Struts2框架是在struts1和webwork基础上的web框架

Struts2的配置文件名称为struts.xml，文件放在src目录下

Strut.xml一般配置格式：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"

"http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>

<constant name="默认常量名称" value="默认常量值"></constant>（struts修改配置默认常量也可在web.xml和struts.properties中配置）

<package name="包名用于区分action在不同的package标签下，无其他功能作用" extends="表示该package继承struts的默认类，该包下配置的action类具有控制器action的作用，一般固定默认是（struts-default）" namsxmlns="default" namespace="与action标签中的name名称构成访问路径，一般默认（/）">

<interceptors>

<interceptor name="拦截器类名称" class="拦截器类名称全路径"></interceptor>

</interceptors>

<global-results>

<result name="全局页面跳转名称，多个action可以跳转同一个页面"></result>

</global-results>（全局界面需要写在所有action之前）

[<action name="网站访问路径名（即名.action的名称）不能重复，一般使用通配符[\*]可以被多个路径访问" class="action类的全路径名称可以重复" method="指定action执行的方法，默认（execute），有多个访问路径需要使用{数值}（表示取出name属性中的第几个通配符的值）">

<interceptor-ref name="指定该action需要使用的拦截器类名">

<param name="excludeMethods（指定不拦截属性名）">不进行拦截的方法名称[,名称]</param>

</interceptor-ref>

<interceptor-ref name="指定默认拦截器总的堆栈包名称"></interceptor-ref>

<result name="指定跳转名与action类方法的返回值相同" type="指定如何到达指定路径地址（转发或者重定向，默认dispatcher转发在服务器内部转发，redirect重定向重新指定网址路径，chain转发到action由于缓存问题一般不用，redirectAction重定向到action）">跳转路径地址（/路径名称，/表示根目录），action直接写控制器名称</result>

</action>]

</package>

<include file="引入外部的配置文件路径名称"></include>

</struts>

Struts中的常用默认常量：struts.i18n.encoding=UTF-8，默认解决action获得的表单通过post提交中文乱码问题，不需要设置编码格式，get提交需要编码转换

Struts2所有请求都通过过滤器来实现，需要在web.xml文件中配置过滤器：

<filter>

<filter-name>action2</filter-name>

<filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>action2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

查看类内容control加鼠标左键点击或者control加shift加t进入选定的类中

过滤器在服务器启动的时候创建，创建完成后执行init方法

Struts分模块开发：可以将配置信息写在其他配置文件中，需要引入到struts核心配置文件中

Struts中action的创建有三种方式：创建一个类不继承类也不实现接口，创建一个类实现Action接口，创建一个类继承ActionSupport类（常用）

访问action的方法有三种方式：设置action标签中的method属性，使用通配符方式，动态访问实现（一般不用）

xml注释多行快键键：CTRL + SHIFT + /，撤销注释：CTRL + SHIFT + \

action类中的方法有返回值必须是string类型的返回值，可以没有返回值一般使用返回none字符串表示没有返回值

eclipse中web项目复制需要修改项目的Context root名字：项目右键->Properties->Web Project Settings->Context root修改名称与项目名称一致

action标签中配置的result是局部界面，局部界面优先级比全局界面优先级高

浏览器无缓存刷新页面：control加F5

Struts2中action获取表单数据的三种方式：

使用ActionContext类获取

ActionContext context=ActionContext.getContext();

Map<String, Object> map=context.getParameters();

Set<String> keys=map.keySet();

for (String key : keys) {

Object[] obj=(Object[]) map.get(key);

System.out.println(Arrays.toString(obj));

使用ServletActionContext类获取，常用

HttpServletRequest request=ServletActionContext.getRequest();

String 变量名=request.getParameter("属性名称");

使用接口注入方式，不常用

public class action类名 extends ActionSupport implements ServletRequestAware{

private HttpServletRequest request;

public void setServletRequest(HttpServletRequest request) {

this.request=request;

}

}

Action操作域对象

HttpServletRequest request=ServletActionContext.getRequest();

request.setAttribute("", "");

HttpSession session=request.getSession();

session.setAttribute("", "");

ServletContext contex=ServletActionContext.getServletContext();

contex.setAttribute("", "");

属性封装：在action类中设置成员变量名称与页面属性名称一致，可以自动将表单提交数据封装到action类中，不能将表单数据直接封装到实体类对象中

模型驱动封装（常用，只能将数据封装到一个实体类对象中）：action需要实现接口ModelDriven类中的getModel方法，action中创建实体类对象，属性封装与模型驱动封装一起使用，只会执行模型驱动封装

public class Demo2 extends ActionSupport implements ModelDriven<实体类名称> {

private 实体类名 实体类对象=new 实体类名();

@Override

public 实体类名getModel() {

// TODO Auto-generated method stub

return 实体类对象;

}

}

表达式封装：（不常用，可以将数据封装到多个实体类对象中）在action中声明实体类，action中有实体类变量的set和get方法，并且表单输入项的name属相值里面有表达式形式（即<input type="输入类型" name="action中的实体类对象名.属性名">）

使用表达式封装将数据封装到list集合中，action中声明list，生产list的set和get方法，表单输入项的name属相值里面有表达式形式（即<input type="输入类型" name="list[数值（表示list中第几个元素，从0开始）].属性名">）

使用表达式封装将数据封装到map集合中，action中声明map，生产map的set和get方法，表单输入项的name属相值里面有表达式形式（即<input type="输入类型" name="map[‘设置字符串key’].属性名">）

EL表达式在jsp中获取域对象的值

OGNL表达式需要struts2标签操作值栈数据，OGNL并不是与struts2一体的也可以单独使用

Jsp中需要<%@ taglib uri="/struts-tags" prefix="s" %>引入struts2的标签库才能使用struts标签

ognl表达式写在struts指定的标签中<s:property value="ognl表达式"/>

struts2中有一种类似域对象的数据存储机制值栈，在action里将数据存放到值栈中，在页面可以从值栈中可以获取数据

sevlet：默认在第一次访问时创建，只创建一次，单实例对象

action：每次访问时都会创建相应的action对象，多实例对象

每个action对象里都只有一个值栈对象

获取值栈对象的多种方式：

常用方式：使用ActionContext类里面的方法获得值栈对象

ActionContext 名称=ActionContext.getContext();

ValueStack 值栈对象名=名称.getValueStack();

值栈内部结构分为root（list集合，常用），Context（map集合，不常用）

Jsp页面使用struts2标签<s:debug></s:debug>可以在页面中查看值栈的结构

Action没有任何操作默认值栈的栈顶元素是action的引用

向值栈中存取数据有多种方式：

调用值栈对象的set方法存放数据：值栈对象.set("属性名称","值")

Set方法存放的数据获取方式：<s:property value="属性名称"/>

调用值栈对象的push方法存放数据：值栈对象.push("值")

Push方法存放的数据获取方式：<s:property value="[（数值，top数组的下标，下标从栈顶开始，后存放的元素在栈顶）].top"/>（向值栈存放数据会存到名为top的数组中，ognl数组表达式为[下标].top）

（常用）在action定义成员（变量，对象，list集合）生成成员（变量，对象，list集合）的get方法，在action的执行方法中（即execute）设置成员（变量，对象，list集合）值，成员（变量，对象，list集合）在值栈中的位置是action中，不需要开辟新的存储空间

在jsp页面使用struts2和ognl表达式从值栈中获取：

字符串<s:property value="属性名称"/>

对象值<s:property value="对象名.属相名"/>

在jsp页面使用struts2和ognl表达式从值栈中获取List集合有多种方法：

<s:property value="list集合名称[数值（表示第一个元素）.属性名]"/>

<s:iterator value="list集合名称">（该标签会遍历list集合中所有元素）

<s:property value="集合中对象的属性名"/>

</s:iterator>

<s:iterator value=" list集合名称" var="对象名称">（每次遍历的对象会放到Context里）

<s:property value="#对象名.属性名"/>（使用ognl表示获取Context中的数据需要加#）

</s:iterator>

Jsp页面中<!—html注释 -->不能注释外部引用的标签，只能通过<%--jsp注释 --%>进行注释

使用jstl标签中的foreach标签加el表达式获取值栈中的list集合：

<c:forEach items="${list集合名}" var="对象名">

${对象名.属性名}

</c:forEach>

El表达式获取的是域对象中的值，域对象存取值是setAttribute和getAttribute方法，底层增强了request对象的getAttribute方法，getAttribute方法先从request域对象中取值，取到则返回，没有取到则把值栈中获取的值放到域对象中获取

使用ognl表达式获取Context中的数据<s:property value="#context的key名称.属性名称"/>

在struts表单标签中使用ognl表达式需要加上%才识别为ognl表达式：

<s:textfield name="文本域名称" value="%{ognl表达式}"></s:textfield>

Struts2框架封装的功能都在拦截器里，有多个拦截器，一般执行默认的拦截器，拦截器执行在action对象创建之后，action的方法执行之前

Struts2的思想：aop（面向切面编程）思想，责任链模式（程序执行多个操作，每个操作完成后才能进行下一个操作）

过滤链模式：一个请求经过多个过滤器进行过滤，每个过滤器放行才能到下一个过滤器

拦截器只可以拦截action，过滤器理论上可以过滤任何内容

拦截器结构：拦截器类继承了AbstractInterceptor类，AbstractInterceptor类实现了Interceptor接口，实现了接口中的init()，destroy()，intercept(ActionInvocation invocation)方法

一般自定义拦截器是让拦截器类继承MethodFilterInterceptor类，MethodFilterInterceptor类可以选择action中某个方法不进行拦截：

public class 拦截器类名extends MethodFilterInterceptor {

protected String doIntercept(ActionInvocation invocation) throws Exception {

return invocation.invoke();//拦截完成返回释放拦截，会根据返回值到达指定界面

}

}

拦截器配置：在需要使用拦截器的action标签所在的package标签中声明拦截器，在使用拦截器的action中配置使用的拦截器

Struts2当action配置自定义拦截器后将不会执行默认的拦截器，需要把默认的拦截器重新配置到action中，有需要不进行拦截的方法可以在拦截器的配置标签中进行配置

拦截器进行拦截action需要进行相应的配置文件即aop思想

${取值是最好加上范围scope（即sessionScope.属性名）}

Struts2表单标签：

<s:form theme="simple（设置表单主题，simple表示不自动换行）">表单标签

<s:textfield name="名称" label="页面前部标签名"></s:textfield>普通文本输入框

<s:password name="名称" label="页面前部标签名"></s:password>密码输入框

<s:radio list="{'（单选框选项名，名称与值一致）'[,'']}/#{'（单选框名称对应值）':'（单选框选项名，名称与值一致）'[,'':'']}" name="名称" label="页面前部标签名"></s:radio>单选框

<s:checkboxlist list="" name="名称" label="页面前部标签名"></s:checkboxlist>复选框

<s:select list="" name="名称" label="页面前部标签名"></s:select>下拉列表

<s:file name="名称" label="页面前部标签名"></s:file>

<s:hidden name="名称" label="页面前部标签名"></s:hidden>

<s:textarea rows="页面显示行数" cols="列数" name="名称" label="页面前部标签名"></s:textarea>

<s:submit value="名称"></s:submit>

<s:reset value="名称"></s:reset>

</s:form>