目标：

1. Struts访问通配符使用
2. struts常量
3. Struts核心业务

\*请求数据启动封装

\*类型转换

\*struts中数据处理

\*文件上传

# Strust2配置

### Struts2的Action的开发的几种方式

方式1：继承ActionSupport

|  |
| --- |
| public class UserAction extends ActionSupport{ |

如果用struts的数据校验功能，必须继承此类！

方式2：实现Action接口

|  |
| --- |
| public interface Action {  public static final String SUCCESS = "success";  public static final String NONE = "none";  public static final String ERROR = "error";  public static final String INPUT = "input";  public static final String LOGIN = "login";  public String execute() throws Exception;  } |

方式3：不继承任何类，不实现任何接口

|  |
| --- |
| **public** **class** UserAction3{ |

### 1.2 通配符

通配符：

在struts中配置信息中，可以用\*与{1}可以优化配置!

|  |
| --- |
| <package name=*"config"* namespace=*"/"* 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 -->  <action name=*"user\_\*"* class=*"cn.itcast.a\_config.UserAction"* method=*"{1}"*>  <result name=*"{1}"*>/{1}.jsp</result>  </action>  </package> |

### 1.3 struts中路径匹配原则

|  |
| --- |
| <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> |
| 访问路径:http://localhost:8080/struts02/user/user\_login OK |
| 访问路径:http://localhost:8080/struts02/user/a/b/user\_login OK |
| 访问路径:http://localhost:8080/struts02/a/b/user/user\_login Not OK |

<http://localhost:8080/struts02/user/a/b/user_login>

Tomcat:

Localhost:找到访问那一台机器

8080:找到tomcat

找到struts02:找到项目名称

/user/a/b:先看有没有这个名称空间，没找到，继续向下；找到就返回

/user/a:先看有没有这个名称空间，没找到，继续向下；找到就返回

/user:先看有没有这个名称空间，没找到，继续向下；找到就返回

/:默认名称空间，还没找到，报错！找到就返回

### 1.4 Struts常量

Struts中默认访问后缀：

Struts1中默认访问后缀是\*.do

Struts2中默认访问后缀是\*.action

如何修改默认访问后缀：

1.struts2的.action访问后缀在哪里定义？

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

Struts.action.extension=action

2.在struts.xml中通过常量修改

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

指定访问后缀为action/do/没有访问后缀都可以

Value=”action,do,” 访问后缀：action/do/不带后缀

Value=”action,do” 访问后缀：action或do

Value=”action” 只能是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"/> |

动态方法调用语法，

actionName+! 即为动态访问调用，如

|  |
| --- |
| <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>  --> |

### 1.5 全局跳转视图配置、配置的各项默认值

全局视图配置：

|  |
| --- |
| <!-- 配置全局跳转视图 -->  <global-results>  <result name=*"success"*>/index.jsp</result>  </global-results> |

配置各项默认值：

|  |
| --- |
| <!-- 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" type="redirect">/WEB-INF/index.jsp</result>-->  <result name=*"success"*>/WEB-INF/index.jsp</result>  </action> |

### Struts中数据处理

对数据操作的方法，（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request/session/application）的map

方式3：实现接口的方法：(RequestAware/SessionAware/ApplicationAware)

### 请求数据自动封装

实现原理：

参数拦截器

<interceptor name=”params” class=”com.opensymphony.xwork2.interceptor.Parameterslnterceptor”/>

方式1：Jsp表单数据填充到action中的属性:

|  |
| --- |
| Register.jsp |
| <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> |
| **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  \*  \*/  **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";  }  } |

### 类型转换

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

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

如果是别的格式，需要自定义类型转换器

局部类型转换器

全局类型转换器

Struts转换器API

|--TypeConverter 转换器接口

|--DefaultTypeConverter 默认类型转换器类

|--StrutsTypeConverter 用户编写的转换器类，继承此类即可

#### 局部类型转换器类

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)

总结：

转换器类，能否给其他Action用？不能。

#### 全局类型转换器类

需要写一个转换器，给所有的action用！

配置全局类型转换器：

--》src/xwork-conversion.properties

--》内容：

Java.util.Date=转换器类(cn.itcast.d\_type.MyConverter)

转换器类完整代码:

|  |
| --- |
| 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  \*  \*  \*/  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;  }  } |

### 5.文件上传

回顾文件上传：

Struts提交的文件上传组件上传文件

前台：

1. 提交方式POST
2. 表单类型 multipart/form-data
3. Input type=file

后台：

Apache提供的FileUpload组件

核心类：

FileItemFactory FileItem的工厂

ServletFileUpload servlet中文件上传的核心类

FileItem 封装了上传的表单文件项的信息

总之，文件上传，处理起来比较麻烦！

Struts的文件上传：

文件上传拦截器帮助我们完成了文件上传的功能：

<intercepter name=”fileUpload” class=”org.apache.struts2.interceptor.FileUploadInterceptor”/>

#### 文件上传Demo:

|  |
| --- |
| <body>  <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>  </body> |
| 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

Struts2默认支持的文件上传最大是2M

通过常量修改：

|  |
| --- |
| <!-- 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类型

|  |
| --- |
| <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> |

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视图对应的页面。

#### Struts的文件下载

文件下载：2种方式

方式1：通过response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”/>