MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- Áp dụng các scope phù hợp để chia sẻ dữ liệu
- Xử lý các sự kiện xảy ra trong ứng dụng web
- Giải thích được cơ chế hoạt động của bộ lọc
- Áp dụng bộ lọc để ghi nhận lịch sử truy xuất và bảo vệ tài nguyên

PHẦN I: SCOPES VÀ WEBLISTENER

BÀI 1: XÂY DỰNG BỘ ĐẾM SỐ KHÁCH THĂM WEB

Bộ đếm khách thăm web là bộ đếm số phiên làm việc (mỗi lần khách ghé thăm chỉ được đếm đúng một lần).

Hướng dẫn:

- Hãy tạo AppListener có cấu trúc tổ chức như sau

```
@WebListener
public class AppListener implements HttpSessionListener, ServletContextListener{
        @Override
        public void contextDestroyed(ServletContextEvent e) {
                // TODO: ghi số đếm trong application scope vào file
        }

        @Override
        public void contextInitialized(ServletContextEvent e) {
                // TODO: đọc số đếm trước đây từ file vào application scope
        }

        @Override
        public void sessionCreated(HttpSessionEvent e) {
                // TODO: tăng số đếm trong application scope lên một
        }

        @Override
        public void sessionDestroyed(HttpSessionEvent e) {
        }
}
```

- Đọc số đếm trước đây từ file vào application scope

```
ServletContext application = e.getServletContext();
Integer visitors = 0;
try {
    String path = application.getRealPath("/visitors.txt"); // đặt tại webroot
    List<String> lines = Files.readAllLines(Paths.get(path));
    visitors = Integer.valueOf(lines.get(0));
} catch (Exception e2) {
    visitors = 100000; // khởi đầu số khách
}
application.setAttribute("visitors", visitors);
```

- Tăng số đếm trong application scope lên một

```
HttpSession session = e.getSession();
ServletContext application = session.getServletContext();
Integer visitors = (Integer) application.getAttribute("visitors");
application.setAttribute("visitors", visitors + 1);
```

- Ghi số đếm trong application scope vào file

```
ServletContext application = e.getServletContext();
Integer visitors = (Integer) application.getAttribute("visitors");
try {
    String path = application.getRealPath("/visitors.txt");
    byte[] data = String.valueOf(visitors).getBytes();
    Files.write(Paths.get(path), data, StandardOpenOption.CREATE);
} catch (Exception e2) {
    e2.printStackTrace();
}
```

- Hiển thị số đếm trang trang JSP bất kỳ

```
<h2>Visitors: ${applicationScope.visitors}</h2>
```

BÀI 2: LƯU VẾT NGƯỜI SỬ DỤNG

Giao diện người sử dụng sẽ có một số thay đổi tùy thuộc vào việc đăng nhập hay chưa và đăng nhập với vai trò gì. Hãy thực hiện bảo mật giao diện theo yêu cầu sau:

- Đầu mỗi trang jsp hiển thị dòng chữ **Welcome you** nếu chưa đăng nhập, còn nếu đăng nhập rồi thì hiển thị **Welcome <<fullname>>,** Đăng xuất
- Nếu đăng nhập rồi và vai trò là admin thì hiển thị thêm link Quản trị

Hướng dẫn:

- Thực hiện lại bài đăng nhập trước đây (Lab5 bài 4) và chú ý đến dòng mã lệnh màu đỏ sau khi đăng nhập thành công. Mục đích của dòng mã này là để duy trì tài khoản đăng nhập vào session scope với tên attribute là user

```
String id = req.getParameter("username");
String pw = req.getParameter("password");

try {
        UserDAO dao = new UserDAO();
        User user = dao.findById(id);
        if(!user.getPassword().equals(pw)) {
                req.setAttribute("message", "Sai mật khẩu!");
        }
        else {
                req.setAttribute("message", "Đăng nhập thành công!");
                req.getSession().setAttribute("user", user);
        }
} catch (Exception e) {
        req.setAttribute("message", "Sai tên đăng nhập!");
}
```

- Viết mã bảo mật giao diện trên các trang jsp theo yêu cầu đặt ra ở trên như sau:

```
<c:choose>
        <c:when test="${empty sessionScope.user}">
                Welcome you
        </c:when>
        <c:otherwise>
                Welcome ${sessionScope.user.fullname}
                <a href="/fpoly/account/sign-out">Đăng xuất</a>
                <c:if test="${sessionScope.user.admin}">
                        <a href="/fpoly/admin/home/index">Quản trị</a>
                </c:if>
        </c:otherwise>
</c:choose>
```

PHẦN II: WEBFILTER

BÀI 3: XÂY DỰNG APPFILTER PHỤC VỤ CHUNG

Hãy xây dựng HttpFilter, RRSharer và AppFilter như hướng dẫn trong bài giảng để thiết lập chế độ mã hóa ký tự là utf-8 và hia sẻ request và response cho các thành phần hoạt động trong cùng một yêu cầu từ người sử dụng.

Dựa trên requests và responses được chia sẻ trong RRSharer, hãy xây dựng các thư viện tiện ích XForm, XCookie, XScope như mô tả sau đây để phục vụ cho lập trình servlet trở nên đơn giản hơn sau này.

XForm là lớp tiện ích cung cấp các phương thức tiện ích làm việc với form nhập bao gồm cả upload file và BeanUtils.

```
import static com.poly.utils.RRSharer.request;

public class XForm {
	/**
	 * Kiểm tra sự tồn tại của tham số
	 * @param name tên tham số
	 * @return true nếu tồn tại, ngược lại là false
	 */
	public static boolean exist(String name) {
		return request().getParameter(name) != null;
	}
	/**
	 * Đọc chuỗi từ tham số form
	 * @param name tên tham số form
	 * @param defval giá trị mặc định
	 * @return Giá trị tham số hoặc defval nếu tham số không tồn tại
	 */
	public static String getString(String name, String defval) {
		String value = request().getParameter(name);
		return value == null ? defval : value;
	}
	/**
	 * Đọc số nguyên từ tham số form
	 * @param name tên tham số form
	 * @param defval giá trị mặc định
	 * @return Giá trị tham số hoặc defval nếu tham số không tồn tại
	 */
	public static int getInt(String name, int defval) {
		String value = getString(name, String.valueOf(defval));
		return Integer.parseInt(value);
	}
	/**
	 * Đọc số thực từ tham số form
	 * @param name tên tham số form
	 * @param defval giá trị mặc định
	 * @return Giá trị tham số hoặc defval nếu tham số không tồn tại
	 */
```

```
        public static double getDouble(String name, double defval) {
                String value = getString(name, String.valueOf(defval));
                return Double.parseDouble(value);
        }
        /**
         * Đọc giá trị boolean từ tham số form
         * @param name tên tham số form
         * @param defval giá trị mặc định
         * @return Giá trị tham số hoặc defval nếu tham số không tồn tại
         */
        public static boolean getBoolean(String name, boolean defval) {
                String value = getString(name, String.valueOf(defval));
                return Boolean.parseBoolean(value);
        }
        /**
         * Đọc thời gian từ tham số form
         * @param name tên tham số form
         * @param defval giá trị mặc định
         * @return Giá trị tham số hoặc defval nếu tham số không tồn tại
         */
        public static Date getDate(String name, Date defval) {
                SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
                String date = getString(name, sdf.format(defval));
                try {
                        return sdf.parse(date);
                } catch (Exception e) {
                        return defval;
                }
        }
        /**
         * Lưu file upload vào thư mục với tên duy nhất
         * @param name tên tham số form
         * @param folder thư mục chứa file
         * @return File kết quả hoặc null nếu không upload
         */
        public static File save(String name, String folder) {
                File dir = new File(XHttp.getRealPath(folder));
                if(!dir.exists()) {
                        dir.mkdirs();
                }
                try {
                        Part part = request().getPart(name);
                        if(part != null && part.getSize() > 0) {
                                String fn = System.currentTimeMillis() +
part.getSubmittedFileName();
```

```
                String filename = Integer.toHexString(fn.hashCode()) +
fn.substring(fn.lastIndexOf("."));
                File file = new File(dir,  filename);
                part.write(file.getAbsolutePath());
                return file;
            }
            return null;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
    /**
     * Đọc các giá trị tham số form vào các thuộc tính cùng tên của bean
     * @return Bean chứa dữ liệu form
     */
    public static <T> T getBean(Class<T> clazz) {
        DateTimeConverter dtc = new DateConverter(new Date());
        dtc.setPattern("MM/dd/yyyy");
        ConvertUtils.register(dtc, Date.class);
        try {
            T bean = clazz.newInstance();
            BeanUtils.populate(bean, request().getParameterMap());
            return bean;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

XCookie là lớp tiện ích cung cấp các phương thức để tạo cookie và gửi về client để lưu lại cũng như đọc lại giá trị của cookie.

```
public class XCookie {
    /**
     * Tạo và gửi cookie về client
     * @param name tên cookie
     * @param value giá trị cookie
     * @param hours thời hạn cookie (giờ)
     * @return Cookie đã gửi về client
     */
    public static void add(String name, String value, int hours) {
        Cookie cookie = new Cookie(name, value);
        cookie.setMaxAge(hours * 60 * 60);
        cookie.setPath("/");
        RRSharer.response().addCookie(cookie);
    }
```

```
        /**
         * Đọc giá trị cookie
         * @param name tên cookie cần đọc
         * @param defval giá trị mặc định
         * @return Giá trị cookie hoặc defval nếu cookie không tồn tại
         */
        public static String get(String name, String defaultValue) {
                Cookie[] cookies = RRSharer.request().getCookies();
                if(cookies != null) {
                        for(Cookie cookie: cookies) {
                                if(cookie.getName().equalsIgnoreCase(name)) {
                                        return cookie.getValue();
                                }
                        }
                }
                return defaultValue;
        }
}
```

XScope là lớp tiện ích cung cấp các phương thức nhằm quản lý các attribute trong các scope

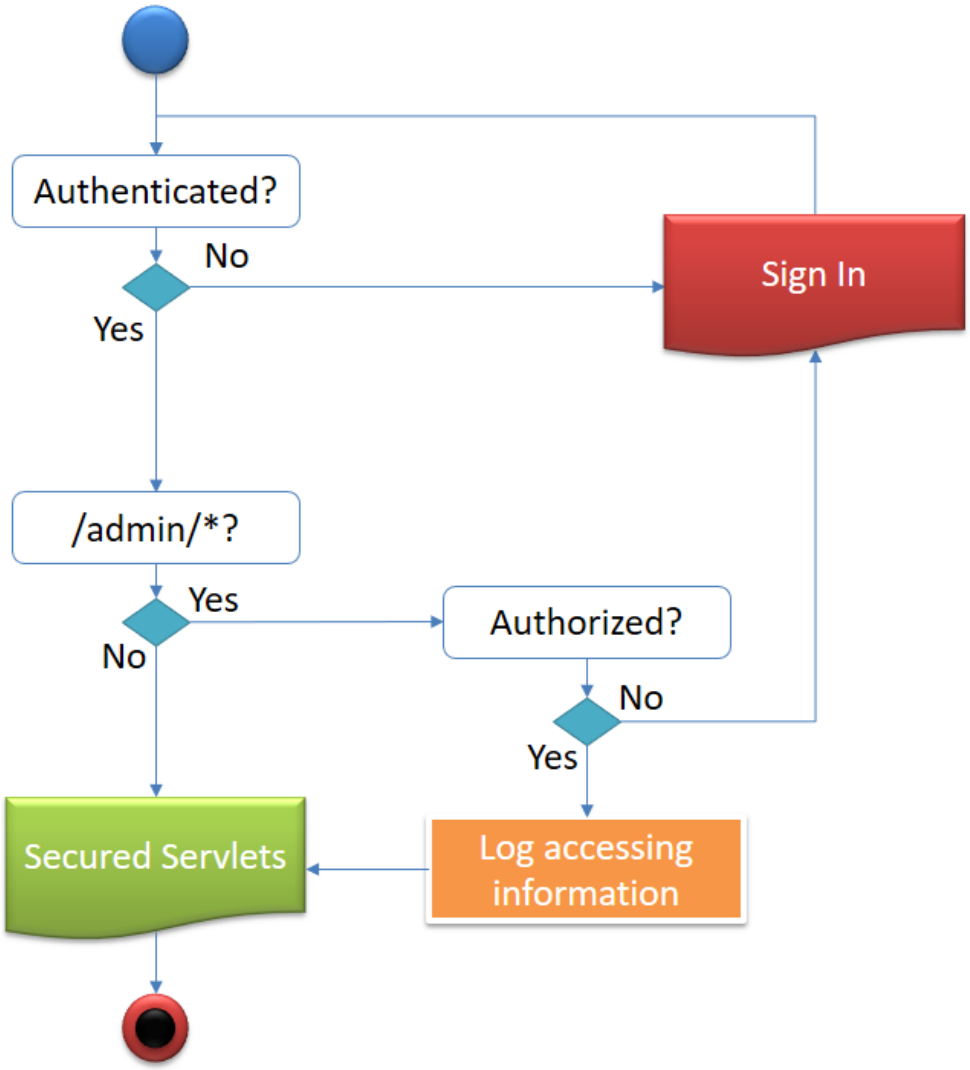
```
/**
 * Cung cấp các phương thức đọc/ghi attribute trong các scope
*/
public class XScope {
        public static HttpServletRequest request() {
                return RRSharer.request();
        }
        public static HttpSession session() {
                return request().getSession();
        }
        public static ServletContext application() {
                return request().getServletContext();
        }
        /**
         * Tạo attribute trong request scope
         * @param name tên attribute
         * @param value giá trị của attribute
         */
        public static void setRequest(String name, Object value) {
                request().setAttribute(name, value);
        }
        /**
         * Đọc attribute trong request scope
         * @param name tên attribute
```

```
     * @return Giá trị của attribute hoặc null nếu không tồn tại
     */
    @SuppressWarnings("unchecked")
    public static <T> T getRequest(String name) {
        return (T) request().getAttribute(name);
    }
    /**
     * Xóa attribute trong request scope
     * @param name tên attribute cần xóa
     */
    public static void removeRequest(String name) {
        request().removeAttribute(name);
    }

    /**
     * Tạo attribute trong session scope
     * @param name tên attribute
     * @param value giá trị của attribute
     */
    public static void setSession(String name, Object value) {
        session().setAttribute(name, value);
    }
    /**
     * Đọc attribute trong session scope
     * @param name tên attribute
     * @return Giá trị của attribute hoặc null nếu không tồn tại
     */
    @SuppressWarnings("unchecked")
    public static <T> T getSession(String name) {
        return (T) session().getAttribute(name);
    }
    /**
     * Xóa attribute trong session scope
     * @param name tên attribute cần xóa
     */
    public static void removeSession(String name) {
        session().removeAttribute(name);
    }

    /**
     * Tạo attribute trong application scope
     * @param name tên attribute
     * @param value giá trị của attribute
     */
    public static void setApplication(String name, Object value) {
```

```
            application().setAttribute(name, value);
        }
        /**
         * Đọc attribute trong application scope
         * @param name tên attribute
         * @return Giá trị của attribute hoặc null nếu không tồn tại
         */
        @SuppressWarnings("unchecked")
        public static <T> T getApplication(String name) {
            return (T) application().getAttribute(name);
        }
        /**
         * Xóa attribute trong application scope
         * @param name tên attribute cần xóa
         */
        public static void removeApplication(String name) {
            application().removeAttribute(name);
        }
}
```

BÀI 5: BẢO MẬT ỨNG DỤNG

Xây dựng AuthFilter để bảo vệ các chức năng trong ứng dụng theo mô tả của lưu đồ thuật toán sau đây:



Hướng dẫn:

- Tham khảo LoggerFilter và AuthFilter
- Trộn lẫn 2 Filter này để viết ra AuthFilter mới
- Nếu trong AuthFilter mới có sử dụng các lớp tiện ích XForm, XCookie và XScope thì hãy cấu hình web.xml để AppFilter chạy trước AuthFilter

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
      version="3.1">
  <filter-mapping>
    <filter-name>app</filter-name>
```

```
        <url-pattern/>
      </filter-mapping>
      <filter-mapping>
        <filter-name>auth</filter-name>
        <url-pattern/>
      </filter-mapping>
    </web-app>
```