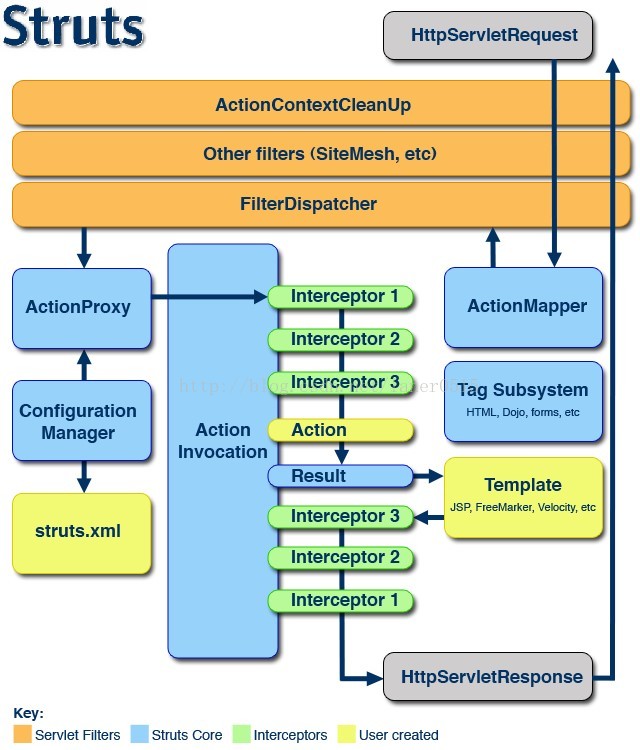
1. Struts2工作原理
2. 工作原理图



1. 工作流程
2. 客户端初始化一个指向Servlet容器（例如Tomcat）的请求
3. 这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin）。
4. 接着FilterDispatcher被调用，FilterDispatcher询问ActionMapper来决定这个请是否需要调用某个Action。
5. 如果ActionMapper决定需要调用某个Action，FilterDispatcher把请求的处理交给ActionProxy。
6. ActionProxy通过ConfigurationManager询问框架的配置文件，找到需要调用的Action类 一般是从struts.xml配置中读取。
7. ActionProxy创建一个ActionInvocation的实例。
8. ActionInvocation实例使用命名模式来调用，在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用。
9. 一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果。返回结果通常是（但不总是，也可能是另外的一个Action链）一个需要被表示的JSP或者FreeMarker的模版。在表示的过程中可以使用Struts2 框架中继承的标签。在这个过程中需要涉及到ActionMapper
10. Struts2特性
11. 访问Servlet API

1.1 创建一个ActionContext

ActionContext context=ActionContext.getContext();

1.2 通过context对象获取Servlet API

(1) Map<String,Object> getApplication() 获取的是application中存储的所有信息

(2) Map<String,Object> getParameters() 相当于 request.getParameterMap();

(3) Map<String,Object> getSession(); 获取的是session中存储的所有信息.

(4) put(String key,Object value); 就相当于request.setAttribute();

(5) Object get(String key) 就相当于request.getAttribute();

1.3 通过实现Aware接口注入方式(真正的Servlet API)

(1) ServletContextAware ： 注入ServletContext对象

(2) ServletRequestAware ：注入 request对象

(3) ServletResponseAware ： 注入response对象

1.4 通过ServletActionContext(真正的Servlet API)

(1) 通过ServletActionContext类中提供了静态方法可以直接获取Servlet API:

1. Action搜索机制

2.1 判断package是否存在

(1) 判断action是否存在，如果不存在则去默认的namespace的package里面寻找，不存在会报错。

(2) 检查上一级路径的package是否存在（直到默认的namespace），重复第一步，没有则报错。

1. 动态方法调用：为了解决一个Action对应多个请求的处理，调用方法
   1. 指定method属性，在Action标签中添加method属性，就会调用指定的方法。默认方法是execute()方法

<action name="addAction"method=”add”class="com.imooc.action.HelloWorldAction">

<result>/add.jsp</result>

</action>

3.2 感叹号方式 – 不推荐使用

(1) 在struts.xml 中添加

<constant name="struts.enable.DynamicMethodInvocation" value="true"></constant>

(2) 在<action>标签中添加<result>标签

<result name=”add”>/add.jsp</result>

<result name=”update”>/update.jsp</result>

(3) 在action类中的方法返回相应的result name

public String add(){return add;}

在浏览器输入地址时候添加！

原来：localhost:8080/helloworld.action

添加：localhost:8080/helloworld!add.action

* 1. 通配符方式 – 建议使用，将method和通配符结合起来

1. 在struts2.3版本以后使用通配符需要配置<allowed-methods>标签

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

<result>/{2}.jsp</result>

</action>

第一个\*对应{1} ， 第二个\*对应{2}

输入：localhost:8080/HelloWorld\_add.action

转化为：

<action name=" HelloWorld \_ add "method=” add”class="com.imooc.action.

HelloWorldAction">

<result>/ add.jsp</result>

</action>

1. 指定多个配置文件 -- <include>标签
2. 指定多个配置文件，如果项目比较大，xml配置文件比较多，那么可以在struts.xml中使用include标签，将多个配置文件包括进来。

<include file="\*\*\*.xml"> </include>

<constant name="struts.i18n.encoding" value="UTF-8"> </constant> //设置配置文件的编码格式

1. 被include的标签一定要符合struts的dtd规范。也就是说被include的xml文件的内部格式要符合struts的xml文件规范。
2. xml文件的编码格式要相同，如果是utf-8，那么都是utf-8。
3. 默认Action --在网页找不到原来输入的网址的情况下，跳转的页面
4. 代码实现：

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

<action name="index">

<result>/error.jsp</result>

</action>

1. Struts后缀 -- .action
   1. truts2后缀，默认是.action或者是空值
2. 在struts.xml中的配置语句

<constant name = "strts.action.extension" value = "xxx,xx,yyy"></constant>//name是固定的，value是指定的后缀，可以是空值

1. 在struts.properties中的配置

struts.action.extension = xxxx,xxx,xx//可以配置多个，以，号分隔

1. 在web.xml中的配置

<filter>//过滤器

<init-param>过滤器中假如<init-param>

<param-name>struts.action.extension</param-name>//struts.action.extension为固定

<param-value>do</param-value>//do为可选后缀

</init-param>//结束init-param

</filter>//结束过滤器

1. 在Action里面接受参数
2. 使用action类中的属性进行接收

声明定义与jsp页面中的name相同的属性就行，且写出属性的getter和setter 方法，这样属性就能自动传递过来。

(2) 定义一个新对象，在action类中利用对象的属性来获取参数值 -- DomainModel

定义一个新对象，在action类中利用user.getUsername()方法来获取。

jsp中name名称要改成 对象.属性 的形式，如：name=”user.username”

(3) 通过modelDriven实现，实现ModelDriven接口

在action上接口 implements ModelDriven<User>，对象需要实例化

private User user = new User();不需要getter/setter方法

在jsp中只需要写属性名就行了

实现：@Override

public User getModel() {

return user;

}

1. 要接收List集合的值

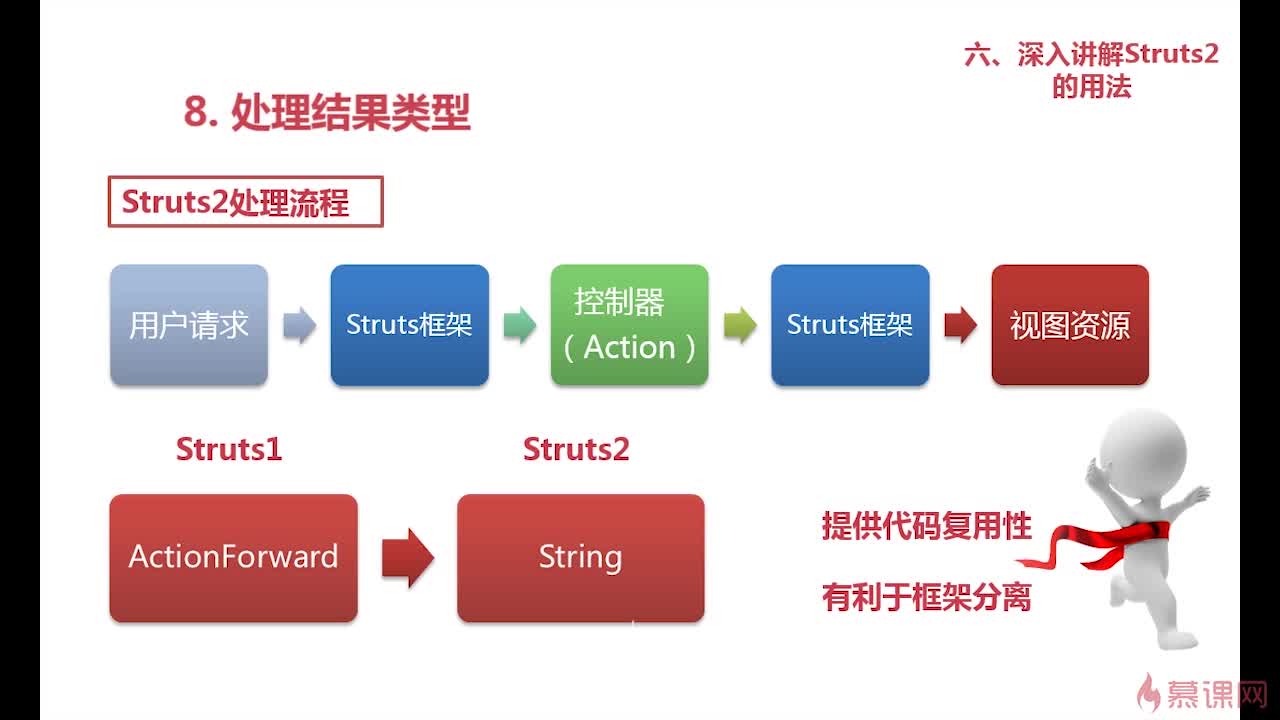
只需要在jsp中的name属性中写上 booklist[0] 这表示具体集合位置的参数就 可以了。

1. 处理结果类型

8.1 <result name=”success”>/success.jsp</result>

<result >/index.jsp</result>

name的默认值为secuss,在action类中返回的String类型映射到相应的<result>标签，然后跳转到相应的jsp视图中。





8.2 结果类型input的效果 – 自动跳转

(1) 当参数类型转换错误时，如age输入框中的类型是字母等情况，方法自动返回input

(2) 当action中存在addFiledError时：

(a) addFileError放在一般执行方法，addFieldError("", "");语句后面有返回input的语句

(b) addFileError放在validate()中,则不需要return INPUT;

(3) FileError的表现形式：

在jsp页面中使用<s:fielderror></s:fielderror>标签，该标签name属性为addFieldError 方法中的参数fieldName，在jsp页面中使用struts标签，

需要导入标签库 语句：<%@ taglib prefix="s" uri="/struts-tags" %>

* 1. 全局结果和局部结果

1. 局部结果：<result>标签是放在<action>标签里面的，只限此<action>标签使用。
2. 全局结果：<result>标签是放在<action>标签之外的，可以多个action公用这个<result>

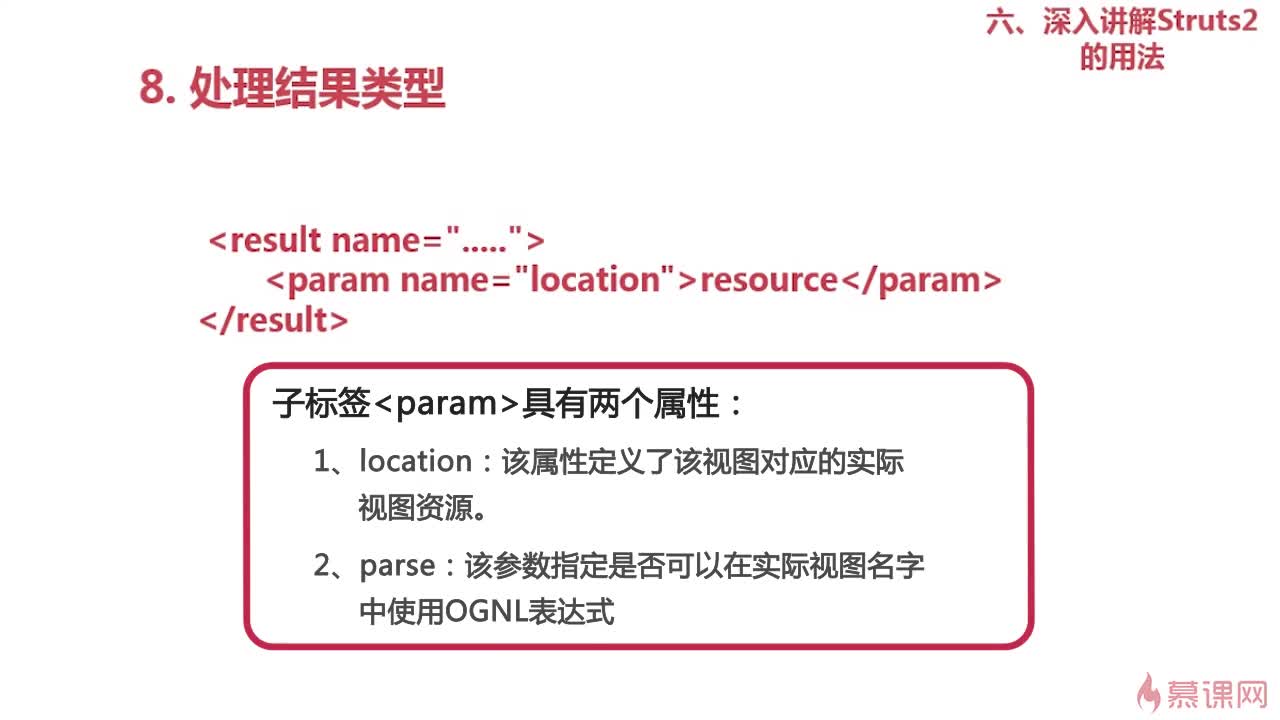
形式：

<global-result>

<result name=”ERROR404”><result>

</ global-result >

1. <result>标签下的子元素



1. <result>标签中的type属性

type的默认值是dispatcher 支持jsp视图技术

属性值：chain：将action和另外一个action链接起来。

属性值：redirect：重定向（会丢失请求参数）。

属性值：plaintext：返回网页源代码。

属性值：stream：返回inputstream用于文件下载。