



SECURITY WITH SERVLET FILER

LẬP TRÌNH JAVA #4 (P6.1)

- ❑ Giới thiệu Servlet Filter
- ❑ Xây dựng Filter
- ❑ Vòng đời của Filter
- ❑ Xây dựng Filter
- ❑ Sắp xếp thứ tự thực hiện của các filter
- ❑ Chia sẻ HttpServletRequest và HttpServletResponse



BIỂU HIỆN THƯỜNG GẶP WEB SECURITY



- ❑ Trang web trước và sau khi đăng nhập có thể khác nhau
- ❑ Không thể thực hiện một số hành vi nếu chưa đăng nhập
- ❑ Sau khi đã đăng nhập
 - ❖ Giao diện có thể khác nhau tùy thuộc vào vai trò
 - ❖ Một số chức năng có thể thực hiện được hoặc không tùy vào vai trò
- ❑ Đăng nhập
 - ❖ Từ trang web
 - ❖ Từ mạng xã hội

❑ XSS – Cross-Site Scripting

- ❖ Ngăn chặn thực hiện của script từ data

❑ CORS – Cross-Site Resource Sharing

- ❖ Ngăn chặn chia sẻ tài nguyên

❑ CSRF – Cross-Site Request Forgery

- ❖ Ngăn chặn các request giả lập

❑ Authentication

- ❖ Ngăn chặn thực hiện khi chưa đăng nhập

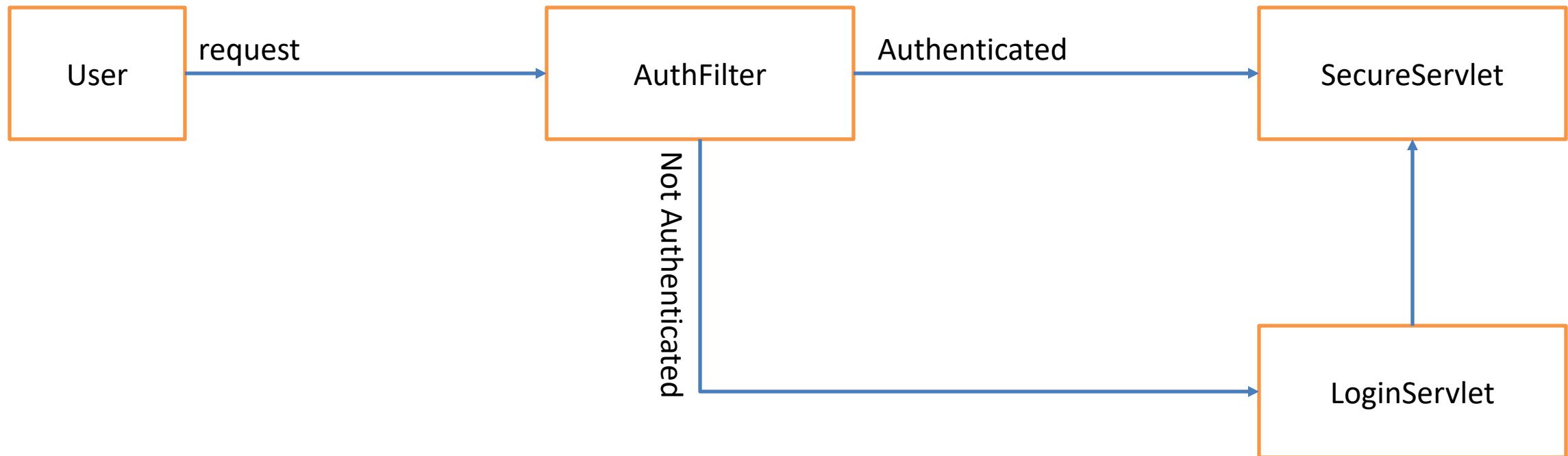
❑ Authorization

- ❖ Ngăn chặn thực hiện khi đăng nhập không đúng vai trò

- ❑ Trong phạm vi môn học chúng ta chỉ nghiên cứu Authentication và Authorization và thực hiện với Filter
- ❑ Authentication là yêu cầu người sử dụng phải thực hiện đăng nhập trước khi thực hiện một vài chức năng nào đó.
- ❑ Authorization là yêu cầu người sử dụng phải thực hiện đăng nhập với một vai trò nào đó trước khi thực hiện một vài chức năng nào đó.

- ❑ Có 2 giải pháp để bảo vệ các chức năng yêu cầu phải đăng nhập trước khi thực hiện
 - ❖ Ẩn chức năng này khỏi giao diện khi user chưa đăng nhập để user không tương tác được.
 - ❖ Vẫn hiển thị lên giao diện nhưng khi user nhấp vào những chức năng này thì
 - Nếu đăng nhập rồi thì cho phép thực hiện
 - Nếu chưa đăng nhập thì chuyển sang trang đăng nhập
- ❑ Ví dụ: Trong Assignment khi người sử dụng xem tiểu phẩm giải trí thì có thể thực hiện Like và Share khi đã đăng nhập vì ứng dụng cần ghi nhận dữ liệu like và share vào CSDL.

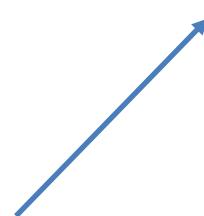
- ❑ Có 2 giải pháp để bảo vệ các chức năng yêu cầu phải đăng nhập đúng vai trò trước khi thực hiện
 - ❖Ẩn chức năng này khỏi giao diện khi user chưa đăng nhập hoặc đã đăng nhập không đúng vai trò để user không tương tác được.
 - ❖Vẫn hiển thị lên giao diện nhưng khi user nhấp vào những chức năng này thì
 - Nếu đăng nhập đúng vai trò rồi thì cho phép thực hiện
 - Nếu chưa đăng nhập hoặc đăng nhập không đúng vai trò thì chuyển sang trang đăng nhập
- ❑ Ví dụ: Trong Assignment khi người sử dụng đăng nhập với vai trò Administrator thì mới thực hiện được các chức năng trong phần quản trị.



- ❑ Xây dựng chức năng đăng nhập @WebServlet("/login")
 - ❖ LoginServlet
 - ❖ Login.jsp
- ❑ Xây dựng chức năng bảo mật @WebServlet("/secure")
 - ❖ SecureServlet
 - ❖ Secure.jsp
- ❑ Xây dựng bộ lọc kiểm tra URL "/secure" @WebFilter("/secure")
 - ❖ AuthFilter

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.getRequestDispatcher("/login.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String username = req.getParameter("username");
        String password = req.getParameter("password");

        UserDAO dao = new UserDAOImpl();
        User user = dao.findById(username);
        if(user == null){
            req.setAttribute("message", "Invalid username");
        } else if(!user.getPassword().equals(password)){
            req.setAttribute("message", "Invalid password");
        } else {
            req.getSession().setAttribute("user", user);
            req.setAttribute("message", "Login successfully");
        }
        req.getRequestDispatcher("/login.jsp").forward(req, resp);
    }
}
```



```
<c:url var="url" value="/login"></c:url>
<i>${message}</i>
<form action="${url}" method="post">
    <input name="username"><br>
    <input name="password" type="password"><hr>
    <button>Login</button>
</form>
```

WEBSERVLET("/SECURE") & SECURE.JSP

```
@WebServlet("/secure")
public class SecureServlet extends HttpServlet{
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        HttpSession session = req.getSession();
        User user = (User) session.getAttribute("user");
        System.out.println(user.getFullname());
        req.getRequestDispatcher("/secure.jsp").forward(req, resp);
    }
}
```

```
<%@ page pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"></head>
<body>
    <h1>Secure Page</h1>
</body>
</html>
```

```
@WebFilter("/secure")
public class AuthFilter implements Filter{
    @Override public void destroy() {}
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        HttpSession session = req.getSession();
        User user = (User) session.getAttribute("user");
        if(user == null){ // chưa có user trong sesion
            session.setAttribute("secureUrl", req.getRequestURI()); // lưu địa chỉ URL vào session
            resp.sendRedirect(req.getContextPath()+" /login"); // chuyển sang trang đăng nhập
        } else {
            chain.doFilter(request, response); // chuyển tiếp đến servlet
        }
    }
    @Override public void init(FilterConfig filterConfig) throws ServletException {}
}
```

- ☐ Sau khi đăng nhập thành công cần phải trở lại trang bảo mật trước đó. Bổ sung đoạn mã sau vào cuối vị trí mã đăng nhập thành công trong doPost() của @WebServlet("/login")

```
String secureUrl = (String) req.getSession().getAttribute("secureUrl");
if(secureUrl != null){
    resp.sendRedirect(secureUrl);
    return;
}
```



DEMOSTATION

- ✓ Giới thiệu Servlet Filter
- ✓ Xây dựng Filter
- ✓ Vòng đời của Filter
- ✓ Xây dựng Filter
- ✓ Sắp xếp thứ tự thực hiện của các filter
- ✓ Chia sẻ HttpServletRequest và HttpServletResponse





Cảm ơn