# 1. Struts操作域对象

能够得到域对象进行操作很重要，因为需要操作域对象中的数据，然后页面可以显示数据。在Struts的Action类中，一般使用以下三种方式来获得域对象并操作。

## 1.1 使用ServletActionContext类

此种方式是直接使用Servlet的API进行域对象的操作。因为ServletActionContext类提供了很多静态方法，可直接拿到与Servlet相关的类。常用的有：

（1）ServletActionContext.getRequest()：获得request对象；

（2）ServletActionContext.getResponse()：获得reponse对象；

（3）ServletActionContext.getServletContext()：获得ServletContext对象，可操作application域对象；

（4）ServletActionContext.getPageContext()：获得pageContext对象。

虽然没有提供session对象，但是我们可以通过request对象得到。

例如：

|  |
| --- |
| **public** String add() {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpServletResponse response = ServletActionContext.*getResponse*();  ServletContext context = ServletActionContext.*getServletContext*();  PageContext pageContext = ServletActionContext.*getPageContext*();  *// 操作域对象* request.setAttribute(**"name"**, **"张三"**);  *// 获得session* HttpSession session = request.getSession();  **return "success"**; } |

这种使用ServletAPI的方式，使软件的耦合性变高了。但是可用Servlet中通用的方法。

## 1.2 使用ActionContext类

Struts将很多Servlet中的对象进行了封装，这样就避免了和Servlet的耦合性。比如ActionContext类中，就将request、session和context这些域对象封装成了Map集合对象，即调用集合的put方法向这些“域对象”中存数据。

（1）通过ActionContext.getContext()方法获得一个ActionContext实例。

（2）调用实例的以下方法得到不同的域对象所对应的Map集合：

getContextMap()：得到request域对象对应的Map集合；

getSession()：得到session域对象对应的Map集合；

getApplication()：得到application域对象对应的Map集合。

例如：

|  |
| --- |
| **public** String add() {  *// 得到ActionContext* ActionContext ac = ActionContext.*getContext*();  *// 得到request对应的Map* Map<String, Object> request = ac.getContextMap();  *// 存数据* request.put(**"date"**, **new** Date());  *// 得到session和application* Map<String, Object> session = ac.getSession();  Map<String, Object> application = ac.getApplication();  **return "success"**; } |

## 1.3 实现XxxAware接口（推荐）

Action类可分别实现RequestAware、SessionAware和ApplicationAware接口。之后实现接口的setXxx(Map<String, Object> map)方法。该方法中的参数就是获取到的对应的request、session和application的Map集合。可以在类中定义request等Map变量，通过setXxx()方法设置好值后，就可直接在类的各个地方使用了。

例如：

|  |
| --- |
| **public class** DemoAction **implements** RequestAware, SessionAware, ApplicationAware {  *// 下面的字段用于接收注入的变量* Map<String, Object> **request**;  Map<String, Object> **session**;  Map<String, Object> **application**;  @Override  **public void** setRequest(Map<String, Object> request) {  *// 进行注入* **this**.**request** = request;  }  @Override  **public void** setSession(Map<String, Object> session) {  **this**.**session** = session;  }  @Override  **public void** setApplication(Map<String, Object> application) {  **this**.**application** = application;  }  *// 下面在各个方法中即可方便的使用了。* **public** String add() {  **request**.put(**"name"**, **"张三"**);  **return "success"**;  } } |

总结：上述三种方式，常用最后一种，较方便。下面两种都和Servlet进行了解耦，但是有些场景还是需要用方式1解决问题。

# 2. 请求数据自定义类型转换

提交请求数据时，Struts会自动将参数值封装到属性中，但是这仅仅能够封装字符串和基本数据类型的值。对于Date类型，则只能封装“yyyy-MM-dd”格式的时间字符串，对于其他对象，其对象的属性类型也需要是字符串或者是基本数据类型。

那么如何将请求的字符串数据转换成其他的数据类型呢？比如我们想将“yyyy年MM月dd日 hh:mm:ss”格式的时间字符串参数值，直接封装成Date类型对象。

这时，我们需要自定义类型转换器。步骤如下：

（1）编写转换器类，该类要继承StrutsTypeConverter。例如需要上传一个人的生日，其时间格式可能是“xxxx年xx月xx日”，或是“19950115”，也可能是“1995-05-04”。要求该转换器都能使Struts进行成功转换。

代码示例（在某个包（比如com.util）下新建MyDateConverter）：

|  |
| --- |
| **package** com.util;  */\*\*  \* Created by zhang on 2017/3/9.  \*/* **public class** MyDateConverter **extends** StrutsTypeConverter {  *// 其中要重写两个方法。一个处理从String转成想要的类型，一个是将该类型转换成String。主要是写第一个方法。  // 先定义支持的时间格式* **private** DateFormat[] **df** = {  **new** SimpleDateFormat(**"yyyy-MM-dd"**),  **new** SimpleDateFormat(**"yyyy年MM月dd日"**),  **new** SimpleDateFormat(**"yyyyMMdd"**)  };  @Override  **public** Object convertFromString(Map context, String[] values, Class toClass) {  *// 说明：  // context表示当前上下文  // values表示表单提交过来的值  // toClass表示要转换的目标类型。该类型是根据Action类的属性传递过来的。能传递过来的原因就是等会我们将配置Action类与转换器之间的联系。总之，对本例来说，toClass的类型需要是Date类型，否则就意味着出错了，直接返回null   // 若无请求参数，直接返回null* **if** (values == **null** || values.**length** <= 0) {  **return null**;  }  *// 若传递过来的不是时间类型，说明可能配置有误，也返回null* **if** (Date.**class** != toClass) {  **return null**;  }  *// 这时就需要对字符串进行封装了。要遍历所有支持的时间格式，若满足转换条件（即正常，没有异常），直接返回转换成功的值* **for** (DateFormat format : **df**) {  **try** {  **return** format.parse(values[0]);  } **catch** (ParseException ex) {  *// 说明转换出错，还要继续，因为可能满足下一个格式。* **continue**;  }  }  *// 最后都不满足，返回null。* **return null**;  }   @Override  **public** String convertToString(Map map, Object o) {  **return null**; *// 这个实际不需要写* } } |

（2）配置转换器类，告诉struts应用此转换器。

需要在使用该转换器的action类所在包下，新建一个配置文件，文件名要满足“Action类名-conversion.properties”，内容就写：“表单提交的名称=转换器类名全路径”。比如在com.zhang.action下新建“DemoAction-conversion.properties”，内容是：

|  |
| --- |
| **birthday**=**com.util.MyDateConverter** |

当然，如果是user对象的birthday属性，那么就写user.birthday=com.util.MyDateConverter。

这样，就配置好了局部类型转换器。可以写Action和JSP了。比如Action中：

|  |
| --- |
| **public class** DemoAction {  **private** Date **birthday**;  **public** String add() {  System.***out***.println(**birthday**);  **return "success"**;  }   **public** Date getBirthday() {  **return birthday**;  }  **public void** setBirthday(Date birthday) {  **this**.**birthday** = birthday;  } } |

那么JSP上传birthday即可。大家自行测试。

上面配置的是局部类型转换器，该转换器只能给一个Action使用，如果想给多个Action使用，则可以用全局类型转换器。即在src下创建文件：xwork-conversion.properties，内容是“要转换的类型=转换器类”，即比如：

|  |
| --- |
| **java.util.Date**=**com.util.MyDateConverter** |

# 3. 文件的上传和下载 - Struts

## 3.1 文件上传

使用Struts上传文件很方便，只要在Action类中写好有关此文件上传的三个属性，那么在Action中就可拿到该上传文件的相关对象，而且这并不影响同时接收其他类型的参数（和以前相同）。Struts中是通过文件上传拦截器实现该功能的。

首先要知道，Struts默认限制文件上传的大小是2M，这是default.properties文件中对“struts.multipart.maxSize”Struts常量的默认配置（单位是字节）。因此，我们一般要根据实际情况修改这个常量配置，否则上传文件大小超过该配置，则Struts会报错。例如我们可以设置成30M：

|  |
| --- |
| **struts.multipart.maxSize**=**31457280** |

这里的31457280是通过30 \* 1024 \* 1024计算出来的，当然在配置文件中国不能写这样的表达式，需要直接写值。

下面通过例子来介绍文件上传。

（1）写好JSP文件，注意表单同样要设置为POST提交，并且设置enctype为“multipart/form-data”。例如下面的JSP中上传了一个文件和一个普通参数：

|  |
| --- |
| <%@ **page pageEncoding**="**UTF-8**" **language**="**java**" **isELIgnored**="**false**" %> <**html**> <**head**>  <**meta charset="UTF-8"** />  <**title**>上传文件</**title**> </**head**> <**body**>  <**form action="${**pageContext.request.contextPath**}/add.action" method="POST" enctype="multipart/form-data"**>  <**p**>提交人：<**input type="text" name="submitter"**></**p**>  <**p**>说明书文件：<**input type="file" name="specification"**></**p**>  <**p**><**input type="submit"**></**p**>  </**form**> </**body**> </**html**> |

（2）编写Action类。对于普通参数还是直接写属性，例如private String submitter。对于文件，需要用三个属性：

属性1是File对象，这就对应该文件的对象，对象名要和表单中名称一致，比如private File specification，即Struts直接能把上传文件封装成File对象，十分方便。

属性2和属性3都是String类型，分别对应文件的名称和文件的MIME类型，这两个属性名称规则分别是“表单中名称 + FileName”和“表单中名称 + ContentType”。

代码示例：

|  |
| --- |
| **package** com.zhang.action;  **import** java.io.File;  **public class** DemoAction {  *// 接收的普通参数* **private** String **submitter**;  *// 接收文件* **private** File **specification**; *// 文件* **private** String **specificationFileName**; *// 文件名* **private** String specificationContentType; *// 文件类型* **public** String add() {  System.***out***.println(**submitter**);  System.***out***.println(**specification**.getAbsolutePath());  System.***out***.println(**specification**.getName());  System.***out***.println(**specificationFileName**);  System.***out***.println(specificationContentType);  **return "success"**;  }   *// 一定要写所有属性的setter和getter。这里省略...* } |

然后自己配置action。当我们访问action提交数据和文件时，发现输出以下内容：

|  |
| --- |
| 09:29:29.164 INFO org.apache.struts2.dispatcher.Dispatcher.getSaveDir(Dispatcher.java:698) - Unable to find 'struts.multipart.saveDir' property setting. Defaulting to javax.servlet.context.tempdir  张三  D:\apache-tomcat\work\Catalina\localhost\JavaWebDemo\upload\_489500ef\_fe73\_4a90\_bb87\_1fa62c9c368a\_00000001.tmp  upload\_489500ef\_fe73\_4a90\_bb87\_1fa62c9c368a\_00000001.tmp  证件照.jpg  image/jpeg |

即Tomcat拿到上传文件后，会生成一个临时文件，而File对象得到的只是这个临时文件。我们需要对临时文件进行操作并且保存。而FileName和ContentType得到的是源文件的名称和类型。上述提示说因为没有配置“struts.multipart.saveDir”常量，因此临时文件存放在了默认临时目录。也可自行配置。不过临时文件会在此次访问之后自动删除。因此特别注意：只要用File对象进行文件流相关操作，而FileName才是原文件名，开发时要分清。

例如我们将上传的文件保存在服务器的E:\upload目录下：

|  |
| --- |
| **package** com.zhang.action;  **import** org.apache.commons.io.FileUtils;  **import** java.util.UUID;  **public class** DemoAction {  *// 接收的普通参数* **private** String **submitter**;  *// 接收文件* **private** File **specification**; *// 文件* **private** String **specificationFileName**; *// 文件名* **private** String **specificationContentType**; *// 文件类型* **public** String add() **throws** IOException {  *// 得到E:\\upload* File uploadDir = **new** File(**"E:\\upload"**);  *// 若不存在此目录，则创建* **if** (!uploadDir.exists()) {  uploadDir.mkdirs();  }  *// 用 uuid + “真”文件名 作为保存文件的名称* File saveFile = **new** File(uploadDir, UUID.*randomUUID*().toString() + **"-"** +**specificationFileName**);  *// 由于Struts引用了commons-io包，包中提供了方便操作文件的方法。可直接将一个File复制到另一个File。  // 这里就将specification复制到要保存地方的saveFile。* FileUtils.*copyFile*(**specification**, saveFile);  **return "success"**;  }   *// 一定要写所有属性的setter和getter...* } |

上述用到了UUID类，UUID是JDK提供的，在java.util包下，调用ramdomUUID()方法即可生成一个随机的UUID对象。再调用toString()即可得到字符串形式。UUID是通用唯一识别码(Universally Unique Identifier)，可用于生成数据库的主键，重复概率极低极低。UUID是128（二进制），用十六进制表示就是32位，上述生成时，默认会加上横线，比如“77944e43-c86e-4881-b9c4-5261cd59dcae”，可以再调用replace(“-”, “”)，替换掉。

上述还用到了commons-io包中的方法。commons-io是apache提供的组件，用于简化IO操作，可自行研究。

这样配置好action后，我们就能成功上传文件到E:\\upload目录下。也可以自行用流的方式复制文件，注意要关闭文件流。

## 3.2 文件下载

文件下载本身就较简单，可以直接使用Servlet的方法（这时就必须使用ServletActionContext来获取response对象了）。若处理下载的action只负责返回下载的文件而不需跳转页面，那么action方法中直接返回null，然后action配置时不需要写result节点。例如：

|  |
| --- |
| **public class** DemoAction {  **public** String down() **throws** IOException {  File file = **new** File(**"E:\\upload\\安装包.exe"**);  HttpServletResponse response = ServletActionContext.*getResponse*();  *// 设置响应头信息* response.setContentLengthLong(file.length());  response.setHeader(**"Content-Disposition"**, **"attachment;filename="** + URLEncoder.*encode*(file.getName(), **"UTF-8"**));  *// FileUtils可直接将文件发送到输出流输出。很方便* FileUtils.*copyFile*(file, response.getOutputStream());  **return null**;  } } |

struts.xml中不需配置result（当然若有跳转页面，则可以配置）：

|  |
| --- |
| <**action name="down" class="com.zhang.action.DemoAction" method="down"**> </**action**> |

相反，使用Struts进行文件下载就比较麻烦，这里就不讲了，没有必要。实在有兴趣可自行研究。

# 4. 结语

至此，Struts基本使用已经讲完，应付大部分开发已经没有问题。关于Struts还有较多的知识，比如：拦截器、Struts标签、Ognl表达式和值栈（StackValue）等，这些就不深入讨论了。大家可参照Struts官方文档自行研究。

但由于近几年Struts出现较多安全漏洞，使用人数较少，目前主流趋势是使用Spring框架。Spring可以说提供了Java EE的一整套解决方案，是Java EE的“事实标准”。因此也不推荐大家深入研究Struts，以后可重点深入研究后面要讲的Spring，因此这里对于Struts只要达到应用即可。