

# Advanced Java

---

## Web based Java Programming

### 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/appIn/resource
  - http: protocol/scheme
  - server: machine name or IP address
  - port: default 80
  - URI: /appIn/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\\_cgBn2Klto](https://youtu.be/N_cgBn2Klto)

## 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 10.1 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