吴老师教学讲义

忽然抚尺一下，群响毕绝。撤屏视之，一人、一桌、一椅、一扇、一抚尺而已

吴　青

QQ:16910735

wuqing_bean@126.com

http://blog.sina.com/accpwulaoshi

Struts2—配置文件

Struts2 中有许多的默认的配置，这些默认的配置都是可以修改的，它们在哪里，如何进行修改？本节讲述 Struts2 中的配置文件。

1. 问题

在前面入门案例中我们可能发现了下面的问题:

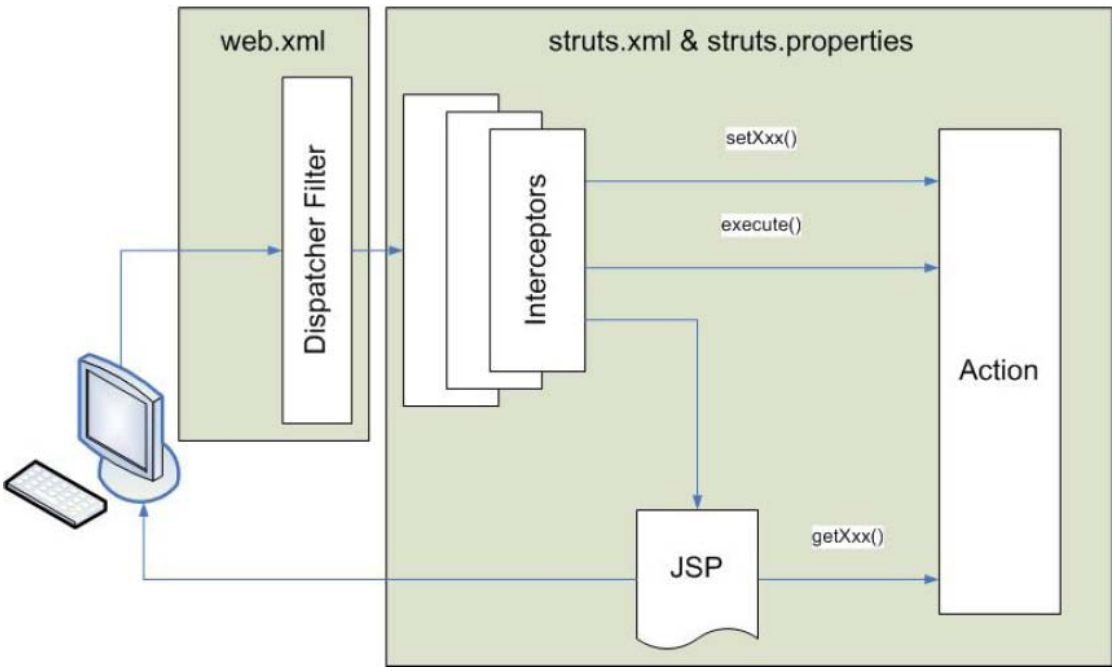
- Struts2 中默认的请求路径是 *.action 而 strtus1 默认的请求路径是*.do,是在 web.xml 中配置 ActionServlet 的时候指定的，我们可以去更改它。但是 Strtus2 中，我们没有在 web.xml 文件中进行配置，那么我们如果想更改这个配置，如变成与 Struts1 一样，请求路径变成*.do ，该如何做呢？
- strtus.xml 放在 src 目录中（编译之后就放到了 /WEB-INF/classes 目录中了），我们也没有在任何地方告诉 struts2 框架这个文件的存在,它是在哪里被加载的？
- 在 struts.xml 文件中有个 package 节点，该节点中有个 extends 属性，struts-default 是怎么回事.

要搞清楚这些问题，就需要了解 struts2 的一些配置文件，以及配置文件如何被加载。

2. Struts2 涉及到的配置文件

struts2 框架在启动的时候会加载很多的配置文件，这一点我们可以为项目添加日志 jar 包(commons-logging-1.0.4.jar,或者 Log4j)，观察输出的日志信息可以看到。

整个配置我们可以分为两大块，一个是在 web.xml 文件中的配置，另一块是 Struts2 框架中的配置。框架中的配置又有执行环境的配置和 Struts2 组件配置。



- web.xml 配置
- 框架执行环境配置(全局配置选项):struts.properties 文件
- 组件配置文件:struts-default.xml,struts-plugin.xml,strtus.xml

3. web.xml 配置

FilterDispatcher 是一个过滤器。注意，在 Struts2.0.X 的时候，使用的是 org.apache.struts2.dispatcher.FilterDispatcher 作为核心控制器,而 Struts2.1 中改成了 org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter。它是整个 web 应用的配置项，需要在 web.xml 中进行配置.

```
<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>
```

如果是一个基本的 web 应用，这样就足够了，剩下的就是配置 web 应用的一些执行环境配置（全局配置）和 web 应用中使用到的组件的配置，如 action 配置，拦截器的配置。

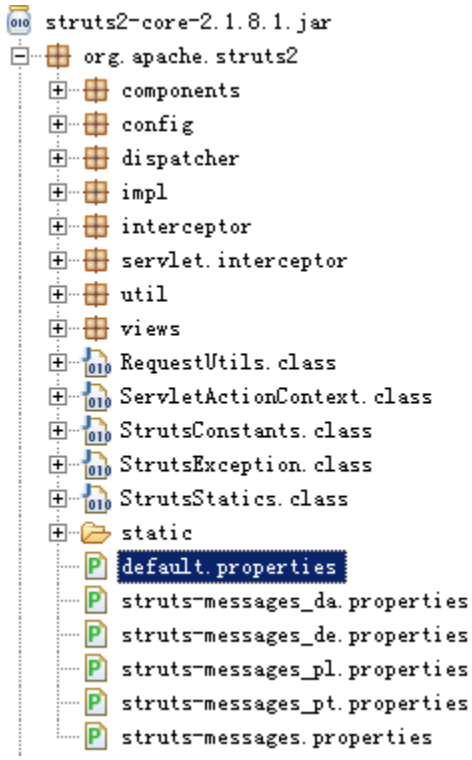
web 应用的执行环境主要是通过 struts.properties 来完成。应用中的组件配置主要是通过 strtus.xml 来完成。

4. struts.properties 文件

这个文件提供了一种更改框架默认行为方式的机制。在一般情况下，如果不是打算让调试更加方便的话，我们没有必要更改这个文件。哪么这些默认的属性在哪里存放呢？

在项目的 src 目录中可以自己编写一个名称为 struts.properties 文件 ,编译以后就放到了/WEB-INF/classes 中，Struts2 框架在启动的时候，会自动读取这个文件，但是在读取这个文件之前，会先到 struts2-core-xxx.jar 包中加载名为 default.properties 文件，这个文件中定义了默认的配置，所以我们可以在 strtus.properties 中定义一些配置覆盖 default.properties 中的配置，如果没有 struts.properties 文件，则采用默认配置。

打开 default.properties 文件我们会看到如下部分类容：



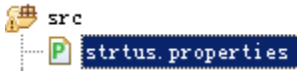
打开该文件 ,可以看到一些全局的配置选项 ,在这些选项中 ,我们发现了有一些这样的配置，其中 struts.action.extension 配置的就是 Struts 默认的请求后缀名。关于更多的配置的含义，后面用到了再讲解，现在暂时不必理会这些配置的含义.

```
### Used by the DefaultActionMapper
### You may provide a comma separated list, e.g. struts.action.extension=action,jnlp,do
### The blank extension allows you to match directory listings as well as pure action names
### without interfering with static resources.
struts.action.extension=action,,
### This can be used to set your default locale and encoding scheme
# struts.locale=en_US
struts.i18n.encoding=UTF-8
```

4.1 更改默认配置

因为 default.properties 文件是存放在 jar 包中的，struts2 启动的时候自动会寻找到。我们不能直接修改这个文件，但是我们可以使用 struts.properties 文件来覆盖 default.properties 文件中的内容。

在 web 项目的 src 的根目录中新建一个 struts.properties,然后将想要修改的属性添加到该文件中，就可以覆盖掉原来的配置.注意:这个文件存放在 src 的根目录中（编译之后放到了/WEB-INF/classes 根目录中）



文件内容:

```
##激活重新载入国际化文件的功能
struts.i18n.reload=true
##修改请求后缀为action或者do
struts.action.extension=action,do
##打开开发者模式，打开之后，我们修改配置文件之后不用重新启动服务器
struts.devMode =true
```

5. struts-default.xml

这个文件用来加载默认启动的组件。它存放在 struts2-core-xxx.jar 包的根目录下，系统启动的时候会加载这个文件。这个文件中配置的组件有类型转换组件，拦截器组件还有结果类型组件等等，关于这些组件的概念后面将会讲到，这里只需要了解。

6. struts-plugin.xml

可以在 struts2 中使用插件，Struts2 在启动的时候，会自动搜索 classpath 中的 jar 包中的 struts-plugin.xml 文件来加载插件。关于插件的应用，将会在后面讲到。

7. struts.xml

struts.xml 文件中包含的是我们开发的 Action 的配置。如前面登录例子中的配置:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd" >
<struts>
    <!-- struts2的Action必须放在指定的包空间下 -->
    <package name="com.wq" extends="struts-default">
        <!-- 定义action -->
        <action name="login" class="com.wq.web.action.LoginAction">
            <!-- 定义处理结果和资源之间的映射关系 -->
            <result name="success">/welcome.jsp</result>
            <result name="error">/error.jsp</result>
        </action>
    </package>
</struts>
```

7.1 在 struts.xml 中覆盖 default.properties 中的全局配置

我们修改全局配置的时候，使用 struts.properties 文件来覆盖 default.properties 文件中的内容。实际上我们可以不用创建 struts.properties 文件也可以做到覆盖 default.properties 文件中的配置，就是直接在 strtuts.xml 文件中使用<constant>配置

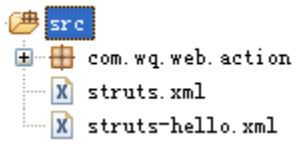
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd" >
<struts>
    <!-- 配置常量，覆盖default.properties中的配置 -->
    <constant name="struts.action.extension" value="do" />
    <!-- struts2的Action必须放在指定的包空间下 -->
    <package name="com.wq" extends="struts-default">
        <!-- 定义action -->
        <action name="login" class="com.wq.web.action.LoginAction">
            <!-- 定义处理结果和资源之间的映射关系 -->
            <result name="success">/welcome.jsp</result>
            <result name="error">/error.jsp</result>
        </action>
    </package>
</struts>
```

7.2 将 struts.xml 拆分成多个配置文件

可以想象，在 web 应用中还会不断的有 action 需要定义，这样 struts.xml 的配置会越来越多，文件会越来越大。

为了避免 struts.xml 文件过于庞大，臃肿，提高 strtus.xml 文件的可读性，我们可以将一个 struts.xml 文件分解成多个配置文件，然后在 strtus.xml 文件中包含其它配置文件.这样 struts2 就可以使用模块化的方式来管理 struts.xml 配置文件了。

我们可以在配置中使用<include> 来包含另外一个配置文件.下面我们新建一个 struts-hello.xml 文件，然后在 struts.xml 文件中包含该文件



struts.xml 文件中的内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd" >
<struts>
    <!-- 配置常量，覆盖default.properties中的配置 -->
    <constant name="struts.action.extension" value="do" />
    <include file="struts-hello.xml" />
</struts>
```

struts-hello.xml 文件中的内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd" >
<struts>
    <!-- struts2的Action必须放在指定的包空间下 -->
    <package name="com.wq" extends="struts-default">
        <!-- 定义action -->
        <action name="login" class="com.wq.web.action.LoginAction">
            <!-- 定义处理结果和资源之间的映射关系 -->
            <result name="success">/welcome.jsp</result>
            <result name="error">/error.jsp</result>
        </action>
    </package>
</struts
```

8. 配置文件加载顺序

配置文件的加载顺序从上到下依次是:我们可以打开 struts 的源代码，设置断点来跟踪启动顺序。通过分析源代码，得到如下结论：

8.1 代码跟踪过程

在 StrutsPrepareAndExecuteFilter 过滤器类中的 init 方法的第一行下一个断点 ,因为 strtus2 框架是通过这个过滤器来启动的。

```
47          InitOperations init = new InitOperations();
48          try {
49              FilterHostConfig config = new FilterHostConfig(filterConfig);
50              init.initLogging(config);
51              Dispatcher dispatcher = init.initDispatcher(config);
52              init.initStaticContentLoader(config, dispatcher);
```

进入 initDispatcher 方法：

```
67      public Dispatcher initDispatcher( HostConfig filterConfig ) {
68          Dispatcher dispatcher = createDispatcher(filterConfig);
69          dispatcher.init();
70          return dispatcher;
71      }
```

进入 init 方法

```
404    public void init() {
405
406        if (configurationManager == null) {
407            configurationManager = new ConfigurationManager(Be
408        }
409
410        try {
411            init_DefaultProperties(); // [1]
412            init_TraditionalXmlConfigurations(); // [2]
413            init_LegacyStrutsProperties(); // [3]
414            init_CustomConfigurationProviders(); // [5]
415            init_FilterInitParameters() ; // [6]
416            init_AliasStandardObjects() ; // [7]
417
418            Container container = init_PreloadConfiguration();
419            container.inject(this);
420            init_CheckConfigurationReloading(container);
```

这个 init 方法中 ,首先创建了一个配置管理器(configurationManager),因为 struts2 有多个配置文件 ,有 properties 文件和 xml 文件 加载这些配置文件需要靠 XXXProvider 对象来完成，分别是: DefaultPropertiesProvider（用于加载 default.properties 文件），StrutsXmlConfigurationProvider(用于加载 *.xml 文件), LegacyPropertiesConfigurationProvider(用于加载 strtus.properties 文件),等等上面的【1】，【2】，【3】就是在将 xxxProvider 放入配置管理器中。真正的加载在下面的 418 行的 init_PreloadConfiguration（ ）方法中完成

```
373    private Container init_PreloadConfiguration() {
374        Configuration config = configurationManager.getConfiguration();
375        Container container = config.getContainer();
```

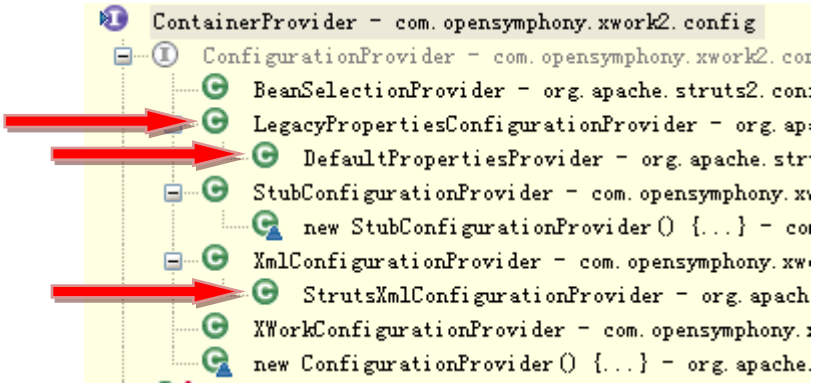
进入 getcontainner 方法，该方法在 ConfigurationManager 类中：

```
51    public synchronized Configuration getConfiguration() {
52        if (configuration == null) {
53            setConfiguration(new DefaultConfiguration(defaultFrameworkBeanName));
54            try {
55                configuration.reloadContainer(getContainerProviders());
56            } catch (ConfigurationException e) {
```

进入 reloadContainner 方法

```
162        for (final ContainerProvider containerProvider : providers)
163        {
164            containerProvider.init(this);
165            containerProvider.register(builder, props);
166        }
```

开始调用 XXXProvider 的 register 方法，注册的先后顺序就是配置文件的加载顺序: ContainerProvider 是一个接口，它被很多 XXXProvider 类实现了:



首先执行的是 DefaultPropertiesProvider 中的 register 方法:

```
40    public void register(ContainerBuilder builder, LocatableProperties props)
41            throws ConfigurationException {
42
43        Settings defaultSettings = null;
44        try {
45            defaultSettings = new PropertiesSettings("org/apache/struts2/defau
46        } catch (Exception e) {
47            throw new ConfigurationException("Could not find or error in org/a
48        }
49
50        loadSettings(props, defaultSettings);
```

接着是 StrutsXmlConfigurationProvider 中的 register 方法:

```
93    public void register(ContainerBuilder containe
94        if (servletContext != null && !containerBu
95            containerBuilder.factory(ServletContext
96                public ServletContext create(Conte
97                    return servletContext;
98                }
99            });
100       }
101       super.register(containerBuilder, props);
```

又调用了父类的 register 方法,在日志中会看到输出这句话，开始解析 xml 文件,包括 struts-default.xml 文件，strtus-plugin.xml,还有 strtus.xml 文件。

```
152   public void register(ContainerBuilder containerBuilder, LocatableProp
153       LOG.info("Parsing configuration file [" + configFileName + "]");
154       Map<String, Node> loadedBeans = new HashMap<String, Node>();
155       for (Document doc : documents) {
```

接着是 LegacyPropertiesConfigurationProvider 中的 register 方法，该方法中加载 struts.properties 文件，如果没有该文件，则输出日志:

```
55    public PropertiesSettings(String name) {
56
57        URL settingsUrl = ClassLoaderUtils.getResource(name + ".properties", getClass());
58
59        if (settingsUrl == null) {
60            LOG.debug(name + ".properties missing");
61            settings = new LocatableProperties();
62            return;
63        }
```

应用中使用 Log4j 查看日志输出情况，调整级别为 INFO 级别：

Time	Thread	Level	Category	Message
0	main	INFO	com.opensymphony.xwork2.config.providers.XmlConfigurationProvider	Parsing configuration file [struts-default.xml]
250	main	INFO	com.opensymphony.xwork2.config.providers.XmlConfigurationProvider	Unable to locate configuration files of the name struts-plugin.xml, skipping
250	main	INFO	com.opensymphony.xwork2.config.providers.XmlConfigurationProvider	Parsing configuration file [struts-plugin.xml]
266	main	INFO	com.opensymphony.xwork2.config.providers.XmlConfigurationProvider	Parsing configuration file [struts.xml]

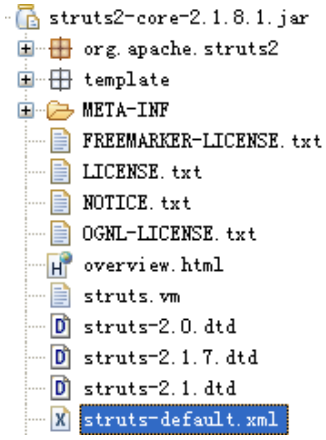
9. struts.xml—包配置

在 strtuts2 中，核心组件就是 Action,拦截器等，struts2 框架使用包来管理 Action 和拦截器等。每个包就是多个 Action，多个拦截器等的集合.package 中有下面几个属性：

- ☑ name：这是一个必填属性，该属性指定该包的名字，该名字是该包被其它包引用的 key
- ☑ extends:可选属性。指定该包继承其它包。继承其它包，可以继承其它包中的 Action 定义。
- ☑ abstract：可选属性。指定该包是不是一个抽象包。抽象包中不能包含 Action 定义。

※注意，父包应该在子包之前定义

在前面的配置中：继承了 stuts2 的默认包 struts-default,那么这个默认包在哪里定义的呢？我们可以查看 struts2-core-XXX.jar 包中有一个 struts-default.xml 文件



这个文件中配置了很多的<bean> 标签和一个<package>标签，<package>标签的 name 就是 struts-default.这个默认的包空间中定义了 struts2 内建的 Result 类型，拦截器等。Struts2 框架每次都会自动加载该文件。我们在 strtus.xml 文件中继承了默认的包空间，所以 struts-default.xml 文件一定比 strtus.xml 文件先加载.

只有继承了正确的父 Package，才能用到所需的预先配置好的特性。在大多数情况下，我们都应该继承“struts-default.xml”配置文件中的“strust-default”Package

10.struts.xml—Action 配置

Action 只是一个控制器，它并不直接对请求者生成任何响应。因此，Action 处理完用户请求后，Action 需要将指定的视图资源呈现给用户。因此，配置 Action 的时候，应该配置逻辑视图和物理视图资源之间的映射。

配置逻辑视图和物理视图之间的映射关系是通过<result>来定义的，每个<result>元素定义逻辑视图和物理视图之间的一次映射，如下面的配置:

```
<struts>
    <constant name="struts.action.extension" value="do"></constant>
    <package name="com.wq" extends="struts-default">
        <action name="login" class="com.wq.web.action.LoginAction">
            <result name="success">/welcome.jsp</result>
            <result name="error">/error.jsp</result>
        </action>
    </package>
</struts>
```

下面是 Action 配置中属性的说明：

- name:提供了执行 Action 所对应的 URL 地址，上面是“login.do” （我们已经将请求路径的后缀改成了*.do），默认是 login.action。
- class: Action 类的完整的类名

下面我们再看看<result>标记:

我们可以看到在 result 节点中多了“name”属性,实际上这个属性是一直都存在的，如果开发人员没有显式指定它的值，那么它的默认值就是“success”,所以上面的配置可以改成:

```
<struts>
    <constant name="struts.action.extension" value="do"></constant>
    <package name="com.wq" extends="struts-default">
        <action name="login" class="com.wq.web.action.LoginAction">
            <result >/welcome.jsp</result>
            <result name="error">/error.jsp</result>
        </action>
    </package>
</struts>
```