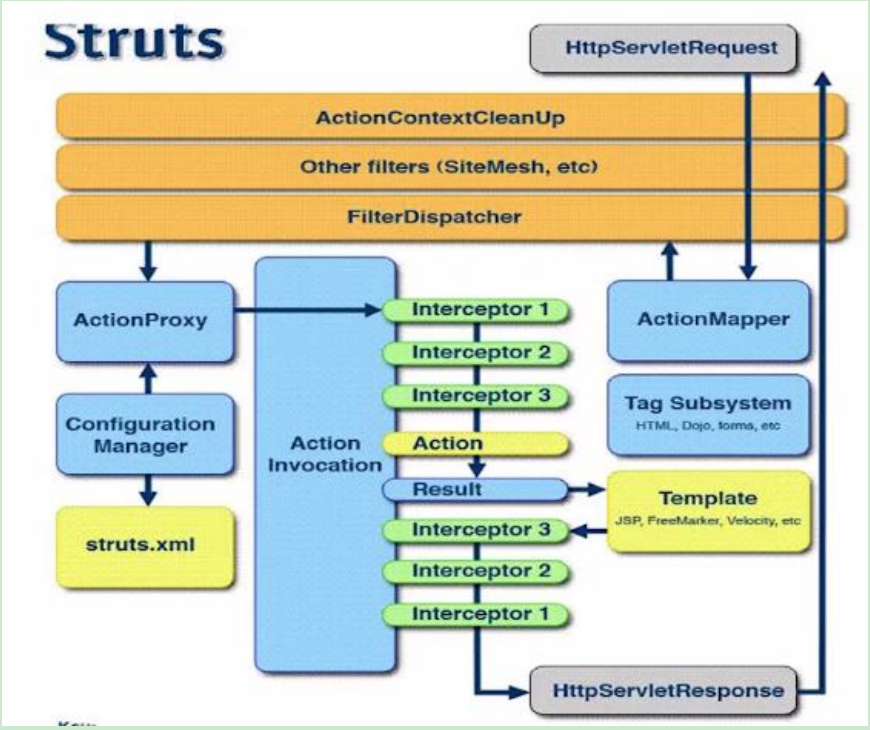
struts2拦截器的应用

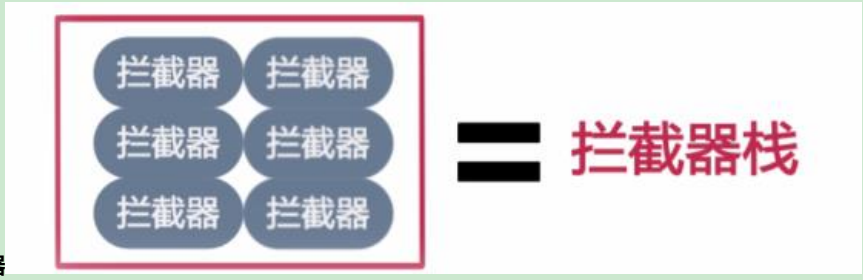
核心功能的实现，就是通过拦截器来实现的

什么是拦截器呢？Struts2大多数核心功能都是通过拦截器来实现的，每个拦截器完成某项功能

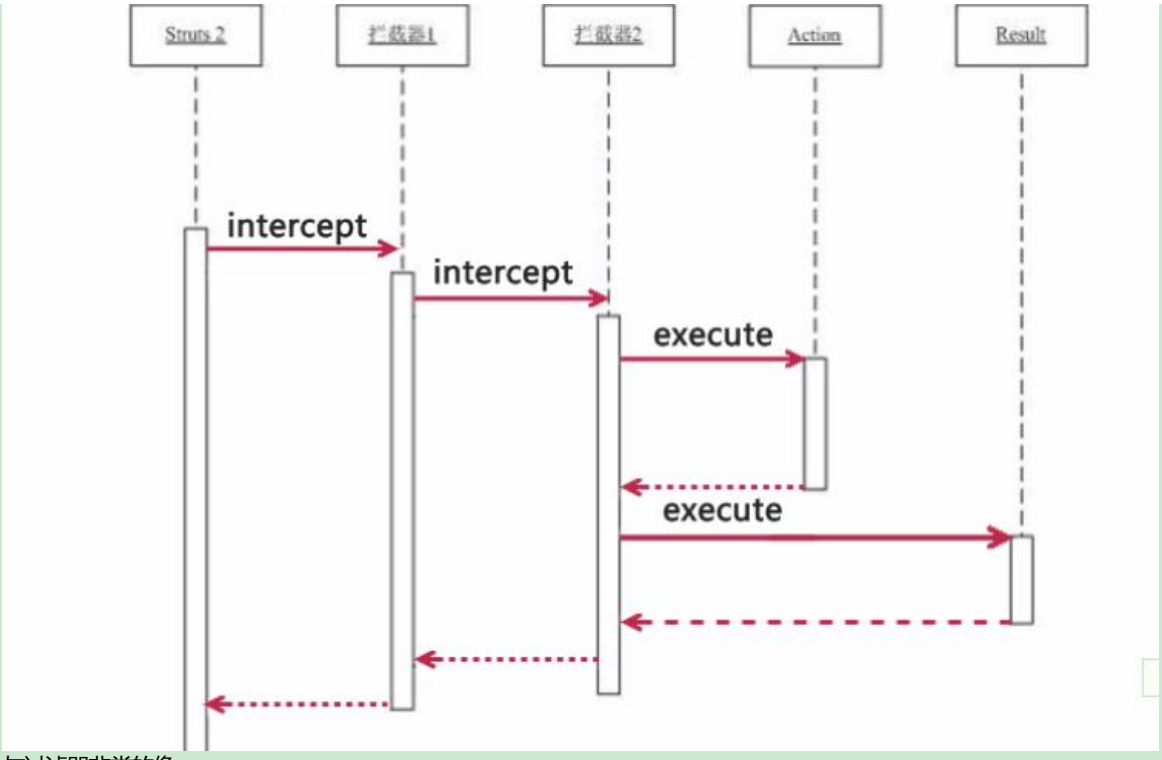
数据转移
数据校验
类型转换



还有一个概念
就是拦截器栈：
从结构上来看，拦截器栈就是多个拦截器的组合



从功能上看，拦截器栈也是拦截器
拦截器的执行过程就是一个递归的过程



与过滤器非常的像

那我们怎么需要定义一个拦截器呢

方式一：
实现Interceptor接口 与Filter
void init ( ) ；初始化拦截器所需要的资源
void destory ( ) ：释放在init ( ) 中分配的资源
String intercept (ActionInvocation ai) throws Exception
实现拦截器功能
利用ActionInvocation参数获取Action状态
返回result字符串作为逻辑视图

有很多时候，我们不需要实现intercept接口，

方法二：继承了AbstractInterceptor类
提供了init ( ) 和destory ( ) 方法的空实现
只需要实现intercept方法即可

拦截器示例

计算器Action执行时间
实现步骤：
创建拦截器
配置拦截器

提示：实现一个拦截器主要过程：
第一步：实现一个拦截器的类，有两个方法，写一个拦截器的类继承自AbstractInterceptor或者实现Interceptor接口，并且在这个类里面实现对应的intercept方法。

```
  1 执行Action之前
long  start=System.currentTimeMillis();
//            2 执行下一个拦截器，如果已经是最后一个拦截器了，则执行目标Action
String result =Invocation.invoke();
//            3 执行完Action之后
long end=System.currentTimeMillis();
System.out.println("执行时间"+(end - start)+"ms");
System.out.println(result);
            return  result;
```

第二步：在struts.xml中对拦截器进行注册（定义），并且在action中对拦截器进行引用。

```
<!--注册拦截器-->
```

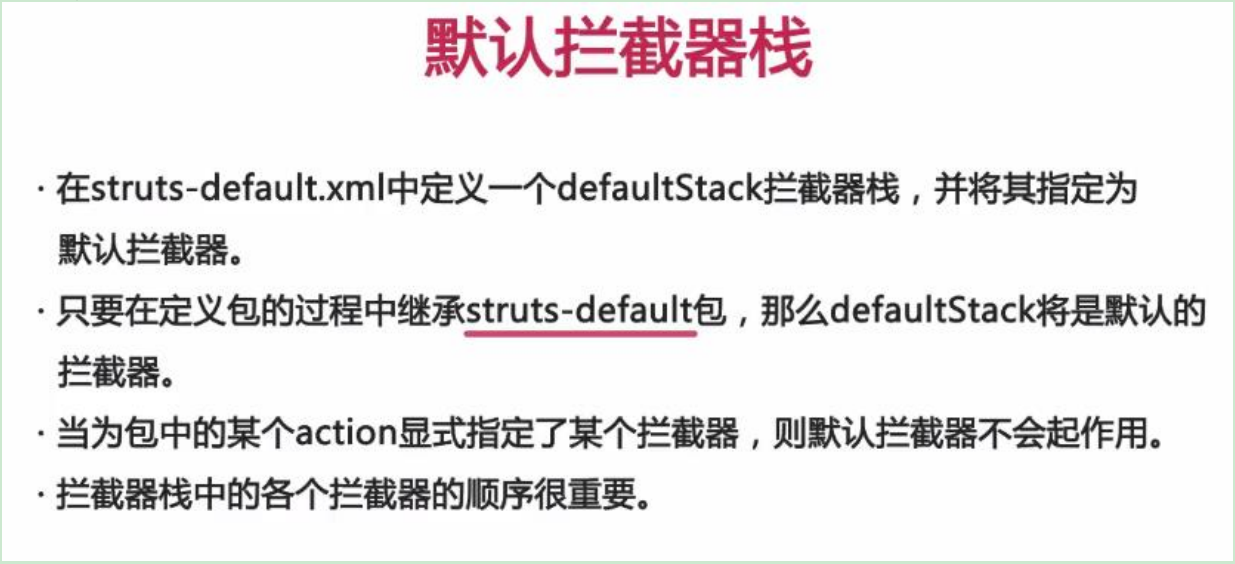
```
<interceptors>
     <interceptor name="mytimer" class="com.imooc.interceptor.TimerInterceptor">

     </interceptor>
</interceptors>
 <action name="timer"  class="com.imooc.action.TimerAction">
     <result>/success.jsp</result>
<!--引用拦截器-->
<interceptor-ref name="mytimer"></interceptor-ref>
 </action>
```

Struts2内建拦截器
params拦截器
负责将请求参数设置为Action属性

struts拦截器：
拷入jar包并build path，jdk设置成1.6的
创建struts2.xml并配置action
在web.xml配置struts2（filterclass:org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter）,其实就是新建Filter
编写action类，继承ActionSupport,继承execute方法，返回SUCCESS
将action配置到struts2.xml中
编写interceptor，在其中的用ActionInvocation的对象调用invoke方法，可以调用action
在struts2.xml中注册interceptor，并在action标签中引用

拦截器栈的引用顺序是需要注意的，因为引用的顺序，就是这些拦截器应用的顺序，

在struts中有一个默认的栈，默认的拦截器栈，这些拦截器的顺序是精心安排的，是非常的重要的 ，不能随便修改，不然不能使用我们使用的基本功能




```
<action name="timer"  class="com.imooc.action.TimerAction">
    <result>/success.jsp</result>
<!--引用拦截器-->
    <!--为Action显示引用拦截器后，默认的拦截器defaultStack不再生效，需要手工引用-->
<interceptor-ref name="defaultStack"></interceptor-ref>
    <interceptor-ref name="mytimer"></interceptor-ref>
</action>
```

params拦截器
负责将请求参数设置为Action属性
staticParams拦截器
将配置文件中的action元素的子元素param参数设置为Action属性
servletConfig拦截器
将源于Servlet API的各种对象注入到Action，必须实现对应接口
fileUpload拦截器
对文件上传提供支持，将支持和元数据设置到对应的Action属性
实际上内部是实现了Commons-FileUpload
exception拦截器
捕获异常，并且将异常映射到用户界面自定义的错误页面
validation拦截器
调用验证框架进行数据验证