Bài 2
Session Cookie Filter Và Annotation

MỤC TIÊU

- Session là gì, ý nghĩa của việc sử dụng Session
- Cookie là gì, ý nghĩa của việc sử dụng Cookie
- Các khái niệm về filter, sử dụng filter để xử lý dữ liệu trước và sau khi đến servlet
- Các loại annotation, lợi ích của việc sử dụng annotation

Giới thiệu

- HTTP là một giao thức phi trạng thái. Các request, response giữa client và server không được lưu trữ lại các giá trị. Tuy nhiên 1 số trường hợp như người dùng thực hiện các bước giao dịch, xác thực,… các giá trị giữa các request cần được lưu lại ở server hoặc client để tái sử dụng.
- Trong JSPServlet hỗ trợ một số các phương pháp để có thể lưu trữ trạng thái như sau:
  - Cookie
  - Session
  - Query string
  - Hidden form field

Cookie

- Cookie là một phần dữ liệu được gửi bởi server tới trình duyệt web của người dùng, được lưu trữ trên máy người dùng và được đọc lại bởi server vào mỗi lần nhận request cho cùng một trang.
- Có thể chứa nhiều cặp name-value và được thay đổi trong các tiêu đề của request và response.
- Được tự động xóa sau khoảng thời gian thiết lập kết thúc.

Mô hình hoạt động của cookie

Ví dụ về cookie

```
//This snippet remember an added item by adding to a cookie

public void doGet (HttpServletRequest request,  HttpServletResponse
response) throwsServletException,  IOException
{
  //If the user wants to add an item in a cookie

     if (values != null) {  ItemId = values[0];

         Cookie getItem = new Cookie("Buy", ItemId);  getItem.setComment("User

          has indicated a desire "
        + "to buy this book from the bookstore.");

         response.addCookie(getItem);
     }
}
```

Các phương thức làm việc với Cookie

- public void setMaxAge(int expiry) : Thiết lập khoảng thời gian tồn tại của cookie
- public int getMaxAge() : Trả về thời gian tồn tại tối đa của cookie
- public void setValue(java.lang.String newValue): Gán giá trị cho cookie
- public java.lang.String getValue() : Trả về giá trị mà cookie đang lưu trữ
- public java.lang.String getName() : Trả về tên cookie
- public void setPath(String uri) : Trả về đường dẫn servlet mà client trả về cookie
- public java.lang.String getPath() : Gán đường dẫn servlet mà client trả về cookie
- public Cookie[] getCookies() : Gọi bởi request để trả về tất cả các cookie đang có
- void addCookie (Cookie cookie) : Gọi bởi response để gửi cookie tới client

Vấn đề bảo mật với cookie

- Javascript có thể được sử dụng để truy xuất cookie từ một máy bởi vậy xảy ra nhiều rủi ro khi sử dụng cookie
- Để bảo mật cookie có thể sử dụng 2 thuộc tính cấu hình khi tạo cookie:
  - Thuộc tính secure: Cookie được gửi chỉ trên các kết nối bảo mật SSL
  - Thuộc tính HttpOnly: Nội dung của cookie không được truy xuất với javascript

```
protected void doGet(HttpServletRequest req,  HttpServletResponse
res) throws ServletException,  IOException {
. . .

  Cookie = new Cookie("Color", "Cyan");
  Response.addCookie(cookie);  cookie.setHttpOnly(true);
  cookie.setSecure(true);

. . .
  boolean status = cookie.isHttpOnly();  out.println("<br>Status of Cookie -
  Marked as HttpOnly
= " + status);
}
```

Session

- Session biểu diễn phiên làm việc của client (browser) với web server. Nó lưu trữ thông tin của phiên làm việc như thông tin đăng nhập, thông tin giao dịch…
- Khi client request lần đầu tiên tới web server thì web server sẽ tự động sinh ra một Session ID duy nhất gửi về client và lưu trữ ở dạng Temporary cookie đồng thời Web server cũng lưu trữ lại Session ID đó ở bộ nhớ Cache, hoặc ở CSDL SQLServer hoặc những nguồn lưu cache khác nhau.
- Các request tiếp theo của client sẽ luôn gửi đi Session ID để Web server nhận biết phiên làm việc.
- Trình duyệt tắt hoặc Session trên web server timeout thì Session ID cũng bị xóa

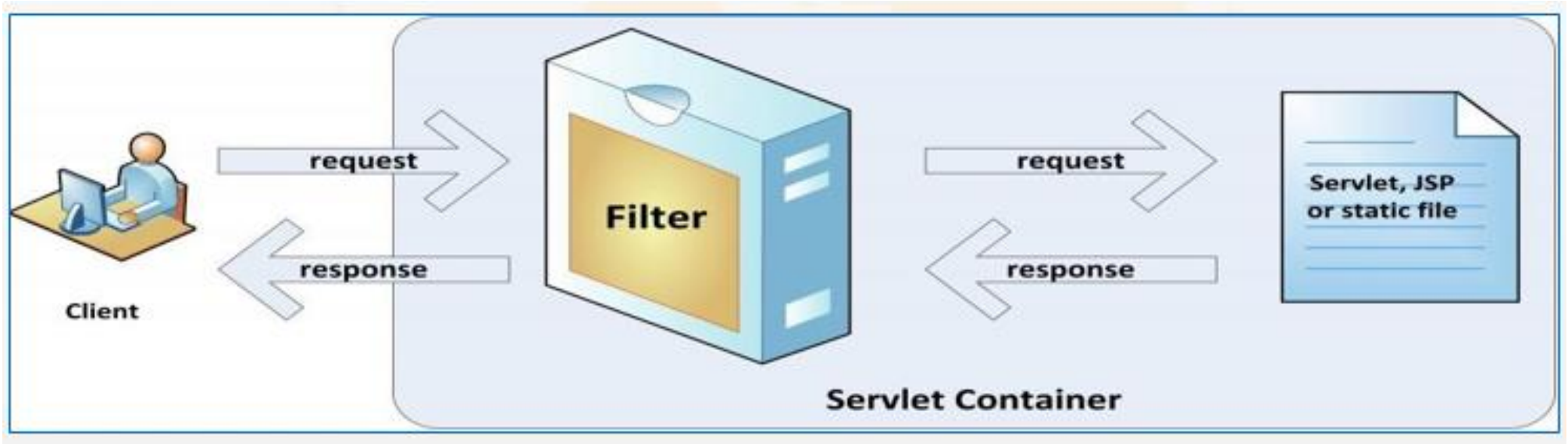
Các phương thức của HttpSession

- public Object getAttribute(String name) : Lấy về giá trị theo tên thuộc tính
- public String getId() : Lấy về định danh duy nhất gán với session này
- public int getMaxInactiveInterval() : Lấy về thời gian tối đa tồn tại session
- public ServletContext getServletContext(): Lấy về đối tượng ServletContext
- public void invalidate() : Hủy session
- public boolean isNew() : Session được sử dụng lần đầu tiên
- public void setAttribute(String name, Object value) : Gán giá trị theo tên thuộc tính
- public void removeValue(String name) : Hủy giá trị theo tên thuộc tính
- public void setMaxInactiveInterval(int interval) : Thiết lập thời gian tối đa tồn tại session

Ví dụ về Session

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");

    String name = request.getParameter("username");
    String pass = request.getParameter("password");

    if(name.equals("admin") && pass.equals("1234")) {
        HttpSession session = request.getSession();
        session.setAttribute("username", name);
        request.getRequestDispatcher("welcome.jsp").forward(request, response);
    }else {
        request.setAttribute("error", "Wrong username or password!");
        request.getRequestDispatcher("login.jsp").forward(request, response);
    }
}
```

- Xử lý dữ liệu từ client trước khi tới servlet hoặc sau servlet trước khi tới client.

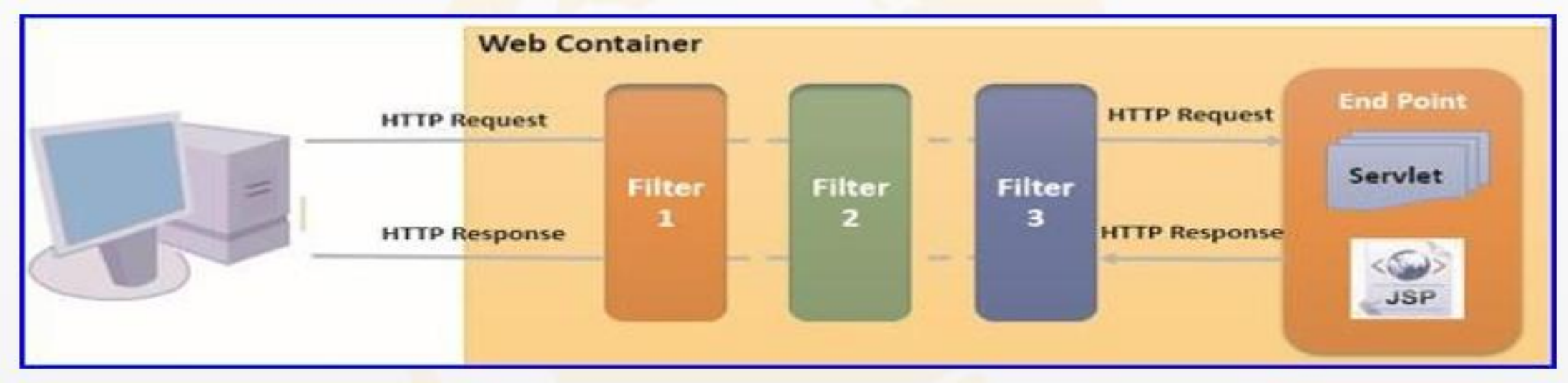


Mục đích sử dụng Filter

- Tối ưu hóa thời gian gửi phản hồi tới client
- Nén kích thước nội dung gửi từ webserver tới client qua internet
- Chuyển đổi định dạng hình ảnh, văn bản và các videos
- Tối ưu hóa băng thông trên mạng
- Xác thực người dùng trong các website bảo mật
- Mã hóa tiêu đề request và response để giải quyết vấn đề bảo mật

Dãy các Filters

- Có thể tạo nhiều filters để xử lý tuần tự công việc trước khi tới xử lý chính, vì vậy chúng ta sẽ có một dãy các filters
- Các filters này được cấu hình để xác định cái nào sẽ được chạy trước, cái nào sẽ được chạy sau



Ví dụ sử dụng Filter

```
public class MyGenericFilter implements javax.servlet.Filter {

   public FilterConfig filterConfig;

  public void init(final FilterConfig filterConfig) {

            this. FilterConfig = filterConfig;
  }
  public void doFilter(final ServletRequest request,  final
     ServletResponse response, FilterChain chain) throws
     java.io.IOException, javax.servlet.ServletException {

            chain. DoFilter(request,response);
}

public void destroy() {   }

}
```

Cấu hình Filter

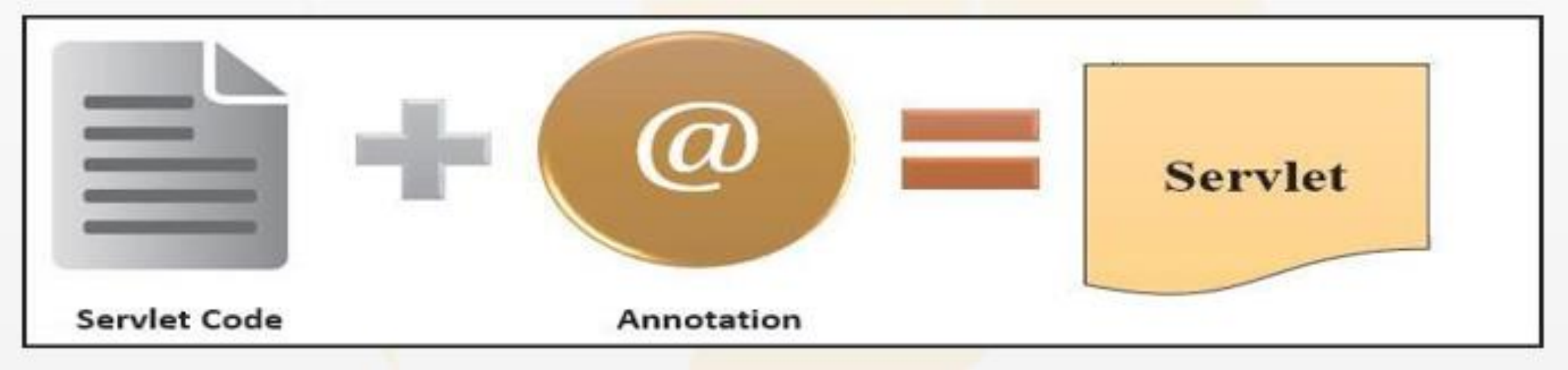
```
<filter>
<filter-name>Message</filter-name>
 <filter-class>servlet.filter.MessageFilter</filter-class>

<!-- Initialization parameters -->
  <init-param>
    <param-name>message</param-name>
    <param-value>Welcome to Filter</param-value>
  </init-param>

</filter>

<filter-mapping>
<filter-name>Message</filter-name>
<servlet-name>MyServlet</servlet-name>
 </filter-mapping>
```

Annotations

- Thay vì cấu hình truyền thống bằng các file .xml, annotation được sử dụng để cấu hình các thành phần của server
- Annotation có thể được cấu hình cho các lớp, các phương thức, các trường, các tham số, các biến cục bộ và các gói.
- Được xử lý khi đoạn mã chứa nó được dịch bởi các trình biên dịch, các công cụ triển khai,…

Các loại annotations

- Servlet API hỗ trợ các kiểu khác nhau của annotations để khai báo các thành phần:
  - Annotation khai báo các servlet
  - Annotation khai báo các filters và các listeners
  - Annotation khai báo các cấu hình bảo mật.



Servlet annotation

```
@WebServlet(
    name = "webServletAnnotation", urlPatterns = {"/hello", "/  helloWebApp"},
    asyncSupported = false,  initParams = {
     @WebInitParam(name = "name", value = "admin"),  @WebInitParam(name =
     "param1", value = "paramValue1"),  @WebInitParam(name = "param2", value =
     " paramValue2")
    }
)
public class HelloAnnotationServlet extends HttpServlet {
 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException,  IOException {
. . .
 out.write("Welcome " + getServletConfig().
                        getInitParameter("name"));
. . .
}
```

Listener annotation

```
. . .
. . .
import javax.servlet.annotation.WebListener;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

@WebListener
public class MySessionListener implements HttpSessionListener{

 @Override
 public void sessionCreated(HttpSessionEvent se) {
    System.out.println("Session Created:: ID="+ se.getSession().getId());
 }

@Override
public void sessionDestroyed(HttpSessionEvent se) {
    System.out.println("Session Destroyed:: ID="+ se.getSession().getId());
 }

}
```

Filter annotation

```
. . .
@WebFilter (value="/hello",  initParams={@WebInitParam(name="message",
    value="Servlet  says: ") })
public MyFilter implements Filter {
 private FilterConfig _filterConfig;

 public void init(FilterConfig filterConfig)  throws
  ServletException
 {
 _filterConfig = filterConfig;
 }

public void doFilter(ServletRequest req,  ServletResponse res,
    FilterChain chain)
 throws ServletException, IOException
 {

PrintWriter out = res.getWriter();
    out.print(_filterConfig.getInitParameter("message"));
}

 public void destroy() {
 // destroy
 }
}
```

HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH

TRÂN TRỌNG CẢM ƠN!

238 Hoàng Quốc Việt, Bắc Từ Liêm, Hà Nội

0968.27.6996

tuyensinh@bachkhoa-aptech.edu.vn

www.bachkhoa-aptech.edu.vn