Exercise Sheet 10

December 21: Chain of Responsibility and other Patterns

Exercise 1

a) The Servlet API!

1. Implement a class `FilteredServlet` extending `HttpServlet`. It should do nothing but respond *only* to GET request and send back a response "Filtered Servlet invoked"!
2. Configure the servlet for the Servlet container (here: Tomcat) by including the following XML-snippet in your web.xml

```
<servlet>
<description></description>
<display-name>filteredServlet</display-name>
<servlet-name>filteredServlet</servlet-name>
<servlet-class>crp.FilteredServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>filteredServlet</servlet-name>
<url-pattern>/filteredServlet</url-pattern>
</servlet-mapping>
```

3. Deploy[1] and run the `FilteredServlet` servlet!
4. What pattern is at work? (Hint: it is *not* Chain of Responsibility!)

b) Now extend the servlet by a filter:

1. implement the following code:

```
public class AuditFilter implements Filter {
    private ServletContext app = null;

    public void init(FilterConfig config) {
        app = config.getServletContext();
    }

    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws java.io.IOException,
            javax.servlet.ServletException {

        // log request
        app.log(((HttpServletRequest) request).getServletPath());
        chain.doFilter(request, response);
    }

    public void destroy() {
    }
}
```

2. Configure the filter for the Servlet container (here: Tomcat) by including the following XML-snippet in your web.xml

[1]Configuration of a Servlet container will be explained in the exercise!

```
<filter>
<filter-name>auditFilter</filter-name>
<filter-class>crp.AuditFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>auditFilter</filter-name>
<url-pattern>/filteredServlet</url-pattern>
</filter-mapping>
```

3. (Re-) deploy[2] and run the `FilteredServlet` servlet again! Does it log?
4. Draw a sequence diagram!
5. What pattern is at work?

Exercise 2

The Servlet API again!

1. Implement a convenience class `StringWrapper` extending the class `Filter` HttpServletResponseWrapper. It should contain the following code:

```
public class StringWrapper extends HttpServletResponseWrapper {
    StringWriter writer = new StringWriter();

    public StringWrapper(HttpServletResponse response) {
        super(response);
    }

    public PrintWriter getWriter() {
        return new PrintWriter(writer);
    }

    public String toString() {
        return writer.toString();
    }
}
```

2. Implement a class `SearchAndReplaceFilter` implementing the `Filter` interface. The class should contain a `doFilter` method that will use the previously defined `StringWrapper` class in order to

   - intercept a `ServletRequest`,
   - search the `FilterConfig` for the Strings “search” and “replace”, and
   - if the strings are found the search and replace values are used to replace the search String with the replace string!

   The `StringWrapper` convenience class is used as follows

```
        StringWrapper wrapper = new StringWrapper(
                (HttpServletResponse) response);
        chain.doFilter(request, wrapper);
        String responseString = wrapper.toString();
        ...
```

[2] Configuration of a Servlet container will be explained in the exercise!

3. In order to test the filter, deploy a JSP (`/index.jsp`) containing the line “Take the Blue Pill?”!
4. Configure the filter for the Servlet container (here: Tomcat) by including the following XML-snippet in your web.xml[3]

```
<filter>
    <filter-name>searchAndReplaceFilter</filter-name>
    <filter-class>crp.SearchAndReplaceFilter</filter-class>
    <init-param>
       <param-name>search</param-name>
       <param-value>Blue</param-value>
    </init-param>
    <init-param>
       <param-name>replace</param-name>
       <param-value>Red</param-value>
    </init-param>
 </filter>
 <filter-mapping>
    <filter-name>searchAndReplaceFilter</filter-name>
    <url-pattern>/index.jsp</url-pattern>
 </filter-mapping>
```

5. Deploy and run everything!
6. Test by calling `http://localhost:8080/tomcat/index.jsp`
7. What pattern is at work?

Exercise 3

Have a look at the JHotDraw sources! Can you find a usage of the Chain of Responsibility pattern? Hint: Search the code base for “Chain”...

Hints

- Consult the literature!
- You can work in pairs, if you want!
- If you want to learn a Java API, look into the java docs!
- Always use the same familiar IDE (suggestion Eclipse)!

[3]any occurrence of “Blue” will be replaced by “Red”