Core Java

Agenda

- Java Web Servers
- Apache Tomcat
- Servlet Hierarchy
- Servlet Life Cycle
- Init parameters
- Load on Startup
- Deployment descriptor (web.xml)
- Request parameters
- Inter-servlet navigation
- State management introduction

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, â€¦
- Java application server = Servlet container + EJB container + Extra services.
  - e.g. JBoss, WebSphere, WebLogic, â€¦
- Extra services includes security (HTTPS), JNDI, Connection pool, â€¦

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:
    - step 0: In environment variables set JAVA_HOME (a new env variable).
        - export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
    - step 1: Open command prompt. Go to tomcat/bin directory (using cd command).
    - step 2: terminal> ./startup.sh
    - step 3: Open Browser and http://localhost:8080/
    - step 4: In tomcat/bin directory run shutdown.bat (from Windows explorer).

Java Servlet Hierarchy

- 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.
- If HelloServlet is user-defined Servlet class,

```
Servlet, ServletConfig, Serializable
    |- GenericServlet
        |- HttpServlet
            |- HelloServlet
```

- 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

- The Java Servlet life cycle defines how a servlet is loaded, instantiated, initialized, serves requests, and is finally destroyed by the servlet container (e.g., Tomcat, Jetty). The servlet life cycle is managed by the web container (servlet container) and defined by Servlet interface methods. These methods must be defined in Servlet class (directly or indirectly).

- **Loading and Instantiation**

  - When a servlet is first requested, the servlet container loads the servlet class into memory and creates an instance of the servlet.
  - Only one instance of the servlet is created by the container, no matter how many requests it processes.
  - After the servlet is instantiated, the container calls the `init(ServletConfig config)` method to initialize it.
  - This method is called only once in the servlet's lifetime and is where you put any initialization code, like setting up resources (database connections, reading configuration, etc.).
  - If this method throws a `ServletException`, the servlet won't be put into service.

- **Request Handling (`service` method)**

  - After initialization, the servlet is ready to process client requests.
  - For each incoming request, the container calls the `service(HttpServletRequest req, HttpServletResponse res)` method.
  - Typically, user-defined servlet classes don't override this method, so method of HttpServlet (super-class) is called.
  - This method determines the type of request (GET, POST, etc.) and dispatches the request to the appropriate handler method (`doGet`, `doPost`, etc.). These methods are usually overridden in user-defined servlet.

- **Destruction (`destroy` method)**

  - When the servlet is no longer needed (typically when the server is shutting down or the servlet is being removed from service), the container calls the `destroy()` method.
  - This is the place for cleanup operations, like closing database connections, freeing resources, etc.
  - Once the `destroy()` method is called, the servlet is marked for garbage collection.

- Note: Servlets are multi-threaded, meaning multiple requests can be processed concurrently using the same servlet instance.

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.

```
@WebServlet(value="/mobile",
    initParams = {
        @WebInitParam(name="color", value="green"),
        @WebInitParam(name="greeting", value="Hi")
    },
    name = "DMC")
public class DmcServlet extends HttpServlet {
    // ...
}
```

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

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

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

Load On Startup

- By default servlet is loaded and initialized on first request. If init() includes heavy processing, the first request will execute slower.
- Alternatively, servlets can be loaded while starting the web server. This can be done by marking servlet as load-on-startup using web.xml or annotation.

```
@WebServlet(value="/mobile",
    loadOnStartup = 1,
    name = "DMC")
public class DmcServlet extends HttpServlet {
    // ...
}
```

- The number after "loadOnStartup" indicate the sequence of loading the servlets if multiple servlets are marked as load-on-startup.
- If multiple servlets load-on-startup number is same, web container arbitrarily choose the sequence.
- If number after "loadOnStartup" is negative, the servlet is not loaded at startup. It will be loaded on first request.

Deployment Descriptor - 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>
    <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>
```

HttpServletRequest

- 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 body (text)
  - PrintWriter out = resp.getWriter();
  - creates a PrintWriter that helps writing in response body. This dynamically generated response will be sent to the client.
- 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.

Assignments

1. Create page newuser.html and RegistrationServlet to Sign up new user.