概念

1、servlet：servlet是一种运行服务器端的java应用程序，具有独立于平台和协议的特性，并且可以动态的生成web页面，它工作在客户端请求与服务器响应的中间层。

2、filter：filter是一个可以复用的代码片段，可以用来转换HTTP请求、响应和头信息。Filter不像Servlet，它不能产生一个请求或者响应，它只是修改对某一资源的请求，或者修改从某一的响应。

3、listener：监听器，从字面上可以看出listener主要用来监听只用。通过listener可以监听web服务器中某一个执行动作，并根据其要求作出相应的响应。通俗的语言说就是在application，session，request三个对象创建消亡或者往其中添加修改删除属性时自动执行代码的功能组件。

4、interceptor：是在面向切面编程的，就是在你的service或者一个方法，前调用一个方法，或者在方法后调用一个方法，比如动态代理就是拦截器的简单实现，在你调用方法前打印出字符串（或者做其它业务逻辑的操作），也可以在你调用方法后打印出字符串，甚至在你抛出异常的时候做业务逻辑的操作。

servlet、filter、listener是配置到web.xml中，interceptor不配置到web.xml中，struts的拦截器配置到struts.xml中。spring的拦截器配置到spring.xml中。

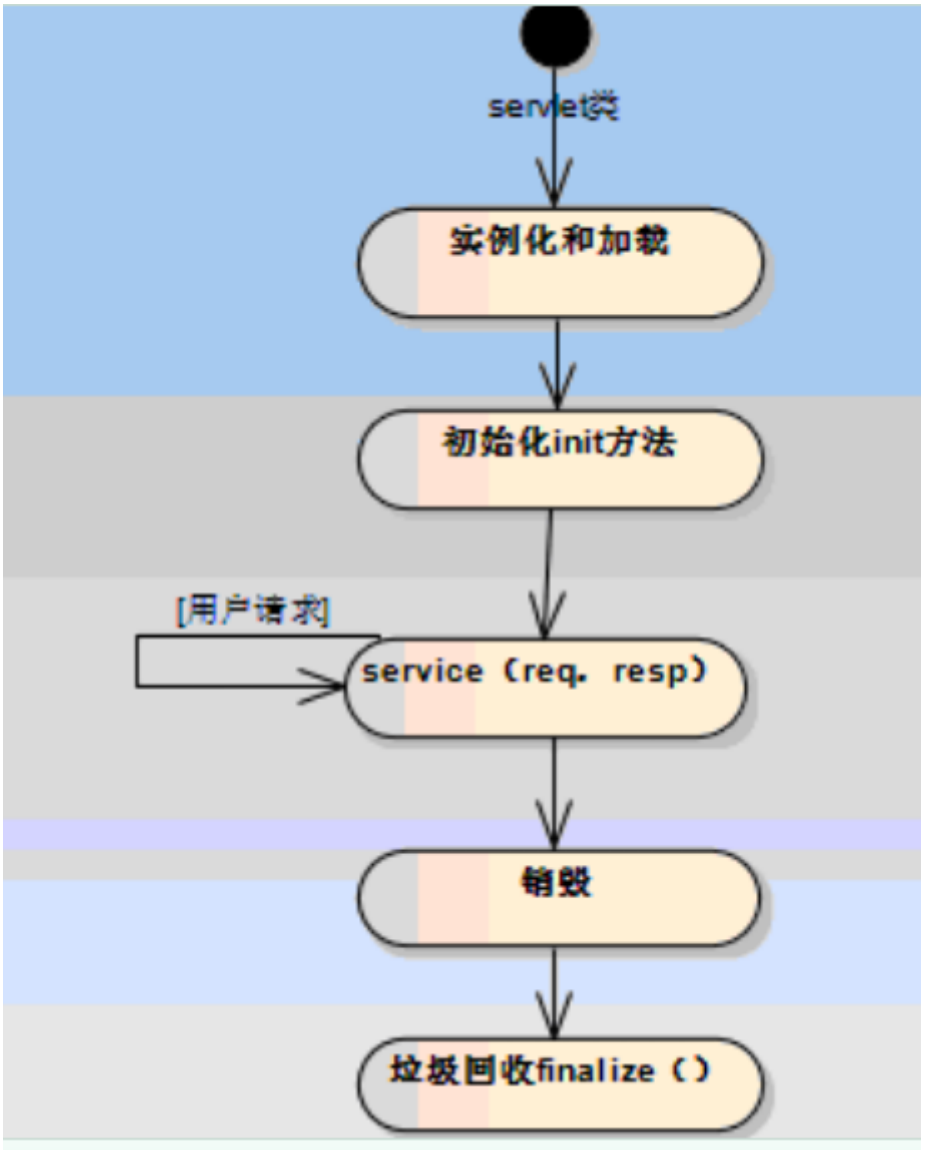
区别

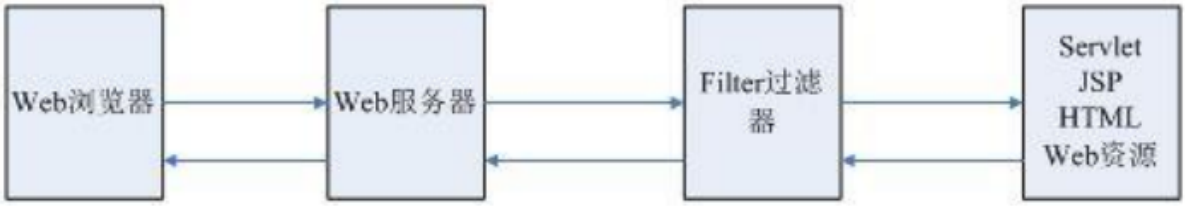
1,servlet 流程是短的，url传来之后，就对其进行处理，之后返回或转向到某一自己指定的页面。它主要用来在　业务处理之前进行控制.

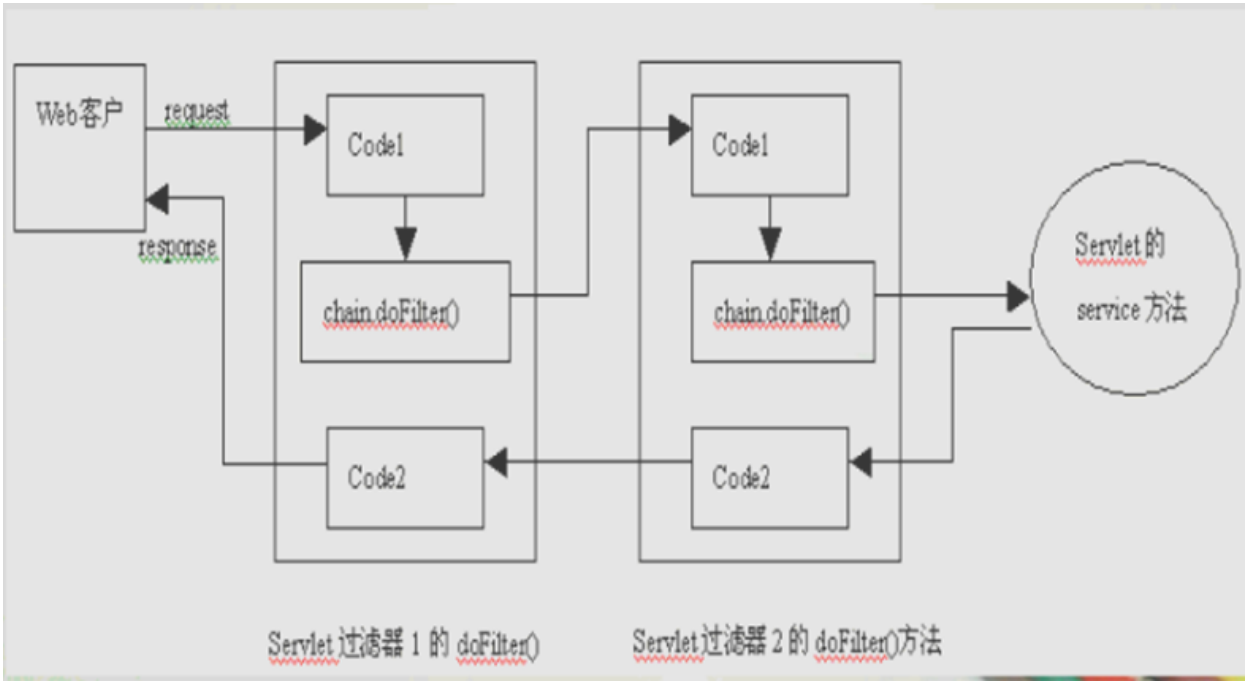
2,filter 流程是线性的， url传来之后，检查之后，可保持原来的流程继续向下执行，被下一个filter, servlet接收等，而servlet 处理之后，不会继续向下传递。filter可用来进行字符编码的过滤，检测用户是否登陆的过滤，禁止页面缓存等

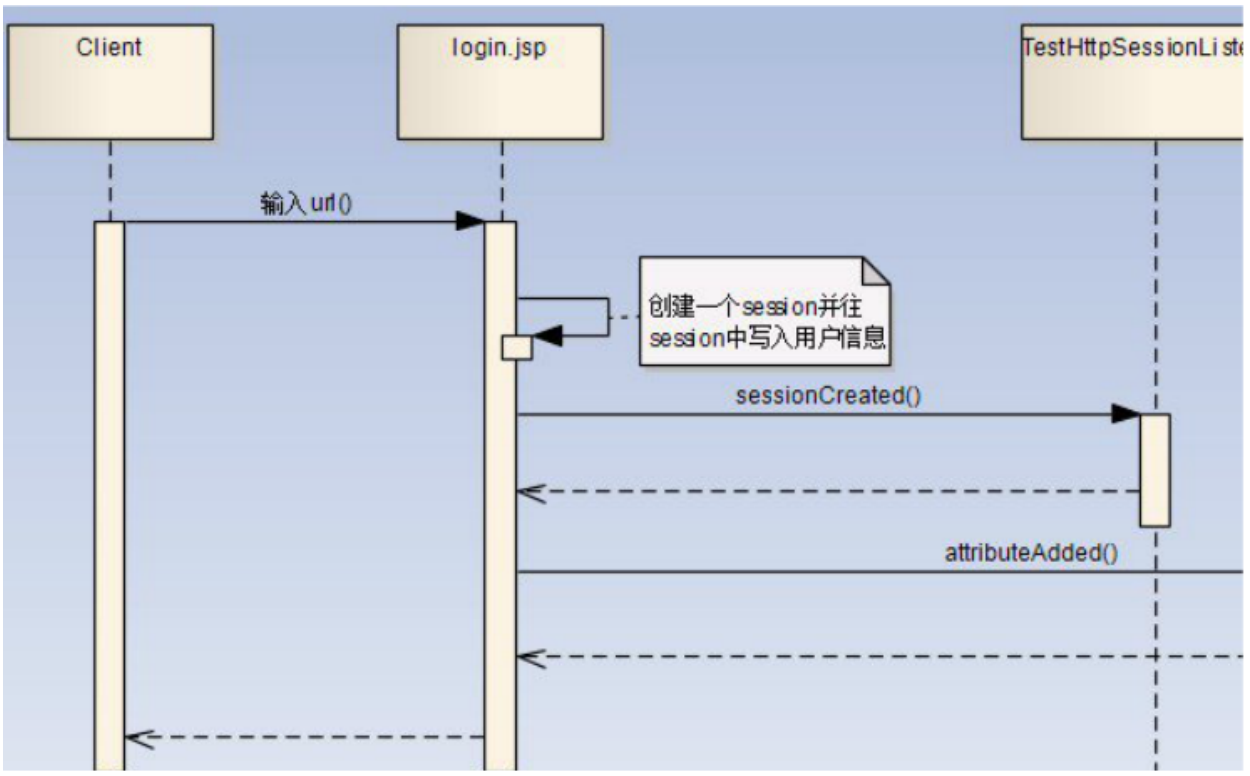
3,servlet,filter都是针对url之类的，而listener是针对对象的操作的，如session的创建，session.setAttribute的发生，在这样的事件发生时做一些事情。

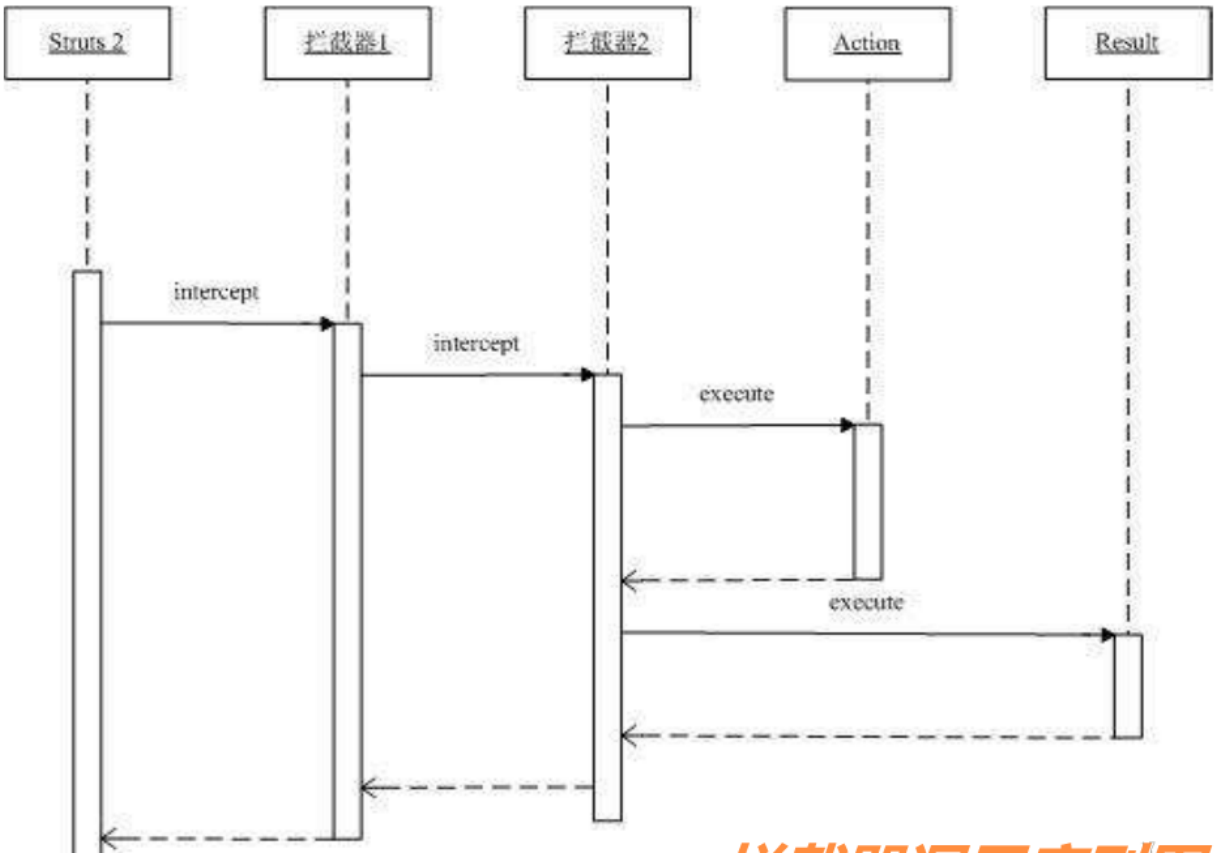
执行过程




Web浏览器
Web服务器
Filter过滤器
Servlet
JSP
HTML
Web资源


Web客户
request
response
Code1
chain.doFilter()
Code2
Code1
chain.doFilter()
Code2
Servlet的
service方法
Servlet过滤器1的doFilter()
Servlet过滤器2的doFilter()方法


Client
login.jsp
TestHttpSessionLis
输入url()
创建一个session并往
session中写入用户信息
sessionCreated()
attributeAdded()



Servlet

处理客户端请求的动态资源

- 接收请求数据：我们都知道客户端请求会被封装成HttpServletRequest对象，里面包含了请求头、参数等各种信息。

- 处理请求：通常我们会在service、doPost或者doGet方法进行接收参数，并且调用业务层（service）的方法来处理请求。

- 完成响应：处理完请求后，我们一般会转发（forward）或者重定向（redirect）到某个页面，转发是HttpServletRequest中

的方法，重定向是HttpServletResponse中的方法，两者是有很大区别的。

Servlet的创建：Servlet可以在第一次接收请求时被创建，也可以在在服务器启动时就被创建，这需要在web.xml的< servlet>中添加一条配置信息 < load-on-startup>5< /load-on-startup>，当值为0或者大于0时，表示容器在应用启动时就加载这个servlet，当是一个负数时或者没有指定时，则指示容器在该servlet被请求时才加载。

Servlet的其他重要方法：
> ServletConfig getServletConfig()
获取servlet的配置信息的方法，所谓的配置信息就是WEB-INF目录下的web.xml中的servlet标签里面的信息

> String getServletInfo()
获取servlet的信息方法

Filter

listener

本质是方法回调。可以监听Application、Session、Request对象，当这些对象发生变化就会调用对应的监听方法。

Ø ServletContext（监听Application）

生命周期监听：ServletContextListener，它有两个方法，一个在出生时调用，一个在死亡时调用；
void contextInitialized(ServletContextEvent sce)：创建Servletcontext时。应用程序已经被加载和初始化。

void contextDestroyed(ServletContextEvent sce)：销毁Servletcontext时。应用程序已经被载出，即关闭。

属性监听：ServletContextAttributeListener，它有三个方法，一个在添加属性时调用，一个在替换属性时调用，最后一个是在移除属性时调用。
void attributeAdded(ServletContextAttributeEvent event)：添加属性时；当有对象加入Application的范围时
void attributeReplaced(ServletContextAttributeEvent event)：替换属性时；当在application的范围有对象取代另一个对象的时
void attributeRemoved(ServletContextAttributeEvent event)：移除属性时；当有对象从application的范围移除时

Ø HttpSession（监听Session）

生命周期监听：HttpSessionListener，它有两个方法，一个在出生时调用，一个在死亡时调用；
void sessionCreated(HttpSessionEvent se)：创建session时。session已经被加载及初始化
void sessionDestroyed(HttpSessionEvent se)：销毁session时。session已经被载出（HttpSessionEvent类的主要方法是getSession(),可以使用该方法回传一个session对象）

属性监听：HttpSessioniAttributeListener，它有三个方法，一个在添加属性时调用，一个在替换属性时调用，最后一个是在移除属性时调用。
void attributeAdded(HttpSessionBindingEvent event)：添加属性时；
void attributeReplaced(HttpSessionBindingEvent event)：替换属性时

void attributeRemoved(HttpSessionBindingEvent event)：移除属性时

Ø ServletRequest（监听Request）

生命周期监听：ServletRequestListener，它有两个方法，一个在出生时调用，一个在死亡时调用；
voidrequestInitialized(ServletRequestEvent sre)：创建request时
void requestDestroyed(ServletRequestEvent sre)：销毁request时

属性监听：ServletRequestAttributeListener，它有三个方法，一个在添加属性时调用，一个在替换属性时调用，最后一个是在移除属性时调用。
voidattributeAdded(ServletRequestAttributeEvent srae)：添加属性时
void attributeReplaced(ServletRequestAttributeEvent srae)：替换属性时
void attributeRemoved(ServletRequestAttributeEvent srae)：移除属性时

感知Session监听：
1：HttpSessionBindingListener监听
⑴在需要监听的实体类实现HttpSessionBindingListener接口
⑵重写valueBound()方法，这方法是在当该实体类被放到Session中时，触发该方法
⑶重写valueUnbound()方法，这方法是在当该实体类从Session中被移除时，触发该方法

2：HttpSessionActivationListener监听
⑴在需要监听的实体类实现HttpSessionActivationListener接口

⑵重写sessionWillPassivate()方法，这方法是在当该实体类被序列化时，触发该方法

⑶重写sessionDidActivate()方法，这方法是在当该实体类被反序列化时，触发该方法