Advanced Java

Agenda

- Java EE Introduction
- HTTP protocol
- Java Web Servers
- Java Servlets

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 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 0: In environment variables set JAVA_HOME (a new env variable).
        - JAVA_HOME=C:\Program Files\Eclipse Adoptium\jdk-11.0.15.10-hotspot
    - step 1: Open command prompt. Go to tomcat/bin directory (using cd command).
    - step 2: cmd> startup.bat
    - step 3: Open Browser and http://localhost:8080/
    - step 4: In tomcat/bin directory run shutdown.bat (from Windows explorer).

Java Servlet

- Servlet is a Java class that is executed in Java web server, when request is done by the client, and produces response that is sent to the client.
- Servlet specs include multiple interfaces like Servlet, ServletRequest, ServletResponse, Filter, RequestDispatcher, ...
- HelloServlet class

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello DMC</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello, Servlet!</h1>");
        Date d = new Date();
        out.println("Current time: " + d.toString());
        out.println("</body>");
        out.println("</html>");
    }
}
```

- 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

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.
- Servlet config in web.xml

```
<servlet>
    <servlet-name>DMC</servlet-name>
    <servlet-class>com.sunbeam.DmcServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DMC</servlet-name>
    <url-pattern>/mobile</url-pattern>
</servlet-mapping>
```

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

Servlet Life Cycle

- Refer slides

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).

Init parameters

- ServletConfig may have some configurable values like JDBC url, username, password, etc.
- They can be attached to config using init-params using annotation or in web.xml.

```xml
<servlet>
    <servlet-name>DMC</servlet-name>
    <servlet-class>com.sunbeam.DmcServlet</servlet-class>
    <init-param>
        <param-name>color</param-name>
        <param-value>pink</param-value>
    </init-param>
    <init-param>
        <param-name>greeting</param-name>
        <param-value>Good Afternoon</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>DMC</servlet-name>
    <url-pattern>/mobile</url-pattern>
</servlet-mapping>
```

- These init params can be accessed in servlet class using getInitParameter() method.

```java
ServletConfig cfg = this.getServletConfig();
String color = cfg.getInitParameter("color"); // returns "green"
```

```java
String message = this.getInitParameter("greeting"); // returns "hi"
```

Assignments

1. Implement JDBC DAOs for Movie Review System. Follow the standard implementation practices.
   - Create DbUtil class to create the connection.
   - Create common Dao class.
   - Create Dao interfaces as given below and then do the implementations.

```
public interface MovieDao extends AutoCloseable {
    public List<Movie> findAll() throws Exception;
    public Movie findById(int id) throws Exception;
}
```

```
public interface ReviewDao extends AutoCloseable {
    public int save(Review r) throws Exception;
    public int update(Review r) throws Exception;
    public List<Review> findAll() throws Exception;
    public List<Review> findByUserId(int userId) throws Exception;
    public List<Review> getSharedWithUser(int userId) throws Exception;
    public Review findById(int id) throws Exception;
    public int deleteById(int reviewId) throws Exception;
    public int shareReview(int reviewId, int userId) throws Exception;
}
```

```
public class UserDao extends AutoCloseable {
    public int save(User u) throws Exception;
    public int update(User u) throws Exception;
    public int updatePassword(int userId, String newPassword) throws Exception;
    public User findByEmail(String email) throws Exception;
    public List<User> findAll() throws Exception;
}
```