Java 4 WEB

Lesson 12 - Home Task Overview

1. Synchronization?

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    synchronized (this) {
        var wr ter = resp.getWriter();
        var ses on = req.getSession();
        writer.wri (FORM);

        boolean servi IsWorking = (boolean) session.getAttribute("working");
        if (!serviceIsW king) {
            return;
        }

        var action = req.getPar meter("action");
        var actionValue = Integer parseInt(action);
        var name = req.getParameter("name");
        var value = req.getParameter("value");

        try {
            var operation = Operations.values()[actionValue];
            operation.execute(attributes, name, value);
        } catch (IllegalArgumentException e) {
            session.setAttribute("exception", e);
            resp.sendRedirect("/error");
        }

        printAttributes(resp);
    }
}
```

2. **doGet** is not the same to **doPost**

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    doGet(req, resp);
}
```

3. Tightly coupled code

```
try {
    var operation = Operations.values()[actionValue];
    operation.execute(attributes, name, value);
} catch (IllegalArgumentException e) {
    session.setAttribute("exception", e);
    resp.sendRedirect("/error");
}
```

4. Lookup by ordinal is unsafe

```
try {
    var operation = Operations.values()[actionValue];
    operation.execute(attributes, name, value);
} catch (IllegalArgumentException e) {
    session.setAttribute("exception", e);
    resp.sendRedirect("/error");
}
```

5. Follow single responsibility principle

```
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
    var httpServletRequest = (HttpServletRequest) request;
    var session = httpServletRequest.getSession();
    var presentDayOfWeek = LocalDate.now().getDayOfWeek();
    boolean todayIsNotWeekend = presentDayOfWeek != DayOfWeek.SATURDAY
            && presentDayOfWeek != DayOfWeek.SUNDAY;

    session.setAttribute("working", todayIsNotWeekend);
    chain.doFilter(request, response);
}
```

5. Follow single responsibility principle

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

    try {
       HttpServletRequest request = ((HttpServletRequest) req);
       if(isMicrosoftEdge(request)){
           logger.error("Browser Microsoft Edge not supported.");
           ServletContext servletContext = req.getServletContext();
           servletContext.setAttribute(ERROR_STATUS_CODE, 403);
           servletContext.setAttribute(ERROR_MESSAGE, "Browser Microsoft Edge not supported.");
           req.getRequestDispatcher("/error").forward(req,resp);
       }else {
           chain.doFilter(req, resp);
       }

    } catch (Exception ex) {
        logger.error("", ex);
        throw ex;
    }

}
```

6. Handle errors correctly

HTTP Status 500 ? Internal Server Error

Type Exception Report

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
java.lang.NullPointerException
        org.geekhub.lesson12.task1.error.ServletError.doGet(ServletError.java:20)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:634)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/9.0.14

7. Use correct dependency scope

```
dependencies {
    compile 'javax.servlet:javax.servlet-api:4.0.0'
```

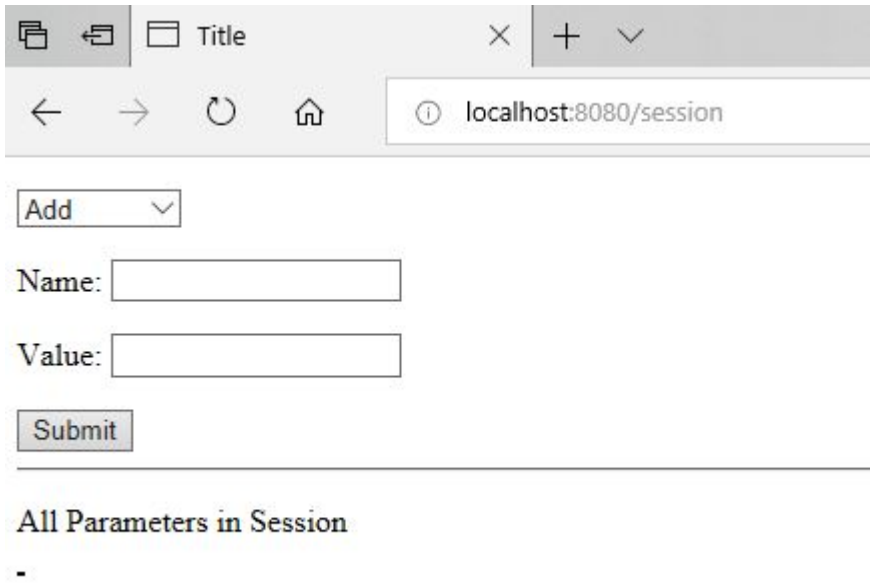
8. Filtering should block request

```
@WebFilter(filterName = "holidayFilter")
public class HolidayFilter implements Filter {

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {

        LocalDate localDate = LocalDate.now();
        DayOfWeek dayOfWeek = localDate.getDayOfWeek();
        if (dayOfWeek == DayOfWeek.SATURDAY || dayOfWeek == DayOfWeek.SUNDAY) {
            System.out.println("chain.doFilter(request, response);");
        } else {
            chain.doFilter(request, response);
        }
    }
}
```

8. Test your code



9. Continuous integration should not fail

10. Useless code

```
@WebFilter(filterName = "TimeExecutingFilter")
public class TimeExecutingFilter implements Filter {
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException {...}

    public void init(FilterConfig config) throws ServletException {

    }

}
```

```
default public void init(FilterConfig filterConfig) throws ServletException {}
```

11. Prefer polymorphic solutions

```
private void invoke(HttpServletRequest req, HttpServletResponse resp) {
    String actionParameter = req.getParameter("action");
    switch (actionParameter) {
        case "empty":
            emptyOperation(req, resp);
            break;
        case "add":
            addAttribute(req, resp);
            break;
        case "update":
            updateAttribute(req, resp);
            break;
        case "invalidate":
            invalidateAttribute(req);
            break;
        case "remove":
            removeAttribute(req, resp);
            break;
    }
}
```

12. Where is request execution info?

```
public class LoggingFilter implements Filter {
    private static Logger log = Logger.getLogger(LoggingFilter.class.getName());

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        final HttpSession session = ((HttpServletRequest) request).getSession();

        final long creationTime = session.getCreationTime();
        final LocalDateTime creationLocalDateTime = Instant.ofEpochMilli(creationTime).atZone(ZoneId.systemDefault())
                .toLocalDateTime();

        final long lastAccessedTime = session.getLastAccessedTime();
        final LocalDateTime lastAccessedLocalDateTime = Instant.ofEpochMilli(lastAccessedTime)
                .atZone(ZoneId.systemDefault()).toLocalDateTime();


        log.info("Creation time: " + creationLocalDateTime);
        log.info("Last Accessed Time: " + lastAccessedLocalDateTime);
        log.info("Duration: " + Duration.between(creationLocalDateTime, lastAccessedLocalDateTime).toMillis()
                + " seconds");

        chain.doFilter(request, response);
    }
}
```

13. Use correct http methods

```
"<form name=\"actionForm\" action=\"session\" method=\"GET\">\n" +
 "<p>Action:\n" +
 "<select name=\"" + ACTION + "\">\n" +
 "<option value=\"\"></option>\n" +
 "<option value=\"" + ADD + "\">Add</option>\n" +
 "<option value=\"" + UPDATE + "\">Update</option>\n" +
 "<option value=\"" + INVALIDATE + "\">Invalidate</option>\n" +
 "<option value=\"" + REMOVE + "\">Remove</option>\n" +
 "</select>\n" +
 "</p>\n" +
 "\n" +
 "<p>Name:\n" +
 "<input type=\"text\" name=\"" + NAME + "\"/>\n" +
 "</p>\n" +
 "\n" +
 "<p>Value:\n" +
 "<input type=\"text\" name=\"" + VALUE + "\"/>\n" +
 "</p>\n" +
 "\n" +
 "<th><input type=\"submit\" value=\"submit\"/></th>\n" +
 "</form>";
```

14. Modification of input arguments is bad

```
public enum Operations {
    ADD {
        @Override
        public void execute(Map<String, String> attributes, String key, String value) {
            if (key.isBlank() || value.isBlank()) {
                throw new IllegalArgumentException("Key or value not entered.");
            }

            if (attributes.containsKey(key)) {
                throw new IllegalArgumentException("Attribute with current name already exists.");
            }

            attributes.put(key, value);
        }
    },
```

15. Only single instance of Servlet Context

```
private void setTimeAttributes(HttpServletRequest request){
    Date dateFinish = new Date();
    ServletContext servletContext = request.getServletContext();
    servletContext.setAttribute("FinishTime", String.valueOf(dateFinish.getTime()));
    String startTime = servletContext.getAttribute("StartTime").toString();
    long startTimeLong = Long.valueOf(startTime);
    long resultTime = dateFinish.getTime() - startTimeLong;
    servletContext.setAttribute("DurationTime", String.valueOf(resultTime));
}
```