Enterprise Java

Agenda

- Introduction
- Java EE
- HTTP Protocol
- Java Web Server (Tomcat)
- WebApp directory structure
- Java Servlet
- Servlet Hierarchy
- web.xml

Web based Java Programming

- Syllabus
    - Foundations
        - JDBC
        - Servlets
        - JSP
    - Core Technologies
        - Hibernate & JPA
        - Spring: Core, JPA, MVC
        - Spring Boot: REST, Spring Data
    - Jdbc --> Hibernate/JPA --> Spring + JPA --> Spring Data
    - OOP --> Spring Core
    - Servlets + Jsp + ... --> Spring MVC + Spring REST
- Prerequisite
    - Discipline, Patience, Hardwork
    - Core Java: Class & Object, Collections (ArrayList, HashMap), IO (PrintWriter), Exceptions, JDBC (PreparedStatement, Transactions, DAO), Reflection & Annotations
    - RDBMS: SQL (CRUD, Joins)
    - HTML
- Schedules
    - 8:00 am to 7:30 pm
- Evaluations
    - Theory: 40 marks (CCEE)
    - Lab: 40 marks
    - Internal: 20 marks (Attendance + Assignments, Quiz)

Java EE

- Java SE -- Java Standard Edition
- Java ME -- Java Micro Edition
- Java EE -- Java Enterprise Edition
    - Enterprise: Business/Organization.

  - Java EE -- Designed to develop applications for enterprises.
  - n-tier applications
    - Database
    - Data access
    - Business logic(s)
    - Presentation (Frontend)
- Java EE -- For developing web applications and web services
- Java EE is a set of specifications (given by Oracle/Sun/Jakarta in form of interfaces). It includes jdbc, servlet, jsp, jsf, ejb, jpa, etc.

HTTP protocol

- HTTP -- Hyper Text Transfer Protocol.
- Connection-less protocol.
- State-less protocol.
- Request-response model.
- Web server is program that enable loading multiple web applications in it.
- Web application is set of web pages (static or dynamic), which are served over HTTP protocol.
- Client makes request by entering URL, click submit, or click hyper-link.
- URL: http://server:port/appln/resource
  - http: protocol/scheme
  - server: machine name or IP address
  - port: default 80
  - URI: /appln/resource
- Request Headers
  - Server/Host: server name/ip + port
  - User-Agent: Browser type/version
  - URI
  - HTTP version: 1.0 or 1.1
  - Content-Type: Type of data in Request body -- application/json, text/...
  - Length: Number of bytes in Request body
  - Method:
    - GET: Get the resource from the server.
      - Request sent when URL entered in address bar, hyper-link is clicked, html form with method=get is submitted.
      - The data (in html form) is sent via URL.
      - Not secured (because data visible in URL).
      - Faster.
    - POST: Post data to the server.
      - Request sent when html form with method=post is submitted.
      - The data (in html form) is sent via request body.
      - More secure
    - HEAD: Send response headers only.
      - No response data is sent to the client.
    - PUT: Put/upload a resource on server.
    - DELETE: Delete a resource from the server.
    - TRACE: Tracing/Information logging

        - OPTIONS: To know which request methods are supported for the resource.
    - Cookies, ...
- Request Body: JSON, Form-Data, or Other.
- Response Headers
    - Status: Code/Text
        - 1xx: Information
        - 2xx: Success e.g. 200 (Ok), 201 (Created), ...
        - 3xx: Redirection e.g. 302
        - 4xx: Client errors e.g. 404 (Not found), 403 (Forbidden), ...
        - 5xx: Server errors e.g. 500 (Internal server error), ...
    - Content-Type: Type of data in Response body
        - text/... : plain, html, xml
        - image/...: png, jpeg, gif, svg
        - audio/...: mp3, wav
        - video/...: mpeg
        - application/...: json, ...
    - Length: Number of bytes in Response Body
    - Cookies, ...
    - Server Info: IP, port, server type, ...
- Quick Revision: https://youtu.be/N_cgBn2KIto

Java Web Server

- There are many web servers from different vendors. But all implement the same Java EE specifications.
- Java web server = Servlet container + Extra services.
    - e.g. Tomcat, Lotus, ...
- Java application server = Servlet container + EJB container + Extra services.
    - e.g. JBoss, WebSphere, WebLogic, ...
- Extra services includes security (HTTPS), JNDI, Connection pool, ...

Apache Tomcat

- Apache tomcat is Java web server (Web container & Extra services).
- Apache tomcat 9.x implements Java EE 8 specs.
    - Servlet 4.0 specs
    - JSP 2.3 specs
    - JSF 2.3 specs
    - Tomcat directory structure
        - bin
        - conf
        - lib
        - webapps
        - work
        - logs
        - temp
- Test tomcat server (without Eclipse STS):
    - step 1: In terminal, go to tomcat/bin directory.

- step 2: terminal> ./startup.sh
- step 3: Open Browser and http://localhost:8080/
- step 4: terminal> ./shutdown.sh

Java EE application structure

- Java web application must have a fixed structure.

```
appln/
    |- *.html, *.jsp
    |- *.js, *.css
    |- *.png, *.jpg
    |- WEB-INF/
        |- web.xml
        |- classes/
            |- *.class
        |- lib/
            |- *.jar
```

- The application is typically compressed (zipped) as appln.war file and copied into tomcat/webapps directory.(
- Then application is accessible from client browser
  - http://server:port/appln/page

Java Servlet

- Servlet is java class that is executed on server side, when request is done by client and produces result, which is sent to the client as response.
- Servlet specs include multiple interfaces like Servlet, ServletRequest, ServletResponse, Filter, RequestDispatcher, ...
- javax.servlet.Servlet interface
  - void init(ServletConfig config) throws ServletException;
  - void service(ServletRequest req, ServletResponse resp) throws IOException, ServletException;
  - void destroy();
- GenericServlet is abstract class that represents protocol-independent servlet.
- HttpServlet represent http based servlet class and user defined servlet classes are inherited from it.
  - Overrides service() method.
  - Provide doGet(), doPost(), doPut(), doDelete(), doHead(), doTrace(), doOptions()
  - Docs: https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html
- Example

```
@WebServlet("/hi")
public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
throws IOException, ServletException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
```

```
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Hello,DAC</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h3>Welcome to Java EE application!</h3>");
            Date now = new Date();
            out.println("Server DateTime: " + now.toString());
            out.println("</body>");
            out.println("</html>");
        }
    }
```

- Hello Servlet application steps
  - step 0: In Settings --> Preferences --> Add Apache Tomcat 9 in Server Runtimes. (One per workspace)
  - step 1: Create "Dynamic Web Project".
  - step 2: In src, create HelloServlet class in some package.
  - step 3: Right click on project, run on Server -- Select Tomcat.
  - step 4: In Browser, http://localhost:8080/projname/hello

Servlet Hierarchy

- javax.servlet.Servlet interface
  - void init(ServletConfig config) throws ServletException;
  - void service(ServletRequest req, ServletResponse resp) throws IOException, ServletException;
  - void destroy();
- GenericServlet is abstract class that represents protocol-independent servlet.
- HttpServlet represent http based servlet class and user defined servlet classes are inherited from it.
  - Overrides service() method.
  - Provide doGet(), doPost(), doPut(), doDelete(), doHead(), doTrace(), doOptions()
  - Docs: https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html

ServletConfig

- Each servlet is associated with a config object -- ServletConfig.
- It stores information about servlet like name, init parameters, url-patterns, load-on-startup, etc.
- This can be accessed in the servlet class in init() method (as argument) or other methods using `ServletConfig cfg = this.getServletConfig();`.
- Note that all servlet classes are indirectly inherited from ServletConfig, so ServletConfig methods are directly available on servlet object (this).

web.xml

- Which of the following deployment descriptor of a Java web application?
  - A. /WEB-INF/Web.xml
  - B. /WEB_INF/web.xml
  - C. /WEB-INF/web.xml
  - D. web.xml

- web.xml is deployment descriptor of web applications. It contains deployment information like servlet configs, jsp configs, session timeout, application security, etc.

HttpServletRequest interface

- https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletRequest.html
- HttpServletRequest interface inherited from ServletRequest interface.
- It is created by webserver for each request and represent http req body & header.
- Request Parameters
  - Data from submitted HTML form (in previous page) in request body (POST) or URL (GET).
  - String paramValue = req.getParameter("param-name");
    - Used with textbox, radiobutton, drop-down, ...
  - String[] paramValues = req.getParameterValues("param-name");
    - Used with checkboxes, listbox, ...
- Request Headers
  - String headerValue = req.getHeader("header-name");
    - e.g. String value = req.getHeader("Content-Type");
  - String[] headerValues = req.getHeaderValues("header-name")
- Request upload
  - InputStream in = req.openInputStream();

HttpServletResponse

- https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletResponse.html
- HttpServletResponse interface inherited from ServletResponse interface.
- It is created by webserver for each request and represent http response body & header.
- Response content type
  - resp.setContentType("text/html"); --> Sets response header Content-Type
- Response send error -- return HTTP status code with message
  - resp.sendError(403);
  - resp.sendError(HttpServletResponse.SC_FORBIDDEN, "Fobidden resource");
- Response download/image
  - OutputStream out = resp.openOutputStream();
  - Need to setup content-type for download (application/octet-stream) or image (image/png) or audio.