servlet过滤器与监听器

前言

Servlet 过滤器和监听器是 Java Web 应用程序中常见的两种组件，它们提供了各种扩展 Web 应用程序功能的方式。

总的来说，过滤器和监听器都可以通过 Java Web 应用程序的配置文件或注解进行使用，方便灵活，并可以很好地实现框架与业务逻辑的分离，提高代码可维护性和扩展性。

过滤器（Filter）

1. 参数验证和转换：可以拦截用户提交的数据，并对数据格式进行验证、修正或转换。
2. 访问控制和认证：可以拦截请求并检查用户是否有访问特定资源的权限。
3. 日志记录：可以拦截请求并输出相应的日志信息，用于系统运行时的监测与故障排除。
4. 资源压缩和解密： 可以拦截响应并对其进行压缩或解密，以提高数据传输效率和安全性。

监听器（Listener）

1. 生命周期监听：可监听 ServletContext、HttpServletRequest 和 HttpSession 等对象的生命周期事件（如创建、销毁、添加属性、删除属性等）。
2. 属性变更监听：可监听属性的变化事件，并在发生变化时触发某些业务逻辑。
3. 请求/响应监听： 可监听 HttpServletRequest 和 HttpServletResponse 对象的事件（如请求到达、请求结束、响应开始、响应结束等），并在发生事件时执行业务逻辑。

过滤器（Filter）代码演示

下面是一个基本的 Servlet 过滤器示例，它拦截所有以 *.do 结尾的请求，输出一些日志信息，并记录请求执行时间：

Java | 复制代码

```java
public class LogFilter implements Filter {

    public void init(FilterConfig config) throws ServletException {
        // 初始化方法，在应用程序启动时调用
    }

    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain) throws IOException, ServletException {
        long startTime = System.currentTimeMillis();
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
        String uri = req.getRequestURI();
        if (uri.endsWith(".do")) { // 拦截以 .do 结尾的请求
            System.out.println("LogFilter: start handling request " + uri);
        }
        chain.doFilter(request, response); // 调用下一个 filter 或 servlet/jsp 处理请求
        if (uri.endsWith(".do")) {
            System.out.println("LogFilter: end handling request " + uri +
                    ", execution time: " + (System.currentTimeMillis() - startTime) + "ms");
        }
    }

    public void destroy() {
        // 销毁方法，在应用程序停止时调用
    }
}
```

在 web.xml 中配置 LogFilter：

HTML | 复制代码

```html
<filter>
  <filter-name>logFilter</filter-name>
  <filter-class>com.example.LogFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>logFilter</filter-name>
  <url-pattern>/*.do</url-pattern>
</filter-mapping>
```

这个过滤器会对所有以 .do 结尾的请求进行拦截和处理，并输出一些日志信息和请求执行时间。这个示例演示了过滤器的基本使用方法，开发者还可以根据具体需求实现不同的功能来增强应用程序的性能、安全性等特性。

监听器（Listener）代码演示

以下是一个使用ServletContextListener和ServletRequestListener实现的简单Web应用程序示例，它跟踪并记录用户访问应用程序的次数：

Java | 复制代码

```java
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.ServletRequestEvent;
import javax.servlet.ServletRequestListener;
import javax.servlet.annotation.WebListener;

@WebListener
    public class VisitorCounterListener implements ServletContextListener, ServletRequestListener {
        private int count; // 用户访问计数器

        public void contextInitialized(ServletContextEvent event) {
            count = 0; // 将计数器设置为0
            // 将计数器保存在ServletContext属性中
            event.getServletContext().setAttribute("visitorCount", count);
        }

        public void requestInitialized(ServletRequestEvent event) {
            count++; // 每当有请求进来，计数器加1
            // 将增加后的计数器再次保存在ServletContext属性中
            event.getServletContext().setAttribute("visitorCount", count);
        }

        // 空实现其他方法（不能省略）
        public void contextDestroyed(ServletContextEvent event) {}

        public void requestDestroyed(ServletRequestEvent event) {}
    }

```

上面的代码实现了ServletContextListener和ServletRequestListener接口，并用于在应用程序范围内跟踪用户访问计数器。在Servlet初始化期间，将初始计数器值设置为0，并在每个ServletRequest初始化时将其递增。最终，将当前计数器值存储在ServletContext属性"visitorCount"中，以供应用程序中的其他部分检索。

要使用此监听器，请将以下代码添加到web.xml文件中：

XML | 复制代码

```xml
<listener>
  <listener-class>com.example.VisitorCounterListener</listener-class>
</listener>
```