

图2.1 struts2.0体系[**架构**](http://lib.csdn.net/base/16)图

针对上节体系架构图，下面分步说明执行流程

Ø  客户端初始化一个指向Servlet容器（例如Tomcat）的请求；

Ø  这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin）；

注：从struts2.1.3后就不需要配置ActionContextCleanUp过滤器了，该过滤器的作用是清空值栈等，但struts会留下自己session和Attribute等

Ø  接着调用struts的StrutsPrepareAndExecuteFilter过滤器，StrutsPrepareAndExecuteFilter询问ActionMapper来决定这个请求是否需要调用某个Action；

备注：早期struts用FilterDispatcher过滤器，现在普遍都用StrutsPrepareAndExecuteFilter。 就名字而言，prepare与execute前者表示准备说指filterinit方法即配置导入；后者表示进行过滤指doFilter方法即request请求转发给对应action去处理

Ø  如果ActionMapper决定需要调用某个Action，StrutsPrepareAndExecuteFilter把请求的处理交给ActionProxy；

Ø  ActionProxy通过Configuration Manager询问框架的配置文件，找到需要调用的Action类；

Ø  ActionProxy创建一个ActionInvocation的实例。

Ø  ActionInvocation实例使用命名模式来调用，在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用。

Ø  一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果。返回结果通常是（但不总是，也可能是另外的一个Action链）一个需要被表示的JSP或者FreeMarker的模版。在表示的过程中可以使用Struts2框架中继承的标签。在这个过程中需要涉及到ActionMapper。

官方文档：<https://struts.apache.org/docs/the-struts-2-request-flow.html>

参考：http://blog.csdn.net/sxb0841901116/article/details/27552315

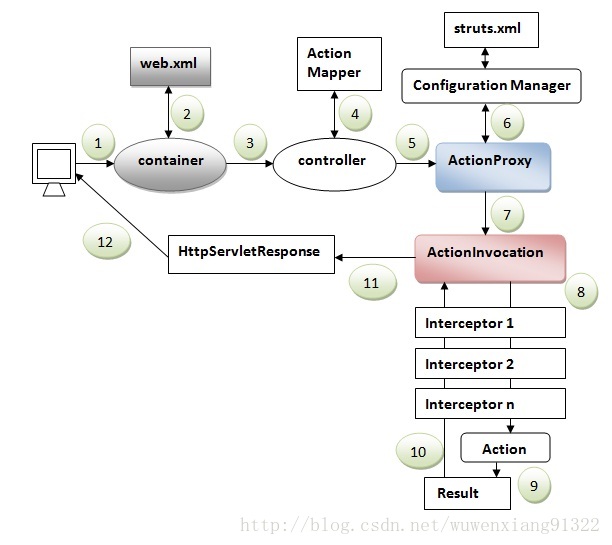
参考：<http://wiki.jikexueyuan.com/project/ssh-noob-learning/Struts2-internal-work.html>

一篇基本总结《Struts技术内幕》的博客：<http://blog.csdn.net/shan9liang/article/details/9281967>

另一篇博客，地址：<http://blog.csdn.net/wuwenxiang91322/article/details/11070513>

**Struts2请求响应流程：**

在struts2的应用中，从用户请求到服务器返回相应响应给用户端的过程中，包含了许多组件如：Controller、ActionProxy、ActionMapping、Configuration Manager、ActionInvocation、Inerceptor、Action、Result等。下面我们来具体看看这些组件有什么联系，它们之间是怎样在一起工作的。



（1）  客户端（Client）向Action发用一个请求（Request）

（2）  [**Container**](http://lib.csdn.net/base/4)通过web.xml映射请求，并获得控制器（Controller）的名字

（3）  容器（Container）调用控制器（StrutsPrepareAndExecuteFilter或FilterDispatcher）。在Struts2.1以前调用FilterDispatcher，Struts2.1以后调用StrutsPrepareAndExecuteFilter

（4）  控制器（Controller）通过ActionMapper获得Action的信息

（5）  控制器（Controller）调用ActionProxy

（6）  ActionProxy读取struts.xml文件获取action和interceptor stack的信息。

（7）  ActionProxy把request请求传递给ActionInvocation

（8）  ActionInvocation依次调用action和interceptor

（9）  根据action的配置信息，产生result

（10） Result信息返回给ActionInvocation

（11） 产生一个HttpServletResponse响应

（12） 产生的响应行为发送给客服端。