任务：

1. 搭建一个简单的Struts 框架
2. 学习struts2的titles

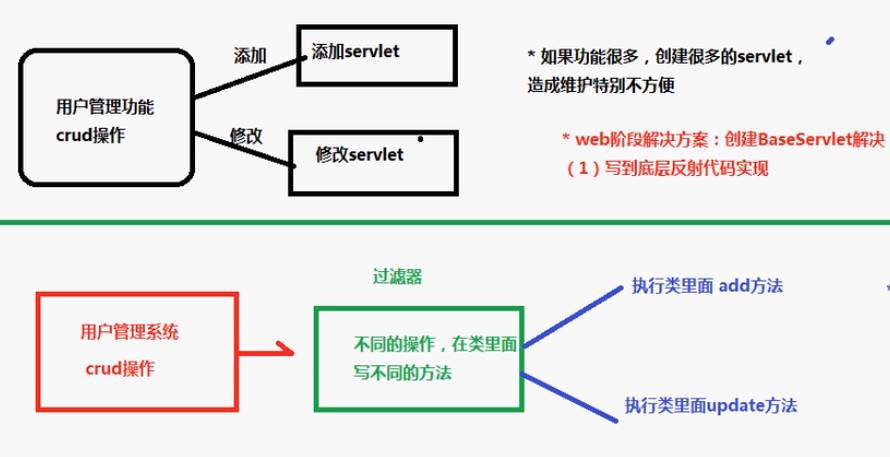
Struts2学习：

**一、struts2概述**

**1.struts2框架应用java ee三层结构中的web层框架。**

**2.struts2框架是在struts1和webwork基础之上发展全新的框架。**

3.struts2解决问题



4.struts2 版本



1. Web层常见框架

①struts2

②springMVC

1. **Struts2框架入门**
2. 导入jar包
3. 创建action

说明：

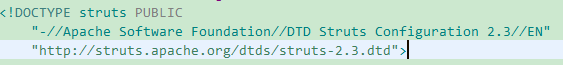


1. 配置action类访问路径
2. 创建struts2核心配置文件

核心配置文件的名称和位置是固定的。

位置中必须在src下面，名称为struts.xml

1. struts.xml引入dtd约束



1. action基本配置

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

<!-- name:名称 -->

<action name=*"hello"* class=*"cn.itcast.action.HelloAction"*>

<!-- 配置方法的返回值到页面中去 -->

<result name=*"ok"*>/hello.jsp</result>

</action>

</package>

访问路径：<http://127.0.0.1:8080/struts2_day01/hello.action>

1. 配置struts2过滤器（核心步骤）

<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>

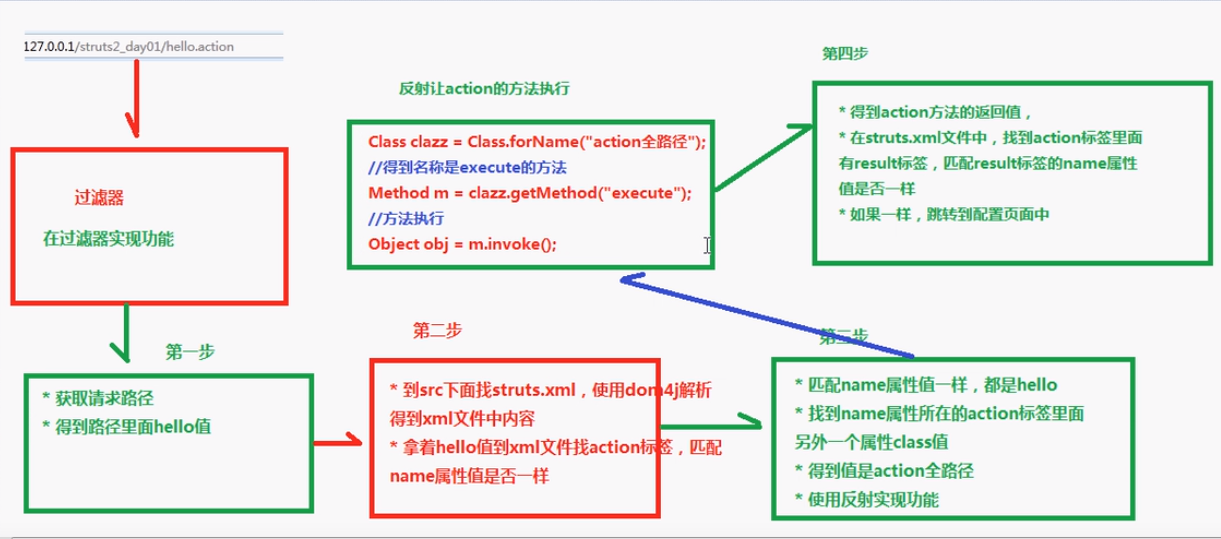
<url-pattern>/\*</url-pattern>

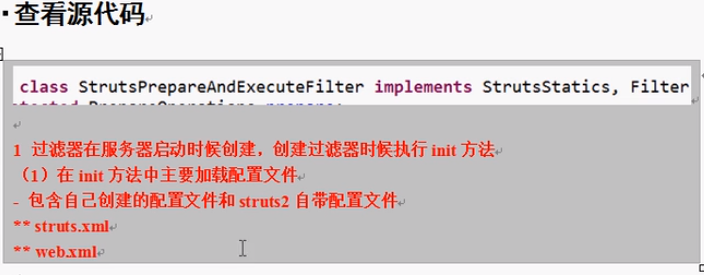
</filter-mapping>

浏览器运行



1. **Struts基本执行过程**



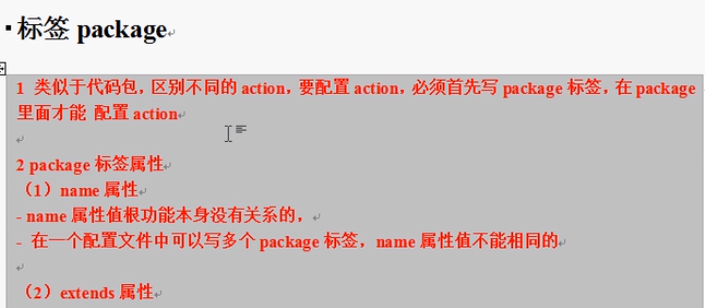


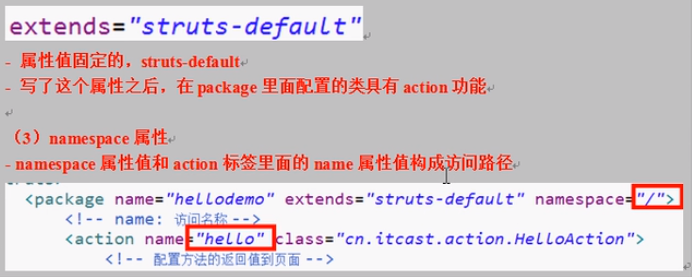
1. **Struts2配置**

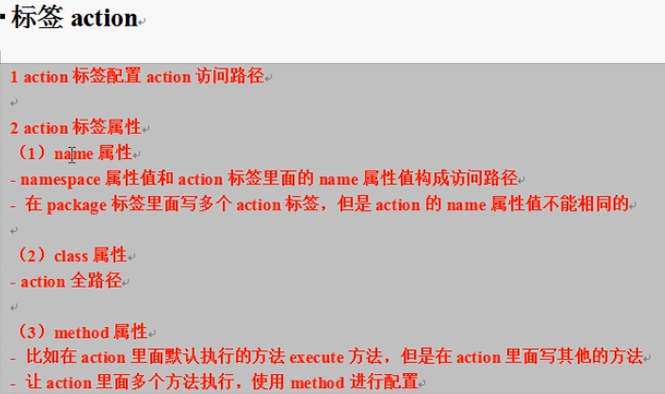
Struts2的核心配置文件

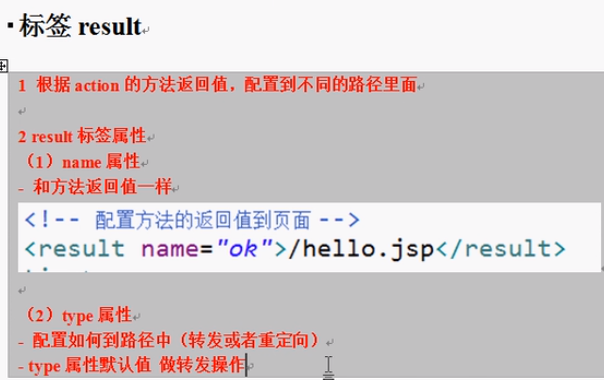


1. 名称和位置固定。
2. 在配置文件中主要三个标签package、action、result，标签里面的属性。

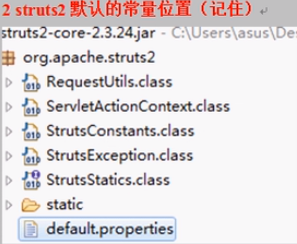








1. 修改struts2默认常量值



1. 常用方式

在struts.xml中进行配置

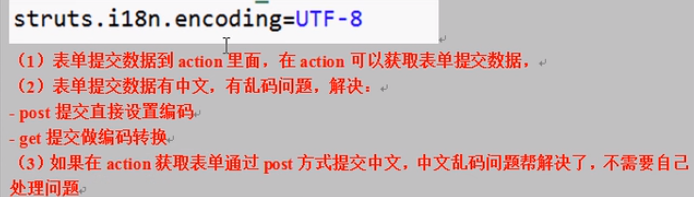
<constant name=*"struts.i18n.encoding"* value=*"UTF-8"*></constant>

1. 其他两种了解

--在src下面创建struts.properties，进行修改

--在web.xml进行配置

1. 介绍最常用常量



1. **Struts2分模块开发**
2. 单独写配置文件，把配置文件引入到核心配置文件中

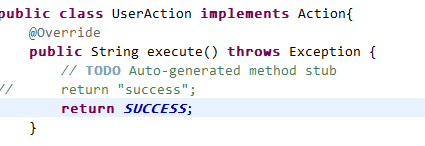
<!-- 引入hello.xml文件 -->

<include file=*"cn/itcast/action/hello.xml"*></include>

1. **Struts2中action编写方式**
2. action编写有三种方式

第一种 创建普通类，这个不继承任何类，不实现任何接口

第二种 创建类，实现接口Action



第三种，继承类ActionSupport（常用）

/\*\*

\* 继承类

\* **@author** modefy

\*

\*/

**public** **class** PersonAction **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

// **TODO** Auto-generated method stub

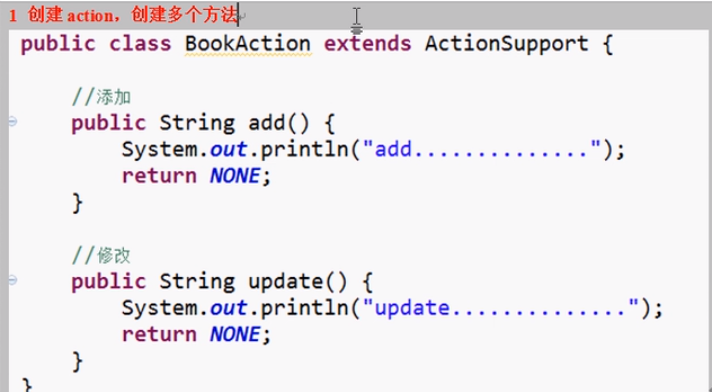
**return** ***SUCCESS***;

}

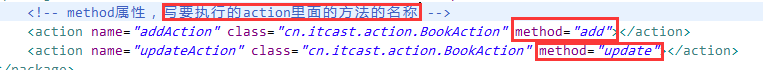
}

1. **Struts2中访问action的方法（重点）**
2. 有三种方式实现

第一种 使用action标签的method属性，在这个属性里面写执行的方法。



使用method进行配置



第二种 使用通配符方式实现（重点）

<!-- 使用通配符方式实现 -->

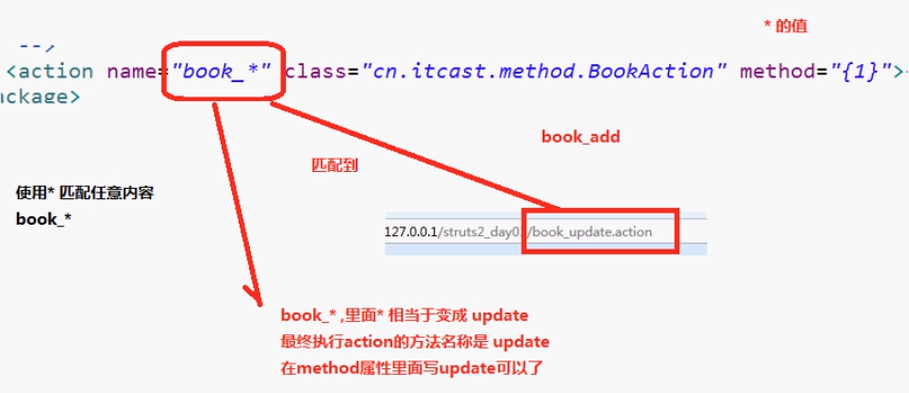
<package name=*"methoddemo"* extends=*"struts-default"* namespace=*"/"*>

<!-- name属性值里面写符号\* -->

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

</package>

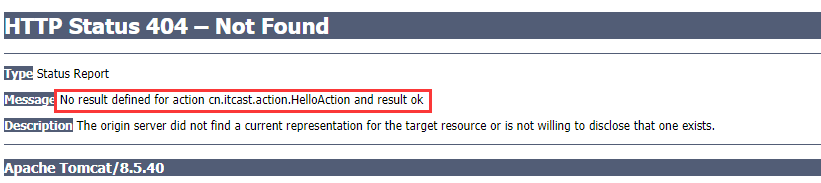
分析：



第三种 动态访问实现（不用）

1. 演示错误

（1）如果action方法有返回值，在配置文件中没有配置，出现错误：



（2）在action里面的方法有返回值，如果有返回值的时候，返回值类型必须是String类型。

（3）action里面的方法可以没有返回值。如果没有返回值，在result标签里就不需要配置。

--把方法写成void

--让返回值，返回“none”

1. **Struts2中结果页面配置**

全局结果页面

1. result标签配置action方法的返回值到不同的路径里面。



<!-- 全局结果页面的配置 -->

<global-results>

<result name=*"success"*>/hello.jsp</result>

</global-results>

局部结果页面



1. 既配置了全局页面，也配置了局部页面，则最终以局部配置为准

result标签中的type属性（重点）

1.type属性：如何到路径中去的（转发还是重定向）

2.type属性值：

（1）默认值，做转发（一次请求）操作，值是dispatcher

<result name=*"success"* type=*"dispatcher"*>/hello.jsp</result>

1. 做重定向（两次请求）操作，值是redirect

<result name=*"success"* type=*"redirect"*>/hello.jsp</result>

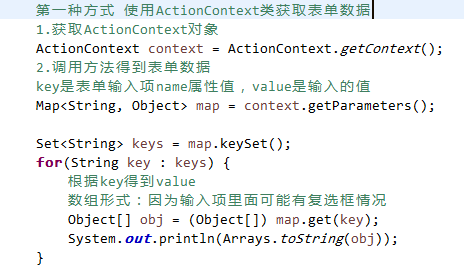
dispatcher、redirect针对的是到页面中的配置。

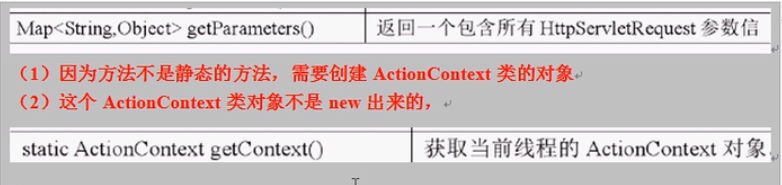
若配置到其他action里面：

--chain: 转发到action ，一般不用，缓存问题

--redirectAction: 重定向到action

1. **在action获取表单提交数据**
2. 之前web阶段，提交表单到servlet里面使用request对象里面的方法获得，getParameter，getParameterMap。
3. 提交表单到action，但是action没有request对象，不能直接使用request对象。
4. action获取表单提交数据主要三种方式
5. 使用ActionContext类





步骤：

1. 创建表单，提交表单到action里面。
2. 在action使用ActionContext获取数据。
3. 使用ServletActionContext类



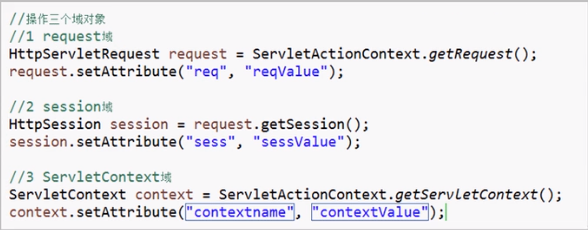
1. 使用接口注入方式（一般不用）



1. **在action操作域对象**

Request、session、servletContext域对象

使用ServletActionContext类操作



1. **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

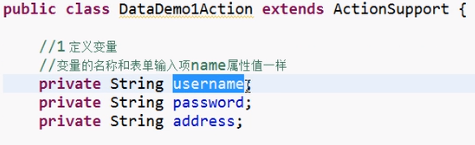


使用框架

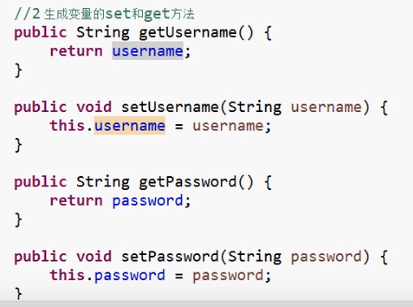
属性封装（会用）

1. 直接把表单提交属性封装到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方法（把set和get方法都写出来）



3.使用属性封装获取表单数据到属性里面，不能把数据直接封装到实体类对象里面。

模型驱动封装（重点）

1.好处：使用模型驱动方式，可以直接把表单数据封装到实体类对象里面。

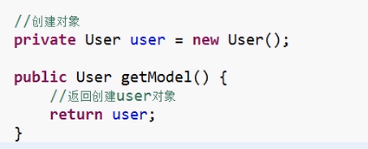
2.实现步骤：

（1）action实现接口ModelDriven



（2）实现接口里面的方法getModel方法，把创建对象返回。

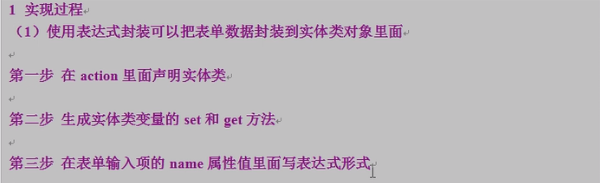
1. 在action里面创建实体类对象。



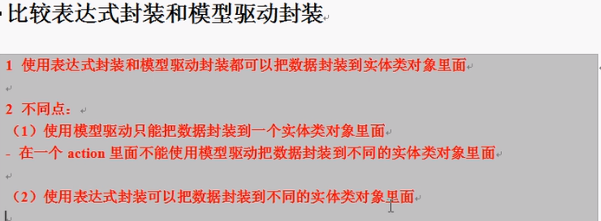
1. 前提条件：表单输入项name属性值和实体类属性名称一样。
2. 使用模型驱动和属性封装注意问题

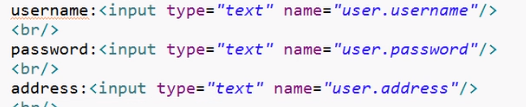
（1）在一个action中，获取表单数据可以使用属性封装或者模型驱动封装，但是不能同时使用属性封装和模型驱动封装获取同一个表单数据，如果同时使用，只会执行模型驱动。

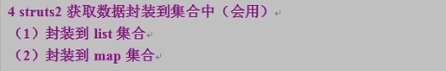
表达式封装（会用即可），有时候归类到属性封装里面











值栈

拦截器

注意：Server 中Tomcat不要使用restart，要先关闭再启动。初始时先publish再start。