



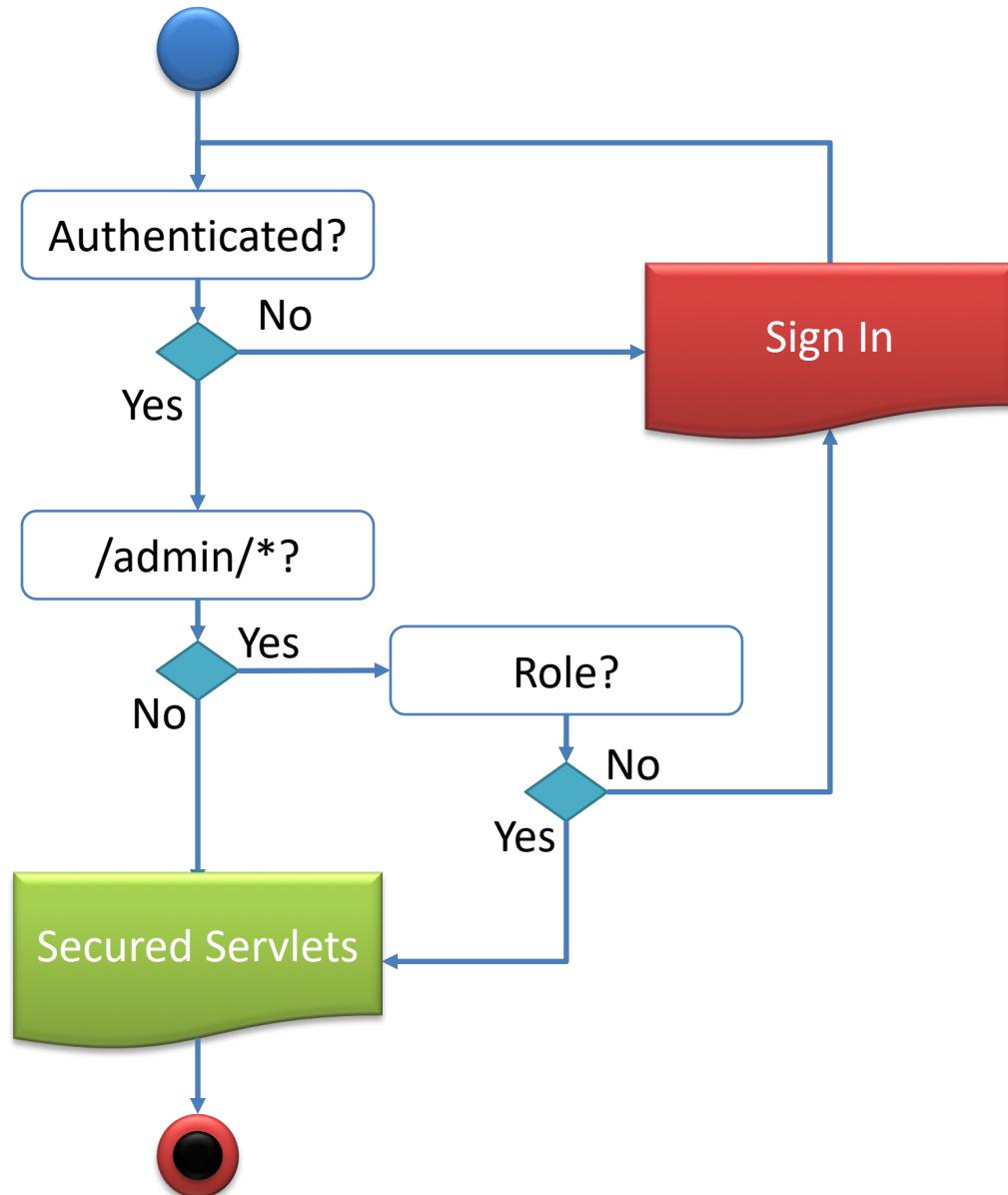
## **SECURITY WITH SERVLET FILER**

---

### **LẬP TRÌNH JAVA #4 (P6.2)**

- Thực hiện Authorization
- Xây dựng giải pháp bảo mật theo yêu cầu của Assignment
- Xây dựng các lớp tiện ích hỗ trợ lập trình Servlet
  - ❖ RRSharer
  - ❖ XParam
  - ❖ XCookie





- Kiểm tra đăng nhập
- Kiểm tra tài nguyên (đường dẫn) yêu cầu truy xuất có phù hợp với vai trò được phân hay không

# GIẢI PHÁP SECURITY THEO YÊU CẦU CỦA ASSIGNMENT

## ❑ Authentication: Yêu cầu đăng nhập

- ❖ "/account/change-password": Đổi mật khẩu
- ❖ "/account/edit-profile": Cập nhật thông tin tài khoản
- ❖ "/video/like/\*": Thích tiểu phẩm video
- ❖ "/video/share/\*": Chia sẻ tiểu phẩm video

## ❑ Authorization: Yêu cầu phải là Administrator (quản trị viên)

- ❖ "/admin/\*": Các trang quản trị

```
@WebFilter({  
    "/account/change-password",  
    "/account/edit-profile",  
    "/video/like/*",  
    "/video/share/*",  
    "/admin/*"  
})  
public class AuthFilter implements Filter{  
    @Override  
    public void destroy() {}  
  
    @Override  
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  
        throws IOException, ServletException {...}  
  
    @Override  
    public void init(FilterConfig filterConfig) throws ServletException {}  
}
```

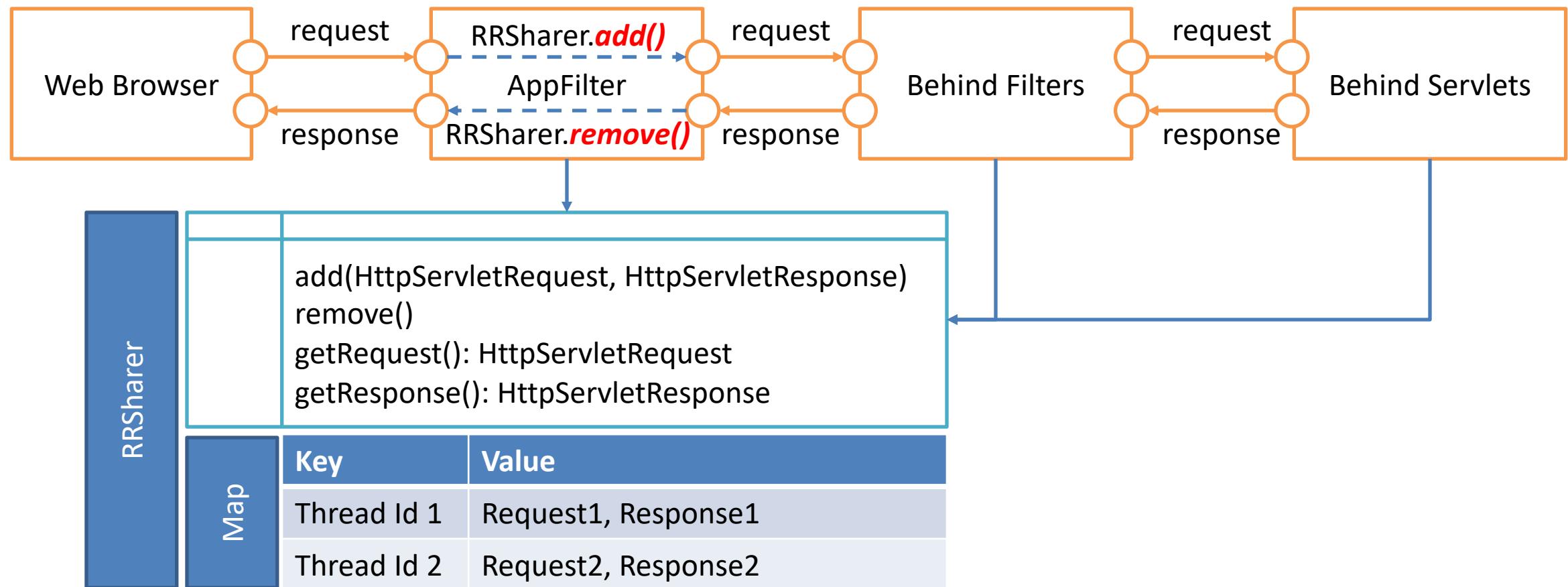
```
@Override  
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  
    throws IOException, ServletException {  
    HttpServletRequest req = (HttpServletRequest) request;  
    HttpServletResponse resp = (HttpServletResponse) response;  
    HttpSession session = req.getSession();  
    User user = (User) session.getAttribute("user");  
    String uri = req.getRequestURI();  
    if((user == null) || (uri.contains("/admin/") && !user.getAdmin())) {  
        session.setAttribute("secureUrl", uri);  
        resp.sendRedirect(req.getContextPath() + "/login");  
    } else {  
        chain.doFilter(request, response);  
    }  
}
```



# DEMOSTATION

---

- Lớp RRSharer là lớp nắm giữ và chia sẻ request và response hiện tại cho các thành phần hoạt động trong cùng một Thread.



```
@WebFilter("/*")
public class AppFilter implements Filter{
    @Override
    public void destroy() {}
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        request.setCharacterEncoding("utf-8");
        response.setCharacterEncoding("utf-8");
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        RRSharer.add(req, resp);
        chain.doFilter(request, response);
        RRSharer.remove();
    }
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {}
}
```

```
public class RRSharer {  
    private static Map<Long, HttpServletRequest> requests = new HashMap<>();  
    private static Map<Long, HttpServletResponse> responses = new HashMap<>();  
    public static void add(HttpServletRequest req, HttpServletResponse resp) {  
        requests.put(Thread.currentThread().getId(), req);  
        responses.put(Thread.currentThread().getId(), resp);  
    }  
    public static void remove() {  
        requests.remove(Thread.currentThread().getId());  
        responses.remove(Thread.currentThread().getId());  
    }  
    public static HttpServletRequest getRequest() {  
        return requests.get(Thread.currentThread().getId());  
    }  
    public static HttpServletResponse getResponse() {  
        return responses.get(Thread.currentThread().getId());  
    }  
}
```



# DEMOSTATION

---

# XÂY DỰNG CÁC LỚP TIỆN ÍCH XPARAM VÀ XCOOKIE

## XParam

```
getString(String name, String defaultValue): String  
getInt(String name, int defaultValue): int  
getDouble(String name, double defaultValue): double  
getBoolean(String name, boolean defaultValue): boolean  
getDate(String name, String pattern): Date  
getFile(String name, String folder): File  
getBean(Class<T> beanClass): T
```

## XCookie

```
get(String name): Cookie  
getValue(String name): String  
add(String name, String value, int expiry): Cookie  
delete(String name)
```

*Đọc giá trị tham số và chuyển đổi sang kiểu thích hợp*

use

## RRSharer

```
-requests: Map<HttpServletRequest>  
-responses: Map<HttpServletResponse>  
  
add(HttpServletRequest, HttpServletResponse)  
remove()  
getRequest(): HttpServletRequest  
getResponse(): HttpServletResponse
```

use

*Làm việc với cookie*

- ❑ Cung cấp các phương thức tiện ích hỗ trợ đọc giá trị tham số và chuyển đổi sang kiểu phù hợp

```
public class XParam {  
    static public String getString(String name, String defaultValue) {...}  
    static public int getInt(String name, int defaultValue) {...}  
    static public double getDouble(String name, double defaultValue) {...}  
    static public boolean getBoolean(String name, boolean defaultValue) {...}  
    static public Date getDate(String name, String pattern) {...}  
    static public File getFile(String name, String folder) {...}  
    static public <T> T getBean(Class<T> beanClass) {...}  
}
```

## XPARAM.GETSTRING() & XPARAM.GETINT()

```
static public String getString(String name, String defaultValue) {  
    HttpServletRequest request = RRSharer.getRequest();  
    String value = request.getParameter(name);  
    if(value == null) {  
        return defaultValue;  
    }  
    return value;  
}  
  
static public int getInt(String name, int defaultValue) {  
    String value = getString(name, String.valueOf(defaultValue));  
    return Integer.valueOf(value);  
}
```

## XPARAM.GETDOUBLE() & XPARAM.GETBOOLEAN()

---

```
static public double getDouble(String name, double defaultValue) {  
    String value = getString(name, String.valueOf(defaultValue));  
    return Double.valueOf(value);  
}  
  
static public boolean getBoolean(String name, boolean defaultValue) {  
    String value = getString(name, String.valueOf(defaultValue));  
    return Boolean.valueOf(value);  
}
```

```
static public Date getDate(String name, String pattern) {  
    String value = getString(name, null);  
    SimpleDateFormat format = new SimpleDateFormat(pattern);  
    try {  
        return format.parse(value);  
    } catch (ParseException e) {  
        return null;  
    }  
}
```

```
static public File getFile(String name, String folder) {  
    HttpServletRequest request = RRSharer.getRequest();  
    try {  
        Part part = request.getPart(name);  
        File dir = new File(request.getServletContext().getRealPath(folder));  
        if(!dir.exists()) {  
            dir.mkdirs();  
        }  
        File file = new File(dir, part.getSubmittedFileName());  
        part.write(file.getAbsolutePath());  
        return file;  
    } catch (Exception e) {  
        throw new RuntimeException(e);  
    }  
}
```

```
static public <T> T getBean(Class<T> beanClass) {  
    DateTimeConverter dtc = new DateConverter(new Date());  
    dtc.setPatterns(new String[] {"MM/dd/yyyy", "yyyy-MM-dd"});  
    ConvertUtils.register(dtc, Date.class);  
    try {  
        T bean = beanClass.getDeclaredConstructor().newInstance();  
        HttpServletRequest request = RRSharer.getRequest();  
        BeanUtils.populate(bean, request.getParameterMap());  
        return bean;  
    } catch (Exception e) {  
        throw new RuntimeException(e);  
    }  
}
```



# DEMOSTATION

---

- ❑ XCookie cung cấp các phương thức tiện ích hỗ trợ làm việc với cookie

```
public class XCookie {  
    // đọc cookie  
    static public Cookie get(String name) {...}  
    // đọc giá trị cookie  
    static public String getValue(String name) {...}  
    // tạo và gửi cookie về client  
    static public Cookie add(String name, String value, int expiry) {...}  
    // xóa cookie  
    static public void delete(String name) {...}  
}
```

```
static public Cookie get(String name) {  
    HttpServletRequest request = RRSharer.getRequest();  
    Cookie[] cookies = request.getCookies();  
    if(cookies != null) {  
        for(Cookie cookie: cookies) {  
            if(cookie.getName().equalsIgnoreCase(name)) {  
                return cookie;  
            }  
        }  
    }  
    return null;  
}
```

```
static public String getValue(String name) {  
    Cookie cookie = XCookie.get(name);  
    if(cookie != null) {  
        String value = cookie.getValue();  
        byte[] bytes = Base64.getDecoder().decode(value);  
        return new String(bytes);  
    }  
    return "";  
}
```

```
static public Cookie add(String name, String value, int expiry) {  
    byte[] bytes = value.getBytes();  
    String encodedValue = Base64.getEncoder().encodeToString(bytes);  
    Cookie cookie = new Cookie(name, encodedValue);  
    cookie.setMaxAge(expiry);  
    cookie.setPath("/");  
    HttpServletResponse response = RRSharer.getResponse();  
    response.addCookie(cookie);  
    return cookie;  
}  
  
static public void delete(String name) {  
    XCookie.add(name, "", 0);  
}
```



# DEMOSTATION

---

- ✓ Giới thiệu Security
- ✓ Xây dựng trang đăng nhập
- ✓ Giới thiệu Authentication và Authorization
- ✓ Thực hiện Authentication
- ✓ Thực hiện Authorization





Cảm ơn