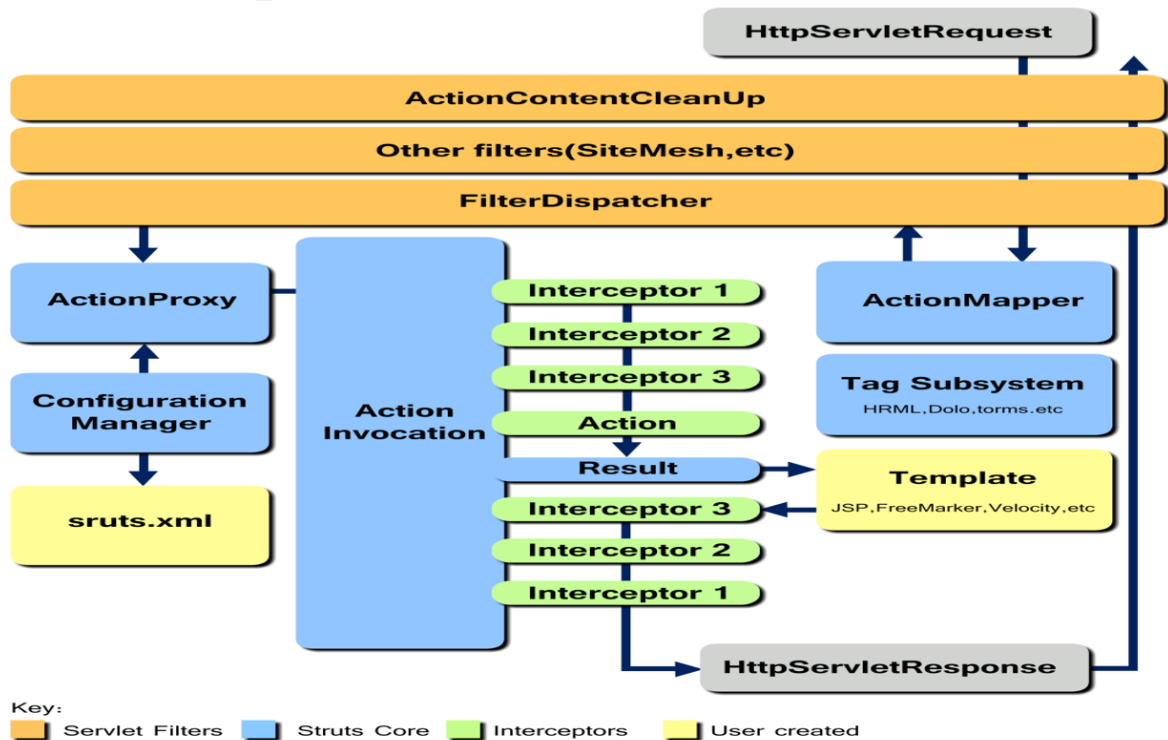


# 1 Struts2 拦截器

## 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

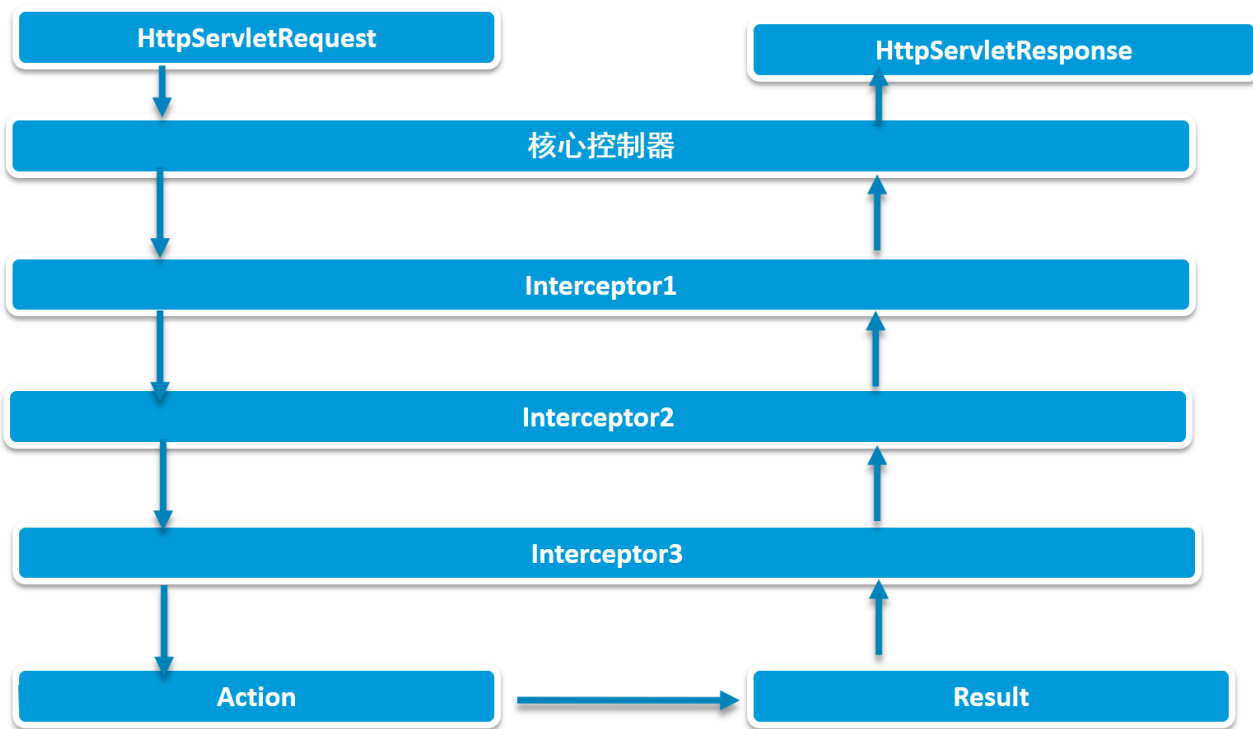
- 1.填写账号密码后点击提交按钮，此时发送登录请求
- 2.请求会进入核心控制器，进入核心控制器之前会先经过一些过滤器
- 3.核心控制器会通过Action代理对象读取请求（读取struts.xml文件）
- 4.进入Action之前会经过一系列的拦截器

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

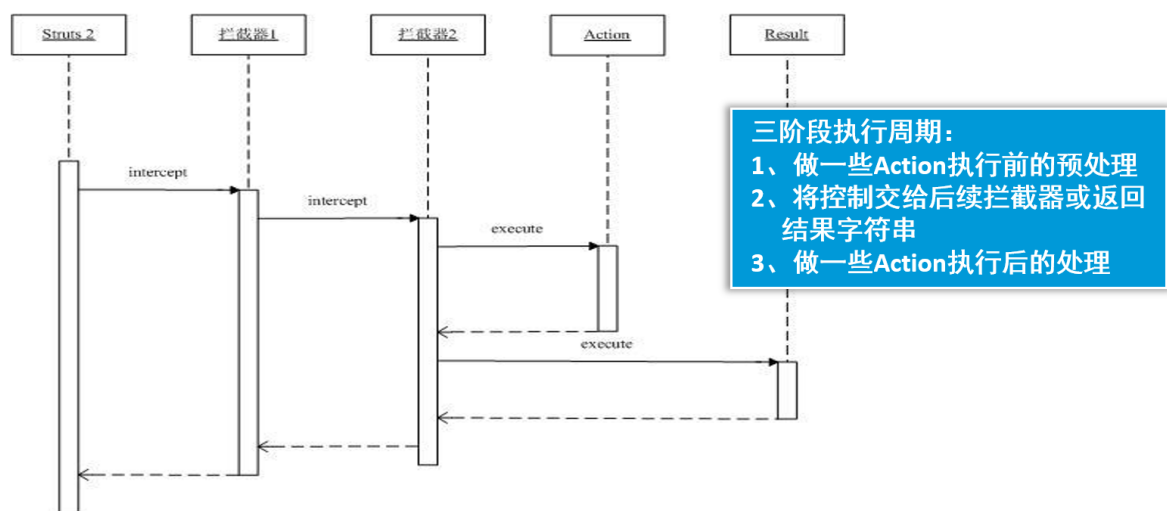
- 早期MVC框架将一些通用操作硬编码在核心控制器中，致使框架灵活性不足、可扩展性降低
- Struts 2将核心功能放到多个拦截器中实现，拦截器可自由选择和组合，增强了灵活性，有利于系统的解耦

### 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

#### ◆ 拦截器的执行过程是一个递归的过程



### 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`

```
1 package com.bdqn.interceptor;
2
3 import com.opensymphony.xwork2.ActionInvocation;
4 import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
5
6
7 /**
8  * 自定义拦截器
9  */
10 public class TestInterceptor extends AbstractInterceptor {
11
12
13     @Override
14     public String intercept(ActionInvocation invocation) throws Exception {
15         System.out.println("进入第1个拦截器");
16         //将请求放行
17         String result = invocation.invoke();
18         System.out.println("退出第1个拦截器");
19         //将结果返回出去
20         return result;
21     }
22 }
```

#### 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

```
1 <!-- 拦截器模块 -->
2 <package name="hello" extends="struts-default">
3
4     <!-- 定义拦截器 -->
5     <interceptors>
6         <!-- 单个拦截器 -->
7         <interceptor name="test1" class="com.bdqn.interceptor.TestInterceptor"/>
8     </interceptors>
9
10 </package>
```

#### 1.2.4.3 使用拦截器

```
1 <!-- 拦截器模块 -->
2 <package name="hello" extends="struts-default">
3
4     <!-- 定义拦截器 -->
5     <interceptors>
6         <!-- 单个拦截器 -->
7         <interceptor name="test1" class="com.bdqn.interceptor.TestInterceptor"/>
8     </interceptors>
9
```

```

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;
```

```
4 import org.apache.struts2.ServletActionContext;
5
6 public class UserAction extends ActionSupport {
7
8     private String userName;
9     private String password;
10
11     public String getUserName() {
12         return userName;
13     }
14
15     public void setUserName(String userName) {
16         this.userName = userName;
17     }
18
19     public String getPassword() {
20         return password;
21     }
22
23     public void setPassword(String password) {
24         this.password = password;
25     }
26
27     //登录方法, 该方法不允许拦截
28     public String login(){
29         if(userName.equals("admin") && password.equals("admin")){
30             //保存session
31
32             ServletActionContext.getRequest().getSession().setAttribute("loginUser",userName);
33             return SUCCESS;//成功
34         }
35         return LOGIN;//失败
36     }
37
38     //以下4个方法需要进行身份验证
39
40     //查询
41     public String find(){
42         return SUCCESS;
43     }
44     //添加
45     public String add(){
46         return SUCCESS;
47     }
48     //修改
49     public String update(){
50         return SUCCESS;
51     }
52     //删除
53     public String delete(){
54         return SUCCESS;
55     }
56
57 }
```

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

```
1 package com.bdqn.interceptor;
2
3 import com.opensymphony.xwork2.Action;
4 import com.opensymphony.xwork2.ActionInvocation;
5 import com.opensymphony.xwork2.interceptor.MethodFilterInterceptor;
6 import org.apache.struts2.ServletActionContext;
7
8 import javax.servlet.http.HttpSession;
9
10 /**
11  * 自定义登录拦截器类
12  */
13 public class SystemInterceptor extends MethodFilterInterceptor {
14     @Override
15     protected String doIntercept(ActionInvocation invocation) throws Exception {
16         //获取Session
17         HttpSession session = ServletActionContext.getRequest().getSession();
18         //判断session是否为空
19         if(session.getAttribute("loginUser") == null){ //为空表示没有登录，没有登录则返回登录
           页面
20             //没有登录则返回登录页面
21             return Action.LOGIN;
22         }
23         //不为空则表示已经登录成功，直接放行
24         return invocation.invoke();
25     }
26 }
```

### 1.2.7.4 配置拦截器

在struts.xml配置文件中配置，代码如下：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!DOCTYPE struts PUBLIC
4     "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
5     "http://struts.apache.org/dtds/struts-2.5.dtd">
6
7 <struts>
8
9
10     <!-- 开启动态方法调用 -->
11     <constant name="struts.enable.DynamicMethodInvocation" value="true"/>
12
13     <!-- 公共配置 -->
14     <package name="default" namespace="/" extends="struts-default">
```



```

15     <!-- 设置白名单 -->
16     <global-allowed-methods>regex:.*</global-allowed-methods>
17 </package>
18
19 <!-- 登录验证模块 -->
20 <package name="user" namespace="/" extends="default">
21
22     <!-- 定义拦截器 -->
23     <interceptors>
24         <!-- 注入登录拦截器类 -->
25         <interceptor name="loginInterceptor"
class="com.bdqn.interceptor.SystemInterceptor"/>
26
27         <!-- 1.定义拦截器栈 -->
28         <interceptor-stack name="myLoginStack">
29             <!-- 2.引用默认的拦截器 -->
30             <interceptor-ref name="defaultStack"/>
31             <!-- 3.引用自定义的登录拦截器 -->
32             <interceptor-ref name="loginInterceptor">
33                 <!-- 注意: includeMethods属性的优先级高于excludeMethods属性 -->
34                 <param name="includeMethods">add*,update*</param>
35                 <!-- excludeMethods: 设置不拦截的方法, 多个方法名之间使用逗号隔开 -->
36                 <param name="excludeMethods">login</param>
37                 <!-- includeMethods:设置拦截的方法, 多个方法名之间使用逗号隔开 -->
38             </interceptor-ref>
39         </interceptor-stack>
40     </interceptors>
41
42     <!-- 使用通配符访问 -->
43     <action name="user_*" class="com.bdqn.web.UserAction" method="{1}">
44         <!-- 登录成功 -->
45         <result name="success">/success.jsp</result>
46         <!-- 登录失败 -->
47         <result name="login">/login.jsp</result>
48
49         <!-- 4.引用拦截器 -->
50         <interceptor-ref name="myLoginStack"/>
51
52     </action>
53 </package>
54
55 </struts>

```

### 1.2.7.5 login.jsp

```

1 <%@taglib prefix="s" uri="/struts-tags" %>
2 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
3 <html>
4 <head>
5     <title>用户登录</title>
6 </head>
7 <body>
8
9 <s:fielderror/>
10

```

```
11 <form action="user_login.action" method="post">
12   <div>
13     <label>用户名: </label>
14     <input type="text" name="userName">
15     <font color="red"><s:fielderror fieldName="userName"/></font>
16   </div>
17   <div>
18     <label>密码: </label>
19     <input type="password" name="password">
20     <font color="red"><s:fielderror fieldName="password"/></font>
21   </div>
22   <div>
23     <input type="submit" value="登录">
24   </div>
25 </form>
26
27 </body>
28 </html>
```

### 1.2.7.6 测试

先不进行登录，访问[http://localhost:8080/user\\_delete.action](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](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"` )

```
1 <form action="upload.action" method="post" enctype="multipart/form-data">
2     <p>选择文件: <input type="file" name="upload"></p>
3     <p>
4         <input type="submit">
5     </p>
6 </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：

```
1 package com.bdqn.web;
2
3 import com.opensymphony.xwork2.ActionSupport;
4
5
6 import java.io.File;
7
8
9 public class UploadAction extends ActionSupport {
10
11     //文件
12     private File upload;//属性名必须与input标签的name属性值一致
13     //文件类型
14     private String uploadContentType;//input标签的name属性值 + ContentType (后缀)
15     //文件名称
16     private String uploadFileName;////input标签的name属性值 + FileName (后缀)
17
18     public File getUpload() {
19         return upload;
20     }
21 }
```

```

22     public void setUpload(File upload) {
23         this.upload = upload;
24     }
25
26     public String getUploadContentType() {
27         return uploadContentType;
28     }
29
30     public void setUploadContentType(String uploadContentType) {
31         this.uploadContentType = uploadContentType;
32     }
33
34     public String getUploadFileName() {
35         return uploadFileName;
36     }
37
38     public void setUploadFileName(String uploadFileName) {
39         this.uploadFileName = uploadFileName;
40     }
41 }

```

### 2.2.2.1 通过IO流上传

```

1  /**
2   * 方法一：使用IO流实现文件上传
3   * @return
4   */
5  public String uploadFileByIO(){
6      //1.获取文件上传的目标地址
7      String path = "E:/upload";
8      FileInputStream fis = null;
9      FileOutputStream fos = null;
10     try {
11         //2.通过输入流读取文件
12         fis = new FileInputStream(upload);
13         //3.获取文件的后缀名
14         String suffix = FilenameUtils.getExtension(uploadFileName);
15         //4.将文件重命名
16         String newFileName = UUID.randomUUID()+"."+suffix;
17         //5.通过输出流将文件保存到服务器
18         fos = new FileOutputStream(path+File.separator +newFileName);
19         //6.通过工具类保存文件
20         IOUtils.copy(fis,fos);
21     } catch (Exception e) {
22         e.printStackTrace();
23     } finally {
24         try {
25             //关闭资源：先用后关
26             if(fos!=null){
27                 fos.close();
28             }
29             if(fis!=null){
30                 fis.close();
31             }

```

```

32         } catch (IOException e) {
33             e.printStackTrace();
34         }
35     }
36     return SUCCESS;
37 }

```

### 2.2.2.2 通过FileUtils工具类上传

```

1  /**
2   * 方法2:通过FileUtils工具类上传
3   * @return
4   */
5  public String uploadSimple(){
6      try {
7          //1.获取文件上传的目标地址
8          String path = "E:/upload";
9          //2.获取文件的后缀名
10         String suffix = FilenameUtils.getExtension(uploadFileName);
11         //3.将文件重命名
12         String newFileName = UUID.randomUUID()+"."+suffix;
13         //4.构建文件路径及文件的相关信息
14         String realPath = path+File.separator +newFileName;
15         //5.借助工具类保存文件
16         FileUtils.copyFile(upload,new File(realPath));
17     } catch (IOException e) {
18         e.printStackTrace();
19     }
20     return SUCCESS;
21 }

```

### 2.2.3 struts.xml

```

1  <!-- 文件上传模块 -->
2  <package name="upload" extends="default">
3      <action name="upload" class="com.bdqf.web.UploadAction" method="uploadSimple">
4          <result>/success.jsp</result>
5      </action>
6  </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

```

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

```

## 2.3 多文件上传

### 2.3.1 mupload.jsp

```
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2 <html>
3 <head>
4     <title>单文件上传</title>
5 </head>
6 <body>
7
8 <form action="mupload.action" method="post" enctype="multipart/form-data">
9     <p>选择文件: <input type="file" name="upload"></p>
10    <p>选择文件: <input type="file" name="upload"></p>
11    <p>
12        <input type="submit">
13    </p>
14 </form>
15
16 </body>
17 </html>
```

### 2.3.2 Action控制器

```
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     public File[] getUpload() {
25         return upload;
26     }
27
28     public void setUpload(File[] upload) {
29         this.upload = upload;
30     }
```

```

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

```

```
37         fis = new FileInputStream(upload[i]);
38         //3.获取文件的后缀名
39         String suffix = FilenameUtils.getExtension(uploadFileName[i]);
40         //4.将文件重命名
41         String newFileName = UUID.randomUUID()+"."+suffix;
42         //5.通过输出流将文件保存到服务器
43         fos = new FileOutputStream(path+File.separator +newFileName);
44         //6.通过工具类保存文件
45         IOUtils.copy(fis,fos);
46     } catch (Exception e) {
47         e.printStackTrace();
48     } finally {
49         try {
50             //关闭资源：先用后关
51             if(fos!=null){
52                 fos.close();
53             }
54             if(fis!=null){
55                 fis.close();
56             }
57         } catch (IOException e) {
58             e.printStackTrace();
59         }
60     }
61 }
62
63
64     return SUCCESS;
65 }
66
67 public File[] getUpload() {
68     return upload;
69 }
70
71 public void setUpload(File[] upload) {
72     this.upload = upload;
73 }
74
75 public String[] getUploadContentType() {
76     return uploadContentType;
77 }
78
79 public void setUploadContentType(String[] uploadContentType) {
80     this.uploadContentType = uploadContentType;
81 }
82
83 public String[] getUploadFileName() {
84     return uploadFileName;
85 }
86
87 public void setUploadFileName(String[] uploadFileName) {
88     this.uploadFileName = uploadFileName;
89 }
90 }
```



### 2.3.2.2 方法2

```
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      * 方法2
25      * @return
26      */
27     public String uploadSimple(){
28
29         for (int i = 0; i < upload.length; i++) {
30             try {
31                 //1.获取文件上传的目标地址
32                 String path = "E:/upload";
33                 //2.获取文件的后缀名
34                 String suffix = FilenameUtils.getExtension(uploadFileName[i]);
35                 //3.将文件重命名
36                 String newFileName = UUID.randomUUID()+"."+suffix;
37                 //4.构建文件路径及文件的相关信息
38                 String realPath = path+File.separator +newFileName;
39                 //5.借助工具类保存文件
40                 FileUtils.copyFile(upload[i],new File(realPath));
41             } catch (IOException e) {
42                 e.printStackTrace();
43             }
44         }
45
46
47         return SUCCESS;
48     }
49
50     public File[] getUpload() {
51         return upload;
52     }
53
54     public void setUpload(File[] upload) {
55         this.upload = upload;
56     }
57 }
```

```

56     }
57
58     public String[] getUploadContentType() {
59         return uploadContentType;
60     }
61
62     public void setUploadContentType(String[] uploadContentType) {
63         this.uploadContentType = uploadContentType;
64     }
65
66     public String[] getUploadFileName() {
67         return uploadFileName;
68     }
69
70     public void setUploadFileName(String[] uploadFileName) {
71         this.uploadFileName = uploadFileName;
72     }
73 }

```

### 2.3.3 struts.xml

```

1  <!-- 文件上传模块 -->
2  <package name="upload" extends="default">
3      <!-- 多文件上传 -->
4      <action name="mupload" class="com.bdqn.web.MoreUploadAction" method="uploadSimple">
5          <result>/success.jsp</result>
6      </action>
7  </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

```
1 <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2 <html>
3 <head>
4     <title>文件下载</title>
5 </head>
6 <body>
7
8 <a href="download.action?fileName=hello.jpg">文件下载(英文名称文件)</a>
9 <hr>
10 <a href="download.action?fileName=你好.jpg">文件下载(中文名称文件)</a>
11
12 </body>
13 </html>
```

### 3.2.2 控制器

```
1 package com.bdqn.web;
2
3 import com.opensymphony.xwork2.ActionSupport;
4
5 import java.io.BufferedInputStream;
6 import java.io.FileInputStream;
7 import java.io.FileNotFoundException;
8 import java.io.InputStream;
9 import java.net.URLEncoder;
10
11 public class DownloadAction extends ActionSupport {
12
13     //输入流：读取要下载的文件信息
14     private InputStream inputStream;
15     //文件名称
16     private String fileName;//与a标签的参数名称保持一致
17
18     /**
19      * 文件下载
20      * @return
21      */
22     public String download(){
23         try {
24             //1.指定文件下载的地址
25             String path = "E:/upload/"+fileName;
26             //指定编码格式
27             fileName = URLEncoder.encode(fileName,"UTF-8");
28             //2.通过输入流读取并下载文件
29             inputStream = new BufferedInputStream(new FileInputStream(path));
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33         return SUCCESS;
34     }
35 }
```

```

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>

```