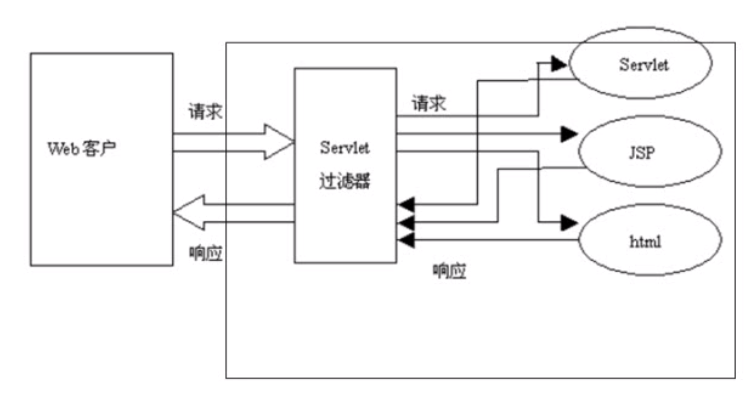
**过滤器（Filter）**

1 过滤过程：

在Servlet进行响应处理之前和处理结束之后，都要被过滤器拦截，即对请求和响应都进行拦截



request.getServletContext().getContextPath() 得到的是web项目的根路径，以“/”开头，即“/项目名称”

request.getRequestDispatcher(“/”) 该方法中路劲若以“/”开头的，则“/”表示web项目的根路径

**过滤器相关的接口：**

1. Filter
2. FilterChain
3. FilterConfig

过滤器程序是一个实现了Filter接口的Java类，与Servlet相似，有Servlet容器进行调用和执行；

过滤器需要在web.xml文件中进行配置和设置拦截的资源类型；可以拦截对JSP、Servlet、HTML和静态图片的请求；

Filter接口：javax.servlet.Filter

三个方法：

1．void init(FilterConfig filterConfig)

在Servlet容器加载web应用时则创建过滤器对象，并调用该方法；此过滤器对象为单例

参数FilterConfig与ServletConfig十分相似；

可在web.xml中配置过滤器的初始化参数；

2．void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

过滤器进行处理的方法，每次过滤器发生拦截都会执行；

参数FilterChain表示一个过滤器的链条，链条上的过滤器会一次执行，它只有一个方法：

void doFilter(ServletRequest request, ServletResponse response)

表示将请求传递给下一个过滤器，若没有下一个过滤器，则传给请求的目标资源；

FilterChain上的过滤器的拦截顺序与<filter-mapping>的配置顺序有关，先配置的先拦截；

3．void destroy()

销毁过滤器对象，释放资源；

在web.xml注册和映射过滤器：

*<!—注册过滤器-->*

<filter>

<filter-name>过滤器名称</filter-name>

<filter-class>过滤器类的完整类名</filter-class>

*<!—配置初始化参数-->*

<init-param>

<param-name>参数名称</param-name>

<param-value>参数值</param-value>

</init-param>

</filter>

*<!—映射过滤器-->*

<filter-mapping>

<filter-name>过滤器名称，需与注册的名称一致</filter-name>

*<!--过滤器所能拦截的请求-->*

<url-pattern>拦截的请求</url-pattern>

</filter-mapping>

过滤器拦截请求的类型

REQUEST：只接受直接请求

FORWARD：只接收转发的请求

INCLUDE：页面使用包含的方式发送请求时触发

ERROR：声明式异常被触发时拦截请求，即在web.xml中配置异常页面的方式

dispatcher标签配置拦截类型

通过在web.xml的<filter-mapping>中添加<dispatcher>标签，指明拦截类型，一个<filter-mapping>可配置多个<dispatcher>标签，表明可拦截多种

禁用缓存的过滤器

response.setDateHeader("Expires",-1);

response.setHeader("Cache-Contorl","no-cache");

response.setHeader("Pragma","no-cache");

过滤器进行汉字转码

在过滤器中更改request的编码，省去在JSP页面或Servlet修改编码的步骤

requset.setCharacterEncoding(encode);

request和response的包装类

HttpServletRequestWarpper类，实现HttpServletRequest接口

该类需要一个HttpServletRequest的具体实现类作为参数才能创建对象，其实现是通过调用其参数的HttpServletRequest的相应方法来完成的，所以，当需要根据项目的需求去修改request的部分方法的功能是，可以通过继承HttpServletRequestWarpper，重写目标方法，已达到增强request功能的目标。

request和response的实现是由服务器产商自行完成的，所以使用不同的服务器，request和response的具体实现类是有所不同的。