# Day32 log4j 与Struts2

## log4j 日志管理的框架

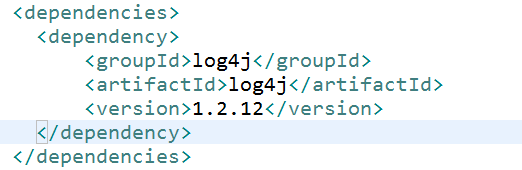
### 出现原因

项目上线之后,项目中所有的日志语句都应该被取消掉;

项目运行过程中为了方便收集错误日志信息.

### 用法

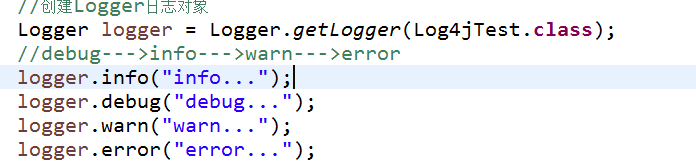
#### .添加log4j的jar包依赖:



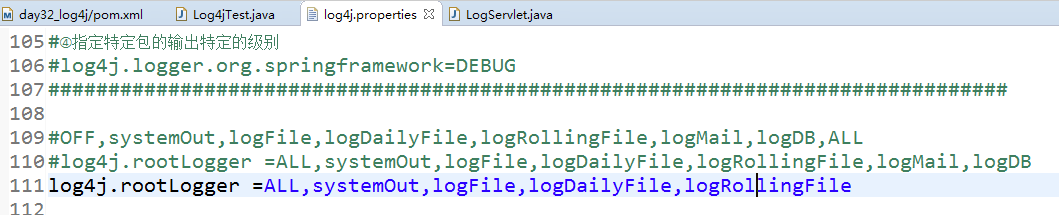
#### .进行log4j的配置,编写log4j.properties文件



#### . 具体使用

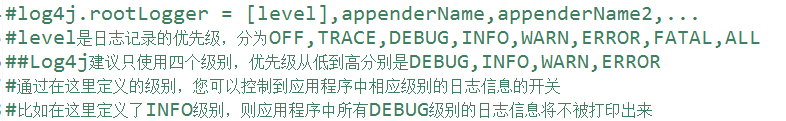


如果要关闭日志在控制台的输出功能:



只需要将log4j.rootLogger后面的systemOut给去掉即可.

### log4j日志级别:



## Struts2

### 1.Struts2简介

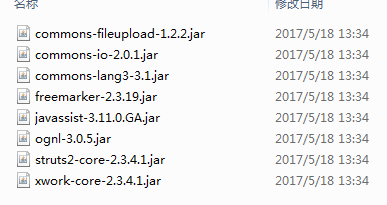
Struts2是一个基于MVC模式的流行的和成熟的Web框架,也就是说Struts2框架主要用来解决三层架构中web层功能,web层的相关功能基本上都可以用Strut2来实现.Struts2本质上就是Servlet.

Struts2框架从名字上看是Struts1的后续版本,其实两者之间关系不大.Struts2框架是与WebWork合作,更多的是参考了WebWork的xwork.jar功能.

Struts2内部是采用过滤器和拦截器来实现的,来处理用户的请求和响应.

## Struts2基本使用

### 1.相关依赖包



commons-XXX:引用通用的jar包.

commons-fileupload:实现文件上传;

commons-io:给文件上传功能进行依赖的.

commons-lang3:是对java语言的扩展包.

struts2-core:Struts2框架的核心包.

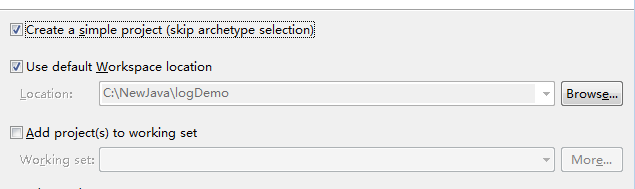
xwork-core:WebWork的核心包,也是整个Struts2中的核心和底层包.

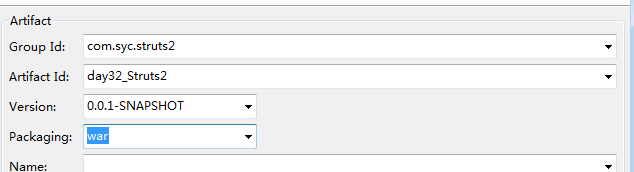
ognl:ognl是一种对象图像导航语言,主要进行数据的操作,类似于el表达式.

freemarker:负责视图的展现.

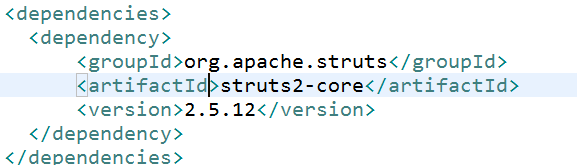
### 2.项目创建过程

①.创建Maven的Web项目

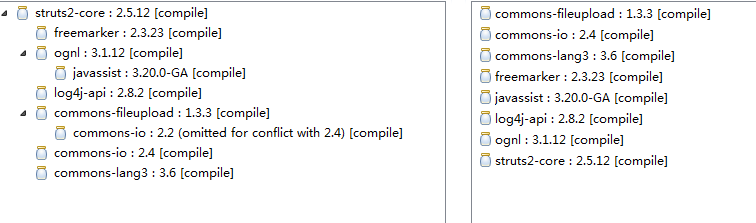




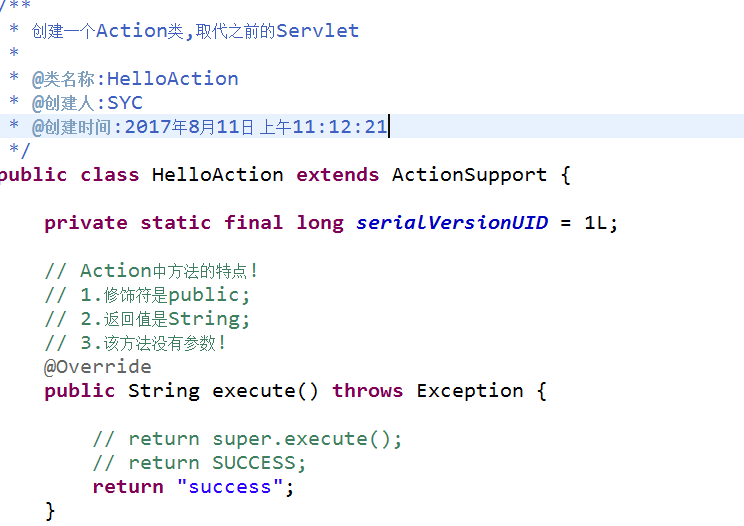
### 3.在maven中引入strut2-core核心包.



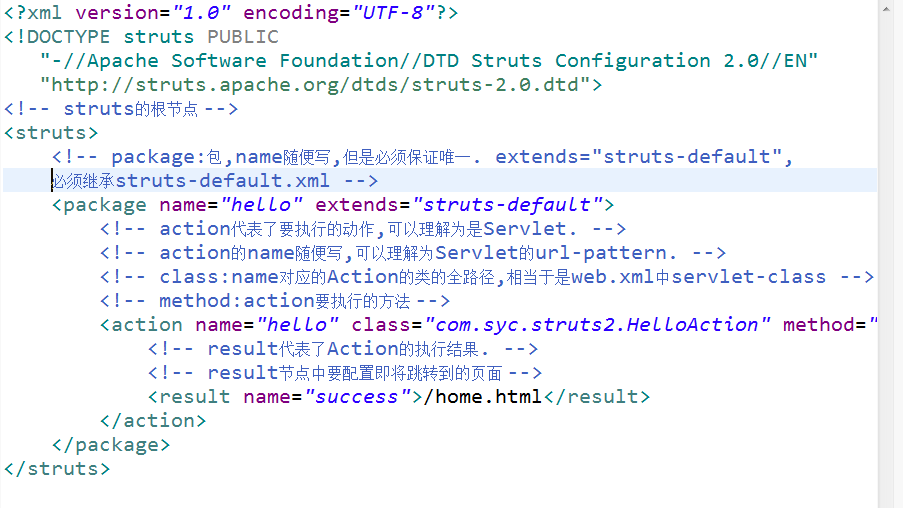
形成如下图所示的结构:

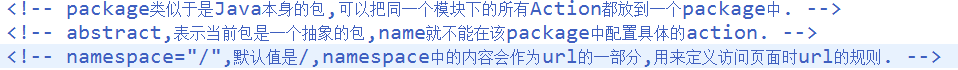


### 4.创建Action类



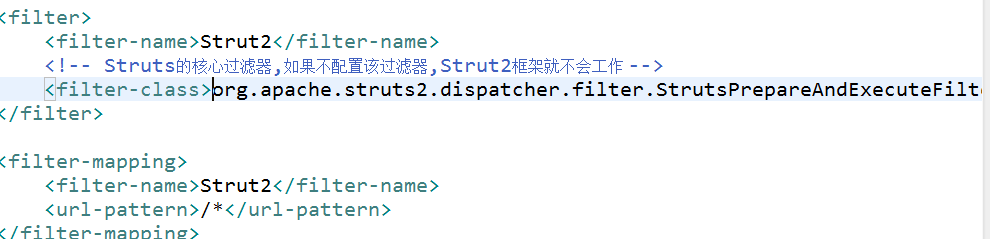
### 5.配置struts.xml文件



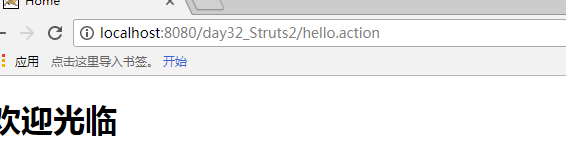


### 6.配置web.xml

配置Struts2的核心过滤器



### 7.在浏览器中进行访问:



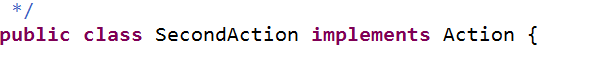
注意:struts2框架action的默认后缀是.action

## 创建Action的方式

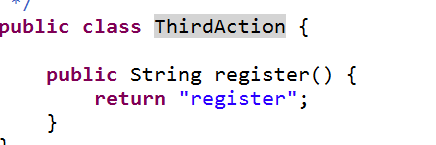
### 1.第一种方式:继承ActionSupport



### 2.第二种方式:实现Action接口



### 3.第三种方式:直接创建一个普通类.

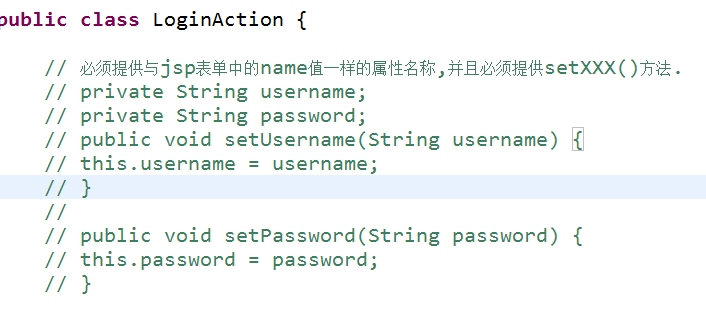


### 4.Action接口提供的常量:



## jsp与Action传值

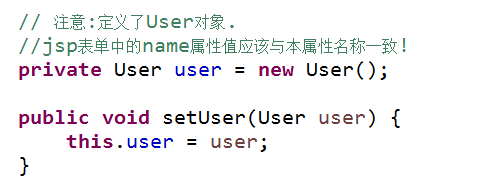
### 没有JavaBean的情况



在Action类中,定义某个属性,并且给该属性提供对应的setXXX()方法.

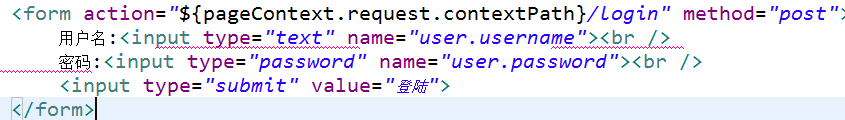
在jsp中,表单中的标签的name的值必须与Action中属性的名称一致!

### 有JavaBean的情况:



在Action类中,定义出来该Bean的一个对象,并且给该对象提供对应的setXXX()方法.

jsp中此时写法:

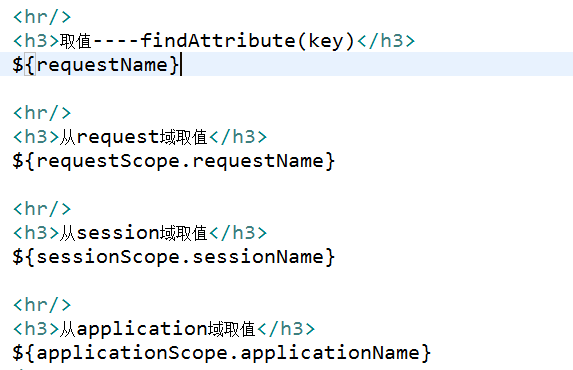


### Action给JSP传值

在Action中,将信息存储到某个域中.



在JSP中取出域中存储的信息:



## Struts2的执行流程与工作原理(重点)

### 用户打开浏览器,发来了一个HttpServletRequest请求;

2.request请求会先交给ActionMapper类, ActionMapper会对request进行简单的封装处理,处理的结果就能决定这个request是否能够执行下去;

### 3.当服务器中的项目接收到request请求后,

FilterDispather(就是StrutsPrepareAndExecuteFilter)是整个Struts2的核心调度中心,也是整个MVC中的C.它会根据ActionMapper的结果来决定是否处理请求;

1. FilterDispatcher对象会把ActionMapper传递给ActionProxy,创建一个ActionProxy代理对象.ActionProxy此时会取出来request中的url.当ActionProxy拿到这个url地址之后,它会与struts.xml中的action的name进行对比.ActionProxy类之所以能够实现url的对比,就是因为在此时ActionProxy类中还加载了另一个类ConfigurationManager类, ConfigurationManager类负责从struts.xml中进行解析,读取action的配置.
2. 如果在第4步中,ActionProxy的获取的url与struts.xml的url是想匹配的,此时会创建出ActionInvocation类对象. ActionInvocation类对象里面描述了Action的整个运行过程.
3. 在ActionInvocation执行完毕之后,就开始执行execute()方法,紧接着就会执行一些列的拦截器.
4. 经过层层的拦截器之后,就到了我们自己写的Action类.系统会根据Action类的结果,封装成不同的Result对象.
5. 系统会根据不同的Result,跳转到不同的页面视图Template.
6. Template页面视图经过拦截器再一次拦截过滤之后,最终生成Response响应结果.浏览器就会拿到该response结果.

在浏览器进行展现.