# 一、国际化概念（了解）

## 1、什么是国际化

软件的国际化：软件开发时，要使它能同时应对世界不同地区和国家的访问，并针对不同地区和国家的访问，提供相应的、符合来访者阅读习惯的页面或数据。

## 2、什么需要国际化

程序：需要国际化。

数据：是什么样的就是什么样的。

比如：

用户注册的表单，有用户名，密码这5个汉字，在zh\_CN语言环境，显示的就是用户名和密码。但是在en\_US语言环境，显示的就应该是username和password。这就是程序。

用户名输入的是【张三】，密码输入的是【test】，那无论在什么语言环境都应该是是【张三】和【test】。这就是数据。

## 3、固定文本的国际化

例如：消息提示，错误提示和菜单，导航栏等等固定文本。

步骤：

### 3.1、创建一个消息资源包

一个资源包由多个文件组成，这些文件名都有命名规范：**主要文件名\_语言代码\_国家代码.properties**。 语言代码：由iso规定的。国家代码：有iso规定的

当文件只有**主要文件名.properties**时，表明它是默认资源包。浏览器会根据不同的语言环境找对应语言环境的资源包，当没有时，找默认的。

每个资源包的内容都由相同的key和对应语言环境的value组成。

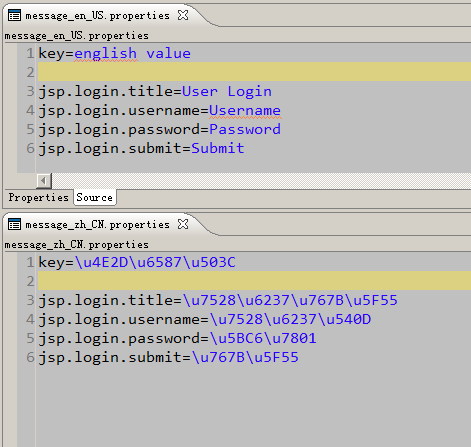
比如：

message\_zh\_CN.properties message\_zh\_HK.properties message\_en\_US.properties

### 3.2、读取资源包中的内容



**jsp中使用国际化：**

** **

**使用jstl的fmt标签：**

****

# 二、Struts2中的国际化（了解）

## 1、Struts2中使用国际化的前提

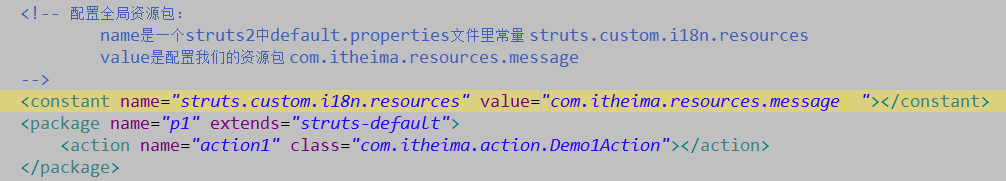
首先，我们要知道，在Struts2中，所有的消息提示都是基于国际化的。

其次，要想在Struts2中使用国际化，动作类必须继承ActionSupport类。

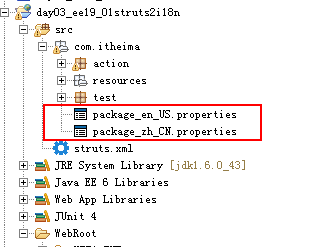
## 2、Struts2中使用国际化

### 2.1、配置资源包

**a、配置全局资源包**

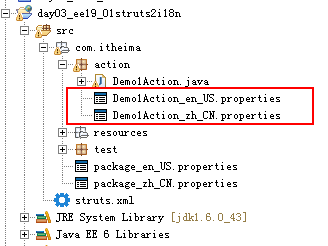
****

**b、配置包范围的资源包**

****资源包名称命名规范：**package\_语言代码\_国家代码.properties(固定的)**。以此种命名方式的资源包能被该包及其子包中的动作类访问。

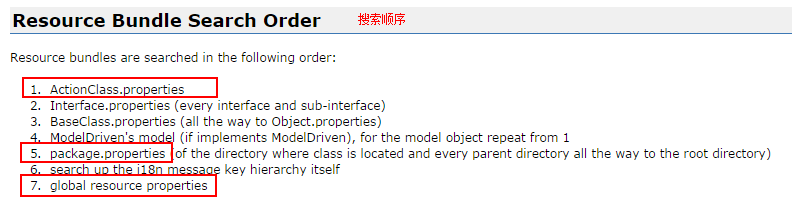
优先级：高于全局消息资源包

**c、局部消息资源包(只为动作类来使用的)**

****资源包名称命名规范：**动作类名称\_语言代码\_国家代码.properties**。以此种命名方式的资源包，只为动作类服务。

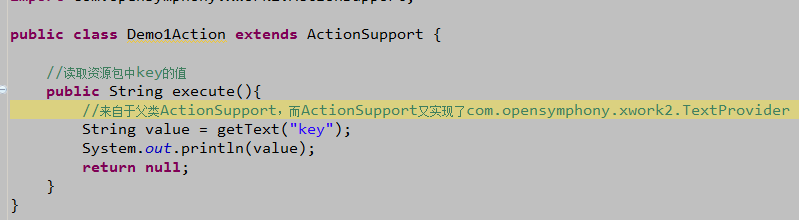
优先级最高（就近原则）。

Struts2中资源包的搜索顺序：



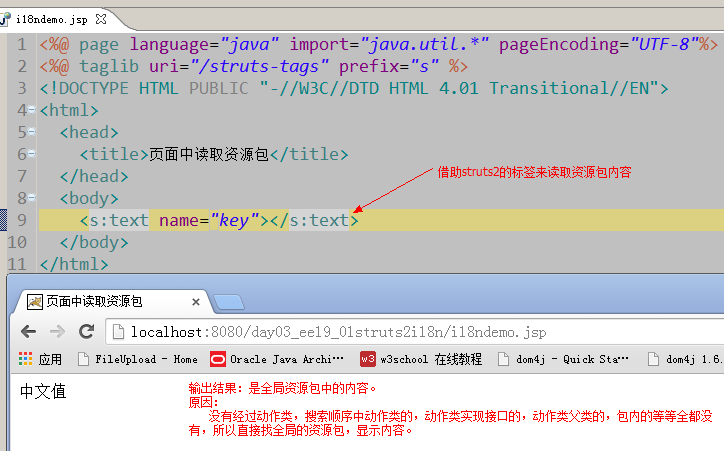
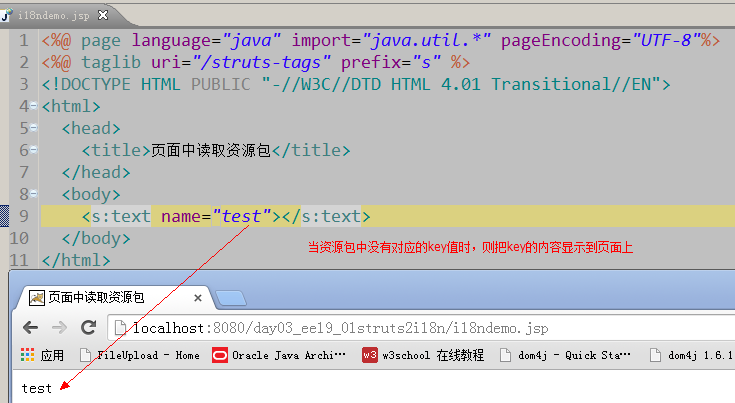
### 2.2、读取资源包的内容

**a、动作类中的读取方式（实际开发中几乎从来不用）**

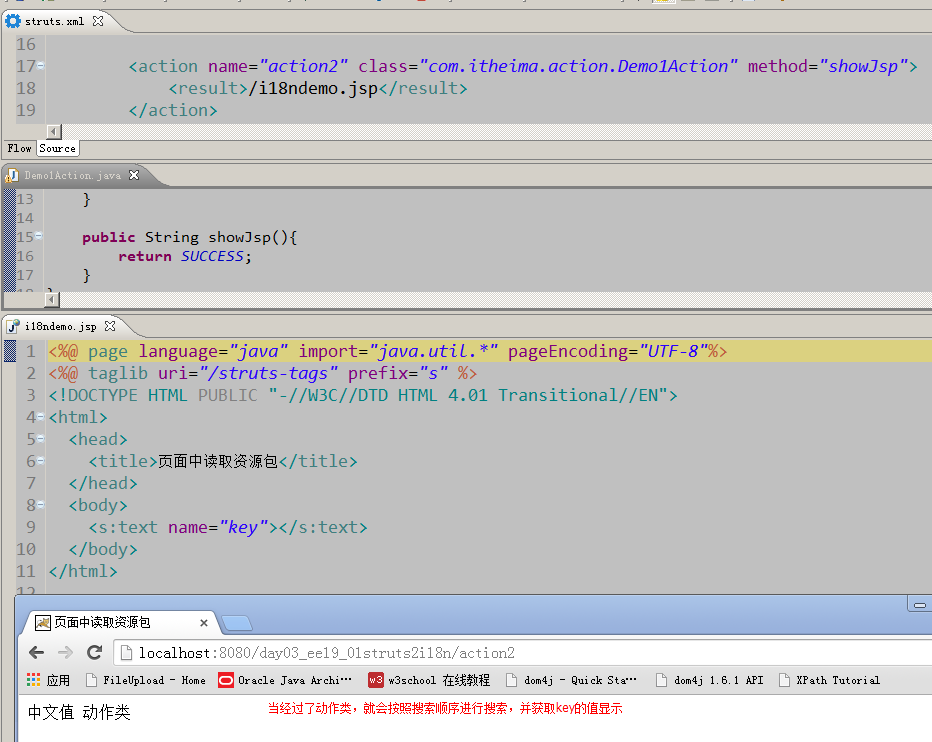
****

**b、在页面中读取资源包内容**

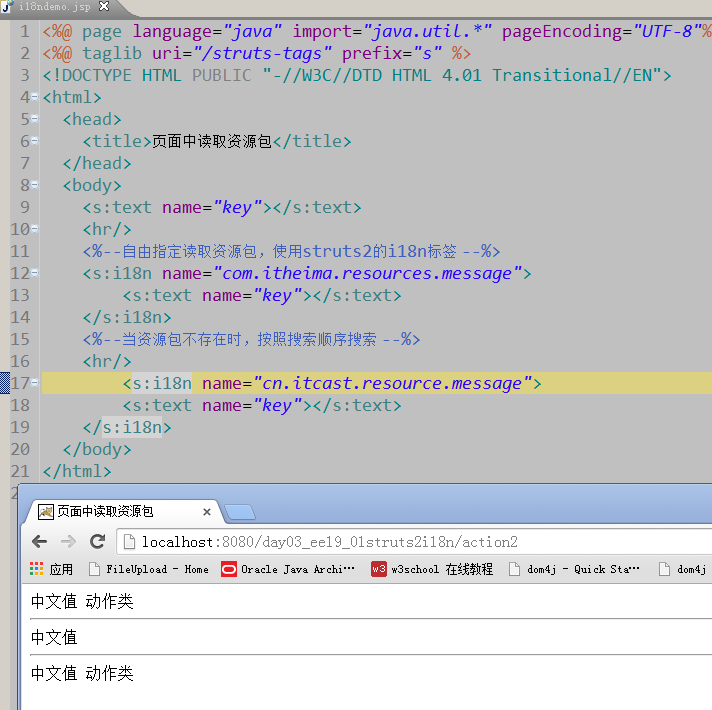
**直接访问jsp：**

** **

**通过动作类访问jsp**

****

**c、自由指定读取资源包**

****

**以下内容全都是重点**

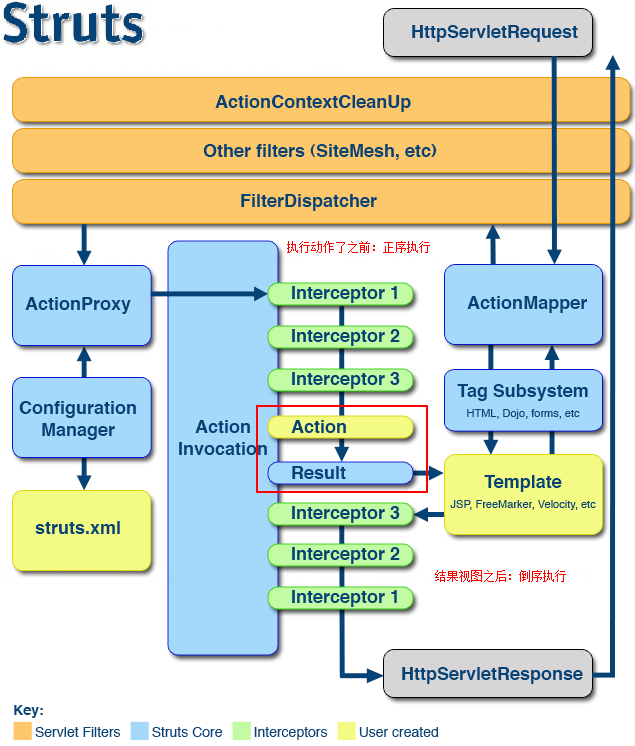
# 三、Struts2中的拦截器（特别重要）

## 1、拦截器的重要性

Struts2中的很多功能都是由拦截器完成的。比如：servletConfig，staticParam，params，modelDriven等等。

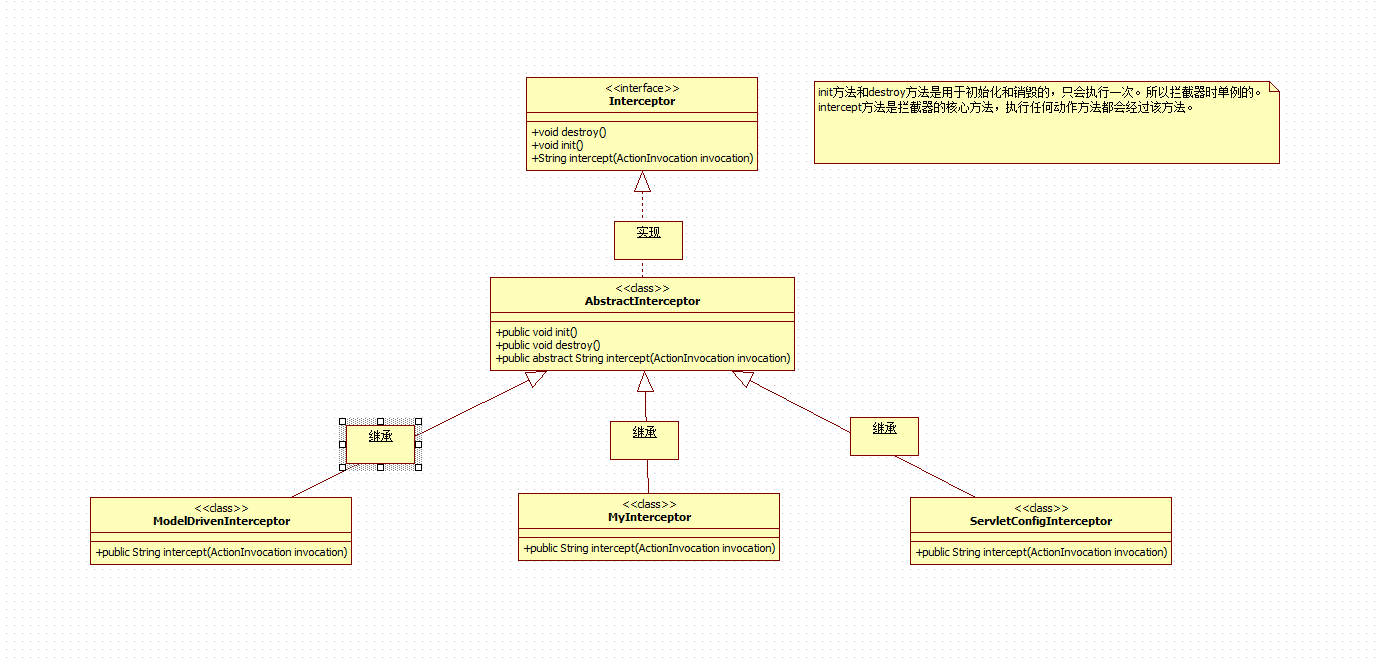
是**AOP编程思想**的一种应用形式。

## 2、拦截器的执行时机：

****

## 3、自定义拦截器

### 3.1、拦截器的类试图(初级版本)：

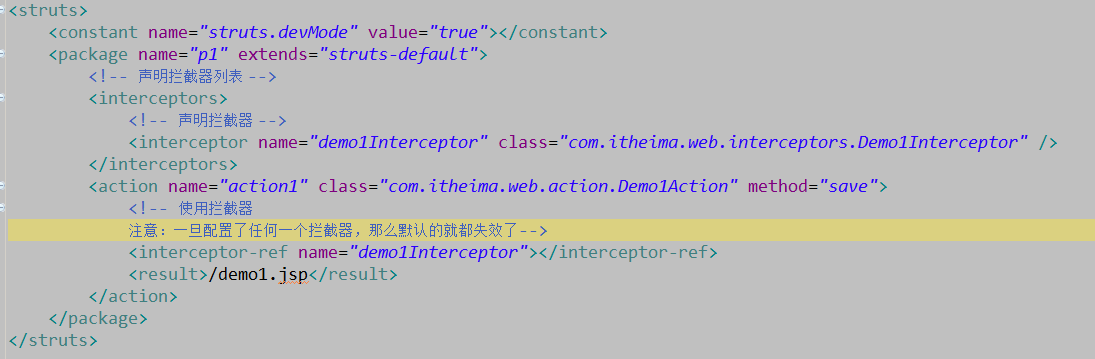


### 3.2、编写步骤：

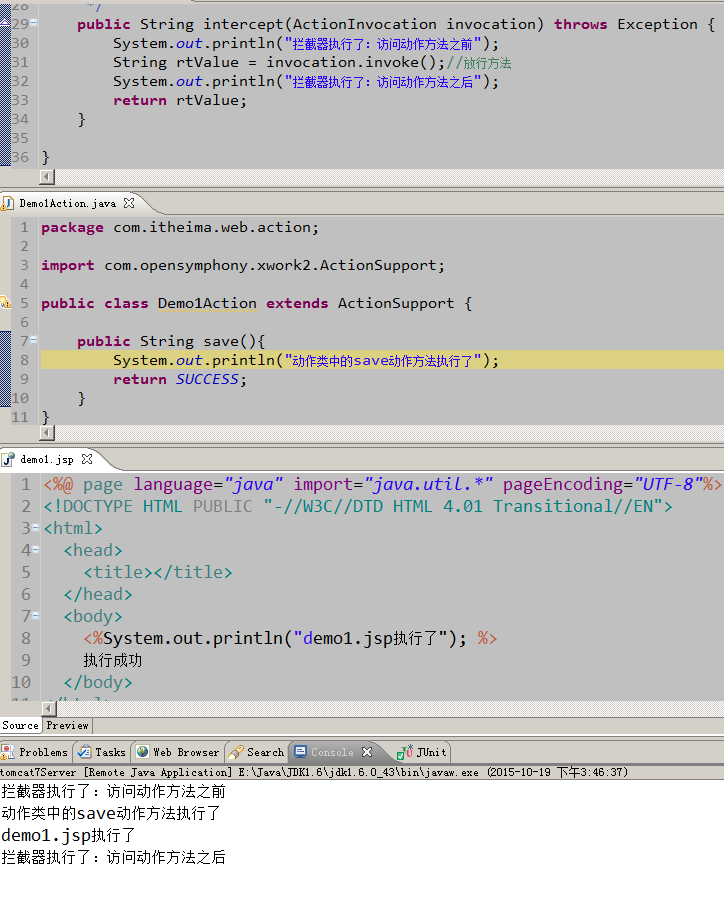
a、编写一个类，继承AbstractInterceptor类或者实现Interceptor接口。重写intercept方法。



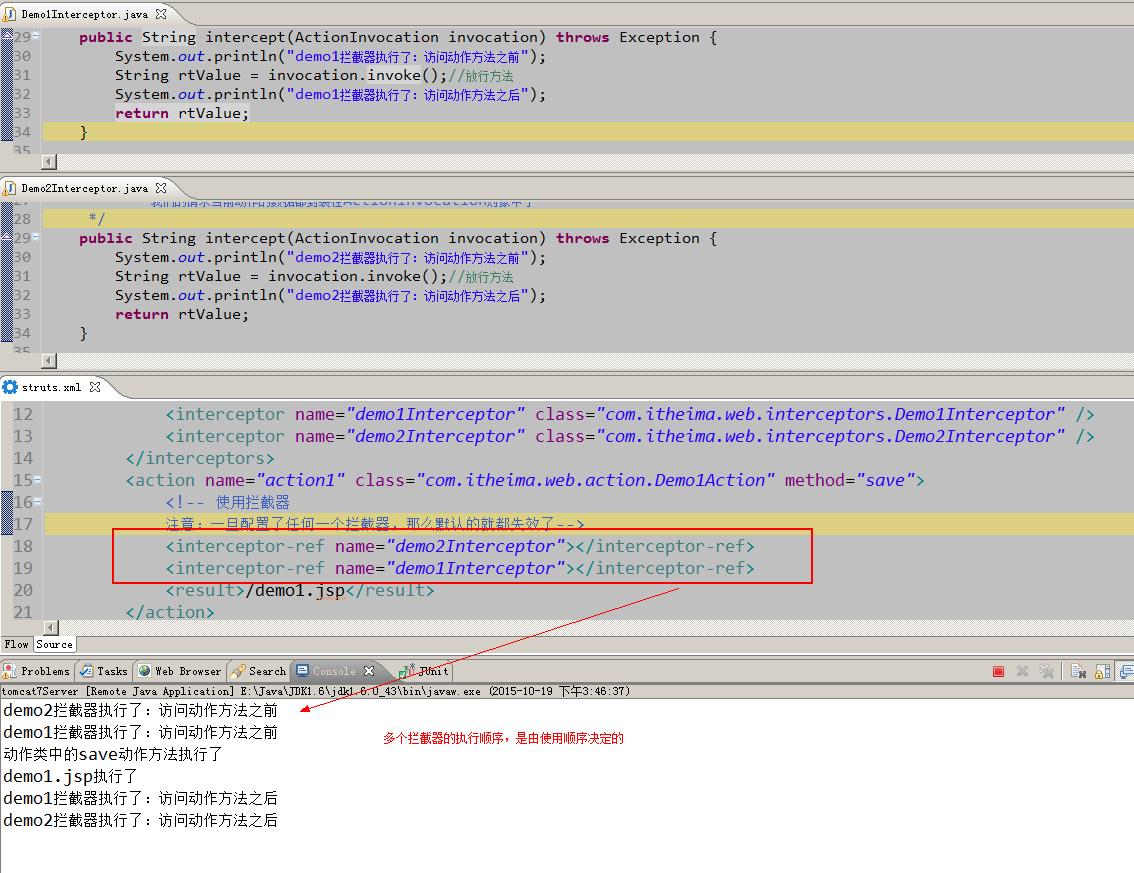
b、配置拦截器：注意拦截器必须先声明再使用



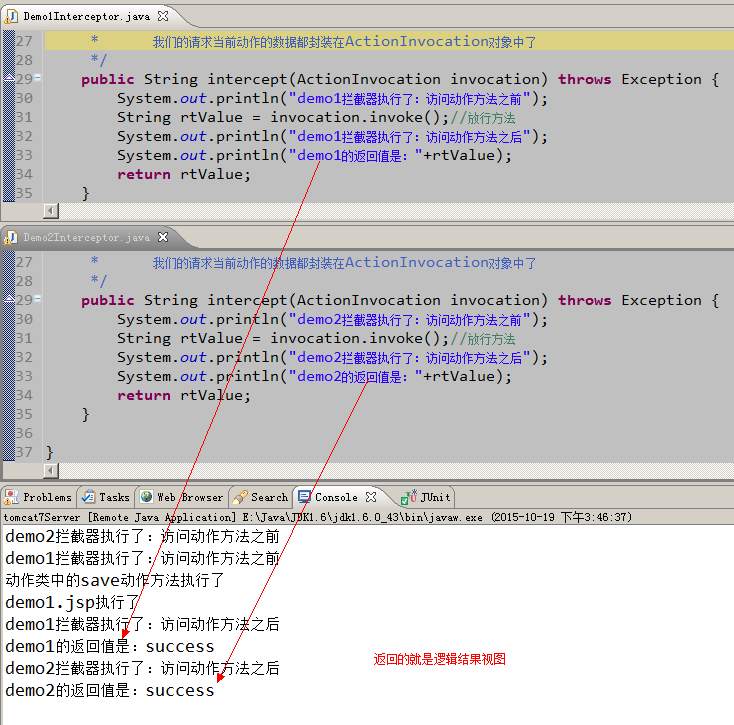
### 3.3、执行顺序



### 3.4、多个拦截器的执行顺序



### 3.5、intercept方法的返回值



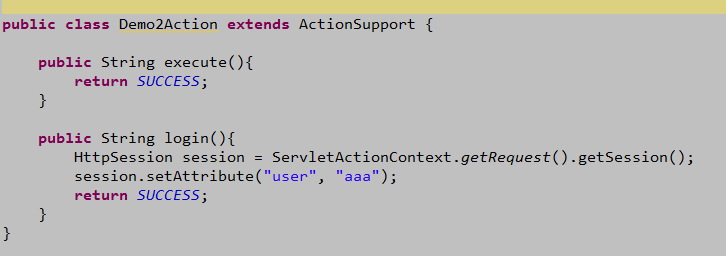
## 4、拦截器的应用：

### 4.1、检查登录的拦截器案例

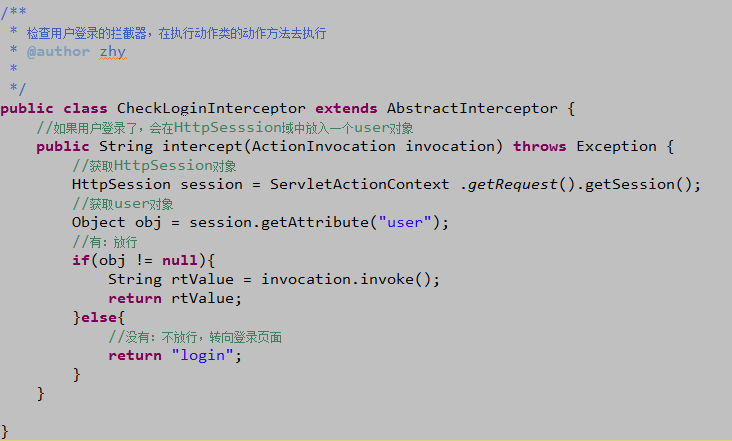
配置文件：



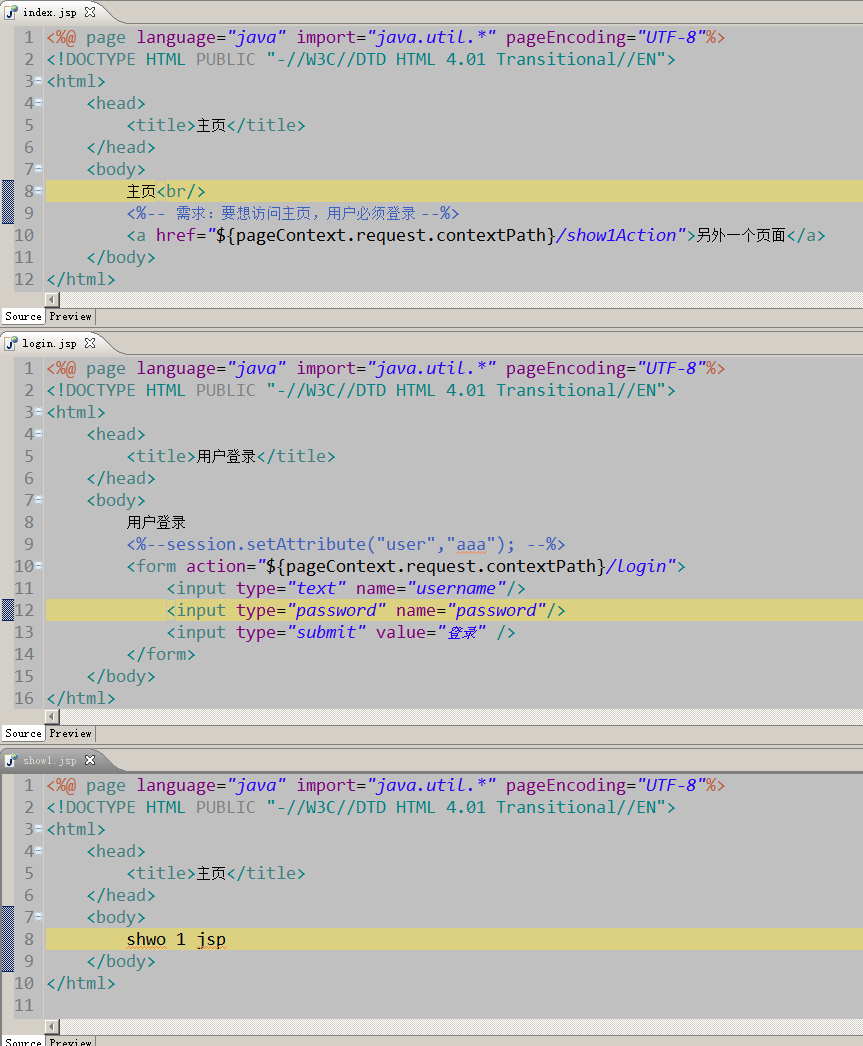
动作类：



拦截器：



页面：



### 4.2、案例中的问题

**问题：**由于我们写了自己的拦截器，默认的拦截器不起作用了。

**解决办法：**

a、把默认拦截器加入到配置文件中



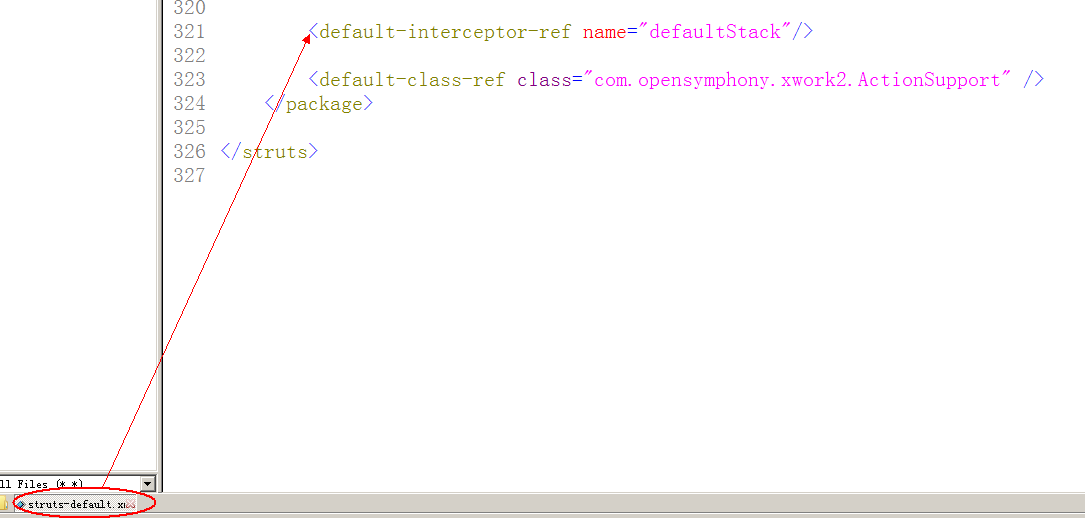
**b、a中暴露的问题：**当有多个拦截器时，需要改写的地方非常多。

**解决办法：**抽取公共的包，把全局配置放入公共包中。



**c、b中的问题**：还要再每个动作方法中引入拦截器。能不能不写呢？

思路：我们在设置【开发模式】时，覆盖掉了一个default.properties中的常量，能不能把struts-default.xml中的默认拦截器栈的设置给覆盖掉呢？答案是可以的。



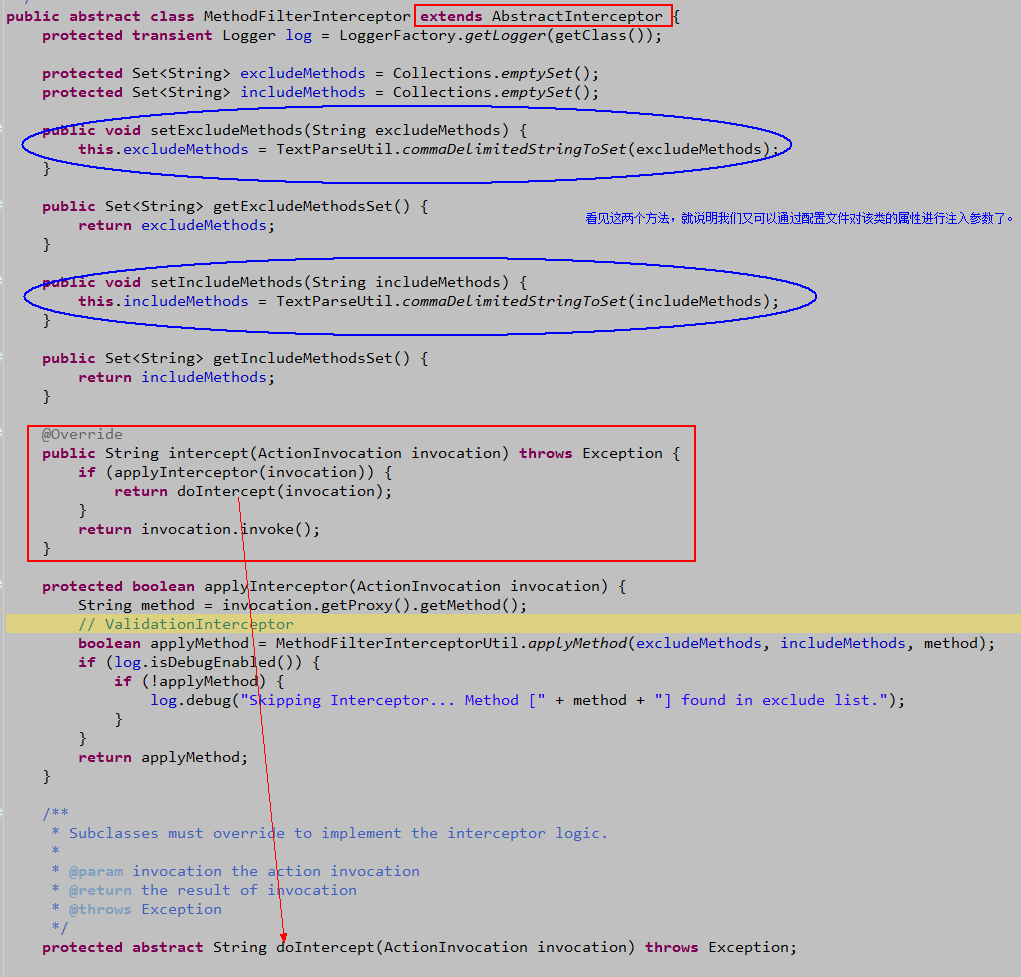
**解决办法：**



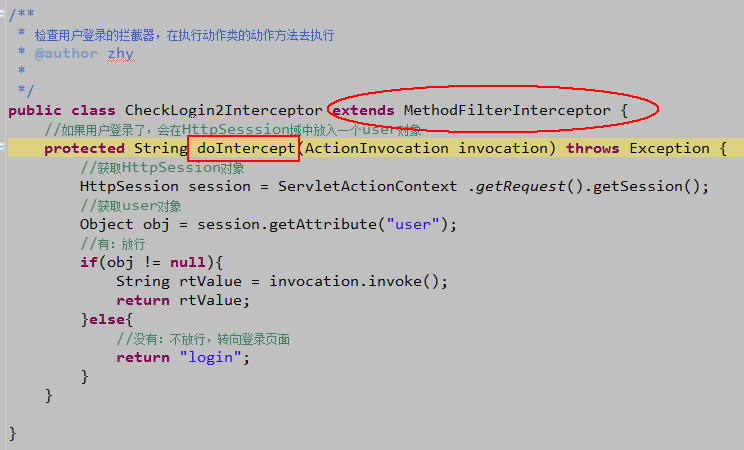
**d、c中出现的问题**：当使用了默认拦截器栈，这时候三个动作login,showIndex和show1Action都将被检查登录的拦截器拦截。

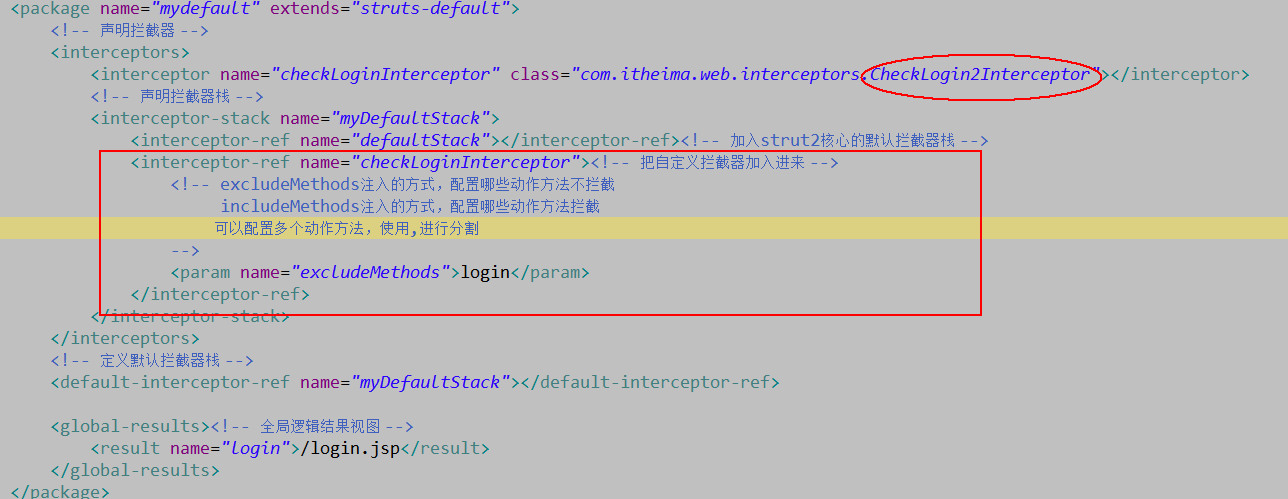
**解决办法：**

需要通过AbstractInterceptor类的子类入手，通过查看发现，该类还有一个子类是抽象的：



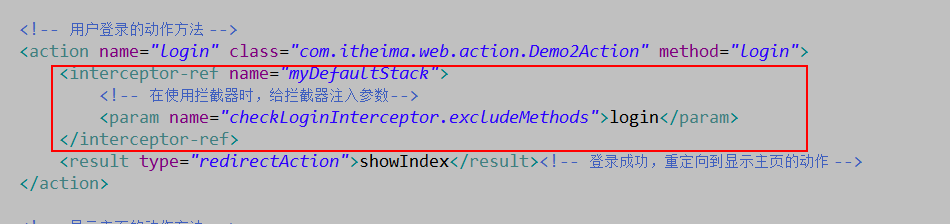
所以我们在自定义拦截器时，还可以继承MethodFilterInterceptor并且重写doIntercept方法。

并且在struts的配置文件中，配置需要拦截哪些方法，和需要放过哪些方法。

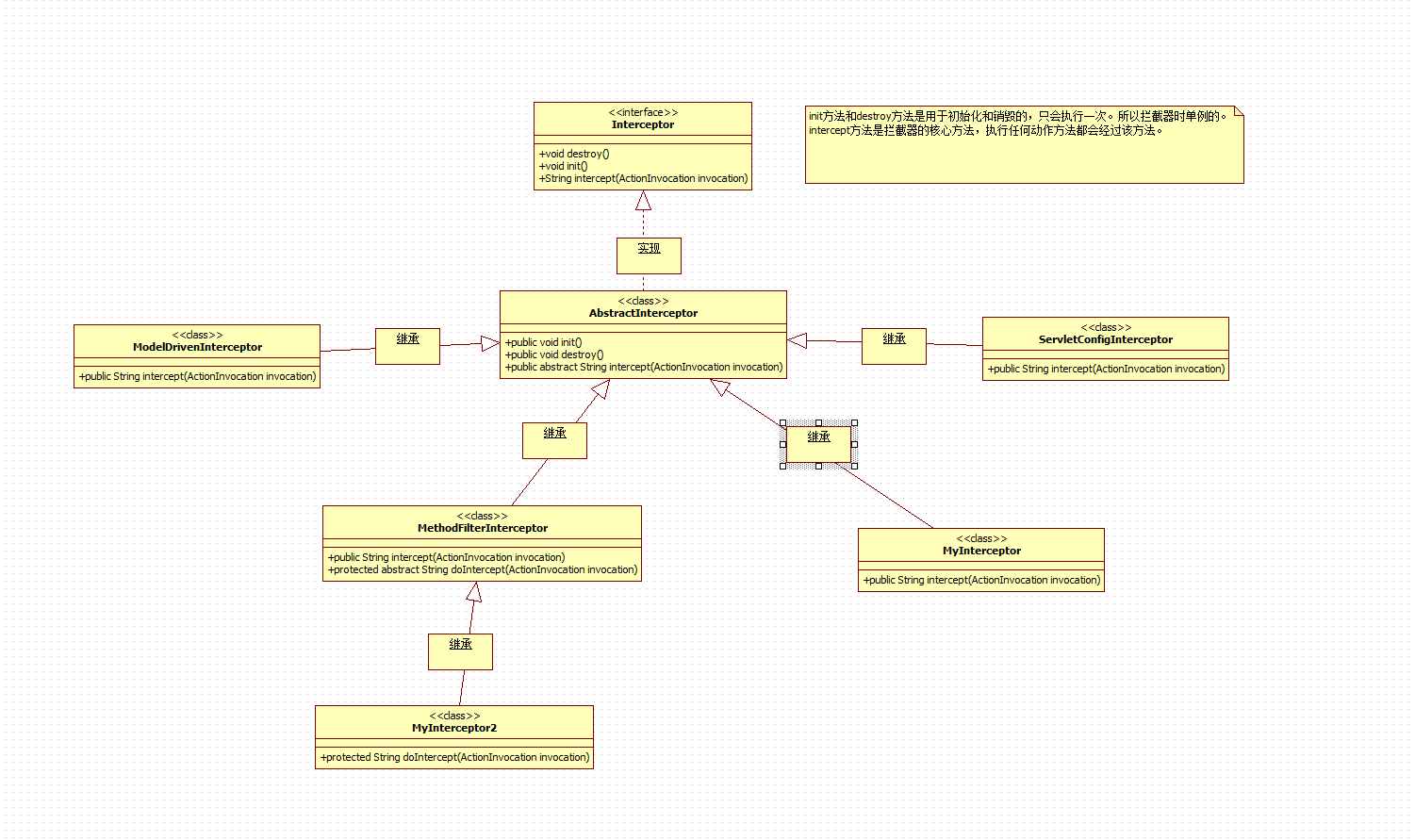


**e、d中遗留的问题**：我们在声明时配置了哪些方法需要拦截，哪些方法不需要拦截。但是在没有写动作类和动作方法之前，不确定方法名叫什么。

**解决办法：**我们需要在使用拦截器的时候给它注入参数。



### 4.3、拦截器类视图（全）：



# 四、文件的上传（拦截器）和下载（stream结果类型）（需要练一遍）

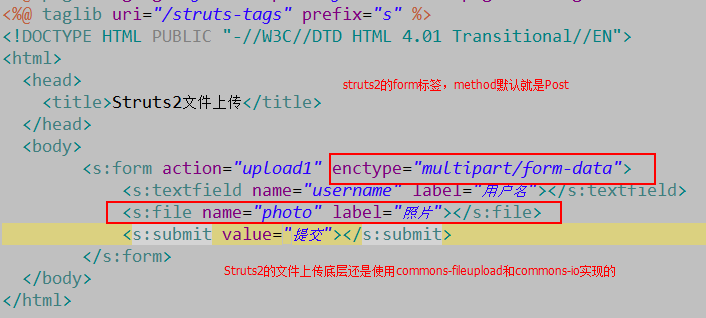
## 1、文件上传

必要前提：

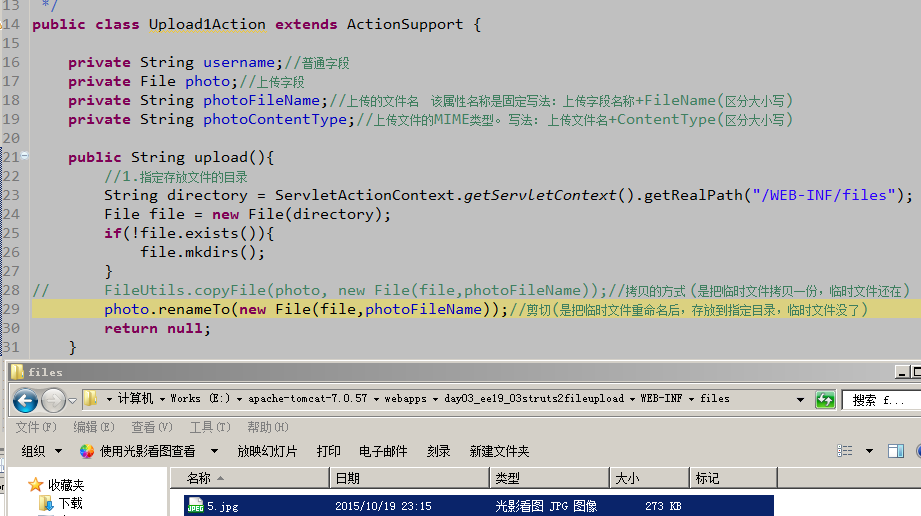
a.表单method必须是post；

b.enctype取值必须是multipart/form-data；

c.提供文件选择域。



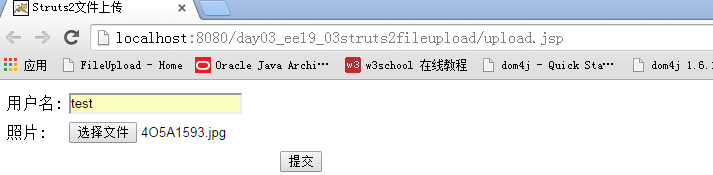
动作类：

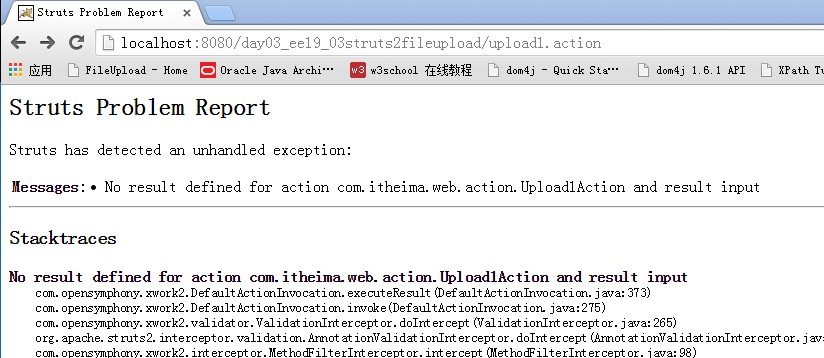
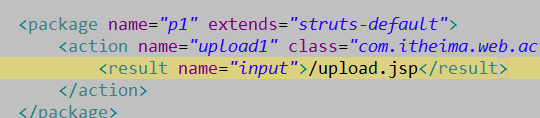


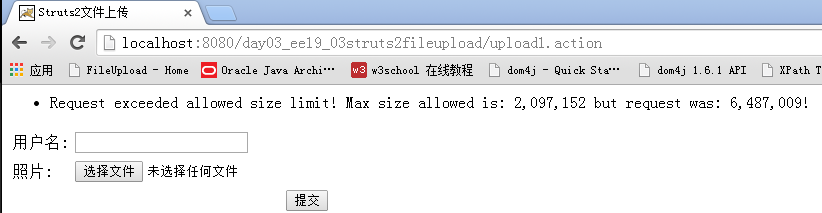
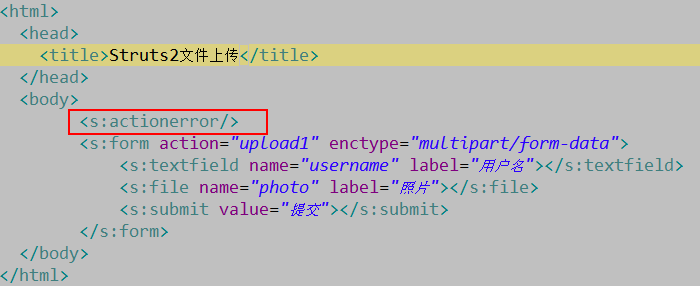
## 2、文件上传的配置

### 2.1、文件上传大小限制（默认是2MB）

如果上传文件超过了默认大小，upload拦截器会转向一个input的逻辑视图。

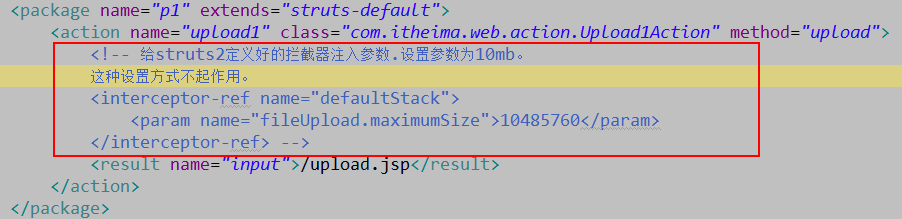
 

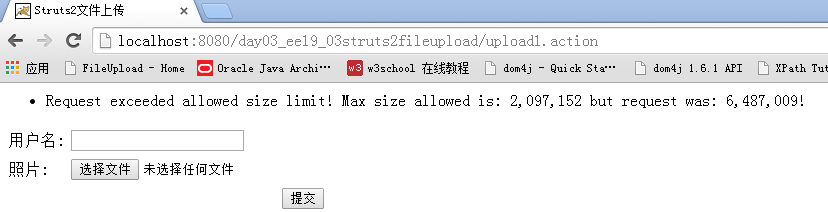
 



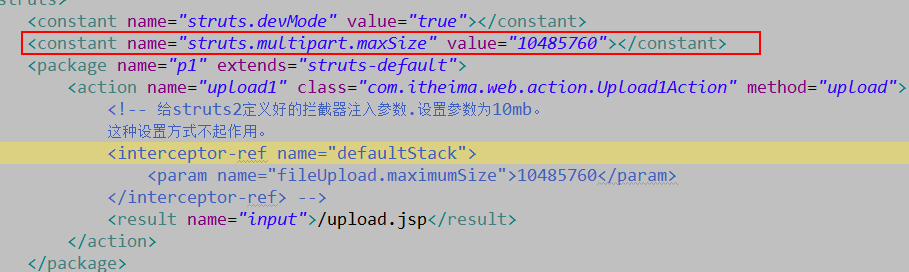
**a、改变上传文件大小限制：**

思路1：给Struts2的拦截器注入参数：（行不通）





思路2：在struts.xml中改变default.properties文件中的常量。

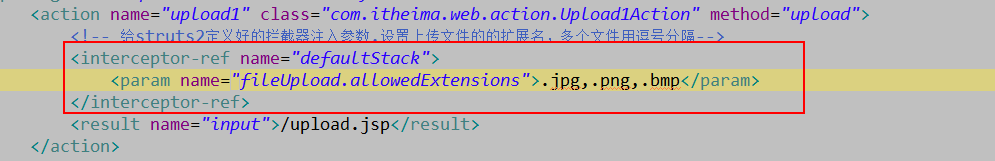


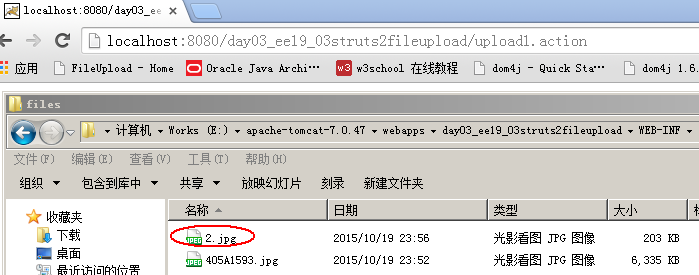


### 2.2、限制文件上传的类型

a、通过限制上传文件的扩展名

思路：给fileUpload拦截器注入参数

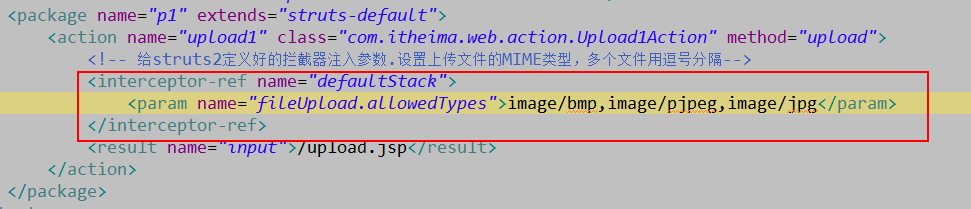




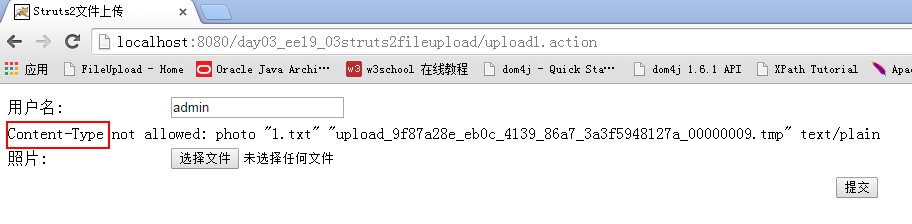
当上传非限定扩展名时：有如下错误提示



b、通过限制上传文件的MIME类型

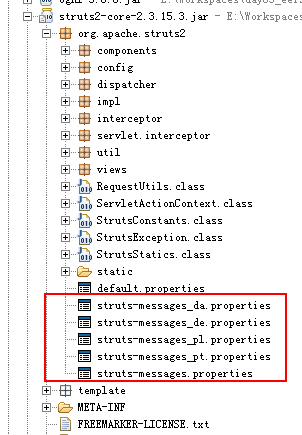


当上传非限定MIME类型时：有如下错误提示



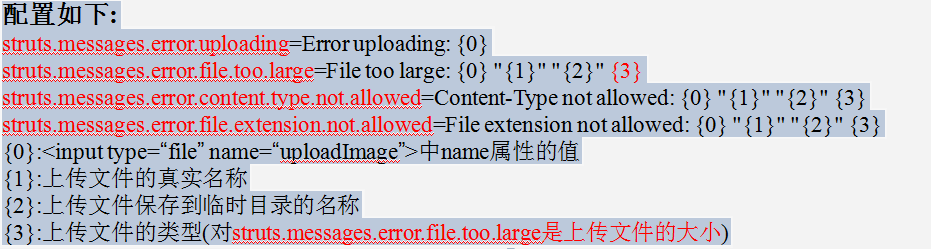
## 3、出错后的错误信息中文提示(Struts2中的所有文本提示都是基于国际化的)

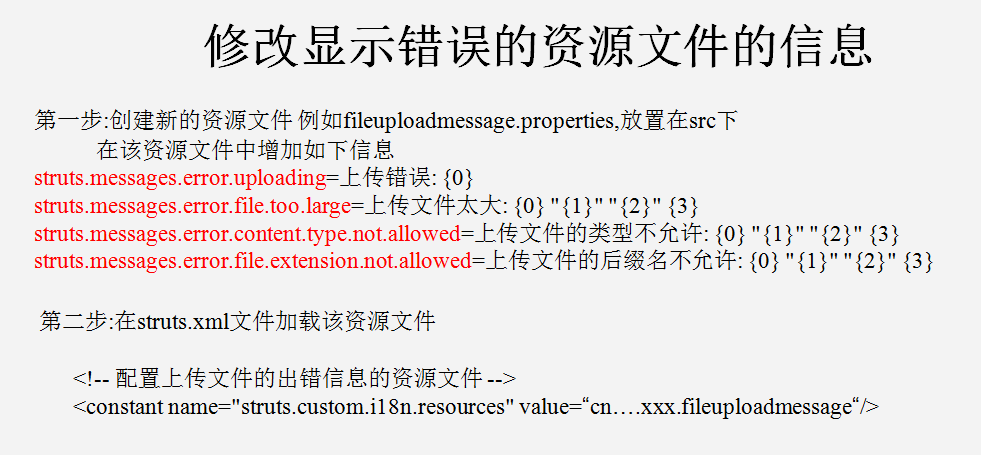
默认信息提示在：struts2-core.jar\org.apache.struts2\struts-message.properties



解决办法：用国际化消息资源包，把对应的key取值改为中文即可。

常用的key值：



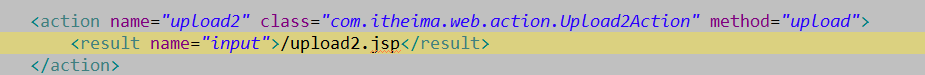


## 4、多文件上传

jsp页面：



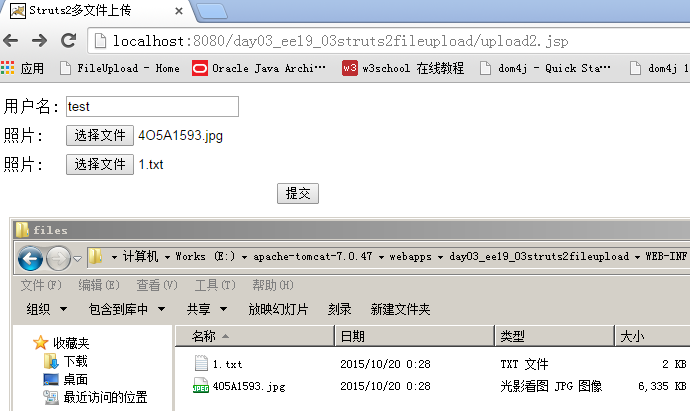
配置文件：



动作类：



运行结果：

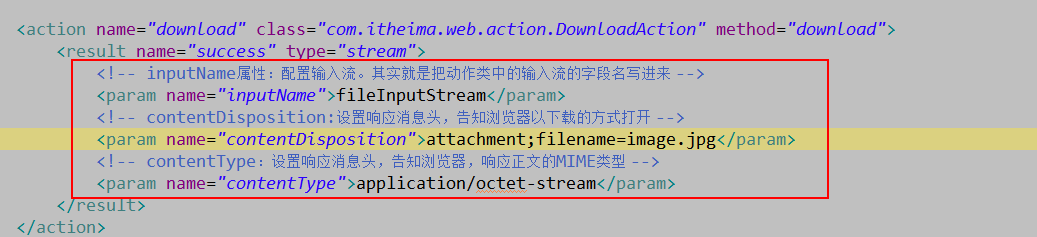


## 5、文件下载：其实就是一种结果类型（Stream）

动作类：



配置文件：



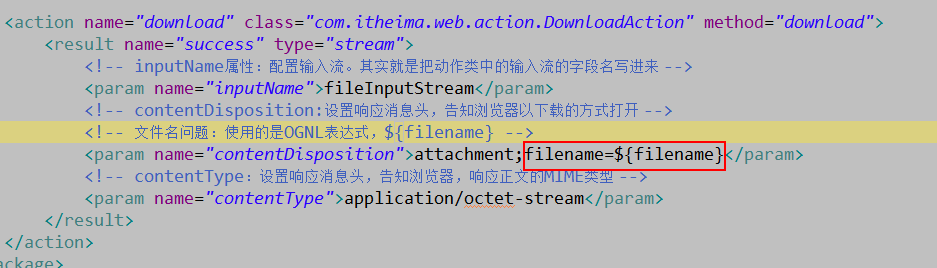
运行结果：



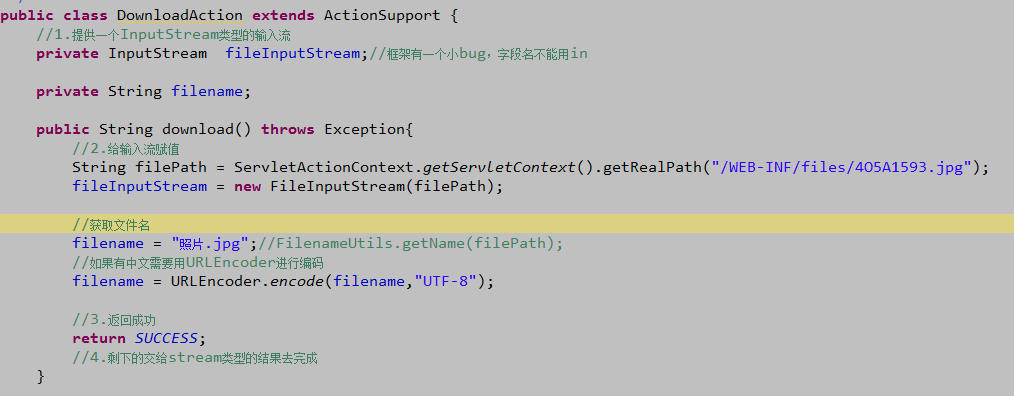
问题：文件名不能在配置文件中写死，需要根据实际情况获取。

解决办法：

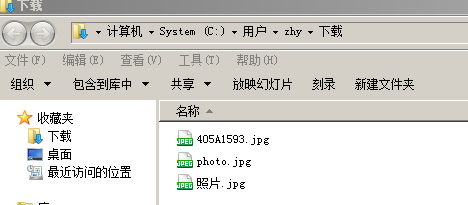
动作类：



配置文件：



运行结果：



# 五、OGNL简介（非常重要）

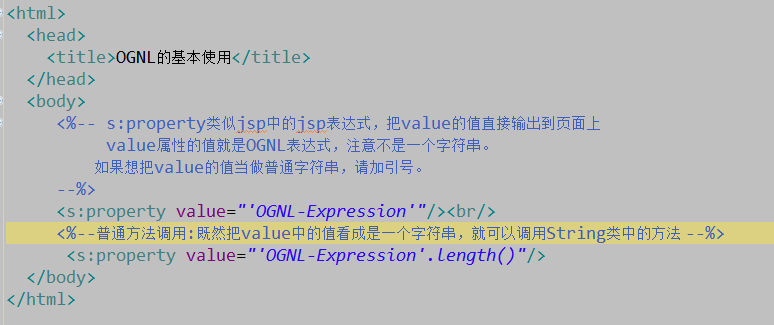
## 1、什么是OGNL

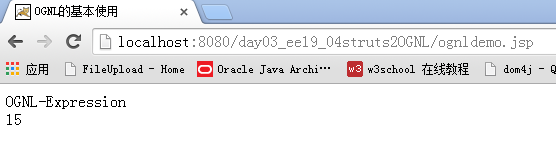
OGNL是Object Graphic Navigation Language（对象图导航语言）的缩写，它是一个单独的开源项目。 Struts2框架使用OGNL作为默认的表达式语言。

## 2、OGNL的功能

**前提：OGNL是struts2整合的一个开源项目，所以在struts2中，要想使用OGNL表达式，必须使用Struts2标签库**

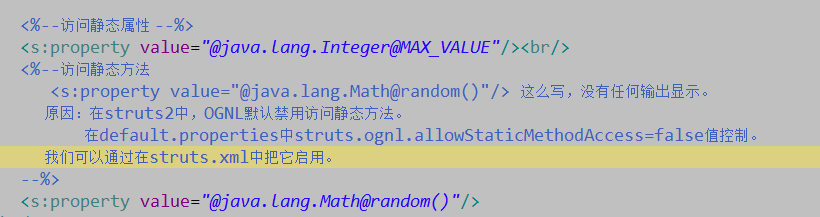
### 2.1、支持普通方法的调用





(EL表达式只能调用静态方法)

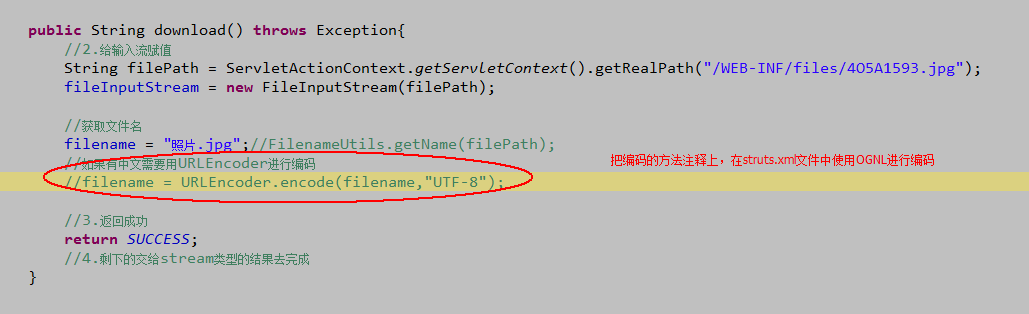
### 2.2、访问静态成员（静态属性，静态方法）







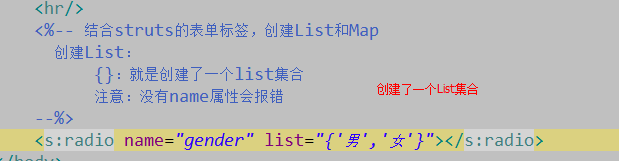
**OGNL改写文件下载：**

****

****

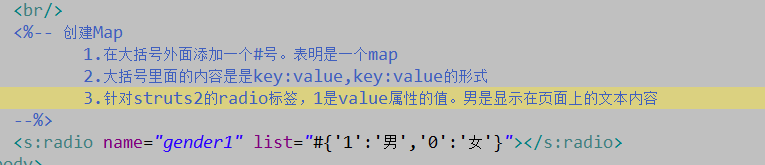
### 2.3、操作集合对象

a、创建List对象：





b、创建Map对象：





# 六、contextMap（非常重要）

## 1、动作类的生命周期

明确：动作类是多例的，每次动作访问，动作类都会实例化。所以是线程安全的。与Struts1的区别是，struts1的动作类是单例的。

## 2、请求动作的数据存放

**问题：**

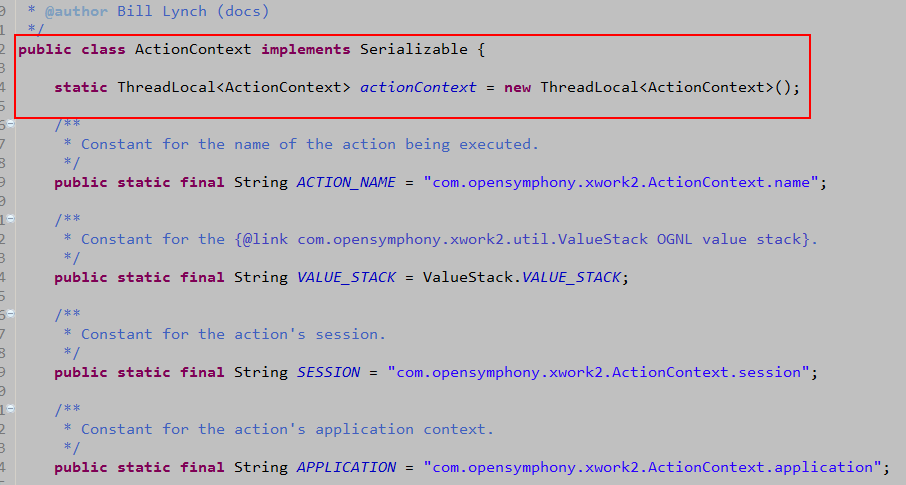
每次请求时，都会产生一些请求数据，这些数据存放到哪里去了？

**明确：**

在每次动作执行前，核心控制器StrutsPrepareAndExecuteFilter都会创建一个**ActionContext**和**ValueStack**对象。且每次动作访问都会创建。

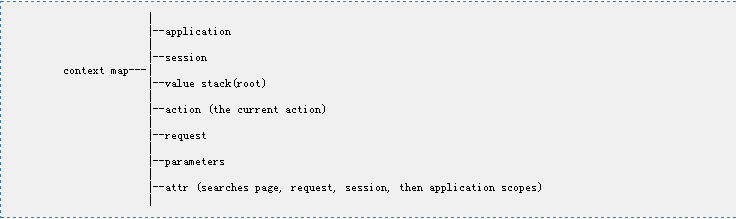
这两个对象存储了整个动作访问期间用到的数据。并且把数据绑定到了**线程局部变量（ThreadLocal）**上了。所以是线程安全的。





## 3、contextMap:存储数据

Struts2的官方文档对contextMap的说明：



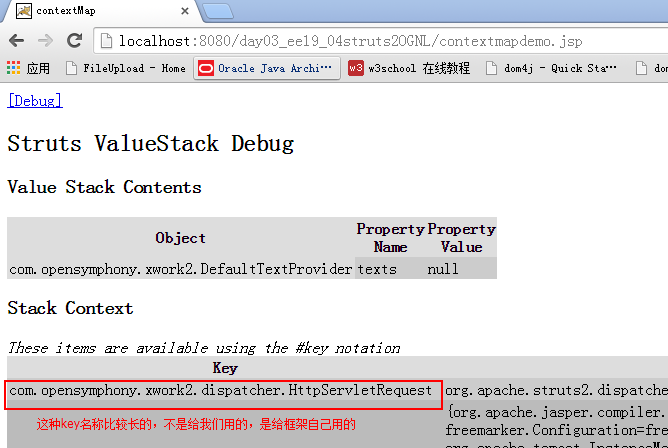
|  |  |  |
| --- | --- | --- |
| **contextMap中存放的主要内容** | | |
| **Key** | **Value** | **说明** |
| value stack (root) | java.util.List | 没有root这个key。它是一个list。 |
| application | java.util.Map<String,Object> | ServletContext中的所有属性。 |
| session | java.util.Map<String,Object> | HttpSession中的所有属性。 |
| request | java.util.Map<String,Object> | ServletRequest中的所有属性。 |
| parameters | java.util.Map | 参数 |
| attr | java.util.Map | 把页面、请求、会话、应用范围内的所有属性放到一起。 |

**注意：**

除了value stack之外，全是map，而contextMap也是一个map。其实就是Map中又封装的Map。（很像dbutils中KeyedHandler封装数据的结构，只是封装数据的结构）

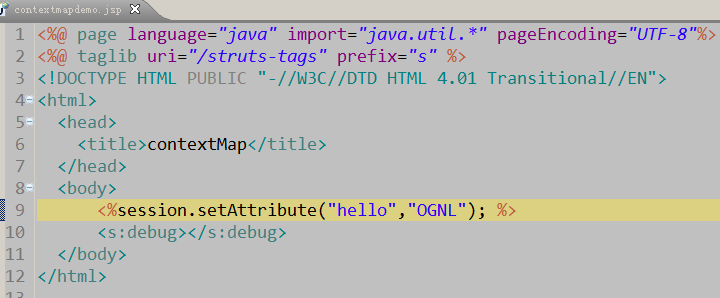
**查看contextMap中的数据：**

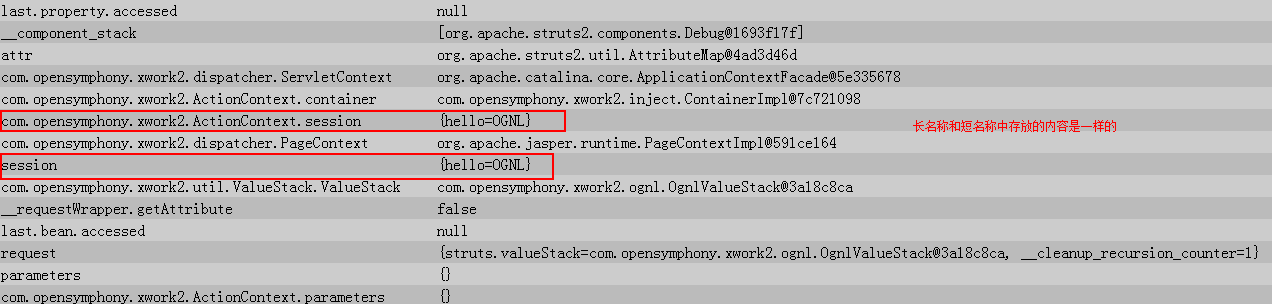
在页面上使用**<s:debug/>**

****

****

**测试存入数据：**

****

****