## struts2配置

### 1.1 struts2中action开发的几种方式

方式1：继承ActionSupport

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

如果使用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ActionSupport，也不实现Action接口

### 1.2通配符

通配符：在struts配置信息中，\*和{1}可以优化配置

|  |
| --- |
| <!-- 使用通配符优化配置 -->  <action name=*"user\_\*"* class=*"com.zhenhunfan.a\_config.UserAction3"* method=*"{1}"*>  <result name=*"{1}"*>/JSP/a\_config/{1}.jsp</result>  </action> |

### 1.3 struts常量

struts中默认访问后缀

struts1中是\*.do

struts2中是\*.action

访问：<http://localhost:8081/StrutsDemo2/user/login> OK

访问：<http://localhost:8081/StrutsDemo2/user/login.action> OK

访问：<http://localhost:8081/StrutsDemo2/user/login.do> NO

配置文档中action的name属性不能配置成xxx.action

**如何修改默认访问后缀：**

1. struts2的.action的访问后缀在哪定义

struts2-core-2.2.1.jar/org.apache.struts2/default.properties

struts.action.extension=action,,

1. 在struts.xml中通过常量修改后缀

|  |
| --- |
| <constant name=*"struts.action.extension"* value=*"action,do,"*></constant> |

代表后缀可为action、do或者没有

|  |
| --- |
| 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：

actionName!methodName

### 1.4 全局跳转视图配置、配置的各项默认值

全局跳转视图配置

作用：当多个返回结果都是跳转到相同的页面上时，减少配置项

|  |
| --- |
| <!-- 全局跳转视图配置 -->  <global-results>  <result name=*"success"*>/JSP/a\_config/index.jsp</result>  </global-results> |

## struts中的数据处理

方式1：直接获取servletapi

核心类：ServletActionContext提供的静态方法

优点：相比其他方式，可以获取更多的信息，比如项目路径等

缺点：在使用action的时候，引入了servlet，有潜在的隐患

方式2：通过ActionContext获取不同的map（request/session/application）

优点：书写简单

缺点：没法获取额外信息，同时每个方法里面都需要调用，影响性能

|  |
| --- |
| /\*\*  \* 演示数据处理  \* **@author** zhanghong  \*  \*/  **public** **class** DataAction **extends** ActionSupport{      @Override  **public** String execute() **throws** Exception {    /\*\*  \* 数据处理  \* 方式1： 直接拿到ServletApi, 执行操作  \*/  /\*HttpServletRequest request = ServletActionContext.getRequest();  HttpSession session = request.getSession();  ServletContext application = ServletActionContext.getServletContext();  // 操作  request.setAttribute("request\_data", "request\_data1");  session.setAttribute("session\_data", "session\_data1");  application.setAttribute("application\_data", "application\_data1");\*/        /\*\*  \* 【推荐：解耦的方式实现对数据的操作,减少对servlet的依赖】  \* 方式2： 通过ActionContext类  \*/  /\*ActionContext ac = ActionContext.getContext();  // 得到Struts对HttpServletRequest对象进行了封装，封装为一个map  // 拿到表示request对象的map  Map<String,Object> request = ac.getContextMap();  // 拿到表示session对象的map  Map<String, Object> session = ac.getSession();  // 拿到表示servletContext对象的map  Map<String, Object> application = ac.getApplication();    // 数据  request.put("request\_data", "request\_data1\_actionContext");  session.put("session\_data", "session\_data1\_actionContext");  application.put("application\_data", "application\_data1\_actionContext");\*/    **return** ***SUCCESS***;  }  } |

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

优点：性能高，可以用于写BaseAction

缺点：书写复杂，也不能获取额外信息

|  |
| --- |
| /\*\*  \* 数据处理, 方式3： 实现接口的方法  \* **@author** zhanghong  \*  \*/  **public** **class** DataAction2 **extends** ActionSupport **implements** RequestAware, SessionAware, ApplicationAware{    **private** Map<String, Object> request;  **private** Map<String, Object> session;  **private** Map<String, Object> application;    // struts运行时候，会把代表request的map对象注入  @Override  **public** **void** setRequest(Map<String, Object> request) {  **this**.request = request;  }    // 注入session  @Override  **public** **void** setSession(Map<String, Object> session) {  **this**.session = session;  }    // 注入application  @Override  **public** **void** setApplication(Map<String, Object> application) {  **this**.application = application;  }  @Override  **public** String execute() **throws** Exception {    // 数据  request.put("request\_data", "request\_data1\_actionAware");  session.put("session\_data", "session\_data1\_actionAware");  application.put("application\_data", "application\_data1\_actionAware");    **return** ***SUCCESS***;  }  } |

## 请求数据自动封装和类型转换

JSP表单数据填充到action的属性

JSP表单数据填充到action的对象属性

实现原理：使用了参数拦截器

action对基本类型和日期类型会自动转换

假如要把特别的字符串转换成特别的对象，需要用到类型转换器

**Struts转换器api**

|---------TypeConvert 转换器接口

|----DefultTypeConvert 默认类型转换器

|-----StrutsTypeConvert struts类型转换器

**Struts2中如何配置自定义转换器?**

1、自定义转换器继承StrutsTypeConverter

2、重写convertFromString和convertToString方法

3、注册转换器

3.1 在Action所在包中建立

Action名-conversion.properties

3.2 在3.1文件中添加以下数据

需要转换的字段名=自定义转换器类的权限定名

birthday=com.zhenhunfan.d\_type.MyConverter