第 3 章 Struts2 开发

本章导读

Struts2 框架是 Apache 开源社区原有的 Struts 框架和 Open Symphony 社区 WebWork2 框架的合并版本，它集成了这两大流行的 MVC 框架各自的优点，主要以 WebWork 的设计思想为核心，提供了更加灵活的控制层和组件实现技术。

本章要点

- Struts2 的体系结构
- Struts2 的安装与配置
- Struts2 框架的主要配置文件

3.1 Struts2 结构

3.1.1 Struts2 体系结构

Struts2 框架提供了更灵活的控制层和组件实现技术，Struts2 框架主要的功能组件有 Action 组件、拦截器组件、国际化本地资源包以及 XML 配置文件等。图 3.1 为 Struts2 框架的体系结构图。

(1) HttpServletRequest 代表了浏览器客户端的一次 HTTP 请求和服务器程序处理结果的一次 HTTP 响应输出。

(2) ActionMapper 其实是 HttpServletRequest 和 Action 调用请求的一个映射，它屏蔽了 Action 对于 Request 等 Java 类的依赖。Struts2 中它的默认实现类是 DefaultActionMapper，ActionMapper 很大的用处可以根据自己的需要来设计 url 格式，它自己也有 Restful 的实现，具体可以参考文档的 docs\actionmapper. html。

(3) FilterDispatcher 代表 Struts2 框架的过滤器组件，是 Struts2 的核心控制器，负责拦截所有的客户端请求，通过 web. xml 文件被加入到 Web 应用当中，当有客户端请求到达时，它就会进行拦截，然后将根据配置文件将请求转发给相应的业务逻辑控制器进行处理。Struts2 框架包含一系列的标准过滤器组件链，该组件链主要由 ActionContextCleanUp 和核心过滤器组件 FilterDispatcher 构成。ActionContextCleanUp 主要应用在整合 SiteMesh 框架。

(4) Action 是 Struts2 的业务逻辑控制器，负责处理客户端的请求并将处理结果输出给客户端。

(5) ActionProxy 是 Action 的代理，由 ActionProxyFactory 创建，它本身不包括 Action

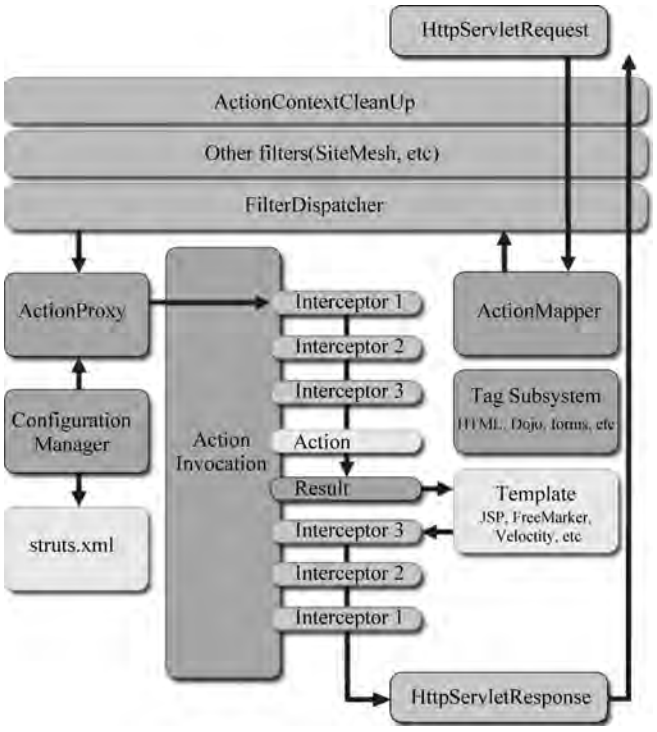

图 3.1　Struts2 体系结构

实例，DefaultActionProxy 是默认的 ActionProxy 代理。ActionProxy 作用是如何取得 Action，ActionProxy 创建一个 ActionInvocation 的实例，同时 ActionInvocation 通过代理模式调用 Action。但在调用之前 ActionInvocation 会根据配置加载 Action 相关的所有 Interceptor。该组件在 Struts2 框架中发挥着非常重要的作用。它是 action 和 xwork 中间的一层。正因为 ActionProxy 的存在导致 Action 调用更加简洁。

(6) ActionInvocation 是 Xworks 中 Action 调度的核心。ActionInvocation 是一个接口，它的作用是如何执行 Action，拦截器的功能就是在 ActionInvocation 中实现的。DefaultActionInvocation 是 Webwork 对 ActionInvocation 的默认实现。

(7) Interceptor 代表 Struts2 框架的拦截器组件，利用拦截器进行 AOP 编程(面向切面编程)，实现权限验证等功能。

(8) Template 是开发人员自己开发的各个部分的程序，Struts2 支撑多种表现层的技术，如 JSP、FreeMarker 等。

(9) ConfigurationManager 提供对客户端应用程序配置文件的访问，也就是 Struts2 中配置文件的解析器。

(10) Struts. xml 文件是 Struts2 框架的配置文件，主要负责配置业务逻辑控制器 Action，以及用户自定义的拦截器等，是 Struts2 各个组件之间的纽带。

3.1.2　工作流程

一个请求在 Struts2 框架中的基本流程如图 3.2 所示。

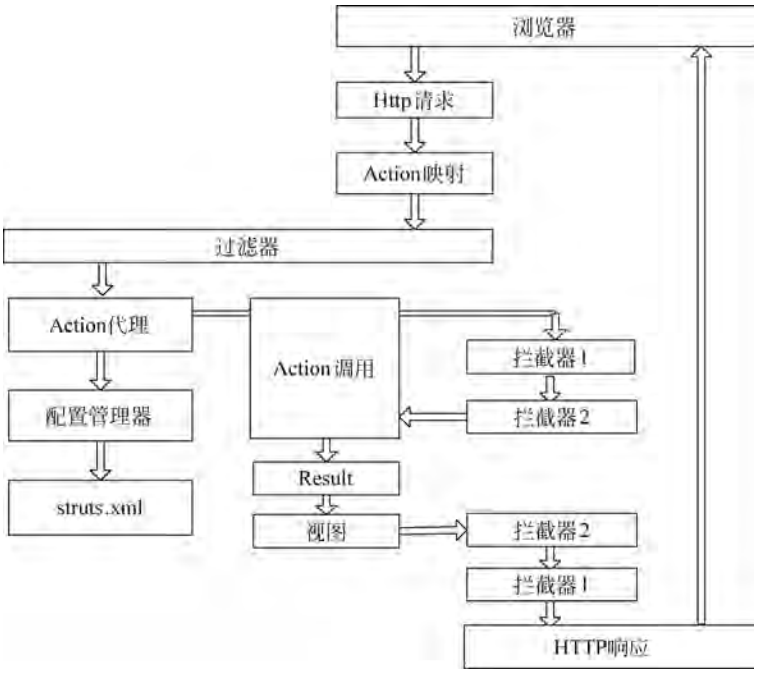

图 3.2　Struts2 工作流程

(1) 首先浏览器端发送一个 HttpServletRequest 请求。

(2) 核心控制器 StrutsPrepareAndExecuteFilter 根据请求决定调用合适的 Action。

(3) Struts2 的拦截器链自动对请求进行相关应用的拦截，如 validation(数据验证)或文件的上传下载等功能。

拦截器的调度流程大致为：ActionInvocation 初始化时，根据配置文件的设置，加载 Action 相关的所有 Interceptor，然后通过 ActionInvocation. invoke 方法调用 Action 实现。

(4) 回调 Action 的 execute 方法，该 execute 方法先获取用户请求参数，然后执行某种数据库的操作，既可以将数据保存到数据库，也可以从数据库中查询数据。实际上，Action 只是一个控制器，它会调用业务逻辑组件来处理用户的请求。

(5) Action 的 execute 方法将处理的结果存入 Stack Context 中，并返回一个字符串，核心控制器 StrutsPrepareAndExecuteFilter 将根据返回的字符串跳转到指定的视图资源，该视图资源将会读取 Stack Context 中的信息，并在浏览器生成响应数据。这些响应数据可以是 HTML 页面、图像、各种格式的文档等，并且所支持的视图技术也非常多，如 JSP、Velocity、FreeMarker 等模板技术。

在实际中，使用 Struts2 框架开发用户登录功能，需要用户创建一个登录界面 login. jsp，一个系统主页面 index. jsp，一个 Action 类 LoginAction. java，一个拦截器类 LoginInterceptor. java，另外需要对配置文件 web. xml、struts. xml 进行相应的设置。那么该模块的实际工作过程如图 3.3 所示。

用户在浏览器中输入 http://localhost:8080/Test/login. jsp 地址，调用 login. jsp 文件，在登录界面上填写相应的用户名和密码，然后提交请求给 Action 类处理，Struts2 框架根据配置文件 web. xml 里的设置对请求进行过滤，符合要求就交给 Struts2 框架，Struts2

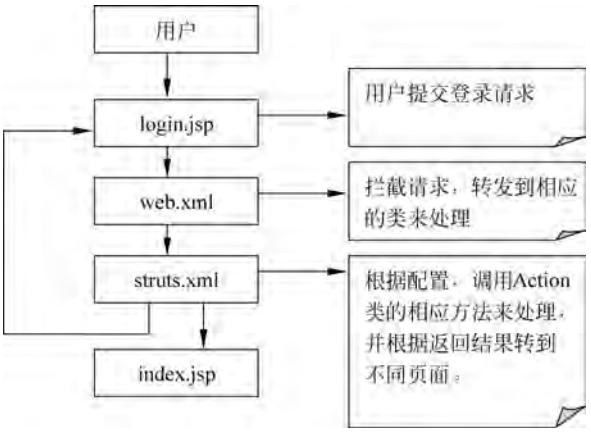

图 3.3　实际模块的工作过程

框架会根据配置文件 struts.xml 调用相应的 Action 类 LoginAction.java 来处理，如果该 Action 类定义了拦截器，那么就拦截请求，调用 LoginInterceptor.java 对用户名、密码等信息进行验证，然后再将控制权转移回 Action 类 LoginAction.java，如果处理结果为成功，则转向系统主页面 index.jsp，否则转回登录页面 login.jsp 重新输入值。

3.1.3　安装与配置

使用 Struts2 框架进行 web 开发或者运行 Struts2 的程序就必须先配置好 Struts2 的运行环境。首先配置 JDK 环境变量，然后安装 web 服务器，可以选择 Tomcat 作为运行的服务器。然后安装 Struts2 框架的 JAR 文件包。

方式 1　当开发工具不支持 Struts2 框架时，需要开发人员手工下载 JAR 包并添加到项目中去。

登录 http://struts.apache.org/网站下载 Struts2 完整版 struts-2.5-all.zip，将下载的 zip 文件解压缩，打开其文件夹，里面包含以下 4 个目录(见图 3.4)。

图 3.4　Struts-2.5 目录

不同版本的 Struts2 框架系统库，它们的核心库文件的文件名略有差别，Struts-2.5 版本主要有以下几个文件夹：

- apps 目录下的文件为 DEMO 示例，是学习 Struts2 非常有用的资料。
- docs 目录下的文件为系统的帮助文件。

- src 为 Struts2 框架源代码文件所在的目录。
- lib 包含 Struts2 框架的核心类库，以及 Struts2 的第三方插件类库。

打开 Struts2 开发包的 lib 文件夹，把 struts2-core-2.5.jar 、log4j-api-2.5.jar、ognl-3.1.4.jar、commons-logging-1.1.3.jar、freemarker-2.3.23.jar、commons-io-2.4.jar、commons-lang-2.4.jar、javassist-3.20.0.GA.jar、commons-fileupload-1.3.1.jar 拷贝到项目的\WebRoot\WEB-INF\lib 目录下。如果需要在 Web 应用中使用 Struts2 的更多特性，则需要将相应的 JAR 文件复制到 Web 应用的 WEB-INF/lib 目录下即可。

方式 2 当开发工具支持 Struts2 框架时，可以采用菜单项的方式安装 Struts2 框架的 JAR 包。本书使用 Myeclipse2015 作为开发工具，添加 Struts2 框架的步骤如下：

(1) 建立一个 Web 项目。打开 Myeclipse2015 建立一个 Web 项目，命名为“Struts2Test”，如图 3.5 所示。

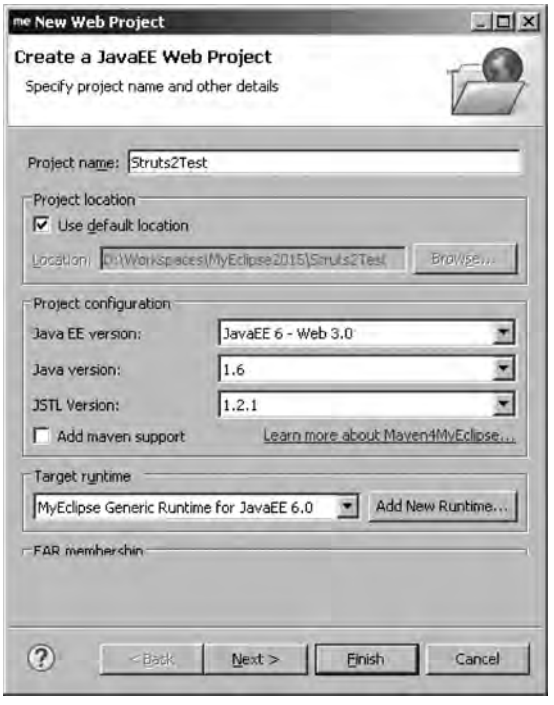

图 3.5 新建 Web 项目

(2) 加载 Struts2 类库。用鼠标右击项目名，在弹出的菜单项上选择【MyEclipse】(见图 3.6)，然后选择【Project Facets[Capabilities]】菜单，最后选择【Install Apache Struts (2.x) Facet】，弹出如图 3.7 所示对话框。

选择配置文件中对文件类型的过滤，*.action 表示将对扩展名是.action 的文件进行过滤；*.do 表示将对扩展名是.do 的文件进行过滤；/* 表示将对所有的文件进行过滤。

然后单击【Next】按钮，出现如图 3.8 所示对话框，选择要添加的 Struts2 类库文件，默认是 Core，单击【Finish】按钮完成。

Struts2 框架加载成功后，将该项目名展开，可以看到该项目下导入了很多 Struts2 类库，如图 3.9 所示。

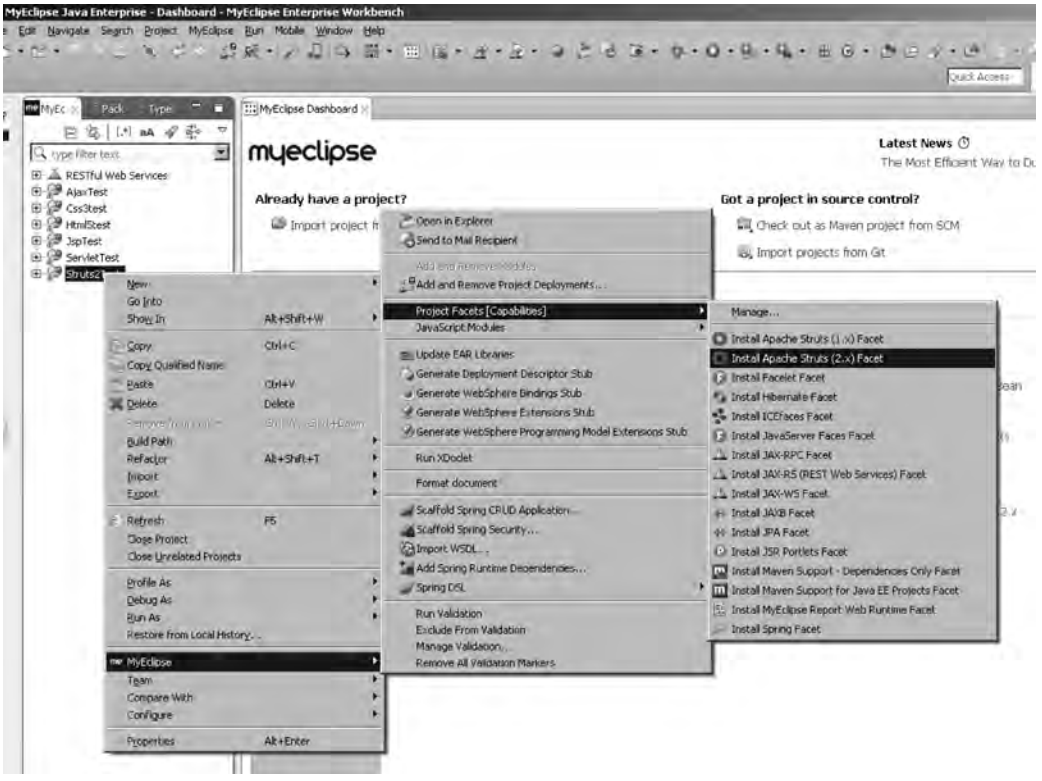

图 3.6　加载 Struts 2 类库

图 3.7　选择过滤文件类型

注意：web.xml 文件存储在 WEB-INF 目录下，struts.xml 存储在 classes 目录下，lib 下存储的是所必需的类库包，如 Struts2 核心类库包等。

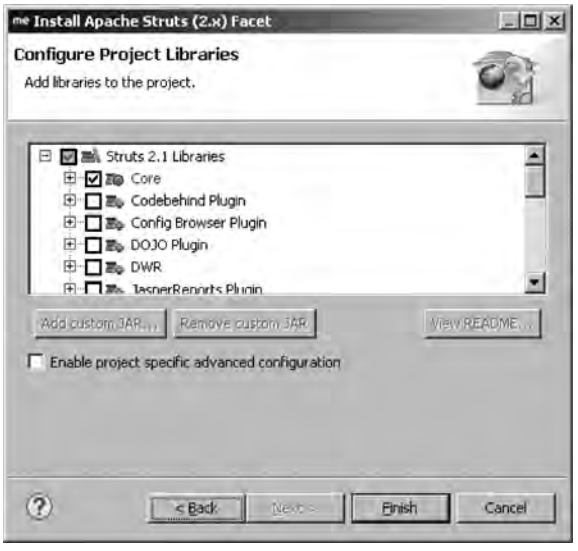

图 3.8　选择 Struts2 类库

图 3.9　加载后的包文件

3.2　配置文件 web.xml

Struts2 框架能够为程序提供良好的管理机制，这样开发人员就能够脱离繁杂的管理工作而专心于业务。然而框架并不可能知道开发人员的业务，只有开发人员把自己的业务通过配置文件注册给框架，框架才知道自己要管理的资源是什么。Struts2 框架中的配置文件 web.xml 由框架自动加载，对其自身进行配置。

3.2.1　文件的作用

Struts2 框架就是通过 web.xml 配置了核心控制器，核心控制器也就是过滤器，对用户请求和处理程序响应的内容进行处理。

在 Struts2 中，开发人员可以自定义过滤器，要求所有过滤器必须实现 java.Serlvet.Filter 接口，这个接口中含有 3 个过滤器类必须实现的方法：

- init(FilterConfig)：Servlet 过滤器的初始化方法，Servlet 容器创建 Servlet 过滤器实例后将调用这个方法。
- doFilter(ServletRequest,ServletResponse,FilterChain)：完成实际的过滤操作，当用户请求与过滤器关联的 URL 时，Servlet 容器将先调用过滤器的 doFilter 方法，返回响应之前也会调用此方法。FilterChain 参数用于访问过滤器链上的下一个过滤器。

- destroy()：Servlet 容器在销毁过滤器实例前调用该方法，这个方法可以释放 Servlet 过滤器占用的资源。

定义好过滤器后，需要在 web.xml 中进行配置。web.xml 文件存放在项目中 WebRoot/WEB-INF 的文件夹下。根据版本不同，里面的内容也有所不同。低版本的 web.xml 代码如下：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< web - app version = "2.5"
    xmlns = "http://java.sun.com/xml/ns/javaee"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema - instance"
    xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/JavaEE/web - app_2_5.xsd">
    < filter >
        < filter - name > struts 2 </filter - name >
        < filter - class > org.apache.struts2.dispatcher.FilterDispatcher </filter - class >
    </filter >
    < filter - mapping >
        < filter - name > struts 2 </filter - name >
        < url - pattern >/ * </url - pattern >
    </filter - mapping >
</web - app >
```

而对于 Struts2.1.6 及以上版本的框架，web.xml 代码如下：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< web - app version = "2.5"
    xmlns = "http://java.sun.com/xml/ns/javaee"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema - instance"
    xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web - app_2_5.xsd">
    < 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 > * .action </url - pattern >
    </filter - mapping >
</web - app >
```

从上述的例子可以看出，不同版本的 web.xml 文件主要有两处不同的地方：

(1)
```
< web - app version = "2.5"
    xmlns = "http://java.sun.com/xml/ns/javaee"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema - instance"
    xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/Javaee/web - app_2_5.xsd">
```

文件头中 Javaee 变成小写。

(2)
```
<filter-class>
    org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
</filter-class>
```

核心控制器的类名发生了变化，由原来的 org.apache.struts2.dispatcher.FilterDispatcher 变成 org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter。

3.2.2 常用属性

下面详细介绍 web.xml 文件中常用的一些属性。

```
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/Javaee/web-app_2_5.xsd">
```

这段代码是普通的 XML 文档定义，定义了当前的文件版本是 2.5 版本，通过 xmlns 引入了命名空间 http://java.sun.com/xml/ns/javaee，通过 xmlns:xsi 定义了 xml 遵循的标签规范，通过 xsi:schemaLocation 定义 xmlschema 的地址，也就是 xml 书写时需要遵循的语法，由两部分组成，前面部分就是命名空间的名字，后面是 xsd(xmlschema)的地址。

接下来是对核心控制器的定义，也可以是用户自定义的过滤器，基本语法格式如下：

```
<filter>
<filter-name>过滤器名</filter-name>
    <filter-class>过滤器对应类</filter-class>
    <init-param>
        <param-name>参数名称</param-name>
        <param-value>参数值</param-value>
    </init-param>
</filter>
<filter-mapping>
<filter-name>过滤器名</filter-name>
    <url-pattern>URL 关联方式</url-pattern>
</filter-mapping>
```

核心控制器的定义包含以下基本的元素：

- <filter>元素表示对过滤器进行定义。
- <filter-name>元素是<filter>元素的子元素，表示过滤器的名字，该名字用来对其具体的类进行调用。
- <filter-class>元素也是<filter>元素的子元素，表示过滤器具体的类存放的路径。
- <init-param>元素表示初始化参数的定义，其子元素<param-name>表示参数名称，<param-value>表示定义的参数值。
- <filter-mapping>元素指定让 Struts2 框架来处理用户的哪些请求(URL)。
- 子元素<filter-name>表示过滤器的名字，与前面<filter>元素的子元素的名字一致。

- <url-pattern>表示过滤器的 URL 映射规则。过滤器必须和特定的 URL 关联才能发挥作用。

当用户发送一个请求后，web.xml 中配置的 Struts2 核心控制器就会过滤该请求。过滤器的映射规则有 3 种：

(1) 完全匹配，例如下面代码：

```
<filter-mapping>
    <filter-name>过滤器名</filter-name>
    <url-pattern>xxx.action</url-pattern>
</filter-mapping>
```

(2) 路径匹配，例如下面代码匹配了根路径下的全部请求：

```
<filter-mapping>
    <filter-name>过滤器名</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

(3) 扩展名匹配，例如下面代码匹配了所有扩展名为.action 的文件：

```
<filter-mapping>
    <filter-name>过滤器名</filter-name>
    <url-pattern>*.action</url-pattern>
</filter-mapping>
```

如果用户的请求是以.action 结尾，过滤器就会进行过滤，将该请求转入 Struts2 框架处理。Struts2 框架接收到 *.action 请求后，将根据 *.action 请求前面的“*”来决定调用哪个业务。

另外，配置过滤器时，还可以指定一系列的初始化参数，主要的参数：

(1) config：该参数的值是一个以英文逗号“,”隔开的字符串，每个字符串都是一个 XML 配置文件。Struts2 框架将自动加载该属性指定的系列配置文件。如果没有指定该属性则默认使用 struts-default.xml、struts-plugin.xml、struts.xml 这三个配置文件。下面代码指定 Struts2 框架自动加载 mystruts2.xml 文件。

```
<init-param>
   <!-- 配置 Struts 2 的配置文件 -->
<param-name>config</param-name>
 <param-value>
mystruts2.xml
   </param-value>
</init-param>
```

(2) actionPackages，用来配置 Struts2 框架默认加载的 Action 包结构。参数的值是一个字符串类型的包空间，如果有多个包空间，可以用英文“,”符号隔开，这样，Struts2 框架将扫描指定的包空间下的所有 Action 类。

例如：

```
<filter>
<init-param>
    <param-name>actionPackages</param-name>
    <param-value>com.test.action</param-value>
</init-param>
</filter>
```

这样 Struts2 就会去 com.test.action 包下面找所有实现了 Action 的类。

(3) configProviders：如果用户需要实现自己的 ConfigurationProvider 类，用户可以提供一个或多个实现了 ConfigurationProvider 接口的类，然后将这些类的类名设置成该属性的值，多个类名之间以英文逗号“,”隔开。

例如：

```
<init-param>
<!-- 配置 Struts 2 框架的配置提供者类 -->
<param-name>configProviders</param-name>
   <param-value>com.struts2.test.web.MyConfigurationProvider</param-value>
</init-param>
```

(4) 配置 Struts2 常量，每个< init-param >元素配置一个 Struts2 常量，其中< param-name >子元素指定了常量 name，而< param-value >子元素指定了常量 value。在这里配置的常量等价于在 struts.properties 文件中配置的属性。

```
<init-param>
<!-- 配置 Struts2 框架的常量 -->
<param-name>truts.enable.DynamicMethodInvocation</param-name>
<param-value>false</param-value>
</init-param>
```

3.2.3 案例

案例 1 批量设置请求编码。

为了避免提交数据的中文乱码问题，需要在每次使用请求之前设置 request.setCharacterEncoding("gb2312")编码格式，这样确实很麻烦。Filter 可以批量拦截修改 servlet 的请求和响应。下面编写 EncodingFilter.java 文件，代码见例 3.1。

例 3.1 EncodingFilter.java

```
package com;
public class EncodingFilter implements Filter {
public void init(FilterConfig config) throws ServletException
{ }
public void destroy()
{ }
```

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException
  {
    request.setCharacterEncoding("gb2312");
chain.doFilter(request, response);
}
}
```

类 EncodingFilter 实现了 Filter 接口，Filter 接口中定义的三个方法都要在 EncodingFilter 中实现，其中 doFilter()的代码实现主要的功能：为请求设置 gb2312 编码，执行 chain.doFilter()会继续下面的操作。转换成对应 HttpServletRequest 和 HttpServletResponse 才能进行下面的 session 操作和页面重定向。

为了让 filter 发挥作用还需要在 web.xml 进行配置。见例 3.2。

例 3.2　web.xml 部分代码

```
<filter>
<filter-name>EncodingFilter</filter-name>
<filter-class>com.EncodingFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>EncodingFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Filter 标签用来定义过滤器，filter-mapping 标签告诉服务器把哪些请求交给过滤器处理，这里的“/*”表示所有请求，这样，所有的请求都会先被 EncodingFilter 拦截，并在请求里指定 gb2312 字符编码。

案例 2　用 filter 控制用户访问权限。

出于信息安全和其他一些原因的考虑，项目中的一些页面要求用户满足了一定条件之后才能访问。例如让用户输入账号和密码，如果输入的信息正确就在 session 里做一个成功的标记，这里的成功标志就是 session 中的 username 有值，其后在请求保密信息的时候判断 session 中是否有已经登录成功的标记，存在则可以访问，不存在则禁止访问。假设要保护的页面是 admin/index.jsp 编写 SecurityFilter.java，控制用户访问权限。代码见例 3.3 和例 3.4。

例 3.3　SecurityFilter.java

```
package com;
public class SecurityFilter implements Filter {
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {
HttpServletRequest req = (HttpServletRequest) request;
HttpServletResponse res = (HttpServletResponse) response;
HttpSession session = req.getSession();
   if (session.getAttribute("username") != null) {
   chain.doFilter(request, response);
```

```
}
else {
res.sendRedirect("../failure.jsp");
}
}
```

例 3.4　web.xml 部分代码

```
<filter>
<filter-name>SecurityFilter</filter-name>
<filter-class>com.SecurityFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>SecurityFilter</filter-name>
<url-pattern>/admin/*</url-pattern>
</filter-mapping>
```

定义 SecurityFilter 过滤器，让它过滤匹配/admin/*的所有请求，/admin/路径下的所有请求都会接受 SecurityFilter 的检查 因为 Filter 本来设计成为多种协议服务，http 协议仅仅是其中一种，将 ServletRequest 和 ServletResponse 转换成 HttpServletRequest 和 HttpServletResponse 才能进行下面的 session 操作和页面重定向。得到了 http 请求之后，可以获得请求对应的 session，判断 session 中的 username 变量是否为 null，如果不为 null，说明用户已经登录，就可以调用 doFilter 继续请求访问的资源。如果为 null，说明用户还没有登录，禁止用户访问，并使用页面重定向跳转到 failure.jsp 页面显示提示信息。因为/failure.jsp 的位置在/admin/目录的上一级，所以加上两个点才能正确跳转到 failure.jsp，两个点“..”代表当前路径的上一级路径。

3.3　配置文件 struts.properties

3.3.1　文件作用

Struts2 提供了很多可配置的属性，通过这些属性的设置，可以改变框架的行为，从而满足不同的 Web 应用的需求。这些属性可以在 struts.properties 文件中进行设置，该文件是 Struts2 框架的全局属性文件，也是自动加载的文件。

配置文件 struts.properties 是标准的 Java 属性文件格式，使用“#”号作为注释字符，该文件是由一系列的 key-value 对组成，每个 key 就是一个 Struts2 的属性，该 key 对应的 value 就是一个 Struts2 的属性，该文件位于 classpath 下，通常放在 Web 应用程序的/WEB-INF/classes 目录下。struts.properties 文件里的代码见例 3.5。

例 3.5　struts.properties

```
###指定加载 Struts2 配置文件管理器，默认为 org.apache.struts2.config.DefaultConfiguration
### 开发者可以自定义配置文件管理器，该类要实现 Configuration 接口，可以自动加载 struts2
配置文件.
# struts.configuration = org.apache.struts2.config.DefaultConfiguration
```

```
### 设置默认的 locale 和字符编码
# struts.locale = en_US
struts.i18n.encoding = UTF-8
### 指定 struts 的工厂类
# struts.objectFactory = spring
### 指定 spring 框架的装配模式
### 装配方式有: name, type, auto, and constructor (name 是默认装配模式)
struts.objectFactory.spring.autoWire = name
### 该属性指定整合 spring 时,是否对 bean 进行缓存,值为 true or false,默认为 true。
struts.objectFactory.spring.useClassCache = true
### 指定类型检查
#struts.objectTypeDeterminer = tiger
#struts.objectTypeDeterminer = notiger
### 该属性指定处理 MIME-type multipart/form-data,文件上传
# struts.multipart.parser = cos
# struts.multipart.parser = pell
struts.multipart.parser = jakarta
# 指定上传文件时的临时目录,默认使用 javax.servlet.context.tempdir
struts.multipart.saveDir =
struts.multipart.maxSize = 2097152
### 加载自定义属性文件 (不要改写 struts.properties!)
# struts.custom.properties = application,org/apache/struts2/extension/custom
### 指定请求 url 与 action 映射器,默认为 # org.apache.struts2.dispatcher.
mapper.DefaultActionMapper
#struts.mapper.class = org.apache.struts2.dispatcher.mapper.DefaultActionMapper
### 指定 action 的后缀,默认为 action
struts.action.extension = action
### 被 FilterDispatcher 使用
### 如果为 true 则通过 jar 文件提供静态内容服务。
### 如果为 false 则静态内容必须位于 <context_path>/struts
struts.serve.static = true
### 被 FilterDispatcher 使用
### 指定浏览器是否缓存静态内容,测试阶段设置为 false,发布阶段设置为 true。
struts.serve.static.browserCache = true
### 设置是否支持动态方法调用,true 为支持,false 不支持。
struts.enable.DynamicMethodInvocation = true
### 设置是否可以在 action 中使用斜线,默认为 false 不可以,想使用需设置为 true。
struts.enable.SlashesInActionNames = false
### 是否允许使用表达式语法,默认为 true。
struts.tag.altSyntax = true
### 设置当 struts.xml 文件改动时,是否重新加载。
### - struts.configuration.xml.reload = true
### 设置 struts 是否为开发模式,默认为 false,测试阶段一般设为 true。
struts.devMode = false
### 设置是否每次请求,都重新加载资源文件,默认值为 false。
struts.i18n.reload = false
###标准的 UI 主题
### 默认的 UI 主题为 xhtml,可以为 simple、xhtml 或 ajax
struts.ui.theme = xhtml
###模板目录
```

```
struts.ui.templateDir = template
#设置模板类型。可以为 ftl、vm 或 jsp
struts.ui.templateSuffix = ftl
###定位 velocity.properties 文件。默认 velocity.properties
struts.velocity.configfile = velocity.properties
### 设置 velocity 的 context。
struts.velocity.contexts =
### 定位 toolbox。
struts.velocity.toolboxlocation =
### 指定 web 应用的端口。
struts.url.http.port = 80
### 指定加密端口
struts.url.https.port = 443
### 设置生成 url 时,是否包含参数。值可以为: none、get 或 all
struts.url.includeParams = get
### 设置要加载的国际化资源文件,以逗号分隔。
# struts.custom.i18n.resources = testmessages,testmessages2
### 对于一些 web 应用服务器不能处理 HttpServletRequest.getParameterMap()
### 像 WebLogic、Orion 和 OC4J 等,须设置成 true,默认为 false。
struts.dispatcher.parametersWorkaround = false
### 指定 freemarker 管理器
# struts.freemarker.manager.classname = org.apache.struts2.views.
freemarker.FreemarkerManager
### 设置是否对 freemarker 的模板设置缓存
### 效果相当于把 template 拷贝到 WEB_APP/templates。
struts.freemarker.templatesCache = false
### 通常不需要修改此属性。
struts.freemarker.wrapper.altMap = true
### 指定 xslt result 是否使用样式表缓存。开发阶段设为 true,发布阶段设为 false。
struts.xslt.nocache = false
### 设置 struts 自动加载的文件列表。
struts.configuration.files = struts-default.xml,struts-plugin.xml,struts.xml
### 设定是否一直在最后一个 slash 之前的任何位置选定 namespace。
struts.mapper.alwaysSelectFullNamespace = false
```

3.3.2 常用属性

该文件中包含很多属性,下面进行介绍:

struts.configuration:该属性指定加载 Struts2 配置文件的文件管理器。默认值是 org.apache.struts2.config.DefaultConfiguration。

struts.locale:设置 Web 应用的默认 Locale。

struts.i18n.encoding:设置 Struts2 应用编码的默认使用字符集,如需获取中文请求参数值,应该将该常量值设置为 GBK 或者 GB2312。

struts.objectFactory:该属性默认值是 spring,设置 Struts2 默认的 ObjectFactory Bean。

struts.objectFactory.spring.autoWire:设置 Spring 框架的自动装配模式,该属性默认值是 name,即默认根据 Bean 的 name 进行自动装配。

struts.objectFactory.spring.useClassCache：该属性指定整合 Spring 框架时，是否缓存 Bean 实例，该属性只允许使用 true 和 false 两个属性值，默认值是 true，通常不建议修改该属性的值。

struts.objectTypeDeterminer：该属性指定 Struts2 的类型检测机制，通常支持 tiger 和 notiger 两个属性值。

struts.multipart.parser：该属性指定处理 multipart/form-data 的 MIME 类型（文件上传）请求的框架，该常量支持 cos、pell 和 jakarta 等常量值，即分别对应使用 cos 的文件上传框架、pell 上传及 common-fileupload 文件上传框架。该属性的默认值为 jakarta。如果需要使用 cos 或者 pell 的文件上传方式，则应该将对应的 JAR 文件复制到 Web 应用中。例如，使用 cos 上传方式，则需要自己下载 cos 框架的 JAR 文件，并将该文件放在 WEB-INF/lib 路径下。

struts.multipart.saveDir：该属性的默认值是 javax.servlet.context.tempdir，该属性指定上传文件的临时保存路径。

struts.multipart.maxSize：该属性设置 Struts2 文件上传中整个请求内容允许的最大字节数。

struts.custom.properties：该属性指定 Struts2 应用加载用户自定义的属性文件，该属性文件配置的常量不会覆盖 struts.properties 文件中配置的常量。如果需要加载多个自定义属性文件，多个自定义属性文件的文件名应以英文逗号“,”隔开。

struts.mapper.class：指定将 HTTP 请求映射到指定 Action 的映射器，Struts2 提供了默认的映射器：org.apache.struts2.dispatcher.mapper.DefaultActionMapper。默认映射器根据请求的前缀与 Action 的 name 属性完成映射。

struts.action.extension：该属性指定需要 Struts2 处理的请求后缀，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：该属性设置是否通过 JAR 文件提供静态内容服务，该属性只支持 true 和 false 属性值，该属性的默认属性值是 true。

struts.serve.static.browserCache：该属性设置浏览器是否缓存静态内容。当应用处于开发阶段时，如果希望每次请求都获得服务器的最新响应，则可设置该属性为 false。

struts.enable.DynamicMethodInvocation：该属性设置 Struts2 是否支持动态方法调用，该属性的默认值是 true。如果需要关闭动态方法调用，则可设置该属性为 false。

struts.enable.SlashesInActionNames：该属性设置 Struts2 是否允许在 Action 名中使用斜线，该属性的默认值是 false。如果希望允许在 Action 名中使用斜线，则可设置该属性为 true。

struts.tag.altSyntax：该属性指定是否允许在 Struts2 标签中使用表达式语法，因为通常都需要在标签中使用表达式语法，故此属性应该设置为 true，该属性的默认值是 true。

struts.devMode：该属性设置 Struts2 应用是否使用开发模式。如果设置该属性为 true，则可以在应用出错时显示更多、更友好的出错提示。该属性只接受 true 和 flase 两个值，该属性的默认值是 false。通常，应用在开发阶段，将该属性设置为 true，当进入产品发布阶段后，则该属性设置为 false。

struts.i18n.reload 该属性设置是否每次 HTTP 请求到达时，系统都重新加载资源文件。该属性默认值是 false。在开发阶段将该属性设置为 true 会更有利于开发，但在产品发布阶段应将该属性设置为 false。开发阶段将该属性设置了 true，将可以在每次请求时都重新加载国际化资源文件，从而可以让开发者看到实时开发效果；产品发布阶段应该将该属性设置为 false，是为了提供响应性能，每次请求都需要重新加载资源文件会大大降低应用的性能。

struts.ui.theme：该属性指定视图标签默认的视图主题，该属性的默认值是 xhtml。

struts.ui.templateDir：该属性指定视图主题所需要模板文件的位置，该属性的默认值是 template，即默认加载 template 路径下的模板文件。

struts.ui.templateSuffix：该属性指定模板文件的后缀，该属性的默认属性值是 ftl。该属性还允许使用 ftl、vm 或 jsp，分别对应 FreeMarker、Velocity 和 JSP 模板。

struts.configuration.xml.reload：该属性设置当 struts.xml 文件改变后，系统是否自动重新加载该文件。该属性的默认值是 false。

struts.velocity.configfile：该属性的默认值为 velocity.properties。该属性指定 Velocity 框架所需的 velocity.properties 文件的位置。

struts.velocity.contexts：该属性指定 Velocity 框架的 Context 位置，如果该框架有多个 Context，则多个 Context 之间以英文逗号“,”隔开。

struts.velocity.toolboxlocation：该属性指定 Velocity 框架的 toolbox 的位置。

struts.url.http.port：该属性指定 Web 应用所在的监听端口。该属性通常没有太大的用途，只是当 Struts2 需要生成 URL 时（例如 url 标签），该常量才提供 Web 应用的默认端口。

struts.url.https.port：该属性类似于 struts.url.http.port 常量的作用，区别是该常量指定的是 Web 应用的加密服务端口。

struts.url.includeParams：该属性指定 Struts2 生成 URL 时是否包含请求参数。该属性接受 none、get 和 all 三个值，分别对应于不包含、仅包含 GET 类型请求参数和包含全部请求参数。

struts.custom.i18n.resources：该属性指定 Struts2 应用所需要的国际化资源文件，如果有多份国际化资源文件，则多个资源文件的文件名以英文逗号“,”隔开。

struts.dispatcher.parametersWorkaround：该属性的默认值是 false。对于某些 Java EE 服务器，不支持 HttpServlet Request 调用 getParameterMap()方法，此时可以设置该常量值为 true 来解决该问题。对于 WebLogic、Orion 和 OC4J 服务器，通常应该设置该常量为 true。

struts.freemarker.manager.classname：该属性指定 Struts2 使用的 FreeMarker 管理器。该属性的默认值是 org.apache.struts2.views.freemarker.FreemarkerManager，这是 Struts2 内建的 FreeMarker 管理器。

struts.freemarker.wrapper.altMap：该属性只支持 true 和 false 两个值，默认值是 true，通常无须修改该常量值。

struts.xslt.nocache：该属性指定是否关闭 XSLT Result 的样式表缓存。当应用处于开发阶段时，该常量通常被设置为 true；当应用处于产品使用阶段时，该常量通常被设置为

false。

struts.configuration.files：该属性指定 Struts2 框架默认加载的配置文件，如果需要指定多个默认加载的配置文件，则多个配置文件的文件名之间以英文逗号“,”隔开。默认值是 struts-default.xml,struts-plugin.xml,struts.xml，所以 Struts2 框架默认加载 struts.xml 文件。

实际应用中，struts.properties 不推荐使用，因为所有的设置都可以通过 struts.xml 里的常量来实现。

3.4 配置文件 struts.xml

3.4.1 文件作用

配置文件 struts.xml 是 Struts2 框架的核心配置文件，主要用于配置和管理开发人员编写的 Action，以及 Action 包含的 result 定义、Bean 的配置、常量的配置、包的配置和作用于 action 的拦截器的配置等。

该文件通常放在 Web 应用程序的 Web-INF/classes 目录下，将被 Struts2 框架自动加载。struts.xml 文件的基本结构如下所示：

```
<!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 必须放在一个指定的包空间下定义 -->
<package name="default" extends="struts-default">
<action name="login" class="org.qiujy.web.struts.action.LoginAction">
<result name="success">/success.jsp</result>
<result name="error">/error.jsp</result>
</action>
</package>
</struts>
```

前 3 行是 XML 的头，定义了基本信息。文件的根元素是<struts>标签，其他标签都是包含在它里面的，很多元素都是可以重复定义的，以配置不同的内容。

3.4.2 常用属性

1. 包配置(package)

Struts2 框架使用包来管理 Action 和拦截器等，每个 package 就是多个 Action、多个拦截器、多个拦截器引用的集合，从而简化维护工作，提高了代码的重用性。另外，package 元素可以扩展其他的包，从而“继承”原有包的所有定义，也可以添加自己的包特有的配置，以及修改原有包的部分配置。

Package 元素的常用属性如表 3.1 所示。

表 3.1　Package 的常用属性

属　性　名	必选/可选	说　　明
name	必选	指定包名，这个名字将作为引用该包的键。注意，包的名字必须是唯一的，在一个 struts. xml 文件中不能出现两个同名的包。
extends	可选	允许一个包继承一个或多个先前定义的包。
abstract	可选	将其设置为 true，可以把一个包定义为抽象的。抽象包不能有 action 定义，只能作为“父”包被其他包所继承。
namespace	可选	将保存的 action 配置为不同的命名空间。

例如：配置一个名为 stu 的包的代码如下：

```
<package name = "stu" extends = "struts - default">
```

2. 命名空间的配置(namespace)

Struts2 以命名空间的方式来管理 Action，主要针对大型项目 Action 重名的问题，因为不在同一个命名空间的 Action 可以使用相同的 Action 名字，同一个命名空间不能有同名的 Action。代码见例 3.6。

例 3.6　命名空间示例部分代码

```
<!-- default 包在默认的命名空间中 -->
<package name = "default" extends = "struts - default">
<action name = "foo" class = "cn.com.web.LoginAction">
        <result name = "success">/foo.jsp</result>
    </action>
<action name = "bar" class = "cn.com.web. LoginAction></action>
</package>
<!-- mypackage1 包在根命名空间中 -->
<package name = "mypackage1" namespace = "/">
    <action name = "moo" class = "cn.com.web.LoginActionTwo">
        <result name = "success">/moo.jsp</result>
    </action>
</package>
<!-- mypackage2 包在/accp 命名空间中 -->
<package name = "mypackage2" namespace = "/accp">
<action name = "foo" class = "cn.com.web.LoginActionThree">
<result name = "success">/foo2.jsp</result>
</action>
</package>
```

当发起/moo. action 请求时，框架会在根命名空间“/”中查找 moo. action，如果没找到再到默认命名空间下查找。在此例中，Mypackage 中存在 moo. action，因此执行 cn. com. web. LoginActionTwo 类。

如果发起/accp/foo. action 请求时，框架会在/accp 命名空间查找 foo. action，找到后执行 cn. com. web. LoginActionThree 类。

如果发起/accp/bar. action 请求时，框架会在/accp 命名空间下查找 bar. action，没有找到，

那么转到默认命名空间下查找，此时查找 bar.action 文件，执行 cn.com.web.LoginAction 类。

3. Action 的配置

Action 主要配置 action 类的调用，Struts2 框架的核心功能是 Action，开发好 action 后，需要在 struts.xml 中进行配置，action 元素配置 action 的物理路径以及映射路径，用来告诉 Struts2 框架，针对某个 URL 的请求应该交给哪个 action 类进行处理。详细的属性在第 4 章介绍。

```
<action name="login" class="org.qiujy.web.struts.action.LoginAction">
<result name="success">/success.jsp</result>
<result name="error">/error.jsp</result>
</action>
```

name 表示 action 的名字，Struts2 框架根据 action 的名字查找相应的类，调用时 class 表示 action 类具体存放的物理路径。

4. 包含配置(include)

利用 include 元素，可以将一个 struts.xml 配置文件分割成多个配置文件，然后在 struts.xml 中使用 include 元素引入其他配置文件。比如一个网上购物程序，可以把用户配置、商品配置、订单配置分别放在 3 个配置文件 user.xml、goods.xml 和 order.xml 中，然后在 struts.xml 中将这 3 个配置文件引入：

```
<struts>
    <include file="user.xml"/>
    <include file="goods.xml"/>
    <include file="order.xml"/>
</struts>
```

5. 常量配置(constant)

通过 struts.xml 文件中的常量配置，可以指定 Struts2 框架的属性，其实这些属性也可以在其他配置文件中指定，例如在 web.xml 配置文件的< init-param >元素中可以指定常量，也可以在 struts.properties 文件中定义属性来实现。反过来，struts.properties 配置文件中的所有属性都可以通过< constant >标记配置在 struts.xml 中：

```
<struts>
    <!-- 设置开发模式 -->
    <constant name="struts.devMode" value="true"/>
    <!-- 设置编码形式为 GB2312 -->
    <constant name="struts.i18n.encoding" value="GB2312"/>
    <!-- 省略其他配置信息 -->
</struts>
```

6. Bean 的配置

Struts2 框架是一个具有高度可扩展性的框架，其大部分的核心组件都不是以直接编码的方式写在代码中的，而是以可配置的方式来管理 Struts2 的核心组件，这样，就使得这些核心组件具有可插可拔的功能，降低了代码的耦合度。

当开发人员需要扩展该框架的核心组件，或者替换 Struts2 的核心组件时，只需要提供自己的组件实现类，并将该组件实现类部署在 struts.xml 中就可以。使用< bean >元素在 struts.xml 文件中定义 Bean，通常有如下两个作用。

(1) 创建该 Bean 的实例，将该实例作为 Struts2 框架的核心组件使用。

(2) Bean 包含的静态方法需要注入一个值。

例如下面的 struts.xml 文件中的 Bean 配置，该 Bean 实现了 ObjectFactory 接口，实现类是 MyObjectFactory。配置代码片段如下：

```
< struts >
    < bean type = "com.opensymphony.xwork2.ObjectFactory" name = "myfactory"
    class = "com.opensymphony.xwork2.myapp.MyObjectFactory"/>
</struts >
```

bean 元素有如下几个属性：

- class：必填属性，它指定 Bean 实例的实现类。
- type：可选属性，它指定 Bean 实例实现的 Struts2 的规范，该规范通常是通过某个接口来体现，因此该属性的值通常是一个 Struts2 接口。如果需要将 Bean 实例作为 Struts2 组件来使用，则应该指定该属性的值。
- name：可选属性，该属性指定 Bean 实例的名字，对于有相同 type 类型的多个 Bean，它们的 name 属性不能相同。
- scope：可选属性，该属性指定 Bean 实例的作用域，属性值只能是 default、singleton、request、session 或者 thread 之一。
- static：可选属性，该属性指定 Bean 是否使用静态方法注入，通常而言，当指定了 type 属性时，该属性值不应该指定为 true。
- optional：可选属性，该属性指定该 Bean 是否是一个可选的 Bean。

3.4.3 案例

案例 3 这是一个比较完整的 struts.xml 文件，里面演示了各种元素的使用方法与用途，并进行了详细的注释。代码见例 3.7。

例 3.7 struts.xml

```
<?xml version = "1.0" encoding = "GBK"?>
<!-- 下面指定 Struts 2.1 配置文件的 DTD 信息 -->
<!DOCTYPE struts PUBLIC
    " - //Apache SoftwareFoundation//DTD Struts Configuration 2.1//EN"
    "http://struts.apache.org/dtds/struts - 2.1.dtd">
<!-- struts 是 Struts 2 配置文件的根元素 -->
< struts >
    <!-- 下面元素可以出现零次，也可以出现无数次 -->
    < constant name = "" value = "" />
    <!-- 下面元素可以出现零次，也可以出现无数次 -->
    < bean type = "" name = "" class = "" scope = "" static = ""optional = "" />
    <!-- 下面元素可以出现零次，也可以出现无数次 -->
```

```
<include file = "" />
<!-- package 元素是 Struts 配置文件的核心,该元素可以出现零次,或者无数次 -->
<package name = "必填的包名" extends = "" namespace = "" abstract = ""
    externalReferenceResolver >
    <!-- 该元素可以出现,也可以不出现,最多出现一次 -->
    <result - types >
        <!-- 该元素必须出现,可以出现无数次 -->
        <result - type name = ""class = "" default = "true|false">
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <param name = "参数名">参数值</param > *
        </result - type >
    </result - types >
    <!-- 该元素可以出现,也可以不出现,最多出现一次 -->
    <interceptors >
        <!-- 该元素的 interceptor 元素和 interceptor - stack 至少出现其中之一,
        也可以二者都出现 -->
        <!-- 下面元素可以出现零次,也可以无数次 -->
        <interceptor name = ""class = "">
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <param name = "参数名">参数值</param > *
        </interceptor >
        <!-- 下面元素可以出现零次,也可以无数次 -->
        <interceptor - stack name = "">
            <!-- 该元素必须出现,可以出现无数次 -->
            <interceptor - ref name = "">
                <!-- 下面元素可以出现零次,也可以无数次 -->
                <param name = "参数名">参数值</param > *
            </interceptor - ref >
        </interceptor - stack >
    </interceptors >
    <!-- 下面元素可以出现零次,也可以无数次 -->
    <default - interceptor - ref name = "">
        <!-- 下面元素可以出现零次,也可以无数次 -->
        <param name = "参数名">参数值</param >
    </default - interceptor - ref >
    <!-- 下面元素可以出现零次,也可以无数次 -->
    <default - action - ref name = "">
        <!-- 下面元素可以出现零次,也可以无数次 -->
        <param name = "参数名">参数值</param > *
    </default - action - ref >?
    <!-- 下面元素可以出现零次,也可以无数次 -->
    <global - results >
        <!-- 该元素必须出现,可以出现无数次 -->
        <result name = ""type = "">
            <!-- 该字符串内容可以出现零次或多次 -->
            映射资源
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <param name = "参数名">参数值</param > *
        </result >
    </global - results >
```

```
        <!-- 下面元素可以出现零次,也可以无数次 -->
        <global-exception-mappings>
            <!-- 该元素必须出现,可以出现无数次 -->
            <exception-mapping name=""exception="" result="">
                异常处理资源
                <!-- 下面元素可以出现零次,也可以无数次 -->
                <param name="参数名">参数值</param>*
            </exception-mapping>
        </global-exception-mappings>
        <action name=""class="" method="" converter="">
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <param name="参数名">参数值</param>*
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <result name=""type="">
                映射资源
                <!-- 下面元素可以出现零次,也可以无数次 -->
                <param name="参数名">参数值</param>*
            </result>
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <interceptor-ref name="">
                <!-- 下面元素可以出现零次,也可以无数次 -->
                <param name="参数名">参数值</param>*
            </interceptor-ref>
            <!-- 下面元素可以出现零次,也可以无数次 -->
            <exception-mapping name=""exception="" result="">
                异常处理资源
                <!-- 下面元素可以出现零次,也可以无数次 -->
                <param name="参数名">参数值</param>*
            </exception-mapping>
        </action>
    </package>*
    <!-- unknown-handler-stack 元素可出现零次或 1 次 -->
    <unknown-handler-stack>
        <!-- unknown-handler-ref 元素可出现零次或多次 -->
        <unknown-handler-ref name=" ">...</unknown-handler-ref>*
    </unknown-handler-stack>?
<struts>
```

思考与练习

(1) 简述 Struts2 体系结构?

(2) 命名空间是什么? 怎样配置?

(3) Struts.xml 文件有什么作用?