Advanced Java

Agenda

- ServletContext
- Applet vs Servlets
- CGI vs Servlets
- Filters
- Listeners

ServletContext

- Refer yesterday's notes pdf.

Movie Review System using Servlets

- AddUserServlet
  - HTML control type="date" always sends date in form "yyyy-MM-dd".
  - If using java.util.Date

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String birthDate = req.getParameter("birth");
Date birth = sdf.parse(birthDate);
```

  - If using java.sql.Date

```
String birthDate = req.getParameter("birth");
Date birth = Date.valueOf(birthDate);
```

- LoginServlet -- same as VotingSystem
- AddReviewServlet -- same as VotingSystem's CandidateEditServlet GET & POST.
- EditReviewServlet -- same as VotingSystem's CandidateEditServlet GET & POST -- Keep "id" control invisible instead of readonly. Hidden controls are not visible in browser, but visible in View Source.

```
out.printf("<input type='hidden' value='%s'><br/>", r.getId());
```

- DeleteReviewServlet -- same as VotingSystem's CandidateDeleteServlet.
- LogoutServlet -- same as VotingSystem's LogoutServlet.
- ReviewsServlet (url-pattern /reviews) -- similar to VotingSystem's ResultServlet.

```
HttpSession session = req.getSession();
User curUser = (User)session.setAttribute("curUser");

out.println("<a href='reviews?type=all'>All Reviews</a> | ");
out.println("<a href='reviews?type=my'>My Reviews</a> | ");
out.println("<a href='reviews?type=shared'>Shared Reviews</a>");

String type = req.getParameter("type");
List<Review> list = new ArrayList<>();
if(type==null || type.equals("all"))
    list = reviewDao.findAll();
else if(type.equals("my"))
    list = reviewDao.findByUserId(curUser.getId());
else if(type.equals("shared"))
    list = reviewDao.getSharedReviewsWithUser(curUser.getId());

out.println("<table>");
// print table header
out.println("<tbody>");
```

```
for(Review r:list) {
    out.println("<tr>");
    out.printf("<td>%d</td>", r.getId());
    //out.printf("<td>%d</td>", r.getMovieId());
    Movie m = movieDao.findById(r.getMovieId());
    out.printf("<td>%s</td>", m.getTitle());
    out.printf("<td>%d</td>", r.getUserId());
    out.printf("<td>%d</td>", r.getRating());
    out.printf("<td>%s</td>", r.getReview());
    out.printf("<td>%s</td>", r.getModified());
    out.println("<td>");
    if(r.getUserId() == curUser.getId())
        out.printf("<a href='editreview?id=%s'>Edit</a><a href='delreview?id=%s'>Edit</a>...", r.getId(), r.getId());
    out.println("<td>");
    out.println("</tr>");
}
out.println("</tbody>");
out.println("</table>");
```

- ShareReviewServlet -- Refer hint from day03.pdf (at the end).

Applet vs Servlets

- Applet is a Java class that is downloaded into client machine (when request is made) and executed in client browser JRE/JVM plugin.
- Applet Life Cycle: init(), start(), paint(), stop(), destroy() called by JRE in browser (a.k.a. Applet container)
- Servlet is a Java class that is executed "in web-server" when request is received from client and produces response that is sent back to the client.
- Servlet Life Cycle: init(), service() and destroy() called by JRE in web-server (a.k.a. Web container).

CGI vs Servlets

- CGI stands for Common Gateway Interface -- Standard of how a program can communicate with web.
- Program can be written in any language (that supports CGI) e.g. C, Fortran. These programs executed in OS (native applications) i.e. platform dependent.

- For each request from the client, a new "process" is created, that execute the request handling program and produces response that is sent back to the client.
- Less efficient (Processes are heavy-weight).
- Servlet is standard of implementing Java program that can handle HTTP requests.
- Servlet is written in Java and runs on Web container (JVM) i.e. platform independent.
- For each request from the client a new thread is created, that execute the service() method and produces response that is sent back to the client.
- More efficient (Threads are light-weight).

Aspect Oriented Programming

- Aspect Oriented Programming enables us to implement additional functionalities (a.k.a. cross-cutting concerns) without modifying core business logic.
- Cross-cutting concerns: Security, Logging, Monitoring, Performance measurement, Transaction management, ...
- Typically AOP is done as pre-processing or post-processing or both.
- In Core Java applications, AOP is done using Aspect/J framework or Java proxies.
- In Web Java applications, AOP is done using Filters.

Filters

- Filters is way of implementing AOP in Java EE applications. Filters are used to perform pre-processing, post-processing or both for each request.
- Multiple filters can be executed in a chain/stack before/after handling request.
- javax.servlet.Filter interface is used to implement Filters.
    - void init(FilterConfig filterConfig);
    - void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain);
    - void destroy();
- https://docs.oracle.com/javaee/7/api/javax/servlet/Filter.html
- Can be configured with @WebFilter or in web.xml (similar to servlets).

Listeners

- Listeners are used to handle web application level events. Examples of events:
    - Application start
    - Application end

    - Session start
    - Session end
    - Request start
    - Request end
    - Attribute added (in session/application/request)
    - Attribute removed (from session/application/request)
    - Attribute changed (in session/application/request)
- There are many listener interfaces. Refer docs.
    - ServletContextListener -- To handle application initialized and destroy events.
        - void contextInitialized(ServletContextEvent sce);
            - Called by web container when application is started/deployed i.e. servlet context is created.
            - Example: load and register JDBC driver, initialize a connection pool, ...
        - void contextDestroyed(ServletContextEvent sce);
            - Called by web container when application is stopped i.e. when web server shutdown.
            - Example: release a connection pool, ...
        - Implement this listener to perform one time initialization and destruction for the whole application.
    - HttpSessionListener -- To handle session initialized and destroy events.
        - sessionCreated() method is called when req.getSession() is called first time for any client. You may add any session attribute in it immediately after creating session.
        - sessionDestroyed() method is called when session is invalidated or time-out.
    - ServletRequestListener -- To handle request initialized and destroy events.
    - ServletContextAttributeListener
    - HttpSessionAttributeListener
    - ServletRequestAttributeListener
- Listener class must implement one or more listener interface.
- Can be configured with @WebListener OR in web.xml.

```
<listener>
    <listener-class>pkg.MyListener</listener-class>
</listener>
```