# 更新

7月12日重新排整：更新了拦截器，增加了其它，修改了很多

# Struts2 概述

Struts2要掌握的有：

1、Action类的操作：获取页面的数据、数据往页面传递数据

2、值栈和ognl：valueStack接口，实现类OhnlValueStack对象，贯穿整个action的生命周期，一个action对应一个valueStack对象。

3、拦截器和struts2标签库：默认调用很多拦截器帮你做了很多功能：获取页面封装属性param拦截器类，类型转换等等

Struts2版本：老师建议使用小版本最稳定的：struts-2.3.24-all.zip

# Struts2框架入门

第一步 导入jar包

struts-2.3.24-all.zip 解压

在lib中有jar包，不能把这些jar都导入到项目中，这里有100多个，有些是spring整合包

到apps目录里面，找到示例程序，从示例程序复制jar包：struts2-blank.war打开把lib中的全部复制即可

commons-fileupload-1.3.1.jar：Struts文件的上传和下载

commons-io-2.2.jar：文件读取

commons-lang3-3.1.jar：为java.lang包提供扩展

freemarker-2.3.19.jar：FreeMarker是一个模板引擎，一个基于模板生成文本输出的通用工具

ognl-3.0.6.jar：支持ognl表达式

javassist-3.11.0.GA.jar：分析、编辑和创建JAVA字节码的类库

struts2-core-2.3.16.3.jar:Struts2的核心包

xwork-core-2.3.16.3.jar:xwork的核心包

asm-\*.jar 类似javassits

log4j-1.2.17.jar 日志类库

第二步 web中的配置过滤器：不用配置核心配置struts2路径位置

<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>

第三步 创建struts2核心配置文件，两个必须（名称 struts.xml，位置 类根路径下）

<!DOCTYPE struts PUBLIC

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

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

<struts>

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

<!-- name:url访问的名称 不写method属性就是访问默认的execute() -->

<action name="halle" class="com.wwl.action.halleAction" method=””>

<result name="ok">/a.jsp</result><!-- 配置方法的返回值：转发和重定向等-->

</action>

</package>

</struts>

第四步 写个普通类（①继承actionContext类②普通类③实现action接口）

setvlet操作：都会执行service方法，①继承HTTPServlet重写类里面的方法②web.xml中配置setvlet

status2的action操作：①在status.xml进行配置②每次访问action时候默认执行名称execute方法

public class HAction {

public String execute() { //在第三步中action标签中属性method没有写：就直接访问此方法

ActionContext context = ActionContext.getContext();

ValueStack stack = context.getValueStack(); 获取值栈对象

//现在user在栈顶的位置 同名取值ROOT中栈顶的

第一种： 调用方法set方法 等同于request存储属性一样map结构，看下面调用

Stack.set(”username”,”温文亮”);

//第二种 push：对象、基本类型都可以，

stack.push(user); 网页调用：你可以查看结构：栈中都是放入对象里的属性，看下面调用

return "ok";}

页面调用①el表达式②struts标签

<%@taglib uri=”/struts-tags” prefix=”s”%>

第一种 set 压栈到root顶上，查看debug看到root栈中分配：java.util.HashMap看不到key和值

<s:property value=” userName”/>

第二种 push 如果是对象，进行压到root栈顶上：但是它不是对象名称：是对象里的所有属性列出来能看到值

<s:property value=” userName”/>

# Action类操作

第一步 创建action类

第一种 创建普通类，继承类 ActionSupport（常用）

ActionSupport实现了第三种Action类，可以不用强制要写execute()方法

public class ExtendsAction extends ActionSupport{

第二种 创建普通类、这个类不继承任何类，不实现任何接口

public class A(){}

第三种 创建普通类：实现接口Action（几乎不用，此接口都是常量，需要实现默认方法）

public class InfacterAction implements Action{

public String execute() throws Exception { //必须实现此方法

return SUCCESS; }

第二步 类中获取页面form表单数据操作

第一种模型驱动封装获取页面数据 可以用匿名很方便（最常用的）

public class ExtendsAction implements ModelDriven<User>{ 第一步action类实现接口 ModelDriven对象泛型

private User user = new User(); //第二步 创建对象，前提要求，页面的name要和实体类属性名一致

public User getModel() { //第三步 实现接口里面的方法 getModel方法，把创建对象返回

return user;

}

使用模型驱动和属性封装注意问题：

在一个action中，获取表单数据可以属性封装，使用模型驱动封装，

不能同时使用属性封装和模型驱动封装获取同一个表单数据

如果同时使用，只会执行模型驱动

第二种 Struts2获取数据：通过设置成员变量设置set方法，

**成员属性中获取数据** 两步即可然后调用

（1）在action类成员变量位置定义变量名称：变量名称和页面表单的name属性值一样

（2）生成变量的set方法 注意页面是找set方法中的名称的，不是变量

原理：内省技术：具体百度，BeanUtils框架也是通过此方法的：所有请求经过滤器，过滤器中会调用默认很多拦截器

**实体类属性中获取数据（重点）**

Action类操作

Private User user;

Public void setUser(User user){

This.user = user;

页面

<input type=”text” name=”user.userName”> //表达式

**List集合封装**

**类中：List<user> list;**

**页面：**<input type=”text” name=”list[0].name”>

**Map集合封装**

**类中：**Map<String User> map;

**页面：**<input type=”text” name=”map[‘one’].name”>获取user对象第一个数据

一般配合循环取值

第三种 原始获取方式：ServletActionContext类可以获取request、response、application

Static httpServletRequest getRequest():获取Web应用的HttpServletRequest对象

Static HttpServletResponse getResponse():获取Web应用的HttpServletResponse对象

Static ServletContext getServletContext():获取Web应用的ServletContext对象

Static PageContext getPageContext():获取Web应用的PageContext对象

第四种 ActionContext类获取 map

Map<String,Object>getParamerers() 返回一个包含所有HttpServletRequest参数信息

需要创建ActionContext类的对象：ActionContext.getContext();

具体步骤：

public String add() {

//获取ActionContext对象

ActionContext ac=ActionContext.getContext();

//调用方法得到表单数据

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

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

for (String str : set) {

//数组形式；因为输入项里面可能有复选框情况

Object[] obje=(Object[]) map.get(str);

}

return NONE;

}

第三步 发送数据到页面

第一步 定义属性和页面相同的表单name中的值

第二步 属性封装get()方法 这两步即可会自动把值放到值栈中（看值栈）

第四步 页面获取数据：①el表达式②struts标签配合ognl

先把数据放入值栈，页面再从值栈中获取数据

# 值栈和OGNL

查看结构 s:debug标签为两部分：root（值栈）servlet（原始的）

一、 root ：CompoundRoot（实现类）继承的是ArrayList结构是：list集合

这里维护的都是自己定义的action类(值栈对象)

获取值栈对象 常用方式：使用ActionContext类里面的方法得到值栈对象

第一步先获取ActionContext类对象

ActionContext context= ActionContext.getContext();

第二步调用方法得到值栈对象

ValueStack stack = context.getValueStack();

①每次访问action时候，都会创建action对象②在每个action对象里面都会有一个值栈对象（只有一个）

老师在action类的方法中：创建了几个值栈对象进行对比了都是同一个对象：值栈的特点：先进后出（后进先出）

什么是值栈？

ValueStack是struts的一个接口，OgnlValueStack是实现类，客户端发起一个请求Struts2会创建action实例同时创建OgnlValueStack值栈实例，它贯穿整个Action的生命周期

二、context ：OgnlContext（实现类）实现了Map接口，结构：map集合

此类中的key都是固定的：key：request、response、application、parameters、attr

获得context中的request对象：ServletActionContext.getRequest()，它就是HTTPServletRequest对象

**使用attr获取域对象**

servlet和action区别

（1）Servlet：默认在第一次访问时候创建，创建一次，单实例对象

（2）Action：访问时候创建，每次访问action时候，都会创建action对象，创建多次，多实例对象

向root值栈的数据存取操作：

第一种 获取值栈对象，调用值栈对象里面的set方法（重要）

User user= new User();

ActionContext context = ActionContext.getContext();

ValueStack stack = context.getValueStack(); 获取值栈对象

Stack.set(”n”,”温文亮”); set会压栈到root顶上，查看debug看到root栈中分配：java.util.HashMap值为null，看不到key和值

Stack.push(user); 调用push方法 如果是对象，也会压到root栈顶上：会把对象中的属性全部列出来：属性名字和值

第二种 在action类中定义变量、生成变量的get方法

/默认情况下是已经将action对象压入到值栈，所以属性也在值栈中

Private User user； 这里可以集合，单纯属性，都可以的

Public User getUser(return user) //生成get方法

Public String execute（）{ //在执行的方法里面项对象设置值

user =new User();

user.setName(”你好”)

return ”success”;}

ActionContext context = ActionContext.getContext();

ValueStack stack = context.getValueStack();

此方式在值栈中存储的方式是String

页面获取数据

第一种：对应上面:第一种存储值栈中的方式

使用struts2的标签+ognl表达式获取值栈数据: <s:property value=”ognl表达式”/>

<%@taglib uri=”/struts-tags” prefix=”s”%>

set方法

<s:property value=” n”/> //配合struts标签直接名字即可

push方法 对象中的属性全部列出来：属性名字和值

<s:property value=” userName”/> //userName是user对象中的其中一个属性名字

获取list集合对象 set方法

一 通过下标取值

<s:property value=”list[0].username”/>

二 action类中有成员变量名称为”list”的集合对象

<s:iterator value=”list”>

<!--遍历list得到list里面每个User对象 --!>

<s:property value=” username”/>

<s:property value=”password”/>

</s:iterator>

三 方式和第二种方式的遍历标签一样，多了var

<!--遍历list值栈list集合，得到每个User对象

机制：把每次遍历出来的user对象放到context里面获取context里面数据特点：写ognl表达式，使用特殊符号#，这个#都是用来取值context值栈中的 --!>

<s:iterator value=”list” var=”user”>

<s:property value=” #user.username”/>

<s:property value=” #user.password”/>

</s:iterator>

向Context中存取数据：就是servlet原始操作了

获取对象 然后就是setattribute了，这里不写了

Static httpServletRequest getRequest():获取Web应用的HttpServletRequest对象

Static HttpServletResponse getResponse():获取Web应用的HttpServletResponse对象

Static ServletContext getServletContext():获取Web应用的ServletContext对象

Static PageContext getPageContext():获取Web应用的PageContext对象

页面取数据使用struts标签方式： el标签当然是可以得了

<s:property value=” #request.name”/> //获取方式 #号加context固定对象进行获取

## 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，这个表达式功能更加强大，

（1）在struts2里面操作值栈数据

（2）一般把ognl在struts2操作：和struts2标签一起使用操作值栈

3 OGNL不是struts2的一部分，单独的项目，经常和struts2一起使用

（1）使用ognl时候首先导入jar包，struts2提供jar包

### OGNL入门案例

导包ognl-3.0.6.jsr javassist-3.11.0.GA.jar

调用普通方法

OgnlContext context =newOgnlContext(); //获得context

Object root=context.getRoot() //获得根对象

Object o=Ognl。getValue(”helloworld”,context,root); //执行表达式

System.out.println(obj)

访问对象的静态方法

OgnlContext context =newOgnlContext(); //获得context

Object root=context.getRoot() //获得根对象

Ognl.getValue(”@java.lang.Math@random()”,cibtext,root) // @[类全名（包括包路径）]@[方法名 | 值名]

OGNL的#、%使用

#使用 （1）向request域放值 （2）在页面中使用ognl获取

<s:property value="#request.name"/> //获取request中的name值

% 强制使用OGNL表达式，struts有些不使用OGNL的如文本框

在struts2标签里面使用ognl表达式，如果直接在struts2表单标签里面使用ognl表达式不识别，只有%之后才会识别。

<input type="text" name="name" value="<s:property value='#request.name'/>"/> 这样可以的

这是struts2标签：标签不能嵌套标签的 使用#request.name 会原样显示在页面- 使用标签会报错

<s:textfield name="name" value="%{#request.name}"/>

% 另一个用处 不使用OGNL表达式

# 拦截器

概述 只能拦截action类：在action类创建对象之后，类中的方法执行之前执行此拦截器

1 struts2是框架，封装了很多的功能，struts2里面封装的功能都是在拦截器里面

2 struts2框架核心功能都是依赖拦截器实现的，每次执行默认的拦截器

3 struts2里面默认拦截器位置：核心文件core中的struts-defalt.xml，打开文件最下面：

标签：<interceptor-stack name=”defaultStack”>里面都是默认拦截器

<interceptor-stack name="defaultStack">

<interceptor-ref name="exception"/>

<interceptor-ref name="alias"/>

<interceptor-ref name="servletConfig"/>

<interceptor-ref name="i18n"/>

<interceptor-ref name="prepare"/>

<interceptor-ref name="chain"/>

<interceptor-ref name="scopedModelDriven"/>

.............

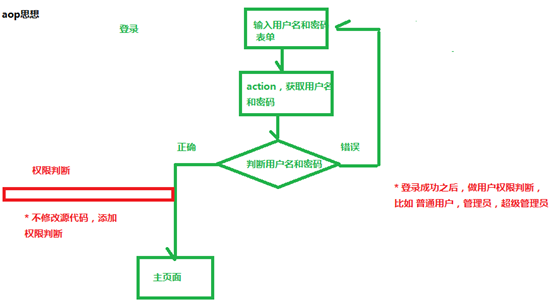
</interceptor-stack>

拦截器底层原理

拦截器底层使用两个原理

第一个 aop思想：跟spring一样

Aop是面向切面（方面）编程，有基本功能，扩展功能，不通过修改源代码方式扩展功能



第二个 责任链模式

（1）在java中有很多的设计模式，责任链模式是其中的一种

（2）责任链模式和过滤链很相似的

责任链模式：

要执行多个操作，有添加、修改、删除三个操作。

首先执行添加操作，添加操作执行之后 做类似于放行操作，执行修改操作，修改操作执行之后做类似于放行操作，执行删除操作

过滤链：一个请求可有多个过滤器进行过滤，每个过滤器只有做放行才能到下一个过滤器

查看源代码

打开web中配置的核心类：org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter

（1）执行action



（2）创建action对象，使用动态代理方式



（3）执行action的方法



（4）执行很多的拦截器，遍历执行



类似于放行的操作的方法



重要的概念

|  |
| --- |
| 1 过滤器和拦截器区别  （1）过滤器：过滤器理论上可以任意内容，比如html、jsp、servlet、图片路径  （2）拦截器：拦截器只可以拦截action  2 Servlet和action区别  （1）servlet默认第一次访问时候创建，创建一次，单实例对象  （2）action每次访问时候创建，创建多次，多实例对象 |

## 自定义拦截器

|  |
| --- |
| 在struts2里面有很多的拦截器，这些拦截器是struts2封装的功能，但是在实际开发中，struts2里面的拦截器中可以没有要使用的功能，这个时候需要自己写拦截器实现功能  简单步骤 写类继承MethodFilterInterceptor 核心配置文件配置struts  第一步 继承MethodFilterInterceptor 抽象类  public class LoginInterceptor extends MethodFilterInterceptor {  //实现抽象类中的方法 doIntercept  protected String doIntercept(ActionInvocation invocation) throws Exception {  // 判断session里面是否有名称是username的值  HttpServletRequest request = ServletActionContext.getRequest();  Object obj = request.getSession().getAttribute("username");  if(obj != null) {  return invocation.invoke();//做类似于放行操作，执行action的方法  } else {  return "login";//不执行action的方法，返回页面，result标签里面找到名称是login的值  }  }  第二步 核心配置文件struts的配置  配置局部拦截器：两步：package标签下定义拦截器，然后action下配置拦截器  <package name="user" namespace="/" extends="struts-default">  <!—第一 定义拦截器 -->  <interceptors>  <interceptor name="test" class="com.wwl.interceptor.InterceptorTest"/>  </interceptors>  <!—第二 在action标签内引入拦截器(一旦引入自定义拦截器，默认拦截器就不执行了，需要再手动引入) -->  <action name="users\_\*" class="userAction" method="{1}">  <result name="verific\_no">/register.jsp</result>  <interceptor-ref name="defaultStack"/> <!-- 手动引入默认拦截器 -->  <interceptor-ref name="test"/>  </action>  配置全局拦截器栈 package标签下定义拦截器 ，在定义拦截器标签内，引入拦截器  <!—第一 定义拦截器 -->  <interceptors>  <interceptor name="test" class="com.wwl.interceptor.InterceptorTest"/>  <!—第二 本标签内制作拦截器(一旦引入自定义拦截器，默认拦截器就不执行了，需要再手动引入) -->  <interceptor-stack name="myStack">  <interceptor-ref name="defaultStack"/><!-- 手动引入默认拦截器 -->  <interceptor-ref name="test"/> <!-- 引入自己定义的拦截器 -->  </interceptor-stack>  <!—第三 在action标签内引入拦截器-->  <action name="users\_\*" class="userAction" method="{1}">  <result name="verific\_no">/register.jsp</result>  <interceptor-ref name="myStack"/> <!—这里引入的是配置全局栈的名称了 -->  </action>  拦截器结构  查看源代码看拦截器结构 随便找一个struts-defalt.xml下:默认配置的拦截器类发现如下  <interceptor name="modelDriven"class="… ModelDrivenInterceptor"/>  ModelDrivenInterceptor继承了abstractInterceptor类它又实现了Interceptor  class ModelDrivenInterceptor extends AbstractInterceptor —— class AbstractInterceptor implements Interceptor  Interceptor接口 里面有三个方法  void init(); 初始化操作  void destroy() 销毁  String intercept(ActionInvocation invocation)：拦截逻辑的操作  开发中，建议使用继承MethodFilterInterceptor  - 写类继承 MethodFilterInterceptor类实现，此类是在xwork-core-2.3.24.jar中，它是继承了abstractInterceptor，只有一个抽象方法要实现，扩展了功能：通过配置文件的配置  就可以实现拦截效果  （如果继承或实现接口也可以的，但是老师说要进行反射自己写底层操作） |

# Struts2的标签库

|  |
| --- |
| struts2标签jsp页面中3个常用  1 s:property： 和ognl表达式在jsp页面中获取值栈数据  2 s:iterator: 获取值栈list集合数据，表示list集合  3 s:debug: 查看值栈结构和数据 |

Struts2表单标签（会用）

|  |
| --- |
| 1 html表单标签  form : action、method、enctype  输入项：大部分在input里面封装 type=”值”  - text：普通输入项- password：密码输入项- radio：单选输入项- checkbox：复选输入项  - file：文件上传项- hidden：隐藏项- button：普通按钮- submit：提交按钮  - image：图片提交- reset：重置- select：下拉输入项- textarea：文本域  2 在struts2里面对应html表单标签大部分都有 |

# Struts详细配置

1名称和位置是固定的

2 在配置文件中主要三个表package、action、result 标签里面的属性

## 标签package

1 类似于代码中的包，区分不同的action。要配置action，必须首先写package标签，在package里面才能配置action

2属性

（1）name属性

name属性值根功能本身没有关系的，

- 在一个配置文件中可以写多个package标签，name属性值不能相同的

（2）extends属性

- 属性值固定的，struts-default

- 写了这个属性之后，在package里面配置的类具有action功能

（3）namespace属性：不写也可以,默认值：/

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

## 标签action

1 action标签配置action访问路径

2 action标签属性

（1）name属性

- namespace属性值和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属性值不能相同的

（2）class属性

- action全路径

（3）method属性：指定action类中执行方法

- 如没有写此属性：action类里面默认就只执行execute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：不仅是页面，访问其它action也可以

注意：在类中配置没有返回值或者none的返回值是可以不用写此标签的

2 result标签属性

（1）name属性

- 和类中的方法返回值一样

（2）type属性

- 配置如何到路径中（转发或者重定向）

- type属性默认值做转发操作

（1）dispatcher默认值，做转发操作

（2）redirect做重定向操作

dispatcher、redirect，这两个值一般针对到页面中配置

（3）chain转发到action，一般不用，缓存问题

（4）redirectAction重定向到action

<result name="success" type="redirect"></result>

## 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）:核心文件下的org.apache.struts2下    3 修改struts2默认常量值  （1）常用的方式  - 在struts.xml中进行配置    （2）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  4 介绍最常用常量    （1）表单提交数据到action里面，在action可以获取表单提交数据，  （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  （3）如果在action获取表单通过post方式提交中文，中文乱码问题帮解决了，不需要自己处理问题 |

## 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

<!-- 引入a.xml文件此xml文件内容和strtus.xml一致 -->

<include file="com/wwl/a.xml"></include>

## 全局结果页面

作用：如两个类的返回值相同，返回最终页面都是相同的页面，就会有相同配置result标签（感觉做无用功，写了两个相同的），全局写了，就可以去掉result的标签了

要在package下写：如在别的package是不生效的哦

<global-results>

<result name="ok">/a.jsp</result>

</global-results>

即配置了全局又配置了局部，访问的肯定是局部了

## 访问action类三种方法

1有三种方式实现

第一种使用struts.xml中action标签的method属性，在这个属性里面写执行的action方法

此配置意思就是：浏览器http://localhost:8080/struts2Demo/halle访问add的方法，因此方法没有返回值（跳转页面）：所以action中没有result标签

<action name="halle" class="com.wwl.action.halleAction" method="add"></action>

缺陷：action每个方法都需要配置，如果action里面有多个方法，配置很多的action标签

第二种 使用通配符方式实现（重点）

在action标签里面name属性，name属性值里面写上“\*”号：表示匹配任意内容

1、name属性值里面写符号 \*

(1)执行action里面的add方法，访问 book\_add，使用book\_\*可以匹配到，\*相当于变成 add

(2)执行action里面的update方法，访问book\_update，使用book\_\*可以匹配到，\*相当于变成 update

上面两个路径使用 book\_\* 可以匹配到

2、method属性写值：取值{1}，如果有多个\*可以{2}......

<action name="book\_\*" class="cn.itcast.method.BookAction" method="{1}">

第三种 动态访问实现（几乎不用）

先要通过常量开启配置

Struts.xml就是普通的配置，不用写method属性

然后是在url上以！（action类中的方法名）.action形式操作

## 其它配置

打开开发模式 意思就是有：控制台提示信息，建议正式服：去掉

<constant name="struts.devMode" value="true"></constant>

修改struts不用老重启服务器

<constant name="struts.configuration.xml.reload" value="true"></constant>

编码配置

constant name="struts.i18n.encoding" value="默认就是UTF-8">

# 其它

标签问题

注意一旦定义了var是在context中了，就不在root值栈中了，需要加#号

构建了一个list集合 ognl认为是集合，这样context上有值

<s:iterator var="c" value="list”>

<s:property value="#c.userName"/> 有定义 var 需要加#获取

<s:property value="userName"/> 没有var的取值方式 直接写属性名

</s:iterator>

构建了map

<s:iterator var="i" value="#{'aa':'41','bb':'15','cc':'17'}">

<s:property value="key"/> <s:property value="value"/>

</s:iterator>

取得session 方法

第一种

ActionContext.getContext().getSession().put("user", user); //通过此方法放入session 返回值是map

取值：${sessionScope.user.username}

第二种

ServletActionContext获取request对象

EL表达式获取值栈数据原理

El作用：

1 EL表达式获取域对象值

2 向域对象里面放值使用setAttribute方法，获取值使用getAttribute方法

底层增强request对象里面的方法getAttribute方法

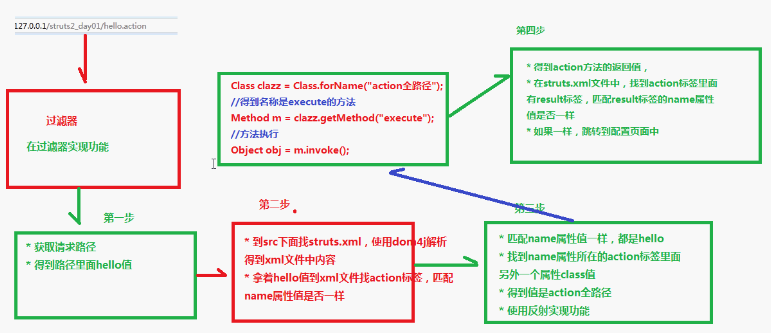
（1）首先从request域获取值，如果获取到，直接返回

（2）如果从request域获取不到值，到值栈中把值获取出来，把值放到域对象里面

源码中的代码



Strtus2执行过程



查看源码

StrutsPrepareAndExecuteFilter：此类是配置在web中的过滤器核心配置

public class StrutsPrepareAndExecuteFilter implements StrutsStatics, Filter

过滤器在服务器启动时候创建，创建过滤器时候执行init方法

包含自己创建的配置文件struts.xml和struts2自带配置文件

init\_FileManager();

init\_DefaultProperties(); // [1]

init\_TraditionalXmlConfigurations(); // [2]

init\_LegacyStrutsProperties(); // [3]

init\_CustomConfigurationProviders(); // [5]

init\_FilterInitParameters() ; // [6]

init\_AliasStandardObjects() ; // [7]

init\_DefaultProperties(); // [1]

加载org.apache.strus.default.properties配置的是struts2的所有常量

init\_TraditionalXmlConfigurations(); // [2]

加载struts-default.xml、struts-plugin.xml、struts.xm;

init\_LegacyStrutsProperties(); // [3]

加载用户自定义struts.properties

init\_CustomConfigurationProviders(); // [5]

加载用户配置的提供对象

init\_FilterInitParameters() ; // [6]

加载web.xml

init\_AliasStandardObjects() ; // [7]

加载标准对象

根据上面的代码我们可以得出配置文件的加载顺序如下

Default.properties

Struts-default.xml;

Struts-plugin.xml;

Struts.xml 配置Action以及常量。

Struts.properties 配置常量

Web.xml 配置核心过滤器及常量

前三个配置文件我们不用关心，是struts2内部的配置文件，我们无法修改，能修改的文件就是后三个，这几个配置文件的加载是有一定的顺序的，这三个配置文件都可以修改struts2的常量值，要记住的是，后加载配置文件中常量值会将先加载的配置文件常量的值给覆盖