Hackademics Hanoi
Hack your passion, Code your life!

CHƯƠNG 15 – SERVLET ADVANCED

Nội dung

1. Servlet Config
2. Servlet Context
3. Kỹ thuật Include, Forword, Redirect
4. Events, Listeners
5. Servlet filter

1. SERVLET CONFIG

SERVLET CONFIG

- Thông tin trong tập tin web.xml có thể được lấy bằng cách sử dụng đối tượng servletConfig
- Cấu hình các đối số cho một servlet trong web.xml
- Cách lấy ServletConfig: Dùng phương thức **getServletConfig()** của Servlet interface sẽ trả cho ta một đối tượng ServletConfig
- Thường tạo đối tượng ServletConfig toàn cục lưu trữ cấu hình ứng dụng Web
- Mothods:

Methods of ServletConfig interface

1. **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
2. **public Enumeration getInitParameterNames():**Returns an enumeration of all the initialization parameter names.
3. **public String getServletName():**Returns the name of the servlet.
4. **public ServletContext getServletContext():**Returns an object of ServletContext.

SERVLET CONFIG

```
<servlet>
  <servlet-name>DemoServlet</servlet-name>
  <servlet-class>DemoServlet</servlet-class>
  <init-param>
  <param-name>username</param-name>
  <param-value>admin</param-value>
  </init-param>
  <init-param>
  <param-name>password</param-name>
  <param-value>passOk</param-value>
  </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>DemoServlet</servlet-name>
  <url-pattern>/servlet1</url-pattern>
</servlet-mapping>
```

```
ServletConfig config=getServletConfig();
Enumeration<String> e=config.getInitParameterNames();
String str="";
while(e.hasMoreElements()){
            str=e.nextElement();
            out.print("<br>Name: "+str);
          out.print(" value: "+config.getInitParameter(str));
}
```

2. SERVLET CONTEXT

SERVLET CONFIG

- Trong cùng một ứng dụng web, các servlet chia sẻ cùng một môi trường
- Servlet container chia sẻ môi trường trong Servlet thông qua giao diện Servlet Context
- Servlet API có định nghĩa giao diện cho phép giao tiếp giưa servlet và servelt container

- Khởi tạo Servet Context
  - Servlet Context được khởi tạo khi ứng dụng được nạp
  - Mỗi ứng dụng sẽ có duy nhất một Servet Context
  - Cấu hình servlet context với các giá trị attribute khởi tạo trong web.xml

- Các tham số (attribute) được khởi tạo qua Servlet Context thường được dùng để cung cấp một số thông tin đặc biệt như:
  - Thông tin kết nối cơ sở dữ liệu
  - Thông tin về nhà phát triển: tên, email,....
- Đối tượng ServetContext được chứa trong ServletConfig
- Cách lấy ServetContext: Dùng phương thức getServletContext**()**
  - ServletContext context = getServletContext();
- Có ba phương thức ServletContext tương ứng với các attribute:
  - getAttribute,
  - setAttribute
  - và removeAttribute

SERVLET CONFIG

```
ServletContext context = getServletContext();
Enumeration<String> eC = context.getInitParameterNames();
String strC = "";
while (eC.hasMoreElements()) {
        strC = eC.nextElement();
        out.print("<br>Name: " + strC);
        out.print(" value: " + context.getInitParameter(strC));
}
```

```
<context-param>
    <param-name>dbname</param-name>
    <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>
<context-param>
    <param-name>dbusername</param-name>
    <param-value>demo</param-value>
</context-param>
<context-param>
    <param-name>dbpassword</param-name>
    <param-value>psDemo</param-value>
</context-param>
```

3. Kỹ thuật Include, Forward, Redirect

Include

- Đôi khi chèn thêm các nội dung tĩnh hoặc động giống nhau trong web resource: thêm copyright information, footer, banner, widget,... vào trong response trả về từ một Web component
- Static resource: Thêm nội dung tĩnh vào response của servlet đang muốn đính kèm
- Dynamic web component (Servlet hoặc JSP):
  - Gửi request tới Web component được đính kèm
  - Thực thi web component được đính kèm
  - Đính kèm kết quả thực thi được vào response của servlet đang xét

CHÚ Ý WEB RESOURCE

- Web resource được đính kèm có thể truy cập tới đối tượng request, nhưng bị hạn chế với đối tượng response
  - Có thể viết vào phần body của response và commit một response
  - Không thể thiết lập các headers hoặc gọi bất kỳ phương thức nào (ví dụ: setCookie) ảnh hưởng tới các headers của response
- Làm thế nào Include?
  - Lấy đối tượng RequestDispatcher từ đối tượng ServletContext

    ```
    RequestDispatcher dispatcher =
        getServletContext().getRequestDispatcher("/banner");
    ```
  - Sau đó. gọi phương thức include() của đối tượng RequestDispatcher với tham số là đối tượng request và response

    ```
    dispatcher.include(request, response);
    ```

Foward

- Khi nào cần sử dụng kỹ thuật Forwading tới một component khác?
  - Khi muốn có một web component thực hiện xử lý sơ bộ 1 request và một component khác đảm nhiệm sinh response
  - Ví dụ: Xử lý một phần request rồi chuyển cho component khác, tùy từng request
- Nên sử dụng khi yêu cầu 1 resource khác phản hồi lại cho 1 user
  - Chú ý: Nếu đã truy cập vào đối tượng ServletOutputStream hoặc PrintWriter trong servlet, sẽ không forward được nữa. Nếu cố tình sẽ có ngoại lệ IllegalStateException

FORWARD

Làm thế nào để Forwarding tới một web resource khác?

• Lấy đối tượng RequestDispatcher từ đối tượng HttpServletRequest

```
RequestDispatcher dispatcher =
request.getRequestDispatcher("/template.jsp");
```

• Nếu cần giữ lại URL gốc để xử lý thêm, có thể lưu lại thành thuộc tính request

• Sau đó, gọi phương thức forward() của đối tượng RequestDispatcher với tham số là đối tượng request và response

```
dispatcher.forward(request, response);
```

Redirect một request

- Hai kỹ thuật điều hướng request

Cách 1:

- response.setStatus(response.SC_MOVE_PERMANTLY);
- response.setHeader(“Location”,”http://...”);

Cách 2:

- public void sendRedirect(String url)

- Không thể redirect khi ServletResponse đã sẵn sàng trả về cho client

4. EVENTS, LISTENERS

EVNET

- Sự khởi tạo hoặc hủy bỏ Servlet Context được gọi là Events
- Được sử dụng để thực hiện các hành động
  - Ghi log khi context được khởi tạo
  - Hỗ trợ khi context bị hủy
  - Đặc tả Servlet định nghĩa giao diện Listener là con đường nhận các thông báo khi có các sự kiện quan trọng có liên quan tới ứng dụng xảy ra

LISTENER

- Các giao diện:
  - **ServletContextListener**
  - **ServletContextAttributeListener**
  - **HttpSessionListener**
  - **ServletRequestListener**
- Cấu hình trong web.xml

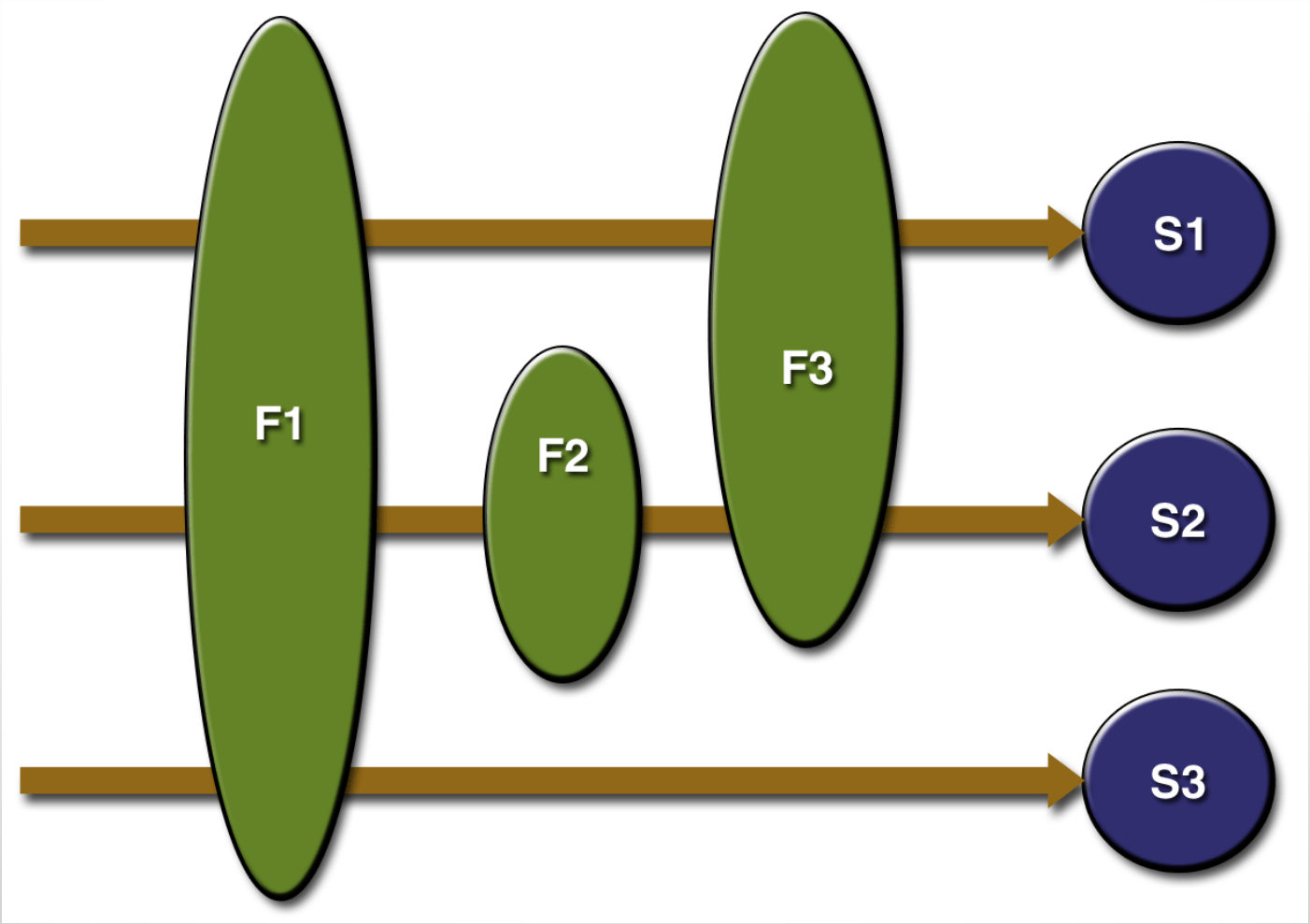
THẺ LISTENER

- Chỉ định các lớp lắng nghe cho các sự kiện ứng dụng
- Chứa một và chỉ một class xác định tên đầy đủ của lớp thực hiện các listener interface
- <listener> <listener-class>com.ed.listener.AppContextListener</listener-class> </listener> <listener> <listener-class>com.ed.listener.AppContextAttributeListener</listener-class> </listener> <listener> <listener-class>com.ed.listener.MySessionListener</listener-class> </listener> <listener> <listener-class>com.ed.listener.MyServletRequestListener</listener-class> </listener>

5. SERVLET FILTER

FILLTER

- Servlet filter là gì và tại sao cần?
- Các servlet Filter móc nối với nhau như thế nào
- API lập trình cho Servlet Filter
- Cấu hình Servlet Filter trong web.xml
- Các bước xây dựng và triển khai filters



SERVLET FILTER LÀ GÌ

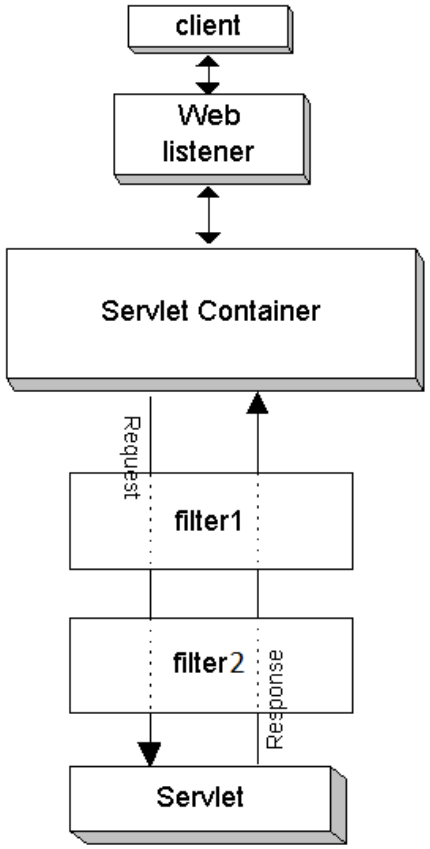
- Là thành phần để chặn và sửa các requests và response
  - Filter có thể liên kết với nhau thành một chuỗi và tích hợp vào hệ thống khi triển khai (deploy)
- Hoạt động như một người bảo vệ ngăn chặn những thông tin không mong muốn được truyền từ một điểm đến một điểm khác
- Ứng dụng:
  - Đếm chặn các truy cập
  - Caching, compression, logging
  - Authentication, access control, encryption
  - Content transformations: chuyển đổi hình ảnh,...

SERVLET FILTER LÀM GÌ

- Kiểm tra (Examine) các request header
- Điều chỉnh lại đối tượng request: Sửa phần dữ liệu hoặc request header
- Điều chỉnh lại đối tượng response sửa phần dữ liệu hoặc response headers
- Gọi đến filter kế tiếp trong chuỗi
- Kiểm tra (Examine) lại các response headers sau khi đã gọi filter kế tiếp trong chuỗi
- Tung ra ngoại lệ để thông báo có lỗi trong quá trình xử lý

CÁC SERVLET FILTER MÓC NỐI VỚI NHAU NHƯ THẾ NÀO

- Các filters có thể móc lối với nhau
  - Thứ tự móc nối: chính là thứ tự các phần tử <filter> trong file web.xml
- Filter đầu tiên trong chuỗi sẽ được Container gọi
  - Qua phương thức doFilter()
  - Filter sau khi thực hiện công việc xong, sẽ gọi filter tiếp theo trong chuỗi: gọi phương thức chain.doFilter()
- Filter cuối cùng gọi phương thức service() của servlet



API SERVLET FILTER

- init(FilterConfig)
  - Được gọi 1 lần duy nhất khi filter được khởi tạo lần dầu
  - Lấy ra đối tượng ServletContext từ đối tượng FilterConfig và lưu vào đâu đó (Vd làm 1 thuộc tính của Filter) để phương thức doFilter() có thể truy cập.
  - Đọc các tham số khởi tạo từ đối tượng FilterConfig qua phương thức getInitParameter()
- destroy()
  - Được gọi duy nhất 1 lần khi container hủy đối tượng filter
  - VD: đóng filte hoặc đóng kết nối Database

API SERVLET FILTER

- doFilter(..)
  - Được gọi mỗi khi filter được kích hoạt
  - Chứa các xử lý của filter
  - Đối tượng ServletRequest được ép kiểu về HttpServletRequest nếu request là HTTP request
  - Gọi filter tiếp theo: chain.doFilter(..)
  - Hoặc chặn request lại:
    - Không gọi phương thức chain.doFilter(..)
    - Filter phải cung cấp output cho client
  - set headers on the response for next entity

CẤU HÌNH SERVLET FILTER WEB.XML

- thẻ <filter>
  - filter-name
  - ...

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>LoginFilter</filter-class>
</filter>

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

TRIỂN KHAI FILTERS

- Tạo một lớp thực thi giao diện Filter
  - Thực thi các xử lý trong phương thức doFiler()
  - Gọi phương thức doFilter() của đối tượng filterChain
- Cấu hình filter cho servlet và các trang Jsp:
  - Có thể dùng anotation
  - Config trong web.xml

HẾT CHƯƠNG 13