

当接收到一个httprequest ,   
a) 当外部的httpservletrequest到来时   
b) 初始到了servlet容器 传递给一个标准的过滤器链   
c) FilterDispatecher会去查找相应的ActionMapper，如果找到了相应的ActionMapper它将会将控制权限交给ActionProxy   
d) ActionProxy将会通过ConfigurationManager来查找配置struts.xml   
       i. 下一步将会 通过ActionInvocation来负责命令模式的实现（包括调用一些拦截Interceptor框架在调用action之前）   
       ii. Interceptor做一些拦截或者初始的工作   
e) 一旦action返回，会查找相应的Result   
f) Result类型可以是 jsp或者freeMark 等   
g) 这些组件和ActionMapper一起返回给请求的url（注意拦截器的执行顺序）   
h) 响应的返回是通过我们在web.xml中配置的过滤器   
i) 如果ActionContextCleanUp是当前使用的，则FilterDispatecher将不会清理sreadlocal ActionContext;如果ActionContextCleanUp不使用，则将会去清理sreadlocals。

1、拦截器是基于**[Java](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/yakoo5/article/details/_blank)**反射机制的，而过滤器是基于函数回调的。  
2、过滤器依赖于servlet容器，而拦截器不依赖于servlet容器。  
3、拦截器只能对Action请求起作用，而过滤器则可以对几乎所有请求起作用。  
4、拦截器可以访问Action上下文、值栈里的对象，而过滤器不能。  
5、在Action的生命周期中，拦截器可以多次调用，而过滤器只能在容器初始化时被调用一次。

org.apache.struts2.dispatcher.FilterDispatcher的四个功能

org.apache.struts2.dispatcher.FilterDispatcher是Struts2的主要的Filter，负责四个方面的功能：  
      **(1)执行Actions  
        (2)清除ActionContext  
        (3)维护静态内容  
        (4)清除request生命周期内的XWork的interceptors**  
               另注：该过滤器应该过滤所有的请求URL。一般被设置为/\*.(过滤所有的请求！)  
    具体：  
        (1)执行Actions  
            过滤器通过ActionMapper对象，来判断是否应该被映射到Action.如果mapper对象指示他应该被映射，过滤链将会被终止，然后Action被调用。这一点非常重要，如果同时使用SiteMesh filter，则SiteMesh filter应该放到该过滤器前，否则Action的输出将不会被装饰。  
        (2)清除ActionContext  
            过滤器为了确保内存溢出，会自动的清除ActionContext。这可能会存在一些问题，在和其它的框架集成时，例如SiteMesh。ActionContextCleanUp提供了怎么处理这些问题的一些信息。  
        (3)维护静态内容  
            过滤器也会维护在Struts2中使用的一些公共的静态的内容，例如**[JavaScript](http://lib.csdn.net/base/javascript" \o "JavaScript知识库" \t "http://blog.csdn.net/yakoo5/article/details/_blank)**文件，CSS文件等。搜索/struts/\*范围内的请求，然后将/struts/后面的值映射到一些struts的公共包中，也可以在你的类路径中搜索。默认情况下会去查找以下包：org.apache.struts2.static.template。这样你只用请求/struts/xhtml/styles.css，XHTML UI主题默认的样式表将会被返回。同样，AJAX UI组件需要的JavaScript文件，也可以在org.apache.struts2.static包中被找到。如果你想加入其它被搜索的包，在web.xml中设置filter时，通过给"actionPackages"初始参数一个逗号隔开的包列表值来设定。  
            需注意的是：他会暴露一些比较敏感的信息例如，properites文件中的**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/yakoo5/article/details/_blank)**连接信息。  
    注：过滤器支持以下初始参数：  
         config - 被调入的逗号隔开的XML文件列表。  
         actionPackages - 被actions扫描的逗号隔开的packages列表。  
         configProviders - 逗号分隔的实现了ConfigurationProvider接口(建造Configuration时被使用)的实现类。  
         \* － 任意的struts常量。  
    通过重载createDispatcher()方法，可以自定义dispather。  
      
    属性列表：  
         (1)actionMapper：通过注入，提供一个ActionMapper实例。  
         (2)dispatcher：暴露给子类一个Dispatcher实例。  
         (3)encoding：存储StrutsConstants.STRUTS\_I18N\_ENCODING的设置。  
         (4)filterConfig：通过初始参数，提供一个FilterConfig实例。  
         (5)lastModifiedCal：在缓存静态content，提供一个格化的日期用于设定头信息。  
         (6)log：提供一个logging实例。  
         (7)patchPrefixs：存储静态资源的路径前缀信息。  
         (8)serveStatic：存储StrutsConstants.STRUTS\_SERVE\_STATIC\_CONTENT的设置。  
         (9)serveStaticBrowserCache：存储StrutsConstants.STRUTS\_SERVE\_STATIC\_BROWSER\_CACHE的设置。  
    方法列表：  
         (1)copy(InputStream input, OutputStream output)：从input复制数据到output。  
         (2)createDispatcher(FilterConfig filterConfig)：创建默认的dispatcher对象，如果需要的话，子类可以重载此方法，自定义一个dispatcher对象。  
         (3)destory()：调用dispatcher.cleanup()，依次释放本地线程，销毁dispatcher对象。  
         (4)doFilter(ServletRequest request, ServeltResponse response, FilterChain chain)：处理一个action或处理一个请求的静态内容。  
         (5)findInputStream(String name, String packagePrefix)：搜索类路径下的静态资源。  
         (6)findStaticResoruce(String name, HttpServletRequest request, HttpServletResponse response)：搜索静态的资源，直接复制到相应response的头信息中。  
         (7)getContentType(String name)：获取指定资源的contentType。  
         (8)getFilterConfig()：获取FilterConfig实例。  
         (9)getServletContext()：给WebLogic的一些版本提供一个工作区。

        (9)init(FilterConfig filterCongfig)：创建默认的dispatcher对象和设置静态资源的默认包信息来初始化filter。  
         (10)parse(String packages)：返回一个数组通过解析一个指定逗号分隔的packages列表。  
         (11)prepareDispatcherAndWrapRequest(HttpServletRequest request, HttpServletResponse response)：对给定的request对象进行封装返回一个封装HttpServletRequest对象。例如显示的处理multipart数据。  
         (12)setMapper(ActionMapper actionMapper)  
         (13)setEncoding(String val)  
         (14)setServeStaticContent(String val)  
         (15)setServeStaticBrowserCache(String val)

附：

Struts2（XWork）提供的拦截器的功能说明：

|  |  |  |
| --- | --- | --- |
| 拦截器 | 名字 | 说明 |
| Alias Interceptor | alias | 在不同请求之间将请求参数在不同名字件转换，请求内容不变 |
| Chaining Interceptor | chain | 让前一个Action的属性可以被后一个Action访问，现在和chain类型的result（<result type=”chain”>）结合使用。 |
| Checkbox Interceptor | checkbox | 添加了checkbox自动处理代码，将没有选中的checkbox的内容设定为false，而html默认情况下不提交没有选中的checkbox。 |
| Cookies Interceptor | cookies | 使用配置的name,value来是指cookies |
| Conversion Error Interceptor | conversionError | 将错误从ActionContext中添加到Action的属性字段中。 |
| Create Session Interceptor | createSession | 自动的创建HttpSession，用来为需要使用到HttpSession的拦截器服务。 |
| Debugging Interceptor | debugging | 提供不同的调试用的页面来展现内部的数据状况。 |
| Execute and Wait Interceptor | execAndWait | 在后台执行Action，同时将用户带到一个中间的等待页面。 |
| Exception Interceptor | exception | 将异常定位到一个画面 |
| File Upload Interceptor | fileUpload | 提供文件上传功能 |
| I18n Interceptor | i18n | 记录用户选择的locale |
| Logger Interceptor | logger | 输出Action的名字 |
| Message Store Interceptor | store | 存储或者访问实现ValidationAware接口的Action类出现的消息，错误，字段错误等。 |
| Model Driven Interceptor | model-driven | 如果一个类实现了ModelDriven，将getModel得到的结果放在Value Stack中。 |
| Scoped Model Driven | scoped-model-driven | 如果一个Action实现了ScopedModelDriven，则这个拦截器会从相应的Scope中取出model调用Action的setModel方法将其放入Action内部。 |
| Parameters Interceptor | params | 将请求中的参数设置到Action中去。 |
| Prepare Interceptor | prepare | 如果Acton实现了Preparable，则该拦截器调用Action类的prepare方法。 |
| Scope Interceptor | scope | 将Action状态存入session和application的简单方法。 |
| Servlet Config Interceptor | servletConfig | 提供访问HttpServletRequest和HttpServletResponse的方法，以Map的方式访问。 |
| Static Parameters Interceptor | staticParams | 从struts.xml文件中将<action>中的<param>中的内容设置到对应的Action中。 |
| Roles Interceptor |  |  |