## 通配符

在struts中配置信息中，可以\*与{1}可以优化配置

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">  <struts>  <package name="config" namespace="/user" extends="struts-default" abstract="false">    <!--  <action name="login" class="cn.itcast.a\_config.UserAction" method="login">  <result name="success">/index.jsp</result>  </action>  <action name="register" class="cn.itcast.a\_config.UserAction" method="register">  <result name="success">/index.jsp</result>  </action>  -->    <!-- 使用通配符优化上面的步骤 -->  <!-- http://localhost:8080/struts02/user\_login -->  <action name="user\_\*" class="cn.itcast.a\_config.UserAction" method="{1}">  <result name="{1}">/{1}.jsp</result>  </action>    </package>    </struts> |

## Struts中的路径匹配

|  |
| --- |
| <package name="config" namespace="/user" extends="struts-default" abstract="false">    <action name="user\_\*" class="cn.itcast.a\_config.UserAction" method="{1}">  <result name="{1}">/{1}.jsp</result>  </action>    </package> |
| <http://localhost:8080/struts/user/user_login> ok |
| <http://localhost:8080/struts/user/a/b/user_login> ok |
| <http://localhost:8080/struts/a/b/user/user_login> 不行 |

Tomcat

Localhost 找到哪一台计算机

8080 找到tomcat

Struts02 找到项目名称

/user/a/b 先看有没有这个名称空间，没找到往下找，找到就返回

/user/a 先看有没有这个名称空间，没找到往下找，找到就返回

/user 先看有没有这个名称空间，没找到往下找，找到就返回

Struts常量

## Struts中的默认访问路径

Struts1中默认访问路径是\*.do

Struts2中默认访问路径是\*.action

如何修改默认访问路径

1. Struts2的.action访问后缀在哪里定义？

Struts-core-2.3.4-1.jar/org.apache.struts/default.properties

Struts.action.extension=action,,

1. 在struts.xml中修改

<constant name="struts.action.extension" value="action,do,"></constant>

这样action do 或者什么都不写都可以

Value=”action,do” 只能是do或者是action

|  |
| --- |
| 指定默认编码集,作用于HttpServletRequest的setCharacterEncoding方法 和freemarker 、velocity的输出  <constant name="struts.i18n.encoding" value="UTF-8"/>  自定义后缀修改常量  <constant name="struts.action.extension" value="do"/>  设置浏览器是否缓存静态内容,默认值为true(生产环境下使用),开发阶段最好关闭  <constant name="struts.serve.static.browserCache" value="false"/>  当struts的配置文件修改后,系统是否自动重新加载该文件,默认值为false(生产环境下使用),开发阶段最好打开  <constant name="struts.configuration.xml.reload" value="true"/>  开发模式下使用,这样可以打印出更详细的错误信息  <constant name="struts.devMode" value="true" />  默认的视图主题  <constant name="struts.ui.theme" value="simple" />  与spring集成时，指定由spring负责action对象的创建  <constant name="struts.objectFactory" value="spring" />  该属性设置Struts 2是否支持动态方法调用，该属性的默认值是true。如果需要关闭动态方法调用，则可设置该属性  为 false  <constant name="struts.enable.DynamicMethodInvocation" value="false"/>  上传文件的大小限制  <constant name="struts.multipart.maxSize" value=“10701096"/> |

## 全局视图跳转配置

|  |
| --- |
| <!-- 配置全局跳转视图，必须放在action前面 -->  <global-results>  <result name="success">/index.jsp</result>  </global-results> |

## 配置的默认值

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">  <struts>  <package name="config2" namespace="/" extends="struts-default">    <!-- 动态方法调用: http://locahost:8080/struts02/user!login  <action name="user" class="cn.itcast.b\_config2.UserAction">  <result name="success">/index.jsp</result>  </action>  -->    <!-- 配置全局跳转视图 -->  <global-results>  <result name="success">/index.jsp</result>  </global-results>        <!-- 通配符: http://locahost:8080/struts02/user\_login -->  <action name="user\_\*" class="cn.itcast.b\_config2.UserAction" method="{1}">  </action>    <!--  <action name="test" class="cn.itcast.b\_config2.TestAction" method="execute">  返回结果标记success对应的页面再当前action中没有配置，  所以会去找全局配置有是否有success标记对应的页面  </action>  -->    <!-- 配置各项默认值 -->  <!--  name 只配置了访问路径名称  class 默认执行的action在struts-default有配置  <default-class-ref class="com.opensymphony.xwork2.ActionSupport" />  method 默认为execute  默认的方法execute返回值为success,对应的页面去全局视图找。    -->  <action name="test"></action>    <!-- 什么情况不配置class？ 即处理的aciton -->  <!-- 答案： 当只是需要跳转到WEB-INF下资源的时候。 -->  <action name="test2">  <result name="success">/WEB-INF/index.jsp</result>  </action>    </package>    </struts> |

## Struts中的数据处理

对数据操作的所有方法：（把数据保存到域中）

### 方式1：

直接获取servletapi

核心类：ServletActionContext提供的静态方法

### 方式2：

通过ActionContext获取不同（代表requset/session/application）的map

|  |
| --- |
| package cn.itcast.c\_data;  import java.util.Map;  import javax.servlet.ServletContext;  import com.opensymphony.xwork2.ActionContext;  import com.opensymphony.xwork2.ActionSupport;  /\*\*  \* 数据处理  \* @author Jie.Yuan  \*  \*/  public class DataAction\_bak extends ActionSupport{    @Override  public String execute() throws Exception {    // 1. 请求数据封装； 2. 调用Service处理业务逻辑，拿到结果数据    // 3. 数据保存到域中    /\*  \*  // Struts中对数据操作，方式1： 直接拿到ServletApi, 执行操作  HttpServletRequest request = ServletActionContext.getRequest();  HttpSession session = request.getSession();  ServletContext application = ServletActionContext.getServletContext();  // 操作  request.setAttribute("request\_data", "request\_data1");  session.setAttribute("session\_data", "session\_data1");  application.setAttribute("application\_data", "application\_data1");  \*/      // 【推荐：解耦的方式实现对数据的操作】  // Struts中对数据操作，方式2： 通过ActionContext类  ActionContext ac = ActionContext.getContext();  // 得到Struts对HttpServletRequest对象进行了封装，封装为一个map  // 拿到表示request对象的map  Map<String,Object> request = ac.getContextMap();  // 拿到表示session对象的map  Map<String, Object> session = ac.getSession();  // 拿到表示servletContext对象的map  Map<String, Object> application = ac.getApplication();    // 数据  request.put("request\_data", "request\_data1\_actionContext");  session.put("session\_data", "session\_data1\_actionContext");  application.put("application\_data", "application\_data1\_actionContext");    return SUCCESS;  }  } |

### 方式3：

实现接口的方式：（RequestAware/SessionAware/ApplicationAware）

|  |
| --- |
| package cn.itcast.c\_data;  import java.util.Map;  import org.apache.struts2.interceptor.ApplicationAware;  import org.apache.struts2.interceptor.RequestAware;  import org.apache.struts2.interceptor.SessionAware;  import com.opensymphony.xwork2.ActionSupport;  /\*\*  \* 数据处理, 方式3： 实现接口的方法  \* @author Jie.Yuan  \*  \*/  public class DataAction extends ActionSupport implements RequestAware, SessionAware, ApplicationAware{    private Map<String, Object> request;  private Map<String, Object> session;  private Map<String, Object> application;    // struts运行时候，会把代表request的map对象注入  @Override  public void setRequest(Map<String, Object> request) {  this.request = request;  }    // 注入session  @Override  public void setSession(Map<String, Object> session) {  this.session = session;  }    // 注入application  @Override  public void setApplication(Map<String, Object> application) {  this.application = application;  }  @Override  public String execute() throws Exception {    // 数据  request.put("request\_data", "request\_data1\_actionAware");  session.put("session\_data", "session\_data1\_actionAware");  application.put("application\_data", "application\_data1\_actionAware");  //  return SUCCESS;  }  } |

## 请求数据的自动封装

Jsp表单数据填充到action中的对象中：

|  |
| --- |
| Register.jsp |
| <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">    <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>    <body>  <form action="${pageContext.request.contextPath }/user\_register.action" method="post">  用户名:<input type="text" name="user.name"><br/>  密码:<input type="text" name="user.pwd"><br/>  年龄:<input type="text" name="user.age"><br/>  生日:<input type="text" name="user.birth"><br/>    <input type="submit" value="注册">  </form>  </body>  </html> |

|  |
| --- |
| package cn.itcast.d\_type;  import java.util.Date;  /\*\*  \* Struts核心业务： 请求数据自动封装以及类型转换  \* @author Jie.Yuan  \*  \*/  public class UserAction {    // 对象类型，一定给get方法  private User user;  public void setUser(User user) {  this.user = user;  }  public User getUser() {  return user;  }  // 处理注册请求  public String register() {  System.out.println(user.getName());  System.out.println(user.getPwd());  System.out.println(user.getAge());  System.out.println(user.getBirth());  return "success";  }  } |

## 类型的自动转换

Struts中jsp提交的数据，struts会自动转换为action中的类型；

对于基本数据类型以及日期类型会自动转换。

日期类型只支持yyyy-MM-dd格式

如何格式，需要自定义类型转换器：

局部类型转换器

全局类型转换器

### 局部类型转换器

转换器开发步骤

1. 写转换器类
2. 配置转换器类（告诉struts应用自己的转换器类）

---》在同包的action目录下，新建一个properties文件

---》命名规则：ActionClassName-conversion.properties

举例：cn.itcast.d\_type/UserAction-conversion.properties

1. 内容

User.birth=转换器类全路径（cn.itcast.d\_type.MyConverter）

局部类型转换器不能给其他转换器用。

### 全局类型转换器

* **实现**

1. 自定义转换器继承StrutsTypeConverter

|  |
| --- |
| package cn.itcast.d\_type;  import java.text.DateFormat;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.Map;  import org.apache.struts2.util.StrutsTypeConverter;  /\*\*  \* 自定义类型转换器类  \*  \* @author Jie.Yuan  \*  \*/  public class MyConverter extends StrutsTypeConverter {  // 新需求： 要求项目中要支持的格式,如: yyyy-MM-dd/yyyyMMdd/yyyy年MM月dd日..  // 先定义项目中支持的转换的格式  DateFormat[] df = { new SimpleDateFormat("yyyy-MM-dd"),  new SimpleDateFormat("yyyyMMdd"),  new SimpleDateFormat("yyyy年MM月dd日") };  /\*\*  \* 把String转换为指定的类型 【String To Date】  \*  \* @param context  \* 当前上下文环境  \* @param values  \* jsp表单提交的字符串的值  \* @param toClass  \* 要转换为的目标类型  \*/  @Override  public Object convertFromString(Map context, String[] values, Class toClass) {  // 判断: 内容不能为空  if (values == null || values.length == 0) {  return null;  }  // 判断类型必须为Date  if (Date.class != toClass) {  return null;  }    // 迭代：转换失败继续下一个格式的转换； 转换成功就直接返回  for (int i=0; i<df.length; i++) {  try {  return df[i].parse(values[0]);  } catch (ParseException e) {  continue;  }  }  return null;  }  @Override  public String convertToString(Map context, Object o) {  return null;  }  } |

2、重写convertFromString和convertToString方法

3、注册转换器

3.1 在项目src目录下建立以下固定文件

**xwork-conversion.properties**

3.2 在3.1文件中添加以下数据

需要转换的类类型=转换器类的权限定名

如: java.util.Date= cn.itcast.converter.DateConverter

* **总结**

该拦截器负责对错误信息进行拦截器<interceptor name="conversionError“

class="org.apache.struts2.interceptor.StrutsConversionErrorInterceptor"/>

## 文件上传

|  |
| --- |
| <form action="${pageContext.request.contextPath }/fileUploadAction" method="post" enctype="multipart/form-data">  用户名:<input type="text" name="userName"><br/>  文件:<input type="file" name="file1"><br/>    <input type="submit" value="上传">  </form> |

|  |
| --- |
| package cn.itcast.e\_fileupload;  import java.io.File;  import org.apache.commons.io.FileUtils;  import org.apache.struts2.ServletActionContext;  import com.opensymphony.xwork2.ActionSupport;  public class FileUpload extends ActionSupport {  // 对应表单：<input type="file" name="file1">  private File file1;  // 文件名  private String file1FileName;  // 文件的类型(MIME)  private String file1ContentType;  public void setFile1(File file1) {  this.file1 = file1;  }  public void setFile1FileName(String file1FileName) {  this.file1FileName = file1FileName;  }  public void setFile1ContentType(String file1ContentType) {  this.file1ContentType = file1ContentType;  }      @Override  public String execute() throws Exception {  /\*\*\*\*\*\*拿到上传的文件，进行处理\*\*\*\*\*\*/  // 把文件上传到upload目录    // 获取上传的目录路径  String path = ServletActionContext.getServletContext().getRealPath("/upload");  // 创建目标文件对象  File destFile = new File(path,file1FileName);  // 把上传的文件，拷贝到目标文件中  FileUtils.copyFile(file1, destFile);    return SUCCESS;  }  } |

## 文件上传细节处理

文件大小限制

Struts默认支持的文件上传最大是2m.

通过修改常量：

|  |
| --- |
| <!-- 4. 修改上传文件的最大大小为30M -->  <constant name="struts.multipart.maxSize" value="31457280"/> |

限制上传文件的运行的类型

需求：只运行txt/jpg后缀的文件。

拦截器：注入参数从而限制文件上传的类型。

错误提示

当文件上传出现错误时候，struts内部会返回input视图（错误视图）。所以就需要我们在struts.xml中配置input视图对应的错误页面！

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">  <struts>  <package name="upload\_" extends="struts-default">  <!-- 注意： action 的名称不能用关键字"fileUpload" -->  <action name="fileUploadAction" class="cn.itcast.e\_fileupload.FileUpload">    <!-- 限制运行上传的文件的类型 -->  <interceptor-ref name="defaultStack">    <!-- 限制运行的文件的扩展名 -->  <param name="fileUpload.allowedExtensions">txt,jpg,jar</param>    <!-- 限制运行的类型 【与上面同时使用，取交集】  <param name="fileUpload.allowedTypes">text/plain</param>  -->    </interceptor-ref>    <result name="success">/e/success.jsp</result>    <!-- 配置错误视图 -->  <result name="input">/e/error.jsp</result>  </action>    <action name="down\_\*" class="cn.itcast.e\_fileupload.DownAction" method="{1}">  <!-- 列表展示 -->  <result name="list">/e/list.jsp</result>  <!-- 下载操作 -->  <result name="download" type="stream">    <!-- 运行下载的文件的类型:指定为所有的二进制文件类型 -->  <param name="contentType">application/octet-stream</param>    <!-- 对应的是Action中属性： 返回流的属性【其实就是getAttrInputStream()】 -->  <param name="inputName">attrInputStream</param>    <!-- 下载头，包括：浏览器显示的文件名 -->  <param name="contentDisposition">attachment;filename=${downFileName}</param>    <!-- 缓冲区大小设置 -->  <param name="bufferSize">1024</param>  </result>  </action>  </package>  </struts> |

## Struts的文件下载

文件下载，2种方式：

方式1：通过response（通过ServletActionContext获取）对象向浏览器写入字节流数据；设置下载的响应头。

方式2：struts的方式

Struts的文件下载

|  |
| --- |
| <result-type name="stream" class="org.apache.struts2.dispatcher.StreamResult"/> |

显示文件列表jsp

|  |
| --- |
| <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">    <title>下载列表</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  </head>    <body>  <table border="1" align="center">  <tr>  <td>编号</td>  <td>文件名</td>  <td>操作</td>  </tr>  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <c:forEach var="fileName" items="${fileNames}" varStatus="vs">  <tr>  <td>${vs.count }</td>  <td>${fileName }</td>  <td>  <!-- 构建一个url -->  <c:url var="url" value="down\_down">  <c:param name="fileName" value="${fileName}"></c:param>  </c:url>    <a href="${url }">下载</a>  </td>  </tr>  </c:forEach>  </table>  </body>  </html> |

下载的java源码

|  |
| --- |
| package cn.itcast.e\_fileupload;  import java.io.File;  import java.io.InputStream;  import java.io.UnsupportedEncodingException;  import java.net.URLEncoder;  import java.util.Map;  import org.apache.struts2.ServletActionContext;  import com.opensymphony.xwork2.ActionContext;  import com.opensymphony.xwork2.ActionSupport;  /\*\*  \* 文件下载  \* 1. 显示所有要下载文件的列表  \* 2. 文件下载  \* @author Jie.Yuan  \*  \*/  public class DownAction extends ActionSupport {      /\*\*\*\*\*\*\*\*\*\*\*\*\*1. 显示所有要下载文件的列表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  public String list() throws Exception {    //得到upload目录路径  String path = ServletActionContext.getServletContext().getRealPath("/upload");  // 目录对象  File file = new File(path);  // 得到所有要下载的文件的文件名  String[] fileNames = file.list();  // 保存  ActionContext ac = ActionContext.getContext();  // 得到代表request的map (第二种方式)  Map<String,Object> request= (Map<String, Object>) ac.get("request");  request.put("fileNames", fileNames);  return "list";  }      /\*\*\*\*\*\*\*\*\*\*\*\*\*2. 文件下载\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    // 1. 获取要下载的文件的文件名  private String fileName;  public void setFileName(String fileName) {  // 处理传入的参数中问题(get提交)  try {  fileName = new String(fileName.getBytes("ISO8859-1"),"UTF-8");  } catch (UnsupportedEncodingException e) {  throw new RuntimeException(e);  }  // 把处理好的文件名，赋值  this.fileName = fileName;  }    //2. 下载提交的业务方法 (在struts.xml中配置返回stream)  public String down() throws Exception {  return "download";  }    // 3. 返回文件流的方法  public InputStream getAttrInputStream(){  return ServletActionContext.getServletContext().getResourceAsStream("/upload/" + fileName);  }    // 4. 下载显示的文件名（浏览器显示的文件名）  public String getDownFileName() {  // 需要进行中文编码  try {  fileName = URLEncoder.encode(fileName, "UTF-8");  } catch (UnsupportedEncodingException e) {  throw new RuntimeException(e);  }  return fileName;  }    } |

配置文件源码

|  |
| --- |
| <action name="down\_\*" class="cn.itcast.e\_fileupload.DownAction" method="{1}">  <!-- 列表展示 -->  <result name="list">/e/list.jsp</result>  <!-- 下载操作 -->  <result name="download" type="stream">    <!-- 运行下载的文件的类型:指定为所有的二进制文件类型 -->  <param name="contentType">application/octet-stream</param>    <!-- 对应的是Action中属性： 返回流的属性【其实就是getAttrInputStream()】 -->  <param name="inputName">attrInputStream</param>    <!-- 下载头，包括：浏览器显示的文件名 -->  <param name="contentDisposition">attachment;filename=${downFileName}</param>    <!-- 缓冲区大小设置 -->  <param name="bufferSize">1024</param>  </result>  </action> |