BUỔI 4 - FILTER

CYBERSOFT.EDU.VN

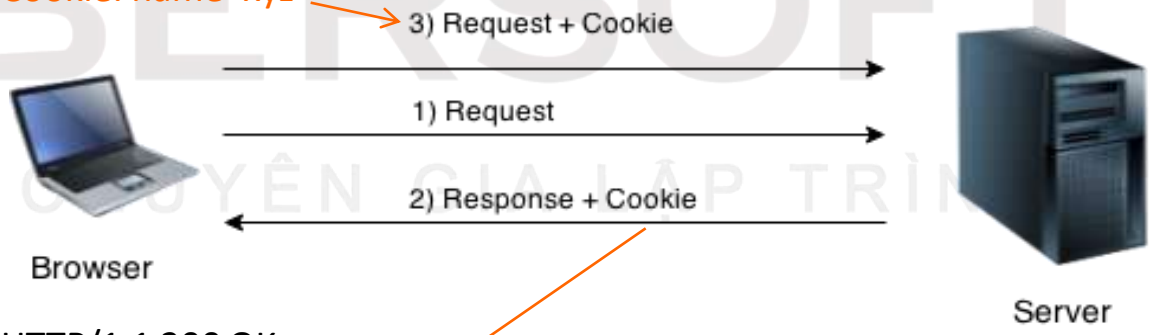
Nội dung

- Khái niệm về Cookie và cách sử dụng.
- Khái niệm và cách sử dụng Session.
- So sánh sự khác biệt giữa cookie và session.
- Filter trong Servlet.
- Xây dựng Filter kiểm tra đăng nhập trong Servlet.
- Thư viện Gson và cách sử dụng.

Cookies

- **Cookie** là một đoạn văn bản được tạo ra để **lưu trữ tạm thời một số thông tin trên trình duyệt** người dùng.
- **Cookie** được truyền từ Server tới Browser và được **lưu trữ trên trình duyệt** khi truy cập vào ứng dụng.
- Mỗi khi người dùng tải ứng dụng, **trình duyệt sẽ gửi cookie để thông báo cho Server** về hoạt động trước đó của người dùng.

GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz



HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT; path=/; domain=myclass.vn
Connection: close
Content-Type: text/html

Phương thức trong Cookie

- void **setMaxAge(**int time)

  Đặt thời gian sống cho cookie (giây).

  Ví dụ: *cookie.setMaxAge(60*60*24);*

- int **getMaxAge**()

  Trả về thời gian sống của cookie.

- String **getName**()

  Trả về tên của cookie.

  Ví dụ: *String name = cookies[i].getName();*

- void **setValue**(String value)

  Gán giá trị cho cookie.

- String **getValue**()

  Trả về giá trị của cookie.

  Ví dụ: *String value = cookies[i].getValue();*

❖ Nếu muốn xóa một cookie bạn chỉ cần đặt maxAge của cookie đó thành về 0.

cookie**.setMaxAge**(0);

Sử dụng Cookie

❖ Thiết lập cookie

```
// Tạo mới cookie
Cookie cookie = new Cookie("name","Cybersoft");
cookie.setMaxAge(60*60*24);  // Đặt thời gian sống
response.AddCookie(cookie);  // Gửi cookie vào Http Reponse
```

❖ Đọc cookie

```
// Lấy danh sách cookie từ Http Request
Cookie[] cookies = request.getCookies();
for(int i = 0; i < cookies.length; i++){
    String name = cookies[i].getName(); // Lấy tên cookie
    String value = cookies[i].getValue();  // Lấy giá trị
}
```

Khi nào sử dụng cookie?

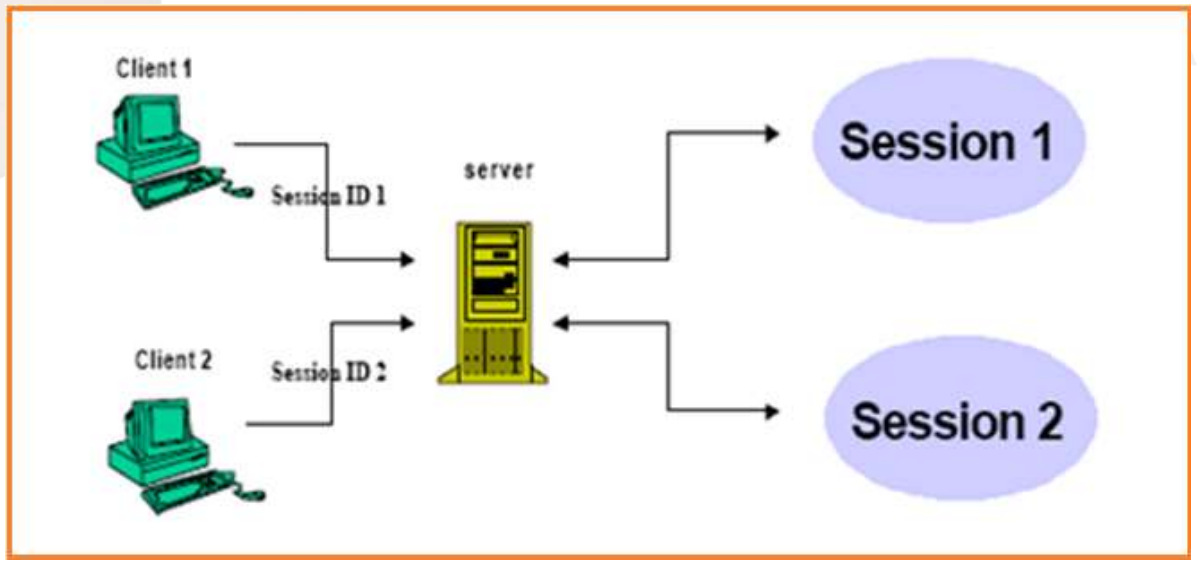
- **Cookie** được lưu trữ trên máy người dùng nên có thể dễ dàng sửa đổi (không bảo mật) vì vậy mà nó được ứng dụng trong một số trường hợp theo dõi thông tin như:
  - ✓Theo dõi tần suất người dùng ghé thăm website.
  - ✓Thời gian truy cập website.
  - ✓Sở thích người dùng.
  - ✓Các mục người dùng đã nhấp chuột.
  - ✓...

Session

- **Session** là một **phiên làm việc** hay **khoảng thời gian** user làm việc với một ứng dụng.
- **Session** được **lưu trữ ở Server** và có thể **tồn tại qua mọi request**.
- Một session bắt đầu khi user truy cập vào hệ thống lần đầu tiên và kết thúc khi user thoát khỏi hệ thống.
- Thông tin session khó sửa đổi (**bảo mật hơn cookie**) nên thường được sử dụng trong việc lưu trữ thông tin đăng nhập của người dùng, thông tin giỏ hàng, ...

Cách server nhận biết session

- Khi user truy cập vào 1 ứng dụng lần đầu, user sẽ được **cấp phát một SessionID duy nhất** để xác định Session của người đó.
- Mỗi Session sinh ra đồng thời sẽ **tạo ra một Cookie để lưu trữ SessionID tương ứng với nó** được lưu trữ trên trình duyệt, **SessionID sẽ được gửi lên Server trong mỗi request** nhờ đó Server sẽ biết được Session đó là của client nào.



Phương thức trong Session

- HttpSession **getSession**()

  Khởi tạo một Session.

- void **setAttribute**(String str, Object obj)

  Gán giá trị cho Session.

- void **setMaxInactiveInterval**(int time)

  Đặt thời gian sống cho Session.

- Object **getAttribute**(String name)

  Lấy giá trị của Session.

- void **removeAttribute**(String name)

  Hủy một Session.

- void **invalidate**()

  Hủy tất cả các Session.

Sử dụng Session

```
// Khởi tạo một session
HttpSession session = request.getSession();
// Gán giá trị cho session
session.setAttribute(“taiKhoan”, Account);
// Lấy giá trị của session
Object account = session.getAttribute(“taiKhoan”);
// Đặt thời gian sống cho session (giây)
session.setMaxInactiveInterval(3600) ;
session.removeAttribute(“taiKhoan”);  // Hủy một session
session.invalidate(); // Hủy tất cả session
```

So sánh Session & Cookie

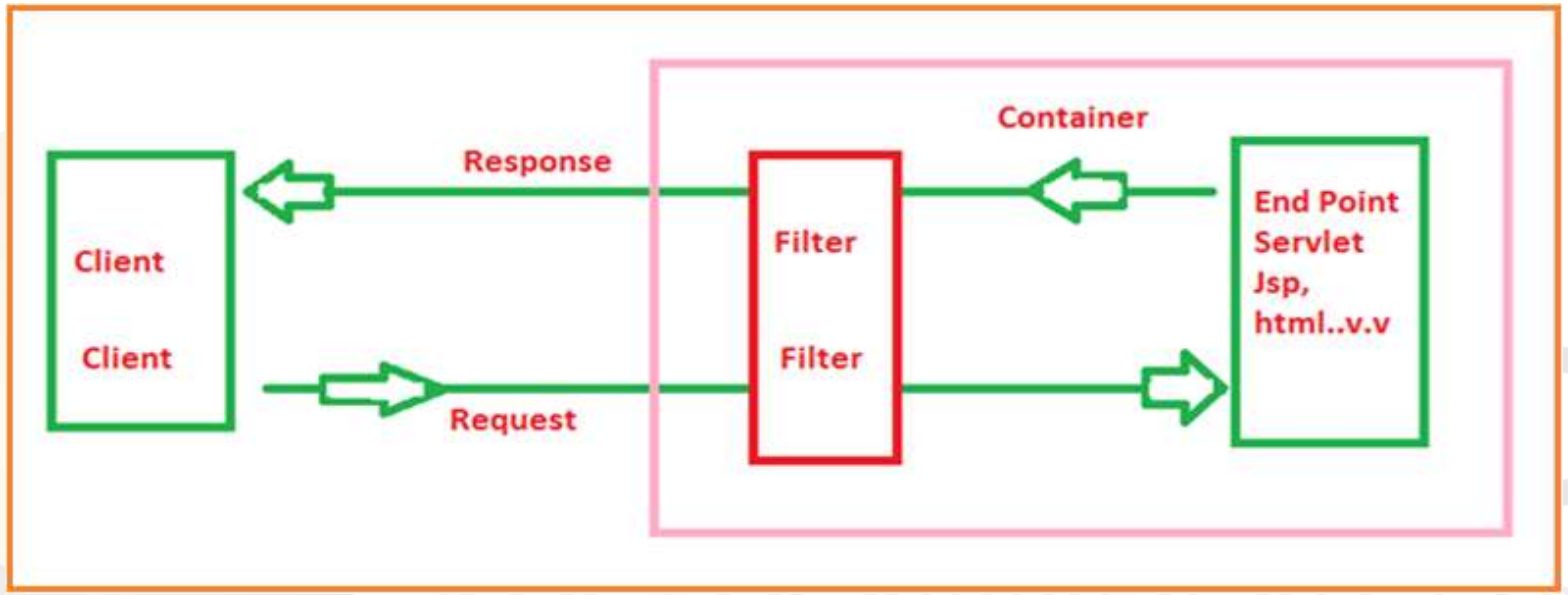
❑ Cookie

- ✓ Cookie được lưu trữ trên trình duyệt người dùng.
- ✓ Dữ liệu Cookie có thể bị sửa đổi, đánh cắp khi chúng được lưu trữ ở phía máy khách.
- ✓ Dữ liệu Cookie chỉ mất đi khi hết hạn.

❑ Session

- ✓ Session được lưu trữ trên máy chủ (server).
- ✓ Dữ liệu Session khó bị thay đổi, đánh cắp do được lưu trữ trên máy chủ.
- ✓ Session mất đi khi hết phiên làm việc (đóng trình duyệt).

Servlet Filter



- Là **các lớp trong Java** được sử dụng trong lập trình Servlet cho mục đích:
  - ✓Xử lý các **Request** từ phía Client **trước khi truy cập vào tài nguyên Backend**.
  - ✓Xử lý các **Response** trả về từ Server **trước khi chúng được gửi lại Client**.

Phương thức của Filter

- **doFilter**(request, response, chain)
  - ✓ Phương thức này được Container **gọi mỗi lần có request gửi đi hoặc có response trả về**.
- **init**(filterConfig)
  - ✓ Phương thức này được Container gọi để đặt Filter vào Service.
- **destroy()**
  - ✓ Phương thức này gọi khi Container hủy Filter khỏi Service.

Tạo một Servlet Filter

```
public class DemoFilter implements Filter{

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        // Code xử lý request trước khi đi vào Servlet đích
        chain.doFilter(request, response);
        // Code xử lý response trước khi trả về cho browser
    }

}
```

Cấu hình Filter bằng Xml

```
<filter>
  <filter-name>DemoFilter</filter-name>
  <filter-class>com.myclass.filter.DemoFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>DemoFilter</filter-name>
  <url-pattern>/home</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>DemoFilter</filter-name>
  <url-pattern>/home</url-pattern>
</filter-mapping>
```

Khai báo danh sách các url sẽ phải đi qua DemoFilter.

Servlet Filter bằng Annotation

```java
@WebFilter(urlPatterns = {"/home", "/role"})
public class DemoFilter implements Filter{

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        // Code xử lý request trước khi đi vào Servlet đích
        chain.doFilter(request, response);
        // Code xử lý response trước khi trả về cho browser
    }
}
```

Khai báo các danh sách các url sẽ phải đi qua Filter này.

Bài tập

- Viết chức năng đăng nhập
  - Form đăng nhập gồm có email và mật khẩu.
  - Truy vấn database để kiểm tra đăng nhập.
  - Sử dụng Session để lưu trữ thông tin.
- Tạo filter có tên AuthFilter
  - Cấu hình bằng Annotation
  - Viết chức năng kiểm tra đăng nhập
    - + Nếu chưa đăng nhập cho chuyển về trang đăng nhập.
    - + Nếu đã đăng nhập chuyển qua trang thông tin người dùng.

Bài tập

❑ Các bước thực hiện

- ✓ **Bước 1**: Tạo form đăng nhập.
- ✓ **Bước 2**: Tạo LoginController.
- ✓ **Bước 3**: Xử lý đăng nhập
  - Truy vấn database kiểm tra đăng nhập.
  - Sử dụng Session lưu thông tin người dùng sau khi đăng nhập thành công.
- ✓ **Bước 4**: Tạo Filter kiểm tra đăng nhập
  - ❖ Kiểm tra session
    - Nếu khác null cho request đi tiếp vào Servlet đích.
    - Ngược lại, chặn request và redirect về trang đăng nhập.

LoginController

```java
@WebServlet(name = "LoginServlet", urlPatterns = "/login")
public class LoginServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;
    private LoginDAO loginDAO = null;
    public LoginServlet() {
        loginDAO = new LoginDAO();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        req.getRequestDispatcher("/views/login/index.jsp").forward(req, resp);
    }

    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
}
```

Xử lý đăng nhập

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
    String email = req.getParameter("email");
    String password = req.getParameter("password");

    LoginDto loginDto = new LoginDto(email, password);
    int result = loginDAO.checkLogin(loginDto);
    if(result > 0) {
        HttpSession session = req.getSession();
        session.setAttribute("LOGIN_USER", loginDto);
        session.setMaxInactiveInterval(1800);
        resp.sendRedirect(req.getContextPath() + "/home");
    }
    else {
        req.setAttribute("message", "Sai tên đăng nhập hoặc mật khẩu!");
        req.setAttribute("login", loginDto);
        req.getRequestDispatcher("/views/login/index.jsp").forward(req, resp);
    }
}
```

AuthFilter

```java
@WebFilter(urlPatterns = "/*")
public class AuthFilter implements Filter{

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest)request;
        HttpServletResponse resp = (HttpServletResponse)response;

        if(!req.getServletPath().startsWith("/login")) {
            if(req.getSession().getAttribute("LOGIN_USER") != null) {
                chain.doFilter(request, response);
            }
            else {
                resp.sendRedirect(req.getContextPath() + "/login");
            }
        }
        else {
            chain.doFilter(request, response);
        }
    }
}
```

Bài tập nâng cao

❑ Viết chức năng phân quyền cho hệ thống.

✓ Chỉ cho phép người dùng có quyền “admin” mới được phép truy cập vào các RoleController và UserController.

✓ Các Controller còn lại bắt buộc phải đăng nhập mới cho phép truy cập (tất cả các quyền có thể truy cập).

❖ ***Gợi ý**: Sử dụng phương thức **getServletPath()** của đối tượng HttpServletRequest để check url.*

Thư viện Gson

❑Gson là gì?

➢ **Gson** là thư viện cung cấp các phương thức **toJson()** và **fromJson()** đơn giản để **chuyển các đối tượng Java sang JSON và ngược lại**.

❖ **Phương thức sử dụng:**

✓ **toJson()** : phương thức dùng để **chuyển Java Object sang chuỗi Json** (Serialization). Phương thức này **có 1 đối số là đối tượng cần chuyển sang chuỗi Json**.

✓ **fromJson()** : phương thức dùng để **chuyển chuỗi Json sang Java Object** (quá trình này được gọi là Deserialization). Phương thức này **có 2 đối số**, đối số **đầu tiên là chuỗi json**, đối số **thứ hai là kiểu dữ liệu Java Object ứng với chuỗi json**.

Thư viện Gson

❖ **Tải thư viện và kéo vào dự án.**

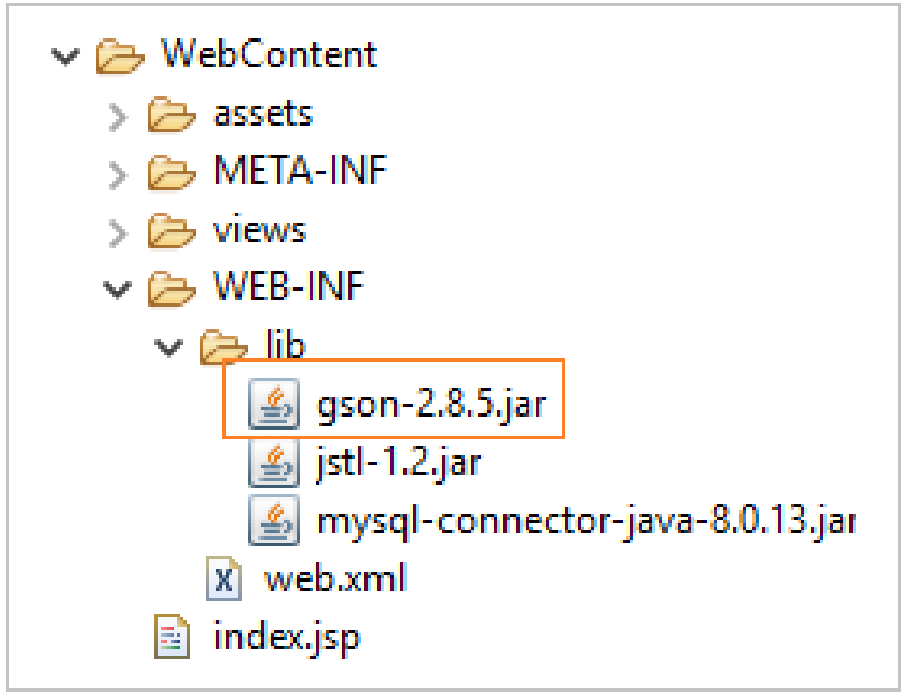
➢ **Link tải: https://mvnrepository.com/artifact/com.google.code.gson/gson/2.8.5**

Tải thư viện

License	Apache 2.0
Categories	JSON Libraries
Date	(May 22, 2018)
Files	jar (235 KB) View All
Repositories	Central Mulesoft
Used By	9,865 artifacts

Tải về

Kéo vào dự án



Thư viện Gson

❑ toJson()

Chuyển đối tượng sang JSON

```
Person person = new Person("GP Coder", "Viet Nam");
Gson gson = new Gson();
// Serialization
String json = gson.toJson(person);
System.out.println(json);
// {"name":"GP Coder","location":"Viet Nam"}
```

❑ fromJson()

Chuyển JSON sang đối tượng

```
Person person = new Person("GP Coder", "Viet Nam");
Gson gson = new Gson();
// Serialization
String json = gson.toJson(person);
System.out.println(json);
// {"name":"GP Coder","location":"Viet Nam"}

// Deserialization
Person person2 = gson.fromJson(json, Person.class);
System.out.println(person2);
```

Trả JSON về Client

- **Trả dữ liệu về Client theo kiểu JSON**.

- Các bước thực hiện:

1. Tạo mới đối tượng PrintWriter.
2. Set ContentType trả về là “application/json”.
3. Tạo mới đối tượng Gson.
4. Sử dụng phương thức toJson() để chuyển đổi dữ liệu về dạng JSON.
5. Sử dụng phương thức write() của PrintWriter để trả JSON về cho client.

Trả JSON về Client

```
Person person = new Person("GP Coder", "Viet Nam");

PrintWriter out = resp.getWriter();
resp.setContentType("application/json");
resp.setCharacterEncoding("UTF-8");

Gson gson = new Gson();
// Serialization
String json = gson.toJson(person);
// Trả JSON về cho client
out.write(json);
out.flush();
```

AJAX

❑ Ajax là gì?

- ✓ **Ajax** là một kỹ thuật lập trình được viết bằng Javascript để tạo các ứng dụng web nhanh và mượt mà hơn.
- ✓ **Ajax** giúp cập nhật các phần nhỏ trong trang mà không cần load lại trang.
- ✓ Để sử dụng Ajax cần phải nhúng thư viện **Jquery** vào trang web (html).

- ❖ **Tham khảo**: https://www.w3schools.com/jquery/jquery_ref_ajax.asp
- ❖ **Thư viện Jquery**: https://jquery.com/download/

Sử dụng AJAX

```
$.ajax({
    url: '/account/login', // url đích
    type: 'POST', // loại requsest (GET/POST/PUT/DELETE)
    dataType: 'json', // Kiểu dữ liệu nhận về từ server
    contentType: 'application/json', // Kiểu dữ liệu gửi lên server
    data: { // Dữ liệu gửi lên server để xử lý
        username: 'admin',
        password: '123456'
    }
})
.done(function(response){
    // Xử lý kết quả trả về ở đây
})
.fail(function(err){
    // Xử lý lỗi ở đây
});
```

DỰ ÁN CRM

❑ Nội dung thực hiện

- ➢ Thiết kế giao diện trang đăng nhập.
- ➢ Tạo module đăng nhập – đăng xuất.
- ➢ Cấu hình filter cho ứng dụng.
- ➢ Tạo module quản lý công việc.

Tổng kết

❑ Nội dung cần nắm

- ✓ Cookie trong Servlet.
- ✓ Session trong Servlet.
- ✓ Tạo và cấu hình Filter.
- ✓ Sử dụng Filter để kiểm tra đăng nhập.
- ✓ Sử dụng thư viện Gson để làm việc với JSON.
- ✓ Sử dụng Ajax để gửi và nhận dữ liệu từ server.
- ✓ Áp dụng kiến thức vào dự án CRM.