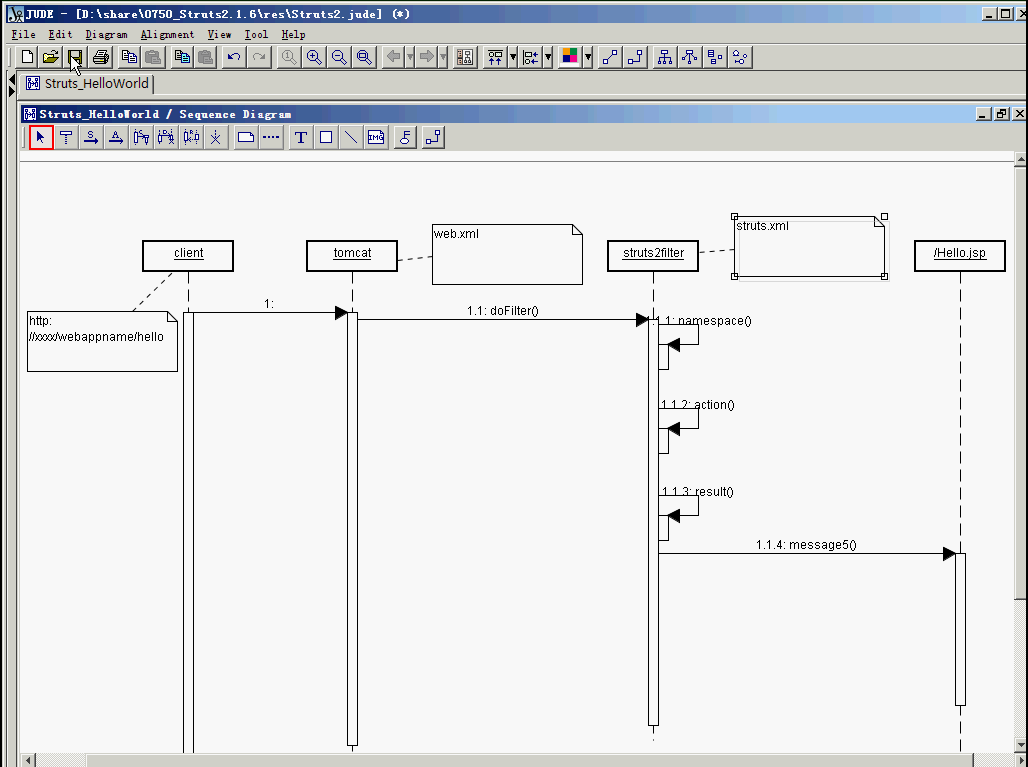
Hello world运行原理:



Namespace:

namespace决定了action的访问路径，默认为""，可以接收所有路径的action

namespace可以写为/，或者/xxx，或者/xxx/yyy，对应的action访问路径为/index.action，

/xxx/index.action，或者/xxx/yyy/index.action.

namespace最好也用模块来进行命名

action

struts1启动的时候就把action对应的类new出对象来，所以每次都用同一个对象； struts2就每次访问都new一个对象出来，所以每次都不同。

具体视图的返回可以由用户自己定义的Action来决定

具体的手段是根据返回的字符串找到对应的配置项，来决定视图的内容

具体Action的实现可以是一个普通的java类，里面有public String execute方法即可或者实现Action接口

不过最常用的是从ActionSupport继承，好处在于可以直接使用Struts2封装好的方法

Path:

struts2中的路径问题是根据action的路径而不是jsp路径来确定，所以尽量不要使用相对路径。

<a href="index.jsp">index.jsp</a>

虽然可以用redirect方式解决，但redirect方式并非必要。

解决办法非常简单，统一使用绝对路径。（在jsp中用request.getContextRoot方式来拿到webapp的路径）或者使用myeclipse经常用的，指定basePath

动态方法调用(DMI):

Action执行的时候并不一定要执行execute方法

可以在配置文件中配置Action的时候用method=来指定执行哪个方法（还会执行execute方法吗？我觉得不会）

也可以在url地址中动态指定（动态方法调用DMI）（推荐）

<a href="<%=context %>/user/userAdd">添加用户</a>

<br />

<a href="<%=context %>/user/user!add">添加用户</a>

<br />

前者会产生太多的action，所以不推荐使用

<action name="userAdd" class="com.bjsxt.struts2.user.action.UserAction" method="add">

<result>/user\_add\_success.jsp</result>

</action>

<action name="user" class="com.bjsxt.struts2.user.action.UserAction">

<result>/user\_add\_success.jsp</result>

</action>

通配符：

使用通配符，将配置量降到最低<br />

<a href=*"actions/Studentadd"*>添加学生</a>

<a href=*"actions/Studentdelete"*>删除学生</a>

<br />

不过，一定要遵守"约定优于配置"的原则

<br />

<a href=*"actions/Teacher\_add"*>添加老师</a>

<a href=*"actions/Teacher\_delete"*>删除老师</a>

<a href=*"actions/Course\_add"*>添加课程</a>

<a href=*"actions/Course\_delete"*>删除课程</a>

<action name=*"Student\*"* class=*"com.struts.action.StudentAction"* method=*"{1}"*>

<result>/Student{1}\_success.jsp</result>

</action>

<action name=*"\*\_\*"* class=*"com.struts.action.{1}Action"* method=*"{2}"*>

<result>/{1}\_{2}\_success.jsp</result>

</action>

用action属性接收参数：

使用action属性接收参数<a href="user/user!add?name=a&age=8">添加用户</a>

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

<action name=*"user"* class=*"com.struts.action.UserAction"*>

<result name=*"success"*>/Studentadd\_success.jsp</result>

</action>

</package>

用DomainModel接收参数：

<a href="user/user!add?user.name=a&user.age=8">添加用户</a>

在action里面不再有name,age属性了，而是user,user是一个model（javabean），

有时候需要一个dto（data transfer object）来传递数据。

用ModelDriven接收参数：

<a href="user/user!add?name=a&age=8">添加用户</a>

在action里面要实现一个接口：implements ModelDriven<User>

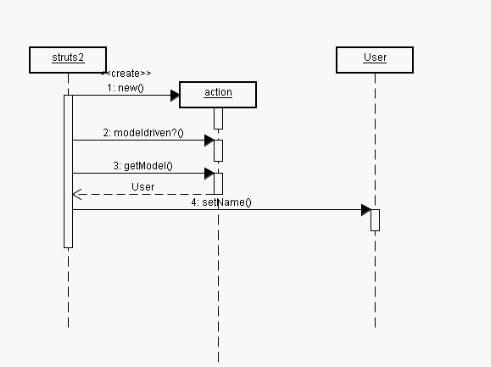
属性是user，但是要new出来：private User user = new User();

而且要实现一个方法：@Override

public User getModel() {

return user;

}

这里重要的是理解mvc，

中文问题：

在struts.xml里面配置：<constant name=”struts.i18n.encoding” value=”GBK”/>

重点是查看缺省配置文件，在struts-core里面又struts-default.xml文件，还有default.properties文件可以查看。以后还有经常查看struts的docs文档

简单数据验证：

Action里面：**if**(name == **null** || !name.equals("admin")) {

**this**.addFieldError("name", "name is error");

**this**.addFieldError("name", "name is too long");

**return** *ERROR*;

}

Jsp页面：**<%@taglib uri="/struts-tags" prefix="s" %>**

<s:fielderror fieldName=*"name"* theme=*"simple"*/>

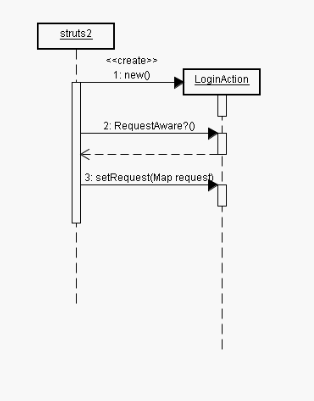
<br />

<s:property value=*"errors.name[0]"*/>

<s:debug></s:debug>

一般用第二种比较多，debug用来显示所有信息

访问web元素：这个内容很重要，必须理解，内容就不写了，去看练习的代码就可以了，这里只显示了最常用的Ioc方法的机制图



模块包含：

就是包含一个普通的xml文件：

在struts.xml文件里写: <include file="login.xml" />

默认Action:

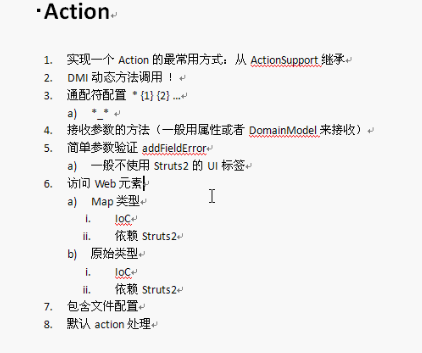
<default-action-ref name="index"></default-action-ref>

<action name="index">

<result>/default.jsp</result>

</action>

Action总结：



Result：

常见四种类型：去查文档，例如第三种chain的时候如果要跳到其他package的时候如何跳

<action name="r1">

<result type="dispatcher">/r1.jsp</result>

</action>

<action name="r2">

<result type="redirect">/r2.jsp</result>

</action>

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

<action name=*"ognl"* class=*"com.struts.OgnlAction"*>

<result>/ognl.jsp</result>

</action>

<action name=*"test"* class=*"com.struts.TestAction"*>

<result type=*"chain"*>

<param name=*"actionName"*>ognl</param>

<param name=*"namespace"*>/ognl</param>

</result>

</action>

</package>

<action name="r4">

<result type="redirectAction">r2</result>

</action>

Global\_Results\_全局结果集：

在一个包里面可以写一个全局的result，这样的话就可以只写一个让其他action都可以执行。如果其他包也想用到这个Global\_Results，就需要继承。下面是例子：

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

<global-results>

<result name="mainpage">/main.jsp</result>

</global-results>

<action name="index">

<result>/index.jsp</result>

</action>

<action name="user" class="com.bjsxt.struts2.user.action.UserAction">

<result>/user\_success.jsp</result>

<result name="error">/user\_error.jsp</result>

</action>

</package>

<package name="admin" namespace="/admin" extends="user">

<action name="admin" class="com.bjsxt.struts2.user.action.AdminAction">

<result>/admin.jsp</result>

</action>

</package>

动态结果集\_dynamic\_result：这个一般用到不多

<action name="user" class="com.bjsxt.struts2.user.action.UserAction">

<result>${r}</result>

</action>

在action里面：

private String r;

public String execute() throws Exception {

if(type == 1) r="/user\_success.jsp";

else if (type == 2) r="/user\_error.jsp";

return "success";

}

一次request只有一次value stack。

带参数的结果集:一般在forward的时候不用传参数，因为共用一个值栈。在redirect的时候需要用到。

Action里面不用写什么，struts.xml里面写：（type是jsp传过来的参数，也是action里面的参数）

<action name="user" class="com.bjsxt.struts2.user.action.UserAction">

<result type="redirect">/user\_success.jsp?t=${type}</result>

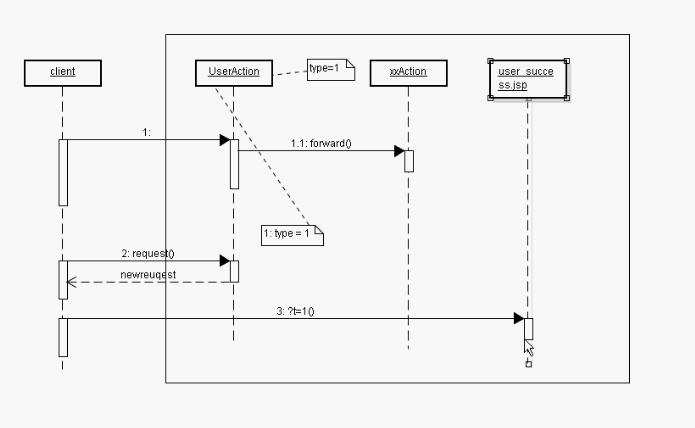
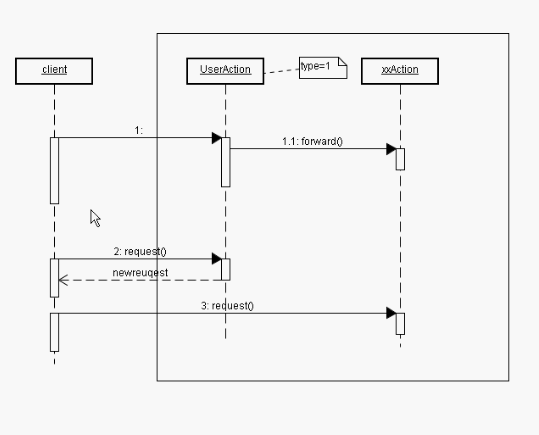
</action>

user\_success.jsp里面：（第一个是验证的，没有值，因为只是jsp页面跳转，没有用到action，也就没有值栈。第二个才能取到值，可以查看debug！）

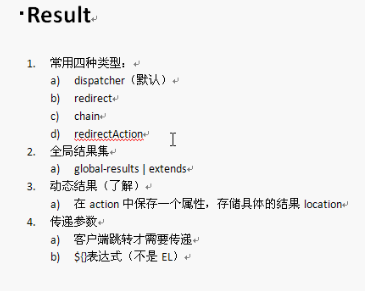
from valuestack: <s:property value="t"/><br/>

from actioncontext: <s:property value="#parameters.t"/>

<s:debug></s:debug>



Result总结：





# Struts-Tags

1. 通用标签：
   1. property
   2. set
      1. 默认为action scope，会将值放入request和ActionContext中
      2. page、request、session、application
   3. bean
   4. include(对中文文件支持有问题，不建议使用，如需包含，改用jsp包含)
   5. param
   6. debug
2. 控制标签
   1. if elseif else
   2. iterator
      1. collections map enumeration iterator array
   3. subset
3. UI标签
   1. theme
      1. simple xhtml(默认) css\_xhtml ajax
4. AJAX标签
   1. 补充
5. $ # %的区别
   1. $用于i18n和struts配置文件
   2. #取得ActionContext的值
   3. %将原本的文本属性解析为ognl，对于本来就是ognl的属性不起作用
      1. 参考<s:property 和 <s:include

# 定义自己的theme

1. css（覆盖struts2原来的css）
2. 覆盖单个文件
3. 定义自己的theme
4. 实战：
   1. 把所有主题定义为simple
   2. fielderror特殊处理
   3. 自己控制其他标签的展现

**OGNL 看练习的项目**

1. Object Graph Navigation Language
2. 想初始化domain model，可以自己new，也可以传参数值，但这时候需要保持参数为空的构造方法
3. 其他参考ognl.jsp
4. **什么时候在stack中会有两个Action？chain**

**命名约定:**

* 1. **数据库表说明文档**
  2. **设计约定：简单就是美**

**表的命名：\_Model名**

**字段命名：保持和属性名一致（尽量不要起名和数据库命名冲突）**

**库名：项目名**

* 1. **用层来划分包**
  2. **XXXAction**
  3. **\*-\*的配置**
  4. **前台的package命名空间/**
  5. **后台的package命名空间/admin**
  6. **Package的命名**

1. 设计约定（编码规定）
   1. 原则：简单就是美
   2. 库名：项目名
   3. 表的命名：\_Model名
   4. 字段：保持和属性名一致（尽量不要起名和数据库命名冲突）
   5. 用层来划分包com.bjsxt.bbs.action model(bean) service dto(vo)
   6. Action XXXXAction
   7. \*-\*
   8. /front
   9. /admin
   10. package “action” adminAction

# 项目开发顺序-以BBS2009的名义

1. 建立界面原型
2. 建立Struts.xml
   1. 确定namespace
   2. 确定package
   3. 确定Action的名称，空的方法
   4. 确定Result
   5. 将界面原型页面进行修改，匹配现有设置
   6. 测试
   7. 做好规划！！！！！
3. 建立数据库（或者实体类）
4. 建立Model层
5. 建立Service层（后面讲了Hibernate后再完善）
   1. 此时可以使用JUnit进行单元测试了
6. 着手开发

**声名式的异常处理:**

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

**<package name="bbs2009\_default" 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>**

**<package name="admin" namespace="/admin" extends="bbs2009\_default" >**

**<action name="index">**

**<result>/admin/index.html</result>**

**</action>**

**<action name="\*-\*" class="com.bjsxt.bbs2009.action.{1}Action" method="{2}">**

**<result>/admin/{1}-{2}.jsp</result>**

**<result name="input">/admin/{1}-{2}.jsp</result>**

**<!-- <exception-mapping result="error" exception="java.sql.SQLException" />**

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

**</action>**

**<!--**

**<action name="category" class="com.bjsxt.bbs2009.action.CategoryAction">**

**<result>/admin/category\_list.jsp</result>**

**<result name="add\_input">/admin/category\_add\_input.jsp</result>**

**<result name="update\_input">/admin/category\_update\_input.jsp</result>**

**</action>**

**-->**

**</package>**

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

**<default-action-ref name="Category\_list" />**

**<action name="Category\_list" class="com.bjsxt.bbs2009.action.CategoryAction" method="list">**

**<result>/index.jsp</result>**

**</action>**

**</package>**

# 声明式异常处理

1. 在Actoin中进行异常映射
2. 在package中进行全局异常映射
3. 使用继承共用异常映射
4. Struts2中异常处理由拦截器实现（观察struts-default.xml）
   1. 实际上Struts2的大多数功能都由拦截器实现

# I18N

1. I18N原理
   1. ResourceBundle和Locale的概念
   2. 资源文件
   3. native2ascii
2. Struts的资源文件
   1. Action – Package – App级
   2. 一般只用APP
      1. struts.xml custom.i18n
   3. PropertiesEditor插件
      1. 解压
      2. features plugin 覆盖到myeclipse中的eclipse目录里
   4. 动态语言切换
      1. request\_locale=en\_US

**源码解析：**

