# 一、框架概述

## 1、框架的意义与作用：

所谓框架，就是把一些繁琐的重复性代码封装起来，使程序员在编码中把更多的经历放到业务需求的分析和理解上面。

特点：封装了很多细节，程序员在使用的时候会非常简单。

## 2、五大框架：

Struts2，Hibernate，Spring , SpringMvc, Mybatis(Ibatis)

SSH

SSM/SSI

## 3、学好框架：

由于框架中细节很多，知识点比较零散，课后总结和做好笔记就变得尤为重要。

# 二控制器：MVC中的控制器

## 1、MVC回顾

M：Model 模型，封装数据。javabean

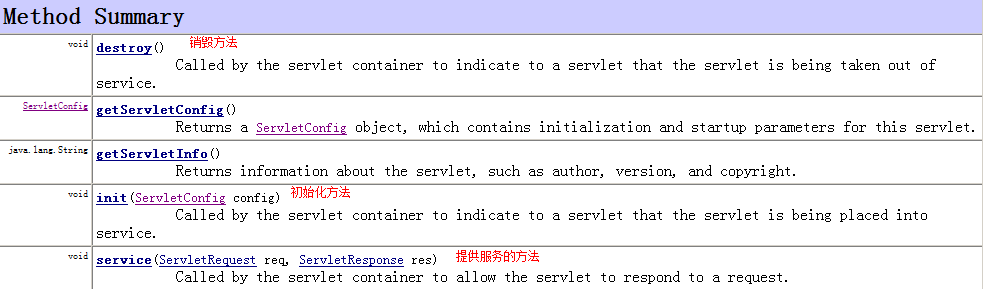
V：view 视图，展示界面。jsp

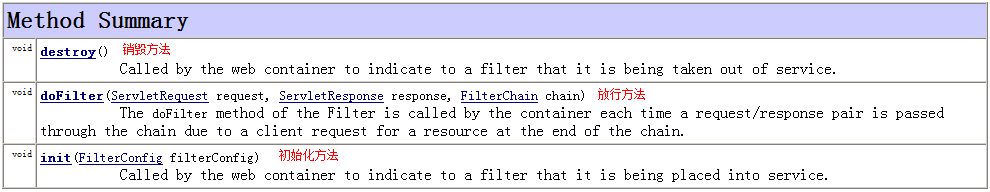
C：Controller 控制器，控制程序流程。Servlet

分层：为了简化企业级开发的 而提出来的分层，每一层有相应的功能。

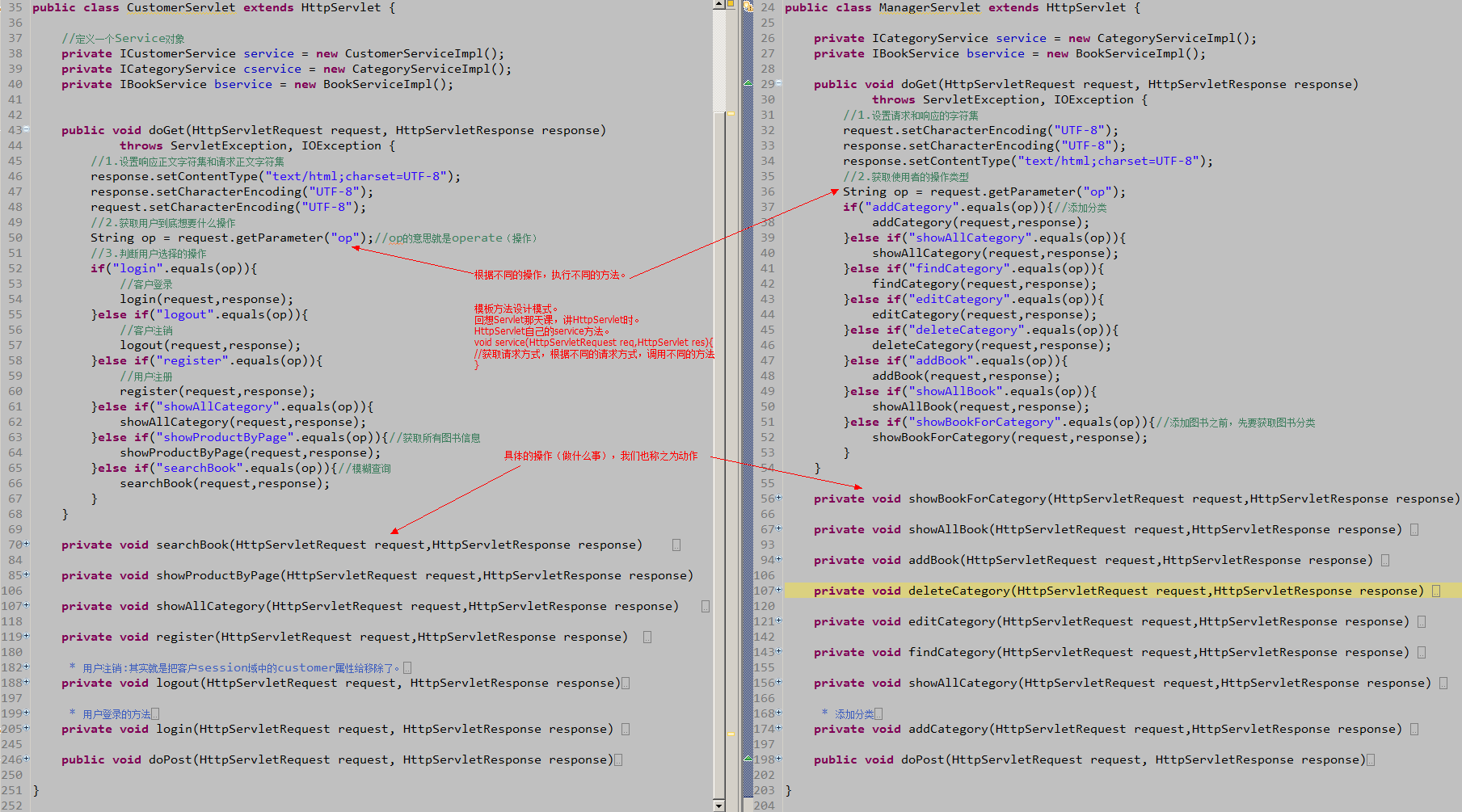
## 2、Servlet和Filter

Servlet:



Filter：

# 三、案例中的问题



**规律：**

**1、获取请求正文，用户要做什么事情**

**2、根据不同的请求，做出不同的判断**

**3、执行具体的方法代码（动作）**

**4、转向页面，展示给用户**

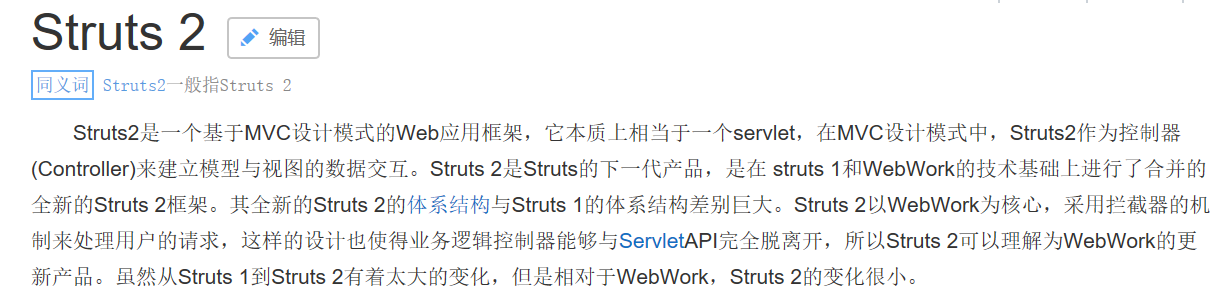
**缺陷：**

**1、重复性劳动太多，具体的执行代码也有冗余代码。**

**2、到底要做什么全是硬编码，像用户要做什么事情，对应执行什么代码，可以写在配置文件中。**

**3、具体的代码方法放到了控制器中，过于臃肿。**

# 四、Struts2简介



## 1、Struts2概述

Struts2是Apache发行的MVC开源框架。注意：充当控制器层或者说是web层的框架

## 2、Struts2的来历

Struts1：也是apache开发的一套mvc的开源框架。在2005年之前非常流行。

弊端：Struts1的核心控制器就是一个Servlet。随着使用者的增多，弊端开始出现。

Struts2：在long long ago，有一个设计超前的框架XWork，后来推出了XWork1和WebWork2。Struts2就是apache和OpenSymphony组织合并开发出来。

里面包含了WebWork2的核心及Struts的一些特性和功能。除此之外，和Struts1没有任何关系了。

# 五、搭建Struts2开发环境

## 1、下载Struts2开发包

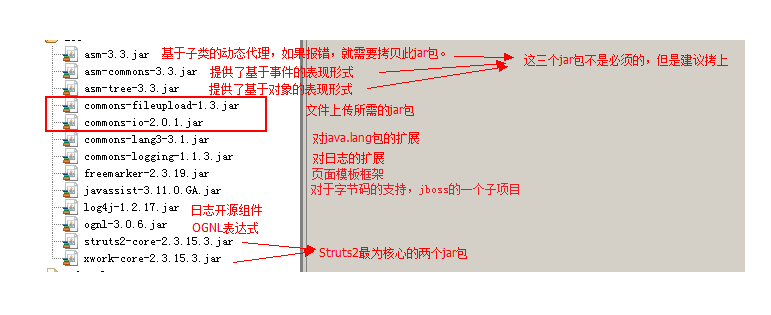
http://struts.apache.org

## 2、开发包目录结构



## 3、搭建开发环境

### 3.1、拷贝必要jar包到classpath中

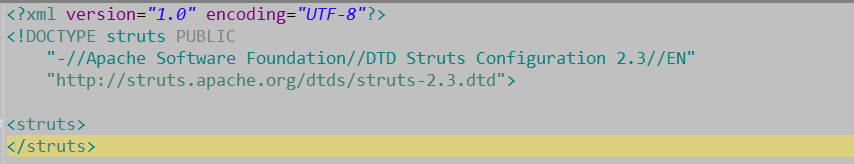


拷贝哪些jar包？

找到Struts2自带的例子中，struts-blank的war包，在它的\WEB-INF\lib目录下的jar包全拷贝。

### 3.2、建立Struts2的配置文件

at the top of classpath（在最顶层的构建路径）,建立一个默认名称为**struts.xml**的配置文件。



注意：

1.文件名大小写。

2.创建位置。

3.该名称允许修改，但是我们一般不改。

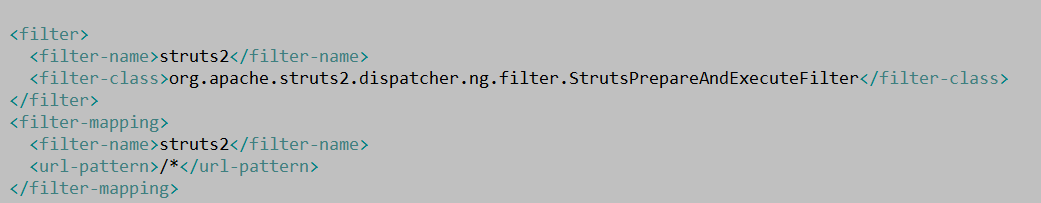
### 3.3、配置控制器

a、配置位置：在web.xml中

b、配置什么： struts2已经写好了的一个过滤器。

结论：

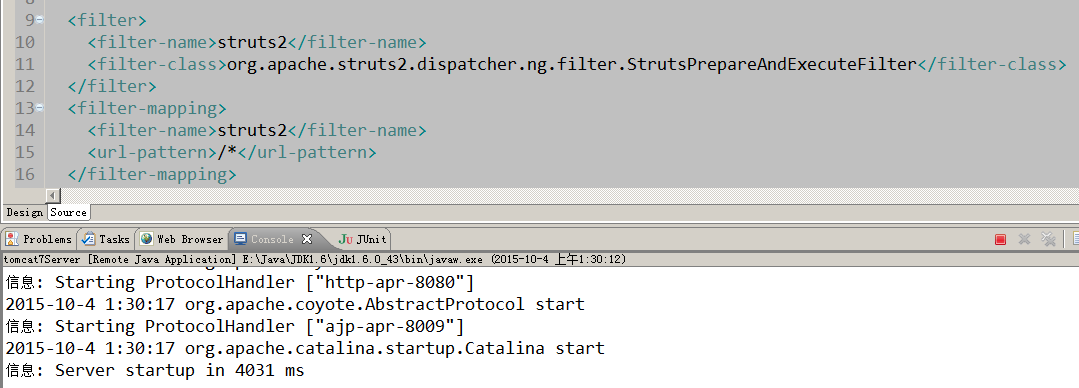
struts2比struts1优秀的一个体现就是，它用了更为强大的过滤器作为控制器了。



|  |
| --- |
| <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> |

### 3.4、验证是否成功

部署应用，启动Tomcat，不报错表示搭建成功。



# 六、第一个Struts2案例

以下步骤是日后实际开发中经常重复的。

## 1、建立一个jsp文件



## 2、在struts.xml文件中配置

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">    <struts>  <package name="default" namespace="/" extends="struts-default">  <action name="hello" class="com.qf.action.HelloAction" method="sayHello">  </action>  </package>  </struts> |

## 3、建立动作类和动作方法

Action类是动作类，是Struts2处理请求，封装数据，响应页面的控制器。需要自己编写。

|  |
| --- |
| public String sayHello(){  System.out.println("Hello Struts2!!");  return null;  } |

## 4、结果视图页面

## 5、测试：

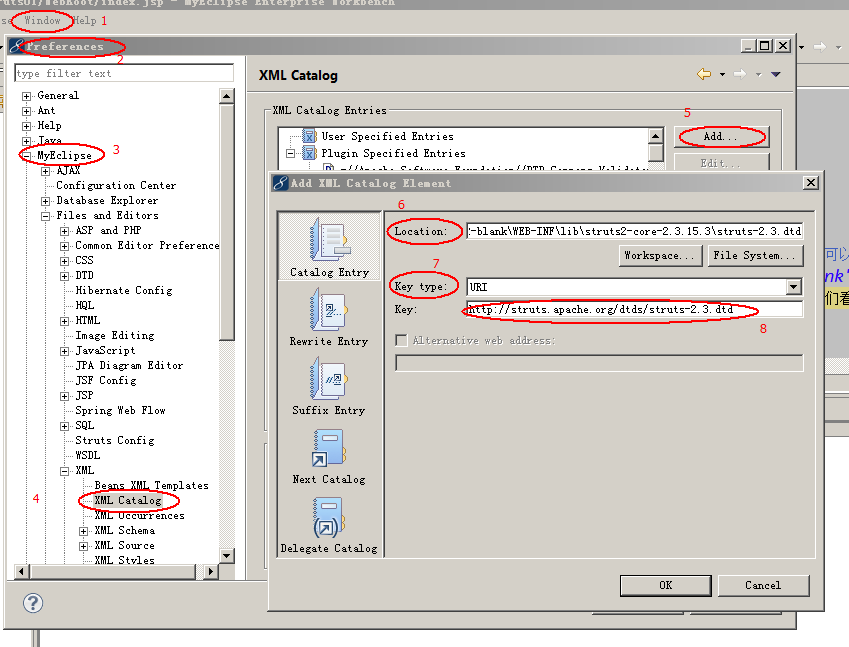
## 6、关于struts.xml没有提示的问题

分析原因：没有找到对应的dtd约束文件。

解决办法：

a.上网

b.不能上网：



## 七、Struts2的配置文件加载的顺序

### 1、加载时机：

当应用被tomcat加载的时候，struts2的配置文件就已经被加载过了。

### 2、加载顺序

顺序 配置文件名 所在位置 说明

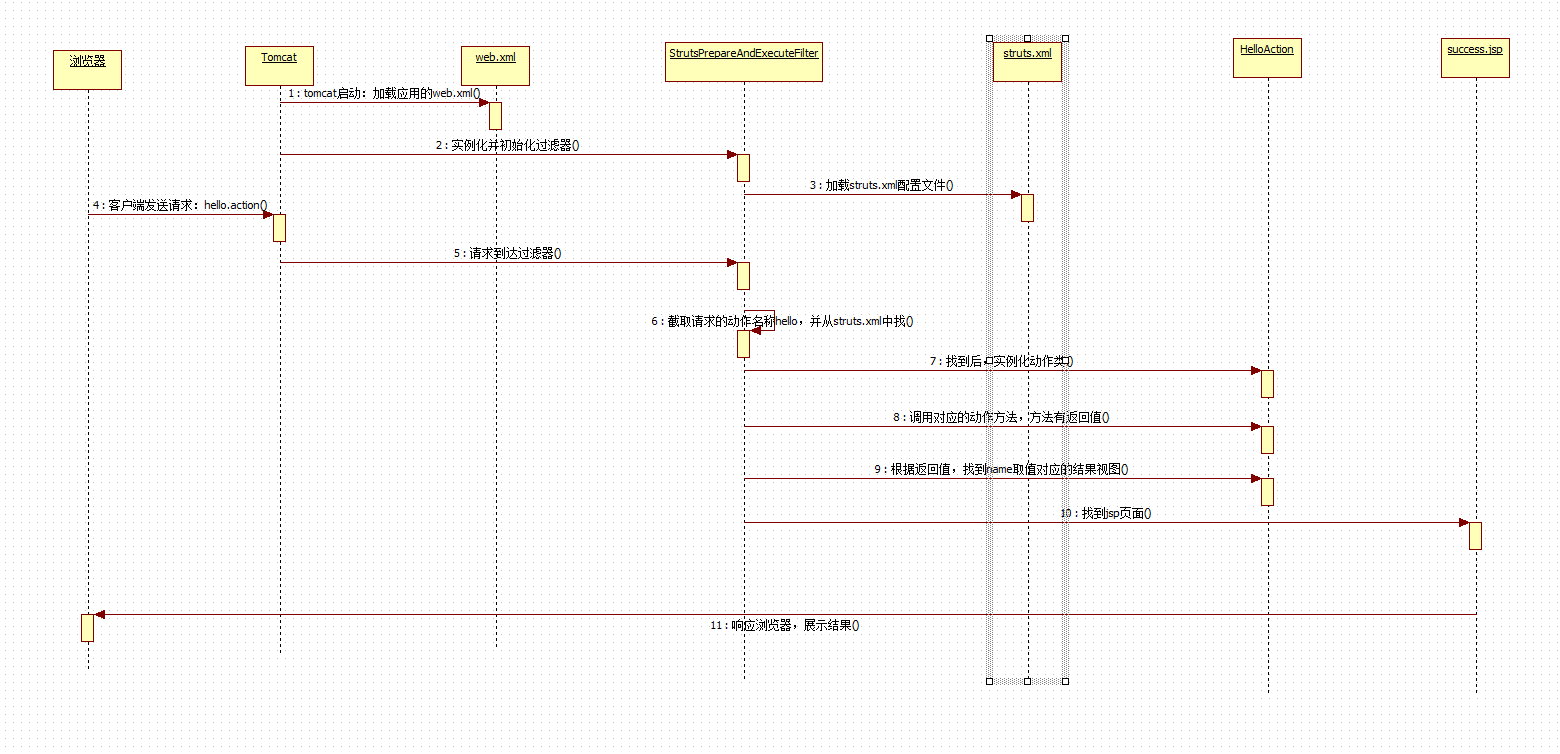
|  |
| --- |
| 1 default.properties struts2-core-2.3.15.3.jar\org\apache\struts2 不能修改  struts.i18n.encoding=UTF-8 解决乱码  struts.multipart.maxSize=2097152 上传文件的一个最大值 2M  struts.action.extension=action,, struts2的action默认的访问的后缀名  struts.serve.static.browserCache=true 告诉浏览器是否把一些静态的资源缓存到浏览器  struts.devMode = false 开发者 模式 会提示一些比较详细的错误信息 自己加载修改后的struts.xml配置文件  2 struts-default.xml struts2-core-2.3.15.3.jar 不能修改  3 strtuts-plugin.xml 在struts2提供的插件jar包中 不能修改  4 struts.xml 我们的应用中 我们修改的：推荐  5 struts.properties 我们的应用中 我们修改的  6 web.xml 我们的应用中 我们修改的，可以给过滤器配置参数  注意：配置信息会被覆盖。 |
|  |

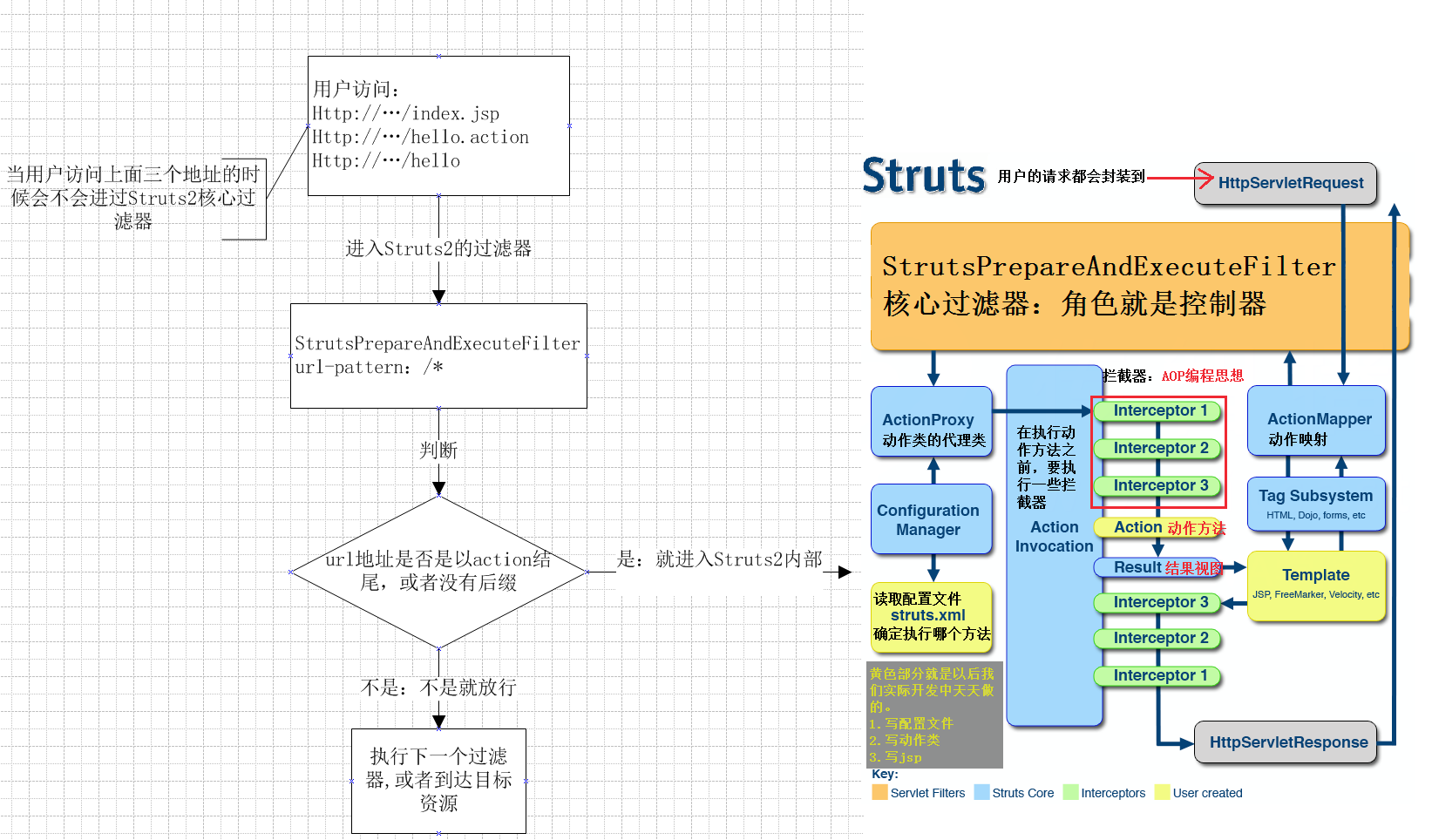
**注意：**

1、Struts2提供了两种配置的方式。一种是key=value的方式，即使用.properties文件。另一种是xml文件配置。我们推荐使用xml文件（它能描述层级关系）。

|  |
| --- |
| 重点了解的配置文件  \* default.properties -- 在org/apache/struts2/目录下，代表的是配置的是Struts2的常量的值  \* struts-default.xml -- 在Struts2的核心包下，代表的是Struts2核心功能的配置（Bean、拦截器、结果类型等）  \* struts.xml -- 重点中的重点配置，代表WEB应用的默认配置，在工作中，基本就配置它就可以了！！（可以配置常量）  \* web.xml -- 配置前端控制器  Struts2框架的核心是StrutsPrepareAndExecuteFilter过滤器，该过滤器有两个功能  Prepare -- 预处理，加载核心的配置文件  Execute -- 执行，让部分拦截器执行    StrutsPrepareAndExecuteFilter过滤器会加载哪些配置文件呢？  通过源代码可以看到具体加载的配置文件和加载配置文件的顺序  init\_DefaultProperties(); -- 加载org/apache/struts2/default.properties  init\_TraditionalXmlConfigurations(); -- 加载struts-default.xml,struts-plugin.xml,struts.xml  init\_LegacyStrutsProperties(); -- 加载自定义的struts.properties.  init\_CustomConfigurationProviders(); -- 加载用户自定义配置提供者  \* init\_FilterInitParameters() ; -- 加载web.xml |

### 3.第一个案例的执行过程



****

|  |
| --- |
| 1. 执行的流程  编写的页面，点击超链接，请求提交到服务器端。  请求会先经过Struts2的核心过滤器（StrutsPrepareAndExecuteFilter）  过滤器的功能是完成了一部分代码功能  就是一系列的拦截器执行了，进行一些处理工作。  咱们可以在struts-default.xml配置文件中看到有很多的拦截器。可以通过断点的方式来演示。  拦截器执行完后，会根据struts.xml的配置文件找到请求路径，找到具体的类，通过反射的方式让方法执行。    2. 总结  JSP页面-->StrutsPrepareAndExecuteFilter过滤器-->执行一系列拦截器（完成了部分代码）-->执行到目标Action-->返回字符串-->结果页面（result）-->页面跳转 |

## 八.入门总结之struts.xml配置文件的配置

### 1. <package>标签，如果要配置<Action>的标签，那么必须要先配置<package>标签，代表的包的概念

包含的属性

name -- 包的名称，要求是唯一的，管理action配置

extends -- 继承，可以继承其他的包，只要继承了，那么该包就包含了其他包的功能，一般都是继承struts-default

namespace -- 名称空间，一般与<action>标签中的name属性共同决定访问路径（通俗话：怎么来访问action），常见的配置如下

namespace="/" -- 根名称空间

namespace="/aaa" -- 带有名称的名称空间

abstract -- 抽象的。这个属性基本很少使用，值如果是true，

把包声明为抽象包，抽象包就是用来被继承的。只要是没有<action>元素的包，就可以声明为抽象包。那么编写的包是被继承的

### 2. <action>标签

代表配置action类，包含的属性

name -- 和<package>标签的namespace属性一起来决定访问路径的

class -- 配置Action类的全路径（默认值是ActionSupport类）

method -- Action类中执行的方法，如果不指定，默认值是execute

### 3. <result>标签

action类中方法执行，返回的结果跳转的页面

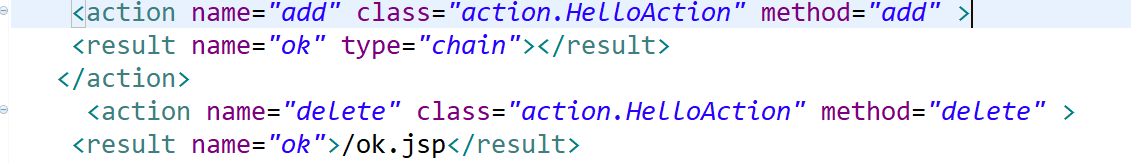
name -- 结果页面逻辑视图名称

type -- 结果类型（默认值是转发dispatcher，也可以设置其他的值）

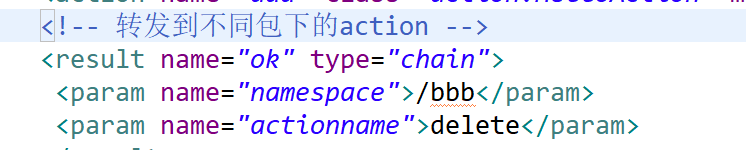
dispatcher 转发到页面 默认值

chain 转发到action

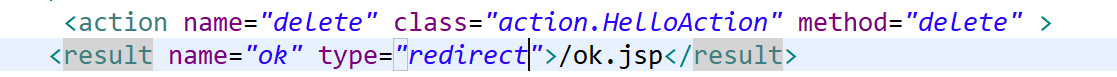
1. 转发到相同包下的action



1. 转发到不同包下的action

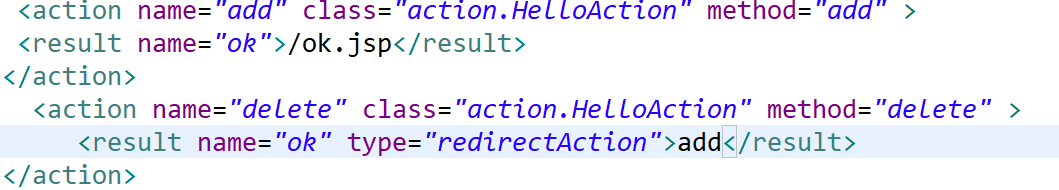


redirect: 重定向到页面

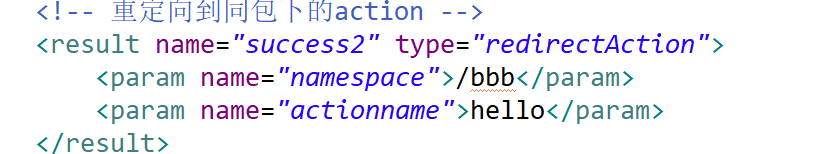


redirectAction:重定向到action

1.重定向到当前包下的action



2.重定向到其他包下的action



## 九．入门总结之Struts2配置常量

### 1. 可以在Struts2框架中的哪些配置文件中配置常量？

struts.xml（必须要掌握，开发中基本上就在该配置文件中编写常量）

<constant name="key" value="value"></constant>

web.xml

在StrutsPrepareAndExecuteFilter配置文件中配置初始化参数

注意：后加载的配置的文件的常量会覆盖之前加载的常量！！

### 2. 需要大家了解的常量

struts.i18n.encoding=UTF-8 -- 指定默认编码集,作用于HttpServletRequest的setCharacterEncoding方法

struts.action.extension=action,, -- 该属性指定需要Struts 2处理的请求后缀，该属性的默认值是action，即所有匹配\*.action的请求都由Struts2处理。如果用户需要指定多个请求后缀，则多个后缀之间以英文逗号（,）隔开

struts.serve.static.browserCache=true -- 设置浏览器是否缓存静态内容,默认值为true(生产环境下使用),开发阶段最好关闭

struts.configuration.xml.reload=false-- 当struts的配置文件修改后,系统是否自动重新加载该文件,默认值为false(生产环境下使用)

struts.devMode = false -- 开发模式下使用,这样可以打印出更详细的错误信息

## 十.入门总结之指定多个struts的配置文件（了解）

1. 在大部分应用里，随着应用规模的增加，系统中Action的数量也会大量增加，导致struts.xml配置文件变得非常臃肿。

为了避免struts.xml文件过于庞大、臃肿，提高struts.xml文件的可读性，我们可以将一个struts.xml配置文件分解成多个配置文件，然后在struts.xml文件中包含其他配置文件。

2. 可以在<package>标签中，使用<include>标签来引入其他的struts\_xx.xml的配置文件。例如：

<struts>

<include file="struts-part1.xml"/>

<include file="struts-part2.xml"/>

</struts>

3. 注意注意注意（重要的事情说三遍）：<include file="com/fm/demo2/struts-part1.xml"/>

## 十一. 技术分析之Action类的三种写法

### 1. 配置文件学习完成，下面的重点是Action类的三种写法

#### 1. Action类就是一个POJO类

什么是POJO类，POJO（Plain Ordinary Java Object）简单的Java对象.简单记：没有继承某个类，没有实现接口，就是POJO的类。

#### 2.Action类可以实现Action接口

Action接口中定义了5个常量，5个常量的值对应的是5个逻辑视图跳转页面（跳转的页面还是需要自己来配置），还定义了一个方法，execute方法。

大家需要掌握5个逻辑视图的常量

SUCCESS -- 成功.

INPUT -- 用于数据表单校验.如果校验失败,跳转INPUT视图.

LOGIN -- 登录.

ERROR -- 错误.

NONE -- 页面不转向.

#### 3.Action类可以去继承ActionSupport类（开发中这种方式使用最多）

## 十二.技术分析之Action的访问

1. 通过<action>标签中的method属性，访问到Action中的具体的方法。

传统的配置方式，配置更清晰更好理解！但是扩展需要修改配置文件等！

具体的实例如下：

页面代码

<a href="${pageContext.request.contextPath}/addBook.action">添加图书</a>

<a href="${pageContext.request.contextPath}/deleteBook.action">删除图书</a>

配置文件的代码

<package name="demo2" extends="struts-default" namespace="/">

<action name="addBook" class="com.qf.demo2.BookAction" method="add"></action>

<action name="deleteBook" class="com.qf.demo2.BookAction" method="delete"></action>

</package>

Action的代码

public String add(){

System.out.println("添加图书");

return NONE;

}

public String delete(){

System.out.println("删除图书");

return NONE;

}

2. 通配符的访问方式:(访问的路径和方法的名称必须要有某种联系.) 通配符就是 \* 代表任意的字符

使用通配符的方式可以简化配置文件的代码编写，而且扩展和维护比较容易。

具体实例如下：

页面代码

<a href="${pageContext.request.contextPath}/order\_add.action">添加订单</a>

<a href="${pageContext.request.contextPath}/order\_delete.action">删除订单</a>

配置文件代码

<action name="order\_\*" class="com.qf.demo2.OrderAction" method="{1}"></action>

Action的代码

public String add(){

System.out.println("添加订单");

return NONE;

}

public String delete(){

System.out.println("删除订单");

return NONE;

}

具体理解：在JSP页面发送请求，http://localhost/struts1/order\_add.action，配置文件中的order\_\*可以匹配该请求，\*就相当于变成了add，method属性的值使用{1}来代替，{1}就表示的是第一个\*号的位置！！所以method的值就等于了add，那么就找到Action类中的add方法，那么add方法就执行了！

3. 动态方法访问的方式（有的开发中也会使用这种方式）

如果想完成动态方法访问的方式，需要开启一个常量，struts.enable.DynamicMethodInvocation = false，把值设置成true。

注意：不同的Struts2框架的版本，该常量的值不一定是true或者false，需要自己来看一下。如果是false，需要自己开启。

在struts.xml中开启该常量。

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

具体代码如下

页面的代码

<a href="${pageContext.request.contextPath}/product!add.action">添加商品</a>

<a href="${pageContext.request.contextPath}/product!delete.action">删除商品</a>

配置文件代码

<action name="product" class="com.qf.demo2.ProductAction"></action>

<action name=*"admin!\*"* class=*"action.AdminAction"* method=*"{1}"*>

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

<result name=*"success1"* type=*"redirect"*>/success1.jsp</result>

<result name=*"success2"* type=*"redirect"*>/success2.jsp</result>

<result name=*"success3"* type=*"redirect"*>/success3.jsp</result>

</action>

Action的类的代码

public class ProductAction extends ActionSupport{

public String add(){

System.out.println("添加订单");

return NONE;

}

public String delete(){

System.out.println("删除订单");

return NONE;

}

}