J2EE Basics

Version Java EE 8 (J2EE 1.8) maintained under Oracle / Jakarta EE 8 (maintained by eclipse foundation)

1. What is J2EE ?(Java Enterprise Edition)

Consists of specifications only .

Which specs ? (Rules or contract)
Specifications of primary services required for any enterprise application.

What is enterprise application ?

An enterprise application (EA) is a large software system platform designed to operate in a corporate environment .

It includes online shopping and payment processing, interactive product catalogs, computerized billing systems, security, content management, IT service management, business intelligence, human resource management, manufacturing, process automation, enterprise resource planning ....

These specifications include ---

Servlet API,JSP(Java server page) API,Security,Connection pooling ,EJB (Enterprise Java Bean), JNDI(Naming service -- Java naming & directory i/f),JPA(java persistence API),JMS(java messaging service),Java Mail, Java Server Faces , Java Transaction API, Webservices support(SOAP/REST) etc...

Vendor of J2EE specs -- Oracle / Sun / Eclipse

Implementation -- left to vendors (J2EE server vendors)
J2EE compliant web server --- Apache -- Tomcat (web server)
Services implemented --- servlet API,JSP API,Security,Connection pooling,JNDI(naming service)

J2EE complaint application server --- web container + EJB (enterprise java bean) container
+ ALL J2EE services implementation

J2EE server Vendors & Products
Apache -- tomcat(web server) / Tomee (app server)
Oracle / Sun --- reference implementation --- Glassfish
Red Hat -- JBoss (wild fly)
Oracle / BEA -- weblogic
IBM -- Websphere

2. WHY J2EE
1. Can support different types of clnts --- thin client(web clnt)
thick clnt --- application clnt(eg : TCP client)

smart clnts -- mobile clnts

2. J2EE server independence --- Create & deploy server side appln --on ANY J2ee compliant server --- guaranteed to produce SAME results w/o touching or re-deploying on ANY other J2EE server

3. Ready made implementation of primary services(eg --- security, conn,pooling,email....)--- so that J2EE developer DOESn't have to worry about primary services ---rather can concentrate on actual business logic.

3. Layers involved in HTTP request-response flow (refer to day1-data\day1_help\diags\request-response-flow.png)

Web browser sends the request (URL)
 eg : http://www.abc.com:8080/day1.1
/day1.1 --- root / context path /web app name

Host --Web server--Web Container(server side JVM)--Web application---HTML/JSP/Servlet....

4. What is dyn web application --- server side appln --deployed on web server --- meant for servicing typically web clnts(thin) -- using application layer protocol HTTP /HTTPS
(ref : diag request-resp flow)

Read --HTTP basics including request & response structure from day1-data\day1_help\j2ee_prerequisites\HTTP Basics

5. Objective ?: Creating & deploying dyn web appln on Tomcat -- For HTML content

6. IDE automatically creates J2EE compliant web application folder structure .
Its details -- Refer to diag (J2EE compliant web app folder structure)

7. What is Web container --- (WC) & its jobs
1. Server side JVM residing within web server.
Its run-time environment for dynamic web components(Servlet & JSP,Filter) .
Jobs ---
1. Creating Http Request & Http response objects
2. Controlling life-cycle of dyn web comps (manages life cycle of servlet,JSP,Filters)
3. Giving ready-made support for services --- Naming,security,Conn pooling .
4. Handling concurrent request from multiple clients .
5. Managing session tracking...

8. What is web.xml --- Deployment descriptor one per web appln
created by -- developer
who reads it -- WC
when --- @ deployment
what --- deployment instructions --- welcome page, servlet deployment tags, sess config, sec config......

---
9. Why servlets? --- To add dynamic nature to the web application

What is a servlet ?
-- Java class (with NO main method) -- represents dynamic web component - whose life cycle will be managed by WC(web container : server side JVM)
no main method
life cycle methods --- init,service,destroy

Job list
1. Request processing
2. B.L
3. Dynamic response generation
4. Data access logic(DAO class --managing DAO layer)
5. Page navigation

Servlet API details --refer to diag servlet-api.png

Objective 1: Test basic servlet life cycle -- init , service ,destroy
10. Creating & deploying Hello Servlet.

Deployment of the servlet
1. Via annotation
eg : @WebServlet(value="/validate")
public class LoginServlet extends H.S {....}
Map :
key -- /validate
value -- F.Q servlet cls name
URL : http://host:port/day1.1/validate?....

2. Using XML tags
How to deploy a servlet w/o annotations --- via XML tags
web.xml

```
<servlet>
 <servlet-name>abc</servlet-name>
<servlet-class>pages.SecondServlet</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>abc</servlet-name>
 <url-pattern>/test2</url-pattern>
</servlet-mapping>
```

WC : map
key : /test2
value : pages.SecondServlet

eg URL --http://host:port/day1_web/hello

At the time of web app deployment ---WC tries to populate map of url patterns , from XML tags (from web.xml). Later ---it will check for @WebServlet annotation

Objective 2: Test basic servlet life cycle -- init , service ,destroy (deployed via xml)

How to read request params sent from the clnt ?

javax.servlet.ServletRequest i/f methods
1. public String getParameter(String paramName)

2. public String[] getParameterValues(String paramName)

Objective 3 : Accept different type of i/ps from user , in HTML form.Write a servlet to display request parameters.

Why HttpServlet classs is declared as abstract class BUT with 100 % concrete functionality ?

It is abstract because the implementations of key servicing methods have to be provided by (e.g. overridden by) servlet developer. Since it's abstract , it's instance can't be created.

A subclass of HttpServlet must override at least one method, usually one of these:

doGet, if the servlet supports HTTP GET requests
doPost, for HTTP POST requests
doPut, for HTTP PUT requests
doDelete, for HTTP DELETE requests
init and destroy, to manage resources that are held for the life of the servlet

If you extend the class without overriding any methods, you will get a useless servlet; i.e. it will give an error response for all requests.(HTTP 405 : Method not implemented) . So , if the class was not abstract, then any direct instance of HttpServlet would be useless.

So the reason for making the HttpServlet class abstract is to prevent a programming error.

As a servlet developer , you can choose to override the functionality of your requirement (eg : doPost) & ignore other methods.