## struts2中提供的类型转换

接口：com.opensymphony.xwork2.conversion.TypeConverter

实现类：com.opensymphony.xwork2.conversion.impl.DefaultTypeConverter

在web中我们使用beanutils直接将表单数据封装到javaBean中。---类型转换

struts2中action得到请求参数，也可以直接封装到javaBean.

struts2 内部提供大量类型转换器，用来完成数据类型转换问题

boolean 和 Boolean

char和 Character

int 和 Integer

long 和 Long

float 和 Float

double 和 Double

Date 可以接收 yyyy-MM-dd格式字符串

数组 可以将多个同名参数，转换到数组中

例如:日期类型，我们传递 yyyy-MM-dd yyyy年MM月dd日格式都可以，但是如果是yyyy/MM/dd

就会出现问题.

关于struts2中的类型转换器:

struts2中的类型转换器根接口是：com.opensymphony.xwork2.conversion.TypeConverter。

自定义类型转换器:

步骤:

1.创建一个类实现TypeConverter接口.

2.重写接口中方法，实现类型转换操作.

3.注册类型转换器.

详解说明:

1.创建一个自定义类型转换器

1.实现TypeConverter需要重写

public Object convertValue(Map<String, Object> context, Object target, Member member, String propertyName, Object value, Class toType);

如果实现接口，这个方法参数太多(6个)

2.不推荐实现接口，可以继承 DefaultTypeConverter类

优点:重写的方法参数没有那么多

public Object convertValue(Map<String, Object> context, Object value, Class toType) {

return convertValue(value, toType);

}

2.怎样注册一个自定义类型转换器.

1.局部--针对于action

配置文件所在位置以及名称: 在Action类所在包 创建 Action类名-conversion.properties ,

配置文件书写: 格式 ： 属性名称=类型转换器的全类名

birthday=com.qf.converter.DateTypeConverter

2.局部--针对于model

配置文件所在位置以及名称: 在model类所在包 创建 model类名-conversion.properties ,

配置文件书写: 格式 ： 属性名称=类型转换器的全类名

3.全局

配置文件所在位置以及名称:在src下创建一个xwork-conversion.properties

配置文件书写: 格式: 要转换的类型全名=类型转换器的全类名

java.util.Date=com.qf.converter.DateTypeConverter

## struts2的标签

### # 符号的用法

\* 获得contextMap中的数据

<s:property value="#request.name"/>

<s:property value="#session.name"/>

<s:property value="#application.name"/>

<s:property value="#attr.name"/>

<s:property value="#parameters.id"/>

<s:property value="#parameters.name"/>

\* 构建一个map集合

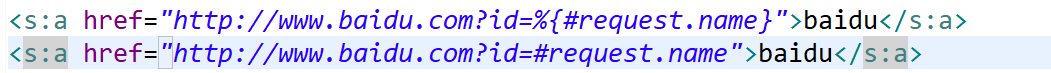
\* 例如：

<s:radio name="sex" list="{'男','女'}"></s:radio>

<s:radio name="sex" list="#{'0':'男','1':'女'}"></s:radio>

### % 符号的用法

\* 强制字符串解析成OGNL表达式。



### $ 符号的用法

\* 在配置文件中可以使用OGNL表达式，例如：文件下载的配置文件。

<action name="download1" class="cn.qf.demo2.DownloadAction">

<result name="success" type="stream">

<param name="contentType">${contentType}</param>

<param name="contentDisposition">attachment;filename=${downFilename}</param>

</result>

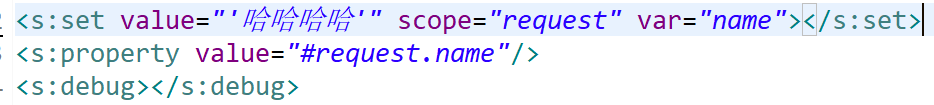
</action>

<%@ taglib uri="/struts-tags" prefix="s"%>

### 逻辑标签：

迭代： <s:iterator value="ognl表达式" var="" >

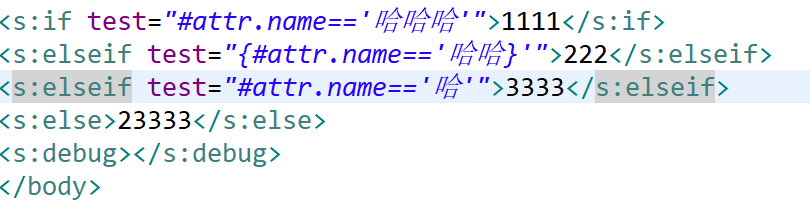
赋值： <s:set var="" value="ognl表达式"/>



获取： <s:property value="ognl表达式"/>

判断： <s:if/> <s:elseif/> <s:else/>





### UI标签：

<s:form/> <form>

<s:textfield/> <input type="text"/>

<s:password/> <input type="password"/>

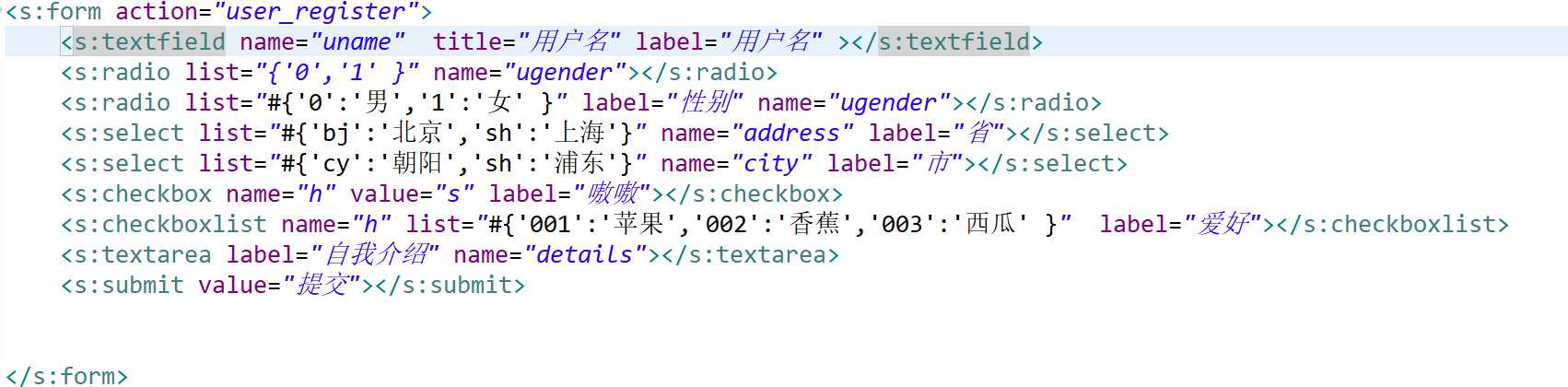
<s:checkbox/> <input type="checkbox"/>

<s:checkboxlist list="多个值"/>

多个<input type="checkbox"/>

<s:radio list="多个值"/> 多个<input type="radio"/>

<s:select list="多个值"/> <select>多个<option/></select/>



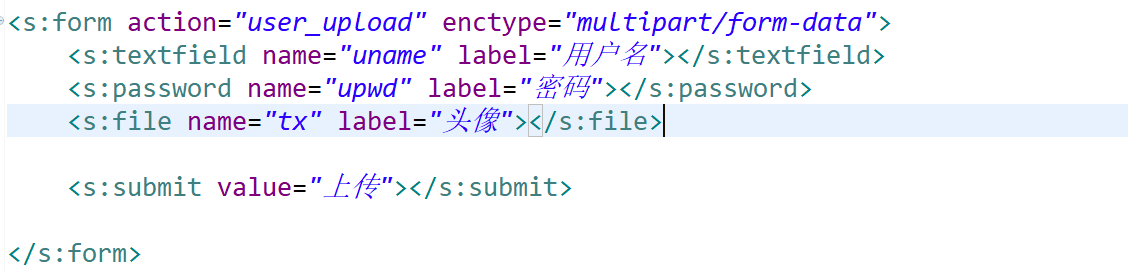
## 文件上传

表单method必须是post；

b.enctype取值必须是multipart/form-data；

c.提供文件选择域。

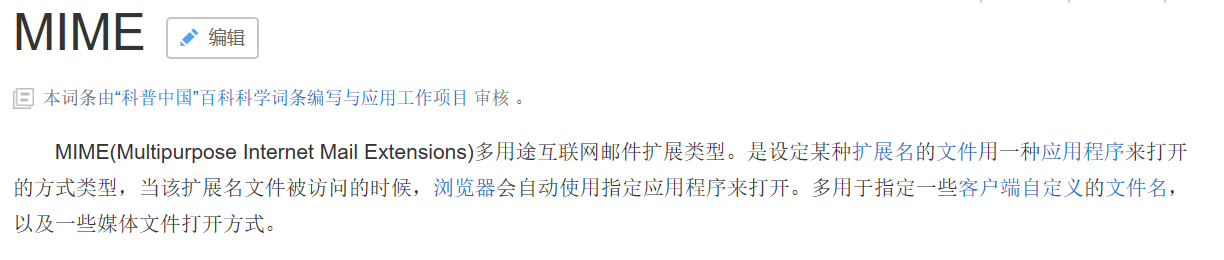
### JSP页面



### 上传Action

|  |
| --- |
| /\*\*  \* 接收上传内容  \*/  //1)接收上传文件 名字来自于表达的file的name属性名称  private File attach;  //2)接收文件类型  private String attachContentType;  //3)接收文件名称  private String attachFileName;  /\*\*  \* 一定要给setter方法.这些setter方法就是上传文件内容进行赋值  \*/  public String upload() throws Exception {  System.out.println(attach);  System.out.println(attachContentType);  System.out.println(attachFileName);  System.out.println(info);    /\*\*  \* 把文件保存到服务器硬盘  \*/  FileUtils.copyFile(attach, new File(savePath+attachFileName));    return SUCCESS;  } |

### 文件上传的类型MIME



### 文件的上传限制

|  |
| --- |
| **第一种方式：通过修改struts2的常量**  <!-- 限制上传文件的大小，。  在default.properties中配的struts.multipart.maxSize=2097152-->  <constant name="struts.multipart.maxSize" value="5242880"></constant>  <action name="upload" class="com.qf.UploadAction" method="upload">  <interceptor-ref name="defaultStack">  <!-- 限制上传文件的扩展名。如果有多个，使用逗号分隔  <param name="fileUpload.allowedExtensions">jpg,png,bmp</param>-->  <!-- 限制上传文件的MIME类型。 如果有多个，使用逗号分隔 -->  <param name="fileUpload.allowedTypes">image/jpg,image/jpeg,image/png</param>  </interceptor-ref>  <result name="input">/upload.jsp</result>  </action> |

## 多文件上传

### Jsp页面

|  |
| --- |
| <%--文件上传的必要前提：  1.请求方式必须是post  2.enctype属性的取值必须是multipart/form-data  3.提供一个文件选择域  --%>  <s:form action="upload2.action" enctype="multipart/form-data">  <s:textfield name="username" label="用户名" />  <s:file name="photo" label="照片" />  <s:file name="photo" label="照片" />  <s:submit value="上传"/>  </s:form> |

### Action代码

|  |
| --- |
| //表单上提供的字段  private String username;  private File[] photo;  //struts2在文件上传时提供的属性  private String[] photoFileName;//上传的文件名。上传字段名称+FileName 注意大小写  private String[] photoContentType;//上传文件的MIME类型。上传字段名称+ContentType 注意大小写      public String upload(){  //1.拿到ServletContext  ServletContext application = ServletActionContext.getServletContext();  //2.调用realPath方法，获取根据一个虚拟目录得到的真实目录  String filePath = application.getRealPath("/WEB-INF/files");  //3.如果这个真实的目录不存在，需要创建  File file = new File(filePath);  if(!file.exists()){  file.mkdirs();  }  //4.把photo存过去  //剪切：把临时文件剪切指定的位置，并且给他重命名。 注意：临时文件没有了  for(int i=0;i<photo.length;i++){  photo[i].renameTo(new File(file,photoFileName[i]));  }    return null;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public File[] getPhoto() {  return photo;  }  public void setPhoto(File[] photo) {  this.photo = photo;  }  public String[] getPhotoFileName() {  return photoFileName;  }  public void setPhotoFileName(String[] photoFileName) {  this.photoFileName = photoFileName;  }  public String[] getPhotoContentType() {  return photoContentType;  }  public void setPhotoContentType(String[] photoContentType) {  this.photoContentType = photoContentType;  } |

## 文件下载：

|  |
| --- |
| **<result name=*"success"* type=*"stream"*>**  **<param name=*"contentType"*>application/octet-stream</param>**  **<param name=*"contentDisposition"*>attachment;filename=${fileName}</param>**  **<param name=*"inputName"*>inputStream</param>**  **</result>** |

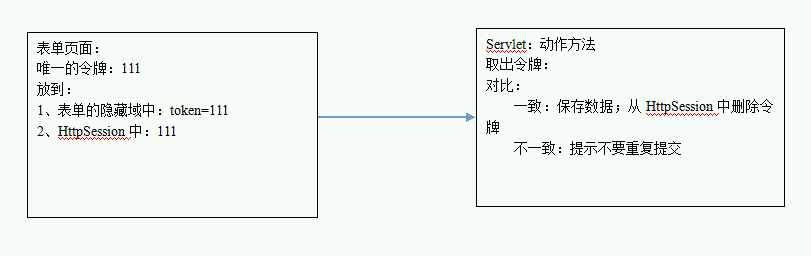
### 下载action

|  |
| --- |
| public class DownloadAction extends ActionSupport {    private InputStream inputStream;  //创建一个文件名的属性  private String filename;    public String download()throws Exception{  //1.找到文件的存储路径  String filePath = ServletActionContext.getServletContext().getRealPath("/WEB-INF/files/upload.jpg");  //2.把文件读到一个InputStream中  inputStream = new FileInputStream(filePath);    //3.返回一个成功  return SUCCESS;  //4.由一个叫做stream的结果类型为我们把剩下的事情做完。  }  public InputStream getInputStream() {  return inputStream;  }  public void setInputStream(InputStream inputStream) {  this.inputStream = inputStream;  }  public String getFilename() {  return filename;  }  public void setFilename(String filename) {  this.filename = filename;  }  } |

### 下载配置

|  |
| --- |
| <action name="download" class="com.qf.DownloadAction" method="download">  <result name="success" type="stream">  <!-- 给stream的结果类型注入参数：Content-Type -->  <param name="contentType">**application/octet-stream**</param>  <!-- 告知客户浏览器以下载的方式打开  <param name="contentDisposition">attachment;filename=photo.jpg</param> -->    <!-- 注入字节输入流 :取值要写动作类中set方法的名称，首字母改成小写-->  <param name="inputName">inputStream</param>  </result>  </action> |

# Struts2防止表单重复提交



### Struts2中的解决办法：

#### 使用重定向

Struts2给我们提供了2个拦截器，oken与tokenSession，都可以防止表单重复提交

**遗留的问题：防不住后退，再提交。**

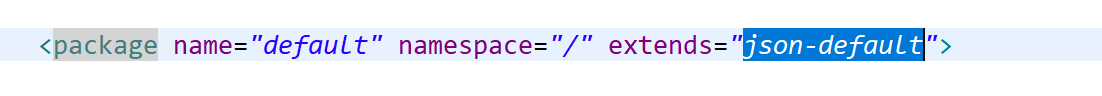
token与tokenSession的区别

1）相同点：都是为解决struts2中防止表单重复提交而生；

2）不同点：使用token拦截器，如果发生重复提交时，页面会自动跳转到invalid.token的这个result页面；使用tokenSession拦截器时，依然会响应目标页面，但不会执行tokenSession之后的拦截器，就像什么都没有发生一样。

# Struts2整合Json

导入插件包：struts2-json-plugin-2.3.24.jar



# Echarts生成图表插件