目录

[Struts2概述 9](#_Toc28843)

[1. Struts2框架应用javaee三层结构中web层框架 9](#_Toc17276)

[2. Struts2框架在Struts1和webwork基础之上发展全新的框架 9](#_Toc32475)

[3. Sturts2解决的问题 9](#_Toc3575)

[5. web层常见框架 9](#_Toc10602)

[（1） struts2 9](#_Toc5995)

[（2） springMVC 9](#_Toc31815)

[Struts2框架入门 9](#_Toc12616)

[Execute返回值必须是string!!!!!! 10](#_Toc22264)

[（1） 创建struts2核心配置文件 10](#_Toc2438)

[（2） 引入dtd约束（也是通过示例程序的复制过来） 10](#_Toc29169)

[Struts2执行过程 11](#_Toc14236)

[查看源代码 12](#_Toc11331)

[1. 过滤器在服务器启动时创建，创建过滤器时执行init方法 12](#_Toc30219)

[（1） 在init方法中主要加载配置文件 12](#_Toc30573)

[Struts2配置 12](#_Toc5449)

[（1） Sturts2的核心配置文件 12](#_Toc23324)

[标签package 12](#_Toc6054)

[（1） name属性 12](#_Toc22938)

[（2） extends属性 12](#_Toc25974)

[（3） namespace属性 12](#_Toc25455)

[标签action 13](#_Toc200)

[1. action标签配置action访问路径 13](#_Toc12062)

[2. action标签属性 13](#_Toc27754)

[（1） name属性 13](#_Toc2376)

[（2） class属性 13](#_Toc27485)

[（3） method属性 13](#_Toc13894)

[标签result 13](#_Toc28799)

[1. 根据action的方法返回值，配置到不同的路径里面 13](#_Toc26020)

[2. result标签属性 13](#_Toc31926)

[（1） name属性 13](#_Toc28207)

[（2） type属性 13](#_Toc26409)

[Struts2常量配置 14](#_Toc32695)

[（1） 常用的方式 14](#_Toc10380)

[（2） 还有两种方式（了解） 14](#_Toc8125)

[介绍常用常量 14](#_Toc18831)

[分模块开发 15](#_Toc12441)

[1. 单独写配置文件，把配置文件引入到核心配置文件中 15](#_Toc21991)

[Action三种编写方式 15](#_Toc32489)

[1. 创建普通类，不继承任何类，也不实现任何接口（自己写execute方法） 15](#_Toc145)

[2. 创建类，实现Action接口 15](#_Toc28477)

[3. 创建类，继承ActionSupport类 16](#_Toc19263)

[访问action里面的其他方法（重点！！！） 16](#_Toc30300)

[1.有三种实现方式 16](#_Toc21002)

[2. 错误演示 16](#_Toc6306)

[（1） result标签里的属性值和execute的返回值不同 16](#_Toc9697)

[（2） Execute有返回值的话必须是String 16](#_Toc20796)

[（3） execute方法可以没有返回值，没有返回值时，在result标签中不需要配置 16](#_Toc13138)

[3. 使用action标签method属性 17](#_Toc2125)

[（1） 创建方法 17](#_Toc24725)

[（2） 使用Method配置 17](#_Toc14150)

[4. 使用通配符（重点！！！使用最多） 17](#_Toc9480)

[1. 在action属性标签里有name属性，name属性里面写\*星号 17](#_Toc5366)

[（1） \*理解，表示匹配任意内容 17](#_Toc11646)

[5. 动态访问（知道即可，不用） 18](#_Toc12061)

[结果页面配置 19](#_Toc31758)

[全局结果页面配置 19](#_Toc17794)

[局部结果页面配置 19](#_Toc10049)

[result标签中的type属性 20](#_Toc6881)

[（1） 默认值做转发操作，dispatcher 20](#_Toc15316)

[（2） 重定向操作，redirect 20](#_Toc21041)

[在action里获取表单提交的数据 21](#_Toc7530)

[（1） 使用ActionContext类 21](#_Toc25348)

[（2） 使用ServletActionContext类 21](#_Toc26074)

[（3） 使用接口注入方式（一般不用） 21](#_Toc17945)

[使用ActionContext类获取 21](#_Toc11)

[（1） 不是静态方法，要创建类使用 21](#_Toc17241)

[（2） 这个ActionContext类对象不是New出来的 21](#_Toc25200)

[1.具体演示 21](#_Toc21715)

[（1）创建表单，提交表单到action 21](#_Toc11941)

[（2）在action中使用ActionContext来获取表单数据 21](#_Toc28270)

[通过ServletActionContext类获取 22](#_Toc26305)

[（1） 调用类里面的静态方法，得到request对象 22](#_Toc14452)

[使用接口注入 22](#_Toc11852)

[在action操作域对象 23](#_Toc31516)

[Struts2封装获取表单数据方式 23](#_Toc31759)

[原始方式获取表单封装到实体类对象 23](#_Toc11198)

[属性封装 24](#_Toc2823)

[1. 直接把表达提交属性封装到action的属性里面 24](#_Toc20572)

[2. 实现步骤 24](#_Toc3136)

[（1） 在action成员变量位置定义变量 24](#_Toc21656)

[（2） 生成变量的set和get方法 24](#_Toc26742)

[模型驱动封装（重点） 24](#_Toc860)

[1.使用模型驱动，可以把表单数据直接封装到实体类对象 24](#_Toc6042)

[2.实现步骤： 24](#_Toc12988)

[（1）action实现ModelDriven接口 24](#_Toc1463)

[（3） 实现接口里面的getModel方法 -把创建对象返回 24](#_Toc23507)

[（3）在action里面创建实体类对象 24](#_Toc30667)

[（4） 使用模型驱动和属性封装注意问题： 25](#_Toc26288)

[表达式封装（会用）（也可以归类到属性封装里） 25](#_Toc14473)

[1. 实现过程 25](#_Toc29329)

[（1） 使用表达式也可以将表单数据封装到实体类对象里面 25](#_Toc21548)

[第一步 在action里面声明实体类 25](#_Toc25847)

[第二步 生成实体类的set和get方法 25](#_Toc9138)

[第三步 在表单输入项的name属性值里写表达式 25](#_Toc7770)

[比较表达式封装和模型驱动封装 26](#_Toc11078)

[表单数据封装到集合 26](#_Toc27016)

[封装到list集合 26](#_Toc11423)

[第一步：在action里面声明List类 26](#_Toc8948)

[第二步：生成变量的get和set方法 26](#_Toc5526)

[第三步：在表单输入项里面输入表达式 27](#_Toc16255)

[封装到map集合 27](#_Toc29771)

[第一步：在action里面声明List类 27](#_Toc22143)

[第二步：生成变量的get和set方法 27](#_Toc9613)

[第三步：在表单输入项里面输入表达式 28](#_Toc12563)

[OGNL概述 28](#_Toc22370)

[1. 之前web阶段，学习过EL表达式，EL表达式在jsp中获取域对象里面的值 28](#_Toc12549)

[2. OGNL是一种表达式，这个表达式的功能更强大 28](#_Toc2267)

[（1） 在struts2里面操作值栈数据 28](#_Toc13256)

[（2） 一般把ognl在struts2操作，和Srtuts2标签一起食用操作值栈 28](#_Toc14757)

[3. OGNL不是struts2的一部分，是单独的项目，经常和struts2一起食用 28](#_Toc31314)

[（1）使用ognl时候先导入jar包，struts2提供Jar包 28](#_Toc20205)

[OGNL入门案列 28](#_Toc9445)

[1. 使用ognl和struts2实现计算字符串长度 28](#_Toc8336)

[（1） 在java代码中调用字符串.length(); 28](#_Toc2566)

[2. 使用struts2标签 28](#_Toc30900)

[（1） 使用jtsl时，导入jar包之外，在jsp页面引入标签库 28](#_Toc10793)

[（2） 使用struts2表情实现操作 29](#_Toc24890)

[什么是值栈 29](#_Toc19944)

[获取值栈对象 30](#_Toc13737)

[值栈内部结构 30](#_Toc2746)

[1. 值栈分为两部分 30](#_Toc25982)

[第一部分：root,结构是list集合 30](#_Toc2685)

[第二部分：context,结构map集合 31](#_Toc29198)

[2. struts2里面标签s:debug，使用这个标签可以查看值栈结构和存储值 31](#_Toc5214)

[向值栈放数据 32](#_Toc19544)

[1. 向值栈放数据的多种方式 32](#_Toc2522)

[第一种 获取值栈对象，调用值栈对象里面的Set方法 32](#_Toc6774)

[第二种 获取值栈对象，调用值栈对象里面的push方法 33](#_Toc24813)

[第三种（常用） 在action类定义变量，生成变量的get方法 33](#_Toc27025)

[向值栈放对象 33](#_Toc11446)

[第一步 定义对象变量 33](#_Toc14471)

[第二步 生成变量Get方法 33](#_Toc4103)

[第三步 在执行方法里设置对象值 33](#_Toc26783)

[向值栈放List集合 34](#_Toc4547)

[第一步 定义List集合变量 34](#_Toc19711)

[第二步 生成变量Get方法 34](#_Toc18690)

[第三步 在执行方法里设置对象值或添加内容 34](#_Toc5831)

[从值栈获取数据 34](#_Toc18239)

[获取字符串 34](#_Toc2805)

[获取对象 35](#_Toc6438)

[获取List集合 35](#_Toc6276)

[第一种方法 35](#_Toc15045)

[第二种方法 35](#_Toc22626)

[第三种方法 35](#_Toc541)

[其他操作 36](#_Toc3628)

[1. 使用set方法向值栈存数据，取数据 36](#_Toc23402)

[2. 使用push方法向值栈存数据，取数据 36](#_Toc14303)

[（1） 使用Push方法设置值，只有数据，没有名称 36](#_Toc19621)

[（2） 向值栈放数据，把数据存到数组里面，通过数组获取数据 36](#_Toc27601)

[ONGL的#、%使用 36](#_Toc8709)

[#使用 36](#_Toc32688)

[1.使用#获取Context里面的数据 36](#_Toc29576)

[3. 演示操作 37](#_Toc2149)

[%使用 37](#_Toc29646)

[拦截器底层原理 38](#_Toc514)

[AOP思想 38](#_Toc15054)

[2. 画图演示 39](#_Toc10274)

[责任链模式 39](#_Toc4907)

[AOP思想和责任链模式如何应用到拦截器 40](#_Toc2673)

[（2）画图分析 40](#_Toc18225)

[（3） 查看源代码 40](#_Toc1577)

[重要的概念 41](#_Toc4806)

[过滤器和拦截器区别 41](#_Toc21709)

[Servlet和action区别 41](#_Toc18897)

[自定义拦截器 41](#_Toc8554)

[自定义登录拦截器 42](#_Toc32483)

[第一步：创建类，继承MethodFilterInterceptor类 43](#_Toc24862)

[第二步：重写MethodFilterInterceptor类里面的方法写拦截器逻辑 43](#_Toc5161)

[第三步：配置action和拦截器关系（注册拦截器） 44](#_Toc20675)

[配置不拦截方法 44](#_Toc12529)

[Struts2的标签库 45](#_Toc18661)

[Struts2表单标签 46](#_Toc10306)

# Struts2概述

1. Struts2框架应用javaee三层结构中web层框架
2. Struts2框架在Struts1和webwork基础之上发展全新的框架
3. Sturts2解决的问题



1. 
2. web层常见框架
3. struts2
4. springMVC

# Struts2框架入门

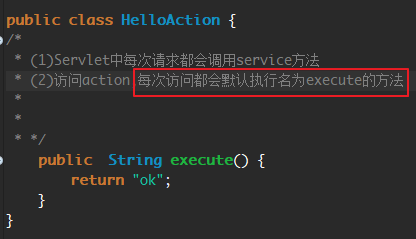
第一步：导入jar包（用示例Web复制）





用压缩软件打开→WEB-INF→lib复制所有jar包

第二步：创建Action类



**Execute返回值必须是string!!!!!!**

第三步：配置Action的访问路径

1. 创建struts2核心配置文件

-核心配置文件名和位置是固定的

-位置必须在src下面，名称struts.xml

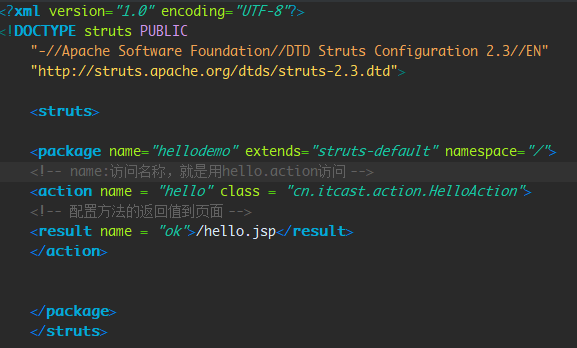
1. 引入dtd约束（也是通过示例程序的复制过来）

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

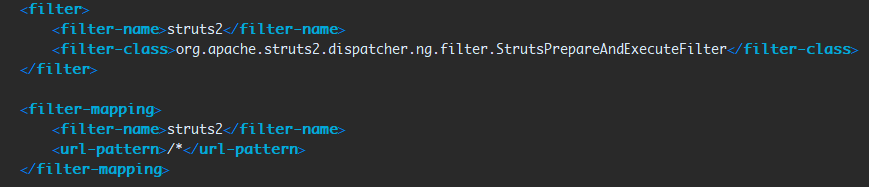
"http://struts.apache.org/dtds/struts-2.3.dtd">

（3）



访问路径http://localhost/struts\_day01/**hello.action**

**第四步：在web.xml配置过滤器（重要，否则不能访问，同样用示例文件的web.xml复制）**



<filter>

<filter-name>struts2</filter-name>

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

</filter>

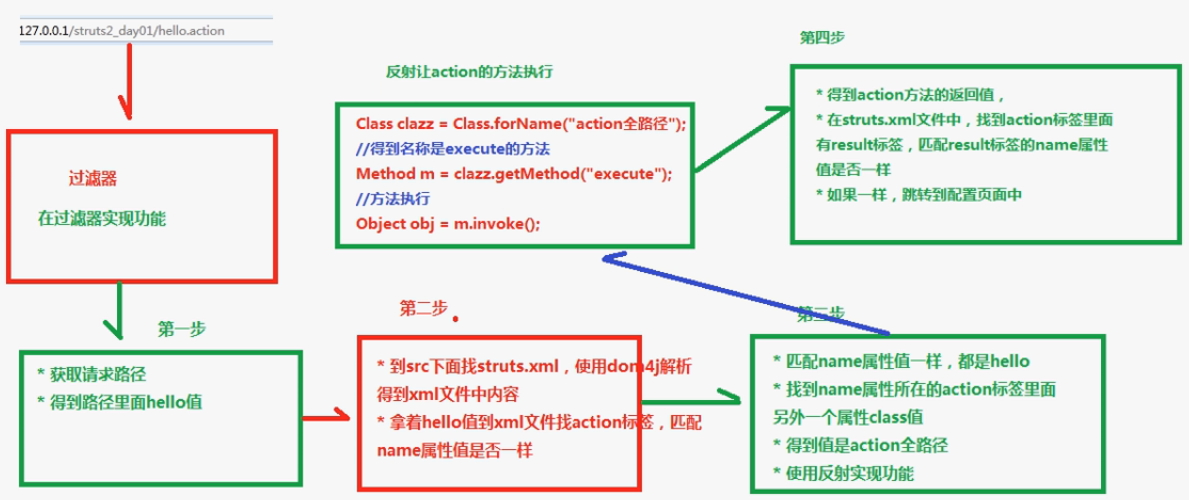
<filter-mapping>

<filter-name>struts2</filter-name>

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

</filter-mapping>

# Struts2执行过程



# 查看源代码



1. 过滤器在服务器启动时创建，创建过滤器时执行init方法
2. 在init方法中主要加载配置文件

-包含自己创建的配置文件和struts2自带的配置文件

Struts2.xml web.xml

# Struts2配置

## Sturts2的核心配置文件

1. 名称和位置是固定的 （src下创建struts.xml）
2. 在配置文件中主要有三个标签 package,action,result标签里面的属性

## 标签package

1. 类似于代码包，区别不同的action，要配置action，必须重写package标签，在package标签里面才能配置action
2. package标签属性
3. name属性

-name属性值跟功能本身没有关系

-在一个配置文件中可以多个package标签，name属性不能相同

1. extends属性

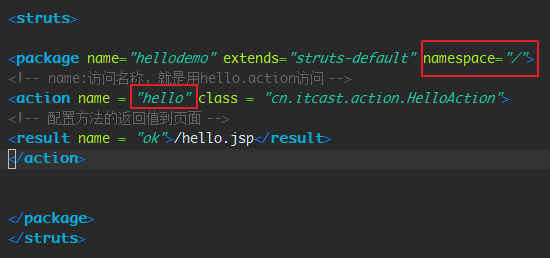


-属性值是固定的,struts-default

-写了这个属性之后，在package里面配置的类有action功能

1. namespace属性

-namespace属性值和action标签里面的name属性值构成访问路径



构成访问路径的/hello.action

## 标签action

1. action标签配置action访问路径
2. action标签属性
3. name属性

-namespace出行之和action标签里面的name属性值构成访问路径

-在package标签里面写多个action标签，但是Action的name属性值不能相同

1. class属性

-action的全路径

1. method属性

-不如在action里面默认执行的方法execute方法，但是在action里面写其他的方法

-让action里面多个方法执行，使用method进行配置

## 标签result

1. 根据action的方法返回值，配置到不同的路径里面
2. result标签属性
3. name属性

-和execute方法的返回值相同

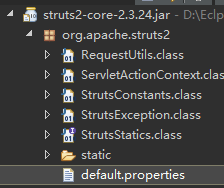
1. type属性

-配置如何到路径中（转发或重定向）

-type属性默认做转发操作

## Struts2常量配置

1. struts2框架，帮我们实现一部分功能，struts2里面有常量，在常量里面封装一部分功能
2. Struts2默认的常量位置（记住）



3.修改struts2默认常量值

1. 常用的方式

-在sturts.xml中设置



在<struts>标签下

1. 还有两种方式（了解）

-在src下面创建struts.properties进行修改

-在web.xml进行配置

## 介绍常用常量



1. 表单提交数据到action里面，在action可以获取表单数据
2. 表单提交数据有中文，有乱码问题，解决：

-post提交直接设置编码

-get提交做编码转换

1. 如果在action里面获取表单通过post方式提交中文，中文乱码问题已经解决，不需要自己处理

# 分模块开发

1. 单独写配置文件，把配置文件引入到核心配置文件中



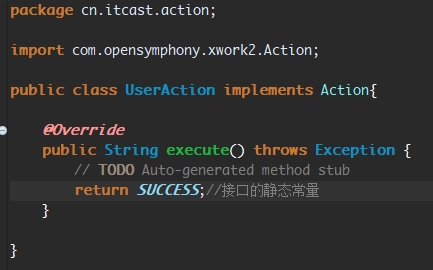
hello.xml文件就是原来Struts.xml文件的内容

路径如果是在src目录下，直接hello.xml就可以，否则加全路径

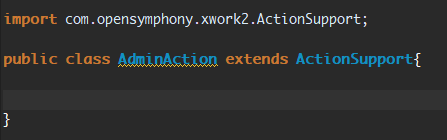
# Action三种编写方式

（表单提交**有很多种**，但是常用的是get/post）

1. 创建普通类，不继承任何类，也不实现任何接口（自己写execute方法）
2. 创建类，实现Action接口



1. 创建类，继承ActionSupport类





# 访问action里面的其他方法（重点！！！）

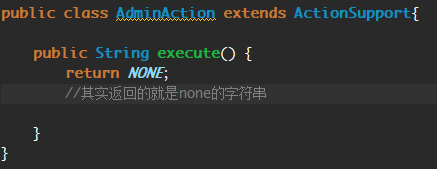
## 1.有三种实现方式

1. 使用action标签中的method属性，在这个属性里面写要执行的action的方法
2. 使用通配符方式实现
3. 动态访问方式实现（少用）

## 错误演示

1. result标签里的属性值和execute的返回值不同
2. Execute有返回值的话必须是String
3. execute方法可以没有返回值，没有返回值时，在result标签中不需要配置

-返回值类型写void或让返回值返回none（建议返回none）

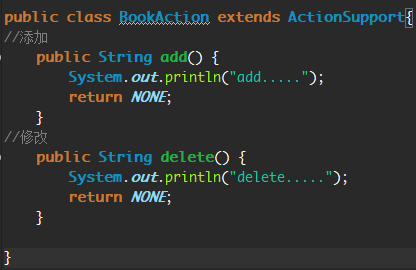


**PS:（这样配置的话，还是会跳转到hello.jsp）**

**（如果不配置result标签，把这行删掉，返回的是空白页）**

## 使用action标签method属性

1. 创建方法



1. 使用Method配置



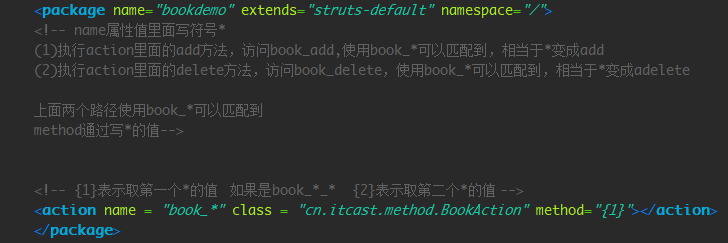
缺陷：action中的每个方法都需要配置，如果很多方法的话，会很麻烦（适用于方法较少的action）

## 使用通配符（重点！！！使用最多）

1. 在action属性标签里有name属性，name属性里面写\*星号
2. \*理解，表示匹配任意内容

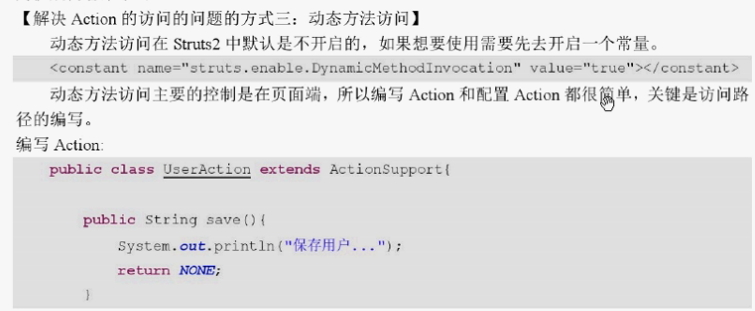
-比如访问hello, \*可以匹配到

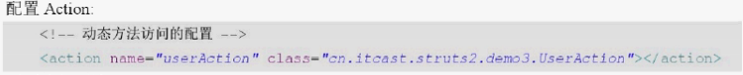
-访问delete, \*可以匹配到





## 动态访问（知道即可，不用）

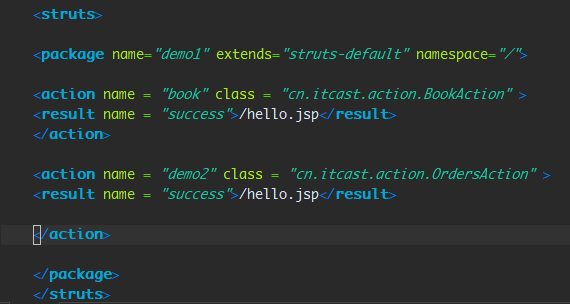






# 结果页面配置

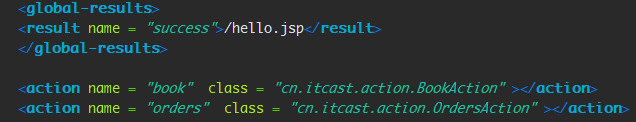
1. result标签配置方法返回值，到不同的路径
2. 创建两个action,执行默认的execute方法，让这两个action的方法都返回同一个值，配置到同一个页面



**（原始方法）**

## 全局结果页面配置

如果多个action，方法里面返回值相同，到的页面也相同，这时候可以用全局页面结果配置



写在package标签里面

## 局部结果页面配置

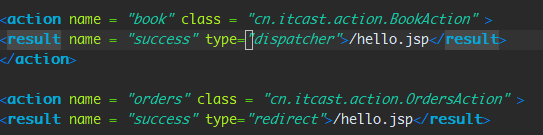


局部和全局都配置的话，以局部配置为准

## result标签中的type属性

1.在result标签里面，除name属性外，还有type属性

2.type属性：配置如何到路径去（转发or重定向）



1. 默认值做转发操作，dispatcher



1. 重定向操作，redirect

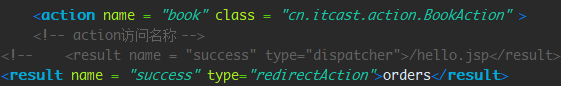


（ctrl+f5，清除缓存刷新）

1. 上面两个值dispatcher、redirect，这两个值一般针对到页面中配置，配置到其他的action里面

-chain：转发到action，一般不用，解决缓存问题（相当于转发）

-redirectAction：重定向到action



# 在action里获取表单提交的数据

1. 之前web阶段，表单提交到servlet，在servlet里面使用request对象里面的方法，getParameter,getParameterMap
2. 表单提交到action，但是没有request对象，不能直接使用
3. action中获取表单提交的数据，主要是有**三种方式**
4. **使用ActionContext类**
5. **使用ServletActionContext类**
6. **使用接口注入方式（一般不用）**

## **使用ActionContext类获取**



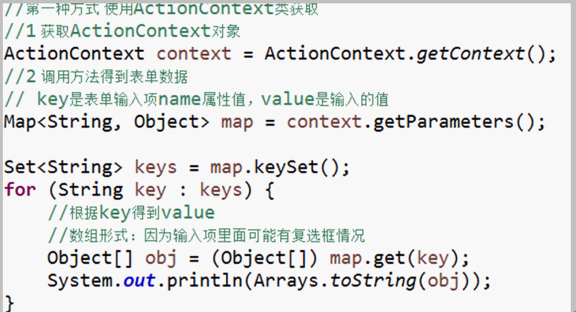
1. 不是静态方法，要创建类使用
2. 这个ActionContext类对象不是New出来的



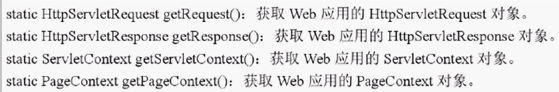
1.具体演示

（1）创建表单，提交表单到action

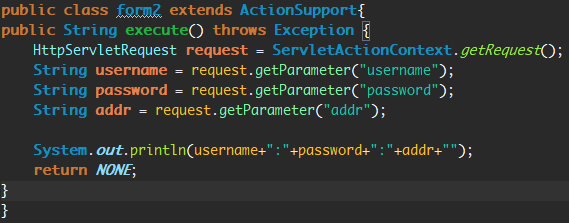
（2）在action中使用ActionContext来获取表单数据



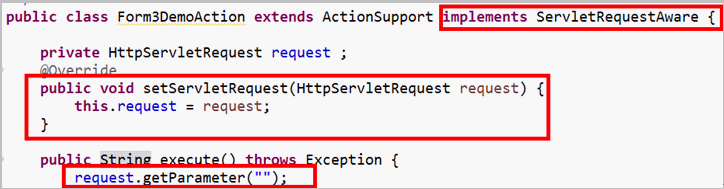
## 通过ServletActionContext类获取

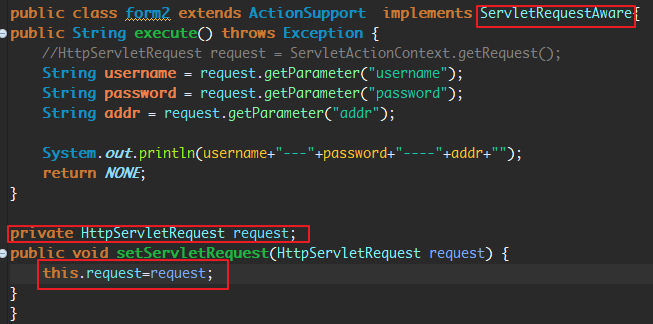


1. 调用类里面的静态方法，得到request对象



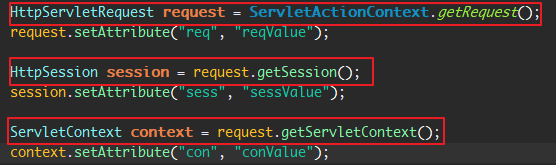
## 使用接口注入





# 在action操作域对象

1. request、session、servletcontext（作用范围由小到大）
2. 使用ServletActionContext类操作域对象



# Struts2封装获取表单数据方式

## 原始方式获取表单封装到实体类对象

（实体用get.set方法）

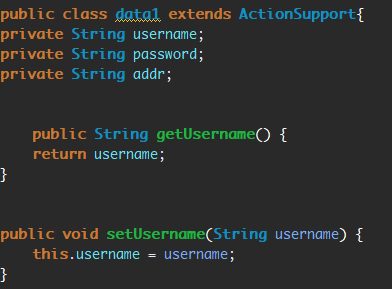


## 属性封装

1. 直接把表达提交属性封装到action的属性里面
2. 实现步骤
3. 在action成员变量位置定义变量

**-变量名称和表单输入项的name属性值一样**

1. 生成变量的set和get方法



使用属性封装获取表单数据到属性里面，并不能封装到实体类对象里面，还是要new对象之后set

## 模型驱动封装（重点）

1.使用模型驱动，可以把表单数据直接封装到实体类对象

2.实现步骤：

（1）action实现ModelDriven接口



1. 实现接口里面的getModel方法 -把创建对象返回

（3）在action里面创建实体类对象



1. 使用模型驱动和属性封装注意问题：

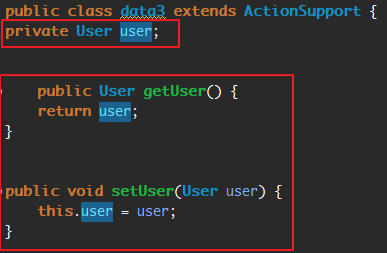
同一个表单同时使用这两种方法的话只会执行模型驱动，如果是不同的表单则分开执行

## 表达式封装（会用）（也可以归类到属性封装里）

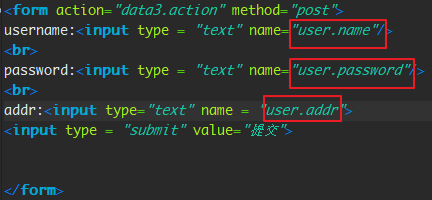
1. 实现过程
2. 使用表达式也可以将表单数据封装到实体类对象里面

第一步 在action里面**声明**实体类

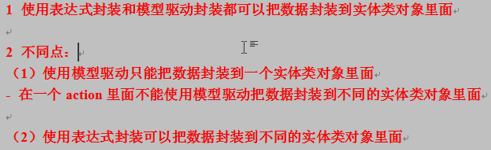
第二步 生成**实体类**的set和get方法



第三步 在表单输入项的name属性值里写表达式



## 比较表达式封装和模型驱动封装

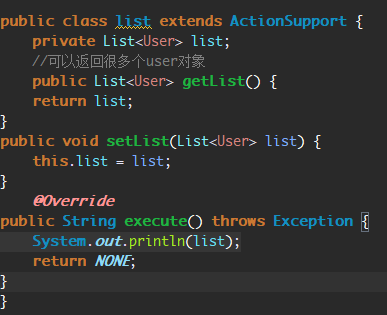


# 表单数据封装到集合

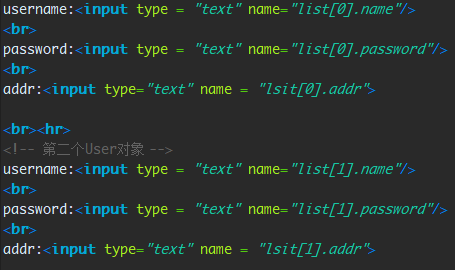
## 封装到list集合

第一步：在action里面声明List类

第二步：生成变量的get和set方法



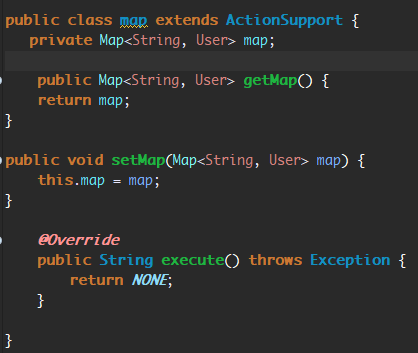
第三步：在表单输入项里面输入表达式



## 封装到map集合

第一步：在action里面声明List类

第二步：生成变量的get和set方法



第三步：在表单输入项里面输入表达式



# OGNL概述

1. 之前web阶段，学习过EL表达式，EL表达式在jsp中获取域对象里面的值
2. OGNL是一种表达式，这个表达式的功能更强大
3. 在struts2里面操作值栈数据
4. 一般把ognl在struts2操作，和Srtuts2标签一起食用操作值栈
5. OGNL不是struts2的一部分，是单独的项目，经常和struts2一起食用

（1）使用ognl时候先导入jar包，struts2提供Jar包

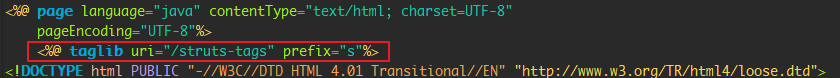
# OGNL入门案列

1. 使用ognl和struts2实现计算字符串长度

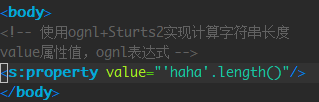


1. 在java代码中调用字符串.length();
2. 使用struts2标签
3. 使用jtsl时，导入jar包之外，在jsp页面引入标签库

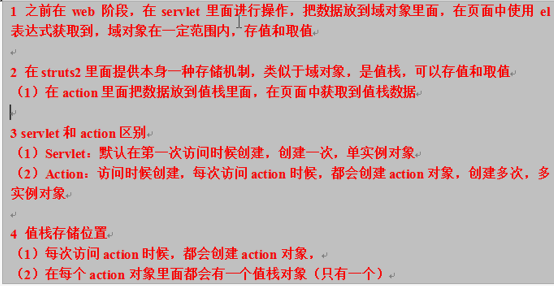
使用struts2时，在jsp页面引入标签库



1. 使用struts2表情实现操作



# 什么是值栈



# 获取值栈对象



# 值栈内部结构

1. 值栈分为两部分

第一部分：root,结构是list集合



第二部分：context,结构map集合

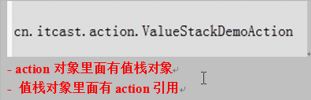




1. struts2里面标签s:debug，使用这个标签可以查看值栈结构和存储值



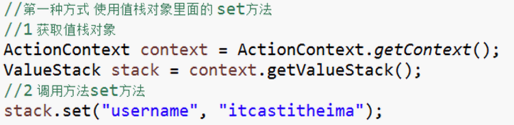




# 向值栈放数据

1. 向值栈放数据的多种方式

第一种 获取值栈对象，调用值栈对象里面的Set方法





第二种 获取值栈对象，调用值栈对象里面的push方法





第三种（常用） 在action类定义变量，生成变量的get方法



# 向值栈放对象

实现步骤：

第一步 定义对象变量

第二步 生成变量Get方法

第三步 在执行方法里设置对象值



# 向值栈放List集合

实现步骤：

第一步 定义List集合变量

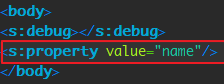
第二步 生成变量Get方法

第三步 在执行方法里设置对象值或添加内容

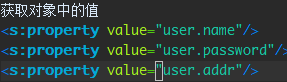


# 从值栈获取数据

## 获取字符串

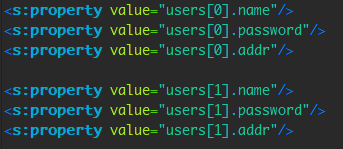
value=变量名称

## 获取对象



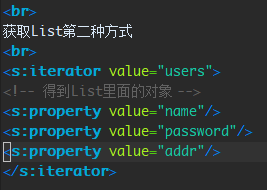
## 获取List集合

第一种方法

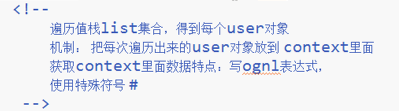


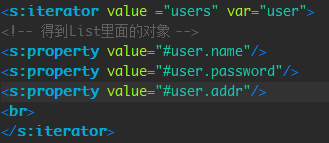
第二种方法





第三种方法





# 其他操作

1. 使用set方法向值栈存数据，取数据



1. 使用push方法向值栈存数据，取数据
2. 使用Push方法设置值，只有数据，没有名称
3. 向值栈放数据，把数据存到数组里面，通过数组获取数据



# ONGL的#、%使用

## #使用

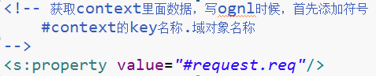
1.使用#获取Context里面的数据



1. 演示操作

先获取request域







## %使用



# Struts2拦截器概述



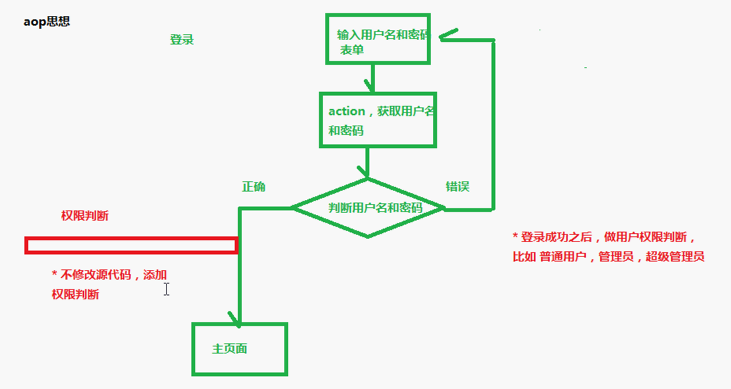


# 拦截器底层原理

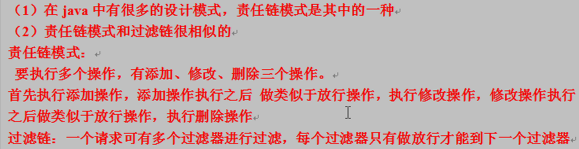
## AOP思想

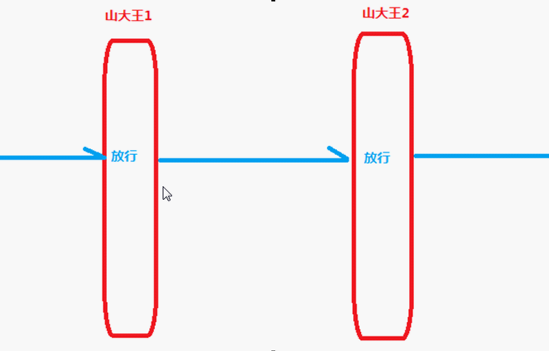


1. 画图演示

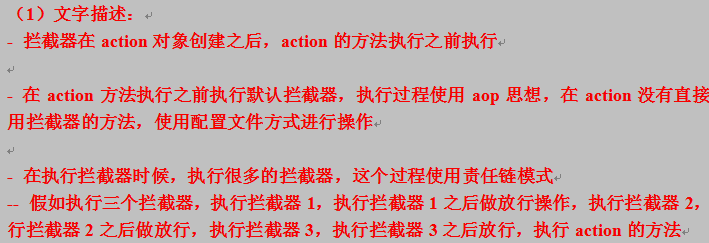


## 责任链模式

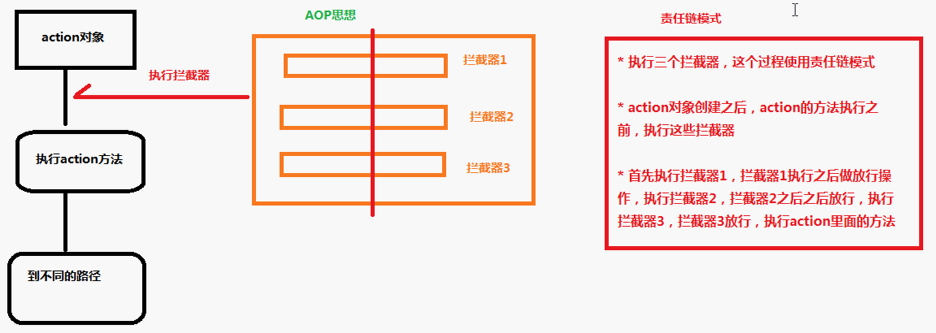




## AOP思想和责任链模式如何应用到拦截器

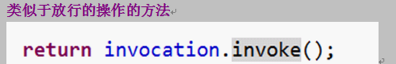


（2）画图分析



1. 查看源代码





# 重要的概念

## 过滤器和拦截器区别



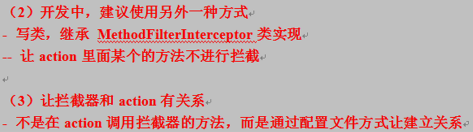
## Servlet和action区别



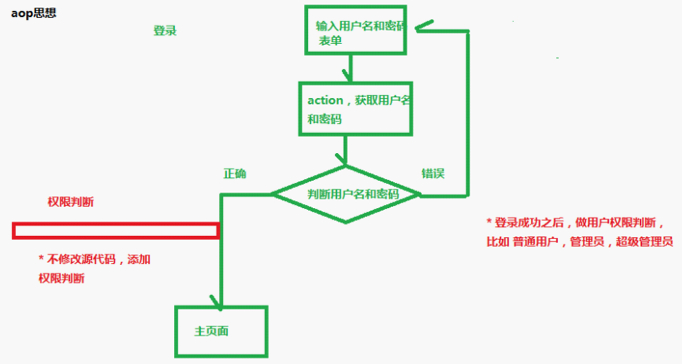
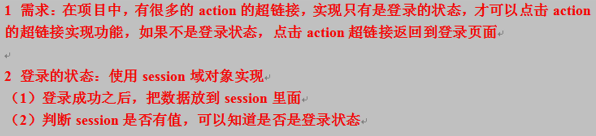
# 自定义拦截器



**实际开发中，建议使用这种方式**



# 自定义登录拦截器

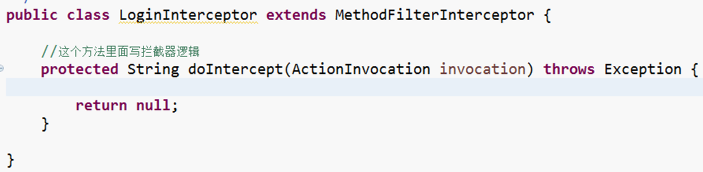








第一步：创建类，继承MethodFilterInterceptor类



第二步：重写MethodFilterInterceptor类里面的方法写拦截器逻辑

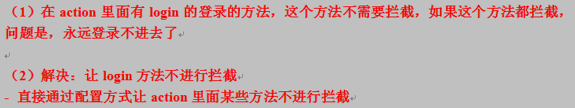


第三步：配置action和拦截器关系（注册拦截器）

↑如果使用了自定义拦截器，默认的拦截器会失效，想用的话要手动使用

PS：配置拦截器，会对action里面的所有方法进行拦截

## 配置不拦截方法

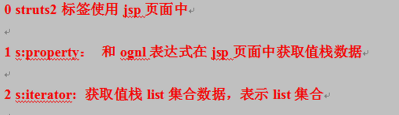




**一点小问题：**

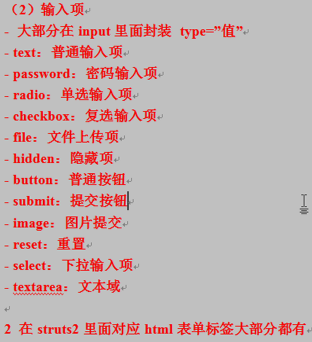


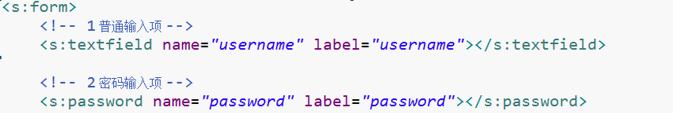
# Struts2的标签库

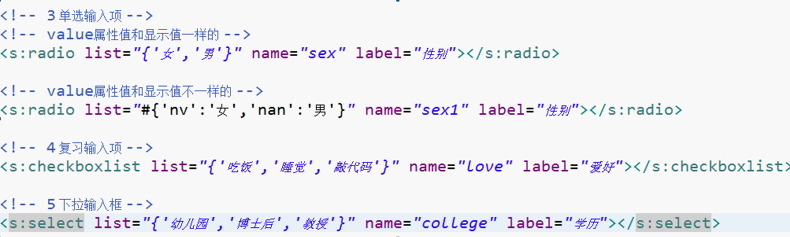


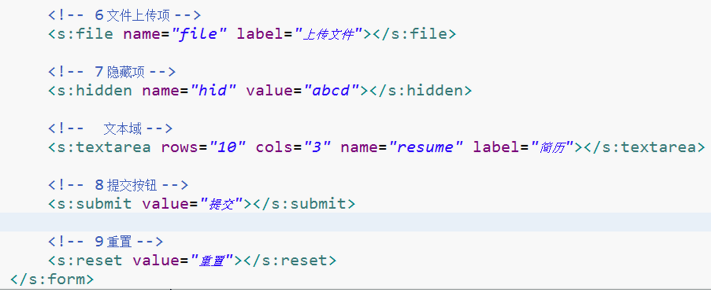


# Struts2表单标签









# 