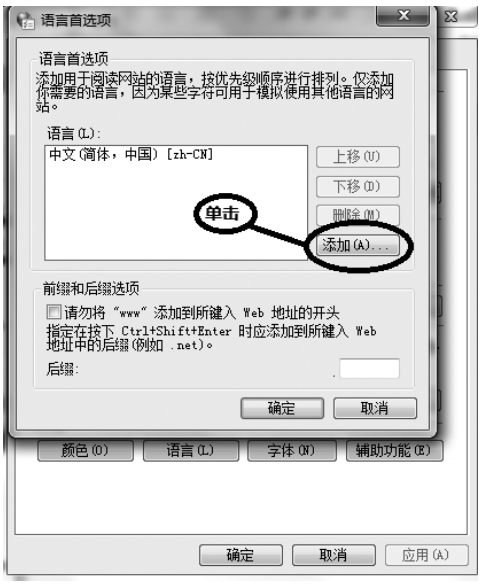

图 3-10 “语言首选项”对话框(一)

(4) 单击图 3-10 中的“添加”按钮,弹出“添加语言”对话框,如图 3-11 所示,找到“英语(美国)[en-US]”选项。

(5) 在图 3-11 中选定所需的语言后单击“确定”按钮,返回到“语言首选项”对话框,如图 3-12 所示,此时语言区域中已添加了“英语(美国)[en-US]”,选择“英语(美国)[en-US]”

图 3-11 “添加语言”对话框

图 3-12 “语言首选项”对话框(二)

选项，单击“上移”按钮，把英语作为浏览器的首选语言，最后单击“确定”按钮完成设置。

经过以上步骤配置后，浏览器的首选语言已是英语，浏览器在运行 JSP 页面(login.jsp)时，将优先使用英语，如果有英文国际化资源文件，将显示英文的登录页面，如图 3-6 所示。在图 3-6 所示页面中的 UserName 中输入 QQ 并在 UserPassword 中输入 123 后，单击 Login 按钮，页面将跳转到如图 3-13 所示的登录成功页面。

图 3-13　英文版的登录成功页面

美国国家航空航天局(National Aeronautics and Space Administration，NASA)因疏忽英制和公制计量单位的转换问题，导致火星气候宇宙飞船项目失败。应该从变化发展的客观实际出发，根据不同国家或地区的实际情况进行软件项目开发。

3.2　Struts2 的拦截器

拦截器是 Struts2 的核心组件，Struts2 的绝大部分功能都是通过拦截器来完成的，所以 Struts2 的很多功能都构建在拦截器的基础上。


Struts2 的拦截器

3.2.1　Struts2 拦截器的基础知识

拦截器(Interceptor)体系是 Struts2 的一个重要组成部分，正是大量的内置拦截器实现了 Struts2 的大部分操作。

当 StrutsPrepareAndExecuteFilter 拦截到用户请求后，大量的拦截器将会对用户请求进行处理，然后才调用用户自定义的 Action 类中的方法来处理请求。例如，params 拦截器将 HTTP 请求中的参数解析出来，并将这些解析出来的参数设置为 Action 的属性；servletConfig 拦截器直接将 HTTP 请求中的 HttpServletRequest 实例和 HttpServletResponse 实例传给 Action；国际化拦截器 I18N 对国际化资源进行操作；文件上传拦截器 fileUpload 将文件信息传给 Action。另外，还有数据校验拦截器对数据校验信息进行拦截。

对于 Struts2 的拦截器体系而言，当需要使用某个拦截器时，只需在配置文件 struts.xml 中配置即可；如果不需要使用该拦截器，只需在配置文件 struts.xml 中取消配置即可。Struts2 的拦截器可以理解为一种可插拔式的设计思想，所以 Struts2 框架具有非常好的可扩展性。

拦截器实现了面向切面编程(Aspect-Oriented Programming，AOP)的设计思想，拦截

是 AOP 的一种实现策略。

AOP 是目前软件开发中的一个热点，也是 Spring 框架中的一项重要内容。利用 AOP 可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高开发的效率。

3.2.2 Struts2 拦截器实现类

在项目开发中，Struts2 内置的拦截器可以完成项目的大部分功能，但有些与项目业务逻辑相关的通用功能则需要通过自定义拦截器来实现，如权限控制、用户输入内容的控制等。

自定义拦截器需要实现 Struts2 提供的 Interceptor 接口。通过实现该接口可以开发拦截器类。该接口的代码如例 3-8 所示。

【例 3-8】 Struts2 的拦截器接口(Interceptor.java)。

```
import com.opensymphony.xwork2.ActionInvocation;
import java.io.Serializable;

public interface Interceptor extends Serializable {
    void destroy();
    void init();
    String intercept(ActionInvocation invocation) throws Exception;
}
```

该接口提供了 3 个方法。

(1) destroy()方法：与 init()方法对应，用于在拦截器执行完之后释放 init()方法里打开的资源。

(2) init()方法：由拦截器在执行之前调用，主要用于初始化系统资源，如打开数据库资源等。

(3) intercept(ActionInvocation invocation)方法：拦截器的核心方法，实现具体的拦截操作，返回一个字符串作为逻辑视图。与 Action 一样，如果拦截器能够成功调用 Action，则 Action 中的 execute()方法返回一个字符串类型值，将其作为逻辑视图返回，否则，返回开发者自定义的逻辑视图。

在 Java 语言中，有时候通过一个抽象类来实现一个接口，在抽象类中提供该接口的空实现。这样在编写类时，可以直接继承该抽象类，不用实现那些不需要的方法。Struts2 框架也提供了一个抽象拦截器类(AbstractInterceptor)，该类对 init()和 destroy()方法进行空实现，因为很多时候实现拦截器都不需要申请资源。在开发自定义拦截器时，通过继承这个类开发起来简单方便。AbstractInterceptor 类的代码如例 3-9 所示。

【例 3-9】 Struts2 的抽象拦截器(AbstractInterceptor.java)。

```
import com.opensymphony.xwork2.ActionInvocation;

public abstract class AbstractInterceptor implements Interceptor {
    public void init()
    {
    }
    public void destroy()
```

```
    {
    }
    public abstract String intercept(ActionInvocation invocation) throws Exception;
}
```

3.2.3 Struts2 拦截器应用实例

本节项目是对自定义拦截器使用的练习。下面通过一个文字过滤项目来熟悉自定义拦截器的使用。

1. 项目介绍

该项目开发一个网上论坛过滤系统，如果网友发表不文明或者不和谐的语言，将通过拦截器对这些文字进行自动替代。该项目编写一个自定义拦截器(MyInterceptor.java)，代码如例 3-10 所示；项目有一个发表新闻评论的页面(news.jsp)，代码如例 3-11 所示；评论页面对应的业务控制器是 PublicAction 类，代码如例 3-13 所示；评论成功跳转到 success.jsp 页面，代码如例 3-12 所示。此外，还需要配置 web.xml，代码如例 1-3 所示；配置 struts.xml，代码如例 3-14所示。项目的文件结构如图 3-14 所示。

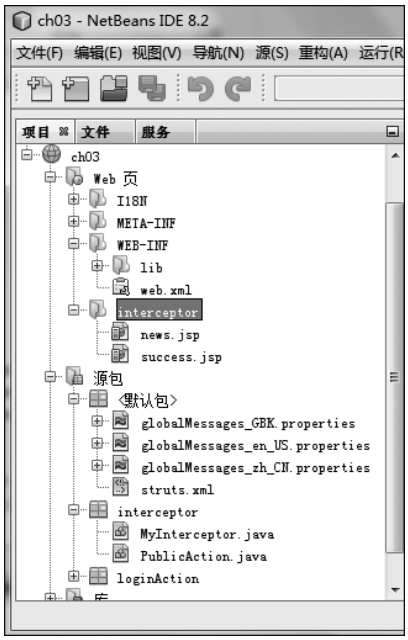

图 3-14 项目的文件结构

2. 在 web.xml 中配置核心控制器 StrutsPrepareAndExecuteFilter

参考 1.3.1 节中的例 1-3。

3. 编写自定义拦截器

编写一个自定义拦截器用于对发表的评论内容进行过滤，代码如例 3-10 所示。

【例 3-10】 自定义拦截器的编写(MyInterceptor.java)。

```
package interceptor;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class MyInterceptor extends AbstractInterceptor {
    public String intercept(ActionInvocation ai) throws Exception {
        //获取 Action 的实例
        Object object =ai.getAction();
        if (object!=null)
        {
            if(object instanceof PublicAction)
            {
                PublicAction ac=(PublicAction)object;
                //获取用户提交的评论内容
                String content =ac.getContent();
                //判断用户提交的评论内容是否有需要过滤的内容
```

```
                if(content.contains("讨厌"))
                {
                    //以"喜欢"代替要过滤的"讨厌"
                    content =content.replaceAll("讨厌", "喜欢");
                    //把替代后的评论内容设置为 Action 的评论内容
                    ac.setContent(content);
                 }
                //对象不空,继续执行
                 return ai.invoke();
            }else{
                //返回 Action 中的 LOGIN 逻辑视图字符串,参考 2.3.1 节
                 return Action.LOGIN;
            }
        }
        return Action.LOGIN;
        }
}
```

4. 编写视图组件(JSP 页面)发表评论

编写一个如图 3-15 所示的评论页面,代码如例 3-11 所示。

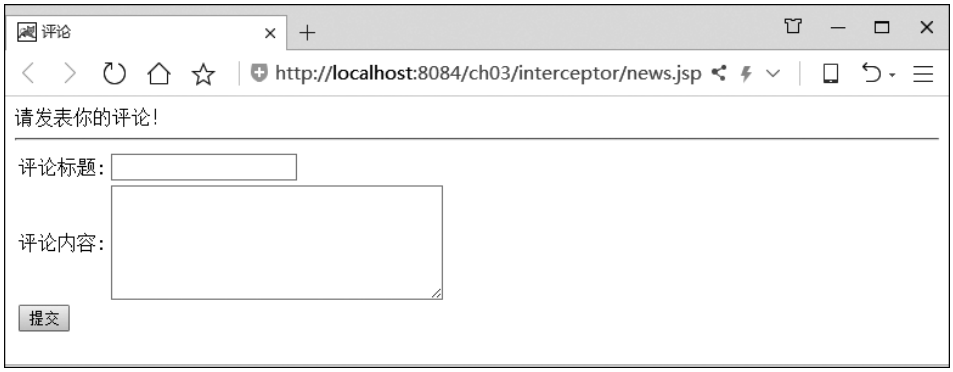

图 3-15　评论页面

【例 3-11】　评论页面(news.jsp)。

```
<%@page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
    <head>
        <title>评论</title>
    </head>
    <body>
        请发表你的评论!
        <hr>
        <s:form action="public" method="post">
            <s:textfield name="title" label="评论标题" />
            <s:textarea name="content" cols="36" rows="6" label="评论内容"/>
            <s:submit value="提交"/>
        </s:form>
    </body>
</html>
```

评论成功页面的代码如例 3-12 所示。

【例 3-12】 评论成功页面(success.jsp)。

```
<%@page contentType="text/html; charset=UTF-8" %
<%@taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <title>评论成功</title>
    </head>
    <body>
        评论如下:
        <hr>
        评论标题:<s:property value="title"/>
        <br>
        评论内容:<s:property value="content"/>
  </body>
</html>
```

5. 编写业务控制器 Action

news.jsp 对应的业务控制器是如例 3-13 所示的 PublicAction 类。

【例 3-13】 评论页面对应的业务控制器(PublicAction.java)。

```
package interceptor;
import com.opensymphony.xwork2.ActionSupport;

public class PublicAction extends ActionSupport{
    private String title;
    private String content;
    public void setTitle(String title) {
        this.title =title;
    }
    public String getTitle() {
        return title;
    }
    public void setContent(String content) {
        this.content =content;
    }
    public String getContent() {
        return content;
    }
    public String execute() {
        return SUCCESS;
    }
}
```

6. 在 struts.xml 中配置自定义拦截器和 Action

修改配置文件 struts.xml,在配置文件中配置拦截器和 Action,代码如例 3-14 所示。

【例 3-14】 配置拦截器和 Action(struts.xml)。

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <!--Configuration for the default package. -->
    <constant name="struts.custom.i18n.resources" value="globalMessages" />
    <constant name="struts.i18n.encoding" value="utf-8" />
    <package name="I18N" extends="struts-default">
        <!--Struts2 规定,对拦截器的配置应放到 Action 的配置之前。例如,在 3.1 节中对
        Action 已经配置,在 3.2 节中对拦截器配置时应把对拦截器的配置放到所有 Action 配
        置的前面,即在 package 中,拦截器的配置总是放到前面,Action 的配置放到后面-->
       <interceptors>
            <!--文字过滤拦截器配置,replace 是拦截器的名字-->
           <interceptor name="replace" class="interceptor.MyInterceptor" />
        </interceptors>
        <action name="checkLogin" class="loginAction.LoginAction">
            <result name="success">/I18N/loginSuccess.jsp</result>
            <result name="error">/I18N/login.jsp</result>
        </action>
        <!--文字过滤 Action 的配置-->
        <action name="public" class="interceptor.PublicAction">
            <result name="success">/interceptor/success.jsp</result>
            <result name="login">/interceptor/success.jsp</result>
            <!--Struts2 系统默认拦截器-->
            <interceptor-ref name="defaultStack" />
            <!--使用自定义拦截器-->
            <interceptor-ref name="replace"/>
        </action>
    </package>
</struts>
```

7. 项目部署和运行

项目部署后运行,运行效果如图 3-15 所示。输入完评论后单击“提交”按钮,如图 3-16 所示,跳转到评论成功页面,如图 3-17 所示。可以看出,经过拦截器拦截后文字被过滤。

图 3-16　输入评论

图 3-17　经过拦截器拦截以后的数据

3.3　Struts2 的输入校验

在网络上，由于用户对要输入数据格式理解的多样性，导致用户输入的数据与开发者的意图不一致。为了保证数据在存储过程中的正确性和一致性，必须对用户输入的信息进行校验。只有通过了严格校验的数据才能提高系统的安全性和健壮性，保证系统的正常运行。Struts2 框架内置了许多功能强大的输入校验器，内置的输入校验器基本上可以实现主要的输入验证功能，程序员不需要编写任何校验代码即可完成项目开发中遇到的常用输入校验。另外，为了扩展 Struts2 框架的输入校验功能，Struts2 框架允许使用validate()方法、validateXxx()方法和自定义校验器，实现对 Struts2 框架中输入验证的扩展。

3.3.1　Struts2 输入验证的基础知识

网络上，为了保证网站的稳定运行，良好的输入验证机制是前提，也是一个成熟系统的必备条件。对数据进行验证通常可以分为两部分：首先，验证用户输入数据的有效性；其次，在用户输入无效的数据后为用户提供友好的提示信息。


Struts2 输入验证的基础知识

1. 需要输入校验的原因

在互联网上，Web 站点是对外提供服务的，由于站点的开放性，Web 站点保存的数据主要是从客户端接收到的。输入数据的用户来自不同的行业，他们有不同的生活习惯、教育背景，从而不能绝对保证输入内容的正确性。例如，用户操作计算机不熟练、输入出错、网络问题或者恶意输入等，这些都可能导致数据的异常。如果对数据不加校验，有可能导致系统阻塞甚至系统崩溃。如图 3-18 所示页面展示的就是用户随意输入数据的一种表现。

在图 3-18 所示页面中，可能存在用户名和密码的位数过长、年龄和电话数据的输入不符合事实等数据异常情况。如果在数据库表的设计中没有定义那么长的字段，就可能会导致数据库异常。所以，必须对客户端输入的信息进行校验。

Java Web 项目中系统的输入校验方式有客户端校验和服务器端校验两种。Struts2 框架对用户输入数据的校验也可以分为两种：客户端校验和服务器端校验。为了实现完全的数据输入校验，需要将这两种验证方式紧密结合、互相协作。

2. 客户端校验

客户端校验可以在客户端通过 JavaScript 脚本或者 Ajax 对用户输入的数据进行基本

图 3-18　无输入验证的用户注册页面

校验。下面通过使用 JavaScript 脚本语言对客户端的输入数据进行验证。

1）项目介绍

本项目使用 JavaScript 脚本对注册页面进行验证。JavaScript 脚本写在＜head＞中，注册页面（register.jsp）的代码如例 3-15 所示；注册页面对应的业务控制器为 RegistAction 类，代码如例 3-17 所示；如果输入的数据验证成功，进入验证成功页面（success.jsp），代码如例 3-16 所示。此外，还需要配置 web.xml，代码如例 1-3 所示；配置 struts.xml，代码如例 3-18 所示。项目的文件结构如图 3-19 所示。

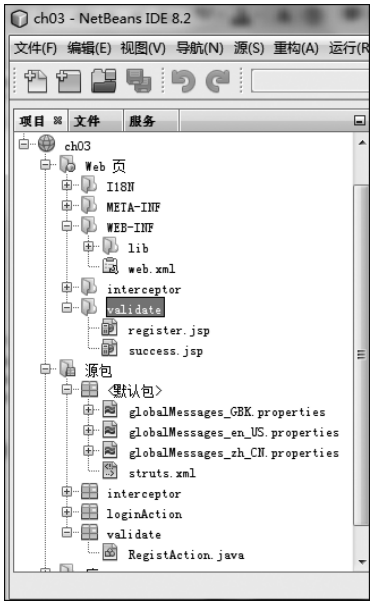

图 3-19　项目的文件结构

2）在 web.xml 中配置核心控制器 StrutsPrepareAndExecuteFilter

参考 1.3.1 节中的例 1-3。

3）编写视图组件（JSP 页面）

注册页面如图 3-20 所示。代码如例 3-15 所示。输入的注册数据如果通过验证则跳转到验证成功页面，代码如例 3-16 所示。

图 3-20　注册页面

【例 3-15】 具有 JavaScript 脚本验证功能的注册页面(register.jsp)。

```
<%@page contentType="text/html; charset=UTF-8" %>
<%@taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <title>用户注册页面</title>
        <!--检验输入表单数据的函数-->
        <script language="JavaScript">
            function trim(str)
            {
                //使用正则表达式去掉字符的前后空格
                return str.replace(/^\s*/,"").replace(/\s*$/,"");
            }
            function check(form)
            {
                //定义错误标志字符串
                var errorStr="";
                //获取表单的 4 个数据
                var userName=trim(form.userName.value);
                var userPassword=trim(form.userPassword.value);
                var userAge=trim(form.userAge.value);
                var userTelephone=trim(form.userTelephone.value);
                var pattern =/^\d{8}$/;
                //判断用户名是否为空
                if(userName==null||userName=="")
                {
                    errorStr="用户名不能为空!";
                }
                else if(userPassword.length>16||userPassword.length<6)
                {
                    errorStr="密码长度必须为 6~16";
                }
                else if(userAge>130||userAge<0)
                {
                    errorStr="年龄必须为 0~130";
                }
                else if(!pattern.test(userTelephone)){
                    errorStr="电话号码由 8 位阿拉伯数字组成!";
                }
                if(errorStr=="")
                {
                    return true;
                 }else
                {
                    alert(errorStr);
                    return false;
                }
            }
        </script>
    </head>
    <body>
        <center>
            请输入注册信息…
            <hr>
```

```
            <s:form action="register.action" method="post" onSubmit=
                "return check(this);">
                <table border="1">
                    <tr>
                        <td>
                            <s:textfield name="userName" label="姓名"
                            size="16"/>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <s:password name="userPassword" label="密码"
                            size="18"/>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <s:textfield name="userAge" label="年龄"
                            size="16"/>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <s:textfield name="userTelephone" label="电话"
                            size="16"/>
                        </td>
                    </tr>
                    <tr>
                        <td><s:submit value="提交"/></td>
                    </tr>
                </table>
            </s:form>
        </center>
    </body>
</html>
```

【例 3-16】 验证成功后的页面(success.jsp)。

```
<%@page contentType="text/html; charset=UTF-8" %>
<%@taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <title>验证成功</title>
    </head>
    <body>
        验证通过,用户信息如下:
        <hr>
        姓名:<s:property value="userName"/>
        <br>
        密码:<s:property value="userPassword"/>
        <br>
        年龄:<s:property value="userAge"/>
        <br>
        电话:<s:property value="userTelephone"/>
```

```
    </body>
</html>
```

4）编写业务控制器 Action

注册页面对应的业务控制器是 RegistAction 类，代码如例 3-17 所示。

【例 3-17】 注册页面对应的业务控制器(RegistAction.java)。

```
package validate;
import com.opensymphony.xwork2.ActionSupport;

public class RegistAction extends ActionSupport{
      private String userName;
      private String userPassword;
      private int userAge;
      private String userTelephone;
      public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName =userName;
    }
    public String getUserPassword() {
        return userPassword;
    }
    public void setUserPassword(String userPassword) {
        this.userPassword =userPassword;
    }
    public int getUserAge() {
        return userAge;
    }
    public void setUserAge(int userAge) {
        this.userAge =userAge;
    }
    public String getUserTelephone() {
        return userTelephone;
    }
    public void setUserTelephone(String userTelephone) {
        this.userTelephone =userTelephone;
    }
    public String execute(){
        return SUCCESS;
    }
}
```

5）修改 struts.xml，配置 Action

修改配置文件 struts.xml，代码如例 3-18 所示。

【例 3-18】 在 struts.xml 中配置 Action(struts.xml)。

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <constant name="struts.custom.i18n.resources" value="globalMessages" />
    <constant name="struts.i18n.encoding" value="utf-8" />
    <package name="I18N" extends="struts-default">
        <interceptors>
            <!--文字过滤拦截器配置,replace 是拦截器的名字-->
            <interceptor name="replace" class="interceptor.MyInterceptor" />
        </interceptors>
        <action name="checkLogin" class="loginAction.LoginAction">
            <result name="success">/I18N/loginSuccess.jsp</result>
            <result name="error">/I18N/login.jsp</result>
        </action>
        <!--文字过滤 Action 配置-->
        <action name="public" class="interceptor.PublicAction">
            <result name="success">/interceptor/success.jsp</result>
            <result name="login">/interceptor/success.jsp</result>
             <!--Struts2 系统默认拦截器-->
            <interceptor-ref name="defaultStack" />
            <!--使用自定义拦截器-->
            <interceptor-ref name="replace"/>
        </action>
        <action name="register" class="validate.RegistAction">
            <result name="input">/validate/register.jsp</result>
            <result name="success">/validate/success.jsp</result>
        </action>
    </package>
</struts>
```

6）项目部署和运行

注册页面运行效果如图 3-20 所示。如果输入的密码长度不符合验证要求将出现如图 3-21 所示的页面，如果数据符合验证要求，将出现如图 3-22 所示的页面。

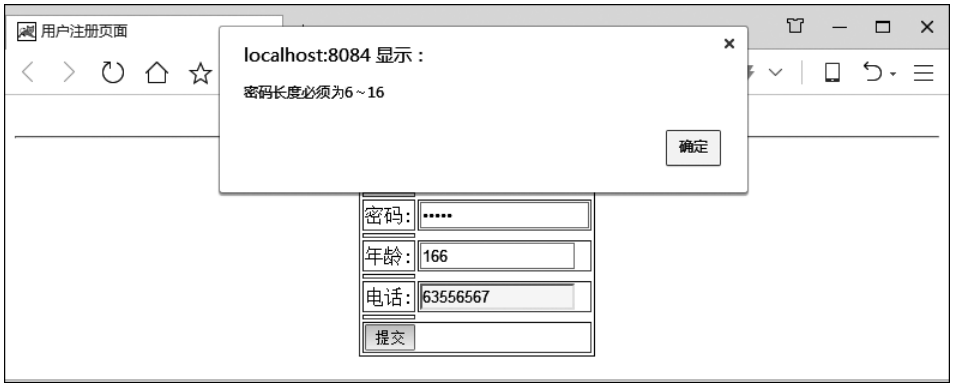

图 3-21　输入的密码长度不符合验证要求

从上面的实例可以看出，当客户端访问注册页面时，如果输入的数据不符合验证要求就无法实现注册功能。但是服务器发送到客户端的页面是静态页面，可以很方便地在浏览器选择菜单“查看”→“源文件”命令，查看这些代码并修改代码，这样就能轻易绕过客户端校