# Struts2 环境配置

1. 新建java web 工程
2. 讲struts2下载后解压包路劲中的apps中struts2-blank解压
3. 将目录下的WEB-INF/classes中的struts.xml配置文件考入javaweb工程src目录下
4. 引入WEB-INF/lib 中的jar包
5. 将WEB-INF下web.xml中的filter配置考入javaweb工程的web.xml文件中
6. 将struts.xml中示例package注释 并增加自己的配置(请求,视图映射关系)

<package name=*"default"* namespace=*"/"* extends=*"struts-default" namespace=””*>

<action name=*"hello11" method=”someFun”*>

<result name=”success”>

/Hello.jsp

</result>

</action>

</package>

将以上配置完成后 访问 localhost:8080/项目名称/namespace值,没有 重写就默认为空/hello11 则当方法返回值为success(不写默认为success)时 struts2输出Hello.jsp页面内容

# 基础使用方法

Action 基本配置 Action的作用和运行流程

1. 建立java类 集成supportAction 重写excuse方法 (默认配置)
2. 配置struts.xml文件
3. 使用通配符 开发约定优于规则
4. 传参 直接get传参 模型保存传值 DTO中间层转存,再设置到模型(用于提交数据与模型数据不完全匹配的中间层转层一道)
5. 乱码解决 使用spring的过滤器

# View展现相关 数据保存和页面反馈

1. 表单的数据简单验证,在Action中进行数据校验,并将错误信息使用 this.addFieldError方法进行储存错误信息,jsp页面使用struts2标签进行<s:property value=”errors.储存的错误名字[序号]”/>进行显示

2.request session application(

2.1访问web元素 拿到request session application(

使用Map进行将对象初始化,然后在jsp页面拿到,struts内部会将值全部映射到相对应的对象中)

拿到上一步保存的值,OGNL表达式 使用#request.设置的名字 可以拿到/也可以使用request.getAttrubite(“设置的名字”)也可以 || 使用 #attr.属性名字来访问时会找各个对象

2.2使用依赖注入的方式获取request session application(最常用,不用自己new)

RequestAware, SessionAware, ApplicationAware 实现这三个接口 并重写方法,定义Map类型的request session application的变量 并在Action执行方法的时候将需要保存的值存储,在jsp页面取的即可

2.3拿到真实的三大对象

定义 并初始化 然后设置值

**private** HttpServletRequest request;

**private** HttpSession session;

**private** ServletContext application;

2.4 实现ServletRequestAware 接口并定义三个对象,将他们初始化

# 模块包含

1. 使用<include file=”xxxxx.xml” />配置将一个XML文件进行包含,相当于复制过来,开发中避免同时修改一个xml的冲突

# 默认Action

指定某个Action为默认,不用敲则可访问.访问一个不存在的action,也直接跳转到默认Action

1. 使用配置 <default-action-ref name=”package中的某个action的配配置的name”></default-action-ref>

# Action总结

1. 实现一个Action最常用的方式:从ActionSupport继承
2. DMI动态方法调用 !
3. 通配符 \*\_\* {1}{2}...
4. 接收参数的方法 (一般使用DomainModel来接收)
5. 简单参数的验证 addFieldError
6. 访问web元素 (Map类型:IoC,依赖Struts2 || 原始类型:IoC,依赖Struts2)
7. 包含文件配置
8. 默认Action的处理

# Result基础

<!-- 可以配置result的类型 不指定默认为dispatcher (运用服务器跳转,forward) -->

<!-- dispatcher 只能跳转到一个jsp页面 -->

<!-- chain 服务器跳转到一个Action -->

<!-- redirect 重定向 到jsp-->

<!-- redirectAction 重定向到一个Action -->

# Global-result

当 返回global-result中配置的result name的时候,统一跳转(同一package中)

<global-results>

<result name=*"globalPage"*>/error.jsp</result>

</global-results>

# Package的继承

可以继承一个自定义的package,并可以使用里面配置的global-result 或者 全局不存在的action的跳转

使用关键字 extends 继承某个package即可

# 动态结果配置

在目标action中定义参数,根据逻辑的不同返回不同的jsp页面,jsp页面的字符串使用另一个参数,并将配置中的值指定为这个参数 action中定义了 String r; 例如 r = “1.jsp”

<package name=*"denamicResult"* namespace=*"/denamic"* extends=*"struts-default"*>

<action name=*"denamicResult"* class=*"com.wanggang.action.DenamicResultAction"*>

<result name=*"success"*>${r}</result>

</action>

</package>

# 带参数的结果转发

Action接受到一个请求,并获得这个请求带的参数,将这个参数redirect到另一个地方 action中定义了type

<!-- 带参数的结果 -->

<package name=*"param"* namespace=*"/param"* extends=*"struts-default"*>

<action name=*"p"* class=*"com.wanggang.action.ParamAction"* method=*"getParam"*>

<result name=*"success"* type=*"redirect"*>/param.jsp?type=${type}</result>

</action>

</package>

这个词保存在parameters中,使用

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

# OGNL简单的URL带的值获取显示

<s:property value=*"user.namel"*/> 标签 value=”这里是OGNL表达式获取URL中带着的参数值” (Action中必须定义,再生成set get 方法后使用)

% 将内部内容转换为OGNL表达式

#属性名字 取context中的值

属性名字 取值站里面的值

$ 在配置文件中使用属性的值作为内容

# Struts2 Tags

1. 通用标签

<s:property/>取值

<s:set/>设置值 有scope 指定位置 默认为request actionContext中

<s:bean/>

<s:include/>

<s:param/>

<s:debug><s:debug/>

1. 控制标签

If else if else

Iterator迭代

Subset截取

1. UI标签

# Struts2 全局错误页面配置

<!-- 错误页面 -->

<package name=*"BBS"* extends=*"struts-default"*>

<global-results>

<result name=*"error"*>/error.jsp</result>

</global-results>

<global-exception-mappings>

<exception-mapping result=*"error"* exception=*"java.lang.Exception"*></exception-mapping>

</global-exception-mappings>

</package>

# Json数据API写法

使用jackson 完成json数据转换

//测试接口

public class TestAction extends ActionSupport {

public void test() throws IOException{

HttpServletResponse res = ServletActionContext.getResponse();

res.setContentType("text/html;charset=utf-8");

Map m = new HashMap();

List list = new ArrayList();

list.add("111");

list.add("1121");

list.add("1131");

list.add("1141");

list.add("11413411");

m.put("state", "SUCCESS");

m.put("msg", "返回成功");

m.put("response", list);

ObjectMapper mapper = new ObjectMapper();

String jsonstr = mapper.writeValueAsString(m);

System.out.println(jsonstr);

res.getWriter().write(jsonstr); //返回字符串

}

}