



## SERVLET SCOPE & FILTER

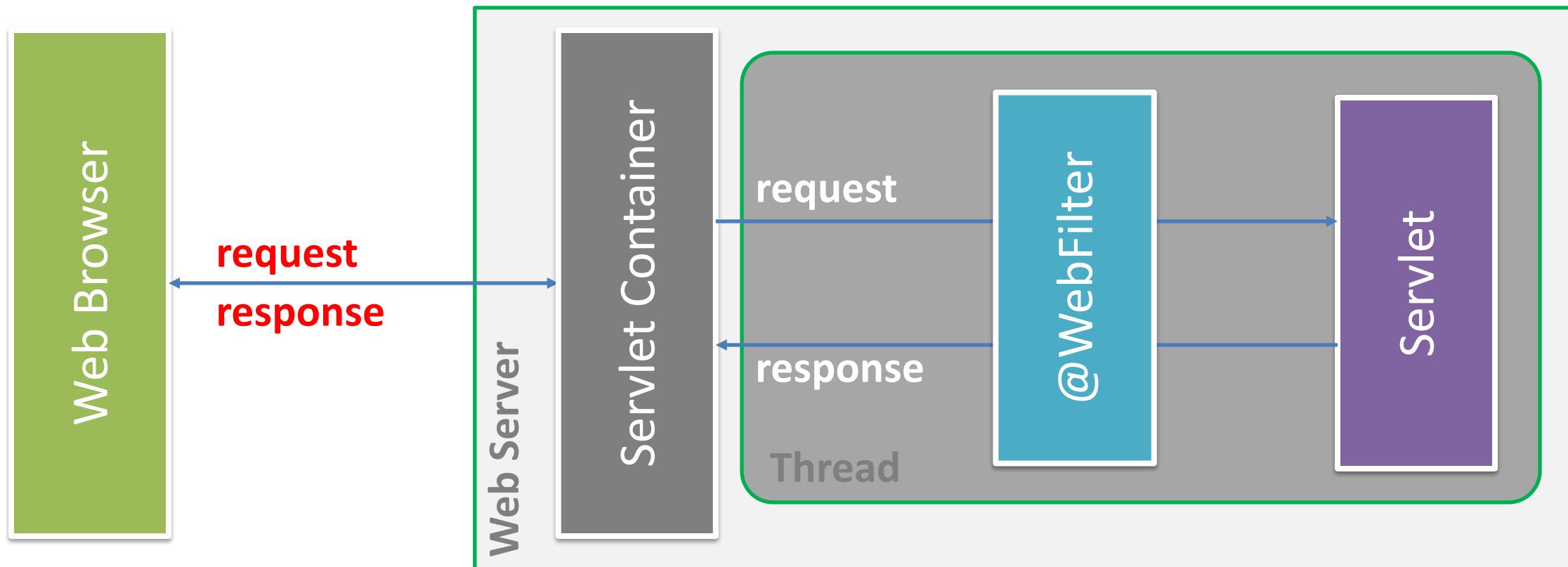
---

### LẬP TRÌNH JAVA #4 (P5.2)

- ❑ Giới thiệu Filter
- ❑ WebFilter
  - ❖ @urlPatterns
  - ❖ @dispatcherTypes
  - ❖ @initParams
- ❑ Vòng đời của Filter
- ❑ Xây dựng UTF8Filter và LoggerFilter
- ❑ Sắp xếp thứ tự thực hiện của các filter



- ❑ **@WebFilter** được sử dụng để thực hiện các công việc trước khi request chuyển đến servlet và sau khi servlet thực hiện xong.



*Chú ý: Filter.doChain() và Servlet.service() luôn hoạt động trong cùng một Thread*

```
@WebFilter(urlPatterns = {"url1", "url2", "urlN"}, dispatcherTypes = DispatcherType.REQUEST)
public class MyFilter implements Filter{
    @Override
    public void destroy() {
        // TODO Chạy trước khi bị giải phóng
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        // TODO Chạy trước khi đến servlet/jsp
        chain.doFilter(request, response); // chuyển tiếp đến servlet/jsp
        // TODO Chạy sau khi servlet/jsp thực hiện xong
    }
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        // TODO Chạy sau khi được nạp vào
    }
}
```

*Chú ý: nếu trong doFilter() có gọi chain.doFilter() thì request mới được chuyển tiếp đến Servlet hoặc Filter kế tiếp, ngược lại thì Servlet hoặc Filter kế tiếp không được thực hiện*

☐ urlPatterns xác định các url bị lọc. Cú pháp tương tự @WebServlet

- ❖ **\*.php**: chỉ lọc các địa chỉ url với đuôi là .php
- ❖ **/\***: lọc tất cả các địa chỉ url
- ❖ **/abc/\***: chỉ lọc tất cả địa chỉ bắt đầu với /abc/
- ❖ **/abc/def.php**: lọc một địa chỉ /abc/def.php

☐ dispatcherTypes xác định các cách kích hoạt bộ lọc

- ❖ **REQUEST**: lọc mỗi khi có request (mặc định)
- ❖ **INCLUDE**: lọc mỗi khi được include()
- ❖ **FORWARD**: lọc mỗi khi được forward()
- ❖ **ERROR**: lọc mỗi khi có lỗi

```
@WebFilter(urlPatterns = "/*", dispatcherTypes = DispatcherType.REQUEST)
public class Utf8Filter implements Filter{
    @Override
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
            throws IOException, ServletException {
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        chain.doFilter(req, resp);
    }
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {}
    @Override
    public void destroy() {}
}
```



# DEMOSTATION

---

## ❑ Vấn đề của javax.servlet.Filter

- ❖ javax.servlet.Filter phục vụ cho nhiều công nghệ (FTP và HTTP) nên đối số của doChain() là ServletRequest và ServletResponse.
- ❖ Trong khi lập trình web thì chúng ta cần HttpServletRequest và HttpServletResponse nên cần phải thực hiện ép kiểu.
- ❖ javax.servlet.Filter chứa 3 phương thức init(), destroy() và doFilter(). Vì vậy khi thực thi theo interface này chúng ta phải viết cả 3 phương thức nói trên. Tuy nhiên trong thực tế đôi khi chúng ta không quan tâm đến init() và destroy() và chỉ viết mã cho doFilter()

## ❑ Giải pháp: xây dựng HttpFilter thừa kế Filter

- ❖ Khai báo lại init(), destroy() và doFilter() với từ khóa default
- ❖ Bổ sung doFilter(HttpServletRequest, HttpServletResponse, FilterChain) và gọi nó trong default doFilter()

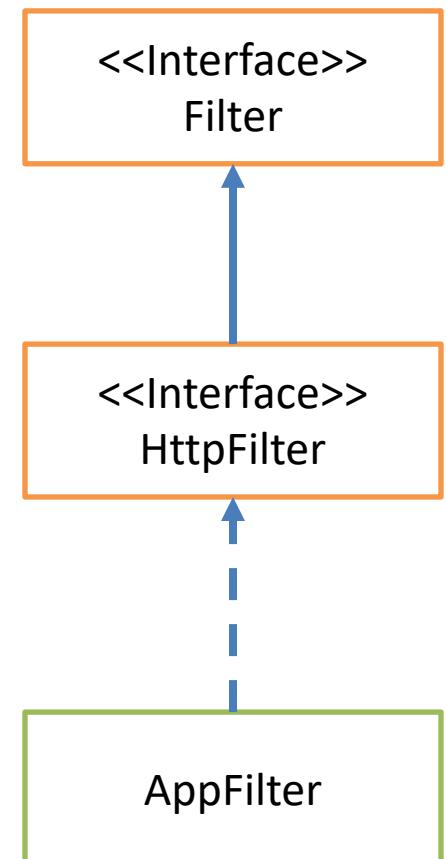
```
public interface HttpFilter extends Filter {  
    @Override  
    default void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)  
        throws IOException, ServletException {  
        HttpServletRequest request = (HttpServletRequest)req;  
        HttpServletResponse response = (HttpServletResponse)resp;  
        this.doFilter(request, response, chain);  
    }  
    void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain chain)  
        throws IOException, ServletException;  
  
    @Override  
    default void init(FilterConfig e) throws ServletException {}  
  
    @Override  
    default void destroy() {}  
}
```

## ☐ Tạo một Servlet Filter

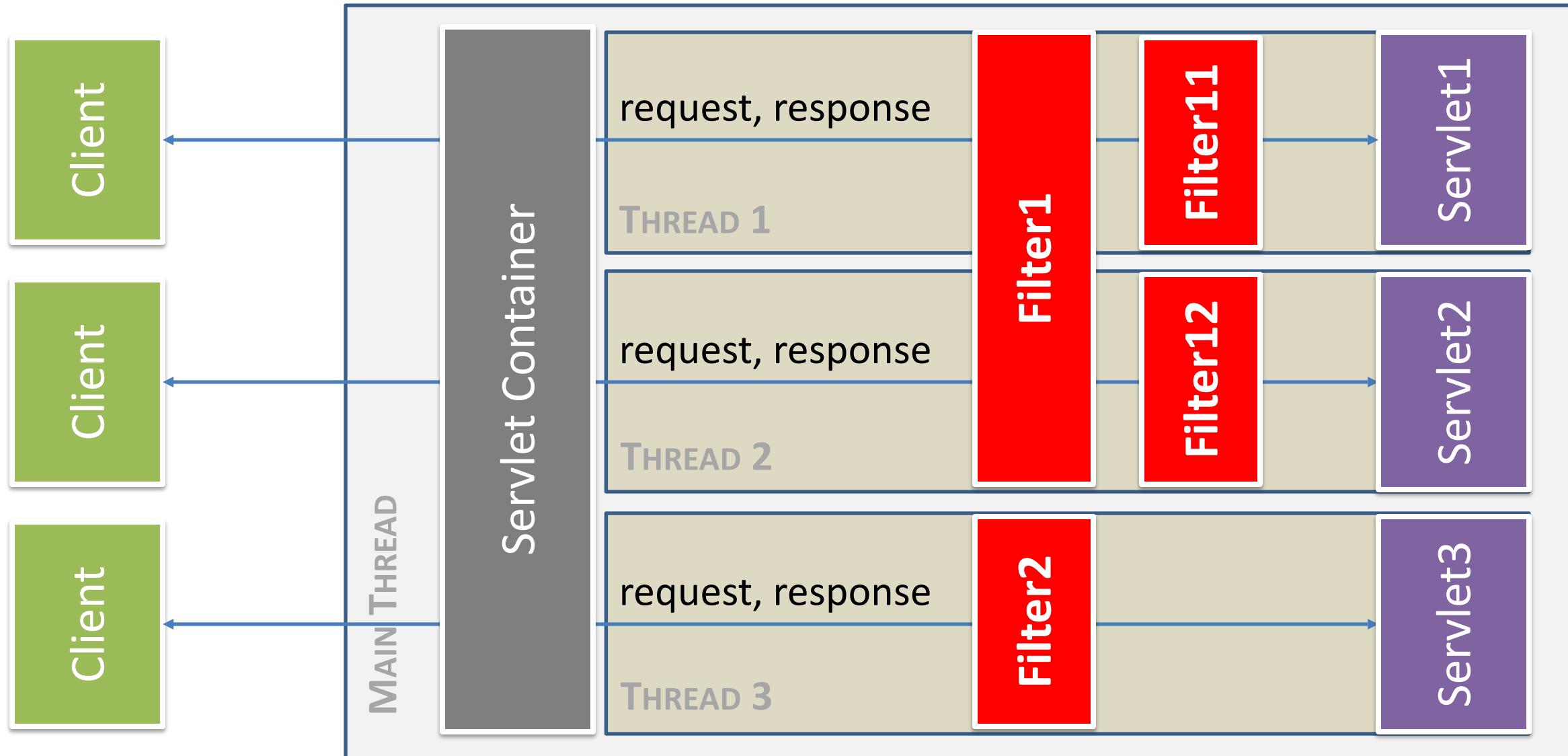
- ❖ Thực thi theo interface HttpFilter thay vì Filter
- ❖ Khi cần chúng ta có thể override init() và destroy()

## ☐ Ví dụ: Tạo AppFilter thiết lập utf-8 cho tất cả request

```
@WebFilter("/*")
public class AppFilter implements HttpFilter{
    @Override
    public void doFilter(HttpServletRequest req, HttpServletResponse resp,
        FilterChain chain) throws IOException, ServletException {
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        chain.doFilter(req, resp);
    }
}
```



```
@WebFilter("/admin/*")
public class LoggerFilter implements HttpFilter{
    @Override
    public void doFilter(HttpServletRequest req, HttpServletResponse resp,
                         FilterChain chain) throws IOException, ServletException {
        User user = (User) req.getSession().getAttribute("user");
        String uri = req.getRequestURI();
        Date time = new Date();
        // ghi nhận user, uri, time vào CSDL hoặc file
        chain.doFilter(req, resp);
    }
}
```



Một servlet/jsp có thể được lọc bởi nhiều filter và một filter có thể lọc nhiều servlet

# SẮP XẾP THỨ TỰ THỰC HIỆN WEB FILTER

```
@WebFilter(filterName="filter1", urlPatterns="/url1/*")
public class Filter1 implements Filter {}
```

```
@WebFilter(filterName="filter2", urlPatterns="/url2/*")
public class Filter2 implements Filter {}
```

*Filter nào mapping  
trước sẽ thực hiện  
trước*

```
<filter-mapping>
    <filter-name>filter1</filter-name>
    <url-pattern />
</filter-mapping>
<filter-mapping>
    <filter-name>filter2</filter-name>
    <url-pattern />
</filter-mapping>
```

web.xml



# DEMOSTATION

---

- @WebFilter
  - urlParttern
  - dispatcherType
- Ứng dụng ghi nhận truy xuất người sử dụng
- Sắp xếp thứ tự thực hiện các filter





Cảm ơn