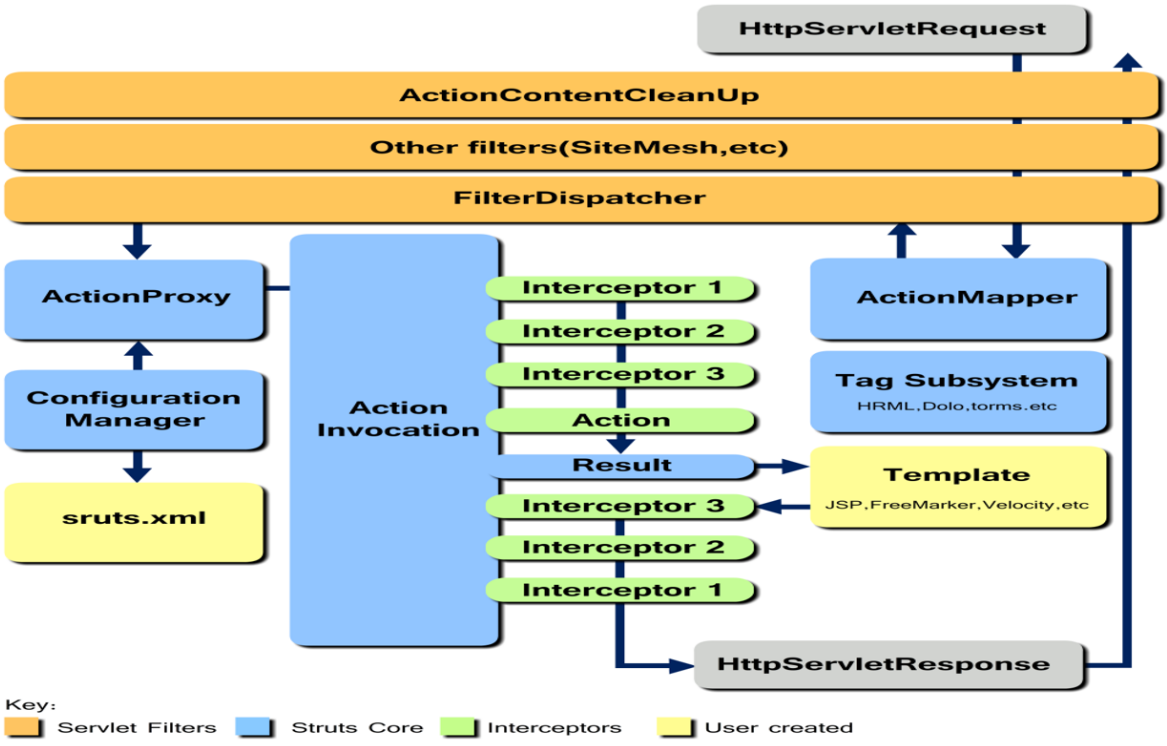
1 Struts2 拦截器

1.1 Struts2 架构

1.1.1 Struts2体系结构图



1.1.2 执行流程说明

用户发送request请求，请求会先经过一系列的过滤器，过滤器放行后经过核心控制器FilterDispatcher，核心控制器会使用Action代理对象读取struts.xml配置文件，然后创建struts2控制器Action的实例，进入Action控制器之前会先经过一系列的拦截器，拦截放行后进入Action，根据返回的结果字符串result会选择相应的视图，在响应到客户端之前也会经过一系列的拦截器。

例如：用户登录的流程

1.填写账号密码后点击提交按钮，此时发送登录请求

2.请求会进入核心控制器，进入核心控制器之前会先经过一些过滤器

3.核心控制器会通过Action代理对象读取请求（读取struts.xml文件）

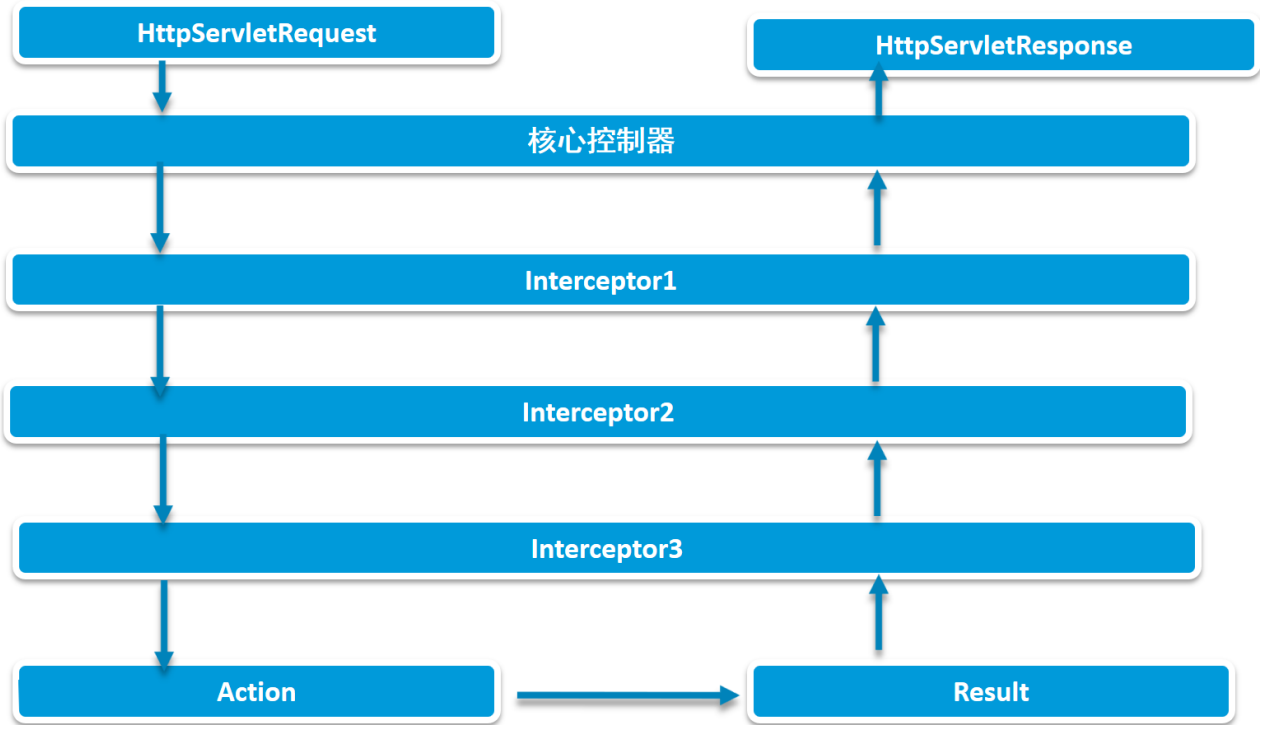
4.进入Action之前会经过一系列的拦截器

5.进入Action控制器执行后，根据返回的结果字符串选择相应的视图（登录成功，去到首页，登录失败，回到登录页）

1.1.3 Struts2 核心接口和类

名称	作用
ActionMapper	根据请求的URI查找是否存在对应Action调用
ActionMapping	保存调用Action的映射信息，如namespace、name等
ActionProxy	在XWork和真正的Action之间充当代理
ActionInvocation	表示Action的执行状态，保存拦截器、Action实例
Interceptor	在请求处理之前或者之后执行的Struts 2组件

1.1.4 Struts2流程简图



1.2 Struts2的拦截器

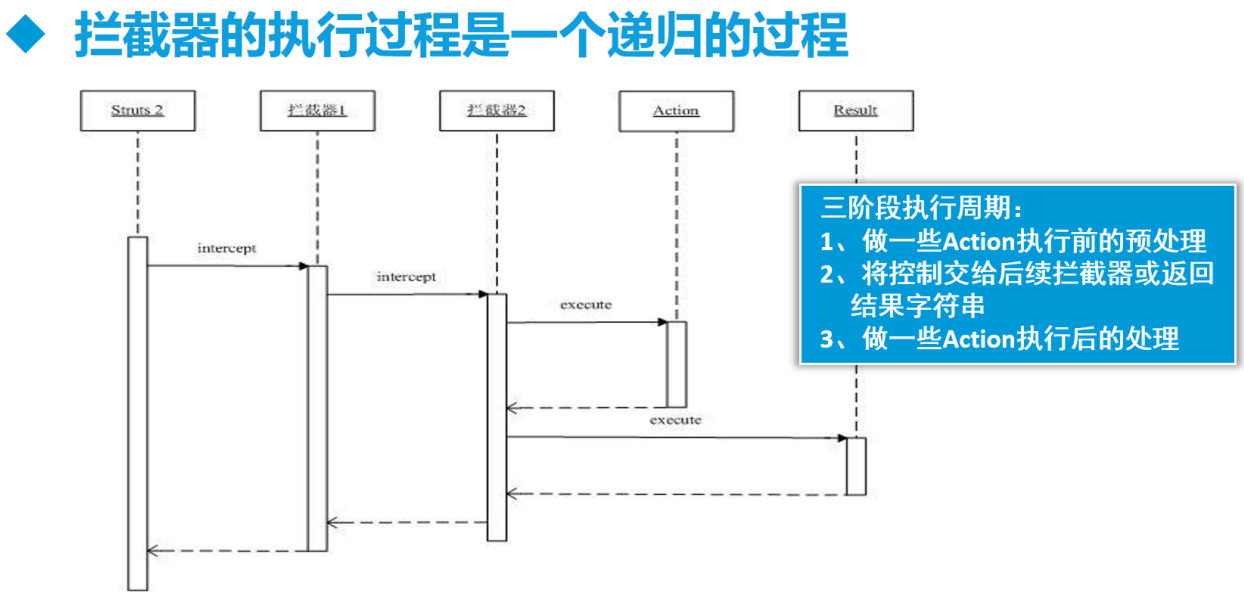
1.2.1 为什么要使用拦截器

- 早期MVC框架将一些通用操作硬编码在核心控制器中，致使框架灵活性不足、可扩展性降低
- Struts 2将核心功能放到多个拦截器中实现，拦截器可自由选择和组合，增强了灵活性，有利于系统的解耦

1.2.2 拦截器简介

- Struts 2大多数核心功能是通过拦截器实现的，每个拦截器完成某项功能
- 拦截器方法在Action执行之前和之后执行
- 拦截器栈
  - 从结构上看，拦截器栈相当于多个拦截器的组合
  - 在功能上看，拦截器栈也是拦截器
- 拦截器与过滤器原理很相似
- 为Action提供附加功能时，无需修改Action代码，使用拦截器来提供

1.2.3 拦截器工作原理



1.2.4 拦截器的使用

1.2.4.1 创建自定义拦截器类

注意：自定义拦截器类可以继承 `AbstractInterceptor` 或 `MethodFilterInterceptor`

```java
package com.bdqn.interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;


/**
 * 自定义拦截器
 */
public class TestInterceptor extends AbstractInterceptor {


    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        System.out.println("进入第1个拦截器");
        //将请求放行
        String result = invocation.invoke();
        System.out.println("退出第1个拦截器");
        //将结果返回出去
        return result;
    }
}
```

1.2.4.2 定义拦截器

注意：在struts.xml配置文件中的<package>标签定义拦截器

```xml
<!-- 拦截器模块 -->
<package name="hello" extends="struts-default">

    <!-- 定义拦截器 -->
    <interceptors>
        <!-- 单个拦截器 -->
        <interceptor name="test1" class="com.bdqn.interceptor.TestInterceptor"/>
    </interceptors>

</package>
```

1.2.4.3 使用拦截器

```xml
<!-- 拦截器模块 -->
<package name="hello" extends="struts-default">

    <!-- 定义拦截器 -->
    <interceptors>
        <!-- 单个拦截器 -->
        <interceptor name="test1" class="com.bdqn.interceptor.TestInterceptor"/>
    </interceptors>

```

```
10
11      <action name="hello" class="com.bdqn.web.HelloAction" method="helloWorld">
12          <result>/success.jsp</result>
13          <!-- 引用拦截器 -->
14          <interceptor-ref name="test1"/>
15          <!-- 引用默认的拦截器 -->
16          <interceptor-ref name="defaultStack"/>
17      </action>
18  </package>
```

1.2.5 拦截器的配置

第一步：定义拦截器

`<interceptors>`

`<interceptor name="拦截器名称" class="拦截器类路径">`

`</interceptors>`

第二步：引用拦截器

`<interceptor-ref name="引用的拦截器名称">`

> 拦截器配置如下：

```
<!-- 公共配置 -->
<package name="default" namespace="/" extends="struts-default"...>

<!-- 拦截器模块 -->
<package name="hello" extends="default">

    <!-- 1.定义拦截器 -->
    <interceptors>
        <!-- 单个拦截器 -->
        <interceptor name="test1" class="com.bdqn.interceptor.TestInterceptor"/>
    </interceptors>

    <action name="hello" class="com.bdqn.web.HelloAction" method="helloWorld">
        <result>/success.jsp</result>
        <!-- 2.引用拦截器 -->
        <interceptor-ref name="test1"/>
        <!-- 3.引用默认的拦截器 -->
        <interceptor-ref name="defaultStack"/>
    </action>
</package>
```

1.2.6 Struts2的默认拦截器

- **params拦截器**
  - 负责将请求参数设置为Action属性
- **servletConfig拦截器**
  - 将源于Servlet API的各种对象注入到Action
- **fileUpload拦截器**
  - 对文件上传提供支持
- **exception拦截器**
  - 捕获异常，并且将异常映射到用户自定义的错误页面
- **validation拦截器**
  - 调用验证框架进行数据验证
- **workflow拦截器**
  - 调用Action类的validate()，执行数据验证

1.2.7 案例：自定义登录拦截器

1.2.7.1 需求

> 准备UserAction控制器，该控制器中存在4个方法，分别是 `find()`，`add()`，`update()`，`delete()`,这4个方法没有登录的情况下不允许直接访问(必须进行登录成功后才能够访问)

1.2.7.2 环境准备

> 准备UserAction控制器

```
1 package com.bdqn.web;
2
3 import com.opensymphony.xwork2.ActionSupport;
```

```java
import org.apache.struts2.ServletActionContext;

public class UserAction extends ActionSupport {

    private String userName;
    private String password;

    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;
    }

    //登录方法，该方法不允许拦截
    public String login(){
        if(userName.equals("admin") && password.equals("admin")){
            //保存session

 ServletActionContext.getRequest().getSession().setAttribute("loginUser",userName);
            return SUCCESS;//成功
        }
        return LOGIN;//失败
    }


    //以下4个方法需要进行身份验证

    //查询
    public String find(){
        return SUCCESS;
    }
    //添加
    public String add(){
        return SUCCESS;
    }
    //修改
    public String update(){
        return SUCCESS;
    }
    //删除
    public String delete(){
        return SUCCESS;
    }

}
```

> 先测试没有添加拦截器的效果，在没有添加拦截器的情况下是能够访问所有方法的

1.2.7.3 创建登录验证的拦截器类

```java
package com.bdqn.interceptor;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.MethodFilterInterceptor;
import org.apache.struts2.ServletActionContext;

import javax.servlet.http.HttpSession;

/**
 * 自定义登录拦截器类
 */
public class SystemInterceptor extends MethodFilterInterceptor {
    @Override
    protected String doIntercept(ActionInvocation invocation) throws Exception {
        //获取Session
        HttpSession session = ServletActionContext.getRequest().getSession();
        //判断session是否为空
        if(session.getAttribute("loginUser") == null){ //为空表示没有登录，没有登录则返回登录页面
            //没有登录则返回登录页面
            return Action.LOGIN;
        }
        //不为空则表示已经登录成功，直接放行
        return invocation.invoke();
    }
}
```

1.2.7.4 配置拦截器

> 在struts.xml配置文件中配置，代码如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
        "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>



    <!-- 开启动态方法调用 -->
    <constant name="struts.enable.DynamicMethodInvocation" value="true"/>

    <!-- 公共配置 -->
    <package name="default" namespace="/" extends="struts-default">
```

```
        <!-- 设置白名单 -->
        <global-allowed-methods>regex:.*</global-allowed-methods>
    </package>

    <!-- 登录验证模块 -->
    <package name="user" namespace="/" extends="default">

        <!-- 定义拦截器 -->
        <interceptors>
            <!-- 注入登录拦截器类 -->
            <interceptor name="loginInterceptor"
class="com.bdqn.interceptor.SystemInterceptor"/>

            <!-- 1.定义拦截器栈 -->
            <interceptor-stack name="myLoginStack">
                <!-- 2.引用默认的拦截器 -->
                <interceptor-ref name="defaultStack"/>
                <!-- 3.引用自定义的登录拦截器 -->
                <interceptor-ref name="loginInterceptor">
                    <!-- 注意：includeMethods属性的优先级高于excludeMethods属性 -->
                    <param name="includeMethods">add*,update*</param>
                    <!-- excludeMethods：设置不拦截的方法，多个方法名之间使用逗号隔开 -->
                    <param name="excludeMethods">login</param>
                    <!-- includeMethods:设置拦截的方法，多个方法名之间使用逗号隔开 -->
                </interceptor-ref>
            </interceptor-stack>
        </interceptors>

        <!-- 使用通配符访问 -->
        <action name="user_*" class="com.bdqn.web.UserAction" method="{1}">
            <!-- 登录成功 -->
            <result name="success">/success.jsp</result>
            <!-- 登录失败 -->
            <result name="login">/login.jsp</result>

            <!-- 4.引用拦截器 -->
            <interceptor-ref name="myLoginStack"/>


        </action>
    </package>

</struts>
```

1.2.7.5 login.jsp

```
<%@taglib prefix="s" uri="/struts-tags" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户登录</title>
</head>
<body>

<s:fielderror/>

```

```
<form action="user_login.action" method="post">
    <div>
        <label>用户名: </label>
        <input type="text" name="userName">
        <font color="red"><s:fielderror fieldName="userName"/></font>
    </div>
    <div>
        <label>密码: </label>
        <input type="password" name="password">
        <font color="red"><s:fielderror fieldName="password"/></font>
    </div>
    <div>
        <input type="submit" value="登录">
    </div>
</form>

</body>
</html>
```

1.2.7.6 测试

先不进行登录，访问http://localhost:8080/user_delete.action请求，如果能够跳转到login.jsp页面，说明拦截器配置生效。

进行登录操作，登录的请求是不会被拦截的，如果登录请求被拦截，说明拦截器的配置出错。

进行登录操作，登录成功后，访问http://localhost:8080/user_delete.action请求，能够去到成功success.jsp页面，说明拦截器配置是正确的。

2 文件上传

2.1 Commons-FileUpload组件

Commons是Apache开放源代码组织的一个Java子项目，其中的FileUpload是用来处理HTTP文件上传的子项目

Commons-FileUpload组件特点

1. 使用简单：可以方便地嵌入到JSP文件中，编写少量代码即可完成文件的上传功能
2. 能够全程控制上传内容
3. 能够对上传文件的大小、类型进行控制

注意：在struts2中，`struts-core`依赖已经包含`commona-fileupload`组件的相关依赖，所以在struts2的项目中无需导入`commona-fileupload`组件的相关依赖。

2.2 单文件上传

2.2.1 upload.jsp

页面三要素：

1. form表单的提交方式必须是`post`提交
2. form表单必须设置`enctype="multipart/form-data"`属性
3. form表单需要提供文件域组件(`type="file"`)

```
<form action="upload.action" method="post" enctype="multipart/form-data">
    <p>选择文件: <input type="file" name="upload"></p>
    <p>
        <input type="submit">
    </p>
</form>
```

2.2.2 Action控制器

在Action控制器中需要提供3个成员变量，分别是文件、文件类型、文件名称，代码如下所示：

```
package com.bdqn.web;

import com.opensymphony.xwork2.ActionSupport;


import java.io.File;


public class UploadAction extends ActionSupport {

    //文件
    private File upload;//属性名必须与input标签的name属性值一致
    //文件类型
    private String uploadContentType;//input标签的name属性值 + ContentType（后缀）
    //文件名称
    private String uploadFileName;////input标签的name属性值 + FileName（后缀）

    public File getUpload() {
        return upload;
    }

```

```
    public void setUpload(File upload) {
        this.upload = upload;
    }

    public String getUploadContentType() {
        return uploadContentType;
    }

    public void setUploadContentType(String uploadContentType) {
        this.uploadContentType = uploadContentType;
    }

    public String getUploadFileName() {
        return uploadFileName;
    }

    public void setUploadFileName(String uploadFileName) {
        this.uploadFileName = uploadFileName;
    }
}
```

2.2.2.1 通过IO流上传

```
/**
 * 方法一：使用IO流实现文件上传
 * @return
 */
public String uploadFileByIO(){
    //1.获取文件上传的目标地址
    String path = "E:/upload";
    FileInputStream fis = null;
    FileOutputStream fos = null;
    try {
        //2.通过输入流读取文件
        fis = new FileInputStream(upload);
        //3.获取文件的后缀名
        String suffix = FilenameUtils.getExtension(uploadFileName);
        //4.将文件重命名
        String newFileName = UUID.randomUUID()+"."+suffix;
        //5.通过输出流将文件保存到服务器
        fos = new FileOutputStream(path+File.separator +newFileName);
        //6.通过工具类保存文件
        IOUtils.copy(fis,fos);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            //关闭资源：先用后关
            if(fos!=null){
                fos.close();
            }
            if(fis!=null){
                fis.close();
            }
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return SUCCESS;
}
```

2.2.2.2 通过FileUtils工具类上传

```
/**
 * 方法2:通过FileUtils工具类上传
 * @return
 */
public String uploadSimple(){
    try {
        //1.获取文件上传的目标地址
        String path = "E:/upload";
        //2.获取文件的后缀名
        String suffix = FilenameUtils.getExtension(uploadFileName);
        //3.将文件重命名
        String newFileName = UUID.randomUUID()+"."+suffix;
        //4.构建文件路径及文件的相关信息
        String realPath = path+File.separator +newFileName;
        //5.借助工具类保存文件
        FileUtils.copyFile(upload,new File(realPath));
    } catch (IOException e) {
        e.printStackTrace();
    }
    return SUCCESS;
}
```

2.2.3 struts.xml

```
<!-- 文件上传模块 -->
<package name="upload" extends="default">
    <action name="upload" class="com.bdqn.web.UploadAction" method="uploadSimple">
        <result>/success.jsp</result>
    </action>
</package>
```

注意：在struts2中，默认上传的文件大小不得超过2M，可以通过以下配置进行调整

```
<!-- 修改文件上传的限制大小，单位：字节byte -->
<!-- 修改大小为50M -->
<constant name="struts.multipart.maxSize" value="52428800"/>
```

2.3 多文件上传

2.3.1 mupload.jsp

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>单文件上传</title>
</head>
<body>

<form action="mupload.action" method="post" enctype="multipart/form-data">
    <p>选择文件: <input type="file" name="upload"></p>
    <p>选择文件: <input type="file" name="upload"></p>
    <p>
        <input type="submit">
    </p>
</form>

</body>
</html>
```

2.3.2 Action控制器

```java
package com.bdqn.web;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.UUID;

public class MoreUploadAction extends ActionSupport {

    //文件
    private File[] upload;//属性名必须与input标签的name属性值一致
    //文件类型
    private String[] uploadContentType;//input标签的name属性值 + ContentType（后缀）
    //文件名称
    private String[] uploadFileName;////input标签的name属性值 + FileName（后缀）


    public File[] getUpload() {
        return upload;
    }

    public void setUpload(File[] upload) {
        this.upload = upload;
    }
```

```
31
32      public String[] getUploadContentType() {
33          return uploadContentType;
34      }
35
36      public void setUploadContentType(String[] uploadContentType) {
37          this.uploadContentType = uploadContentType;
38      }
39
40      public String[] getUploadFileName() {
41          return uploadFileName;
42      }
43
44      public void setUploadFileName(String[] uploadFileName) {
45          this.uploadFileName = uploadFileName;
46      }
47  }
```

2.3.2.1 方法1

```
1   package com.bdqn.web;
2
3   import com.opensymphony.xwork2.ActionSupport;
4   import org.apache.commons.io.FileUtils;
5   import org.apache.commons.io.FilenameUtils;
6   import org.apache.commons.io.IOUtils;
7
8   import java.io.File;
9   import java.io.FileInputStream;
10  import java.io.FileOutputStream;
11  import java.io.IOException;
12  import java.util.UUID;
13
14  public class MoreUploadAction extends ActionSupport {
15
16      //文件
17      private File[] upload;//属性名必须与input标签的name属性值一致
18      //文件类型
19      private String[] uploadContentType;//input标签的name属性值 + ContentType（后缀）
20      //文件名称
21      private String[] uploadFileName;////input标签的name属性值 + FileName（后缀）
22
23
24      /**
25       * 方法一：使用IO流实现文件上传
26       * @return
27       */
28      public String uploadFileByIO(){
29          //1.获取文件上传的目标地址
30          String path = "E:/upload";
31          FileInputStream fis = null;
32          FileOutputStream fos = null;
33
34          for (int i = 0; i < upload.length; i++) {
35              try {
36                  //2.通过输入流读取文件
```

```
                fis = new FileInputStream(upload[i]);
                //3.获取文件的后缀名
                String suffix = FilenameUtils.getExtension(uploadFileName[i]);
                //4.将文件重命名
                String newFileName = UUID.randomUUID()+"."+suffix;
                //5.通过输出流将文件保存到服务器
                fos = new FileOutputStream(path+File.separator +newFileName);
                //6.通过工具类保存文件
                IOUtils.copy(fis,fos);
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    //关闭资源：先用后关
                    if(fos!=null){
                        fos.close();
                    }
                    if(fis!=null){
                        fis.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }


        return SUCCESS;
    }

    public File[] getUpload() {
        return upload;
    }

    public void setUpload(File[] upload) {
        this.upload = upload;
    }

    public String[] getUploadContentType() {
        return uploadContentType;
    }

    public void setUploadContentType(String[] uploadContentType) {
        this.uploadContentType = uploadContentType;
    }

    public String[] getUploadFileName() {
        return uploadFileName;
    }

    public void setUploadFileName(String[] uploadFileName) {
        this.uploadFileName = uploadFileName;
    }
}
```

2.3.2.2 方法2

```java
package com.bdqn.web;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.UUID;

public class MoreUploadAction extends ActionSupport {

    //文件
    private File[] upload;//属性名必须与input标签的name属性值一致
    //文件类型
    private String[] uploadContentType;//input标签的name属性值 + ContentType（后缀）
    //文件名称
    private String[] uploadFileName;////input标签的name属性值 + FileName（后缀）

    /**
     * 方法2
     * @return
     */
    public String uploadSimple(){

        for (int i = 0; i < upload.length; i++) {
            try {
                //1.获取文件上传的目标地址
                String path = "E:/upload";
                //2.获取文件的后缀名
                String suffix = FilenameUtils.getExtension(uploadFileName[i]);
                //3.将文件重命名
                String newFileName = UUID.randomUUID()+"."+suffix;
                //4.构建文件路径及文件的相关信息
                String realPath = path+File.separator +newFileName;
                //5.借助工具类保存文件
                FileUtils.copyFile(upload[i],new File(realPath));
            } catch (IOException e) {
                e.printStackTrace();
            }
        }


        return SUCCESS;
    }

    public File[] getUpload() {
        return upload;
    }

    public void setUpload(File[] upload) {
        this.upload = upload;
```

```
    }

    public String[] getUploadContentType() {
        return uploadContentType;
    }

    public void setUploadContentType(String[] uploadContentType) {
        this.uploadContentType = uploadContentType;
    }

    public String[] getUploadFileName() {
        return uploadFileName;
    }

    public void setUploadFileName(String[] uploadFileName) {
        this.uploadFileName = uploadFileName;
    }
}
```

2.3.3 struts.xml

```
<!-- 文件上传模块 -->
<package name="upload" extends="default">
    <!-- 多文件上传 -->
    <action name="mupload" class="com.bdqn.web.MoreUploadAction" method="uploadSimple">
        <result>/success.jsp</result>
    </action>
</package>
```

3 文件下载

3.1 Stream结果类型

名称	作用
contentType	设置发送到浏览器的**MIME**类型
contentLength	设置文件的大小
contentDisposition	设置响应的**HTTP**头信息中的**Content-Disposition**参数的值
inputName	指定**Action**中提供的**inputStream**类型的属性名称
bufferSize	设置读取和下载文件时的缓冲区大小

3.2 文件下载案例

3.2.1 页面

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>文件下载</title>
</head>
<body>

<a href="download.action?fileName=hello.jpg">文件下载(英文名称文件)</a>
<hr>
<a href="download.action?fileName=你好.jpg">文件下载(中文名称文件)</a>

</body>
</html>
```

3.2.2 控制器

```java
package com.bdqn.web;

import com.opensymphony.xwork2.ActionSupport;

import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.net.URLEncoder;

public class DownloadAction extends ActionSupport {

    //输入流：读取要下载的文件信息
    private InputStream inputStream;
    //文件名称
    private String fileName;//与a标签的参数名称保持一致

    /**
     * 文件下载
     * @return
     */
    public String download(){
        try {
            //1.指定文件下载的地址
            String path = "E:/upload/"+fileName;
            //指定编码格式
            fileName = URLEncoder.encode(fileName,"UTF-8");
            //2.通过输入流读取并下载文件
            inputStream = new BufferedInputStream(new FileInputStream(path));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return SUCCESS;
    }
```

```
35
36
37      public InputStream getInputStream() {
38          return inputStream;
39      }
40
41      public void setInputStream(InputStream inputStream) {
42          this.inputStream = inputStream;
43      }
44
45      public String getFileName() {
46          return fileName;
47      }
48
49      public void setFileName(String fileName) {
50          this.fileName = fileName;
51      }
52  }
```

3.2.3 struts.xml

```
1   <package name="download" extends="default">
2       <action name="download" class="com.bdqn.web.DownloadAction" method="download">
3           <result name="success" type="stream">
4               <!-- 设置文件下载的类型 -->
5               <param name="contentType">application/octet-stream</param>
6               <!-- 设置参数名称inputName,参数值是inputStream, 这个inputStream是DownloadAction的
    成员变量名称 -->
7               <param name="inputName">inputStream</param>
8               <!-- 设置以附件的形式下载 -->
9               <param name="contentDisposition">attachment;filename="${fileName}"</param>
10              <!-- 缓冲区大小 -->
11              <param name="bufferSize">4096</param>
12          </result>
13      </action>
14  </package>
```