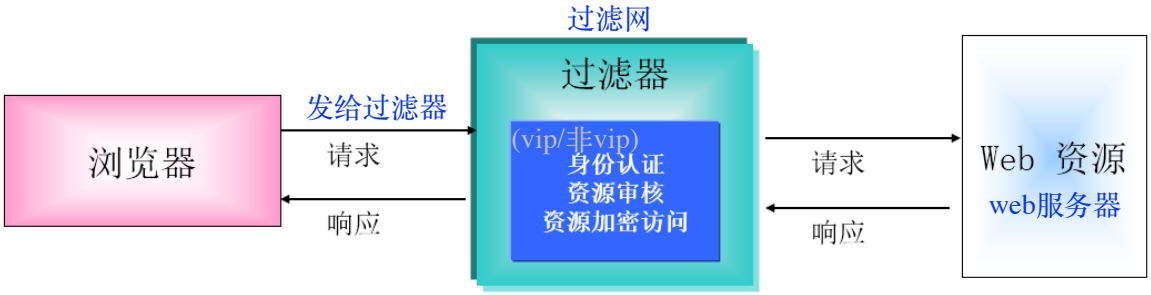
第五章 Filter+Listener核心技术

5.1 Filter过滤器（重点）

5.1.1 基本概念

- Filter本意为"过滤"的含义，是JavaWeb的三大组件之一，三大组件为：Servlet、Filter、Listener。
- 过滤器是向 Web 应用程序的请求和响应处理添加功能的 Web 服务组件。
- 过滤器相当于浏览器与Web资源之间的一道过滤网，在访问资源之前通过一系列的过滤器对请求进行修改、判断以及拦截等，也可以对响应进行修改、判断以及拦截等。

5.1.2 工作方式



5.1.3 使用方式

- 自定义类实现Filter接口并重写doFilter方法。

```
public class LoginFilter implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException,ServletException {
        //TODO 处理逻辑，必须调用下面的方法
        chain.doFilter(request,response);
    }
}
```

- 在web.xml文件中配置过滤器。

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>com.lagou.LoginFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

5.1.4 Filter接口

（1）基本概念

- javax.servlet.Filter接口主要用于描述过滤器对象，可以对资源的请求和资源的响应操作进行筛选操作。

（2）常用的方法

方法声明	功能介绍
void init(FilterConfig filterConfig)	实现过滤器的初始化操作
void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) filterChain.doFilter(servletRequest, servletResponse); // 放行	执行过滤操作的功能
void destroy()	实现过滤器的销毁操作

5.1.5 FilterConfig接口

（1）基本概念

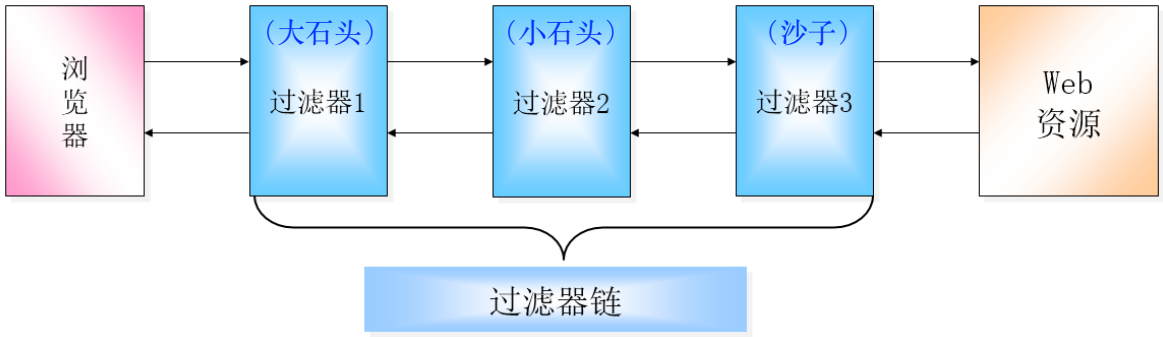
- javax.servlet.FilterConfig接口主要用于描述过滤器的配置信息。

（2）常用方法

方法声明	功能介绍
String getFilterName()	获取过滤器的名字
String getInitParameter(String name)	获取指定的初始化参数信息
Enumeration getInitParameterNames()	获取所有的初始化操作名称
ServletContext getServletContext()	获取ServletContext对象

5.1.6 多个过滤器的使用 多层过滤

- 如果有多个过滤器都满足过滤的条件，则容器依据映射的先后顺序来调用各个过滤器。



5.1.7 过滤器优点

- 实现代码的“可插拔性”，即增加或减少某个功能模块，不会影响程序的正常执行。
- 可以将多个相同处理逻辑的模块集中写在过滤器里面，可实现重复利用、也方便代码的维护。

5.2 Listener监听器（重点）

5.2.1 基本概念

- Servlet规范中定义的一种特殊的组件，用来监听Servlet容器产生的事件并进行相应的处理。
- 容器产生的事件分类如下：
  - 生命周期相关的事件。
  - 属性状态相关的事件。
  - 存值状态相关的事件。
- 底层原理是采用接口回调的方式实现。

5.2.2 基本分类 八大监听器

监听器类型	功能介绍
javax.servlet.ServletRequestListener	监听request作用域的创建和销毁
javax.servlet.ServletRequestAttributeListener	监听request作用域的属性状态变化
javax.servlet.http.HttpSessionListener	监听session作用域的创建和销毁
javax.servlet.http.HttpSessionAttributeListener	监听session作用域的属性状态变化
javax.servlet.ServletContextListener	监听application作用域的创建和销毁
javax.servlet.ServletContextAttributeListener	监听application作用域的属性状态变化
javax.servlet.http.HttpSessionBindingListener	监听对象与session的绑定和解除
javax.servlet.http.HttpSessionActivationListener	监听session数值的钝化和活化

钝化：内存写入硬盘
活化：硬盘写入内存

5.2.3 监听器详解

（1）ServletRequestListener监听器

- 在ServletRequest创建和关闭时都会通知ServletRequestListener监听器。
- 常用方法如下：

方法声明	功能介绍
void requestInitialized(ServletRequestEvent sre)	实现ServletRequest对象的初始化
void requestDestroyed(ServletRequestEvent sre)	实现ServletRequest对象的销毁

（2）ServletRequestAttributeListener监听器

- 向ServletRequest添加、删除或者替换一个属性的时候，将会通知ServletRequestAttributeListener监听器。
- 常用方法如下：

方法声明	功能介绍
void attributeAdded(ServletRequestAttributeEvent srae)	增加属性时触发
void attributeReplaced(ServletRequestAttributeEvent srae)	修改属性时触发
void attributeRemoved(ServletRequestAttributeEvent srae)	删除属性时触发

（3）HttpSessionListener监听器

- 当一个HttpSession刚被创建或者失效（invalidate）的时候，将会通知HttpSessionListener监听器。
- 常用方法如下：

方法声明	功能介绍
void sessionCreated(HttpSessionEvent se)	当一个HttpSession对象被创建时会调用这个方法
void sessionDestroyed(HttpSessionEvent se)	当一个HttpSession超时或者调用HttpSession的invalidate()方法让它销毁时，将会调用这个方法

（4）HttpSessionAttributeListener监听器

- HttpSession中添加、删除或者替换一个属性的时候，将会通知HttpSessionAttributeListener监听器。
- 常用方法如下：

方法声明	功能介绍
void attributeAdded(HttpSessionBindingEvent se)	当往会话中加入一个属性的时候会调用这个方法
void attributeRemoved(HttpSessionBindingEvent se)	当从会话中删除一个属性的时候会调用这个方法
void attributeReplaced(HttpSessionBindingEvent se)	当改变会话中的属性的时候会调用这个方法

（5）ServletContextListener监听器

- 在ServletContext创建和关闭时都会通知ServletContextListener监听器。
- 常用方法如下：

方法声明	功能介绍
void contextInitialized(ServletContextEvent sce)	当ServletContext创建的时候，将会调用这个方法
void contextDestroyed(ServletContextEvent sce)	当ServletContext销毁的时候（例如关闭应用服务器或者重新加载应用），将会调用这个方法

（6）ServletContextAttributeListener监听器

- 向ServletContext添加、删除或者替换一个属性的时候，将会通知ServletContextAttributesListener监听器
- 常用方法如下：

方法声明	功能介绍
void attributeAdded(ServletContextAttributeEvent scae)	往ServletContext中加入一个属性的时候触发
void attributeRemoved(ServletContextAttributeEvent scae)	从ServletContext中删除一个属性的时候触发
void attributeReplaced(ServletContextAttributeEvent scae)	改变ServletContext中属性的时候触发

（7）HttpSessionBindingListener监听器

该监听器无须在配置文件中进行配置

- HttpSession中绑定和解除绑定时，将会通知HttpSessionListener监听器。
- 常用方法如下：

有对象与session绑定

方法声明	功能介绍
void valueBound(HttpSessionBindingEvent event)	有对象绑定时调用该方法
void valueUnbound(HttpSessionBindingEvent event)	有对象解除绑定时调用该方法

（8）HttpSessionActivationListener监听器

（不常用）

- 当有session数值的钝化和活化操作时，将会通知HttpSessionActivationListener监听器。
- 常用方法如下：

钝化：序列化操作 将object -> 序列 持久化（session数据保存在硬盘里）
活化：反序列化 字节序列 -> object （从硬盘中读出session数据）

方法声明	功能介绍
void sessionWillPassivate(HttpSessionEvent se)	有钝化操作时调用该方法
void sessionDidActivate(HttpSessionEvent se)	有活化操作时调用该方法

- 配置context.xml文件的方式如下：

```
<Manager className="org.apache.catalina.session.PersistentManager"
saveOnRestart="true">
    <!-- 配置文件存放的路径信息，可以自由指定 -->
    <Store className="org.apache.catalina.session.FileStore"
directory="C:\session"/>
</Manager>
```

5.2.4 实战案例

- 自定义类实现监听器接口并重写相关的方法。

多少个session就有多少个在线用户

```
public class OnlineUser implements HttpSessionListener,ServletContextListener {
    ServletContext ctx = null;
 一个web项目对应唯一的一个ServletContext
    // 初始化ServletContext
    public void contextInitialized(ServletContextEvent e) {
        ctx = e.getServletContext();
```

```
    }
    // 销毁ServletContext
    public void contextDestroyed(ServletContextEvent e) {
         //将ServletContext设置成null;
    }
    // 当新创建一个HttpSession对象时
    public void sessionCreated(HttpSessionEvent e) {
        //将当前的在线人数加上1，并且保存到ServletContext(application)中
    }
    // 当一个HttpSession被销毁时（过期或者调用了invalidate()方法）
    public void sessionDestroyed(HttpSessionEvent e) {
        //将当前人数减去1，并且保存到ServletContext(application)中
    }
}
```

- 在web.xml中配置监听器

```
<listener>
    <listener-class> com.lagou.listener.OnlineUser </listener-class>
</listener>
```