

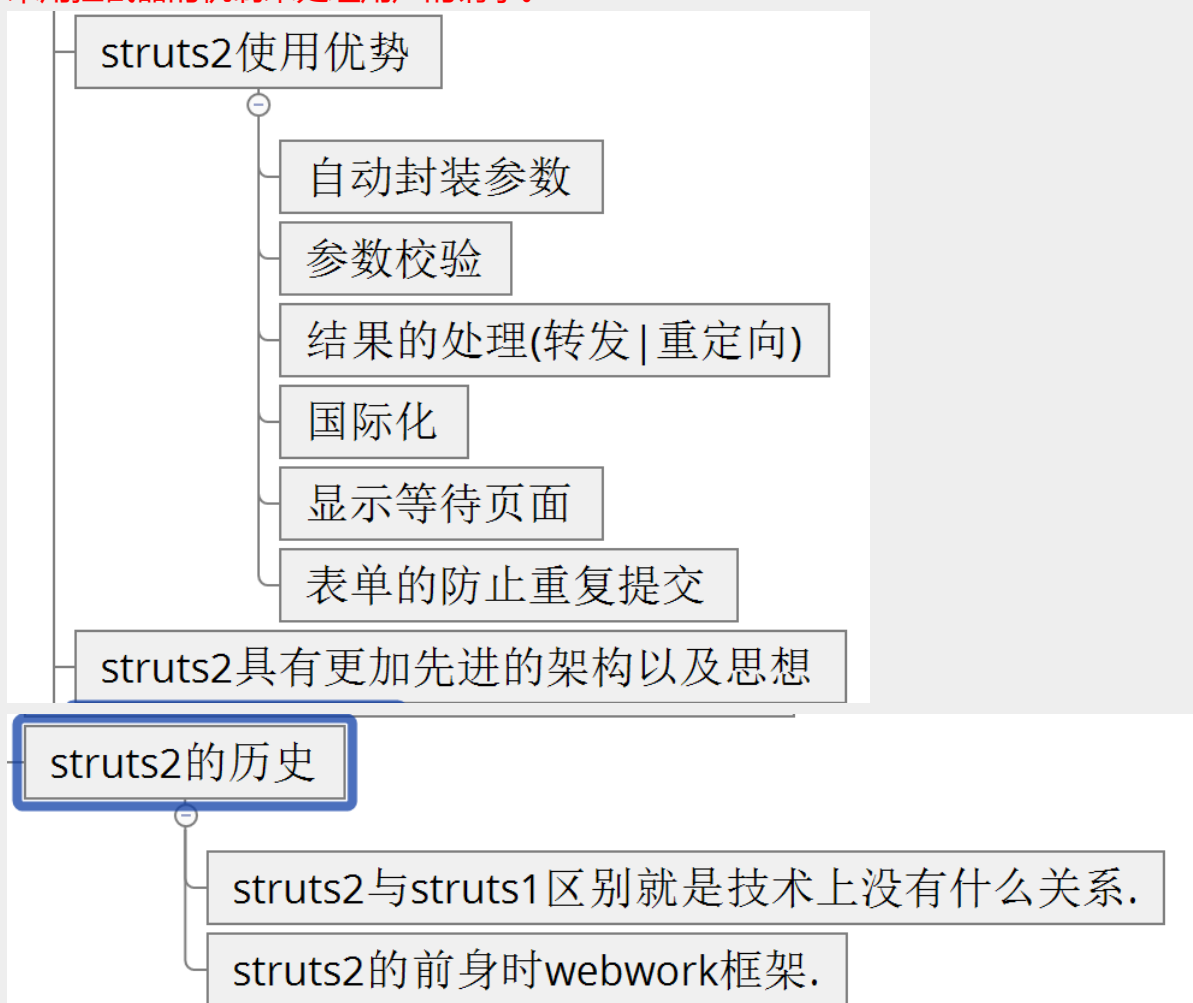
Struts2入门

一、Struts2框架概述

1.1 什么是Struts2

在介绍Struts2之前，先来认识一下Struts1。Struts1是最早基于MVC模式的轻量级Web框架，它能够合理的划分代码结构，并包含验证框架、国际化框架等实用工具框架。但是随着技术的进步,Struts1的局限性也越来越多的暴露出来。为了符合更加灵活、高效的开发需求，Struts2框架应运而生。

1. Struts2是在Struts1和WebWork技术的基础上进行合并后的全新框架，也是一个MVC框架。
2. 虽然Struts2的名字与Struts1相似，但其设计思想却有很大不同，实际上，Struts2是以WebWork为核心的。
3. 采用拦截器的机制来处理用户的请求。
















1.2 常见WEB层框架

Struts2、Struts1、WebWork、SpringMVC

二、Struts2快速入门

2.1 导包

我们不用去导所有的Struts2包，因为包中还包含着其他框架的内容，会出错。在apps文件夹下有示例程序，解压可以导入其引入的包

struts-2.3.24-all > struts-2.3.24 > apps > struts2-blank > WEB-INF > lib				
名称	修改日期	类型	大小	
 asm-3.3.jar	2013/11/23 17:55	Executable Jar File	43 KB	
 asm-commons-3.3.jar	2013/11/23 17:55	Executable Jar File	38 KB	
 asm-tree-3.3.jar	2013/11/23 17:55	Executable Jar File	21 KB	
 commons-fileupload-1.3.1.jar	2014/2/19 16:21	Executable Jar File	68 KB	
 commons-io-2.2.jar	2013/11/23 17:55	Executable Jar File	170 KB	
 commons-lang3-3.2.jar	2014/1/2 21:45	Executable Jar File	376 KB	
 freemarker-2.3.22.jar	2015/4/3 7:09	Executable Jar File	1,271 KB	
 javassist-3.11.0.GA.jar	2013/11/23 17:55	Executable Jar File	600 KB	
 log4j-api-2.2.jar	2015/4/19 12:04	Executable Jar File	131 KB	
 log4j-core-2.2.jar	2015/4/19 12:04	Executable Jar File	808 KB	
 ognl-3.0.6.jar	2013/11/23 17:55	Executable Jar File	223 KB	
 struts2-core-2.3.24.jar	2015/5/3 12:25	Executable Jar File	813 KB	
 xwork-core-2.3.24.jar	2015/5/3 12:23	Executable Jar File	661 KB	

2.2 编写Action类

```
1 package cn.itheima.actions;
2
3 public class HelloAction {
4     /*
5      * (1) 每次访问servlet时候，都会执行service方法
6      * (2) 访问action时执行execute方法
7      * */
8     public String execute(){
9         return "ok";
10    }
11
12
13 }
```

2.3 书写src/struts.xml文件

同样可以去示例程序中寻找struts.xml的配置文件，最主要是得到配置文件的约束条件。

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE struts PUBLIC
3     "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
4     "http://struts.apache.org/dtds/struts-2.3.dtd">
5 <struts>
6     <package name="helldemo" extends="struts-default" namespace="/">
7         <!--name:访问的资源名称-->
8         <action name="hello" class="cn.itheima.actions.HelloAction">
9             <result name="ok">/hello.jsp</result>
10        </action>
11    </package>
12 </struts>

```

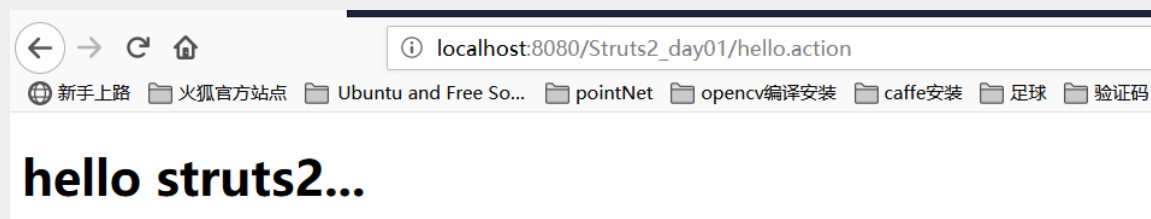
2.4 将Struts2核心过滤器配置到web.xml

```

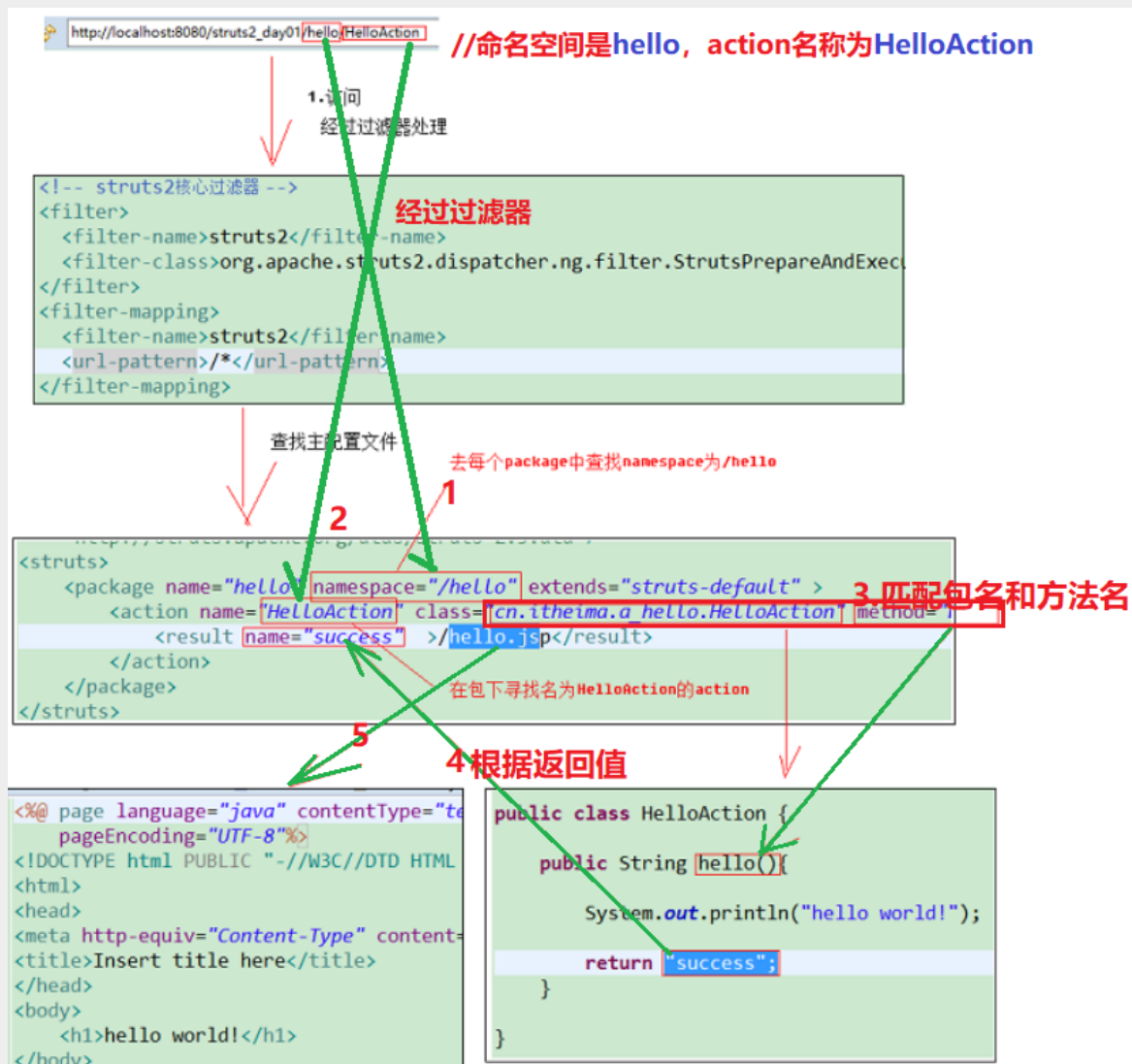
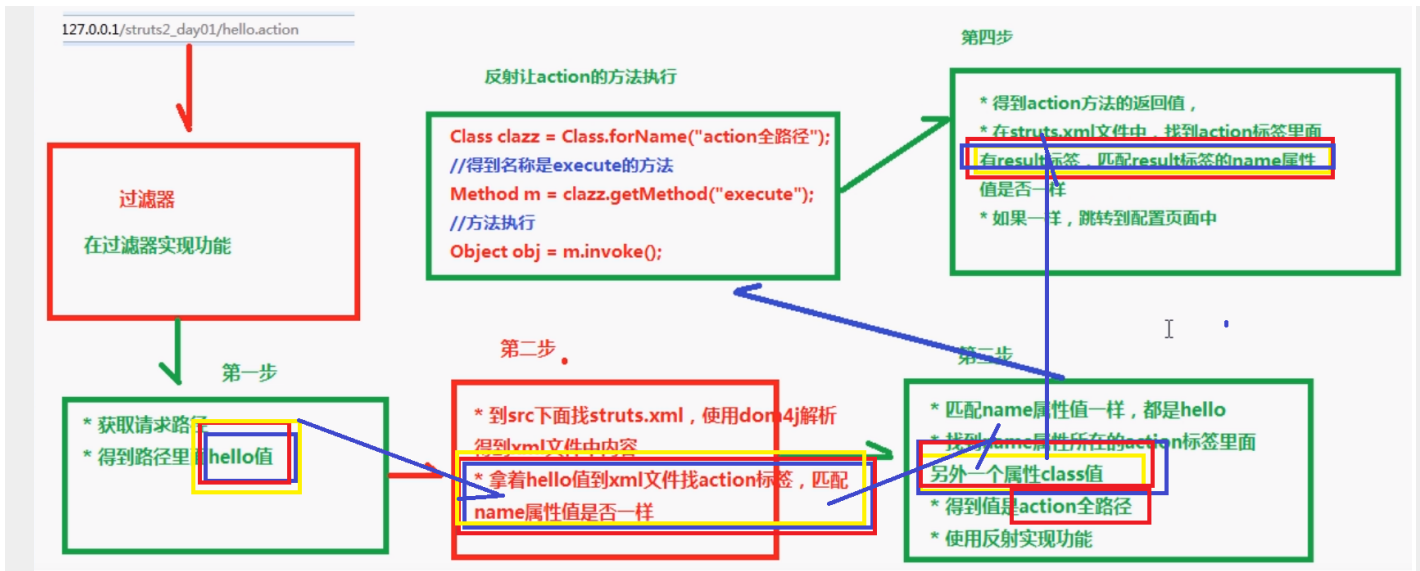
4 <filter>
5     <filter-name>struts2</filter-name>
6     <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
7 </filter>
8
9 <filter-mapping>
10    <filter-name>struts2</filter-name>
11    <url-pattern>/*</url-pattern>
12 </filter-mapping>

```

2.5 测试



三、Struts2基本执行过程



四、Struts2配置

4.1 struts.xml基本配置

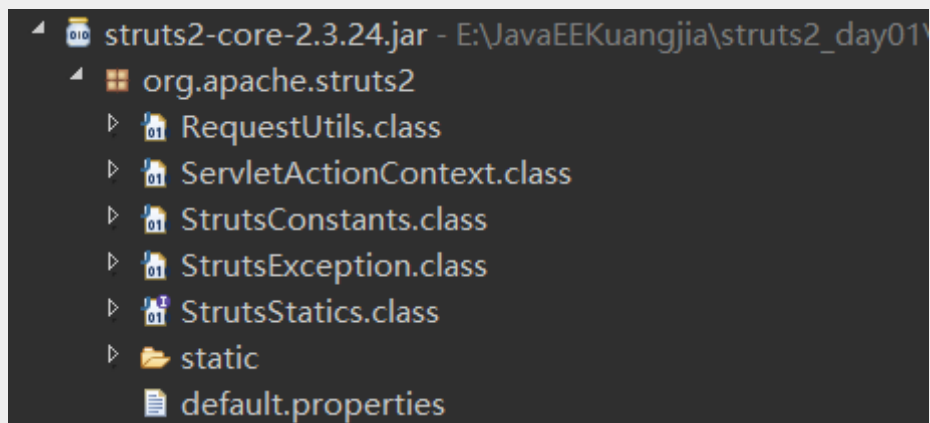
```

<!-- package: 将Action配置封装, 就是可以在Package中配置很多action.
      name属性: 给包起个名字, 起到标识作用, 随便起, 不能其他包名重复
      namespace属性: 给action的访问路径中定义一个命名空间
      extends属性: 继承一个 指定包
      abstract属性: 包是否为抽象的; 标识性属性, 标识该包不能独立运行, 专门被继承 //可选
-->
<package name="hello" namespace="/hello" extends="struts-default" >
  <!-- action元素 配置action类 //访问名称
      name属性: 决定了Action访问资源名.
      class属性: action的完整类名
      method属性: 指定调用Action中的哪个方法来处理请求
  -->
  <action name="HelloAction" class="cn.itheima.a_hello.HelloAction" method="hello" >
    <!-- result元素 结果配置
      name属性: 标识结果处理的名称, 与action方法的返回值对应.
      type属性: 指定调用哪一个result类来处理结果, 默认使用转发.
      标签体: 填写页面的相对路径
    -->
    <result name="success" type="dispatcher" >/hello.jsp</result>
  </action>
</package>
43 <!-- 引入其他struts配置文件 -->
44 <include file="cn/scct/b_dynamic/struts.xml"></include>
45 <include file="cn/scct/c_default/struts.xml"></include>

```

4.2 struts2常量配置

4.2.1 默认常量配置位置



4.2.2 修改struts2常量配置

4.2.2.1 修改方式

方式1:src/struts.xml

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

方式2:在src下创建struts.properties

struts.properties

```
1 struts.i18n.encoding=UTF8
```

方式3:在项目的web.xml中

```
<!-- 配置常量 -->
<context-param>
  <param-name>struts.i18n.encoding</param-name>
  <param-value>UTF-8</param-value>
</context-param>
```

顺序

根据上面的代码我们可以得出配置文件的加载顺序如下

```
* default.properties
* struts-default.xml
* struts-plugin.xml
* struts.xml      ---- 配置 Action 以及常量. (*****)
* struts.properties ---- 配置常量
* web.xml         ---- 配置核心过滤器及常量.
```

4.2.2.2 常量配置

```
<!-- i18n:国际化. 解决post提交乱码 -->
<constant name="struts.i18n.encoding" value="UTF-8"></constant>
<!-- 指定反问action时的后缀名
      http://localhost:8080/struts2_day01/hello/HelloAction.do
-->
<constant name="struts.action.extension" value="action,"></constant>
<!-- 指定struts2是否以开发模式运行
      1. 热加载主配置.(不需要重启即可生效) 默认是false 可以任意修改, 默认后缀名为action或为空
      2. 提供更多错误信息输出, 方便开发时的调试
-->
<constant name="struts.devMode" value="true"></constant>
```

4.3 struts.xml配置进阶

4.3.1 动态方法

```
3 //动态方法调用
4 public class Demo1Action {
5     public String add(){
6         System.out.println("添加用户!");
7         return "success";
8     }
9     public String delete(){
10        System.out.println("删除用户!");
11        return "success";
12    }
13    public String update(){
14        System.out.println("修改用户!");
15        return "success";
16    }
17    public String find(){
18        System.out.println("查找用户!");
19        return "success";
20    }
21
22    为了不固定调用它的这
23    些方法,
```

4.3.1.1 方法一(不推荐)

```
6 <!-- 配置动态方法调用是否开启常量
7     默认是关闭的,需要开启
8     --> //只要这个开启就行
9 <constant name="struts.enable.DynamicMethodInvocation" value="true"></constant>
10
11 <package name="dynamic" namespace="/dynamic" extends="struts-default" >
12     <!-- 动态方法调用方式2:通配符方式
13         使用{1} 取出第一个星号通配的内容
14         -->
15     <action name="Demo1Action" class="cn.scct.b_dynamic.Demo1Action" >
16         <result name="success" >/hello.jsp</result> //无需配置method属性
17     </action>
18     <!-- <action name="Demo1Action_*" class="cn.itheima.b_dynamic.Demo1Action" method="
19         <result name="success" >/hello.jsp</result>
20     </action> -->
21 </package>
```

//只要在类名和方法名间加上! 即可

http://localhost:8080/struts2_day01/dynamic/Demo1Action!add.action

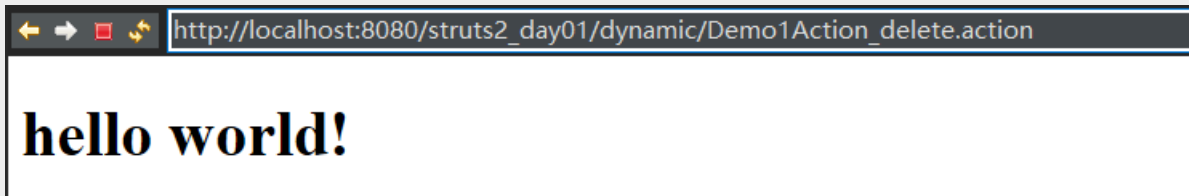
hello world!

4.3.1.2 方法二(通配符)

```

<package name="dynamic" namespace="/dynamic" extends="struts-default" >
  <!-- 动态方法调用方式2:通配符方式
        使用{1} 取出第一个星号通配的内容
        -->
        比如当访问/dynamic/Demo1Action_add时, method方法就是add
  <!-- <action name="Demo1Action" class="cn.scct.b_dynamic.Demo1Action" >
        <result name="success" >/hello.jsp</result>
  </action> -->
        表示用表达式{1}匹配第一个通配符*
  <action name="Demo1Action_*" class="cn.itheima.b_dynamic.Demo1Action" method="{1}" >
        <result name="success" >/hello.jsp</result>
  </action>
</package>

```



4.3.2 默认配置

```

5 <struts>
6   <package name="default" namespace="/default" extends="struts-default" >
7     <!-- 找不到包下的action,会使用Demo2Action作为默认action处理请求 -->
8     <default-action-ref name="Demo2Action"></default-action-ref>
9     <!-- method属性:execute -->
10    <!-- result的name属性:succcess -->
11    <!-- result的type属性:dispatcher 转发 -->
12    <!-- class属性:com.opensymphony.xwork2.ActionSupport -->
13    <action name="Demo2Action" >
14      <result >/hello.jsp</result>
15    </action>
16  </package>
17 </struts>

```

五、action书写方式

Action类的书写方式

方式1

```
//方式1: 创建一个类. 可以是POJO  
//POJO: 不用继承任何父类. 也不需要实现任何接口  
//使struts2框架的代码侵入性更低.  
public class Demo3Action {
```

方式2

```
//方式2: 实现一个接口Action  
// 里面有execute方法, 提供action方法的规范.  
// Action接口预置了一些字符串. 可以在返回结果时使用. 为了方便  
public class Demo4Action implements Action {
```

方式3

```
//方式3: 继承一个类. ActionSupport  
// 帮我们实现了 Validateable, ValidationAware, TextProvider, LocaleProvider .  
// 如果我们需要用到这些接口的实现时, 不需要自己来实现了.  
public class Demo5Action extends ActionSupport{
```