# 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。

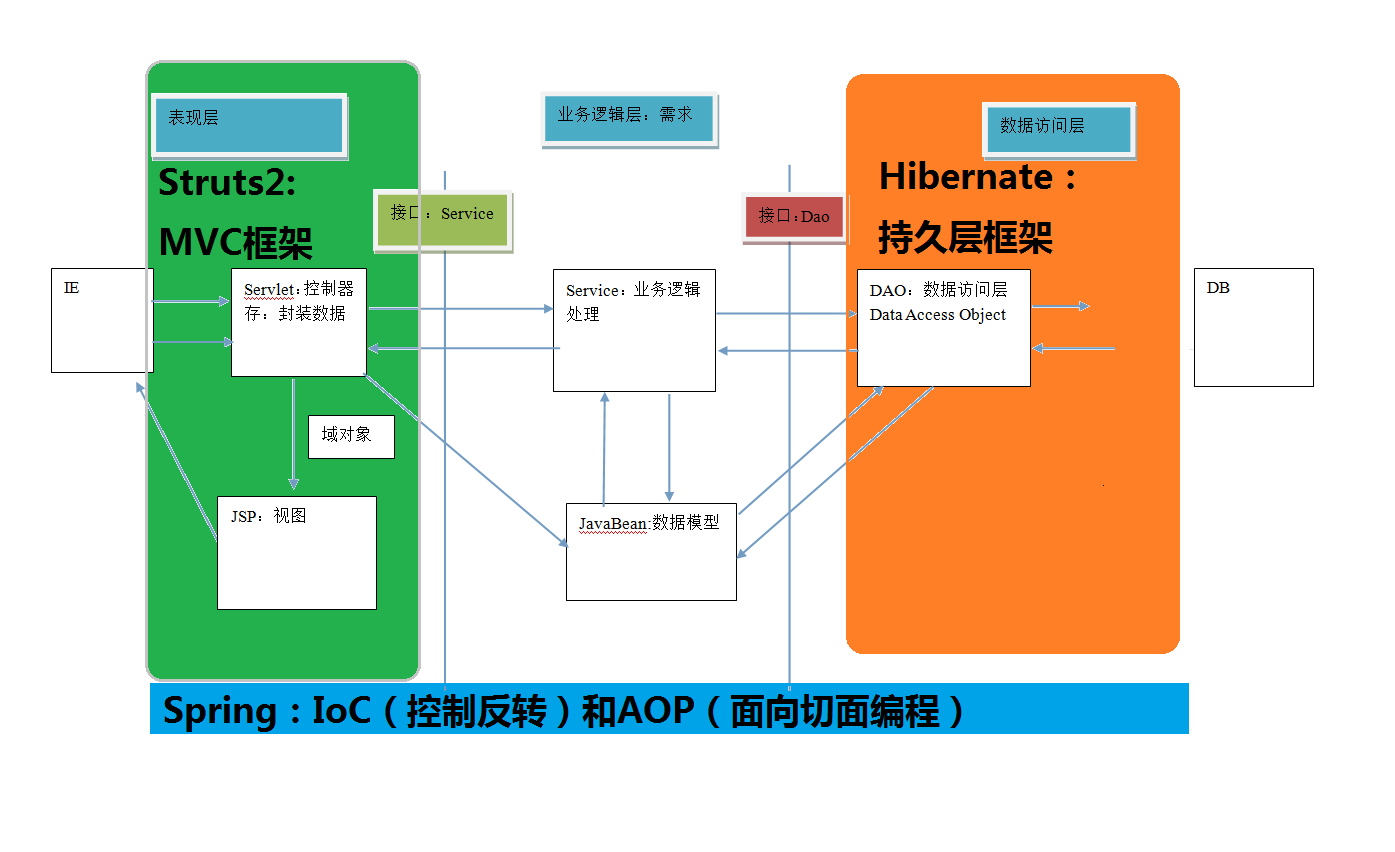
## 2、三大框架：

Struts2，Hibernate，Spring

## 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、关于三层架构



# 三、控制器：MVC中的控制器

## 1、MVC回顾

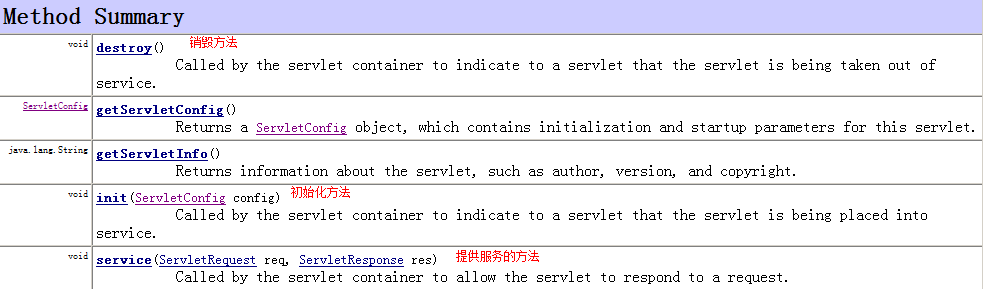
M：Model 模型，封装数据。javabean

V：view 视图，展示界面。jsp

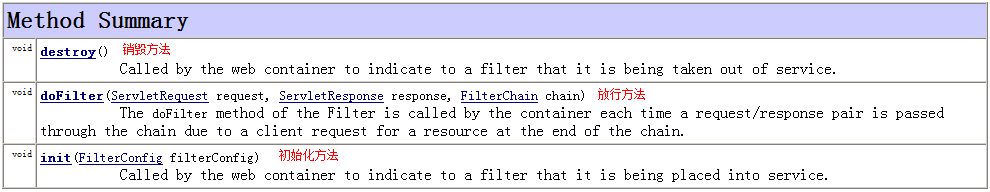
C：Controller 控制器，控制程序流程。Servlet

## 2、Servlet和Filter

Servlet:



Filter：

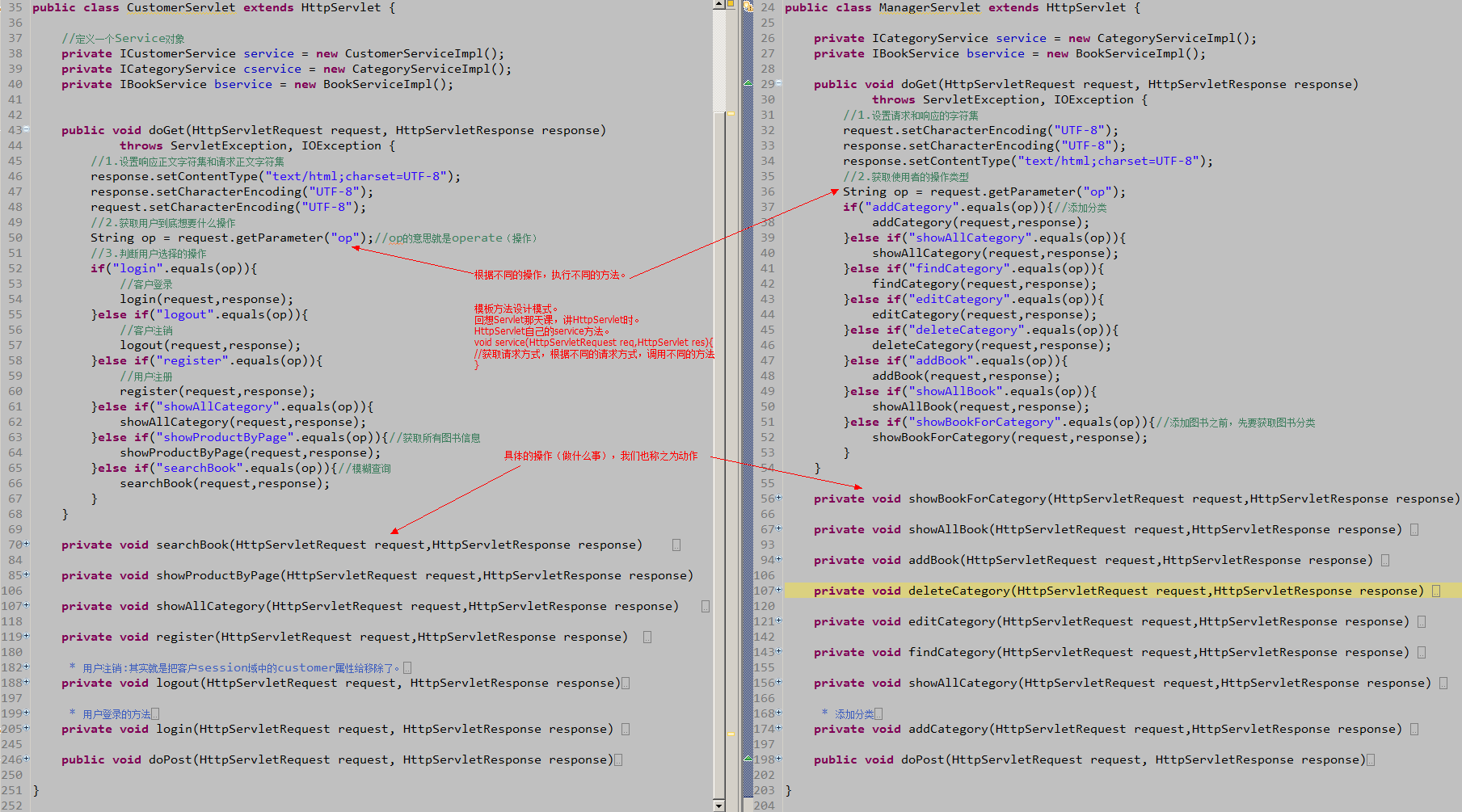


总结：

Servlet能做的事情，过滤器也可以做到。并且过滤器比Servlet还多了一个放行的功能，因此过滤器比Servlet功能更为强大。

结论就是：过滤器同样也适合做控制器。

# 四、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。**

# 五、Struts2简介

## 1、Struts2概述

Struts2是Apache发行的MVC开源框架。注意：它只是表现层（MVC）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。

# 六、搭建Struts2开发环境

## 1、下载Struts2开发包

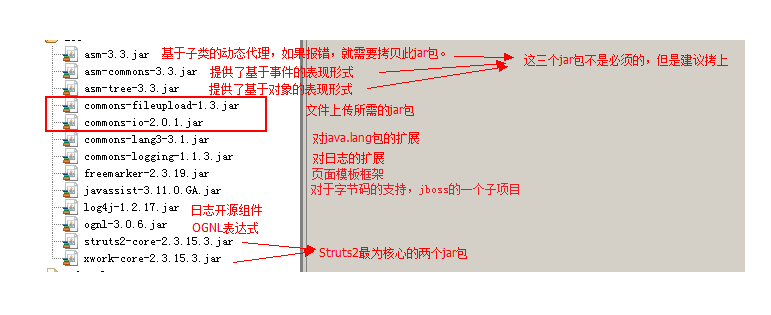
http://struts.apache.org

## 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

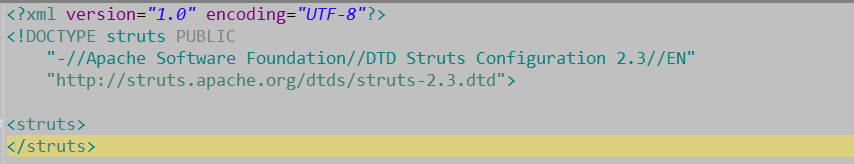


拷贝哪些jar包？

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。

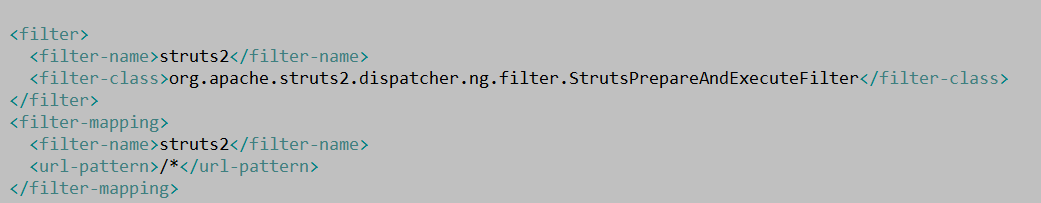
### 3.3、配置控制器

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。

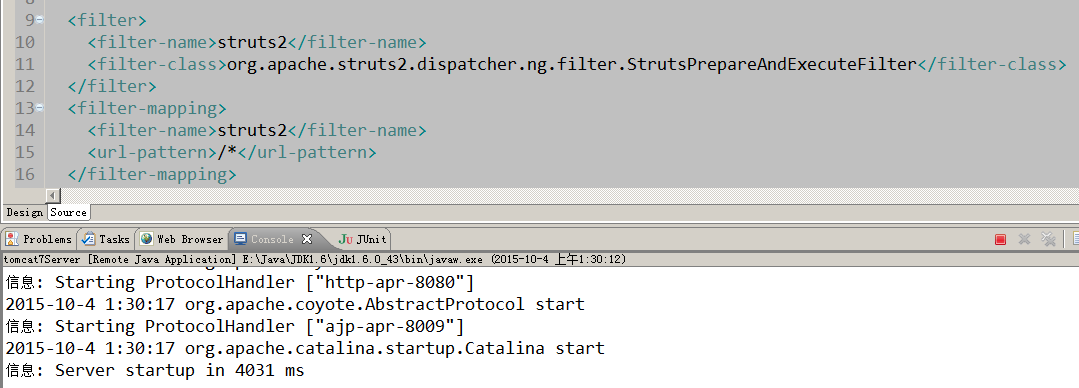
结论：

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。



### 3.4、验证是否成功

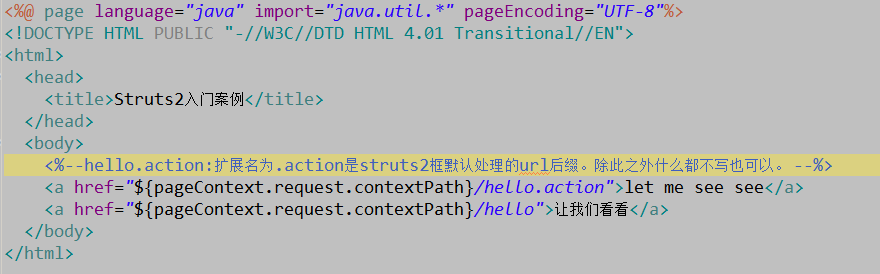
部署应用，启动Tomcat，不报错表示搭建成功。



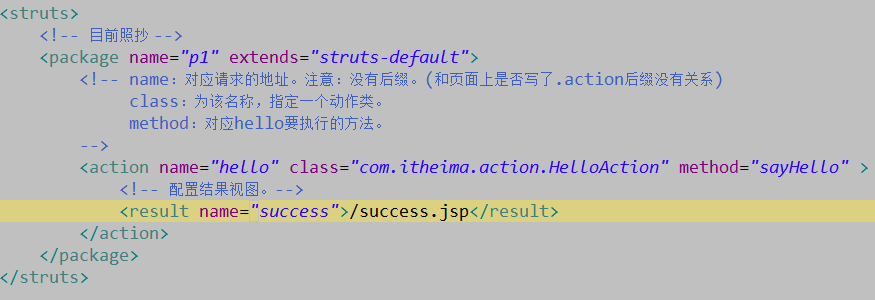
# 七、第一个Struts2案例

以下步骤是日后实际开发中经常重复的。

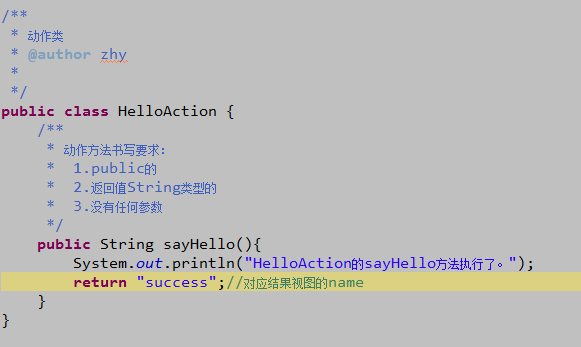
## 1、建立一个jsp文件



## 2、在struts.xml文件中配置



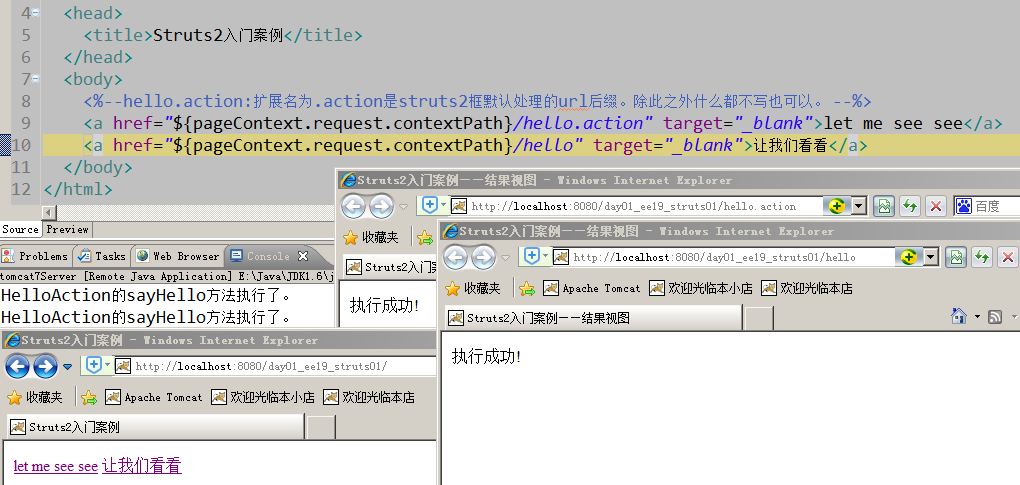
## 3、建立动作类和动作方法



## 4、结果视图页面



## 5、测试：



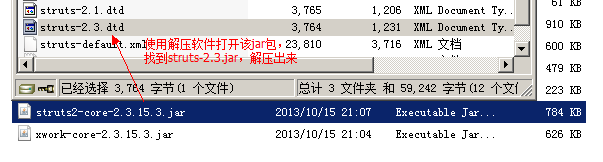
## 6、关于struts.xml没有提示的问题

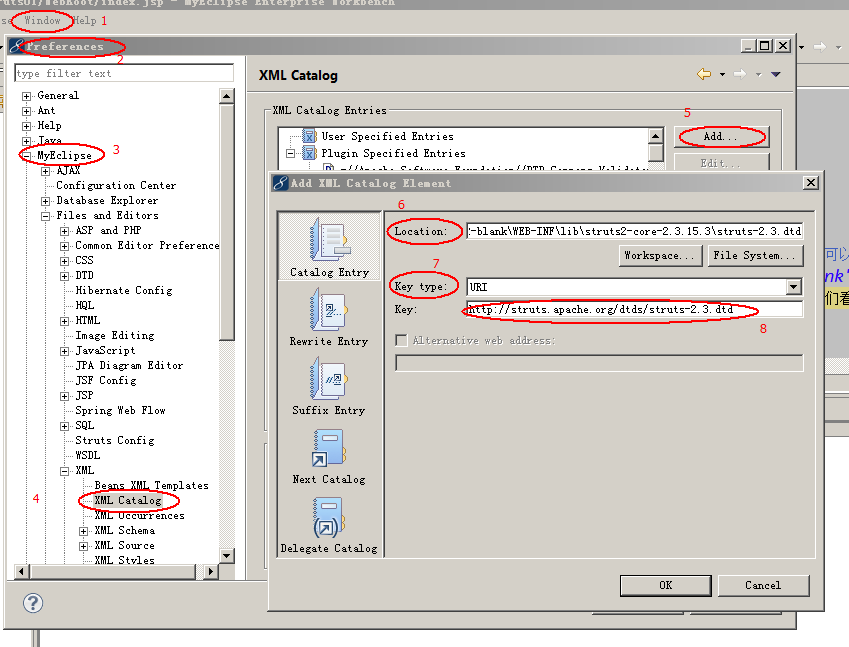
分析原因：没有找到对应的dtd约束文件。

解决办法：

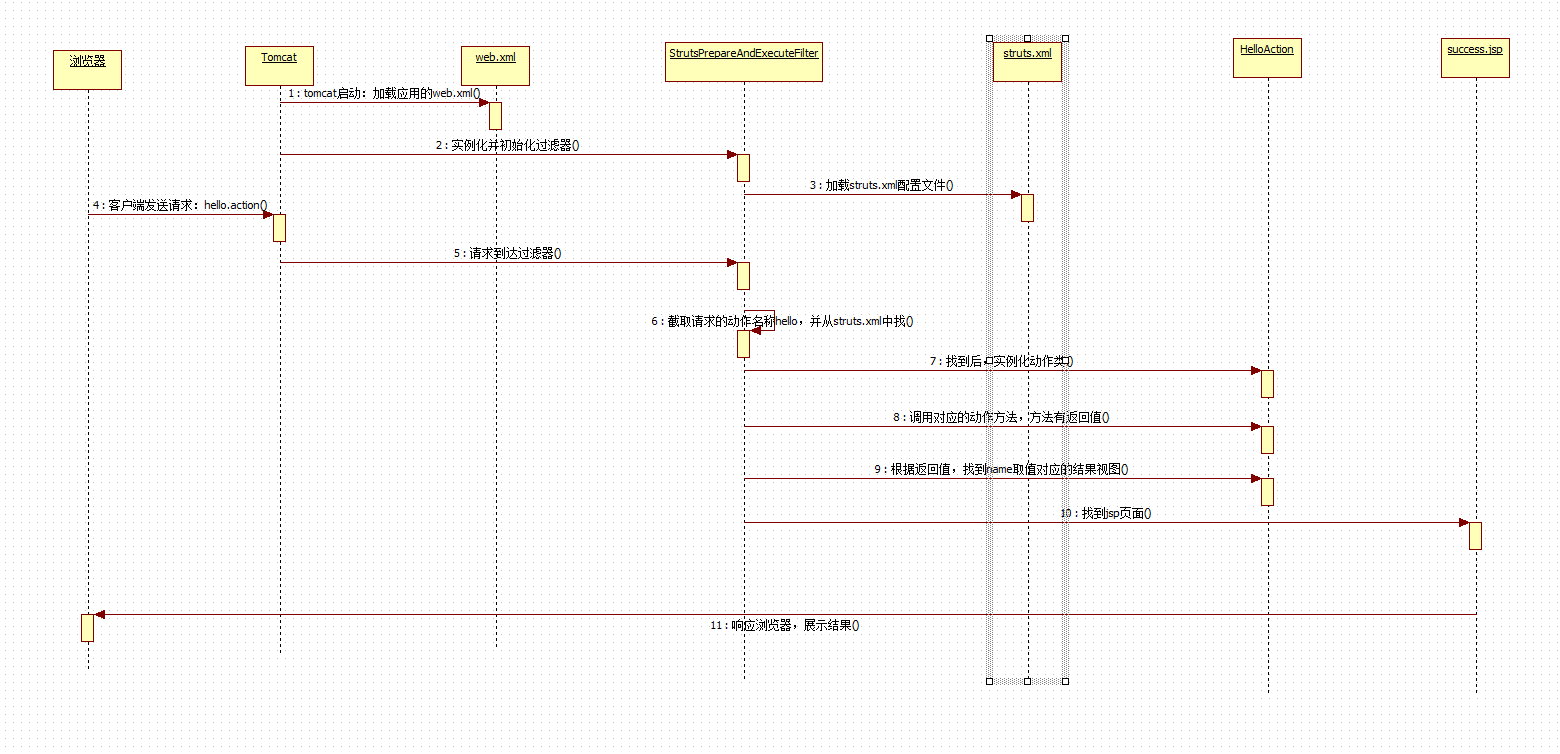
a.上网

b.不能上网：

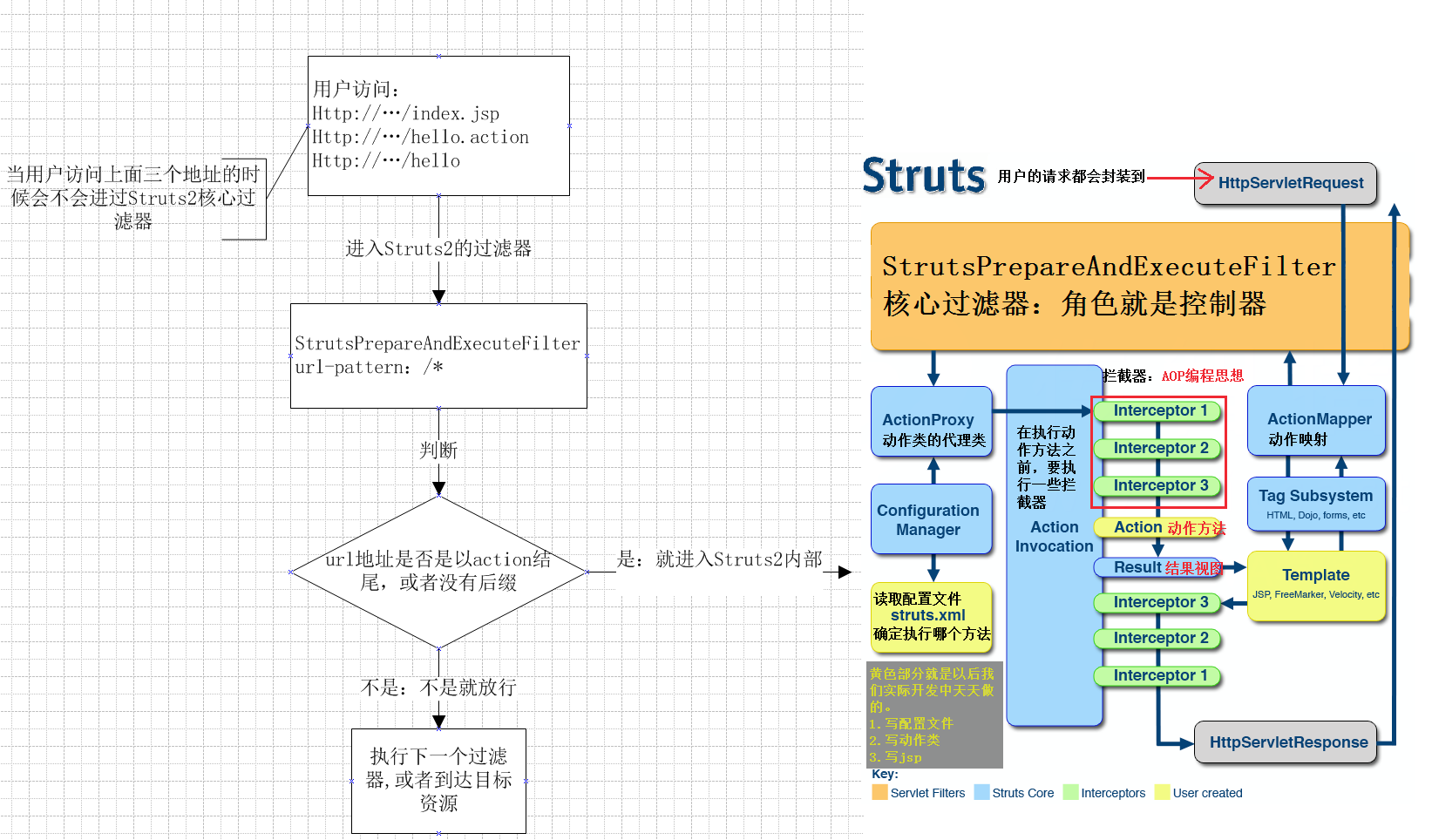




# 八、第一个案例的执行过程



**明确：**

****

# 九、Struts2的配置文件

## 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。

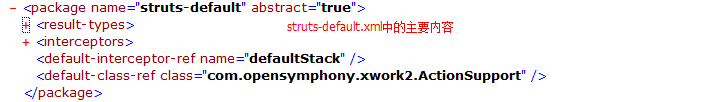
## 2、加载顺序

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
| 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：**

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）。

2、当多个配置文件中，有相同的参数，后面的会把前面的值给覆盖了。



# 十、Struts2框架提供的常量

## 1、常用的常量

常量定义在了default.properties配置文件中，体现形式都是key=value。所有的struts2应用都会用到这些常量。

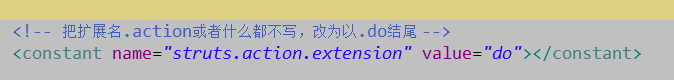
常用的：

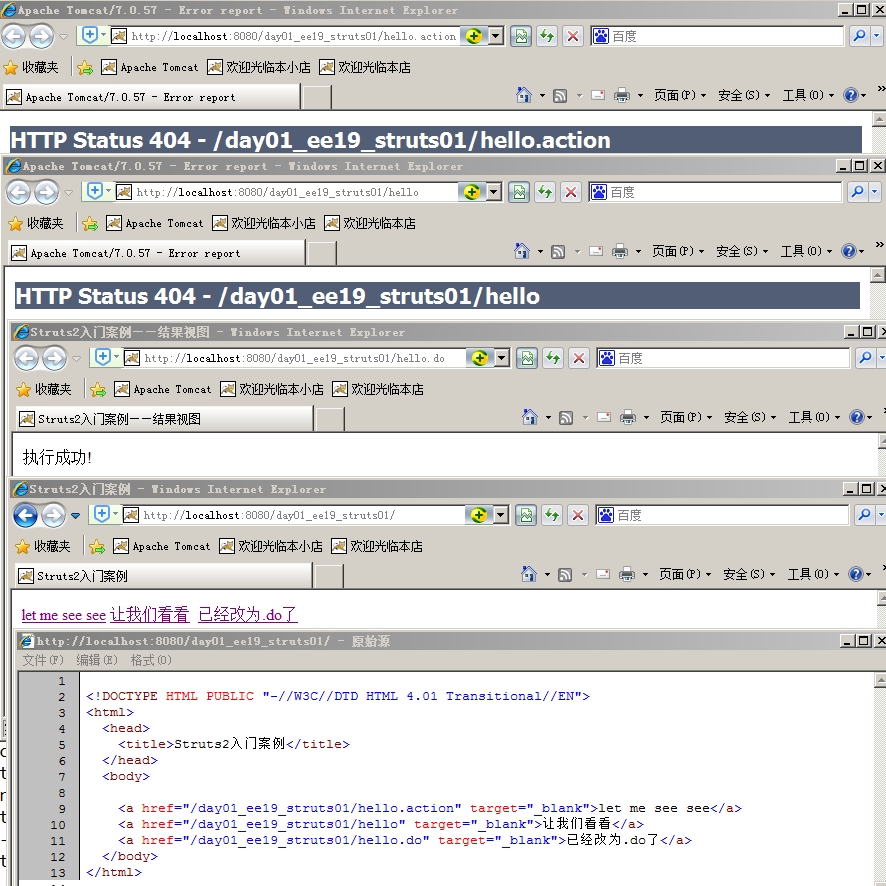
|  |  |  |
| --- | --- | --- |
| 常量名 | 常量值 | 说明 |
| struts.i18n.encoding | UTF-8 | 应用中使用的编码 |
| struts.objectFactory.spring.autoWire | name | 和spring框架整合有关 |
| struts.multipart.parser | jakarta | 指定文件上传用的组件 |
| struts.multipart.maxSize | 2097152 | 文件上传总文件大小限制：2M |
| struts.action.extension | action,, | 能进入Struts2框架内部的url地址后缀名。多个值用逗号分隔 |
| struts.enable.DynamicMethodInvocation | false | 是否允许动态方法调用 |
| struts.devMode | false | 是否是开发模式。开发模式：改了配置文件，不需要重启。输出更多的错误信息。开发阶段建议为true。 |
| struts.ui.theme | xhtml | 页面展示用的主题 |

## 2、在struts.xml中覆盖常量

使用<constant name="" value=""></constant>元素进行覆盖

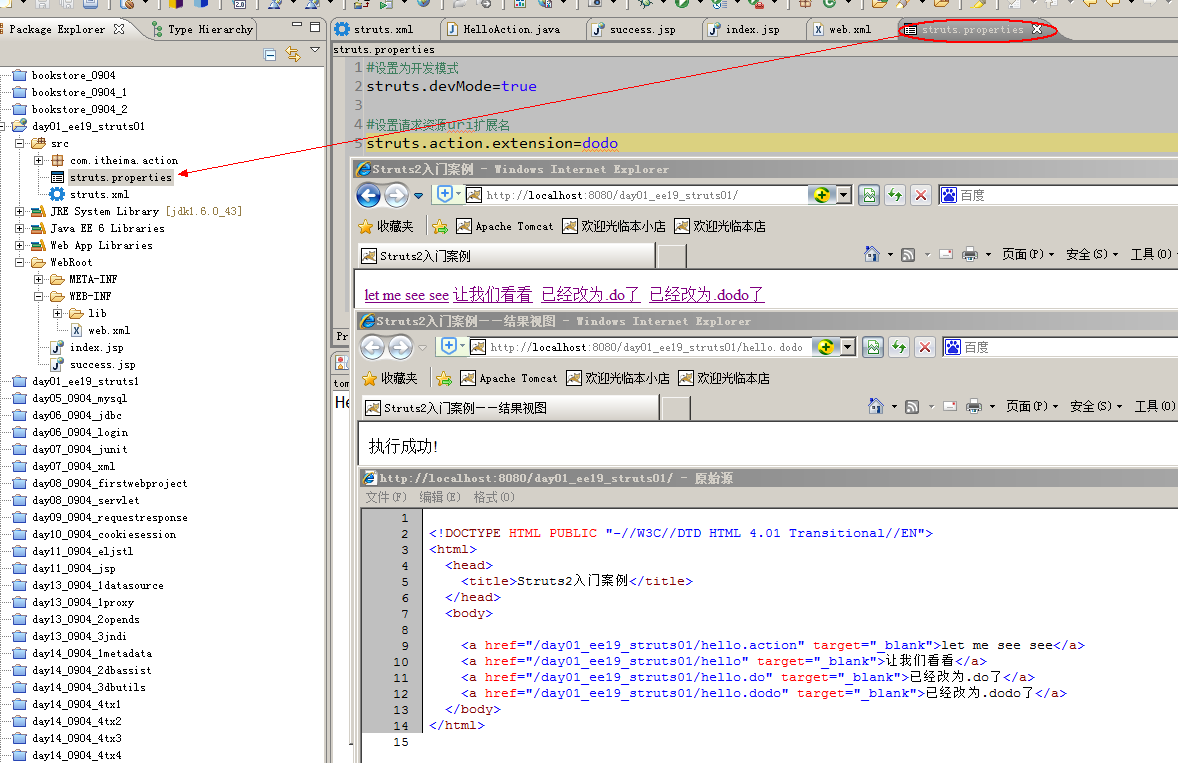
例如：



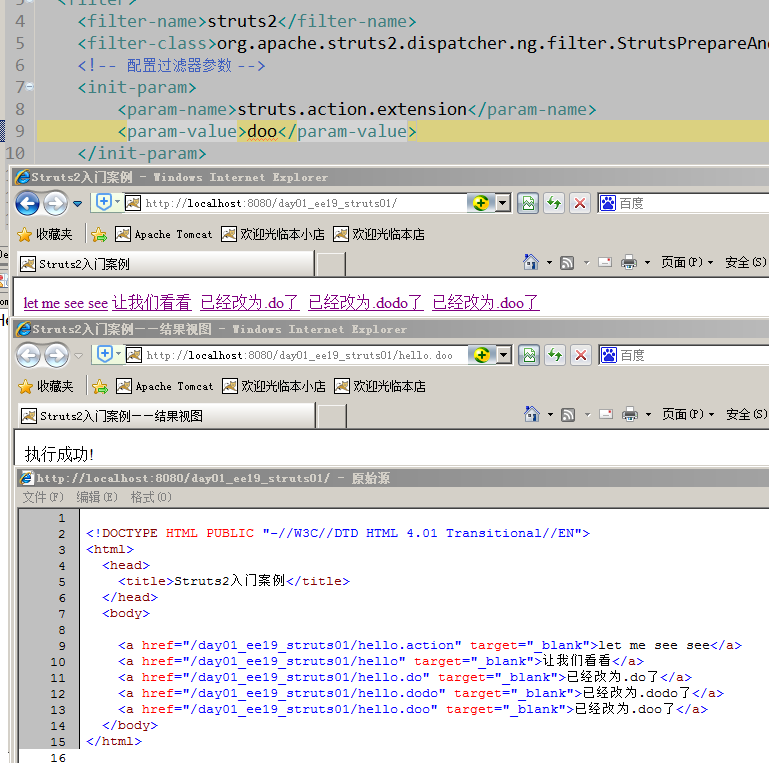


## 3、创建struts.properties文件覆盖

在应用的构建路径中创建一个struts.properties的文件。



## 3、在web.xml中配置过滤器参数



# 十一、xml配置文件的主要元素

## 1、package元素

### 1.1、作用：

在struts2的配置文件中引入了面向对象思想，使用了分包管理。易于管理动作类。便于模块化开发动作类。

### 1.2、属性：

name：包的名称。必须写。且必须唯一。

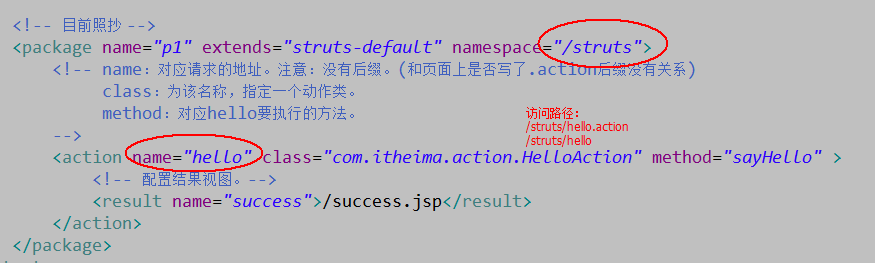
extends：一般情况下需要继承**struts-default**包，但不是必须的。不过如果不继承的话，将无法使用struts2提供的核心功能。

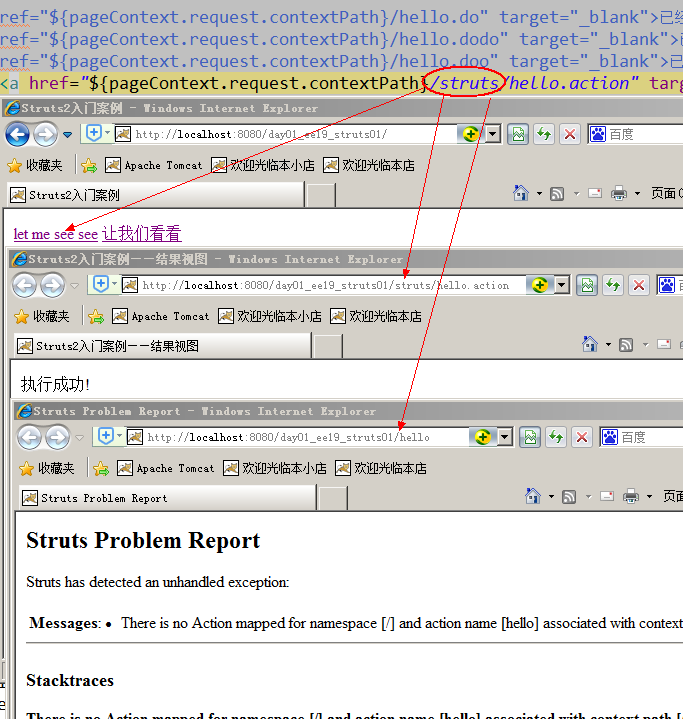
struts-default.xml中定义着struts-default这个包。而struts-default.xml是在我们的struts.xml加载之前加载。

abstract：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。

namespace：名称空间。

名称空间 + 动作名称 = 访问路径





### 1.3、packege中的namespace详解

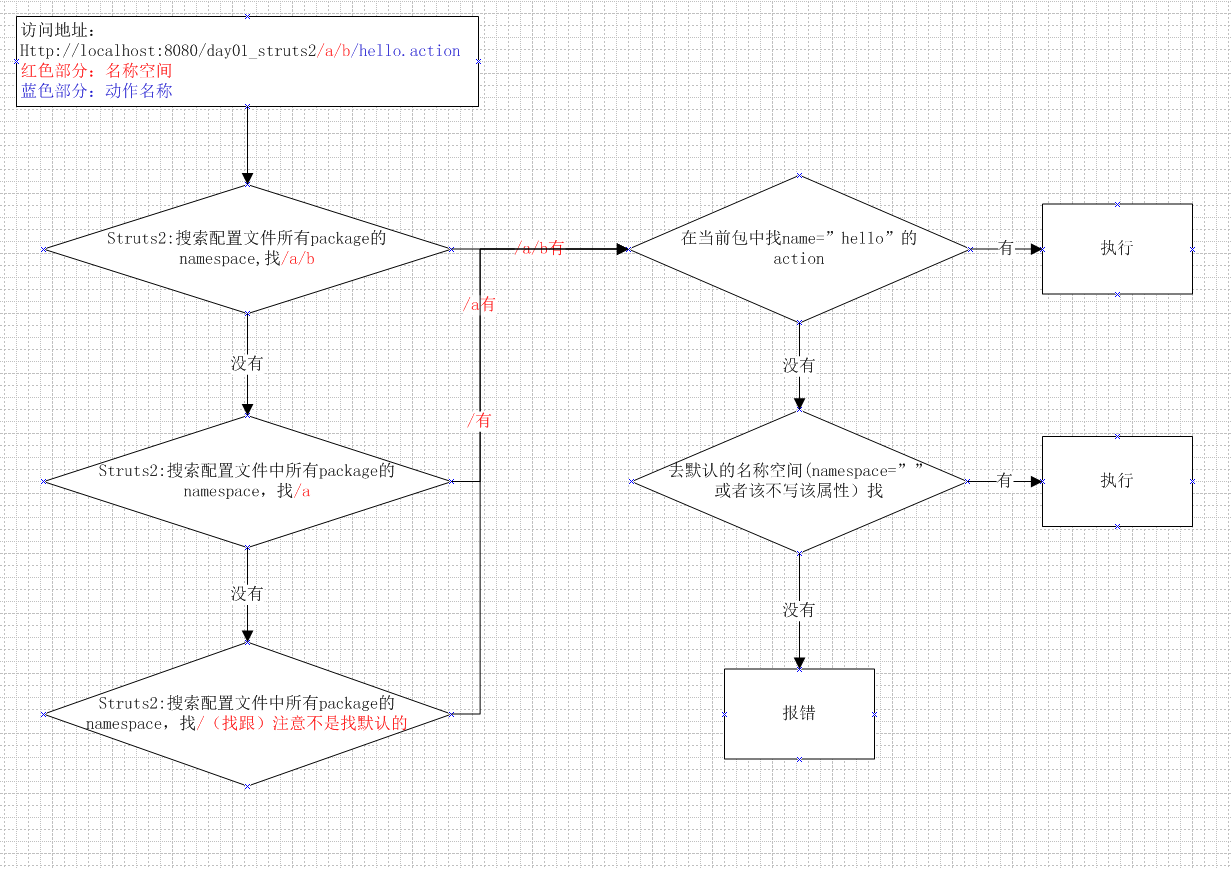
namespace的默认值：

a.不写该属性

b.写了该属性，取值是一个"".

注意：默认值不是/

动作类的搜索顺序：



## 2、action元素

### 2.1、作用：

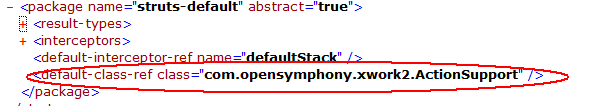
配置动作用的。

### 2.2、属性：

name：动作名称

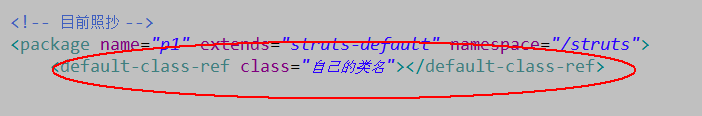
class：动作类全名。默认的动作类是：com.opensymphony.xwork2.ActionSupport

是在struts-default.xml中定义的



要想替换默认动作类：

在应用的struts.xml中，package中加入：



method：动作类中的方法名称。默认是public String execute(){}

要求：

1.public的

2.返回值必须是String

3.没有参数

### 2.3、动作类

a.方式一：动作类就是一个POJO（Plain Old Java Object 原始的java对象），非常简单的javabean。

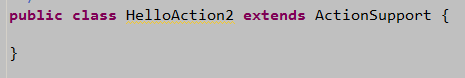


b.方式二：动作类实现com.opensymphony.xwork2.Action接口。

常量：给动作方法返回值用的。用常量可以使你的应用规范和统一。

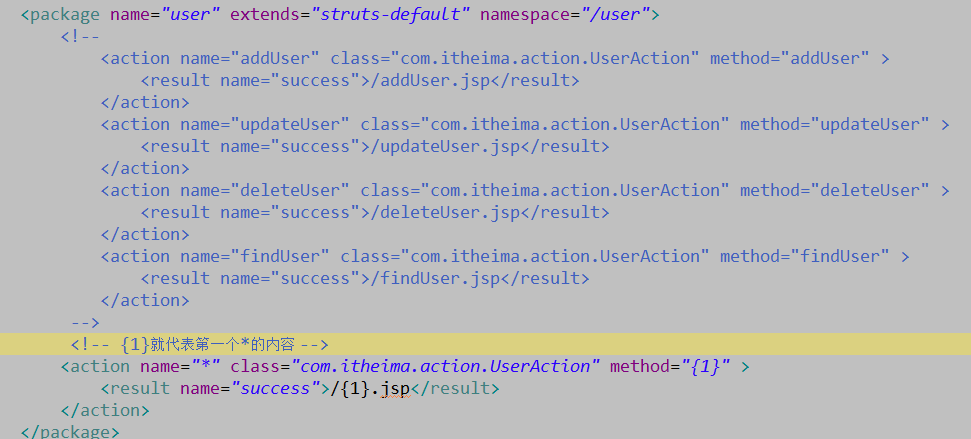
|  |  |  |
| --- | --- | --- |
| 常量变量名 | 对应的值 | 说明 |
| SUCCESS | success | 动作方法执行一切OK |
| ERROR | error | 动作方法执行时遇到了异常 |
| INPUT | input | 回显 |
| LOGIN | login | 一般转向登陆页面 |
| NONE | none | 不转向任何的视图 |

c.方式三：动作类继承com.opensymphony.xwork2.ActionSupport 推荐使用

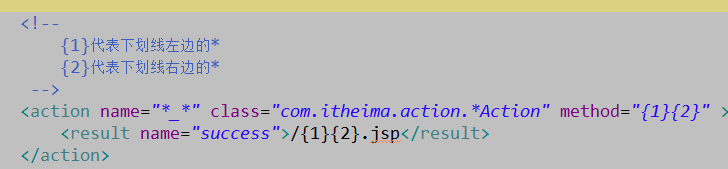


### 2.4、动作的访问

a.使用通配符：

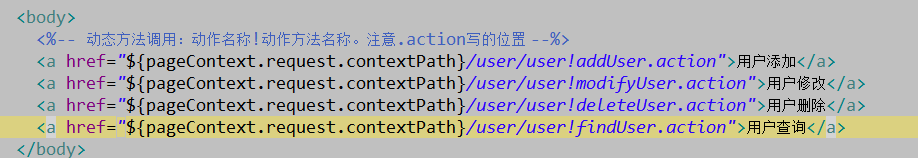


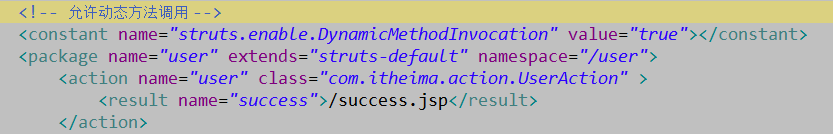
升级版：



优先级：绝对匹配优先。使用通配符的按照在配置文件中的先后顺序进行匹配的。

b.动态方法调用





# 十二、结果类型视图(逻辑结果视图)

前奏：该部分内容指的就是struts配置文件中的result元素的使用

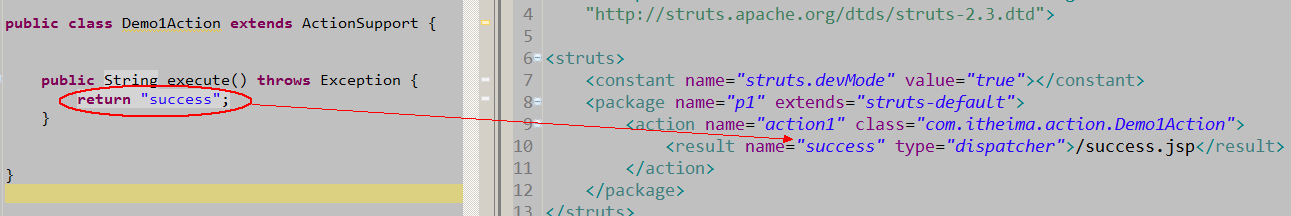
## 1、result元素

作用：

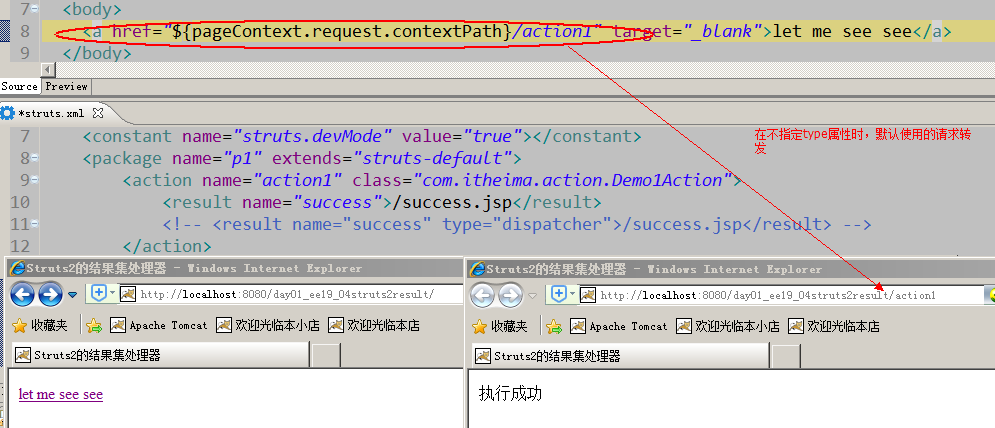
为动作指定结果视图

属性：

name：逻辑视图的名称，对应着动作方法的返回值。默认值是success。

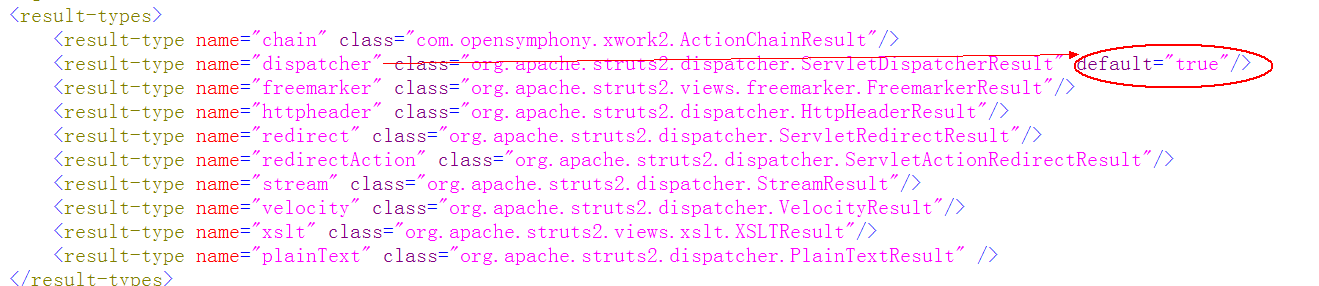


type：结果类型，指的就是用什么方式转到定义的页面。默认是dispatcher。



## 2、result元素中type的取值

type属性的取值在struts-default.xml中定义着。



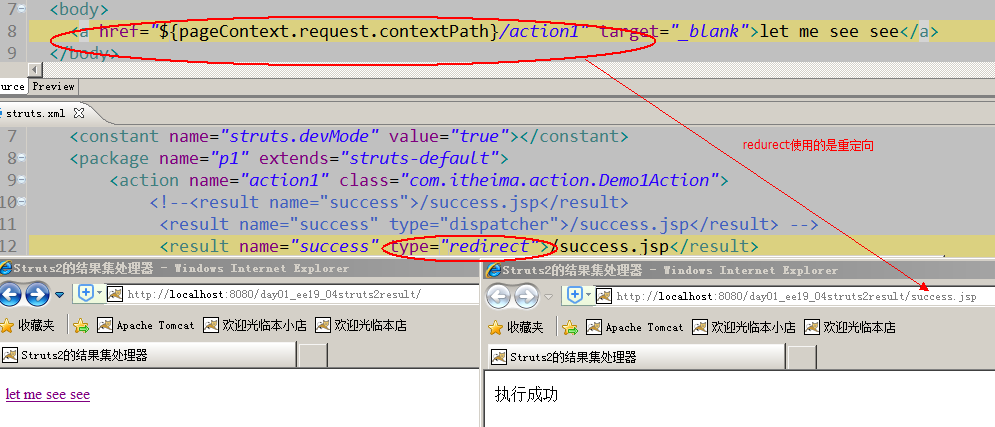
常用结果类型介绍：

dispatcher：(默认值)

使用请求转发，转向一个页面。

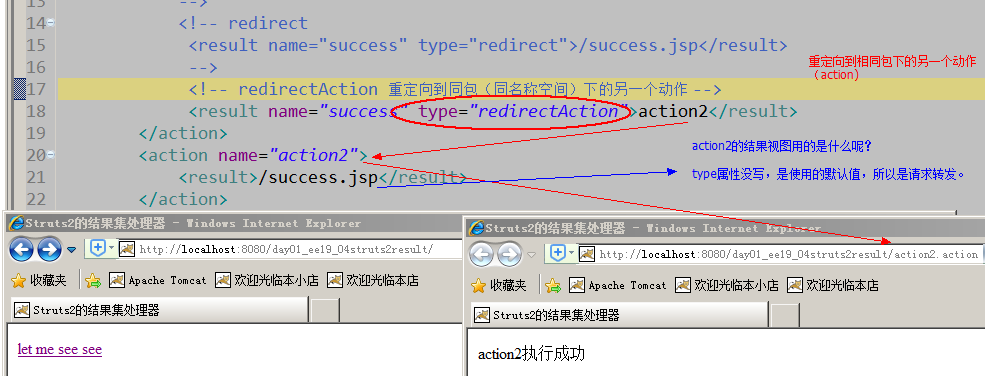
redirect：

使用重定向，转向一个页面。

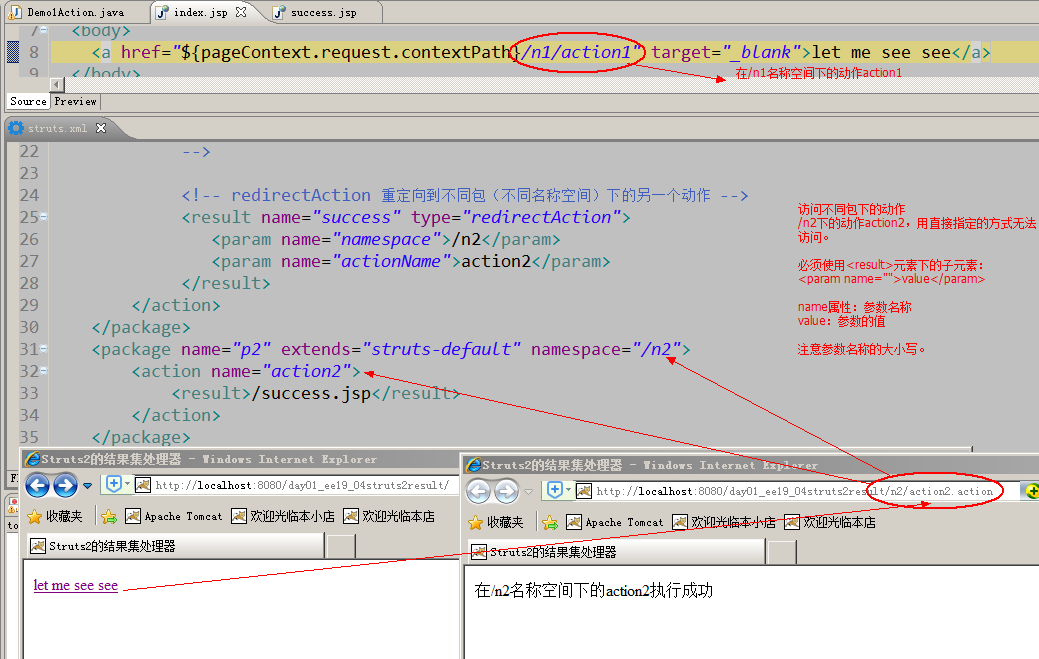


redirectAction：注意：使用的是重定向。

a.重定向到另一个相同名称空间的动作。

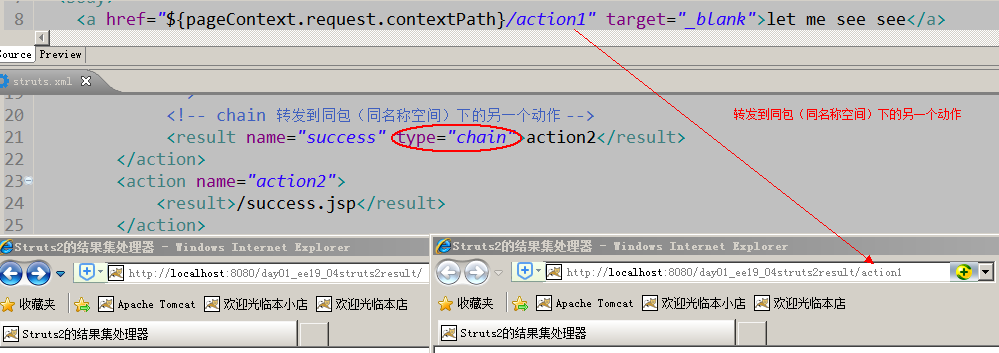


b.重定向到不同名称空间的动作

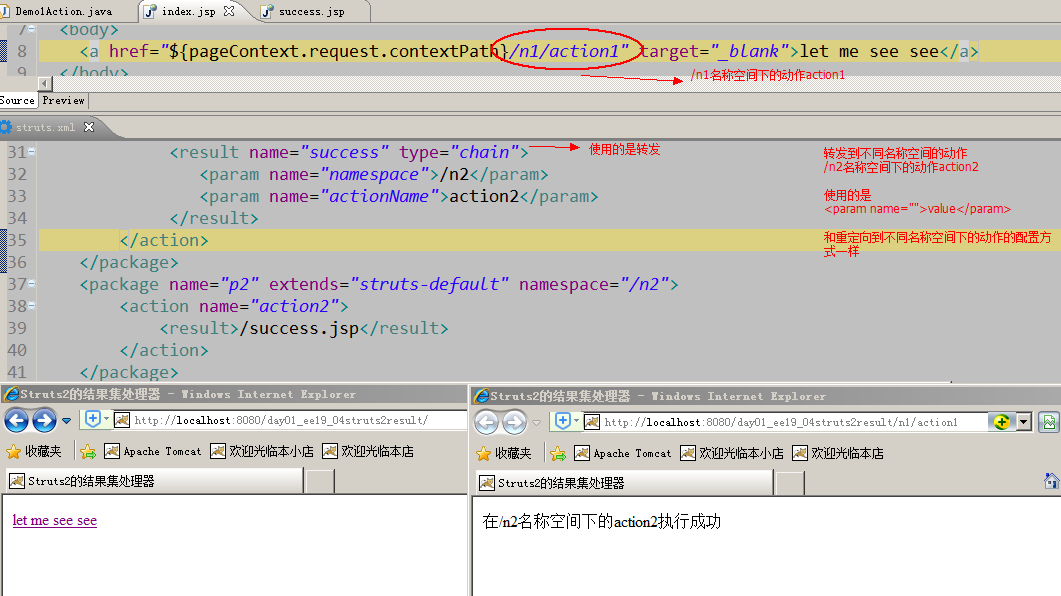


chain：注意： 使用的是请求转发。

a.转发到另一个相同名称空间的动作。



b.请求转发到不同名称空间的动作



## 3、result元素中param子元素

在转发或者重定向到不同包下的动作时，都用到了result元素的子元素param。

param元素的作用：依赖注入(Dependence Injection)思想

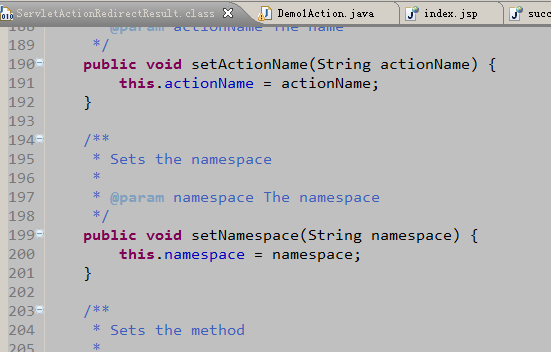
我们通过struts-default.xml中的resultTypes元素中配置可以看出，每个结果类型视图其实都是靠一个类来实现的。而param元素就是将配置的参数，注入到该类中。

调用的是对应类的setter方法进行注入的。

例如：redirectAction结果视图

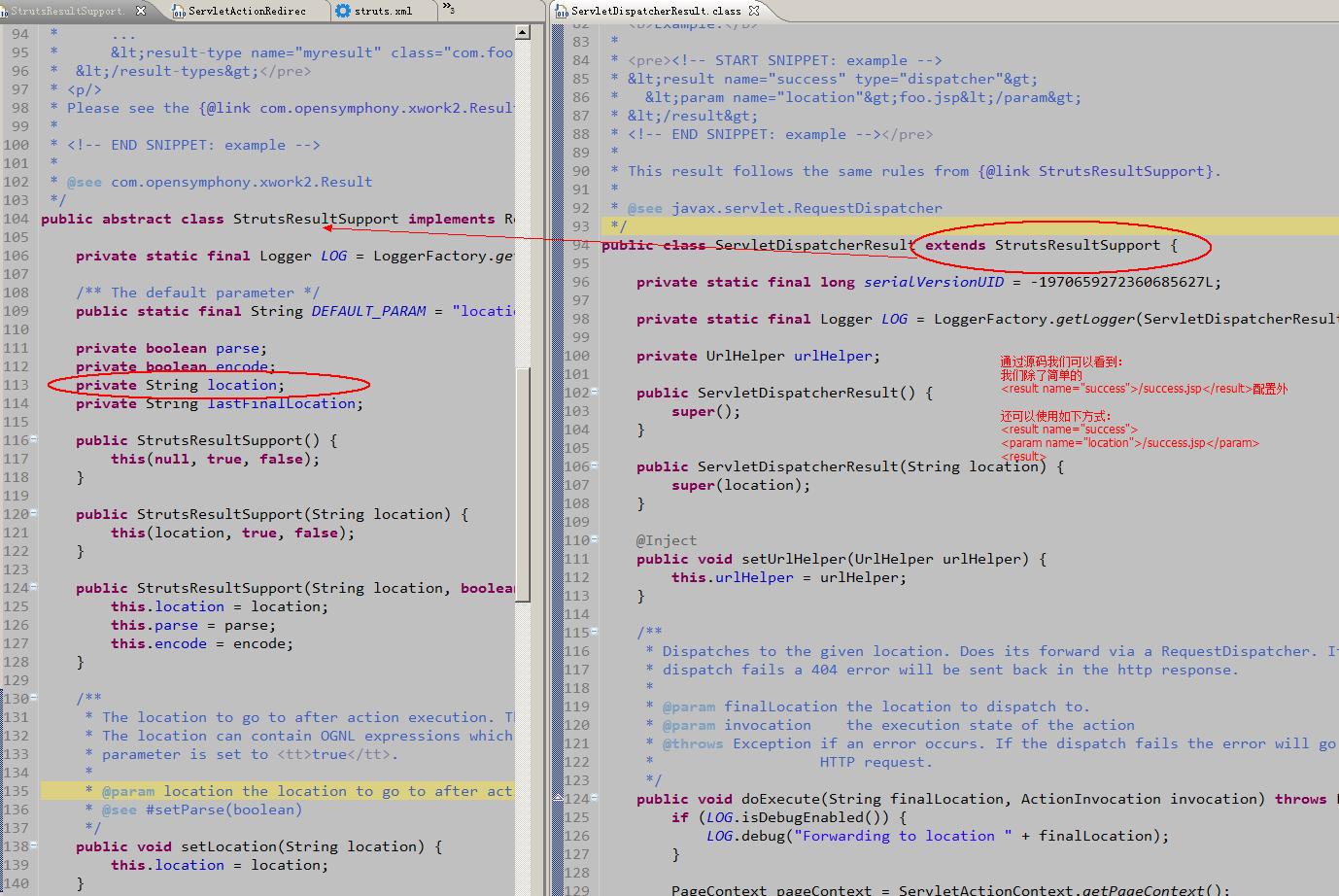


该类中肯定会有对actionName和namespace属性的注入方法（setter方法）。



再比如：默认结果视图dispatcher





## 4、自定义结果类型

通过前面的内容，我们看出，其实结果类型就是一个类，这些类都实现了com.opensymphony.xwork2.Result接口。

或者继承自该接口的实现类org.apache.struts2.dispatcher.StrutsResultSupport。

这些类都有一个doExecute方法，用于执行结果视图。

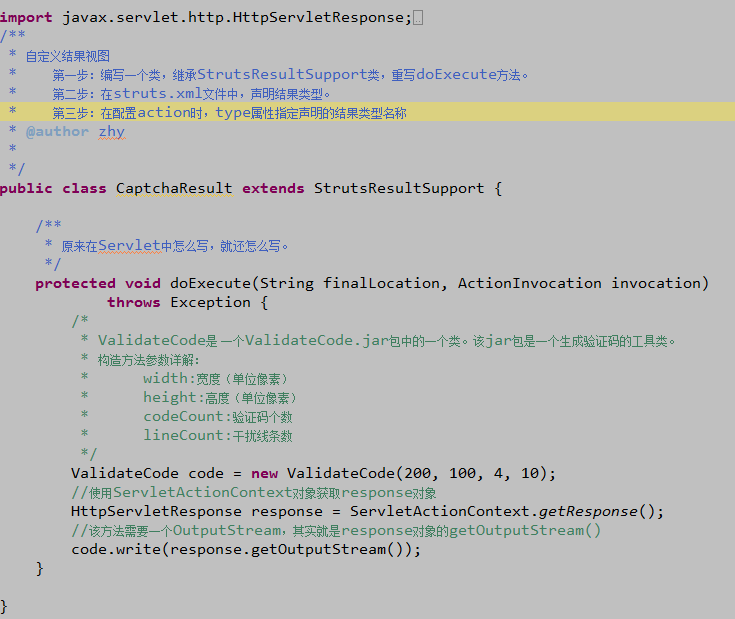
综上：我们也可以自己写一个结果视图。

例子：

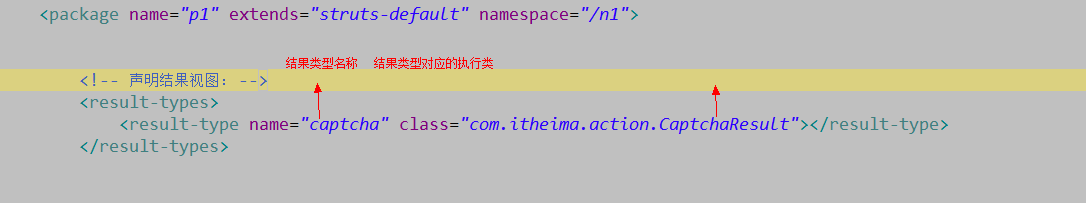
输出CAPTCHA图像的结果类型。

CAPTCHA(Completely Automated Public Turing Test to Tell Computers and Humans Apart 全自动区分计算机和人类的图灵测试)————>简称：验证码。

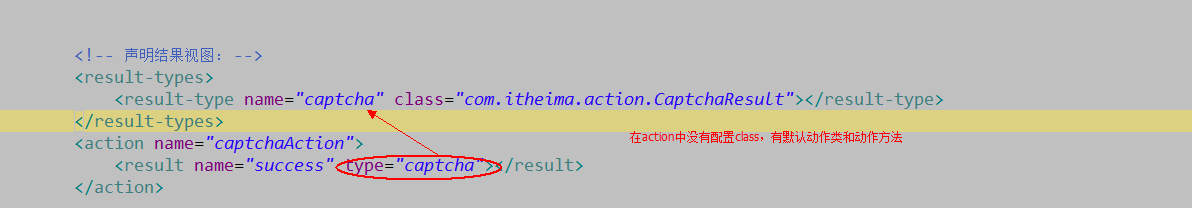
第一步：写一个类，实现接口或者继承接口的实现类



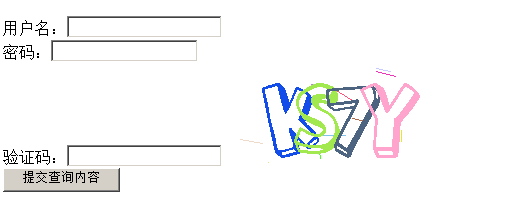
第二步：在struts.xml文件中配置结果类型



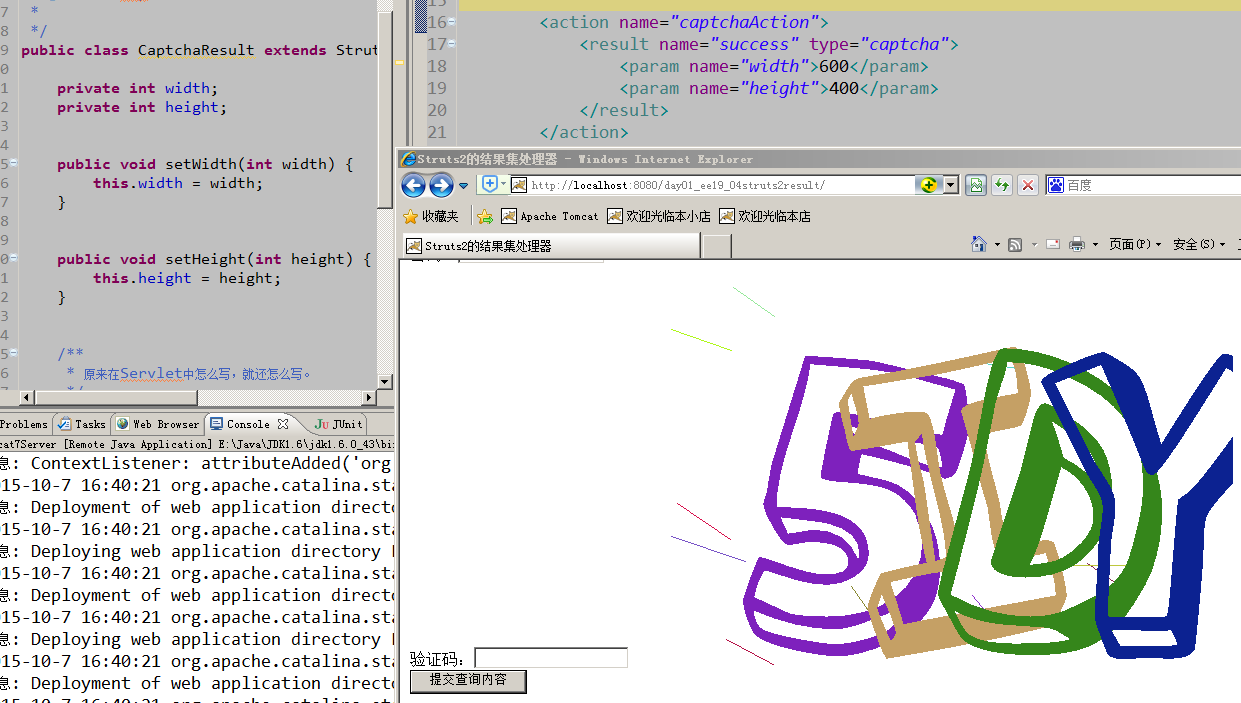
第三步：在action配置时引用



最终结果：

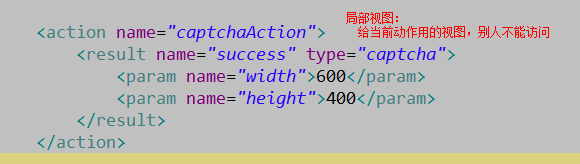


扩展：通过可配置的参数，实现图像大小的调整

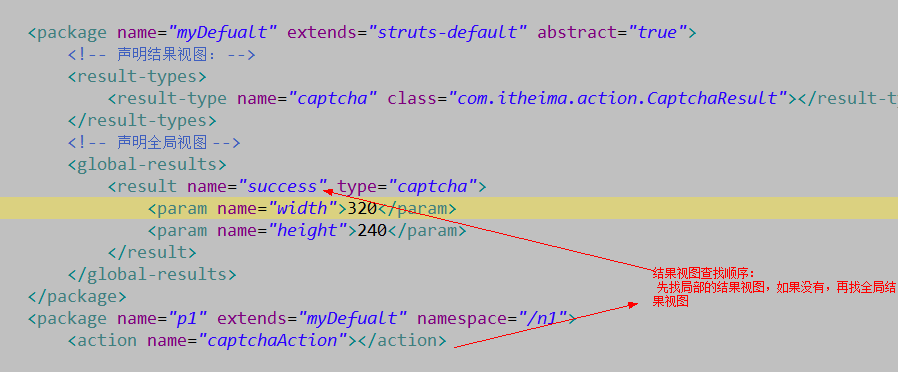


## 5、全局视图和局部视图

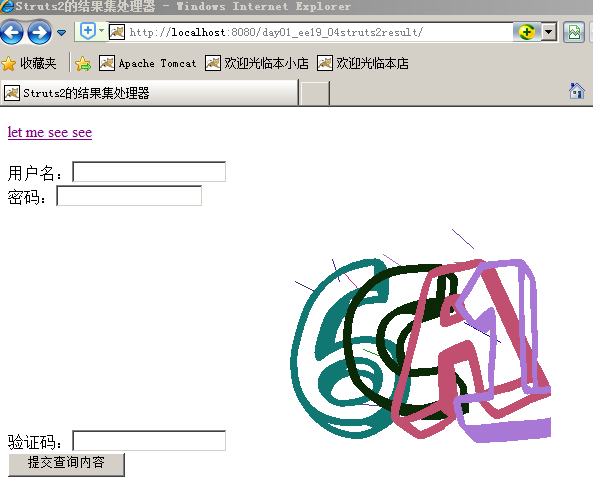
### 5.1、局部视图



### 5.2、全局视图

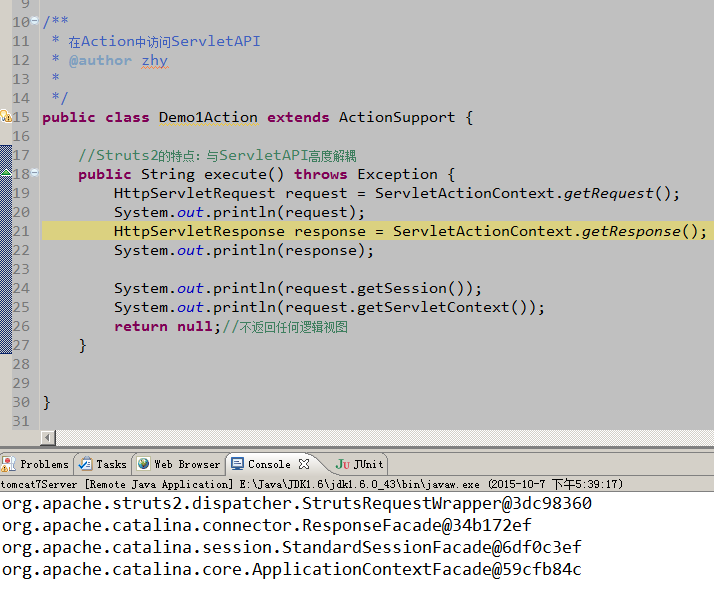


运行结果：

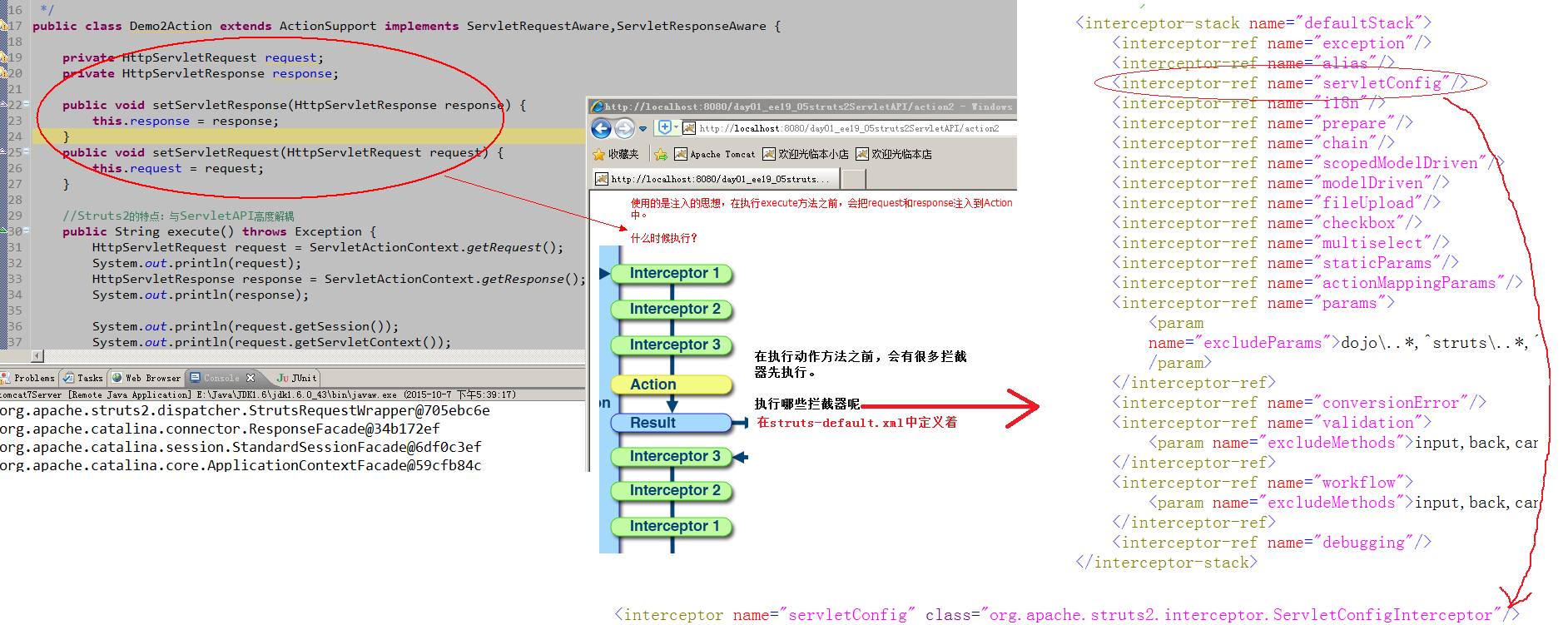


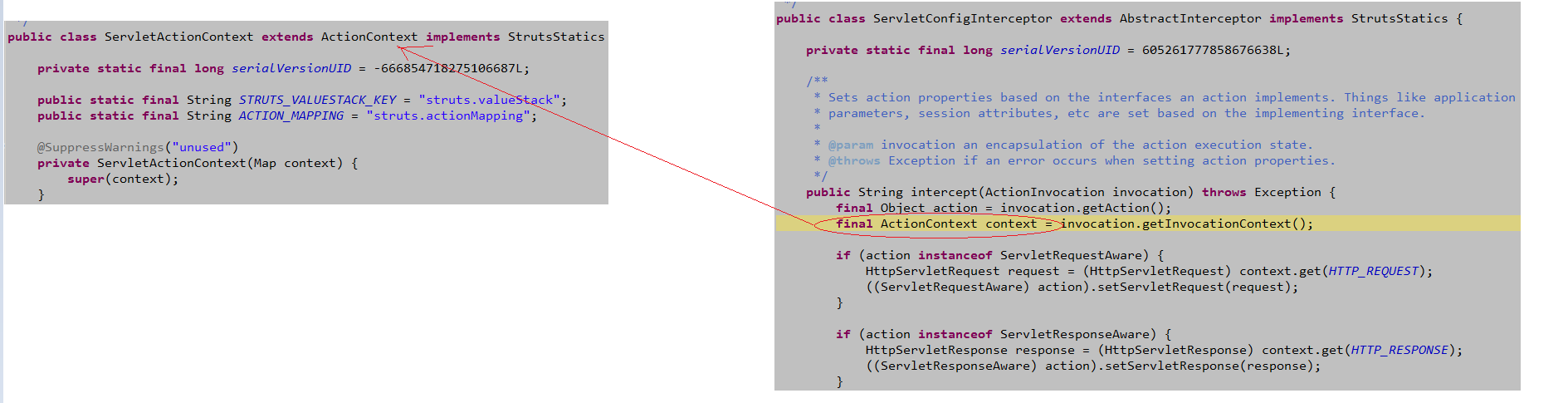
# 十三、在动作类中访问Servlet的API

第一种方式：使用ServletActionContext类



第二种方式：使用实现接口的方式



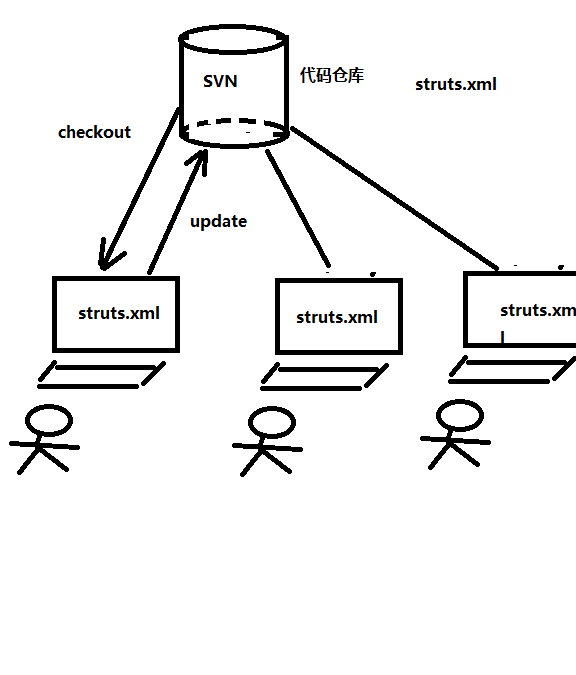


# 十四、分文件编写框架配置文件

## 1、不分文件开发可能产生的问题

就类似于我们在写java类时，所有代码都写在一个类里，甚至写在一个方法里。



当3个人都checkout了struts.xml文件时，第一个人提交了，后面的人在没有更新就提交时，第一个人写的可能就白写了。

## 2、分文件编写Struts2的配置文件

